CHAPTER 12

Recognition Strategies: Intrusion Detection and Prevention

INFORMATION IN THIS CHAPTER

- Intrusion detection
- Network intrusion detection pitfalls:
  - Fragmentation and IP validation
  - Application reassembly
  - Out-of-band problems
  - Centrality difficulties
  - Base-rate fallacy
- Modes of intrusion detection:
  - Network intrusion detection: signatures
  - Network intrusion detection: anomaly based
- Network behavior analyzers
- Wireless IDS
- Network intrusion prevention systems

INTRODUCTION

This chapter expands on the strategy for recognition. Due to the nature of the modern computer infrastructure, compromises are inevitable. Chapter 11 detailed various analysis techniques that a human analyst might use to detect evidence of a compromise on the network. This chapter discusses the natural next step in that process: reducing the workload on the human analyst by automating some of the functions. Intrusion detection and prevention systems aim to do just that.

The formal definition of an intrusion detection system (IDS) is a system that "monitor[s] the events occurring in a computer system or network and analyz[es] them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use

Introduction to Information Security. DOI: http://dx.doi.org/10.1016/B978-1-59749-969-9.00012-2

policies, or standard security practices" [1, p. 2-1]. This National Institute of Standards and Technology (NIST) definition has also been adopted by the Internet Engineering Task Force (IETF) [2]. Intrusion prevention is directly related, for it is "the process of performing intrusion detection and attempting to stop detected possible incidents" [1].

Following NIST, this chapter will use the combined term *intrusion detection and prevention system* (IDPS) for brevity. This does not replace the terms IDS or IPS, which are well established. However, most of the functions of an IDS are shared with an IPS, and devices termed IPSs can usually be configured to disable the prevention activity and function just as an IDS can [1]. So combining the terms is convenient and generally accurate. Any exceptions will be noted.

There are several types of IDPSs, based on what the system is monitoring. The most common is a network IDS, a machine that observes traffic at a choke point in the network and inspects it for intrusions across the organization [3, p. 660]. Network-based IDPSs are also characterized by the fact that they reassemble the packets of the network communication and attempt to interpret them as the target host would. The other three types are host based, wireless, and network behavior analysis (NBA) [1]. This chapter focuses on network-based IDPSs, however, the general principles are relevant to all IDPS technologies. Anti-virus software can be considered a type of host-based IDPS, but these products have expanded to attempt to prevent exploits and have taken on many characteristics of an IDPS [4]. Wireless IDPSs are intended to detect abuse of wireless networks themselves, either interference with the radio spectrum or rogue access points.

This chapter discusses several issues related to IDPSs. First, why instrumenting an IDS is important in addition to the network resistance and frustration strategies previously discussed; also, why it should be instrumented independently of these other devices. Next, the chapter covers some common historical pitfalls of intrusion detection devices, and their fixes, to describe the uses and limitations of IDPSs. Two common modes of detection for IDPSs are then discussed: signature-based and anomaly-based detection. Finally, the modifications necessary to go from an IDS to an IPS, and the ramifications, are discussed before concluding.

It is important to note that an IDPS is a conceptual device. Just as with firewalls and proxies discussed in Chapter 5, actual devices on the market may combine features of more than one category of device. For example, application-level proxies may contain IDPS features specific to an application, or a firewall may contain IDPS features for network traffic. The defender should account for this fact when designing and purchasing a sensor architecture. Sensor architecture and some other network analysis concepts relevant to IDPSs are discussed in Chapter 11.

WHY INTRUSION DETECTION

An IDPS is one of the more important devices in an organization's overall security strategy. There is too much data for any human analyst to inspect all of it for evidence of intrusions, and the IDPS helps alert humans to events to investigate, and prioritize human recognition efforts. An IDPS also serves an important auditing function. If the machines that form the technological backbone of the frustration and resistance strategies are misconfigured, the IDPS should be positioned to detect violations due to these errors. Furthermore, some attacks will exist for a period of time before there is any available patch or mitigation. An IDPS may be able to detect traffic indicative of the new attack, either as soon as a signature is made available or if the attack traffic is generally anomalous.

An IDPS has many actions available when responding to a security event. Generating an alert for human eyes is a common action, but it can also log the activity, record the raw network data that caused the alert, attempt to terminate a session, alter network or system access controls, or some combination thereof [5]. If managed well, the different rules stratify the actions into different categories related to the severity of attack, reliability of rule, criticality of target, timeliness of response required, and other organizational concerns. Once the notifications are stratified, the human operator can prioritize response and recovery actions, which are the topic of Chapter 15.

Returning to our walled-city metaphor from Chapter 5, we already have our static defenses—moats, walls, and gates—as well as the more active defenses, such as guards who inspect people coming into the city. An IDPS is similar to a sentry posted above the city gate and/or in a tower nearby. If the guard gets overrun by a suddenly unruly mob, the guard cannot call for help. The sentry provides a basic defense-in-depth function of recognizing that a problem has occurred and an alert needs to be issued. Also like an IDPS, the sentry may have some immediate corrective actions available, such as telling the gate operator to close it temporarily or in some castles there are grates over the entryway from which a sentry could pour boiling oil to deter invaders who breached the outer defenses. But like an IDPS, while these responses may be effective stop-gap measures, they are not sustainable methods of network management. The most important functions are to alert the authorities so that a recovery of security can begin, and to keep a record of how the incident occurred so a better system can be put in to place going forward.

The three major components of an IDPS and how they interact with the relevant organizational components are summarized in Figure 12.1. The IDPS sensor infrastructure observes activity that it normalizes or processes into events, which the analyzer ingests and inspects for events of interests. The manager processes deals with the events appropriately. In the tower sentry

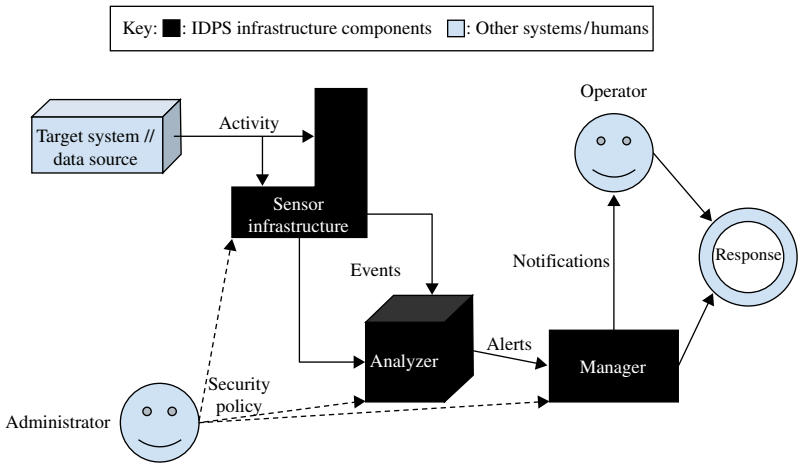

FIGURE 12.1
The basic components of an IDPS and how they interact with their environment.

metaphor, these components are all inside the one human sentry. In an IDPS, they can be part of one computer or distributed to specialized machines.

