Chapter 10

Attack Detection and Defense

Solutions in this Chapter:

- Introduction to the ScreenOS Security Features
- Understanding the Anatomy of an Attack
- Configuring SCREEN Settings
- Applying Deep Inspection
- Setting up Content Filtering
- Understanding Application Layer Gateways
- Applying Best Practices

- Summary
- Solutions Fast Track
- Frequently Asked Questions
Introduction to the ScreenOS Security Features

This chapter will cover the nuts and bolts of the security features in Juniper Networks’ NetScreen firewall products. As you’ve no doubt already discovered, these devices are packed with features that make life easier for administrators—easy to configure VPNs (virtual private networks), built-in DHCP (Dynamic Host Control Protocol) servers, advanced Network Address Translation (NAT) functionality, support for a wide range of routing protocols, and much more. But a firewall’s primary responsibility has always been security—keeping the bad bits out, and letting the good bits in.

In addition to the strong feature set found for network administration is an equally strong set of protective tools. NetScreen firewalls have always protected owners from classic attacks such as Land, Teardrop, and other network layer-based attacks. These defensive SCREEN features allow for zone-specific settings based upon the risk factor of the facing network segment.

And while protecting at the network layer is both important and efficient, in today’s world of application layer-specific attacks, it’s not sufficient security coverage all by itself. Starting with tentative steps for application layer coverage in ScreenOS 4.0 with the Malicious URL feature, NetScreen firewalls now have full application layer coverage for typical Internet-facing protocols with Deep Inspection (DI), found in ScreenOS version 5.0 and later.

Combine the application layer gateway features with the advanced filtering features and antivirus (AV) protection, and a complete coverage picture emerges. But what are we protecting ourselves from?

Understanding the Anatomy of an Attack

There are almost as many different ways of attacking a network as there are hackers who try it, but the majority of attack methods can be categorized as one of the following: manual attacks and automated attacks. Manual attacks are generally still performed by a piece of code or other script, but the attack itself is initiated at the request of a live user, who selects his or her targets specifically. Automated attacks cover the kinds of attacks made by self-propagating worms and other viruses. There’s also a question of the competence of an attacker or complexity of an automated attack, which we’ll discuss.
The Three Phases of a Hack

Most hack attacks follow a series of phases:

1. **Reconnaissance** Initial probing for vulnerable services. Can include direct action against the target, such as port scanning, OS (operating system) fingerprinting and banner capturing, or it can be performing research about the target.

2. **Exploit** The attempt to take control of a target by malicious means. This can include denying the service of the target to valid users. Generally, the ultimate goal is to achieve root, system or administrator level access on the target.

3. **Consolidation** Ensuring that control of the target is kept. This usually means destroying logs, disabling firewalls and antivirus software, and sometimes includes process hiding and other means of obfuscating the attacker’s presence on the system. In some extreme cases, the attacker may even patch the target against the exploit he used to attack the box, to ensure that no one else exploits the target after him.

While each step may have more or less emphasis, depending on the attacker, most hack attacks follow this pattern of progression.

Script Kiddies

For manual attacks, the majority of events are generated by inexperienced malicious hackers, known both in the industry and the hacking underground as “Script Kiddies”. This derogatory reference implies both a lack of maturity (“just a kid”) as well as a lack of technical prowess (they use scripts or other pre-written code instead of writing their own). Despite these limiting factors, what they lack in quality, they more than make up for in quantity. Under a hail of arrows, even the mightiest warrior may fall. These sorts of attacks will generally be obvious, obnoxious, and sudden, and will usually light up your firewall or IDP (Intrusion and Detection Prevention) like a Christmas tree.

The majority of these attacks have no true intelligence behind them, despite being launched by a real person. Generally the reconnaissance phase of these sorts of attacks will be a ‘recon-in-force’ of a SYN packet and immediately transition to phase two by banging on your front door like an insistent vacuum cleaner salesman. Script Kiddies (also “Skr1pt Kiddies”, “Newbies”, or just “Newbs/Noobs”) glean through security websites like Security Focus...
(www.securityfocus.com), Packet Storm Security (http://packetstormsecurity.nl), and other sites that provide proof of concept code for exploits for new scripts to try out. Once they have these scripts, they will blindly throw them against targets. Very few of these amateurs understand exactly how these hacking tools work or how to change them to do something else. Many sites that provide code realize this, and will purposely break the script so that it doesn’t work right, but would with a simple fix after a walk-through of the code by an experienced security professional.

Unfortunately, that only stops the new, inexperienced, or unaffiliated hacker. More commonly, hacking groups or gangs form with a few knowledgeable members at its core, with new inept recruits joining continuously. The people themselves need not live near each other in real life, but rather meet online in Internet Relay Chat (IRC) chat rooms and other instant messaging forums. These virtual groups will amass war chests of scripts, code snippets, and shellcode that work, thanks to the work of the more experienced members. Often, different hacking groups will start hacking wars, where each side attempts to outdo the other in either quantity or perceived difficulty of targets hacked in a single time span. Military targets in particular are seen as more difficult, when in fact generally the security of these sites is often well below corporate standards. Mass website defacements are the most common result from these intergroup hacking wars, with immature, lewd, or insulting content posted to the sites.

A bright side to this problem is that many times a successful breach by these amateurs is not exploited to its fullest, since many of these hackers have no clue to exactly what sort of system they have gained access to, or how to proceed from there. To them, owning (a successful hack which results in a root, administrator, or system-level account) a box (a server), and modifying its presented webpage for others to see and acknowledge is generally sufficient. These sorts of attacks commonly do not proceed to phase three, consolidation.

From a protection standpoint, to defend against these sorts of attacks, it is important to keep DI and IDP signatures updated, and all systems patched, whether directly exposed to the Internet or not. Defense-in-depth is also key to ensure that a successful breach does not spread. The motivation behind these groups is quick publicity, so expect hard, fast, obvious, but thorough strikes across your entire Internet-facing systems.
Black Hat Hackers

Experienced malicious hackers (sometimes called “Black Hat” hackers or just “Black Hats”) tend to have a background of either a Script Kiddie graduating from the underground cyber-gangs or a network security professional or other administrator turning to the ‘dark side’ or a combination of both. In fact, it is common to call law-abiding security professionals “White Hats”, with some morally-challenged, but generally good-intending people termed “Grey Hats”. The clear delineation here is intent—Black Hats are in it for malicious purposes, often profit. This hat color scheme gets its roots from old Western movies and early black and white Western TV shows. In these shows, the bad guys always wore black hats, and the good guys wore white hats. Roles and morality were clearly defined. In the real world, this distinction is more muddled.

Black Hats will slowly and patiently troll through networks, looking for vulnerabilities. Generally, they will have done their homework very thoroughly and will have a good idea of the network layout and systems present before ever sending a single packet directly against your network—their phase one preparation is meticulous. A surprising amount of data can be gleaned from simple tools like the WhoIs database and Google or other web search engines for free. Mail lists and newsgroups when data-mined for domains from a target can reveal a lot of good detail about what systems and servers are used by seeing network and system admins asking questions on how to solve server problems or configure devices for their networks, not to mention the wealth of information gleaned for social engineering. Names, titles, phone numbers, and addresses—it’s all there to use by a skilled impersonator to make a few phone calls and obtain domain information and even usernames and sometimes passwords!

Notes from the Underground…

Social Engineering

Social engineering is the term used to describe the process by which hackers obtain technical information without using a computer directly to do so. Social engineering is essentially conning someone to provide you with useful information that they should not—whether it’s something
obviously important like usernames and passwords or something seemingly innocuous like the name of a network administrator or his phone number.

With a few simple pieces of valid information, some good voice acting and proper forethought, a hacker could convince you over the phone that he or she was a new security engineer, and that CEO is in a huff and needs the password changed now because he can’t get to his e-mail or someone’s going to get fired. “And that new password is what now? He needs to know it so we can login and check it…”

Be sure to train your staff, including receptionists who answer public queries, to safeguard information to keep it out of the hands of hackers. Have authentication mechanisms to prevent impersonation.

The recon portion of the attack for a cautious Black Hat may last weeks or even months—painstakingly piecing together a coherent map of your network. When the decision to move to phase two and actively attack is finally made, the attack is quiet, slight, and subtle. They will avoid causing a crash of any services if they can help it, and will move slowly through the network, trying to avoid IDPs and other traffic logging devices. Phase three consolidation is also very common, including patching the system from further vulnerability, as they do not want some Script Kiddie coming in behind them and ruining their carefully laid plans.

A Black Hat’s motivation is usually a strong desire to access your data—credit cards, bank accounts, social security numbers, usernames, and passwords. Other times it may be for petty revenge for perceived wrongs. Or they may want to figure out a way to divert your traffic to websites they control, so they can dupe users into providing these critical pieces of information to them—this technique is known as phishing (pronounced like fishing, but with a twist). Some phishing attacks will merely copy your website to their own, and entice people to the site with a list of e-mails they may have lifted off your mail or database server. Sometimes malware authors will also compromise websites in a manner similar to a Script Kiddie web defacement, but instead of modifying the content on the site, they merely add additional files to it. This allows them to use the website itself as an infection vector for all who visit the site by adding a malicious JPEG file, Trojan horse binary, or other script into an otherwise innocuous website—even one protected by encryption (via Hypertext Transfer Protocol Secure, known simply as HTTPS).

Defense against these sorts of attacks requires good network security design as well as good security policy design and enforcement. Training employees, especially IT and receptionist or other public-facing employees, about social engi-
neering awareness and proper information control policy is paramount. For the network itself, proper isolation of critical databases and other stores of important data, combined with monitoring and logging systems that are unreachable from potentially compromised servers is key. Following up on suspicious activity is also important.

**Worms, Viruses, and other Automated Malware**

Mentioned in the Notes from the Underground sidebar, the concept for self-propagating programs is nothing new, but the practical application has only been around for the last 15 to 20 years. Since the origins of the Internet are well over 40 years old now, this is significant. Indeed, it’s in the last two to three years that malware has taken a rather nasty turn for the worst, and there’s a good reason behind it.

Early worms were merely proofs-of-concept, either a “See what I can do” or some sort of glimpse at a Cyber Pearl Harbor or Internet Armageddon, and rarely had any purposefully malicious payload. This didn’t keep them from being major nuisances that cost companies millions of dollars year after year. But then some of the more advanced hacking groups started getting the idea that a large group of computers under a single organization’s complete control might be a fun thing to have. And the concept of a zombie army was born.

**Notes from the Underground**

**Are You a Zombie?**

The majority of machines compromised to make a zombie army are unprotected home users, directly connected to the Internet through DSL lines or cable modems. A recent study showed that while 60% of home Internet users surveyed felt they were safe from hackers, only 33% of them had some sort of firewall. Of that minority of Internet users with firewalls, 72% were found to be misconfigured. This means less than 10% of home Internet users are properly protected from attack!

Furthermore, of the users who had wireless access in their homes, 38% of them used no encryption, and the other 62% who did, used wireless encryption schemes with known security flaws that could be exploited to obtain the decryption key. Essentially, every person surveyed
Zombies, sometimes also referred to as Bots (a group of Bots is a Bot-net), are essentially Trojan horses left by a self-propagating worm. These nasty bits of code generally phone home to either an IRC channel or other listening post system and report their readiness to accept commands. Underground hacker groups will work hard to compromise as many machines as they can to build up the number of systems under their command. Bot-nets comprised of hundreds to tens of thousands of machines have been recorded. Typically these groups use the bots to flood target servers with packets, causing a Denial of Service (DoS) attack from multiple points, creating a Distributed Denial of Service (DDoS) attack. Nuking a person or site you didn’t like is fun for these people. But the fun didn’t last long.

