

Evaluating Network Intrusion Detection Signatures, Part Two

Karen Kent Frederick 2002-10-01

Evaluating NID Signatures, Part Two

by *Karen Kent Frederick*

September 25, 2002

In this series of articles, we present recommendations that will help readers to evaluate the quality of network intrusion detection (NID) signatures, either through hands-on testing or through careful consideration of third-party product reviews and comparisons. The [first installment](#) discussed some of the basics of evaluating NID signature quality, as well selecting attacks to be used in testing. This article will conclude the discussion on criteria for choosing attacks and then provide recommendations for generating attacks and creating a good testing environment. We begin by discussing some methods of acquiring attacks and attack traffic.

Downloading Exploits From the Web

One shortcoming with most NID evaluations is that the attacks that are used to evaluate NID signature quality are chosen in an *ad hoc* manner. The tester typically visits a few Web sites that have exploit source code and executables, downloads some of the files, and runs them. There are several potential problems with doing this:

- Because exploits are published by so many individuals and groups, and are hosted in so many different places, there is typically no way of verifying their authenticity. The file could have been changed or replaced since it was originally distributed.
- Some exploits claim to do one thing but really do something else; for example, a program that claims to perform a format string attack against an FTP server might actually install a backdoor on the host that runs the attack. This may be done instead of the attack, or in addition to it.
- An exploit that is available in executable form only cannot be reviewed to confirm that it does what it's supposed to. Exploits that are available in source code form are typically written in any one of several languages; many also include shellcode sequences. In order to review the code of various exploits and confirm that they meet your needs, you'd either need to be at least somewhat familiar with various languages or have access to others in your organization who have the necessary knowledge and the time to assist you.
- Many exploits are purposely "broken" by their authors before they are made publicly

available. Certain instructions or values are modified so that the tool doesn't execute the attack properly. Unless you are skilled in the required languages and exploit techniques, and have the time to dig into the code and figure out how it's supposed to work and what needs to be fixed, you probably won't be able to use such exploits in your testing.

- For each exploit, you'll need to figure out how to use it. What arguments do you need to supply? What other software packages, services or other components is this exploit dependent upon? If the attack will only run from a particular operating system, you may need to acquire or set up a host to run it. If the attack targets a particular operating system, protocol and/or application, you may need to set up a host that meets the target requirements. Many exploits verify the vulnerability of the victim before actually launching the attack; for example, the exploit may connect to the victim, acquire the FTP server banner, and determine whether the FTP server is a vulnerable version based on the information in the banner message. This can rapidly become far too resource-intensive to be practical.
- This may seem like an obvious point, but the exploit is going to attack your systems. Let's assume that you are using test systems for your attack targets, systems that can be wiped out and rebuilt as needed. Suppose an attack that you run replaces your FTP server with a Trojaned version. If you want to do additional testing of the FTP attack detection capabilities of your intrusion detection system, you may first need to replace the Trojaned FTP server with the original version, as other exploits may be expecting and requiring a particular vulnerable version of the FTP server to be present.

If it isn't already clear at this point, performing intrusion detection signature testing with real exploits is often time- and resource-intensive. It can also be dangerous: running real exploits in any type of live environment is very risky. No matter how cautious you are in acquiring and evaluating exploit code, mistakes can and will still occur. A mistyped target IP address can cause a production server to be breached instead of a designated target server. To be safe and to ensure that they do not inadvertently break their own systems, readers should utilize a completely isolated test network and test hosts. Of course, this further increases the time and resources that are needed to perform IDS testing. Therefore, it is often best for all but the most skilled, enthusiastic and patient testers to avoid the use of published exploit code when performing intrusion detection signature testing.

Using IDS Testing Tools

A safer and faster alternative to using real exploits is to purchase and utilize an IDS testing

tool. The best-known IDS testing tool is [Blade IDS Informer](#), from [Blade Software](#). Informer works by replaying IP, UDP and ICMP packets, as well as complete TCP sessions that contain various scans, probes and attacks. You can modify the source and destination IP and MAC addresses that the packets use as needed. Informer comes with hundreds of attacks, divided into categories of related attacks; the user can select which attacks or groups of attacks they would like to use. Blade regularly updates the Informer attack suite so you can keep your IDS testing fairly current with new attacks and attack techniques.

The greatest benefit behind IDS testing tools is that the vendor has gone to the trouble of doing what you might otherwise have to do – downloading and verifying exploits. The IDS testing tool vendor has the expertise to confirm the validity of each exploit and to test it in a controlled environment. The attacks are also “neutralized” by the vendor; although the attack is executed, it is altered as necessary so it does not damage the target host. Besides these attacks, the vendor may also create their own exploits and exploit traffic, based on known vulnerabilities. This gives them full control over the attack without having to rely solely on the use of published exploits.