Figure 12.2 displays the internal components of an IDPS in more detail. The operator interacts with the system components through the graphical user interface (GUI); some systems use a command-line interface for administration in addition to or instead of a GUI. The alarms represent what responses to make. The knowledge base is the repository of rules and profiles for matching against traffic. Algorithms are used to reconstruct sessions and understand session and application data. The audit archives store past events of interest. System management is the glue that holds it all together, and the sensors are the basis for the system, receiving the data from the target systems.

NETWORK INTRUSION DETECTION PITFALLS

A network-based IDPS (NIDPS) has many strengths, but these strengths are also often its weaknesses. A NIDPS strength is that the system reassembles content and analyzes the data against the security policy in the format the target would process it. Another strength is that the data is processed passively, out of band of regular network traffic. A related strength is that a NIDPS can be centrally located on the network at a choke point to reduce hardware costs and configuration management. However, all of these benefits also introduce pitfalls, which will be discussed serially. Furthermore, there are some difficulties that any IDPS suffers from simply due to the fact that the Internet is noisy, and so differentiating security-related weird stuff from

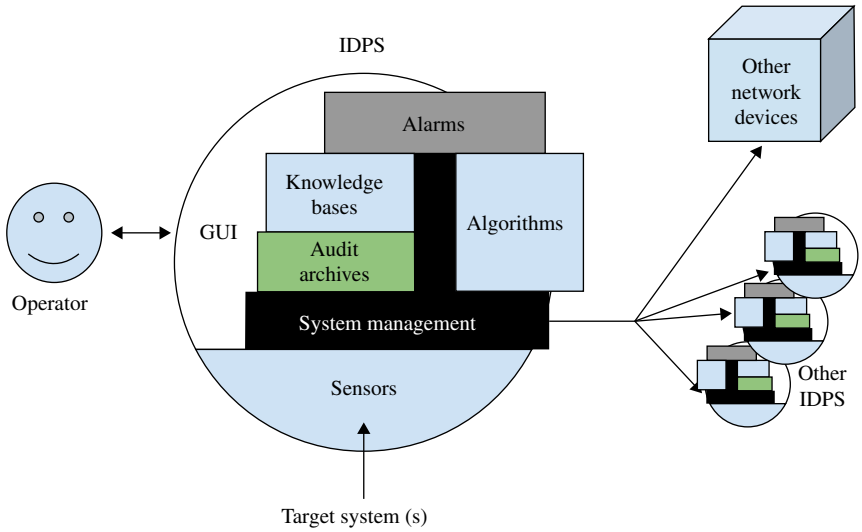

FIGURE 12.2

A more detailed view of how the internal components of an IDPS interact. If two shapes touch within the IDPS, then those two components interact directly.

general anomalies becomes exceptionally difficult. For some example benign anomalies, see Bellovin [6].

The following sections are not intended to devalue IDSs or to give the impression that an IDS is not part of a good security strategy. An IDS is essential to a complete recognition strategy. The following pitfalls are remediated and addressed to varying degrees in available IDPSs. Knowledge of how well a potential IDPS handles each issue is important when selecting a system for use. Despite advances in IDPS technology, the following pitfalls do still arise occasionally. It is important for defenders to keep this in mind: no one security strategy is infallible. Knowing the ways in which each is more likely to fail helps design overlapping security strategies that account for weaknesses in certain systems. For these reasons, we present the following common pitfalls in IDPSs.

Fragmentation and IP Validation

One of the pitfalls of reassembling sessions as the endpoints would view them is that endpoints tend to reassemble sessions differently. This is not just true of applications. This is true of the fundamental fabric of the Internet, the TCP/IP (Transport Control Protocol/Internet Protocol) protocol suite. To handle all possible problems that a packet might have while traversing the network, IP packets might be fragmented. Furthermore, if packets are delayed they might be resent by the sender. This leads to a combination of situations

in which the receiver may receive multiple copies of all or part of an IP packet. The RFCs (request for changes) that standardize TCP/IP behavior are silent on how the receiver should handle this possibly inconsistent data, and so implementations vary [7, p. 280].

Packet fragmentation for evading IDS systems was laid out in detail in a 1997 U.S. military report that was publicly released the following year [8]. Evasion is one of three general attacks described; the other two attacks against IDSs are insertion and denial of service (DoS). Insertion and evasion are both caused, in general, by inconsistencies in the way the TCP/IP stack is interpreted. DoS attacks against IDPSs are not limited to TCP/IP interpretation, and are treated throughout the subsections that follow. DoS attacks are possible through bugs and vulnerabilities, such as a TCP/IP parsing vulnerability like the teardrop attack [9], but when this chapter discusses DoS on IDPSs it refers to DoS specific to IDPSs. DoS attacks such as the teardrop attack are operating system vulnerabilities, and so such things are not IDPS specific, even though many IDPSs may run on operating systems that are affected.

The general problem sketched out by the packet fragmentation issues is that the strength of the IDPS—namely, that it analyzes the data against the security policy in the format the target would process—is thwarted when the attacker can force the IDPS to process a different packet stream than the target will. This can be due to insertion or evasion. For example, if the IDPS does not validate the IP header checksum, the attacker can send blatant attack packets that will initiate false IDPS alerts, because the target system would drop the packet and not actually be compromised. This insertion attack can be more subtle. IP packets have a time-to-live (TTL) value that each router decrements by 1 before forwarding. Routers will drop an IP packet when the TTL of the packet reaches 0. An attacker could send the human responder on lots of confusing, errant clean-up tasks if the TTL of packets are crafted to reach the IDPS, but be dropped before they reach the hosts [8]. And if an attacker knows your network well enough to manipulate TTLs like this, it is technically hard to prevent. The IDPS would have to know the number of router hops to each target host it is protecting—a management nightmare.