Once the reality of a multi-thousand node anonymous, controllable network was created, it was inevitable that economics would enter the picture, and zombie armies were sold to the highest bidder—typically spammers and organized crime. Spammers use these bots to relay spam through, so that ISPs (Internet service providers) couldn’t track them back to the original spammer and shut down their connection. This became so important to spammers that eventually they were contracting ethically challenged programmers to write worms for them with specific features such as mail relay and competitor Trojan horse removal. Agobot, MyDoom, and SoBig are examples of these kinds of worms. Organized crime realizes the simplicity of a cyber-shakedown and extorts high-value transaction networks such as online gambling sites for protection from DDoS attack by bot-nets under the mob’s control.

Protection from these tenacious binaries requires defense-in-depth (security checkpoints at multiple points within your network) as well as a comprehensive defense solution (flood control, access control, and application layer inspection). Many of the Script Kiddie defense methods will also work against most worms, since the target identification logic in these worms is generally limited—phase one recon is usually just a SYN to a potentially vulnerable port. This is because there is only so much space for all that the worm needs to do—scanning, connecting, protocol negotiation, overflow method, shellcode, and propagation method, not to mention the backdoor Trojan. Most worms pick targets completely at random and try a variety of attacks against it, whether it’s a valid target for the attack or not. To solve the complexity problem, many Trojans are now...
split into two or more parts—a small, simple propagating worm with a file transfer stub, and a second stage full-featured Trojan horse, with the phone home, e-mail spamming, etc. The first stage attacks and infects, then loads the second stage for the heavy lifting. This allows for an effective phase three consolidation.

Information obtained by Honeypot Networks (systems designed to detect attacks) shows that the average life expectancy of a freshly installed Windows system without patches connected directly to the Internet without a firewall or other protection is approximately 20 minutes. On some broadband or dial-up connections it can take 30 minutes or longer to download the correct patches to prevent compromise by these automated attack programs. Using the Internet unprotected is a race you can’t win.

Notes from the Underground…

Multi-Vector Malware and the People Who Pay For It

Hacking (the term as used by the media for unauthorized access) is as old as Computer Science itself. Early on, it was mostly innocent pranks, or for learning and exploring. And while concepts for self-replicating programs were bantered around as early as 1949, the first practical viruses did not appear until the early 1980’s.

These early malicious software (or malware) applications generally required a user’s interaction to spread—a mouse button clicked, a file open, a disk inserted. By the late 1980’s, however, fully automated self-replicating software, generally known as worms, were finally realized. These programs would detect, attack, infect, and restart all over again on the new victim without any human interaction. The earliest worms, such as the Morris Worm in 1988, had no purposeful malicious intent, but due to programming errors and other unconsidered circumstances, it still caused a lot of problems.

The earliest worms and hacking attacks targeted a single known vulnerability, generally on a single computing platform. Code Red is a classic example—it targeted only Microsoft Windows Web servers running Internet Information Server (IIS), and specifically a single flaw in the way IIS handled ISAPI (Internet Server Application Programming Interface) extensions. And while they did significant damage, a single flaw on a
single machine tends to confine the attack to a defined area, with a known, specific defense.

Unfortunately, this is no longer the case. Malware is now very complex, and the motivations for malware have changed with it. Early malware was limited to mostly pranks—file deletion, web defacement, CD tray opening, and so on. Later, when commerce came to the Web, and valuable data, like credit card numbers and other personal information were now on-line and potentially vulnerable, greed became a factor in why and how malware authors wrote their code. Recently the culprits are spammers with significant financial clout, who pay programmers to add certain features to their malware, so that spam (unsolicited e-mail), spim (unsolicited instant messages) and spyware can be spread for fun and profit.

NetSky, MyDoom, and Agobot are the newest breeds of these super-worms. New versions come out almost weekly, and certainly after any new major vulnerability announcement. They don’t target just one vulnerability on one platform—they are multi-vector, self-propagating infectors, and they’ll stop at nothing to infiltrate your network. Most exploit at least four different vulnerabilities, as well as brute force login algorithms. These worms even attack each other—NetSky and MyDoom both remove other Trojan horses as well as antivirus and other security programs. A variant of Agobot attempts to overflow the FTP (File Transfer Protocol) server left behind by a Sasser worm infection as an infection vector.

Configuring SCREEN Settings

The SCREEN options on a NetScreen firewall are perhaps the oldest form of protection found on these firewalls. New options and features were added over time to address new threats present on the Internet. In the newer versions of ScreenOS (starting with ScreenOS 3.1, which was a limited-platform release—all devices supported this new feature in ScreenOS 4.0), these options are security-zone specific—each zone may have unique settings applied to it. For all options, these settings are applied as the inspected traffic externally enters the zone—that is, when the stream is read from the interface off the wire, not as it passes through the NetScreen and out another interface.

While NetScreen organizes these attacks by layers and protocols, it’s easier to talk about them more generically by their purpose. The two major functions of the SCREEN features are reconnaissance detection and Denial of Service protection. See Figure 10.1 for the ScreenOS version 5.1.0r1 SCREEN setting page.
Reconnaissance Detection

As mentioned earlier, an attacker will more than likely perform some initial recon-
nnaissance on your systems before launching an attack. Generally these methods are
benign, and so they are therefore easily lost in the clutter of normal traffic.

Port Scans and Sweeps

Port scanning, especially across multiple machines, is the simplest and most
common network reconnaissance method. A variety of tools, most notably NMap
(www.insecure.org/nmap/), perform port scanning as well as more advanced
system identification such as OS fingerprinting and service banner capture.

NetScreens can detect a single system being scanned for ports open (port
scan), a single port being scanned across multiple systems (port sweep), or a com-
bination of the two. Configuring NetScreen to detect these attacks can be done
via the IP Address Sweep Protection and Port Scan Protection options in the
Screening | Screens window. To the right-hand side of these options you’ll
find a user-definable threshold in microseconds. This value both indicates how
quickly ten probes must occur before the detection trips. The default value is
5000 microseconds, or 0.005 seconds. Some protocols can open up several ports
in rapid succession. If you find this triggering often from trusted machines that
you’ve verified have no malware running on them, you may need to adjust this threshold higher to weed out these false positives.

You’ll most commonly detect scans and sweeps from Script Kiddies or other automated, semi-intelligent attacks. More experienced Black Hats will scan more slowly, generally slow enough to avoid being detected by a firewall. This technique of sending port scanning packets infrequently over a long period of time is known as a slow scan.

TCP Protocol Manipulation

Other methods of scanning involve modifying TCP (Transmission Control Protocol) flags to invalid or improper settings. Many stateless routers that are pressed into service as rudimentary firewalls can detect established communications based upon the TCP ACK flag. Scanners will utilize this logic flaw and send ACK scans in which the packet sent will have the ACK bit set—this bypasses most ACL (access control list)-based packet filters, but thanks to the stateful inspection feature in ScreenOS, no TCP packet not matching an established session (created with a proper TCP three-way handshake) may pass. Other TCP flag tomfoolery that can be detected and blocked include:

- **SYN Fragment Protection**  Protects against initial SYN packets with the fragmentation bit also set. SYN packets generally have no data and therefore cannot be fragmented.

- **TCP Packet Without Flag Protection** Also called a NULL scan. This is an invalid flag configuration.

- **SYN and FIN Bits Set Protection** Sometimes called an open-close scan. SYN flags begin sessions, while FIN flags generally end them—a single packet that both requests and closes a session is anomalous.

- **FIN Bit With No ACK Bit in Flags Protection** Per RFC, when closing a connection with FIN, you need to acknowledge (ACK) the byte count you received. This detects a packet not matching that requirement.

IP Protocol Manipulation

Other, more obscure methods of network mapping involve detecting Internet Protocol (IP)-layer parameters. ScreenOS supports blocking these probes with a slew of IP option anomalies:

[www.syngress.com](http://www.syngress.com)
- Bad IP Option Protection
- IP Timestamp Option Detection
- IP Security Option Detection
- IP Stream Option Detection
- IP Record Route Option Detection
- IP Loose Source Route Option Detection
- IP Strict Source Route Option Detection
- IP Source Route Option Filter

If for some reason you have need for these services, generally you’ll already know about it. If these do not sound familiar to you, it’s a safe bet you don’t need them. Most of these activities have no valid use on a network and are generally safe to block.

Flood Attacks

Flooding is one of the oldest, yet still very popular methods of attack. The problem was, connection requests could come in over the network faster than most systems could properly handle them. Thanks to Moore’s Law, CPU capacity has outstripped network capacity, and this problem, while not completely mitigated, is considerably reduced. It’s still prudent to block these attacks as far out to the perimeter as possible, if for no other reason than to clean up the clutter and keep unnecessary traffic out of your network.

NetScreen offers three different flood protection queues based upon the protocol that they handle:

- ICMP (Internet Control Message Protocol)
- UDP (User Datagram Protocol)
- TCP (SYN flood)

ICMP is the most straightforward of the flood protections. A threshold value of total ICMP packets per second (from all IP addresses) is set, and if that threshold (default of 1000 p/s) is exceeded in a particular second, the remainder of the ICMP packets for that second, as well as all of the ICMP packets for the next second, are dropped. Furthermore, sessions are not made for dropped packets.
UDP flood protection is essentially the same as ICMP flood protection, but uses a separate threshold and queue. It uses a threshold value (default of 1000 p/s) that if exceeded, drops all remaining UDP packets from all IP addresses for that second as well as the next.

TCP SYN flood protection is the most complicated flood protection, due to the NetScreen’s ability to proxy the three-way handshake. It allows for a variety of different threshold settings:

- **Attack Threshold** This controls how many packets per second must arrive at a single IP/port pair before the NetScreen begins proxying SYNs. Any SYNs above this threshold for the remainder of that second are proxied, until the proxy queue is full.

- **Alarm Threshold** This controls when an alarm should be logged for a potential SYN flood. This number should be lower than your attack threshold – it is a warning that you could be having trouble.

- **Source Threshold** This threshold is separate from the attack threshold where the total number of SYN packets from a particular source IP are counted. The source threshold setting is very useful for isolating scanning worm infections on end user systems. Set this number relatively low on your user-space segment (see “Zone Isolation” in the “Applying Best Practices” section) and notice that when an infected host tries to open up 100 new connections per second to other targets, attempting to infect them, this feature will throttle that attack to a manageable level.

- **Destination Threshold** This threshold is also separate from the attack threshold, and is similar to source threshold, except the number of sessions compared is for a particular destination IP. This is also measured in packets per second, and should be used for servers or other important machines to keep the overall level of new TCP connections to a set maximum.

- **Timeout** This is how long a SYN should be kept in the proxy queue before being flushed as an invalid connection request. Its default setting is 20 seconds, which is very generous. I would recommend something lower, perhaps as low as 5 to 7 seconds, depending on the latency of your network. Keep in mind that any properly negotiated three-way handshake will automatically clear the entry from the queue.
- **Queue Size** Specifies the number of SYNs that can be proxied and monitored before dropping new SYNs. A larger number uses more memory (since it needs to remember the IP address and port number of the session requested), and also takes longer to scan the queue for completed three-way handshakes, resulting in a higher initial connection latency.

There is a special case when a NetScreen is in *transparent mode* and it needs to proxy the SYN for a session, but the destination MAC (Media Access Control) address hasn’t been learned yet, and isn’t in the NetScreen’s ARP (Address Resolution Protocol) table. This could occur on a large layer 2 network where the destination MAC has aged off of the NetScreen device, or it could be that the destination IP doesn’t exist, and therefore the MAC cannot be learned. The **Drop Unknown MAC** option allows you to set the behavior of the NetScreen device when this situation occurs. By default, NetScreen will pass a packet with an unknown destination MAC and ***not*** proxy it. With this option set, NetScreen will drop the packet instead.

