Check Point Firewall-1's Stateful Inspection
Michael J. Nikitas
April 14, 2001
Version 1.2b

As major advances in technology are made, we need to consider not only the benefits, but also the drawbacks. With any new product, protocol, or application our lives become easier and productivity is increased. The overall benefit of these advances is immeasurable. However, just as these advancements help to benefit the greater good of all, they also serve to benefit others who choose to use technology for malicious activity. Be it just a "script kiddy" defacing a website, or a hacker that breaks into a corporate LAN to steal company secrets; technological advances are benefiting all, both good and bad. These types of activities have brought to the forefront the need for network security, not only to protect businesses, organizations, and corporations, but also the home PC, which is just as susceptible to attack. This problem has caused many to think about network security and has made it a major topic of the 21st century. When considering security, the first piece of a security architecture is usually a firewall. This is typically the point in the network, regardless of location, that separates your investments from the Internet. Firewalls are typically the first line of defense for any network. That stated, this paper will focus on one aspect of a firewall, how it actually inspects the packets going to and from the network. I will primarily focus on Check Point Firewall-1's stateful inspection. In order to effectively do this I will give an overview of the two other primary types of firewalls: Packet Filters and Application Layer Gateways. Then I will explore how stateful inspection actually works.

Packet filters are most commonly routers, although in some instances hosts or bridges. More specifically, from Building Internet Firewalls, "devices that do this are packet filtering bridges. They are rarer than packet filtering routers, mostly because they are dedicated security devices that don't provide all the other functions that routers do."[1] Packet filters predominantly make decisions at layer three, the Network layer, of the OSI model, however, they can also make forwarding decisions at layer 4, the Transport layer, as well. Typically, packet filtering is implemented in the form of ACL's (access control list) and operate in conjunction with normal routing. The primary function of routers is to route traffic, and to do so efficiently. This is one of the benefits of a packet filter, it is generally considered to have the best performance of the three types of firewalls.

However, there are many disadvantages that can affect the performance of a packet filter. The first is that access control lists can be very difficult to develop. Creating one takes a lot of thought and the administrator must be absolutely sure of what he/she is trying to accomplish and of the syntax to use in writing the ACL. They can be very difficult to write for people who have not had experience in writing them. To make it more difficult, there is no real way to verify what has been written in the ACL. There are no means of verifying that one rule conflicts with another or that the order has a least restrictive rule ahead of a more restrictive rule, causing that rule to never be used. These lists can quickly become unmanageable.

Due to the nature of packet filters, only inspecting data at layers three and four, they do not provide any real security against an attack. Check Point Support writes: "The reason is that they are not application aware-that is they cannot understand the context of a given communication, making them easier for hackers to break."[2] They can prevent access to specific devices and can direct traffic to where the administrator wishes it to go, but they have no way of inspecting the actual data in the packets. Thus leaving portions of the network susceptible to attack. For instance, assume the packet filter is in front of the DMZ (de-militarized zone), where all of your public web and FTP servers are located. The packet filter can allow only internet traffic to port 80 on the web servers, but it has no way of inspecting the data to prevent an attack against a server on that port. Another drawback is that these devices are typically managed via SNMP and through TELNET. These are two very insecure protocols that pass data in cleartext. Thus, if a hacker were able to capture these packets from a compromised host, he would be able to gain access to the box and make modifications to his\her advantage and this would all be done as a privileged user. It is important to note these two methods are not the only way to manage and administer these devices and they can be secured through other methods.