Another result of Ptacek and Newsham's report [8] was some research into how different operating systems handle different fragmentation possibilities [10]. Some NIDPS implementations now utilize these categories when they process sessions, and also include methods for the NIDPS to fingerprint which method the hosts it is protecting use so the NIDPS can use the appropriate defragmentation method [11]. This method improves processing accuracy, but management of this mapping is not trivial. Further, network address translation (NAT) and Dynamic Host Configuration Protocol (DHCP) will cause inconsistencies if the pool of computers sharing the IP space does not

share the same processing method. This subtle dependency highlights the importance of a holistic understanding of the network architecture—and keeping the architecture simple enough that it can be holistically understood.

Application Reassembly

NIDPSs perform reassembly of application data to keep states of transactions and appropriately process certain application-specific details. The precise applications reassembled by an IDPS implementation vary. Common applications like File Transfer Protocol (FTP), Secure Shell (SSH), and Hypertext Transfer Protocol (HTTP) are likely to be understood. Down the spectrum of slightly more specific applications, Gartner has published a business definition for "next-generation" IPSs that requires the system understand the content of files such as portable document format (PDF) and Microsoft Office [12]. The ability to process this large variety of applications when making decisions is a significant strength of IDPS devices, as most other centralized network defense devices are inline and cannot spend the time to reassemble application data. Proxies can, but they are usually application specific, and so lack the broader context that IDPSs usually can leverage.

The large and myriad application-parsing libraries required for this task introduce a lot of dependencies into IDPS operations, which can lead to some common pitfalls. First, IDPSs require frequent updates as applications change and bugs are fixed. If the IDPS was only purchased to fill a regulatory requirement and is ignored afterwards, it quickly becomes less and less effective as parsers fall out of date.

Even in the best case where the system is up to date, many of the variable processing decisions that were described earlier related to IP fragmentation are relevant to each application the IDPS needs to parse. The various web browsers and operating systems may parse HTTP differently, for example. This is less a problem in application handling, because as long as the IP packets are reassembled correctly, at least the IDPS has the correct data to inspect. But due diligence in testing rules might indicate that different rules are needed not just per application, but one for each common implementation of that application protocol. Bugs might be targeted in specific versions of application implementations, further ballooning the number of required rules. So far, NIDPSs themselves seem to be able to handle the large number of rules required, although rule management and tuning are arduous for system and security administrators.

Out-of-Band Problems

Although an IDPS is located on a central part of the network, it may not be in the direct line of network traffic. An inline configuration is recommended

only when IPS functionality will be utilized, otherwise an out-of-band configuration is recommended [1]. When an IDS is running out of band it has some benefits, but it also introduces some possible errors. If the IDS is out of band, then if the IDS is dropping packets no network services will suffer. This is a benefit, except that the security team then needs to configure the IDS to alert them when it is dropping packets so they can take that into account. A more difficult problem to detect is if the network configuration that delivers packets to the IDS develops errors, either accidental or forced by the attacker, that result in the IDS not receiving all the traffic in the first place. There is a similar problem with other resource exhaustion issues, whether due to attacks or simply to a large network load, at the transport and application layer.

Inline architectures have to make harder decisions about what to do when the IPS resources are exhausted. Despite the best planning, resource exhaustion will happen occasionally; if nothing else, adversaries attempt to cause it with DoS attacks. Whether the IPS chooses to make network performance suffer and drop packets, or it chooses to make its analysis suffer and not inspect every packet, is an important decision. The administrator should make the risk analysis for this decision clear. This is an example of a failure control situation [2]. In general, a fail-secure approach is recommended; in this example the IPS would fail-secure by dropping packets. This approach fails securely because no attack can penetrate the network because of the failure, unlike the other option.

In either case, the resource exhaustion failure still causes damage. The IDPS cannot log packets it never reads, and if its disk space or processor is exhausted, then it cannot continue to perform its recognition functions properly. Therefore, appropriately resourcing the IDPS is important. On large networks, this will likely require specialized devices.

Centrality Problems

Since the NIDPS is centrally located, it has a convenient view of a large number of hosts. However, this central location combined with the passive strategy of IDPS also means that data can be hidden from view. Primarily this is due to encryption, whether it is IPSec [13,14], Transport Layer Security (TLS) [15], or application-level encryption like pretty good privacy (PGP) [16]. Encryption is an encouraged, and truthfully necessary, resistance strategy (Chapter 8). However, if application data is encrypted, then the IDPS cannot inspect it for attacks. This leads to a fundamental tension—attackers will also encrypt their attacks with valid, open encryption protocols to avoid detection on the network. One strategy to continue to detect these attacks is host-based detection. The host will decrypt the data, and can perform the IDPS function there. However, this defeats the centralized nature of NIDPS, and also thwarts the broad correlation abilities that only a centralized sensor can provide. And

as groups like the Electronic Frontier Foundation encourage citizens with programs like "HTTPS Everywhere" [17], in addition to the push from the security community, the prevalence of encryption will only increase.

On a controlled network it is possible to proxy all outgoing connections, and thereby decrypt everything, send it to the IDPS, and then encrypt it again before it is sent along to its destination. It is recommended to implement each of these functions (encryption proxy, IDPS) on a separate machine, as each are resource-intensive and have different optimization requirements [18].

Base-rate Fallacy

The final problem that IDPSs encounter is that they are trying to find inherently rare events. False positives—that is, the IDPS alerts on benign traffic—are impossible to avoid. If there are too many false positives, the analyst is not able to find the real intrusions in the alert traffic. All the alerts are equally alerts; there is no way for the analyst to know without further investigation which are false positives and which are true positives. Successful intrusions are rare compared to the scope of how much network traffic passes a sensor. Intrusions may happen every day, but if the intrusions become common it does not take an IDPS to notice. Network performance just plummets as SQL Slammer,[1] for example, repurposes your network to scan and send spam. But that is not the sort of intrusion we need an IDPS to find. And hopefully all of the database administrators and firewall rule sets have learned enough from the early 2000s that the era of worms flooding whole networks is passed [19]. It also seems likely criminals realized there was no money in that kind of attack, but that stealing money can be successful with stealthier attacks [20]. Defenders need the IDPS to recognize stealthy attacks.