**Protocol Attacks**

In addition to flood attacks, the SCREEN functions can also block protocol-specific attacks. These are generally legacy attacks—new attacks are blocked with Deep Inspection, discussed below. Protocols and attacks covered:

- **HTTP (Hypertext Transfer Protocol)** Allows the blocking of Java and ActiveX code, as well as ZIP and EXE file downloads.
- **Windows** Allows the blocking of the classic WinNuke (malformed data to port 139) attack.
- **ICMP protocol attacks** Allows the blocking of the *ping of death* (fragment boundary overflow attack), ICMP fragments, and large ICMP packets.
- **TCP protocol attacks** Allows the blocking of *teardrop* (another fragment boundary overflow attack) and *land* (source and destination IP and port are the same) attacks.
Applying Deep Inspection

Juniper Networks’ line of NetScreen firewall products have evolved with security requirements to consistently keep up to date with threats that plague network administrators. Deep Inspection is the newest and most comprehensive coverage to date, with even more protocols coming soon. DI takes network security all the way up the stack to the application layer, inspecting traffic as it would be interpreted by the end host application. This answers the problem vexing many Administrators who are used to solving security at layer 3 and 4—“How do I defend from attacks when I need to leave port 80 open?”

Deep Inspection is a subset of Juniper Networks’ award-winning NetScreen Intrusion Detection and Prevention, with support for protocols typically considered to be Internet-facing—HTTP, SMTP (Simple Mail Transfer Protocol), DNS (Domain Name System), POP3 (Post Office Protocol v3), FTP, and IMAP (Internet Message Access Protocol). Support for MS-RPC (Microsoft Remote Procedure Call) and SMB (Server Message Block) is available in ScreenOS 5.1, though generally these protocols do not traverse the Internet legitimately.

DI examines all incoming packets and assigns them a session (or in the case of stateless protocols such as UDP or ICMP, a pseudo-session). It reassembles fragments, rearranges out-of-order frames, and creates data streams from these packets (errors from overlapping fragments and other tomfoolery are handled by IP layer protocol anomaly inspectors). These streams are then handed off to protocol-specific inspection engines, called Q modules, which further inspect and parse the stream into protocol-specific elements (called contexts) for signature matching. Protocol-specific anomalies are also detected at this stage. For example, DNS requests are matched to DNS replies to ensure that the answer matches the question—this prevents DNS poisoning.

These contexts are what make DI so accurate. With this level of parsing, a signature writer can specify a more targeted portion of the data stream for inspection—this also has the added benefit of increased performance, since only the relevant portion of the stream is inspected for attacks. Take this hypothetical situation:

Say you have a simple, stateless, in-line Intrusion Detection System (IDS) monitoring your network. A new vulnerability (in this case, a secret backdoor left by the developer) is discovered in the mail server, whereby if an e-mail arrives from a specific user (littlepig@bigbadwolf.com for this example), in addition to forwarding the message to the recipient, it also takes whatever attachments are
included with the message and attempts to execute them as programs. Being limited to this stateless IDS, you write a signature that says, “If you see the pattern ‘littlepig@bigbadwolf.com’ go over TCP port 25 then block it”. You then send an e-mail out to all your users, informing them to report any suspicious e-mails that they receive from that address. Later that day, you ask a coworker if he’d received any of the e-mails you were talking about earlier. He gets very confused and asks you what you’re talking about. When you start talking about little pigs and big bad wolves, he gets a funny look on his face and mumbles something about being late to a meeting and hurries off. It’s only after you check your IDS logs that you realize that no one received your e-mail because your own IDS blocked it! It detected the string match based upon the data in your e-mail, and took what it thought was appropriate action.

Take that sample situation and instead use DI’s SMTP-From context, and put your string match there. The same e-mail message you sent out to your users would pass through DI unmolested, because it knows the difference between the SMTP command phase of the session and the SMTP data phase of the session. Your e-mail had a matching string in the data portion of the stream, which isn’t where the vulnerability lies, so DI ignores it. Later on that day, when a hacker tries to test your security, DI detects the match in the SMTP command phase (specifically, in a SMTP From command) and blocks the message from arriving on your mail server.

But what if you missed the memo? Security issues come up every day, and it’s more than a full-time job just to keep abreast of all the details. Is this particular security announcement relevant to your network? Do you run a vulnerable version on any of your servers? Are these servers accessible from the Internet? How does the attack work? What kind of regular expression (RegEx) would detect it? Would your signature trigger on non-malicious traffic (called a false positive) and block legitimate traffic? Would your signature fail to trigger on malicious traffic (called a false negative) and let attacks through? Did you leave the garage door open this morning?

Since not everyone can be a full-time security researcher, Juniper Networks has the Juniper Engineering Security Team do research for you. With a valid subscription, you can receive a well-stocked signature pack as well as regular and periodic updates as new vulnerabilities are announced. Medium through critical (as defined by CERT/CC – www.cert.org/) severity issues, when possible, are covered by DI.
Getting the Database

NetScreen firewall products need a valid DI license key before DI is used. Your Juniper Customer Service Representative can assist you with obtaining one, as well as helping out if there are problems with loading the key on to the device. The device may need a reboot when adding a new license key. Updating subscriptions to already-activated features generally do not require a reboot. Once the license is on the device, you are ready to load your database.

NOTE

All license keys are tied to the serial number of the device for which the key was granted—there is no such thing as a universal key. Trying to load a license key created for one unit and trying to load it on another unit—even the same model—will fail. Additionally, if you ever need to return your unit for replacement under the RMA policy, the new device you receive needs to have new keys issued for it. Support generally will handle this for you automatically, but it’s something to check in case something doesn’t work with the new unit. Many configuration settings are hidden until a valid license key is loaded to activate those features—loading a configuration from an entitled firewall onto another firewall without entitlements could cause the inactive portions of the config to be dropped.

TIP

With ScreenOS 5.1 you can also use the Retrieve Subscriptions Now button in the Configuration | Update | ScreenOS/Keys WebUI page for the device to automatically retrieve keys assigned to it from Juniper Networks’ entitlement server. Note this requires the firewall to have Web access to the Internet in order to connect to the server. For devices that cannot reach the entitlement server directly, the key file must be loaded on the device by hand.

The database file is a pre-compiled binary database file and can be downloaded by the device directly from the Juniper website if the firewall is attached.
to the Internet and firewall policies are in place to permit it to do so. This can be configured to automatically occur on a set schedule so that you’ll never miss an update. You can also force an update from the Internet via the WebUI.

If the firewall is on a private network, or if access is restricted, the file can be manually downloaded from the Internet, and then either placed on an internal Web server for your firewalls to download from (the URL that specifies the location of the attack file is mostly configurable) or it can be loaded by hand by either using the WebUI via HTTP upload direct from the browser or the command line interface (CLI) via TFTP (Thin File Transfer Protocol) file transfer. For the latter method, a TFTP server is required.

Configuring the Firewall for Automatic DI Updates

One of the more handy features of DI is its ability to automatically check for new signature packs and download them as necessary without user intervention. Configuration for this is easy.

**NOTE**

Remember that the device has to have HTTPS access to the Internet in order to automatically update. Read below on how to perform a manual update if HTTPS access is not possible.

Using the WebUI, access **Configuration | Update | Attack Signature**. Figure 10.2 shows this screen as shown on ScreenOS 5.1.0r1. The **Database Server** field is used to select the partial URL from which to download (the current default is still apparently [https://services.netscreen.com/restricted/sigupdates](https://services.netscreen.com/restricted/sigupdates), although since Juniper acquired NetScreen some time ago now, I expect this URL to be updated sometime soon). I say partial, because the latter half of the complete URL is hard-coded as /[model-name]/attacks.bin (for example, /ns500/attacks.bin). As of ScreenOS 5.1.0r1 and 5.0.0r7 (and earlier), there is a bug in which the model name is hard-coded to ns500, regardless of the actual platform type. If you’re looking to configure an internal DI update server, you can use HTTP or HTTPS, and be sure to put the newest attacks file in a subdirectory called /ns500 for the update to work.

The section below the URL entry line allows you three update modes: **None**, **Automatic Notification**, and **Automatic Update**. **None** turns auto-
update off, while Automatic Notification checks to see if there is a new file, but does not download and update; rather it puts an entry in the logs that an update is available. If you don’t check your logs often, but want to make sure the device always has the most current coverage, you can select Automatic Update, which checks for new signature updates and, when available, automatically downloads and installs them.

The remainder of the options for this screen are fairly self-explanatory and include settings for when the device will auto-update (daily, weekly, or monthly), and what time of day to update. There’s also a handy Update Now button that allows you to test your settings.

**Figure 10.2** Automatic DI Signature Update Settings

![Automatic DI Signature Update Settings](312_NetScr_10.qxd_11/24/04_4:41_PM_Page_398)

Loading the Database Manually

Sometimes due to architectural decisions, a firewall may not have direct access to the Internet. In these circumstances, an automatic update may not be possible. Manual updates require you to obtain the update by hand from another system connected to the Internet, then take that image and manually place it on the device in question.

Using the WebUI, manual loading is a straightforward affair. If you have your DI key properly installed, the Deep Inspection Signature Update field will
be available on the Configuration | Update | Attack Signature screen. This field is hidden if the DI key is not installed. Use the **Load File** field to enter the local path to the signature update file, or use the **Browse** button to locate and select the file location. Once you have specified or selected the file location, click **OK** to update the device from the local file.

To perform this action via the CLI, you’ll need to download the signature update to a TFTP server. The syntax for the command is **save attack-db from tftp [server-IP] [path/filename] to flash**. If successful, you’ll see a string of dots generated across your console as TFTP packets arrive. Exclamation points (!) mean packet loss or other network error. Missed packets from an otherwise successful stream are present, but if the NetScreen can’t connect to your TFTP server, you’ll receive a number of beginning exclamation points followed by a TFTP timeout error.

**NOTE**

While your NetScreen device is signed up for subscriptions, you can update the device as many times as you like. Once your subscription has expired, you’ll be ineligible for new updates, but your existing signature pack will continue to work as before. You’ll also still be able to create your own custom signatures even if your subscription has run out.

**Using Attack Objects**

NetScreen-supplied attack objects are organized into groups based upon three criteria: protocol, severity, and type. For ScreenOS 5.0, the only valid severity levels were critical, high, and medium. Beginning with ScreenOS 5.1, the new severity levels of low and info are included. For ScreenOS 5.0, only six protocols were supported: HTTP, FTP, DNS, POP3, SMTP, and IMAP. Beginning with ScreenOS 5.1, this protocol list has expanded to include SMB, MS-RPC, NetBIOS, Gnutella (a popular peer-to-peer file sharing protocol), as well as several instant messaging protocols. For type, there are signatures and anomalies. Signatures are specialized regular expression pattern matching strings applied to contexts that then match malicious or other unwanted traffic in network flows. Protocol anomalies are protocol-specific functions that ensure that the flow adheres to protocol standards or other settings.
Using Attack Groups

Attacks cannot be used individually in a policy—they must be assigned to a group, even if that group contains just a single entry. If a predefined group has entries in it that you do not want to use, you may deactivate them from the group by accessing the Objects | Attacks | Predefined Groups window and clicking View for the group you wish to edit. You will see a listing of all attacks included in the group. The right-hand column has a checkbox next to each entry. To remove an entry from inspection by the group, remove the check mark from its checkbox.

