Using Policy Analysis to Verify the SANS Firewall Checklist

By
David Hurst, CTO
Athena Security, Inc.

Firewall audits typically consider only security rules or ACLs to determine the network protection provided by a firewall. This common practice, however, completely overlooks the role played by network address translation and route rules in overall network protection. Ensuring that a firewall properly implements the desired security policy is essential to maintaining the security of the protected network zones. This paper explores the science of firewall behavior by examining the data flows through a firewall and how security is implemented in the context of such data flow.

We will use the **SANS Firewall Checklist** to provide a useful example of a set of guidelines for auditing a firewall security policy. The purpose of a firewall policy audit is to eliminate any false positives when determining compliance to a set of security requirements. Once armed with accurate data about the risks inherent in the firewall's configuration, security engineers can prioritize remediation activities and work towards the best firewall security posture.

The SANS Firewall Checklist contains many control items that ask questions about the firewall behavior such as:

- Are RFC 1918 and reserved IP addresses blocked as sources?
- Are insecure network services like HTTP, FTP, Telnet, SNMP, LDAP, Net BIOS, or X11 blocked?
- Are potentially risky but required services such as HTTPS, SMTP, and DNS isolated in a DMZ?

These questions can only be answered accurately by analyzing the firewall policy to determine what services are allowed to pass through the firewall from all sources to all destinations.

Simple inspection of the firewall rules is not sufficient to determine whether or not the security policy implemented by the firewall complies with the requirements in the SANS Firewall Checklist.

Articulating that policy as part of a firewall assessment, audit or hardening exercise turns out to be surprisingly difficult, and calculating it requires deep analysis. Simple inspection of the firewall rules is not sufficient to determine whether or not the security policy implemented by the firewall complies with the requirements in the SANS Firewall Checklist. The mere presence in the ruleset of a rule denying an insecure network service like Net BIOS does not ensure that the service is truly blocked from reaching a critical server. The order of the rules is significant and a rule early in the ruleset might mitigate or entirely negate the desired effect of a subsequent rule. Changing the order or inserting a rule in an incorrect location could lead to unintended consequences and even unexpected exposure to security threats. It is necessary to verify the IP addresses of sources and destinations that are affected by the rules. Rules that perform network address translation (or NATing) may cause the IP addresses or subnets that are actually permitted or denied to be different from the addresses that appear in the ACL rules. Rulebases that contain alternating allow and deny rules cause complex interactions between the rules. All of these factors make it very difficult achieve the best firewall policy, or to even know exactly what is going on in a firewall, without performing a deep policy analysis.

Components of Firewall Policy

Firewalls from different vendors vary widely in terms of the configuration languages used, the organization of rules and rulesets, and the interactions between different rulesets. These architectural differences must be considered when calculating firewall policy.

Fundamentally, a firewall implements a security policy as a sequence of rules that match for patterns of data in the communication traffic that is to be allowed or denied access into and out of the protected network zones. In general, these rules have the form < ***P***, *action* >, where ***P*** is a predicate describing what packets to match against and action describes an action to perform when the rule matches. The matching predicate specifies a set of values for network sources, destinations, and services that will match. Network sources and destinations are expressed as combinations of individual IP addresses, subnets (specified as IP address and netmask), and ranges of IP addresses. Network services are specified by the combination of protocol (e.g. TCP, ESP, SCTP) and port numbers for both source and destination. Only TCP and UDP use port numbers. ICMP uses type and code values and other protocols (e.g. GRE, ESP) have no additional parameters.

Rules are organized into rulesets. The order of the rules in a ruleset is important because matching proceeds sequentially through the ruleset. Packets not matched continue on to the next rule in the list until a match is found or the end of the ruleset is reached. The first rule to match an incoming packet causes its associated action to be applied. No further matching takes place within that ruleset. If a packet makes it to the end of a ruleset without matching any rules, a default action is applied. Firewalls may have multiple rulesets. Depending on the path through the firewall taken by the packet and the actions applied by matching rules, the packet may traverse multiple rulesets.

As a gateway mediating access between different network zones, a firewall will have network connections to each zone. These connections each go through a network interface. Network interfaces may be physical, connecting to a locally attached network, point-to-point, connecting to a single remote host, or virtual, connecting to a VPN. In all cases, the network interfaces represent a different path into or out of the firewall. When calculating a firewall policy, each possible path must be evaluated.