Unfortunately for security professionals, statistics teaches us that it is particularly difficult to detect rare events. Bayes' theorem is necessary to demonstrate this difficultly, but let's consider the example of a medical test. What we are interested in finding is the false-positive rate—that is, the chance that the medical test alerts the doctor the patient has the condition when the patient in fact does not. We need to know two facts to calculate the false-positive rate: the accuracy of the test and the incidence of the disease in the population. Let's say the test is 91% accurate, and 0.75% of the population actually has the condition. We can calculate the chance that a positive test result is actually a false positive as follows: Where *Pr* is the probability of the event

[1] Structured Query Language (SQL) is a common database language. SQL Slammer is so named because it exploits a vulnerability in the database and then reproduces automatically through scanning for other databases to exploit.

in brackets ([ ]) and the vertical bar (|) between two events can be read as "given," it means that calculating an event is dependent on, or given, another. For example, the probability that the patient does not have the condition given the test result was positive could be written $Pr[healthy|positive]$. This is the probability the test result is an incorrect alert. Therefore:

$$Pr[healthy|positive] = \frac{Pr[positive|healthy] \times Pr[healthy]}{Pr[positive|sick] \times Pr[sick] + (Pr[positive|healthy] \times Pr[healthy])}$$

There will be a subtle difference here. We are not calculating the false-positive rate. That is simply $Pr[positive|healthy]$. We are calculating the chance that the patient is healthy given the test alerted the doctor to the presence of the condition. This value is arguably much more important than the false-positive rate. The IDPS human operator wants to know if action needs to be taken to recover security when the IDPS alerts it has recognized an intrusion. That value is $Pr[healthy|positive]$, what we're trying to get to. Let's call this value the alarm error, or *AE*. Let's simplify the precding equation by calling the false-positive rate *FPR*, and the true-positive rate *TPR*. The probabilities remaining in the equation will be the rate of the condition in the population, represented by the simple probability that a person is sick or healthy:

$$AE = \frac{FPR \times Pr[healthy]}{TPR \times Pr[sick] + (FPR \times Pr[healthy])}$$

Let's substitute in the values and calculate the *AE* in our example. The test is 91% accurate, so the *FPR* is 9% or 0.09, and the *TPR* is 0.91. If 0.75% of people have the condition, then the probability a person is healthy is 0.9925, and sick is 0.0075. Therefore:

$$AE = \frac{0.09 \times 0.9925}{0.91 \times 0.0075 + (0.09 \times 0.9925)}$$
$$AE = \frac{0.089325}{0.006825 + (0.089325)}$$
$$AE = 92.9\%$$

Therefore, with these conditions, 92.9% of the time when the test says the patient has the condition, the patient will in fact be perfectly healthy. If this result seems surprising—that with a 9% false-positive rate that almost 93% of the alerts would be false positives—you are not alone. It is a studied human cognitive error to underestimate the importance of the basic incidence of the tested-for condition when people make intuitive probability evaluations [21,22]. One can bring intuition in line with reality by keeping in mind that if there are not very many sick people, it will be hard to find them, especially

if the test for something is relatively complicated (like sickness or computer security intrusions). It is the proverbial needle-in-a-haystack problem.

There has been some research into the technical aspects of the effects of the base-rate problem on IDPS alarm rates [23]. The results are not very encouraging—the estimate is that the false-positive rate needs be at or below 0.00001, or 10^{-5}, before the alarm rate is considered "reasonable," at 66% true alarms. However, in the context of other industrial control systems, the studies of operator psychology indicate that in fields such as power plant, paper mill, steel mill, and large ship operations, the operator would disregard the whole alarm system as useless if the true alarm rate were only 66%.

The base-rate fallacy provides two lessons when considering an IDPS. First, when an IDPS advertises its false-positive rate as "reasonable," keep in mind that what is reasonable for a useful IDPS is much lower than is intuitively expected. Second, the base-rate problem has a lot to do with why signature-based operation is the predominant IDPS operational mode. It has much lower false positives, and so even though signatures may miss many more events, they can achieve sufficiently low false-positive rates to be useful. Given how noisy the Internet is, anomaly-based detection is still largely a research project, despite the alluring business case of a system that just knows when something looks wrong. The following two sections describe these two modes of operation.

An additional important point is that a grasp of statistics and probability is important for a network security analyst. For a treatment of the base-rate fallacy in this context, see Stallings [24, ch. 9A]. For a good introductory statistics text that is freely available electronically, see Kadane [25].

PROFILE: MARTIN ROESCH

The Father of Snort

As the story goes, one weekend in 1998 Marty Roesch sat down and wrote a little Linux program for traffic analysis, and after those two days he shared his creation with the open-source community [26]. Roesch was amazed at the positive response from the project, and how many people downloaded the code. This was how one of the most influential open-source projects of the current era was born: the Snort IDS. This led to a quick set of developments for a man who had just graduated from Clarkson University in 1992, a smaller technical school in upstate New York. In early 2001, Roesch hired four employees and founded Sourcefire, a network defense company. While Sourcefire sells a lot of commercial products to enhance Snort's capabilities, they still maintain the core Snort IDS engine as an open-source product.

In 2009, InfoWorld included Snort in their Open-Source Hall of Fame [27]. It was one of only three security-related tools included in the 36-item list. It's hard to accurately convey the impact Snort has had on the network defense

community. Reading the academic literature about intrusion detection systems before 1998, there is a tangible undercurrent of depression in the academic literature. The implementations up to that time were expensive, bulky, and, despite the cost, ineffective. Snort was not only free, but it consumed fewer computational resources than many of the commercial counterparts. Furthermore, it ran well in software and did not require special equipment purchases for smaller networks, unlike most IDSs at the time that did require special hardware.

In 2013, it is expected as a matter of course that you have NIDS defending your network. In 1998 a single NIDS was a luxury item. Snort and Marty Roesch are the primary forces that have bridged that gap. Roesch is still the CTO of Sourcefire, overseeing the technical development of Snort and all the other software components that enhance it. He also occasionally spends stints as the interim CEO, demonstrating a management and salesman skill that is rare in people who can write a functional piece of software in a weekend. Although Sourcefire is best known for its technology, Roesch says that his real goal and vision are to make Sourcefire "instrumental in transforming the way organizations manage and minimize their risks" [26].

MODES OF INTRUSION DETECTION

In broad strokes, NIDPSs can operate with signature-based or anomaly-based detection methods. Practically, both modes are used because they have important benefits and uses. However, it is useful to understand the different benefits, and limitations, of each style of detection, even though a single IDPS that is purchased will likely possess both signature-based and anomaly-based detection capabilities to some extent. For example, Snort (an open-source IDS, see the preceding sidebar) has been extended to perform some anomaly detection capabilities in various ways [28]. This categorization is similar to the reason to understand NIDPSs as a separate kind of device from the various types of firewalls as described in Chapter 5, even though devices on the market often do not strictly adhere to any category. Categorizing is helpful for the network defender to conceptualize and plan a coordinated defense strategy.