Changing active entry settings within a group is a global action that affects the entire device for all policies that use that group. Also note that this does not remove the entry from inspection, it only removes it for the purposes of action against the event—there is no performance improvement for removing signatures from a group. Likewise, there is no performance impact for using DI groups over and over again in different policies. When DI is on, it’s ON, and when it’s off, it’s OFF. The first time you use DI in a policy, DI inspection automatically turns on. When it is removed from every single policy, DI automatically turns off.

Enabling Deep Inspection with a Policy using the WebUI

In order to use DI, you must first create an access policy. Figure 10.3 shows a ScreenOS 5.1 policy crafting window in the WebUI. Create appropriate entries for Source Address, Destination Address, and Service. To choose which DI groups will inspect the traffic through sessions matching this policy, click on the Deep Inspection button.
Figure 10.4 shows the Deep Inspection Configuration window.

**Figure 10.4** Policy Deep Inspection Configuration

Here you’ll find an unsorted, unfiltered drop list of all available Deep Inspection groups. In DI 5.1, you’ll have a pick list of at least 58 items. Following that is a drop-down list that allows you to select which **Action** to perform on the
selected group, as well as a Log option checkbox. Below this you’ll find a table showing the **Currently Defined Attack Groups** assigned to this policy. Different groups within a policy can have different action and logging settings. This is useful, since there can be only one policy that matches traffic between two hosts on a port—multiple duplicate policies are not permitted. As in ScreenOS, the first matched policy for a connection is used. Click on the **Add** button to add the selected group with the selected action and log setting to the defined attacks table. Click the **OK** button when you are finished. These DI options will be applied to your policy. Click on the **OK** button in the main policy editing window to apply your changes to the policy. The resulting DI-enabled policy has the DI inspection magnifying glass icon in the **Actions** column of the policy list. Policy ID 2 in Figure 10.5 has Deep Inspection enabled, while policy ID 3 has both Deep Inspection and Trend Micro antivirus enabled.

**Figure 10.5 Policy Listing in the WebUI**

---

**Enabling Deep Inspection with a Policy using the CLI**

Creating a policy that inspects traffic using many attack groups is a major chore in the WebUI. My personal recommendation is to create the initial policy in the way you feel comfortable—either WebUI or CLI—and then for bulk DI inspection configuration, use the CLI. You’ll find this is a vastly superior method.
If you are managing multiple firewalls and policies, using NetScreen Security Manager (NSM) will make your task many times easier – define an attack and policy once, then simply specify which devices to apply it to.

To get a listing of which attack groups are available, use the **get attack group sort-by name** command:

```
nsSgt-> get attack group sort-by name
Total number of attack groups is 58
```

You should get results similar to the following:

Table 10.1  ScreenOS 5.1 CLI attack groups sample listing

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRITICAL:DNS:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:DNS:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:FTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:HTTP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:HTTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:MSRPC:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:MSRPC:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:POP3:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:SMB:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:SMB:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:SMB:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>CRITICAL:SMTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:DNS:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:DNS:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:FTP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:FTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:HTTP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:HTTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:IMAP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:IMAP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:MSRPC:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:NBDS:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
</tbody>
</table>

Continued
## Table 10.1  ScreenOS 5.1 CLI attack groups sample listing

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH:NBNAME:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:POP3:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:POP3:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:SMB:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:SMB:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:SMTP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>HIGH:SMTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:AIM:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:DNS:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:DNS:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:GNUTELLA:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:HTTP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:HTTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:MSN:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:MSN:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:SMB:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:SMB:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:SMTP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:SMTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:YMSG:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>INFO:YMSG:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>LOW:FTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>LOW:HTTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>LOW:SMTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:DNS:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:DNS:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:FTP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:FTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:HTTP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:HTTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:IMAP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:MSRPC:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:NBDS:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
</tbody>
</table>
Table 10.1 ScreenOS 5.1 CLI attack groups sample listing

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEDIUM:NBNAME:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:POP3:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:POP3:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:SMB:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:SMB:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:SMTP:ANOM</td>
<td>group</td>
<td>pre-defined</td>
</tr>
<tr>
<td>MEDIUM:SMTP:SIGS</td>
<td>group</td>
<td>pre-defined</td>
</tr>
</tbody>
</table>

Total number of attack groups is 58

Cut and paste this output to a text file for handy reference. Now let's edit this policy to add DI. From the command line, type `set policy id [x]` where [x] is the ID number of the policy we’re editing, and press Return. This puts us unto policy-edit mode—you’ll notice the command prompt has changed, and added a `(policy:x)` to the end of the prompt. This lets us know we’re editing policy ‘x’.

All subsequent commands apply only to our current policy until we use the `exit` command to end policy-edit mode.

From here, it’s a simple matter of adding `set attack [attack-group-name] action [action]` to add attack [attack-group-name] to the policy with action of [action]. To enable logging of this group, we need to add another set attack command, this time with logging instead of an action command, like so: `set attack [attack-group-name] logging`. Once you’ve added all your attack groups, remember to use the `exit` command.

Explanation of Deep Inspection

Contexts and Regular Expressions

After using the built-in signatures, I’m sure you’re eager to write a few of your own. Before we jump into how to write a signature, we need to cover some basics on how DI looks for patterns, and how to write instructions to make it recognize bad traffic.

As mentioned earlier, Deep Inspection uses contexts to examine relevant portions of network streams for content. In ScreenOS 5.0, a very limited set of DI contexts were exposed to end users. Table 10.1 shows the only contexts a user
could use to make a signature; many more were available to Juniper signature writers.

**Table 10.2** ScreenOS 5.0.0 User-Accessible Contexts

<table>
<thead>
<tr>
<th>Deep Inspection Protocols</th>
<th>End User Contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP</td>
<td>ftp-command, ftp-username</td>
</tr>
<tr>
<td>HTTP</td>
<td>http-url-parsed</td>
</tr>
<tr>
<td>SMTP</td>
<td>smtp-from, smtp-header-from, smtp-header-to, smtp-rcpt</td>
</tr>
</tbody>
</table>

With ScreenOS 5.1, a whole new slew of protocols with new contexts are available, as well as additional contexts for existing protocols (see Table 10.2).

**Table 10.3** ScreenOS 5.1.0 User-Accessible Contexts

<table>
<thead>
<tr>
<th>Deep Inspection Protocols</th>
<th>End User Contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOL Instant Messenger (AIM)</td>
<td>aim-chat-room-desc, aim-chat-room-name, aim-get-file, aim-nick-name, aim-put-file, aim-screen-name</td>
</tr>
<tr>
<td>DNS</td>
<td>dns-cname</td>
</tr>
<tr>
<td>FTP</td>
<td>ftp-command, ftp-password, ftp-path-name, ftp-username</td>
</tr>
<tr>
<td>Gnutella Peer-to-Peer Protocol</td>
<td>gnutella-http-get-filename</td>
</tr>
<tr>
<td>IMAP</td>
<td>imap-authenticate, imap-login, imap-mailbox, imap-user</td>
</tr>
<tr>
<td>Microsoft Network Chat (MSN)</td>
<td>msn-display-name, msn-get-file, msn-put-file, msn-sign-in-name</td>
</tr>
<tr>
<td>Post Office Protocol ver 3 (POP3):</td>
<td>pop3-auth, pop3-header-from, pop3-header-line, pop3-header-subject, pop3-header-to, pop3-mime-content-filename, pop3-user</td>
</tr>
<tr>
<td>Server Message Block/Common Internet File System (SMB/CIFS)</td>
<td>smb-account-name, smb-connect-path, smb-connect-service, smb-copy-filename, smb-delete-filename, smb-open-filename</td>
</tr>
</tbody>
</table>

[Continued]
Table 10.3  ScreenOS 5.1.0 User-Accessible Contexts

<table>
<thead>
<tr>
<th>Deep Inspection Protocols</th>
<th>End User Contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Mail Transfer Protocol (SMTP)</td>
<td>smtp-from, smtp-header-from, smtp-header-line, smtp-header-subject, smtp-header-to, smtp-mime-content-file-name, smtp-rcpt</td>
</tr>
</tbody>
</table>

There’s not enough space to cover all of these contexts in detail, so in the next section we’re going to hit the highlights and give some examples of what you can do with some of the more popular contexts. A complete context reference can be found in the Juniper documentation.

Before we can talk about writing signatures in contexts, we need to cover the pattern matching syntax, also known as DFA (Deterministic Finite Automaton) syntax (see the “Deep Inspection Search Algorithm” sidebar below). NetScreen DFA syntax is similar to regular expression syntax, but not quite the same. Below we’ll cover the basics of how NetScreens match patterns.

The most straightforward way of looking for data would be an exact match. For example, to find the exact byte-pattern of root in a context, we would simply type root. Note that if the context presents any additional data, like rooter or, if the capitalization does not match, like Root or rOoT, then this simple match string will not match.

In order to insert special matching commands within a search string, the commands have to be identified as commands instead of just more matching text. This command delineation method is commonly known as escaping. In DFA syntax, commands are identified by a preceding backslash (\).

Sometimes, an exact string match is what you want. Most often, however, you want to detect variations and permutations of strings. For a case-insensitive match of alphabetic characters, enclose the string within escaped square brackets. Our earlier search for root with case-insensitive added would be \[root\]. Failing to close the case-insensitive range with an ending delimiter will cause the signature to not work.

This is useful, but what if we need to match this string at the beginning of the stream, but more information comes after it (such as in our rooter example...
above)? For this, we turn to our good friend, dot-star. The dot is used to match any one-byte value (in order to match a literal dot, it must be escaped, like so: \.). Star means zero or more of the previous match (again, to match a literal asterisk, it must be escaped like so: \*). Put these two elements, dot and star, together, and it will match zero or more of anything, which is quite handy. For example, \[root\].* matches Rooter and rooTMan, but not iamroot. For that last match, a dot-star at the beginning is the trick. For example, .*\[root\] matches nicely, as well as .*\[root\].*, which will also match IamRootMan. Many Juniper-authored signatures work exactly this way.

**NOTE**

The dot-star implementation used by Juniper’s IDP and DI is different from the common RegEx implementation. Standard (java/perl/grep) RegEx treats .* as a greedy match; if you put dot-star at the beginning it will always match. Juniper’s implementation is a bit more intuitive, but may surprise someone who is already familiar with using a posix regexp.

While these work great for ASCII character matches, many protocols use non-ASCII bytes. There are two ways to match arbitrary binary data in DI—hexadecimal (hex) and octal representations. For hex, DFA uses an escaped X (for heX) while for Octal, DFA uses an escaped zero (0), which represents the letter O in Octal. Another fundamental difference between these two methods is that an octal match always represents a single byte (so the maximum value is \0377, two bytes would be \0377\0377), while a hex match always represents one or more bytes with \x delimiting the start and end of the range of characters to be evaluated as hex (that is, \xff\x or \x0123456789abcdef\x). White space within the hex range is ignored, so you can space out your match characters by nybbles, bytes, words, etc. For example, you can enter \x 0123456789abcdef \x = \x 01 23 45 67 89 AB CD EF \x = \x 0123 4567 89AB CDEF \x. Failing to close a hex range with an ending delimiter will cause the signature to not work.

There are times when an attack will have one or more methods for gathering the same results, or perhaps you want to combine similar signatures into a single entry. In order to define elements of a match string, parentheses are used, but they are not escaped. To use literal parentheses in a match, you must escape them.
like so: \( \) or use their ASCII hex or octal values. Don’t confuse escaped parentheses with case-insensitive matching brackets that must be escaped in order to work so that they are not misinterpreted as a character class (see below).

When selecting one or the other of a series of options, we use the pipe \( | \) character for an OR operator. That is, match A or B using the entry \((A|B)\). Several ORs can be chained together—any one of them will match: \((A|BC|DEF)\). Note that they need not be a single byte, nor have the same amount of bytes.