Complete analysis requires the following to be taken into account:

- **Rule order**
- **All possible paths (physical and virtual)**
- **Routing actions**
- **Transformation actions**

There are several different types of action that may be applied by a rule. Filtering actions determine if the matched packet will be accepted for further processing or simply dropped. Routing actions determine which network interface a packet must pass through to arrive at its destination. Transformation actions modify the source or destination address or service in the packet. The order in which these actions are applied is significant. For example, if a destination address translation rule is applied before the routing rules, a different route rule will likely match the transformed packet than would have matched the untransformed packet. The common actions applied by firewall rules are summarized in the following list:

allow	Permit the packet to pass on for further processing. No further matching is performed in the current ruleset.
deny	Prohibit the packet from further processing. All processing is abandoned for the packet.
route	Select the interface through which the packet must pass in order to leave the firewall. A packet may only ever match one route rule.
snat	Transform the source address and/or source port of the packet to a target address and/or port.
dnat	Transform the destination address and/or destination port of the packet to a target address and/or port.

Identifying Rule Conflicts

The first step to analyzing a firewall policy is to understand the relationships between all of the rules in each ruleset of the configuration. Because rule order matters, the relative ordering and interaction between rules determines the policy that will actually be applied to a given packet. The intention of a specific rule that does match an incoming packet may not be fulfilled as expected because of a prior matching rule with a different action.

The following simple example, taken from a Cisco ASA configuration, illustrates a simple conflict between two rules.

```
access-list inbound deny tcp any 10.20.1.0 255.255.255.0 eq https
access-list inbound permit tcp any host 10.20.1.89 eq https
```

The first rule denies inbound access to HTTPS service for all hosts on the 10.20.1.0/24 subnet. The second rule permits the same HTTPS service to a single host on the same subnet. However, HTTPS traffic will never be allowed to that destination host because it is denied by the first rule, which takes precedence. This is termed a *shadowing* conflict because the first rule hides or shadows the second rule and prevents the second rule from having any effect. So even though there is an ACL that explicitly permits that HTTPS traffic, the policy actually implemented by the firewall does not.

The preceding example involved only two rules and was pretty easy to spot since the rules were adjacent. Rule conflicts that involve more than two rules can be more difficult to identify, especially when the involved rules are widely separated in the configuration. Consider the following example involving three rules and a service object definition.

```
object-group service web-svcs tcp
  port-object eq www
  port-object eq https

access-list inbound permit tcp any host 192.168.23.10 eq www
access-list inbound permit tcp any host 192.168.23.10 eq https
access-list inbound deny tcp any host 192.168.23.10 object-group web-svcs
```

The first two rules permit WWW and HTTPS services to a destination host and the third rule denies a service group called `web-svcs` to the same destination. In this case, it is the combination of the first two rules that causes the third rule to be shadowed. As a result, the intention of the third rule to block web services packets to the destination host will not be fulfilled.

In the above examples of shadowed rules, the effect of a rule was completely negated by one or more preceding rules. A more insidious type of rule conflict involves overlaps between the matching ranges of different rules that cause the intended effect of a rule to be only partially fulfilled in unexpected ways. Consider the following example, which attempts to satisfy the requirement in the SANS Firewall Checklist to block inbound packets with (potentially spoofed) private (RFC-1918) IP source addresses.

A more insidious type of rule conflict involves overlaps between the matching ranges of different rules that cause the intended effect of a rule to be only partially fulfilled in unexpected ways.

```
object-group network rfc1918
  network-object 10.0.0.0 255.0.0.0
  network-object 172.16.0.0 255.240.0.0
  network-object 192.168.0.0 255.255.0.0

access-list inbound permit icmp any any
access-list inbound deny ip object-group rfc1918 any
```

A simple review of the ACLs in this configuration would find a rule denying these illegal addresses and might conclude that the SANS requirement to block them was satisfied. The reality is that an entire class of packets with the illegal addresses would be allowed by the preceding rule. This is termed a *correlation* conflict because the matching range of the ICMP rule allowing packets from any source address intersects with the matching range of the RFC-1918 rule and compromises the intention to block the RFC-1918 addresses.

In general, finding all the rule conflicts in a ruleset requires comparing every rule in the ruleset with every combination of preceding rules in the ruleset and identifying the specific relations that cause rule conflicts. The matching predicate $\boldsymbol{P}$ consists of matching parameters $\{ \boldsymbol{p}_i .. \boldsymbol{p}_j \}$ that describe source addresses, destination addresses, and services to match. The relation between a matching parameter $\boldsymbol{p}_n$ in rule $\boldsymbol{R}_X$ and the corresponding parameter in another rule $\boldsymbol{R}_Y$ can be either equal, subset, superset, or disjoint. For example, the IP addresses 192.168.10.35 and subnet 192.168.12.0/24 are disjoint, but both are a subset (or contained by) of 192.168.8.0/21. There are four types of rule relations to consider.