Application layer gateways, or more commonly proxies, are the exact opposites of packet filters. They operate at the Application layer, seven of the OSI model. They are generally considered to be slower in performance than packet filters, although with faster and faster machines being developed, this is not as big a discrepancy as it was. Typically, a proxy server has the full benefit of analyzing all of the data for a specific protocol. They are able to perform extensive logging, authentication, and content checking. How a proxy works is two-fold. There is a proxy for each service you wish to use on the network. Each proxy is written specifically for a protocol and can evaluate all the way up to the application layer. The proxy typically runs as a daemon on the machine it is installed on. Once the gateway is set up to perform the desired actions it works as a proxy, interrupting the client server model. When a user wishes to run a specific application, a web browser, for instance, the client makes its request directly to the proxy server. The proxy in turn inspects the packet and then opens a new connection from itself to the destination. The return traffic comes back to the proxy, which in turn sends it back to the client. The client has no knowledge of the proxy, and the device that was being communicated to thinks the request came from the proxy, not the actual user. This actually serves a dual purpose of hiding the protected network as well. All of these features provide for a more secure environment than can be afforded by a typical packet filter, as all of the data in a packet is reviewed.

Application layer gateways have their drawbacks as well. Primarily they are susceptible to attacks to the operating system on which they are loaded. If extreme care has not been taken to harden the OS, then the proxy becomes susceptible to those types of attacks. Due to the fact that these rely on proxies, is another drawback. This means that for every protocol that you wish to pass through the proxy, a new proxy must be written. Due to the speed with which technology is advancing, new protocols are developed often, leaving a lag in time for which a proxy needs to be developed for that specific service. Also, additional software may be required for the client to run a specific service. Not only that, but additional training for those users may be required to teach them how to use the software, taking something that the user knew how to perform and making it more difficult.

Firewall-1's Stateful inspection is based on Check Point's patented INSPECT Engine. This Engine inserts itself between the Data Link layer and the Network layer, layers two and three of the OSI model. This Engine runs in kernel mode making it extremely fast and efficient. Due to this placement, the engine is able to intercept all packets prior to them going up the stack. At that point it is able to compare the packet against the security policy and make a decision on what to do with the packet without exposing the upper layers of the IP stack. If the packet matches a rule in the security policy that allows it to pass, the packet is sent up the stack for processing. This engine has the ability to inspect up to, and including the Application layer of the OSI model, all in the kernel making it extremely robust.

When a packet is received by the firewall, the first thing it does is check it against the state table to see if there is an existing connection to which this packet belongs. If there is then the packet is forwarded along. If there is no matching connection in the state table for that specific packet, then the firewall compares it against the security policy to see if there is a match that allows the packet to pass. If there is, then the connection is added to the state table and all subsequent packets belonging to that conversation will be forwarded along immediately, without being checked against the policy, making this technology extremely efficient. The command to view the state table is fw tab –t connections -u. The state table is stored in hexadecimal format and has the following format: Source IP, Source Port, Destination IP, Destination Port, IP Protocol Type, Kbuf, Type, Flags, and Timeout as seen below an example of a portion of a state table:

```
      localhost:
      -------- connections --------
attributes:refresh,sync,expires 60, free function 3256265368 4, kbuf 1, implies 21, hashsize 32768, limit 25000

      <868d9095, 00000707, ac1a413f, 000000a1, 00000011; 00000000, 00004002, ffffff00; 3637/40>
      <868d9095, 00000707, ac1a413e, 000000a1, 00000011; 00000000, 00004002, ffffff00; 3637/40>
      <ac1ac803, 000000a1, 868d9057, 00000f00, 00000011; 00000000, 00004002, ffff0600; 3613/40>
      <ac1a411e, 0000049d, 868d90cb, 000001bb, 00000006; 00000000, 00006001, ffffff00; 4621/3600>
      <ac1a411e, 0000049c, 868d90cb, 000001bb, 00000006; 00000000, 00006001, ffffff00; 4620/3600>
      <ac1a411e, 0000049b, 868d90cb, 000001bb, 00000006; 00000000, 00006001, ffffff00; 4620/3600>
      <ac1a411e, 0000049a, 868d90cb, 000001bb, 00000006; 00000000, 00006001, ffffff00; 4620/3600>
      <ac1a411e, 000004a1, 868d90cb, 000001bb, 00000006; 00000000, 00006001, ffffff00; 4620/3600>
      <ac1a411e, 000004a0, 868d90cb, 000001bb, 00000006; 00000000, 00006001, ffffff00; 4621/3600>
      <868d90b6, 00008587, ac1a41a5, 000000a1, 00000011; 00000000, 00004002, ffffff00; 3604/40>
```