There are times when you might want to match a large range of characters that would make ORing them all together entirely impractical. For example, ‘all capital letters’ would be \((A|B|C|D|E|F|G\) and so on until \([X|Y|Z]\) which is way too long, and makes reading difficult. To solve this dilemma, we have the character class feature. Character classes use unescaped brackets with a list of characters or a single character range to match on. \([A-Z]\) would solve our ‘all capital letters’ problem. For an arbitrary character class, merely add the characters within bracket in any order: \([ABCcba]\). Character classes also allow for octal codes for non-printable byte value match: \([\000-\017]\) or \([\011\013\020]\) etc.

Yet another use of the character class is to define values not to match. This is known as a negate character class. To negate a character class, merely place a caret \(^{\sim}\) as the first character inside the class. This will not match on a caret – it will negate the remainder of the character class. In ScreenOS 5.0, only a single character is allowed after the caret, while in ScreenOS 5.1, multiple characters are allowed, for example, \(^{\sim}A\) or \(^{\sim}123\). A common state-saver to the traditional dot-star in the middle of a match string (see the “Deep Inspection Search Algorithm” sidebar) is a not-space-star, or \(^{*}\) match string.

The question mark (?) makes the directly preceding match optional. For example, html? matches both html as well as just htm. This is also handy for using with parentheses to make an entire element optional. For example, super(duper)?man matches both superman and superduperman.

One final major matching syntax we’ll cover before jumping into signature writing is the unicode decoder. Many Windows protocols, like SMB, NetBIOS, and MS-RPC can use either traditional ASCII encoding or the new international-friendly unicode encoding. To convert ASCII to unicode, nulls (\000) are inserted after every character. Traditionally, it was very messy to make a string match both normal ASCII and ASCII in Unicode. For example, to match Windows would require W(\000)?i(\000)?n(\000)?d(\000)?e(\000)?w(\000)?s(\000)?, which is almost unreadable. The same match using the unicode decoder is merely \uWindows\u. Be sure to close your decoder with a second \u or the signature will not work.
Table 10.4 includes a quick reference to the match strings described above.

**Table 10.4 NetScreen Search String Syntax Summary**

<table>
<thead>
<tr>
<th>Match String</th>
<th>Usage Notes and Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>The dot character matches any one byte. When a literal dot match is needed, try escaping the dot like so: <code>www\.juniper\net</code>.</td>
</tr>
<tr>
<td>*</td>
<td>The asterisk (or star) matches zero or more of the preceding match. When a literal asterisk is needed, try escaping it: <code>\*</code></td>
</tr>
<tr>
<td>.*</td>
<td>Dot-star is a useful combination that matches zero or more of any characters. Place at the beginning of a match string to search anywhere in the context. Place at the end of a match string to ignore any additional data after the matched string. Remember that Juniper’s implementation of dot-star is not greedy.</td>
</tr>
<tr>
<td>+</td>
<td>The plus sign character matches one or more of the preceding match. For example, <code>AA+</code> matches <code>AAA</code>, but not <code>AA</code> or <code>AAB</code>.</td>
</tr>
<tr>
<td>?</td>
<td>The question mark makes the preceding character/element an optional match. For example, <code>html?</code> matches both <code>htm</code> and <code>html</code>.</td>
</tr>
<tr>
<td><code>\xAB CD\x</code></td>
<td>Matches hexadecimal values. Be sure to close your decoder with a second <code>\x</code> or the signature will not work.</td>
</tr>
<tr>
<td><code>\XABCD\X</code></td>
<td>Slash-zero matches a single byte of octal values <code>\000</code> through <code>\0377</code>. Permitted octal characters are <code>01234567</code> only.</td>
</tr>
<tr>
<td><code>\0oct</code></td>
<td>Case-insensitive search. Alphabetic characters are compared with both upper and lower case. For example, <code>\[dog\]</code> matches <code>dog</code>, <code>DOG</code>, <code>Dog</code>, <code>dOG</code>, and <code>DoG</code>.</td>
</tr>
<tr>
<td><code>( )</code></td>
<td>Parentheses are used to group portions of match strings into a single element. Parentheses are also useful with the pipe character for OR comparisons. For example, `AA(AA</td>
</tr>
<tr>
<td><code>[abc123]</code></td>
<td>Character class. Counts as a single byte that matches any symbol or symbol range inside. Cannot be used inside a case-insensitive search. For example, <code>\[abc\{def\}ghi\]</code> is illegal. Instead, try <code>\[abc\][def][ghi]</code>. Also note that multiple ranges are not allowed, such as <code>[a-zA-Z]</code>. Octal is also supported in order to define non-printable ranges.</td>
</tr>
</tbody>
</table>
Table 10.4 NetScreen Search String Syntax Summary

<table>
<thead>
<tr>
<th>Match String</th>
<th>Usage Notes and Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>[^abc]</td>
<td>Negated character class. Counts as any single byte that does not match the contents inside the brackets. Note that octal is still supported in negated character classes.</td>
</tr>
<tr>
<td>[^ ]</td>
<td>ScreenOS 5.0 DFA only supports a single character in a negate character class, while ScreenOS 5.1 DFA supports multiple characters.</td>
</tr>
<tr>
<td>[^\000]</td>
<td>New in ScreenOS 5.1 is the white-space character, or slash-s. This matches a single space or tab. For ScreenOS 5.0, in order to match the same value, an octal OR group was used: (011</td>
</tr>
<tr>
<td>\s</td>
<td>New in ScreenOS 5.1 is the unicode decoder, or slash-u. Be sure to close your decoder with a second \u or the signature will not work.</td>
</tr>
</tbody>
</table>


Tools and Traps

Deep Inspection Search Algorithm

NetScreen IDP and DI both use a method of searching traffic for malicious patterns very quickly using a technique known as a Deterministic Finite Automaton. A simple explanation of DFA is a tree of all possibilities the search is looking for, combined into a logical table where similar matches are grouped together and searched simultaneously. When a difference between two unique patterns occurs along the line, the search line forks and each subsequent possibility then gets its own line. The total unique search lines these forks create are known as states.

A DFA with a large number of states takes significantly longer to parse though to find a match. Regular expression symbols that generate a large number of states are the wildcard symbols ., *, and +, and the conditional symbol ?. When placed in the middle of a match, they can expand the number of states exponentially, severely impacting perfor-
performance and memory. Use these symbols sparingly in the middle of your signatures. Using them at the beginning or end of your signature does not add states and is actually a handy way to scan for a match where the beginning of the stream of information to match is unknown or variable.

Creating Your Own Signatures

Now that we’ve covered the two major aspects to signature creation – contexts and syntax – let’s put them together and write a few signatures! This section will cover a few of the more popular contexts with some RegEx usage on how to get the most out of them.

To make a new custom signature, access Objects | Attacks | Custom, and click the New button. This will open the signature editor window, which has 5 fields; Attack Name, Attack Context, Attack Severity, Attack Pattern, and Negate. Also note that custom signature names must start with the string CS (for custom signature). We’ll be using this window as we experiment with some of the more common contexts below.

HTTP is the most common protocol, and by far the highest bandwidth consumer. Adding new HTTP signatures impacts the performance of this already heavily burdened protocol, so add new signatures here with care, and try to avoid high-state wildcards (see the “Deep Inspection Search Algorithm” sidebar above) in the middle of signatures and after common matching strings.

To understand how the HTTP contexts work, we need to first examine the HTTP protocol itself. HTTP is a stateless client-server protocol, where a server generally supplies files based upon requests by the client. In addition to the file transfer itself, several protocol-related data exchanges occur, mostly at the beginning, before the actual file transfer – these are known as HTTP headers (inspected by the http-header-user-agent and http-authorization contexts). The client request itself is called a Uniform Resource Locator, or URL. The URL itself is generally broken down into two elements: the path/file and the parameters/variables.

The path and file includes all characters after the request verb but before the question mark, exclusive. The parameters (also called variables) include everything after the question mark, also exclusive. DI has an http-url-parsed context as well as an http-variable-parsed context. These contexts take any kind of URL obfuscating encoding and parse it as it would be by the end server and then apply the signature against the result. This allows us to write a nice, clean URL signature without worrying about encoding schemes or other kinds of IDS evasion techniques. If such
encoding attacks are what you’re looking for, there’s also the unparsed **http-url** context for just the URL, or **http-request** for the entire request, completely unparsed. Let’s try some practical examples of these contexts using the sample exchange below:

The client requests:
GET /etc/pass%77d?bar=yes HTTP/1.1
User-Agent: HappyBrowser v1.1
Host: www.foo.com
Authorization: Basic dXNlcjpwYXNzd29yZAo=

The server responds:
HTTP/1.1 200 OK
Date: Sat, 25 Dec 2004 00:00:01 GMT
Etc…

Notice the %77 in the URL? That decodes to an ASCII w, making the path /etc/passwd; someone was trying to hide the true name of the filename he or she was requesting. This is a fairly common evasion method and it’s easily defeated by the **http-url-parsed** context, since the context itself normalizes (parses) the URL before inspecting it. To match this attack, merely enter `\/[etc/passwd\]` (remembering to add case-insensitivity to catch further evasion) as the attack pattern with an attack context of **HTTP Decoded HTTP URL**. Name this something useful, starting with the **CS**: identifier, such as **CS:HTTP:ETC-PASSWD**, then assign it a severity (like **Medium**) and you’re done! Note we did not need a dot-star at the end of this pattern to account for the bar=yes parameter, as the http-url-parsed context stops before the question mark that delineates path from parameter. This is very similar to the Juniper-supplied signature **HTTP:INFO-LEAK:HTPASSWD-REQUEST**, whose match pattern is `.*\/[htpasswd\]`. The important differences are that this signature is looking for the file .htpasswd at the end of any path, which is covered by the dot-star and a forward slash. The dot in .htpasswd is also escaped for a more accurate match.

Sometimes, you may want to match a particular protocol’s (for example, HTTP or FTP) traffic on a port other than the typical ports used by that protocol. This is where the **Application** setting in the policy editor (see figure 10.6) comes in handy. The **Application** setting is a list of all application layer gateways (ALGs) and DI protocols parsed. The **Application** setting activates all ALGs and DI contexts for that protocol on whatever the service (pre-defined or custom) is
set to. So, to use FTP (with dynamic-gate ALG support) on a non-standard port, you would merely create a custom service for the command port used, then bind the FTP application setting to it. All traffic permitted by this policy will be inspected for FTP protocol conventions, including PORT commands that will be used to open data connection gates. Additionally, any FTP-based DI anomalies or signatures assigned to the policy in the Deep Inspection editing window will also be applied to this custom service.

**Figure 10.6 Policy Editor Application Field**

Now let’s tackle another protocol – Transmission Control Protocol. “But that’s not an Application Layer Protocol!” you shout. And you’re quite right. But there is a single context that can be bound to any TCP service supported by DI called *stream256*, and so for lack of a better description, this is a TCP context. Stream256 is a very simple, but very powerful context. It should be used with care. This context is powerful since it gets around the problem that certain useful contexts may not be customer-accessible. It’s also useful for inspecting protocols not supported by DI. The downside is that it only inspects the first 256 bytes of the stream, but you’d be surprised what you can do with 256 bytes. It inspects any DI-supported service that bound by policy. In order for stream256 to work, however, an **Application** setting (other than **None** or **Ignore**) must be selected.
Be sure not to use a dynamic-port protocol like H.323, SIP, PORTMAPPER, MSRPC-EPM or FTP, unless you are specifically writing a signature for these services, since the ALG will try to unnecessarily parse the protocol for dynamic-gate opening, and will slow traffic going through it. That is, if you’re writing a stream256 signature for HTTP, be sure to apply the HTTP Application setting. For non-DI supported protocols, use an uncommon Application setting like TALK to make stream256 work best.