Network Intrusion Detection: Signatures

Signatures are one of two modes that IDPSs use to detect intrusions. The idea is that the NIDPS detects a known pattern, or signature, of a specific malicious activity that is exhibited within the traffic. Signature-based detection is simpler to implement than anomaly-based detection, and a good signature will have a lower, more consistent false-positive rate. However, signatures can be easy to evade and require the attack to be known before they can be written and the attack detected. Rules can usually be written fairly quickly once an attack is known, but someone, somewhere, must first detect it without a signature.

Signatures are quite flexible. A NIDPS aims to inspect the content of all the traffic that traverses the network. This is opposed to firewalls, which tend to only inspect the headers for the packets. This increased scope is arduous, and

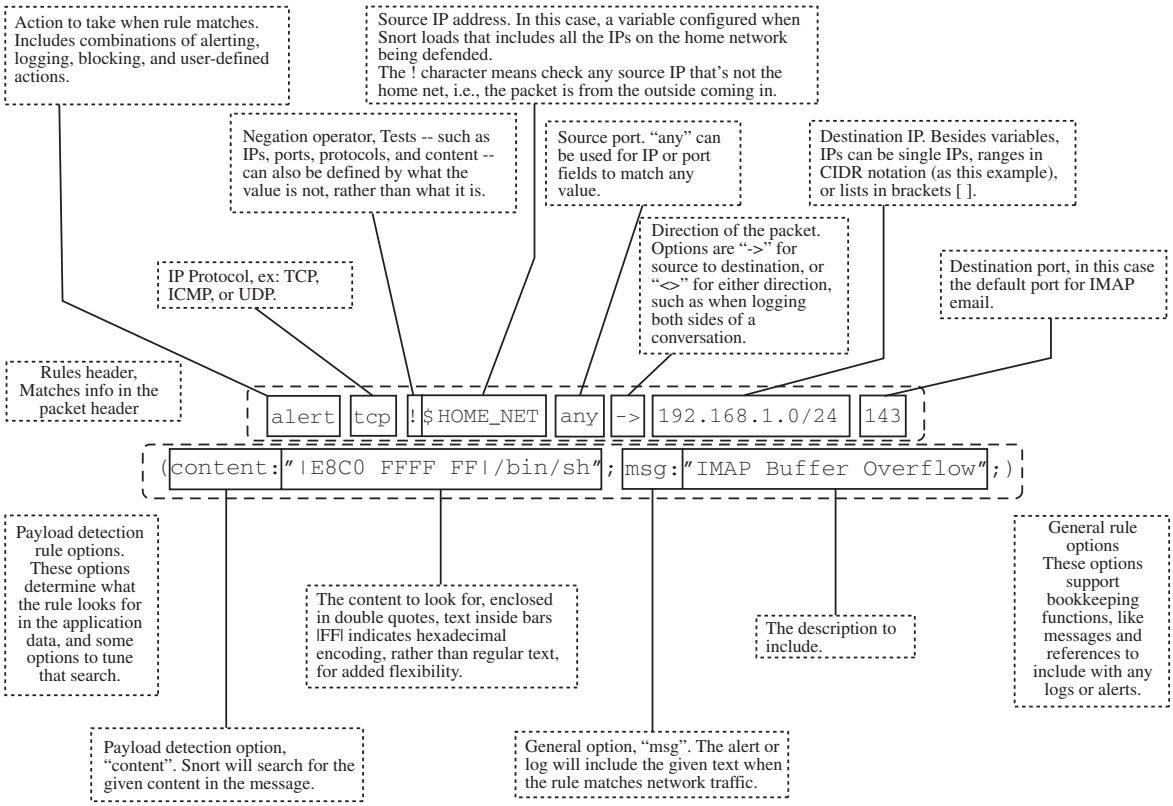

FIGURE 12.3

An example snort rule. The different features and fields of the rule are labeled. Source: *Roesch et al. [30].*

partly explains why NIDPSs can have performance issues. It also is the reason that they are such a useful tool.

To understand what a signature can accomplish, it may be helpful to understand the anatomy of a rule. The de facto standard format for a signature is its expression in a Snort rule. Snort is an open-source NIDPS that began development in 1998, and is primarily signature based. Due to its popularity, many other NIDPSs accept or use Snort-format rules. A rule is a single expression of both a signature and what to do when it is detected. Figure 12.3 displays a sample text Snort rule and annotates the components.[2] For a complete rule-writing guide, see the Snort documentation [29].

Figure 12.3 is not as complex as it may appear at first glance. There are three sections in this rule: one for what kind of rule this is and packets to look at, one for

[2]Compiled Snort rules exist, and these are in binary format, unreadable by humans. They have a different format than this text rule, which will not be discussed here, but the function is essentially the same.

what to look for in the packets, and a third for what to say once the rule finds a match. These sections are enclosed in light-gray dashed lines. The first section is the rules header, which is the more structured of the three. This section declares the rule type. Figure 12.3 is an "alert," one of the basic built-in types.

Rule types define what to do when the rule matches, which we'll return to later. The rest of the rule header will match against parts of the TCP/IP headers for each packet. This essentially performs a packet-filter, like a firewall access control list (ACL; see Chapter 5), even though the syntax and capabilities are different than a firewall. IP addresses can be matched flexibly, either with variables or ranges. Although not demonstrated in this rule, ports can also be specified with ranges or variables. IP addresses can also be specified as "any," for any value matches. The arrow indicates direction, and < > (not pictured) means either direction matches. Although the first versions used to allow it to be either →or←, that soon became confusing. Now only → is allowed, so the source IP is always on the left and the destination is always on the right.

The contents of the rule all go inside parentheses, with different parts separated by semicolons. This is where the flexibility of an IDPS rule can really be leveraged. Figure 12.3 demonstrates two of the simpler sections—payload detection and general—each with only their quintessential option demonstrated. Payload detection dictates what to look for in packet payloads, and it also offers some options to describe how and where to look for the content, which are not demonstrated in the figure. The content field is the quintessential option, as it defines the string that the signature will look for. In the example, there is a particular sequence of bits that can be used to inject malicious data into a particular email process in a known way, and this rule looks for that sequence of bits. Since this sequence is in the application data, it is clear why such signatures can be easily evaded with encryption—encrypting the payload would change these bits while in transit. The general section includes descriptors about the rule whenever an alert is sent or packets logged, so that the analyst does not have to remember that the hex sequence `E8C0FFFFFF` is particular to a buffer overflow in the Internet Message Access Protocol (IMAP), for email.

As the amount of available software has grown, so have the number of vulnerabilities. At the same time, older vulnerabilities do not go away—at least not quickly. If a vulnerability only afflicts a Windows 98 machine, most IDPSs in 2013 do not still need to track or block it. The basic premise is that if a vulnerability does not exist on the network the IDPS is protecting, it does not need signatures to detect intrusions resulting from it. But this in practice provides little respite, and the number of signatures that an IDPS needs to track grows rapidly.

