

Network Intrusion Detection Signatures, Part Three

Karen Kent Frederick 2002-02-19

Network Intrusion Detection Signatures, Part Three

by *Karen Kent Frederick*

last updated February 19, 2002

This is the third in a series of articles on understanding and developing signatures for network intrusion detection systems. In [Part One](#) and [Part Two](#), we examined the use of IP protocol header values, particularly TCP, UDP and ICMP, in network intrusion detection signatures. In this article, we will continue our discussion of signatures by studying the area of protocol analysis, focusing on the examination of values within TCP and UDP payloads. Network intrusion detection using protocol analysis-based signatures is very effective in detecting both known and unknown attacks involving protocols such as DNS, FTP, HTTP and SMTP.

The Basics of Protocol Analysis

The first two articles in this series focused on developing network intrusion detection signatures using values in IP, TCP, UDP and ICMP headers. Now we want to look at signatures that examine the payloads within TCP and UDP packets, which contain other protocols. It's important to understand that a protocol such as DNS is contained within TCP or UDP, which itself is contained within IP. So we first decode a packet's IP header information, which will tell us whether its payload contains TCP, UDP or another protocol. If the payload is TCP, for example, we then need to process some of the TCP header information within the IP payload before we can access the TCP payload. DNS data is contained within UDP and TCP payloads.

Because intrusion detection systems are normally most interested in IP, TCP, UDP and ICMP characteristics, they normally are able to decode some or all of their headers. However, only more advanced intrusion detection systems are capable of performing protocol analysis. Such IDS sensors perform full protocol decoding for DNS, HTTP, SMTP and other widely used protocols. Due to the complexity of decoding each protocol, as well as the sheer number of protocols in wide use, protocol analysis requires much more advanced IDS capabilities than simpler signature techniques, such as "packet grepping". IDS sensors that perform packet grepping simply look for a particular string or sequence of bytes within a packet; the sensor has no real knowledge of the protocol that it is examining, so it can only identify malicious activity that has an obvious, simple signature.

The term “protocol analysis” means that the IDS sensor understands how various protocols work and closely analyzes the traffic of those protocols to look for suspicious or abnormal activity. For each protocol, the analysis is based not only on protocol standards, particularly the RFC’s, but also on how things are implemented in the “real world”. Many implementations violate protocol standards, so it is very important that signatures reflect how things are really done, not how they are ideally done, or many false positives and negatives will occur. Protocol analysis techniques observe all traffic involving a particular protocol and validate it, alerting when the traffic does not meet expectations.

At this point, it is probably not yet clear to you why protocol analysis is a better technique than simple packet grepping. The problem with packet grepping signatures is that they are usually written to look for a particular known exploit. In most cases, you can’t write a packet grepping signature to identify an unknown attack or a variation on a known attack, or to identify general attacks that are attempting to exploit a particular vulnerability. Therefore, packet grepping signatures tend to have limited effectiveness in identifying attacks. Protocol analysis gives you the ability to look for any activity that violates standards or expected behavior, which facilitates having network IDS sensors detecting both known and unknown methods of attack.

Comparing Packet Grepping and Protocol Analysis Signatures

Let’s illustrate the advantage of looking for vulnerabilities instead of exploits with an example. Consider the various exploits and vulnerabilities involving the FTP protocol. If you search the Internet for information on FTP vulnerabilities, you would find dozens, maybe even hundreds of them. Now imagine how many different FTP exploit programs and scripts exist, and how many variations on those programs there could be. And imagine how many of these programs aren’t publicly available. So how can you create a good signature set if you base your signatures on just the exploit programs that you can find? Of course, the answer is that you can’t. You’ll certainly catch some attacks with the signatures, but who knows how many you will miss?

An additional issue is that of timeliness. Obviously, if you are creating a signature set based on known exploits, you can’t write a new signature until you see the exploit. After an exploit has become publicly available, the signature writer must acquire a copy of the exploit it, analyze or test it to determine how it works, and then develop, test and distribute a signature based on the exploit’s characteristics. By definition, this means that a signature that alerts when the exploit occurs will not exist until some time after the exploit is publicly available. In most cases,

there will be a significant delay between the time the exploit is first used and the time that the IDS can recognize its activity.

Now let's think about signature development from the opposite point of view. Rather than focusing on writing signatures to match exploit programs, let's create a signature set based on known and potential vulnerabilities. A signature set based on protocol analysis will have extensive knowledge of the protocol's expected behavior. The FTP signature set would be aware of all valid FTP commands and would note any unknown commands that it detected. This could identify various security issues, such as a non-FTP application being run on an FTP port, or a buffer overflow attempt that provides a string of a hundred 'a' characters for a command name. Another part of the signature set could verify that particular FTP command arguments contain only alphanumeric characters, and alert on any arguments that contain binary data, such as shellcode.