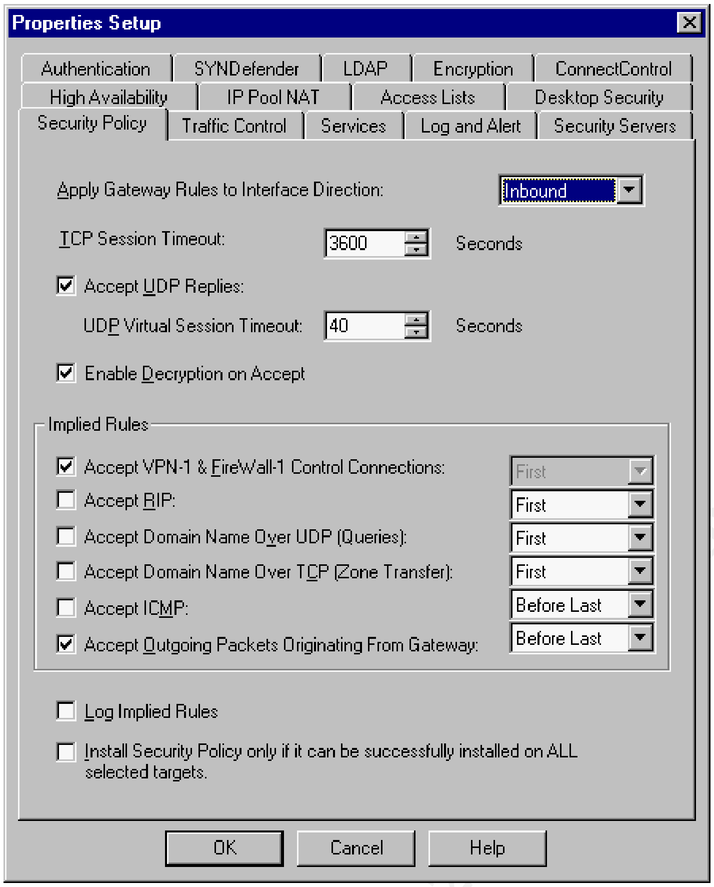
Using Lance Spitzner's perl script fwtable.pl[3] we can decode the contents of the state table to a more human readable format as seen below:

```
---- FW-1 CONNECTIONS TABLE ---


Src_IP          Src_Prt  Dst_IP        Dst_Prt  IP_prot Kbuf   Type    Flags           Timeout

134.141.144.149 1799   172.26.65.63     161     17      0      16386   ffffff00        3637/40
134.141.144.149 1799   172.26.65.62     161     17      0      16386   ffffff00        3637/40
172.26.200.3    161    134.141.144.87   3840    17      0      16386   ffff0600        3613/40
172.26.65.30    1181   134.141.144.203  443     6       0      24577   ffffff00        4621/3600
172.26.65.30    1180   134.141.144.203  443     6       0      24577   ffffff00        4620/3600
172.26.65.30    1179   134.141.144.203  443     6       0      24577   ffffff00        4620/3600
172.26.65.30    1178   134.141.144.203  443     6       0      24577   ffffff00        4620/3600
172.26.65.30    1185   134.141.144.203  443     6       0      24577   ffffff00        4620/3600
172.26.65.30    1184   134.141.144.203  443     6       0      24577   ffffff00        4621/3600
134.141.144.182 34183  172.26.65.165    161     17      0      16386   ffffff00        3604/40
```