There is a second way to view this threat landscape, and that is to ask what vulnerabilities attackers are commonly exploiting, and preferentially defend against them. There is always the chance that a targeted attack will use an

unknown, zero-day vulnerability to still penetrate the system undetected, but that is a different class of attacker. There is a set of common, lucrative criminal malicious software that exploits a small set of known vulnerabilities, and these are the attacks that are all but guaranteed to hit an unprepared network. There are only about 10 vulnerabilities of this mass-exploit quality per year, against only a handful of applications [31]. The best way to defend against such common exploits may not be an IDPS, however, an IDPS should be a device that helps the defenders decide what exploits are most common on their network, and thus which exploits deserve the special attention to prevention, such as the steps described in Guido [31].

Network Intrusion Detection: Anomaly Based

Anomaly-based detection generally needs to work on a statistically significant number of packets, because any packet is only an anomaly compared to some baseline. This need for a baseline presents several difficulties. For one, anomaly-based detection will not be able to detect attacks that can be executed with a few or even a single packet. These attacks, such as the ping of death, do still exist [32], and are much better suited for signature-based detection. Further difficulties arise because the network traffic ultimately depends on human behavior. While somewhat predictable, human behavior tends to be changeable enough to cause NIDPS anomaly detection trouble [33].

While signature-based detection compares behavior to rules, anomaly-based detection compares behavior to profiles [1]. These profiles still need to define what is normal, like rules need to be defined. However, anomaly-based profiles are more like white lists, because the profile detects when behavior goes outside an acceptable range. Unfortunately, on most networks the expected set of activity is quite broad. Successful anomaly detection tends to be profiles such as "detect if ICMP (Internet Control Message Protocol) traffic becomes greater than 3% of network traffic" when it is usually only 1%. Application-specific data is less commonly used. This approach can detect previously unknown threats, however, it can also be defeated by a conscientious attacker who attempts to blend in. Attackers may not be careful enough to blend in, but the particularly careful adversaries are all the more important to catch. In general, adversaries with sufficient patience can always blend in to the network's behavior. Therefore, anomaly detection serves an important purpose, but it is not a panacea, especially not for detecting advanced attackers.

NETWORK BEHAVIOR ANALYZERS

A network behavior analyzer (NBA) is an IDPS device that does not inspect the full packet data [1, p. 6-1]. Since anomaly detection often works on packet headers and dynamics, much of this value can be realized in a NBA

without the overhead of a full-fledged IDPS. There are many advantages to using such a lightweight device for anomaly detection, in which the payload data is much less useful. It is possible to custom build a NBA, however, there are already several existing data formats that store partial network traffic data, and these can be leveraged more easily. One such common format is network flow, which is introduced in Chapter 11.

There are several open-source tools for working with network flow data, such as SiLK [34]. This tool suite includes a NBA called Analysis Pipeline [35]. Since a NBA using network flow only has to process about 5% of the volume of data as an IDPS processing the full-packet capture, it can be a bit more scalable. This savings can be most felt when data from multiple sensor points needs to be aggregated. If full-packet capture needs to be rebroadcast from a monitoring point to a central location, bandwidth to and from that monitoring point needs to be double what it would be without the rebroadcast. The traffic needs to be sent to its intended location and the central anomaly detector, so it needs to be sent twice, doubling the bandwidth. If only flow is aggregated at the anomaly detector, bandwidth only needs to increase about 5%. A 5% increase instead of a 100% increase for full-packet capture makes back-haul more feasible.

WIRELESS IDPS

A wireless IDPS focuses on preventing abuse of the wireless access point and medium in the first place. According to the OSI model layers described in Chapter 11, a wireless IDPS only analyzes up to layer-2 data. Since this data is only useful in point-to-point communications, not end-to-end communications, a wireless IDPS must collect data from the wireless access points. This fact means that the sensor architecture needs to be distributed, unlike a NIDPS or NBA.

Wireless IDPSs monitor the radio waves for abuse or attacks on the wireless access points. They can also detect attempts to establish rogue access points for subversive communication. A wireless IDPS has its own radio antennae that it uses to scan the radio waves and issue commands to devices to correct abuse. The types of events that a wireless IDPS can detect include the following [1, p. 5–8]:

- Unauthorized wireless local area networks (WLANs) and WLAN devices
- Poorly secured WLAN devices
- Unusual usage patterns
- The use of wireless network scanners
- DoS attacks and conditions (e.g., network interference)
- Impersonation and middle-person attacks

Entities attacking a wireless device need to be physically close to the device, unlike most network attacks. Thus, the impact of these events is different than those detected by other IDSs. If an adversary can gain access via a wireless device, the adversary can often evade other defensive technologies, such as those described in Chapter 5. A wireless IDS is helpful in detecting attempts at such attacks.

NETWORK INTRUSION PREVENTION SYSTEMS

A network intrusion prevention system (NIPS) acts like a NIDS, except that it must process packets quickly enough to respond to the attacks and prevent them, rather than merely report the intrusion. This is much easier if the NIPS can enforce a choke point in the traffic flow. Therefore, a NIPS is recommended to be deployed inline, whereas a NIDS is recommended out of band [1]. This decision has several ramifications. It determines what risks the NIPS brings to the system and what remediations are feasible.

There are dangers associated with putting the NIPS inline, namely that it can become a single point of failure for the network. This possibility raises a question of what the NIPS should do when it can no longer keep up with traffic: Does it drop the packets, or pass them along without inspecting them for possible intrusions? If a device has the property that when it fails it maintains the security of the system, it is said to be fail-secure. The fail-secure operation option for an overloaded NIPS is to drop packets, because packets that are not routed cannot damage the system, whereas uninspected packets passed on may. This fail-secure operation may make the network fragile, and if it occurred would cause a DoS condition on that network link, and thus possibly the whole organizational network. Since an IDPS consumes a lot of computational resources, a NIPS will reach this overloaded condition more quickly than other network devices. Therefore, it is important to ensure that a NIPS is sufficiently provisioned before deploying it.