- **Shadowing.** A rule is shadowed when every packet that could match it is matched by some previous rule and the previous rule has a different action. The shadowed rule will never have an effect on policy.
- **Correlation.** Two rules are correlated with each rule matches some of the packets matched by the other and they have different actions. The action performed on the traffic that matches the intersection of the two rules is dependent on the ordering of the rules. The subsequent rule will not be able to apply its policy to the packets that match the intersection.
- **Generalization.** A rule is a generalization of a previous rule if they have different actions and the subsequent rule matches a superset of all the packets matched by the preceding rule. The preceding rule represents an exception to the policy applied by the following rule.

- **Disjoint.** Two rules are disjoint if they match no packets in common. Each rule has no effect on the policy applied by the other rule.

Calculating Firewall Policy

A firewall policy describes the set of packets that will be passed through the firewall, the paths taken by those packets, and any transformations that might be applied to the packets. Individual rule conflicts identify why and where the actual policy applied by the firewall is different than the action of any given individual rule might imply. Calculating the complete firewall policy involves determining the effect of all the rules in aggregate for all sources to all destinations for all services along all paths.

Initially, we will analyze the policy for a single path in the firewall from an ingress interface to an egress interface. Depending on the firewall architecture, one or more rulesets may apply along this path. The policy result for a given path is the aggregate effect of all rules from each ruleset encountered along the path. This is calculated by collecting the matching parameters of each rule and combining them with the policy result from the preceding rules.

Calculating the complete firewall policy involves determining the effect of all the rules in aggregate for all sources to all destinations for all services along all paths.

The analysis begins with an input set $\boldsymbol{I}$ consisting of all of the packets that might possibly enter the ingress interface. To be complete, the input set must enumerate all packets for every possible source address, every possible destination address, and every possible service. The policy result for the path consists of an allow set $\boldsymbol{A}_{path}$ of packets that can possibly reach the exit interface after encountering the rules from each ruleset along the path and a deny set $\boldsymbol{D}_{path}$ of packets that are blocked from reaching the exit interface. The union of $\boldsymbol{A}_{path}$ and $\boldsymbol{D}_{path}$ is the original input set $\boldsymbol{I}$.

For the *j*th rule $< \boldsymbol{P}_j, action_j >$ in a ruleset, we define the current state $\boldsymbol{S}$ as $<\boldsymbol{A}_j, \boldsymbol{D}_j>$, where $\boldsymbol{A}_j$ and $\boldsymbol{D}_j$ denote the set of packets accepted and denied before the *j*th rule. We will use $\boldsymbol{R}_j$ to indicate the collection of remaining packets that could reach the *j*th rule. Then for the first rule in the path, we have the initial value of $\boldsymbol{A}_1 = \boldsymbol{D}_1 = \{ \}$ and $\boldsymbol{R}_1 = \boldsymbol{I}$. For each subsequent rule, the current state $\boldsymbol{S}$ is updated by the following state transformation until the end of the path is reached.

$$\text{If } < \boldsymbol{P}_j, \text{accept} > \quad \rightarrow \quad < \boldsymbol{A}_{j+1}, \boldsymbol{D}_{j+1} > \; = \; < \boldsymbol{A}_j \cup (\boldsymbol{R}_j \cap \boldsymbol{P}_j), \boldsymbol{D}_j >$$

$$\text{If } < \boldsymbol{P}_j, \text{deny} > \quad \rightarrow \quad < \boldsymbol{A}_{j+1}, \boldsymbol{D}_{j+1} > \; = \; < \boldsymbol{A}_j, \boldsymbol{D}_j \cup (\boldsymbol{R}_j \cap \boldsymbol{P}_j) >$$

$$\boldsymbol{R}_{j+1} = \boldsymbol{R}_j - (\boldsymbol{R}_j \cap \boldsymbol{P}_j)$$

For an accept rule, the allow set $\boldsymbol{A}$ is updated with the set of remaining packets that matched $\boldsymbol{P}$. The deny set does not change. For a deny rule, the deny set $\boldsymbol{D}$ is updated with the set of remaining packets that matched $\boldsymbol{P}$. The allow set does not change. The set of remaining packets $\boldsymbol{R}$ is updated with the packets that did not match $\boldsymbol{P}$.

