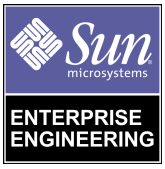
Solaris™ Operating Environment Network Settings for Security

By Alex Noordergraaf and Keith Watson - Global Enterprise Security Service

Sun BluePrints™ OnLine - December 1999



http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-4049-10
Revision 2.0, 11/15/01
Edition, December 1999

Solaris™ Operating Environment Network Settings for Security

The Solaris™ Operating Environment is a general purpose operating system with many configurable, low-level network options that are applicable to security. Some of these should be adjusted to strengthen the security posture of a Solaris system. This article includes descriptions of various known attack methods, not as a step-by-step guide for attackers, but to show system and network administrators the need to set or change a particular network option.

Various trade-offs must be made when enhancing Solaris security. A balance is needed between system manageability and security. The trade-offs are discussed in this article. Not all network security configurations mentioned can be used in all environments. Where changing a particular network setting adversely affects the default system operation, the side effects are discussed.

This article does not discuss high-level network security. High-level network security involves configuring `inetd`, NFS, NIS/NIS+, RPC, DNS, and other application-level services. That topic will be addressed in a future Sun BluePrint™ OnLine article.

The information in this article is applicable to Solaris 2.5.1, 2.6, and 7 Operating Environment. Other versions of the Solaris Operating Environment may have some similar settings. Some investigation will be needed prior to using the settings in this article with other versions of Solaris Operating Environment.

Sun Cluster™ 2.x

The network settings discussed in this article are not supported with Sun Cluster™ 2.x software unless specific arrangements have been made with your Sun Professional Services consultant.

Enabling IP Strict Destination Multihoming may impact the operation of Sun Cluster 2.x systems. The latest version of the `nddconfig` script described in Appendix A will disable IP Strict Destination Multihoming if the Sun Cluster 2.x software is installed.

The Solaris `ndd` Command

Several of the network settings discussed in this article are configured using the Solaris `ndd` command. It is used to examine and set kernel parameters, namely the TCP/IP drivers. Most kernel parameters accessible through `ndd` can be adjusted without rebooting the system. To see which parameters are available use the following `ndd` commands:

```
# ndd /dev/arp \?
# ndd /dev/icmp \?
# ndd /dev/ip \?
# ndd /dev/tcp \?
```

These commands list the parameters for the ARP, IP, ICMP, and TCP drivers. "`\?`" is required to prevent the shell from interpreting the "?" as a special character. Using "`\?`" will list all parameters for the driver and indicate whether the parameter is read only, write only, or read and write. The current parameter value or status information can be read by specifying the driver and parameter names.

```
# ndd /dev/arp arp_debug
0
```

This example shows the output of a `ndd` command examining the debugging status of the ARP driver. (The output "0" indicates that the option is disabled.)

ndd-specified parameter values are integers with “0” meaning disable, “1” meaning enable, or a large integer to set a time or size value. Setting parameters requires the “-set” option, the driver name, the parameter name, and the new value. For example, to enable debugging mode in the ARP driver use the following ndd command:

```
# ndd -set /dev/arp arp_debug 1
```

A Note About Parameter Changes

ndd parameter documentation is not available from Sun. This is a known problem. Setting driver parameters involves making trade-offs. Most parameters involve changing the default Solaris configuration. The default settings are optimal for most situations. Adjusting parameters may affect normal system operation, so Sun does not encourage making parameter changes. Sun may also change the names of parameters in future versions of Solaris software.

All ndd parameter changes suggested in this article include a discussion of the trade-offs, where appropriate. Some settings change the expect operation of systems; these are noted. Most of these parameters are actively on production systems at customer sites. The Global Enterprise Security Service (GESS) group in Sun Professional Services has configured several customer sites using the parameter changes mentioned in this article. No problems have been reported.

While Sun sometimes alters parameter names between versions of Solaris Operating Environment, the parameters discussed in this article are used consistently in versions 2.5.1, 2.6, and 7, with only one exception which is documented. An `init` script to set most of the ndd options in this article is included.

Ultimately, you must decide which settings are appropriate for a specific computing environment.

ARP

The Address Resolution Protocol (ARP) is the protocol used to map 32-bit IP addresses to the address scheme used by the data-link layer. The data-link layer (sometimes called the network link layer), which consists of the operating system device driver and corresponding network interface card, is responsible for dealing with the physical transport media. Each network interface has a unique hardware address, typically assigned by the manufacturer. Sun ethernet interfaces acquire their hardware address from the system PROM. This means that a Sun system with

multiple Ethernet interfaces will have the same hardware address for each interface. A Sun Quad FastEthernet™ card has four unique hardware addresses assigned to each interface. It is also possible to configure the card to use the hardware address from the system PROM. Refer to the Sun Quad FastEthernet card documentation for more information.