The majority of the other contexts are fairly simple and self-explanatory. A few quick comments on some of the more interesting contexts:

- **dns-cname** examines just the DNS hostname of a DNS request, for example, www.juniper.net.

- **ftp-pathname** includes both the path and the filename. Add a dot-star to the beginning of the search pattern to look for files downloaded via FTP.

- **http-authorization** automatically decodes the Authorization: Basic Base64 code into a username:password format. Use `\[username\]:.*` to look for specific usernames and `.*:password` to look for specific passwords.

- **http-header-user-agent** can be used for browser identification, especially peer-to-peer and spyware applications that use the Web. Quite a few of them aren’t clever enough yet to hide their actual program names. Use `Gator:*` to match the Gator spyware program.

- **pop3-user** and **pop3-auth** provide you with username and password, respectively, for POP3 users.

- **smtp-rcpt** is essentially the To: field of an SMTP mail exchange. Good for looking for specific destination e-mail addresses.

- **smtp-mime-content-filename** provides filenames to attachments in e-mails sent via SMTP.

Also recall that signatures cannot be used by themselves; they must be incorporated into an attack group before they can be used. So before we add these attack objects to a policy, we need to make our own custom attack group. Attack groups must begin with the letters **CS:**; for example, **CS:HTTP-ATTACKS**, **CS:FTP-SIGS**, etc. Create a new custom group by accessing Objects | Attacks | Custom Groups and clicking the New button. Once in the new
group creation window, you’ll see a **Group Name** field (again, must start with **CS:**), as well as the **Selected Members** and **Available Members** lists. Merely select which custom signatures (Juniper-supplied signatures cannot be assigned to a custom signature group) you want this group to contain and click the «« button to move them over to the **Selected Members** list. Remove selected members with the «>> button. Once you have this new custom signature group, you’re ready to use it in a policy, just like any other attack group.

---

**Tools & Traps…**

**Advanced DI signature writing using an IDP**

One of the more powerful features of DI is the ability to write your own signatures for it. But one of frustrating things with this feature is the fact you can’t see quite how DI will actually inspect the traffic within its contexts. Does it include the HTTP GET verb? Are the parameters parsed, or just the path?

Since DI is truly a subset of the IDP feature set, if you have an IDP handy, you can use it for advanced DI signature development. This is the same technique used by the actual DI and IDP signature writers at Juniper Networks to develop production signatures. This can be used on a production IDP with minimal impact to network performance, but because of the volume of data provided, you might want this to be done on a lab IDP instead. Also note that the Environment Security Profiler (ESP) feature of the IDP (disabled by default) will have to be turned off before this technique can be used.

To start, log into your IDP remotely via SSH and become the root user. Using the command `scio ccap all` will show you every context that the IDP is parsing against traffic currently flowing through the device and exactly how that traffic is parsed. Press Ctrl + C to stop the display. While displaying, this will slow the unit some, especially if there is a lot of traffic. You can either limit the services to be examined by replacing the all with a service limiter command such as `scio ccap svc http` or `scio ccap svc ftp`. The command has a help system to show you valid services, but for the purposes of DI, remember that the services are somewhat limited.

Once you have your ccap (short for Context Capture) running, execute an attack that you’re interested in blocking through the IDP, and watch it parse your attack into contexts. At this point you should be able
to write a regular expression string to match your attack. Keep in mind that some contexts that exist on an IDP are limited in DI and may not be available. Work backwards from the contexts that are available in DI and find out how it is examined using an IDP. Since DI and the IDP view these contexts in exactly the same way, you can be reasonably certain that if it detects it on an IDP, it will work on DI.

Setting up Content Filtering

Juniper Networks’ NetScreen firewalls support content filtering through two major methods, URL filtering and antivirus. Of course, Deep Inspection could also be used as a sort of content filtering, but it’s not quite as suitable since it doesn’t present the end user with an appropriate error page when a violation is detected like the URL filtering feature does.

URL Filtering

URL filtering is the process of examining HTTP requests for content. Requests to inappropriate sites like those that host pornography, racism, or other offensive sites can be blocked using this feature. This works by comparing the requested URL against a database of classified sites. URLs can be categorically permitted or denied with a variety of configuration settings, which depend on the filtering server/service used. These services charge a recurring fee for updates to the database, as new sites are constantly created on the Internet. The URL filtering software provides Internet usage reports and also keeps track of repeat violators.

Starting with ScreenOS 5.1, NetScreen firewalls support the SurfControl Redirect protocol as well as the legacy WebSense Redirect protocol, but not both at the same time. With SurfControl, ScreenOS 5.1 also supports a special integrated mode on the NS-HSC, NS-5GT, NS-25 and NS-50. This loads the filter database directly on the device.

WebSense Redirect Mode

This is the only method of URL filtering for ScreenOS 5.0 and earlier. It’s fairly straightforward to configure on the NetScreen device – the majority of work required is on the WebSense server side, configuring users, user groups, policies, and exceptions. To use WebSense with ScreenOS 5.1, select this as protocol to use for URL filtering.
WebSense requires a Web server like IIS or Apache in order to operate. Usually that means installing it on Windows 2000 or 2003 Server, although it will run on Windows 2000 or XP Professional with Apache installed. If a Web server is not found, the installation will automatically download and install Apache for you if you permit it to. Setup and installation really is easy. 512 Megabytes of RAM is the minimum requirement, but at least 1 Gigabyte is strongly recommended. See www.websense.com/products/about/Enterprise/ for more product details. They offer a 30-day free trial.

From the firewall side of things, setup is a snap. See Figure 10.7 for an example of a firewall configured for WebSense Redirect Mode. First, enable the Enable URL Filtering option. Next, type the IP address (or DNS domain name if configured for your WebSense server, and your NetScreen has a DNS server configured) into the Server Name field. Enter the port number that the WebSense server is listening on for URL validation requests in the Server Port field (the default port is 15868). Now set a reasonable Communication Timeout value (10 is the default, but you may need more if the WebSense server is being accessed via a VPN). At this point, if the server is up and properly configured, a click on the yellow circle next to Server Connection Status should show that the server is running. If it doesn’t work right away, give it ten to fifteen seconds, and try the yellow circle button again. If the server is still down, check your settings (including DNS!), your routes, and your server. Try to ping the server by IP from a CLI prompt on the NetScreen if you’re having trouble getting the system to work.

There’s also an option to set the behavior of the firewall in the event that the URL server cannot be contacted. Using this option to either block all HTTP or permit all HTTP is a policy decision you’ll have to make on your own, depending on your business requirements and the stability of your WebSense setup.
SurfControl Web Filter for Juniper Networks Security Devices is a competitor to WebSense, and with ScreenOS 5.1 you now have a choice of URL filtering services to select. Like WebSense, SurfControl will work with 512 Megabytes of RAM, but would prefer 1 Gigabyte or more. SurfControl requires either an external MS-SQL database or an internal Microsoft Desktop Engine 2000 (MSDE2000) database. If an MSDE2000 database is not already installed, SurfControl will download and install it for you, similar to how WebSense handles a missing Web server – very handy. Like WebSense they also offer a 30-day free trial. After using both, I found WebSense easier to set up and configure, but unlike WebSense, SurfControl has an Integrated mode that uses public servers (which means no local installation!) that we’ll examine below.

Since it’s essentially the same concept, the configuration settings for SurfControl Redirect Mode are the same as WebSense Redirect Mode. To use SurfControl with ScreenOS 5.1, you’ll first need to select it as the protocol to use for URL filtering. Then fill in the options as you would in WebSense, such as turning on **Enable URL Filtering**, setting a **Server Name**, **Server Port** (the default for the Surf Control Filter Protocol (SCFP) is 62252), and a **Communications Timeout** value. Now it’s time to check the server availability. Click on the yellow circle to ensure you’ve set everything up correctly.
Try some of the troubleshooting tips found above in the WebSense section if things aren’t working right.

SurfControl Integrated Mode

This mode is only available on the newer low-to-midrange model NetScreen firewalls—the NS-HSC, NS-5GT, NS-25 and NS-50. SurfConrol Integrated Mode also requires a feature key to activate. To use SurfControl with ScreenOS 5.1, you’ll first need to select it as the protocol to use for URL filtering. The protocol used for this mode is the SurfControl Content Portal Authority (SC-CPA) protocol.

Once Integrated Mode is selected, configuration options for this mode appear. See Figure 10.8 for a reference. These options include:

- **Server Name** A drop-down list that allows you to select in which major continent (America, Europe, Asia) the device is located so the closest SurfControl server is selected.
- **Host** The actual hostname for the server to use. This value is automatically filled in from the **Server Name** field, but can be overridden.
- **Port** The port to use for communication with the URL filtering server database. SC-CPA’s default port is 9020.
- **Enable Cache** This allows the NetScreen device to cache the results of SC-CPA look-ups, decreasing the response time to end user requests.
- **Cache Size** The size (in kilobytes) allocated for the cache.
- **Cache Timeout** The length of time an entry will age off the cache if not requested.
- **Query Interval** How often the NetScreen device will check with the server for major category updates.
- **Permit** or **Block** Included as a default fallback decision if the server does not respond to a request.

Integrated Mode URL filtering supports the concept of **black lists** (always deny regardless of classification) and **white lists** (always permit regardless of classification) right on the device. These lists, as well as custom URL lists, are created by accessing **Screening | URL Filtering | Profile | Custom List** and clicking the **New** button. In the custom list edit window, add a name to this category (Whitelist, Blacklist, Competitors, etc) and add your first URL and click the
Apply button. The category name will be saved and locked, and the URL added to the list. Add additional URLs (up to a maximum of 20) by entering them in the URL field and clicking the Apply button. When you are done adding URLs, click the OK button to save.

Figure 10.8 URL Filtering Configuration with SurfControl Integrated Mode

Enforcing URL Filtering

In order to instruct the NetScreen device as to which sources and destinations are to be inspected for URL filtering, a policy definition is required. Refer back to Figure 10.3 for the basic policy editing window. Note that at the top of the second section there is a checkbox to enable URL filtering for the policy.

For Integrated Mode filtering, filter decisions (called a filter profile), based upon which category the URL matches, are made on the device itself. If no custom filtering profile exists, then the default ns-profile is compared and action is taken. This profile is a read-only template that can be cloned and the resulting copy modified and used. If a custom profile is created, either from scratch or cloned from the ns-profile, the inspection process becomes a little more complicated. As URLs are offered to the NetScreen for inspection, it first checks the user-defined whitelist, then the user-defined blacklist, then the NetScreen checks the URL for categorization. Once it knows the URL’s category, it first compares...
it to the user-defined filtering profile. Any match against the user profile is acted on as configured and the default profile check is skipped. If the URL category matches no defined categories in the user-defined profile, and the user-defined profile default action is set to \textbf{Permit}, then a final check against the ns-profile is performed, and whatever action is set for that category is then performed.

For Redirected Mode filtering, filter decisions (based upon which category the URL matches) are handled by the remote filtering server. This allows for advanced policy editing, including the ability to override a profile based upon a user login, or have different profiles based upon the time of day or day of the week. These features are offered by the third-party filtering servers, so check their product sheets for specific filters.

\section*{Antivirus Scanning}

ScreenOS 5.0 introduced the new antivirus engine from Trend Micro. This feature is supported on middle-to-low-end devices from NS-HSC through NS-208. A license key is required to activate the feature. Files sent via HTTP, FTP, POP3, IMAP, and SMTP are inspected for viruses right on the device.