There are prevention attempts that a NIDPS can make either inline or out of band. As a general rule, out-of-band remediations are less effective than inline remediations. Inline remediations include performing a firewall action on the traffic, throttling bandwidth usage, or sanitizing malicious content. These three actions are unique to inline systems. The three actions available out of band are attempting to end TCP sessions, usually by spoofing TCP RST packets (packets with the reset flag set); changing the configuration of other network security devices to block future traffic; and executing some arbitrary program specified by the administrator to support functionality the IDPS does not natively support [1, p. 4–12ff]. Out-of-band actions are also available to inline systems, but inline systems do not usually use them because

out-of-band actions are less effective. The six available remediations, both inline and out of band, can be summarized as follows:

- *Perform a firewall action (inline only):* Dynamically create a rule to drop, reject, or log the packet. This rule might be only for the packet in question, or may apply to packets for some period of time into the future.
- *Throttle bandwidth (inline only):* Reduce the bandwidth available to a certain type of activity while it is evaluated further.
- *Sanitize malicious content (inline only):* Remove or overwrite certain parts of the packets before forwarding them along. Some proxies naturally do this by normalizing traffic, such as collecting all fragments of a packet and writing them back in a predictable format before forwarding, which will sanitize any fragmentation attacks.
- *Reset TCP session:* Spoof a packet to both the source and destination as if the IDPS is the other party in the communication, with the RST flag of the TCP header set. This flag is used to represent a forcible end to the TCP session, or a reset. A well-behaved host should abandon the connection, however, this is not guaranteed.
- *Reconfigure other network security devices:* Insert or change rules in devices, such as those described in Chapter 5, to affect future transactions and prevent further damage.
- *Execute a program:* If the IDPS cannot perform a certain action, the data can be passed to another program that can. An example might be to make a domain name system (DNS) or WHOIS (pronouced as "who is") query to include that data in a log file.

The superiority of inline remediations can be captured by discussing race conditions. Race conditions are a situation in which the outcome of the process is unpredictable due to two or more processes occurring in an unpredictable order. The term originates in software engineering [36], however, the concept applies to network behavior as well. At the risk of oversimplification, out-of-band remediations are race conditions and inline remediations are not. Therefore, out-of-band remediations are less reliable because the order in which operations occur is not stable or predictable. The condition is so named because, for example, the adversary and the IDPS are in a race as to which entity can execute its commands to affect the target host first.

There is one large-scale example of using TCP reset packets as a NIPS system. TCP resets are one of the methods used by the censorship system of the People's Republic of China (mainland China), commonly called the "Great Firewall of China." This method is less prone to overblocking and false positives than other methods at such a scale, such as DNS poisoning or dropping packets. On the other hand, it is also easy to evade a TCP-RST NIPS, such as

demonstrated in Clayton, Murdoch, and Watson [37], so long as both endpoints ignore the resets. Therefore, if both parties to a communication are intent on persisting, such as a bot and its command and control server, the defender should assume that a TCP-RST-based defense will be insufficient.

SUMMARY

This chapter discusses automated intrusion detection and prevention, primarily via the network. Intrusion detection and prevention systems are valuable tools in recognizing adversarial attacks on the network and initiating an appropriate response programmatically. Properly configuring an IDPS is a challenge, and the devices must be properly resourced. Problems detecting events with a low base-rate fallacy of occurrence also present a significant challenge to making use of IDPS alerts. Despite these challenges, IDPSs can provide crucial value to the network defender. No defensive strategy should be considered complete without a network IDPS to assist in recognition.

IDPSs operate in two general detection modes: signature based and anomaly based. These modes have different strengths and weaknesses that should be used to complement each other. A network behavior analyzer can leverage the strengths of anomaly detection with less overhead than a full NIDPS. Automatic prevention and remediation mechanisms can be enacted for rules or profiles that indicate particularly dangerous attacks or are particularly certain to be accurate. The available mechanisms vary based on the NIDPS architecture. Chapter 13 continues to discuss recognition strategies, in the context of host-based recognition and forensics.

REFERENCES

[1] Scarfone K, Mell P. Guide to intrusion detection and prevention systems (IDPS). NIST Special Publication 800-94, 2007.

[2] Shirey R. Internet security glossary, Version 2. RFC 4949, 2007.

[3] Anderson RJ. Security engineering: a guide to building dependable distributed systems, 2nd ed. New York: Wiley; 2008.

[4] Abrams SJR, Ghimire D. Corporate AV/EPP comparative analysis—exploit protection 2013. NSS Labs, Inc.; 2013. Retrieved from <https://www.nsslabs.com/reports/corporate-avepp-comparative-analysis-exploit-protection-2013>. Retrieved Feb 4, 2013.

[5] Wood M, Erlinger M. Intrusion detection message exchange requirements. RFC 4766, 2007.

[6] Bellovin SM. Packets found on an internet. *ACM SIGCOMM* Comput Commun Rev 1993;23(3):26–31.

[7] Cheswick WR, Bellovin SM, Rubin AD. Firewalls and internet security: repelling the Wily Hacker. Reading, MA: Addison-Wesley Professional; 2003.

[8] Ptacek TH, Newsham TN. Insertion, evasion, and denial of service: eluding network intrusion detection. Defense Technical Information Center; 1998.

[9] CERT Coordination Center. IP denial-of-service attacks. CERT Advisory CA-1997-28. Carnegie Mellon University, Pittsburgh, PA. Retrieved from <http://www.cert.org/advisories/CA-1997-28.html>; 1997.

[10] Shankar U, Paxson V. Active mapping: resisting NIDS evasion without altering traffic. In: Security and privacy. Proceedings of the 2003 Symposium on IEEE; 2003. p. 44–61.

[11] Novak J. Target-based fragmentation reassembly. Sourcefire Vulnerability Research Team; 2005. Retrieved from <http://www.snort.org/assets/165/target_based_frag.pdf>. Retrieved Feb 4, 2013.

[12] Pescatore J, Young G. *Defining next-generation network intrusion prevention*. Gartner, Stamford, CT; 2011.

[13] Kent S, Seo K. Security architecture for the internet protocol. RFC 4301. Updated by RFC 6040, 2005.

[14] Briscoe B. Tunneling of explicit congestion notification. RFC 6040, 2010.

[15] Dierks T, Rescorla E. The transport layer security (TLS) Protocol Version 1.2. RFC 5246. Updated by RFCs 5746, 5878, 6176, 2008.

[16] Callas J, Donnerhacke L, Finney H, Shaw D, Thayer R. OpenPGP Message Format. RFC 4880, 2007. Updated by RFC 5581.

[17] Electronic Frontier Foundation. HTTPS everywhere FAQ. Retrieved from <https://www.eff.org/https-everywhere/faq>; 2013. Retrieved Feb 4, 2013.

[18] Sourcefire. The case for the next-generation IPS. Whitepaper, 2011.