We have briefly illustrated how the state table is created. We will now take a more in depth look at what actually happens when the firewall receives a connection attempt based on either TCP or UDP. TCP packets are the easiest for the firewall to monitor based on the inherent nature of the protocol, as it actually builds a connection and maintains state itself. When the firewall receives a TCP SYN packet, the first packet in the three-way handshake, (SYN, SYN-ACK, ACK), for setting up the connection the firewall checks the packet against the existing security policy. If there is a rule that allows the communication to take place, it is added to the state table. Lance Spitzner notes: "At this point, the timeout is first set to 60 seconds. The firewall then expects a return packet to build the connection. When it sees this return packet, the timeout is then set to 3600 seconds (60 minutes)."[4] If not, then depending on the action to be taken either the packet is dropped, or a reset is sent to the host. This depends on whether the rule specifies drop or reject. A reject specifies to notify the sender that the communication has failed (an RST is sent), whereas drop sends no response back to the originator. Once the connection has been added to the state table, all further communication attempts based on that specific connection are first checked against the state table, if the entry still exists the packet is allowed to pass until the connection is terminated. This will be either through the normal termination methods (client sends a FIN and destination sends and ACK and Destination then sends a FIN and client sends the final ACK), or the connection is timed out based on the timeout value specified for TCP connections, which we will discuss later. UDP conversations are a little more difficult to handle as the protocol's nature is essentially stateless, meaning the packets are sent on a best effort basis. What happens with UDP packets is the first communication attempt is checked against the rule base. If there is a matching rule that allows the communication the packet is accepted and a pseudo-connection is added to the state table based on the source IP, source port, destination IP, and destination port. All subsequent packets are allowed to pass as long as the state table maintains a connection for that communication. Due to the fact that UDP has no inherent error correcting, if the policy states an action of reject, which allows for a failure notification back to the host, I recognized that an ICMP Port Unreachable message was sent back to the host. When the policy was set to drop, no notification was sent and the attempt was blocked. Additionally, it was discovered during my testing that there is no stateful inspection of ICMP packets. No entries were made to the state table when I passed ICMP through the firewall. From my research I have read that attempts at writing INSPECT code to monitor ICMP statefully have been made. There have been claims that these attempts were successful. I did not verify these claims, as they are not part of the default installation of Firewall-1. They do go to show though, that the INSPECT script is very versatile and flexible. Administrators can create custom scripts on their own to perform the actions they desire.

This brings us to the next logical step, the connection table itself. One might wonder with all of these connections passing through the FW, at what point does the table fill up. Looking at the table.def file, we see that the maximum amount of connections is 25,000. This may seem like a lot but, depending on the size of the network being protected this may not be sufficient. There are several options one has with regards to the state table. The ability is there to increase the size of the state table to a higher value, but let's look a little closer at how it actually works. The firewall has some default timeouts on connections in its state table. These defaults get set back to the full value during communications every time the firewall sees a packet for that connection. Once the connection has been idle for the entire timeout period the connection is aged out of the state table. The default timeouts are as follows: for TCP 3600 seconds, or one hour and for UDP packets 40 seconds. These values are global settings for all TCP and UDP packets and are set in the policy editor. Go to Policy, then select Properties.

Under the Security Policy tab you can change these global values to meet your preferences. A screen shot is provided below:



Apply Gateway Rules to Interface Direction: Inbound

TCP Session Timeout: 3600 Seconds

☑ Accept UDP Replies:

UDP Virtual Session Timeout: 40 Seconds

☑ Enable Decryption on Accept

Implied Rules

☑ Accept VPN-1 & FireWall-1 Control Connections: First

☐ Accept RIP: First

☐ Accept Domain Name Over UDP (Queries): First

☐ Accept Domain Name Over TCP (Zone Transfer): First

☐ Accept ICMP: Before Last

☑ Accept Outgoing Packets Originating From Gateway: Before Last

☐ Log Implied Rules

☐ Install Security Policy only if it can be successfully installed on ALL selected targets.