\section*{Configuring Global Antivirus Parameters}

Access \texttt{Screening | Antivirus | Global} to display the ScreenOS 5.1.0 screen shown in Figure 10.9. The options to configure are:

\begin{itemize}
  \item \textbf{Fail Mode Traffic Permit} If this option is enabled, and for some reason the AV scan fails (either is too large, reaches a compression recursion limit, or some other scan failure), the traffic is still permitted. If the scan is successful, and no virus is found, the traffic is permitted regardless of this setting. If the scan detects a virus, the traffic is dropped, regardless of this setting.
  \item \textbf{AV HTTP Skipmime} If this option is enabled, the following MIME (Multipurpose Internet Mail Extension) types and subtypes are skipped from AV scan:
    \begin{itemize}
      \item application/x-director
      \item application/pdf
      \item audio/*
      \item image/*
    \end{itemize}
\end{itemize}
text/css

- text/html
- video/*

This improves AV scanning performance, since the majority of HTTP traffic uses these MIME types. Until recently this was considered a safe setting, so bypassing is enabled by default. Thanks to recent Microsoft vulnerabilities with JPEG and BMP file formats, this is no longer the case. Unless performance is suffering considerably, do NOT enable this option.

- **Keep Alive**  If this option is enabled, the NetScreen device will keep the HTTP session open to the server with a keep alive request after the file arrives on the NetScreen device, but before it has finished scanning the file. This decreases overall latency of the connection, but is less secure.

- **Trickling**  The method of sending a small portion of the file on to the requesting client so that the client’s browser won’t timeout the connection. The three options for this are **Disable** (which disables the trickling feature), **Default** (if the received file is larger than 3MB, it will trickle 500 bytes for every 1MB of data scanned) or **Custom**, in which you can set your own trickling settings:
  - **Minimum Length**  This sets the minimum file size to start trickling. Files smaller than this will not be trickled at all. Files of this size or larger are trickled according to the following settings.
  - **Trickle Size**  This sets the trickle packet size.
  - **Trickle For Every**  This sets the amount of traffic sent before a single packet is sent.

These settings handle how the device handles traffic it inspects, but how does the NetScreen get these attacks to match against? For these settings, we need to configure the Scan Manager.
Configuring Scan Manager Settings

An antivirus is only as good as its virus database, so to stay protected you need to stay updated. These settings are found by accessing Screen | Antivirus | Scan Manager in your WebUI. Figure 10.10 shows the 5.1.0 version of this screen. This shows important information such as AV license entitlement as well as how current your virus definitions are. Other important features on this screen are:

- **Pattern Update Server** This is the server the device will automatically connect to in order to obtain AV pattern updates from. The default is http://[device-model]-t.activeupdate.trendmicro.com:80/activeupdate/server.ini.

  The [device-model]-t portion of the URL for my NetScreen-5GT was 5gt-t.

- **Auto Pattern Update** This permits or disables the auto-update feature. Leave this option enabled unless you’re experiencing problems obtaining the update.

- **Interval** This sets how often (in minutes) it checks for an update, with a default of 60. Don’t be concerned if the Last Updated on: value
doesn’t coincide within the current time minus the update time. The date/time group at the top is the date/time of the AV pattern file, and not necessarily when it was loaded. Viruses spread most quickly during the first 24 hours (due to lack of antiviral protection) so be sure to leave this interval low.

- **Update Now** A handy button to check your settings and to manually refresh your AV definitions if you’ve turned off automatic updates.

Below that first section there are a few configuration options for handling compressed files. Since a single compressed file can contain one or more files, and those files themselves can be compressed, in order to ensure that all content is checked, the NetScreen device will decompress zip files if possible and examine the results for viruses. This can be time and resource consuming, so some practical limits were introduced, but are user-configurable. Note that compressed file nesting is a common AV evasion technique. Another AV evasion technique is sending a very small compressed file that expands into an extremely large uncompressed file, attached with a second file that is malicious. The idea is that the AV scanner would max out on the benign, but extremely large file and then skip checking any other files after it, including the resulting malicious files. To prevent these kinds of evasion techniques you can adjust the following settings:

- **File Decompression** The number of recursions the scanner will go down into, from 1 to 4.
- **Drop/Bypass and Size Exceeds** This sets the limit for the size of uncompressed files to scan, and also an action if this limit is reached.
- **Drop/Bypass and Number of Files Exceeds** This sets the limit for the number of uncompressed files to scan, and also an action if this limit is reached.

The final section in the Scan Manager configuration screen covers which protocols will be inspected for viruses. Keep in mind, this is a global setting – any protocol turned off here will not be scanned, regardless of how the configuration of the policy that the AV is applied to. Protocol options include **HTTP** (with options for **Webmail** or **All HTTP**), **SMTP**, **POP3**, **FTP**, and **IMAP**. Disabling any of these options will relieve your NetScreen of some burden, but will also increase your likelihood of infection. Turn off these protocols only if you are certain they cannot be passed (for example, if you are using an explicit blocking policy).
For the webmail blocking option, the NetScreen has to have the URL path portion configured for relevant webmail systems in order to determine if it has to check attachments for viruses. By default, three popular webmail sites are pre-configured – AOL, Yahoo, and Hotmail. The settings for these sites may change as the developers for these sites add new features or make other changes. Also, any other webmail site you want filtered will have to be added manually and monitored for effectiveness due to changes later. I would recommend not using this feature unless the NetScreen device simply cannot pass HTTP traffic fast enough in All HTTP mode. Also keep in mind that if you do use this mode, you can set any URL path, not just webmail, to check for viruses, such as popular file download sites like TuCows, Freshmeat, or FilePlanet.

Activating Antivirus Scanning

Activating AV couldn’t be easier. Currently, there is only one AV object on the device, so refer back to Figure 10.3 again and find the Antivirus Objects section. Select the scan-mgr object from the Available AV Object Names column and click the button to move it to the Attached AV Object Names column. Be sure to click the OK button to save the setting. Do this for each policy that needs antivirus scanning on any of the supported protocols.
Understanding Application Layer Gateways

Application Layer Gateways are algorithms within ScreenOS that handle dynamic firewall policies that certain protocols require, such as FTP. Many such protocols were designed without security or other access controls in mind, which can cause problems when firewalls are introduced.

For example, FTP uses multiple sessions to facilitate file transfers – a primary command channel, and secondary data channels for directory listings and file transfers. Often, these data channels will flow in the opposite direction than the original command channel. Since these data channels could connect on any port, it’s almost impossible to create a static firewall policy that would permit these data channels, but still provide adequate protection.

The FTP ALG automatically solves this problem by monitoring the FTP command channel, looking for FTP port commands that specify which source and destination ports are being requested, and dynamically opening up that specific combination of source IP/port and destination IP/port firewall policy (called a gate) that permits the session to flow. Once the session is complete, the gate is immediately closed.

The FTP ALG also handles the special case where the FTP session flows through a NAT interface. In this circumstance, the endpoints don’t always realize their addresses are being translated mid-stream. The FTP port commands use whatever IP the endpoint hosts’ interfaces are configured for, which, in the case of a host behind a NAT firewall, will typically be unreachable from the Internet.

The ALG handles this at the application layer by modifying the ASCII port command in-situ, replacing the inside IP with the IP of the NAT interface. Since port commands are passed as ASCII text, including the IP address, the chances are high that the number of characters that represent the inside IP and the external IP won’t exactly match (for example, an inside address of 192.168.1.5 contains 11 characters, which may be translated to something like 123.123.123.123 at 15 characters or something like 1.2.3.4, which contains only 7). The firewall cannot inject these extra bytes of data without modifying the TCP checksum as well as the TCP sequence numbers. It achieves this by essentially proxying the connection at the TCP layer. This is similar to the SYN proxy feature used by the TCP flood SCREEN setting.

NetScreen ALGs are different from many competitors’ products. Many other firewall vendors utilize full protocol proxies, which themselves are vulnerable to
attack, misconfiguration, or protocol obsolescence as new commands, options, and features are added to a protocol. CheckPoint’s FireWall1 uses tiny proxies to validate data on protocols like HTTP, FTP, and SMTP. While this method is very flexible, it can still cause problems if the proxy encounters a valid command that it has not been programmed to handle, which could cause the session to break, since the proxy won’t forward what it thinks is an invalid command.

Furthermore, since the firewall is participating in the stream at the application layer, it’s very possible (and has even happened) that the proxy itself is vulnerable to a security concern. Since FireWall1 runs on Windows, Linux, and Solaris, shellcode for these platforms is relatively easy to find. NetScreen firewalls do not participate in the exchange at the application layer, which isolates them from these sorts of attacks.

Some protocols just don’t support being proxied. Microsoft’s Server Message Block and Remote Procedure Call both require a real endpoint connection. While these are not commonly Internet-transiting protocols, a good defense-in-depth strategy would still have this traffic flowing through firewalls that need to know how to handle it. A new ALG found in ScreenOS 5.1 allows users to filter at the application layer for MS-RPC by parsing globally unique identifiers (GUIDs) —a unique 128-bit number used by Microsoft to label process endpoints. Custom-defined services are created based upon GUIDs, which are then used in a policy. This enables you to create policies that allow or prevent access to individual processes on a Windows system. This is very handy for protecting from attacks such as Blaster, Sasser, Agobot and others that use MS-RPC as one of their attack vectors.

Others vendors tend to cut corners and, for the sake of performance, will make a very simple ALG-like algorithm that should solve a problem, but has unexpected consequences. Just recently Symantec issued a security update for its DNS ALG. Apparently, the DNS ALG worked like so: if a UDP packet arrived with a source port of 53, it was a DNS reply to a DNS request that had already gone out through the firewall, and would permit the packet through without any session lookup. The ALG would also bypass any incoming policy explicitly blocking the packet, such as destination port, destination IP, or source IP. The flaw in the firewall was so fundamental that it would even bypass protections designed for its own management interface. When this oversight was made public, hackers discovered that sending management packets to the Simple Network Management Protocol (SNMP) port on the firewall from UDP port 53, they could successfully command the firewall and change the firewall’s settings.
without being authenticated. A patch was later released. ScreenOS features are subject to security review at various stages of the development process to avoid fundamental logic flaws such as this.

ScreenOS currently has 26 ALGs, including FTP, DNS, and H.323, with more being released with every new version. These ALGs require little to no configuration to operate properly. They automatically detect appropriate traffic on the registered ports for the protocol they handle and then do their jobs. As mentioned earlier, these ALGs can be reapplied to arbitrary ports using custom service objects as needed.

**Applying Best Practices**

NetScreen firewalls have a wealth of security features to use, but even the best tool can be rendered ineffective through poor implementation. This section hopes to instill some good security practices to use with your NetScreen device.

**Defense-In-Depth**

How many locks do you have on your front door? Just one? Or do you have one lock for the doorknob, another for the deadbolt, and a chain? Do you have an alarm system as well? How about a bat by the bed? If you have all of this, then you already understand what *defense-in-depth* means. Network security is no different. Having a NetScreen firewall protecting your network is a good start to an overall effective network security system. However, it is the components of the whole system working together—internal firewalls, perimeter firewalls, IDPs, authentication services, management, antivirus software, and monitoring services—that make you more secure. The National Security Agency (NSA) has recently released a very informative paper on this subject, located at www.nsa.gov/snac/support/WORMPAPER.pdf.

**Zone Isolation**

An extension of the defense-in-depth concept, zone isolation involves placing different system types (for example, servers, end users, Engineering, Finance, Information Technology, etc) on different zones, which then allow for firewall policies between these dissimilar groups. Do your end users need access to the Finance department’s systems? If so, what kind? Find out how to limit access to just what is necessary. Don’t just assume that because a computer is inside your
perimeter that it’s safe. Keep access to specific areas, zones and data to the smallest possible number of computers.