[19] Travis G, Balas E, Ripley D, Wallace S. Analysis of the SQL slammer worm and its effects on indiana university and related institutions. Indiana University; 2004. Retrieved from <http://paintsquirrel.ucs.indiana.edu/pdf/SLAMMER.pdf>. Retrieved Feb 4, 2013.

[20] Anderson R, Moore T. The economics of information security. Science 2006;314(5799):610–3.

[21] Kahneman D, Tversky A. On the psychology of prediction. Psychol Rev 1973;80(4):237.

[22] Bar-Hillel M. The base-rate fallacy in probability judgments. Acta Psychologica 1980;44(3):211–33.

[23] Axelsson S. The base-rate fallacy and the difficulty of intrusion detection. ACM Trans Inf Syst Secur (TISSEC) 2000;3(3):186–205.

[24] Stallings W. Network security essentials: applications and standards, 4th ed. Englewood Cliffs, NJ: Prentice-Hall; 2011.

[25] Kadane JB. Principles of Uncertainty. Boca Raton, FL: Chapman & Hall; 2011. Retrieved from <http://uncertainty.stat.cmu.edu/wp-content/uploads/2011/05/principles-of-uncertainty.pdf>.

[26] Hopkins S. From Snort to sourcefire to Nasdaq. *Clarkson University Magazine*; 2009, Summer. Retrieved from <http://www.clarkson.edu/alumni_magazine/summer2009/cyber-security_roesch.html>. Retrieved Feb 4, 2013.

[27] Dineley D, Mobley H. The greatest open-source software of all time. *InfoWorld Magazine*; 2009, August 17. Retrieved from <http://www.infoworld.com/d/open-source/greatest-open-source-software-all-time-776>. Retrieved Feb 4, 2013.

[28] Szmit M, Wężyk R, Skowroński M, Szmit A. Traffic anomaly detection with snort Information systems architecture and technology. Information Systems and Computer Communication Networks, Wydawnictwo Politechniki Wroclawskiej, Warsaw; 2007. p. 181–87.

[29] Roesch GC. Martin, Sourcefire 3: writing snort rules. In: SNORT Users Manual 2.9.4. Sourcefire, Inc. Columbia, MD. Retrieved from <http://manual.snort.org/node27.html>; 2013. Retrieved Feb 24, 2013.

[30] Roesch M, et al. Snort-lightweight intrusion detection for networks. Proceedings of the 13th USENIX conference on system administration, Seattle, 1999. p. 229–38.

[31] Guido D. The exploit intelligence project. iSEC Partners; 2011. Retrieved from <http://www.trailofbits.com/resources/exploit_intelligence_project_2_slides.pdf>. Retrieved Apr 4, 2013.

[32] Keizer G. Microsoft patches 1990's-era "Ping of Death". ComputerWorld 2011, August 9.

[33] Uddin M, Rahman AA. Dynamic multilayer signature-based intrusion detection system using mobile agents, arXiv:1010.5036 2010. <http://arxiv.org/abs/1010.5036>. Retrieved Sept 20, 2013.

[34] Software Engineering Institute. CERT NetSA security suite—monitoring for large-scale networks. Retrieved from <http://tools.netsa.cert.org/>; 2013. Retrieved Jul 4, 2013.

[35] Ruef D. Analysis pipeline—streaming flow analysis U.S. government forum of incident response and security teams. Nashville, TN: US-CERT; 2011. Retrieved from <http://www.us-cert.gov/sites/default/files/gfirst/presentations/2011/Analysis_Pipeline.pdf>. Retrieved Feb 4, 2013.

[36] Netzer RHB, Miller BP. What are race conditions? some issues and formalizations. ACM Letters on Programming Languages and Systems 1992, March(1):74–88. Retrieved from <http://doi.acm.org/10.1145/130616.130623>.

[37] Clayton R, Murdoch S, Watson R. Ignoring the Great Firewall of China Privacy enhancing technologies. In: 6th Workshop on Privacy Enhancing Technologies. Cambridge, UK. June 28–30, 2006. p. 20–35.

[38] McDonald M. Adding more bricks to the Great Firewall of China IHT rendezvous. New York Times; 2012. Retrieved from <http://rendezvous.blogs.nytimes.com/2012/12/23/adding-more-bricks-to-the-great-firewall-of-china/>. Retrieved Feb 27, 2013.

Chapter Review Questions

1. What is IP fragmentation? How can it be used to evade an IDPS?
2. What is the base-rate fallacy? What challenge does it present to utilizing IDPS alerts?
3. How does an IDPS signature work?
4. What are two major differences between signature-based detection and anomaly-based detection?
5. Imagine a medieval king trying to test for poison in his food. He has a different food-tester taste each dish before he eats it (this is extravagantly many food-testers, but let's just go with it because it makes the math easier). Let's say the king eats 10 different dishes per day. If a food-tester eats a poisoned dish, there is a 95% chance that person dies within the testing window, before the king eats the food. Due to poor sanitation and food preparation conditions, two food-testers die every 30 days of natural causes during the testing time. If there are two attempts

to kill the king with poisoned food per 360 days, over 5 years (360-day cycles—they have bad calendars), how many food-testers actually died from poisoned food? What is the expected probability if a food-tester died that the food destined for the king was actually poisoned?[3]

Chapter Exercises

1. Use [1, ch. 9] to evaluate an IDPS product. If you need an open-source product to consider, Bro-IDS (*http://www.bro.org/documentation/*) or Snort (*http://snort.org/*) would probably be suitable.
2. Using available reporting, try to determine the top 5 to 10 crucial vulnerabilities that are being exploited right now on the Internet at large.
3. The "Great Firewall of China" has been modernizing to resist evasive users [38]. Can you think of, or find, any methods to circumvent these new additions? Have there been further updates since the end of 2012 that also would need to be evaded?

[3]Answer: If there are 10 attempts to poison the king in 5 years, if the food-testers are 95% effective, then 10×0.95 food-testers should die. During the same course of time, 120 testers will die of natural causes $(2(testers/month) \times 12(months/year) \times 5(years/sample\_period))$. The alarm error rate can be calculated with Bayes' theorem as follows:

$$AlarmError = \frac{FPR \times Pr[no\,poison]}{TPR \times Pr[poison] + (FPR \times Pr[no\,poison])}$$

$$AE = \frac{2/300 \times 3598/3600}{.95 \times 2/3600 + (2/300 \times 3598/3600)}$$

$$AE = \frac{0.00666296296}{0.0005277777778 + (0.00666296296)}$$

$$AE = 92.66\%$$

Therefore, if a food-tester died, there is a 92.66% chance the food was not poisoned. Therefore, the chance that the food was actually poisoned is $1 - 0.9266 = 7.44\%$.