Other parts of a protocol analysis-based signature set would do additional validations on various commands and arguments, to look for any other signs of abnormal or suspicious activity. These validations would correspond to known vulnerabilities and likely unknown vulnerabilities, such as buffer overflows and illegal values. Of course, we can't psychically predict what unknown vulnerabilities exist. But we can anticipate them by checking various fields for abnormal values. There are two primary reasons for this:

1. The vulnerability may involve an unexpected value for a particular field. For example, the largest valid argument for command FOO may be 64 characters, but an attacker specifies an argument with 500 characters.
2. The vulnerability may be coded poorly, using unusual values which are noteworthy but unrelated to the vulnerability. An example would be a protocol header value that is nearly always set to a value of 1, yet a particular exploit happens to be coded to set that value to 16.

Attacks in both categories can be identified using protocol analysis; we can catch these attacks by validating the protocol's data. In the first category, we can create a signature that will identify the likely intent of the attacker's exploit. In the second category, we may not necessarily be able to determine the purpose of the attack, but we can determine that unusual activity is occurring and likely needs further investigation. So rather than basing a signature set on the characteristics of exploits, we instead base the signature set primarily on a careful analysis of the given protocol.

Performing Protocol Analysis on FTP Traffic

When we perform network intrusion detection using protocol analysis, we are examining the header values and/or payload values for the protocol, as applicable. This can cover an incredible range of signatures, from looking for certain strings in email headers that indicate an email-borne worm or virus, to a suspicious URL that could indicate a web server attack, to a FTP command whose argument contains shellcode. For the sake of illustration, we will look at the characteristics of some real and hypothetical FTP vulnerabilities, and discuss how protocol analysis techniques can identify them.

A buffer overflow vulnerability in the FTP MKD command might be exploitable by providing a very long argument to it containing shellcode. Typical packet grepping signatures contain sequences of the shellcode (often 10 to 20 bytes) and need to match it exactly anywhere within the FTP packet. Protocol analysis allows us to identify the argument to the MKD command and to then verify that it is not overly long and that it does not contain binary data, both of which are conditions we'd expect to find if a buffer overflow exploit is being attempted. By checking this argument "generically" instead of looking for particular shellcode sequences, we will find many different attempts to exploit it, rather than just the few known exploits. Also, attackers cannot easily avoid detection by making a change to the shellcode; the packet grepping signatures would be fooled, but the protocol analysis signatures would not.

The FTP command "SITE EXEC" can be used to execute commands on an FTP server, and it has been used in many attacks. "SITE" is the actual FTP command name, and "EXEC" is an argument to the "SITE" command. Packet grepping signatures look for "SITE EXEC" in a packet, trying to make a case-insensitive match of that string. Protocol analysis takes a different approach, looking for the FTP command "SITE" with the argument "EXEC". So what's the difference? Well, attackers may try to evade detection by IDS sensors by adding superfluous whitespace between "SITE" and "EXEC", such as several extra spaces. Many FTP servers ignore the additional spaces, so they see "SITE EXEC" and "SITE EXEC" as having the same meaning. Obviously, a packet grepping signature that just does a string comparison will not be able to match these two strings; a protocol analysis signature that understands how to parse "SITE EXEC", or any variation thereof, into its key components will still be able to identify this attack accurately.

Conclusion

In this series, we have been learning the basic concepts of network intrusion detection signatures. This article has introduced the concept of protocol analysis, which means that the network IDS actually understands how various protocols, such as FTP, are supposed to work, and verifies that the traffic it sees follows the expected behavior for that protocol. Protocol analysis has several benefits over more primitive signature techniques, such as packet grepping. By using a protocol analysis-based IDS solution, you will have much better detection of both known and unknown attacks. By focusing on anomalies within the traffic, rather than simply looking for the signatures of particular exploits, protocol analysis-based signatures are much more difficult for attackers to evade through changes to exploits' code or IDS obfuscation techniques.

The next article in this series will continue the discussion of protocol analysis. We will look at more of the obfuscation techniques that attackers use and explain why protocol analysis is so helpful in overcoming attempts to hide attacks. After that, we will examine the topic of stateful protocol analysis. So far, we have only looked at limited protocol analysis – examining a single FTP packet, or a single FTP command or response. In stateful protocol analysis, we actually examine all traffic within a session – for example, an entire FTP session: the initial TCP connection, the FTP authentication, the interchange of FTP commands and responses, the usage of FTP data connections, and the teardown of the TCP connection. Stateful protocol analysis is one of the most fascinating and exciting areas of network intrusion detection signatures, and we will be examining it in depth in a future article.

Karen Kent Frederick (kkent@bigfoot.com) is a senior security engineer for the Rapid Response Team at NFR Security. She is a graduate of the University of Wisconsin-Parkside and is currently completing her master's in computer science, focusing in network security, through the University of Idaho's Engineering Outreach program. Karen has over 10 years of experience in technical support, system administration and information security. She holds several certifications, including SANS GIAC Certified Intrusion Analyst, GIAC Certified Unix Security Administrator, and GIAC Certified Incident Handler. She is one of the authors of "Intrusion Signatures and Analysis", and she is a contributing author to the "Handbook of Computer Crime Investigation".

Relevant Links

[Network Intrusion Detection Signatures, Part One](#)

Karen Kent Frederick

[Network Intrusion Detection Signatures, Part Two](#)

Karen Kent Frederick

[Privacy Statement](#)

Copyright 2006, SecurityFocus