Egress Filtering

Egress filtering is the process of putting restrictions on outgoing traffic as well as incoming traffic. Many locations only get half of the security picture straight—they block traffic from coming in except to specific Internet-facing servers (mail, Web, DNS, etc.) but let all inside traffic go back out completely unfiltered. Ideally, your outgoing policies should be as complex and stringent as your incoming policies. Or better yet, have no traffic initiated by your end users allowed out to the Internet, but rather have all traffic go through approved and configured proxy servers (such as HTTP or FTP). I highly recommend the highly-configurable open source proxy called Squid that comes with about every distribution of Linux out there) or internal-only servers (such as mail or DNS). This locks down infections since most backdoors don’t support proxies, and those that do can be detected and blocked at your proxy.

Explicit Permits, Implicit Denies

The idiom of “You don’t know what you don’t know” is never truer than in the security business. Firewall administrators who block specific ports and let all others through are asking for trouble through their ignorance. Why Windows XP listens on 10 different ports to perform the same function, I’ll never know. If there’s just one port I miss, a worm or other malicious attack could slip by and tear up my network from the inside. Instead, permit what you know you want permitted, and block everything else—this keeps things simple and threats known.

Retain Monitoring Data

If something does happen, and it usually does, you need to know the breadth and width of the problem, when it started, and how it happened so that you can properly clean it all up and keep it from happening again. You’re going to have enough trouble as it is from hackers hiding their activities through evasion and log file deletion on compromised systems. Don’t compound this problem by not having dedicated, secure machines for logging and keeping the data for a historically significant period of time. This may be the only way to track a Black Hat attacker who spaces his attacks out over hours or days. It doesn’t hurt to look at the logs from time to time as well to ensure things are copasetic.
Keep Systems Updated

If there’s one thing 2004 has taught network security professionals, it’s that automatic operating system updates are a good thing. Windows, MacOS, and many flavors of Linux now support some sort of automatic patching system to react to newly discovered security issues. Use these tools to your advantage, and keep your systems patched. Check in with Windows Update the first Tuesday of the month to see what new vulnerabilities Microsoft has ‘fessed up to, and get to patching!
Summary

This chapter has covered a lot of ground in a short amount of space. Indeed, complete books are available just on subjects covered here. Despite this, we’ve managed to cover all SCREEN, Deep Inspection, URL filtering, and antivirus features, as well as the Application Layer Gateway functions enough to give you a good understanding of the capabilities of each, and how to set them up.

SCREEN features are the legacy security protection and cover things like SYN, UDP, and ICMP floods (with user-configurable thresholds), session-exhaustion attacks (with separate thresholds for source and destination), classic IP and TCP header manipulation, port scans and sweeps, and certain OS-specific DoS attacks. Enabling these attacks imposes almost no performance hit.

Deep Inspection takes security all the way to the application layer for selected protocols. Using stateful contexts to accurately isolate malicious traffic, you can minimize false positives and false negatives and maximize valid hits. Protocols covered in ScreenOS 5.0 included HTTP, FTP, DNS, SMTP, IMAP, and POP3, with a very limited set of contexts exposed to end users for use in their own signatures. ScreenOS 5.1 introduced several new protocols including SMB, MS-RPC, NetBIOS, Gnutella, AIM, and YMSG and included a significant increase in the number of contexts supported for end user’s signatures.

We covered how ScreenOS uses contexts to break down a protocol stream into inspectable fields and uses a DFA to inspect traffic, and how to use customer available contexts and the DFA RegEx syntax to write our own custom signatures.

Also covered was ScreenOS’ support for HTTP URL filtering with a variety of URL filtering options from WebSense and SurfControl, including a method of storing filtering profile right on the device with SurfControl Integrated Mode. The Integrated Mode requires a license key.

Antivirus is an important new feature in ScreenOS 5.0 that allows HTTP, FTP, SMTP, POP3 and IMAP protocols to be inspected for viruses. While there are a variety of settings to configure, the majority of the defaults work well, and with a few clicks AV can be up and running almost effortlessly. Antivirus also requires a license key.

Application layer gateways are the grease that allows certain security-impaired protocols to work through a firewall. FTP, H.323 and dynamic-channel protocols could play havoc with a firewall policy if it wasn’t for ALGs and the gate feature that opens dynamic firewall policies automatically to allow data channels of already permitted command channels of these troublesome protocols.
Finally, we covered some security basics—how to put this all together into a more secure whole. Defense-in-depth, zone isolation, egress filtering, implicit permits, explicit denies, logging and keeping systems up to date aren’t just topics in a CISSP book—they’re real-world solutions for making your network more secure. Applying even just a few of these concepts will go a long way to making the Internet a safer place.

**Solutions Fast Track**

**Introduction to the ScreenOS Security Features**

- NetScreen firewall products pack a diverse range of features into a small, easy-to-use system.
- A firewall’s primary function is always security.
- NetScreen firewall products have a variety of different security methods to stop many different types of attack.
- SCREEN features generally cover IP and TCP layer attacks.
- Deep inspection covers advanced IP, TCP/UDP, and Application Layer attacks.
- Content filtering enforces local usage policy.
- Antivirus keeps mass-mailing worms and other malware out of your network.
- Application Later Gateways handle problem protocols securely.

**Understanding the Anatomy of a Hack**

- Generally, there are three phases of an attack: recon, exploit, and consolidation.
- Different kinds of attackers require different methods for protection.
- Script Kiddies want publicity for their activities and will make their presence very well known.
- Black Hat hackers are more insidious and patient—it takes diligence and consistency to detect them.
Self-propagating worms are becoming more aggressive and complex as time goes by. They will attack like a Kiddy but exploit like a Black Hat.

Defense-in-depth, updated systems and signatures, event log vigilance, good social engineering awareness training and a sound security policy can keep these attacks at bay.

Configuring SCREEEN Settings

- NetScreen legacy screen settings protect networks using a variety of detection methods.
- Network reconnaissance uses port scans, sweeps, and protocol option manipulation to avoid detection.
- Port scans and sweeps are detected, and can be tweaked with user-configurable threshold settings.
- Several TCP flag manipulation techniques are detected, including null scan, SYN/FIN scan, and others.
- A variety IP protocol option manipulation techniques are also detected, such as source-route, time-stamp and others.
- SYN, UDP, and ICMP flood attacks are prevented with user-definable thresholds to maximize throughput while still providing protection.
- Rudimentary HTTP content filtering is supported, with the optional blocking of ActiveX and JavaScripts, as well as EXE and ZIP file downloads.
- Other protocol attacks, such as WinNuke, Land, Ping of Death, and others are also covered.

Applying Deep Inspection

- Deep Inspection covers application layer attacks on selected protocols.
- With ScreenOS 5.0, 6 protocols are covered: HTTP, FTP, DNS, SMTP, POP3, and IMAP.
- Customer-exposed contexts were also extremely limited in ScreenOS 5.0.
ScreenOS 5.1 introduced several new protocols with new contexts: SMB, MS-RPC, NetBIOS, Gnutella, AIM, and YMSG.

ScreenOS 5.1 also added new contexts for ScreenOS 5.0 supported protocols for end users to write signatures with.

Signatures use a custom subset of regular expressions and a DFA string-matching algorithm to detect attacks.

Setting up Content Filtering

- NetScreen firewall products support both URL filtering and, more recently, antivirus filtering.
- Starting with ScreenOS 5.1, NetScreen now also supports SurfControl as well as the legacy WebSense URL filtering system.
- On newer low to midrange NetScreens, SurfControl can also be used in Integrated Mode right on the device.
- NetScreen firewalls support off-loading as well as on-board antivirus inspection starting with ScreenOS 5.1.
- On-board antivirus inspection uses the Trend Micro engine.

Understanding Application Layer Gateways

- Many commonly used protocols, such as FTP, MS-RPC, or H.323, were never designed to be firewalled.
- ALGs enable these security-impaired protocols to work through a firewall by parsing the command channel.
- ALGs are not protocol proxies, which are limited to certain protocols and are subject to change.
- Other competitors’ over-simplified ALGs can contain logic errors leading to security breaches.

Applying Best Practices

- Defense-in-depth distributes security across your infrastructure to prevent single-point-of-failure compromises.
Zone isolation increases the granularity of control over devices in your network.

Egress filtering ensures protects against unknown activity leaving your network.

Explicitly permitting desired traffic and blocking everything else ensures you know what’s allowed where.

Keeping logging systems secure and monitored will help you isolate problems and keep security events from recurring.

Most common operating systems now support automatic updates—use them!

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to www.syngress.com/solutions and click on the “Ask the Author” form. You will also gain access to thousands of other FAQs at ITFAQnet.com.

Q: What’s a “Grey Hat” hacker?
A: A Grey Hat is someone who knows both sides of the security coin (Black Hat and White Hat) and will dabble with both. Like many things in life, ‘good’ and ‘bad’ distinctions are not so binary in the real world. Is it okay to use a malicious attack against an offensive site (hate crime, child porn, spam relay etc.)? Is it okay to attack a host that attacked you first? Computer ethics classes are now a regular component of most security certification courses.

Q: What’s a good setting for the (SYN | UDP | ICMP) flood threshold?
A: There’s no universal setting that could be applied effectively to every network. Network activity, purpose, and traffic are different for each segment, and they need to be tweaked appropriately. Ideally, you’d want a setting that’s just above dropping normal volumes of legitimate flows, so when an actual flood occurs, the NetScreen can react immediately.
Q: Why doesn’t DI have the same coverage as an IDP?
A: NetScreen firewall devices have purpose-built Application-Specific Integrated Circuits (ASICs) that handle the majority of firewall operations, such as policy matching and data encryption. These are physical devices—hardware accelerators—that can’t be modified. Since DI is a relatively new feature, it has to run in software on the system’s CPU. Older NetScreen firewall CPUs were generously sufficient for device management, but are significantly slower than the ~3 GHz single and dual CPUs found on an IDP. This difference in computing power means that a full IDP would be impractical on the currently available platforms. Juniper is actively investigating a new ASIC that incorporates IDP functionality in hardware, as well as ensuring that new generations of NetScreen firewalls have beefier CPUs.

Q: Why is SurfControl Integrated Mode only available on lower-end products?
A: Many users of these more inexpensive products don’t have permanent or full-time onsite IT support personnel, and may not have the time, money, or expertise to configure, run, and maintain their own WebSense or SurfControl URL filtering software. Yet these customers still want (and need) URL filtering. The Integrated Mode utilizes SurfControl’s public servers that SurfControl maintains and supports. Also at risk is the performance impact (including latency) of sending large volumes of URL look-up requests over the Internet to a public server. The high-end products support speeds well over 10 Gb/s of sustained traffic—to look up every URL requested in real-time would waste significant bandwidth.

Q: Why should I bother with egress filtering? It’s a lot of work, and my users are bound to complain about something not working.
A: The initial effort of configuring egress filtering now will save you several orders of magnitude’s worth of work later. The majority of business-related software supports proxying or other filtering. Chances are, complaints from some end users regarding connectivity problems generally arise from software you don’t want running on your network anyway.
Q: Why are there so many different license keys for features?

A: Juniper, like any for-profit company, wants to make money. It also understands that it needs to be competitive in the market. If Juniper sold its devices at a high price with all features enabled (many of which you may or may not use), it would have a difficult time selling them to customers who only needed some of the features and were willing to buy them at a reduced price. This way, a compromise is reached—they will sell you a useful product for a reasonable price, but the flexibility is there for additional features, which you can then buy a la carte. Time-limited license keys also facilitate subscriptions.