Every time the policy is pushed down the table is cleared and existing connections will need to be reset. Usually this is handled by the protocol or application itself. The reason for this is so that if the policy has been modified such that a previous connection is no longer allowed, the firewall will then inspect the next packet to make sure that the connection is still allowed. For TCP connections the firewall will mash the entire packet with the exception of the IP and TCP headers. This will force the receiving endstation to request a re-transmit at which time the packet will be inspected against the rulebase again and normal procedures will follow. The table also has the ability to modify the timeout for a specific protocol, in the event you do not wish to increase the timeout for all transmissions. To do this you will need to edit the init.def file on the management console. From Phoneboy.com;[5] go to the line that reads: ADD_TCP_TIMEOUT(0,0). The first 0 indicates the port the service uses and the second 0 indicates the time you wish the connection to timeout at. For instance, I modified the table to allow TELNET connections to timeout after 5 hours. To modify an entry to timeout higher than 7200 seconds (2 hours) you will need to use the format (0,3600*5). So to implement the changes I desired I added the following line above ADD_TCP_TIMEOUT(0,0), ADD_TCP_TIMEOUT(23,3600*5). The changes I implemented now allowed for TELNET connections to be timed out at 18000 seconds, or five hours, while the rest get timed out at the default of 3600 seconds, or one hour. Here is an example of the modified connections table:

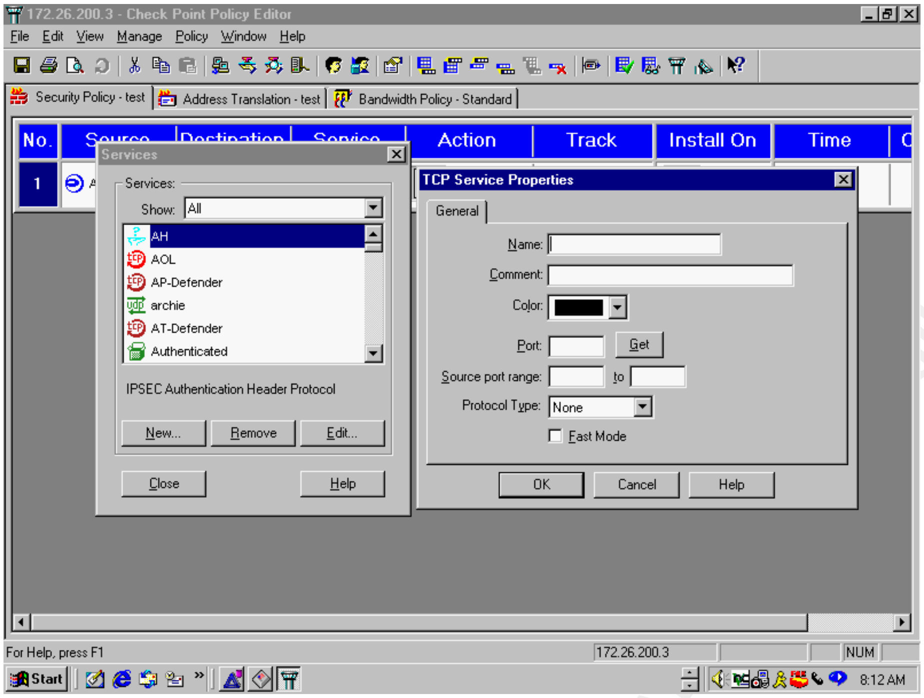
```
---- FW-1 CONNECTIONS TABLE ---

Src_IP          Src_Prt Dst_IP          Dst_Prt IP_prot Kbuf    Type    Flags           Timeout

172.26.200.3    179     172.26.200.1    1025    6       0       16385   ffff0600        3589/3600
172.26.200.3    179     172.26.200.2    4191    6       0       16385   ffff0600        3598/3600
134.141.143.75  1395    172.26.65.93    161     17      0       16386   ffffff00        7/40
134.141.143.75  1395    172.26.65.92    161     17      0       16386   ffffff00        7/40
134.141.143.75  1395    172.26.65.63    161     17      0       16386   ffffff00        7/40
134.141.143.75  1395    172.26.65.62    161     17      0       16386   ffffff00        2/40
134.141.143.75  1395    172.26.65.61    161     17      0       16386   ffffff00        2/40
134.141.143.75  2222    172.26.200.3    258     6       0       16385   ffff0800        3555/3600
134.141.143.75  2225    172.26.65.64    23      6       0       16385   ffffff00        17978/18000
```

Here we can see that a TELNET from 134.141.143.75 to 172.26.65.64 now has a timeout of 5 hours (18000 seconds), where all other connections retain the default properties.