At the end of a ruleset a default action is applied to all packets that did not match any rule. Consequently, all packets will either be accepted or denied by the ruleset, so $\boldsymbol{A} \cup \boldsymbol{D} = \boldsymbol{I}$. The allow set $\boldsymbol{A}$ from the *n*th ruleset along the path is used as the input set $\boldsymbol{I}$ for the *n*th+1 ruleset. The process is repeated for each ruleset along the path until the egress interface is reached.

Example Firewall Policy Analysis

To illustrate how the policy calculation goes, we will use examples from the sample Cisco PIX configuration in Appendix A. This PIX configuration specifies a three-port firewall, connecting an Internal zone, a DMZ zone, and an External zone. The firewall has the following interface configurations.

```
interface ethernet0 auto
interface ethernet1 auto
interface ethernet2 auto
nameif ethernet0 outside security0
nameif ethernet1 inside security100
nameif ethernet2 dmz security50
ip address outside 62.59.14.166 255.255.255.248
ip address inside 172.16.0.1 255.255.0.0
ip address dmz 192.168.1.1 255.255.255.0
```

Consider the path traversing the firewall from the External zone to the DMZ zone. The External zone is connected to the firewall by the `outside` interface, which has an IP address of 62.59.14.166 and a netmask of 255.255.255.248. Since packets are arriving at the firewall from the Internet, the source addresses for packets in the initial set ***I*** can be any IP address. By convention, the notation 0.0.0.0/0 represents the set of all IP addresses.

The packets are subjected to the `acl_outside` filtering ruleset. This ruleset consists of the following rules (and object group definitions).

```
object-group service web_svcs tcp
  port-object eq www
  port-object eq https
object-group service mail_svcs tcp
  port-object eq pop3
  port-object eq imap4
object-group service inet_svcs tcp
  group-object mail_svcs
  group-object web_svcs
  port-object eq domain
object-group network dmz_svrs
  network-object host 62.59.14.163
  network-object host 62.59.14.164

access-list acl_outside deny tcp any 62.59.15.0 255.255.255.0 eq smtp
access-list acl_outside permit tcp any host 62.59.15.110 eq smtp
access-list acl_outside permit tcp any object-group dmz_svrs object-group web_svcs
access-list acl_outside permit tcp any host 62.59.14.165 object-group mail_svcs
access-list acl_outside permit tcp host 208.121.58.99 host 62.59.14.165 eq smtp
access-list acl_outside permit tcp host 66.93.145.29 host 62.59.14.165 eq smtp
access-list acl_outside permit tcp host 67.28.57.31 host 62.59.14.165 eq smtp
```

We will represent a policy result as a set of tuples having five fields containing a single value or range of values for protocol, source address, source port, destination address, destination port. Each tuple represents the set of packets matching the given values and is disjoint from every other tuple. There is no order dependency between the tuples. Applying the policy analysis algorithm described above, the allow set ***A*** of the policy result for the `acl_inside` ruleset consists of the following tuples.

```
( tcp, 0.0.0.0/0, any, 62.59.14.163, http )
( tcp, 0.0.0.0/0, any, 62.59.14.164, http )
( tcp, 0.0.0.0/0, any, 62.59.14.163, https )
( tcp, 0.0.0.0/0, any, 62.59.14.164, https )
( tcp, 66.93.145.29, any, 62.59.14.165, smtp )
( tcp, 66.93.145.29, any, 62.59.14.165, pop3 )
( tcp, 67.28.57.31, any, 62.59.14.165, smtp )
( tcp, 67.28.57.31, any, 62.59.14.165, pop3 )
( tcp, 0.0.0.0/0, any, 62.59.14.165, imap4 )
```

Effect of Address Translation and Routing

This result above is only an intermediate calculation and is not the complete policy result for the path from `outside` to `dmz`. After the `acl_outside` ruleset is applied, the policy result set still has to be processed through the network address translations (NATs) to determine how the packets may be modified by the firewall and the routing table to determine what destinations are valid. The following NAT commands from the sample configuration apply to the path we are analyzing.

```
static (dmz,outside) 62.59.14.163 192.168.1.3 netmask 255.255.255.255 0 0
static (dmz,outside) 62.59.14.164 192.168.1.4 netmask 255.255.255.255 0 0
static (dmz,outside) 62.59.14.165 172.16.1.5 netmask 255.255.255.255 0 0
```

These have the effect of transforming all packets traversing the path with destination addresses of 62.59.14.16{3,4} to 192.168.1.{3,4} and 62.59.14.165 to 172.16.1.5.