IDS testing tools typically have other features that can be very helpful in performing intrusion detection signature testing. Testing tools normally have a user-friendly interface, which makes the process of choosing and launching attacks much easier. With a few minutes of configuration and selection, traffic from many attacks can be replayed. Think of the time you would spend setting up an IDS testing tool, versus downloading, evaluating, compiling and running even ten exploits on your own. Now think of the time if you wanted to use five hundred exploits in your testing. An additional benefit is that the tester doesn’t need any knowledge as to how the attacks work as the vendor has verified what each attack does. Report generation is also at least somewhat automated, reducing the time needed to document your results.

Another nice feature of IDS testing tools is that users are not limited to only the attacks that they include – they can also create custom attacks. For example, Informer can capture and replay attack traffic generated by the user. If you want to use a particular attack, you can do so and record its traffic for future use. This points out an important aspect of intrusion detection testing: repeatability. You may wish to validate the strength of your intrusion detection system’s signatures periodically, to confirm that the signatures not only identify new significant threats, but also continue to detect older threats that are still relevant in your environment. To save time, it’s highly desirable for testing procedures to be easily repeatable. Both the use of a testing tool such as Informer and the ability to capture and replay your own attack traffic at a

later date, without having to rerun the individual attacks, are great ways to make your intrusion detection testing more efficient and consistent.

IDS testing tools also make it possible to perform IDS testing in a wider variety of environments. We emphasized earlier that when you are testing with live and potentially dangerous exploits, it's critical to utilize an isolated test environment. If you are utilizing the standard attacks provided by the testing tool, which are modified to prevent damage, then you are probably relatively safe to run them in a production environment. This is particularly true if you modify the IP addresses of the traffic so that the source and destination addresses are unused in your environment. Also, because tools such as Informer are replaying traffic, not actually performing interactive attacks with a server, you can save substantial time and resources in terms of hardware and software deployment. As long as you have a box to run the IDS testing tool from, you can perform your testing.

Issues with Using IDS Testing Tools

As with everything, there are also some potential issues with relying solely on IDS testing tools for evaluating IDS signature strength. First, you must consider the relevance of the attacks used by the testing tool in your environment. In an environment that uses common protocols and popular applications and operating systems, it's very likely that the attacks provided by the tool will closely match the potential threats against your environment. If your environment is unusual, then you should determine how many of the IDS testing tool's attacks are pertinent for your environment. It does no good to buy a tool that performs 95% of its tests on signatures for operating systems, protocols and applications that your systems do not run.

Another issue with IDS testing tools is that you must trust that the vendor has neutralized the attacks properly. Imagine what would happen if you ran attacks on a production segment and an improperly modified attack breached one of your servers! If you are sufficiently paranoid – and in the security world, it usually pays to be paranoid – you would utilize an IDS testing tool on an isolated network. It can be as simple as setting up one host to run the IDS testing tool, a second host to run the IDS sensor software, and connecting the two hosts through a hub. Of course, if you want to evaluate the quality of signatures in an IDS sensor that's already deployed into production, such a test may not be possible. If you trust the IDS testing tool, you can run it in a production environment; if you don't, you may run its tests in an isolated environment, examine the traffic to confirm that it's not harmful, and then run the tests again in production.

You should keep in mind that the testing tool vendor may have made mistakes in the attacks so that the attack traffic it generates does not properly reflect the actual attack. This can cause inaccurate testing results – an IDS sensor may be able to detect attack X but the IDS testing tool vendor may be replaying an invalid version of attack X. Also, the IDS signature may be intended to identify a harmful part of the attack that the IDS testing tool vendor has removed from the attack so as not to damage a target system.

A final and more subtle point is that the IDS testing tools themselves may introduce a bias into the testing. If a popular IDS testing tool uses 300 known attacks, an IDS vendor may write signatures that specifically match all these known attacks. This would have the effect of giving that vendor a higher score than others on signature quality. Depending on how the signatures are written, this may be accurate – or not. Vendors may write very weak signatures that simply match the testing tool's implementation of particular attacks and don't capture any other instances of the attack. If that's the case, then the testing results will be unfairly biased toward vendors that choose to implement such weak signatures simply for the purpose of improving their test results. To counteract this, you should not rely simply on an IDS testing tool for your evaluation.

Summary

In this article, we have examined in depth two ways to test intrusion detection signature quality: by downloading published exploits from the Internet, and by utilizing an IDS testing tool, specially designed to test signature capabilities. Although there is no direct cost in downloading exploit code, it can be prohibitively expensive to base your IDS testing on downloaded code because of the extensive knowledge, time and computing resources required. Commercial IDS testing tools reduce most of the need for these resources, but their use alone may cause biased test results. The next article in this series will examine other tools that are useful for performing IDS signature testing, as well as tools that are useful in testing NID anomaly detection and the general ability for IDS sensors to detect novel attacks.

Karen Kent Frederick (kkent@bigfoot.com) is a Senior Intrusion Detection Analyst for EDS in Herndon, Virginia. 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 "Inside Network Perimeter Security: The Definitive Guide to Firewalls, Virtual Private Networks, Routers, and Intrusion Detection Systems".

Relevant Links

[Evaluating NID Signatures](#)

Karen Kent Frederick, SecurityFocus

[Privacy Statement](#)

Copyright 2006, SecurityFocus