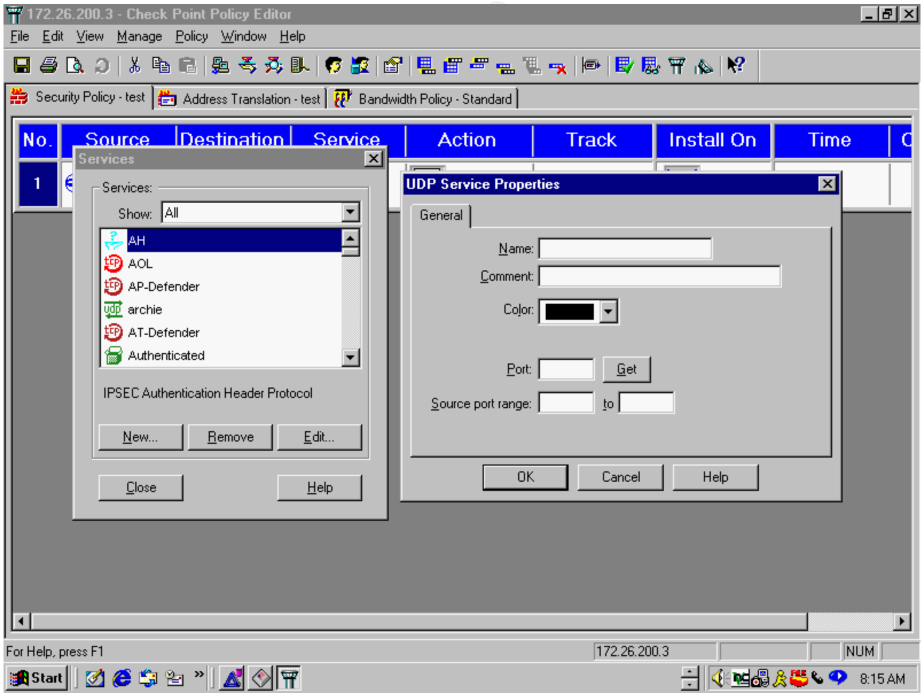
The INSPECT code that is used to allow\disallow packets through the firewall is extremely flexible, Check Point Support notes:

> This provides important system flexibility, allowing Check Point, as well as its technology partners and end-users, to incorporate new applications, services, and protocols, without requiring new software to be loaded. For most new applications, including most custom applications developed by end users, the communication-related behavior of the new application can be incorporated simply be modifying one of Firewall-1's built-in script templates via the graphical user interface. Even the most complex applications can be added quickly and easily via the INSPECT language.[6]

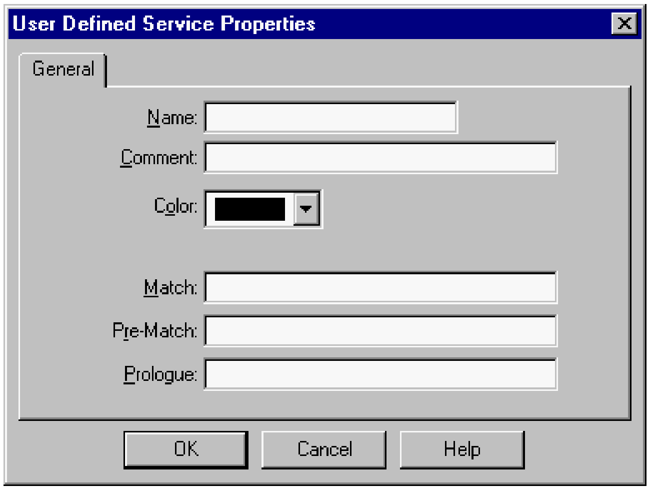
There is documentation and there are classes available on writing in INSPECT. This is beyond the scope of this paper, but it is important to note that this capability exists. However, there are pre-defined templates available in the policy editor that allows for the creation of custom services, those that are not included by default with the application. In the Policy Editor go to Manage then Services. Here you will see a listing of all default services. By selecting New, you will see a listing of the new services that you can work from. Some of the more useful ones are templates for TCP, UDP, RCP, and ICMP. There is also a template for a service of Other in which you can create your own service and what the firewall needs to match on for this service. Screen shots of those templates follow:



Here you can see the ability to specify a new TCP service based upon a specific port, or range of ports.