The sample firewall has a trivial routing table, with only a default route on the `outside` interface. In the absence of additional routes, all packets will be forwarded to the outside interface except for those with destinations in the networks locally connected to each interface. The `dmz` interface provides a local route for the DMZ subnet, 192.168.1.0/24 and the `inside` interface provides a local route for the Internal subnet, 172.16.0.0/16. Only tuples with a destination address contained by the subnet 192.168.1.0/24 will arrive at the `dmz` interface and enter the DMZ zone. The following tuples describe the policy result for the path from `outside` interface to the `dmz` interface.

```
( tcp, 0.0.0.0/0, any, 192.168.1.3, http )
( tcp, 0.0.0.0/0, any, 192.168.1.4, http )
( tcp, 0.0.0.0/0, any, 192.168.1.3, https )
( tcp, 0.0.0.0/0, any, 192.168.1.4, https )
```

Notice that the destination addresses have been replaced by the NAT'ed values 192.168.1.{3,4,5}. The tuples for the mail services (smtp, pop3, imap4) were all removed from the policy result, by the effect of the address transformation and the local route for the Internal zone.

Completing the SANS Firewall Checklist Audit Using Policy Results

At the beginning of this paper, we asked several questions of firewall behavior based on recommendations in the SANS Firewall Checklist. In addition to the policy for the path from `outside` interface to `dmz` interface, we will also use the policy for the path from `outside` to `internal`. This additional policy result is calculated in the method we have described above and contains the following tuples in the allow set.

```
( tcp, 66.93.145.29, any, 172.16.1.5, smtp )
( tcp, 67.28.57.31, any, 172.16.1.5, smtp )
( tcp, 208.121.58.99, any, 172.16.1.5, smtp )
( tcp, 66.93.145.29, any, 172.16.1.5, pop3 )
( tcp, 67.28.57.31, any, 172.16.1.5, pop3 )
( tcp, 208.121.58.99, any, 172.16.1.5, pop3 )
( tcp, 0.0.0.0/0, any, 172.16.1.5, imap4 )
```

The checklist questions can now be easily answered using the policy results that we have calculated for the paths from the External zone to the DMZ and Internal zones.

1. ***Are RFC-1918 and reserved IP addresses blocked as sources?*** These IP addresses include the following ranges:

10.0.0.0	to	10.255.255.255
127.0.0.0	to	127.255.255.255
172.16.0.0	to	172.31.255.255
192.168.0.0	to	192.168.255.255
169.254.0.0	to	169.254.255.255

   By inspection, we can see that these addresses are all allowed into the DMZ zone for the web services and into the Internal zone for the IMAP4 service.

2. ***Are insecure network services like HTTP, FTP, Telnet, SNMP, LDAP, Net BIOS, or X11 blocked?*** According to the policy result, HTTP is allowed into the DMZ but nowhere else, and all of the other insecure services are not allowed.

3. ***Are potentially risky but required services such as HTTPS, SMTP, and DNS isolated in a DMZ?*** The policy result indicates that HTTPS service is only allowed into the DMZ zone, but SMTP is not isolated and is allowed into the Internal zone.

Saving Time and Money with Automated Tools for Policy Analysis

As we have seen, these and other questions raised by the SANS Firewall Checklist can easily be answered by calculating the firewall policy and then querying the policy result. The policy result provides accurate answers about how the firewall behaves. Calculating the policy result can be a very intensive process. Even for the small configuration in the sample firewall, the best firewall policy analysis must handle the complexities of overlapping rules, network address translations, and routes.

The risk of not computing the firewall policy is a misleading and inaccurate assessment of the security risks embodied in the firewall's configuration. Policy calculations are essential to determining whether a specific service or packet is allowed or denied through the firewall.

In the real world, corporate and enterprise firewalls have rulesets numbering in the thousands of rules, multiple network interfaces, and hundreds of network objects that contain hundreds of individual hosts or subnets. The complexity involved rapidly multiplies beyond any possibility of manual analysis.

Automated firewall analysis tools provide for a more reliable, consistent and affordable way to eliminate security risks and maintain a clean rulebase. Until now, firewall auditors and network administrators did not have access to the accuracy afforded by full policy analysis and had to settle for tools that use pattern matching to scan firewall rules, even though they produced inaccurate results.