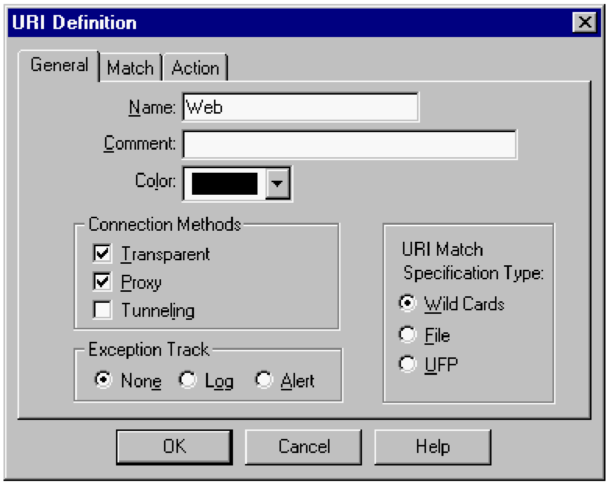


The above capture is similar to the TCP template, but for UDP. Again, you can specify the port or range of ports for a specific UDP service that the firewall will monitor for.



Above is the template for a service of other, meaning it is neither TCP nor UDP, nor does it fall into any of the other pre-defined templates. The Match, Pre-Match and Prologue fields are used to help define the service. What to enter into these fields will require knowledge of the INSPECT language, although there are documents available that are useful in assisting what to enter into these fields, especially for services that someone has already defined.

We will now touch upon the security servers within Firewall-1. There are three main security servers, which further enhance the ability of Firewall-1 to inspect packets. They are the HTTP, SMTP, and FTP security servers. There are also security servers for RLOGIN and TELNET, but these have no methods for content security. The security servers have the ability to look into the packet and accept and reject the packet based on more detailed information. Without going into great detail we will explore these three servers as they offer further security while inspecting packets. The HTTP security server allows the firewall to act as a proxy server for internal hosts. It has three modes in which it can do this, transparent, proxy and tunnel mode. The URI match field specifies how the URL is inspected, either by the firewall, via wildcards, from a UFP (URL Filtering Protocol) server or from a file. The file and UFP server are external sources, the wildcard allows for the inspection of the URL based on HTTP, FTP, Gopher and others. We can do this on a GET, POST, HEAD or PUT. Furthermore, we can strip specific tags from the web traffic such as scripts, ActiveX and Java. Additionally, we can specify that a third party CVP (Content Vectoring Protocol) scanner can monitor these packets. The SMTP security server has the ability to monitor SMTP traffic. It has the ability to re-write fields in the mail request and accept\reject traffic based upon size. It also has the ability to utilize a third party CVP scanner. The FTP security server has the ability to limit GETs and PUTs as well as utilize a CVP scanner. Below I will show the fields utilized to define the HTTP security server:


URI Definition

General | Match | Action

Name: Web

Comment:

Color:

Connection Methods

☑ Transparent

☑ Proxy

□ Tunneling

Exception Track

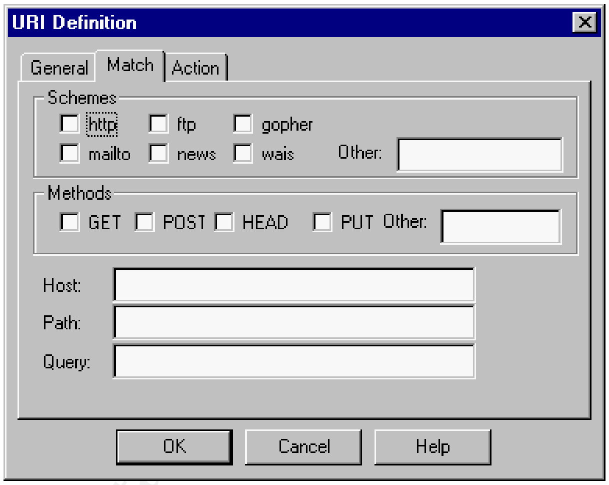
● None ○ Log ○ Alert

URI Match Specification Type:

● Wild Cards

○ File

○ UFP


URI Definition

General | Match | Action

Schemes

□ http □ ftp □ gopher

□ mailto □ news □ wais Other:

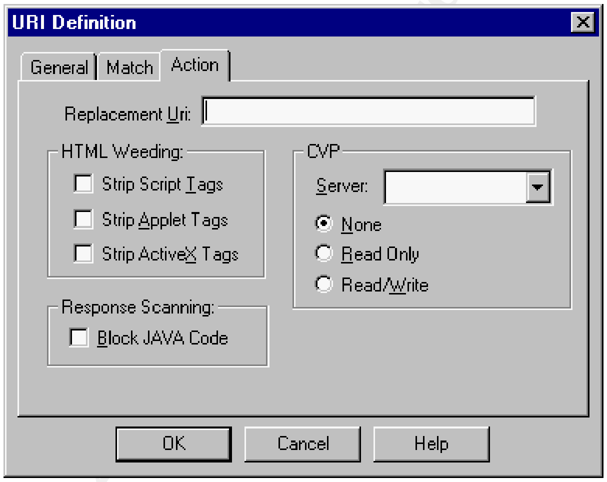
Methods

□ GET □ POST □ HEAD □ PUT Other:

Host:

Path:

Query:


URI Definition

General | Match | Action

Replacement Uri:

HTML Weeding:

□ Strip Script Tags

□ Strip Applet Tags

□ Strip ActiveX Tags

Response Scanning:

□ Block JAVA Code

CVP

Server:

● None

○ Read Only

○ Read/Write

Finally, as with the other types of firewalls, there are inherent issues with Check Point Firewall-1. As noted in the Check Point 4.1 SP2 release notes " With this service pack, only TCP SYN (TCP connection initiation) packets are allowed to be matched to the rulebase, NON-SYN connections that do not belong to any known connection are dropped."[7] This enhancement corrects an issue where the firewall would accept NON-SYN packets and create a connection for them, resulting in a breech of security. Now the firewall will only establish a connection based on a SYN packet.

In conclusion, even though this paper focuses on Check Point Firewall-1's state table it does not mean that it is superior or inferior to other types of firewalls, or even other types of firewall's utilizing stateful inspection. In depth research should be done before determining what type of firewall is right for your company. This will of course depend upon, budget, existing network architecture, and the level of security the organization requires. It is most likely that not one, but some combination of these three, or even all three will be implemented in most sites. The phrase "defense in depth" comes into play here. There is no real way to keep out a determined hacker, but there are ways to deter. The more layers that are in place to protect a company's assets, theoretically, the more secure things become, if only by weeding out the many whom just utilize known compromises, but really have no idea what they are doing.

References:

[1] Chapman, D. Brent, Simon Cooper, and Elizabeth D. Zwicky. Building Internet Firewalls 2nd Edition California: O'Reilly and Associates, Inc, 2000. 106.

[2] Check Point Support. "Stateful Inspection Firewall Technology." January 11, 2000.
URL: http://www.checkpoint.com/products/technology/page2.html

[3] Spitzner, Lance. "Fwtable.txt." July 3, 1999. URL: http://www.enteract.com/~lspitz/fwtable.txt

[4] Spitzner, Lance. "Understanding the FW-1 State Table." November 29, 2000.
URL: http://www.enteract.com/~lspitz/fwtable.html

[5] Unknown. "TCP Timeout for a Specific Service." January 9, 2000.
URL: http://www.phoneboy.com/faq/0203.html

[6] Check Point Support. "Stateful Inspection Firewall Technology." January 11, 2000.
URL: http://www.checkpoint.com/products/technology/page3.html

[7] Check Point Software Technologies. Check Point VPN-1/FireWall-1 Version 4.1 SP2 Release Notes July 2000. 5