Athena FirePAC, available from Athena Security, provides automated policy analysis that identifies the full range of rule conflicts, redundant rules and unused rules that can be removed from the configuration. It can answer any query about the policy result to determine what services are allowed through the firewall, what services are allowed to a given destination, or what happens when a given packet hits the firewall. FirePAC also performs an automated firewall audit that includes over 120 cases of risky or potentially dangerous policies. For PCI compliance reviews, FirePAC runs customized policy calculations to the PCI Zone and produces pass/fail determinations of firewall related control items. Athena FirePAC is available for a free 30 day trial and can be downloaded directly from http://www.athenasecurity.net.

References

Al-Shaer, E. and Hamed, H., "Taxonomy of Conflicts in Network Security Policies", *IEEE Communications Magazine*, Vol. 44, No. 3, March 2006.

Al-Shaer, E. and Hamed, H. "Discovery of Policy Anomalies in Distributed Firewalls." *IEEE Infocom*, March 2004.

Mayer, A., Wool, A., and Ziskind, E.. "Offline firewall analysis". In *International Journal of Information Security*, Springer-Verlag, Berlin, 2006.

Naidu, K. *SANS Firewall Checklist.* http://www.sans.org/score/checklists/FirewallChecklist.pdf.

Schuba, C., Lyles, B. and Spafford, E. "A Reference Model for Firewall Technology" SPARTAN Symposium, March. 1997.

Yuan, L., Mai, J., Su, Z., Chen, H., Chuah, C., and Mohapatra, P. "FIREMAN: A Toolkit for FIREwall Modeling and ANalysis." In *Proceedings of IEEE S&P 2006*, Oakland, California, May 21-24, 2006.

Appendix A: Sample Cisco PIX Firewall Configuration

```
PIX Version 6.3(3)
interface ethernet0 auto
interface ethernet1 auto
interface ethernet2 auto
nameif ethernet0 outside security0
nameif ethernet1 inside security100
nameif ethernet2 dmz security50
hostname mycorp-pix
object-group service web_svcs tcp
  port-object eq www
  port-object eq https
object-group service mail_svcs tcp
  port-object eq pop3
  port-object eq imap4
object-group service inet_svcs tcp
  group-object mail_svcs
  group-object web_svcs
  port-object eq domain
object-group network web_svrs
  network-object host 192.168.1.3
  network-object host 192.168.1.4
object-group network dmz_svrs
  network-object host 62.59.14.163
  network-object host 62.59.14.164
object-group network db_svrs
  network-object host 172.16.1.200
  network-object host 172.16.1.210
access-list acl_outside deny tcp any 62.59.15.0 255.255.255.0 eq smtp
access-list acl_outside permit tcp any host 62.59.15.110 eq smtp
access-list acl_outside permit tcp any object-group dmz_svrs object-group web_svcs
access-list acl_outside permit tcp any host 62.59.14.165 object-group mail_svcs
access-list acl_outside permit tcp host 208.121.58.99 host 62.59.14.165 eq smtp
access-list acl_outside permit tcp host 66.93.145.29 host 62.59.14.165 eq smtp
access-list acl_outside permit tcp host 67.28.57.31 host 62.59.14.165 eq smtp
access-list acl_dmz permit tcp any object-group db_svrs eq 3306
access-list acl_dmz permit tcp any host 172.16.3.45 eq http
access-list acl_inside permit tcp any any object-group inet_svcs
access-list acl_inside permit tcp any object-group web_svrs eq ssh
access-list acl_inside permit tcp any object-group web_svrs eq ftp
access-list acl_inside deny ip any any
access-list nonat permit ip 172.16.0.0 255.255.0.0 192.168.1.0 255.255.255.0
global (outside) 1 62.59.14.162
global (outside) 2 62.59.14.163
nat (inside) 0 access-list nonat
nat (inside) 1 172.16.0.0 255.255.0.0 0 0
nat (dmz) 2 192.168.1.0 255.255.255.0 0 0
static (dmz,outside) 62.59.14.163 192.168.1.3 netmask 255.255.255.255 0 0
static (dmz,outside) 62.59.14.164 192.168.1.4 netmask 255.255.255.255 0 0
static (inside,outside) 62.59.14.165 172.16.1.5 netmask 255.255.255.255 0 0
access-group acl_outside in interface outside
access-group acl_inside in interface inside
access-group acl_dmz in interface dmz
route outside 0.0.0.0 0.0.0.0 62.59.14.161
ip address outside 62.59.14.166 255.255.255.248
ip address inside 172.16.0.1 255.255.0.0
ip address dmz 192.168.1.1 255.255.255.0
Cryptochecksum:74151f0b71a7d641d691819297fc6c74
```