ARP is often referred to as a “dynamic” protocol. This is due to the fact that its operation occurs automatically. The protocol works in the background, without concern to the application user or even network administrator. It is the dynamic nature of ARP which causes security issues.

For the purposes of this discussion, we will use Ethernet (IEEE 802.3). (Token ring and FDDI have similar schemes.)

ARP works by sending an address request and collecting the response to create its mapping of addresses. The hardware addresses are only needed for hosts on the local network. At the lowest level, the Ethernet driver needs the hardware address of the remote system to which it will send a packet. When it does not have that address, it “broadcasts” a request for the missing address. This request, called an “ARP request”, contains the IP address of the host in question and is sent to all systems on the local network. A system may respond with a reply, called an “ARP reply”, which contains the host IP address and hardware address. The response received is used to build a table of IP addresses and hardware addresses.

In the Solaris Operating Environment kernel, there are two tables that maintain the addresses. One table, maintained by the ARP layer, is called the “ARP cache”. It provides some efficiency to the protocol. When a hardware address is requested by the IP layer, the ARP cache is checked first. If the address information does not exist in the local cache, then a ARP request is sent, and the reply is processed. Systems running on Solaris Operating Environment also add unsolicited address information to the ARP cache. This type of entry is special because it was not directly requested. These entries are kept in case the IP layer requests them. After a period of time, all unsolicited entries are deleted from the cache. The default timeout value for unsolicited entries is five minutes and can be adjusted.

The other table for host address mappings is maintained by the IP layer. It is built from information supplied by the ARP layer. This table contains IP routing information for active connections. The IP layer requests hardware addresses from the ARP layer. The address information returned by the ARP layer is inserted into an entry in the IP routing table. By default, an entry will expire 20 minutes after it was added to the table, regardless if it is currently in use.

Another feature of the protocol is called “gratuitous ARP”. This occurs when a host broadcasts an ARP request for its own hardware address. A Solaris system does this at boot time. It is used to detect if another system is using its IP address, indicating a misconfigured system. The other use of gratuitous ARP is to send updated hardware address information. Systems that receive requests like this will automatically update the hardware address information for that host.

ARP Attacks

Several ARP problems can affect a system's expected operation. The TCP/IP network protocol suite requires correct hardware address information to ensure proper deliver of data. An Ethernet frame with an incorrect hardware address will not be processed by the intended system. All hardware address information is collected by the ARP layer. It gathers this information as it is needed and accepts information sent to it. The protocol is also stateless. The problems lie in the fact that the protocol allows any host to provide its own address information (correct or not). One system may provide information on the behalf of another system. Address information received by the ARP layer is processed whether it was directly requested or not. Additionally and more importantly, all address information received by a system is believed to be accurate.

There are two basic types of attacks possible with ARP: denial of service and spoofing. These attacks can prevent normal operations and can be used to compromise other systems on the local network. A denial of service attack will prevent one system from exchanging packets with another on the same network. This makes the system appear to be "off the network". A spoofing attack allows one system to masquerade as another system.

These attacks take advantage of the dynamic nature of the protocol. The simplest attack is denial of service. There are two forms to this attack: local and remote. Locally, an attacker that has root privilege can insert bogus address information in the ARP cache. Packets destined for systems with bogus hardware addresses will not be received by the intended system. An attacker can feed a remote system incorrect address information as well. This is known as cache poisoning. Since the ARP layer always trusts the information it receives, wrong information can be inserted and current ARP entries can be corrupted. An attacker may use the "publish" feature of the ARP layer to broadcast incorrect information about other systems. If two ARP replies are received, at least one reply will be used. It may be the correct one, or it may not. This situation can spread discord throughout the local network and can be difficult to diagnose. ARP packet tools may be used to generate bogus gratuitous ARP requests or send ARP replies to systems to corrupt address information.

The second type of attack is more serious because it can be used to compromise remote systems on the local network. By masquerading as another system, it is possible for an attacker to exploit a trust relationship to gain entry to a target system using a method calling "ARP spoofing". If one system trusts another (for example, through a ".rhosts" file configuration), then it is possible for an attacker to feed false hardware address information to the trusting system and convince it that packets from the attacking system are from the trusted system. For example, host A trusts host B. An attacker on host C wants to log into host A. First, the attacker must disable host B to prevent it from responding to ARP requests. The attacker then configures host C's IP address on a logical network interface and sends an ARP

reply to host A containing host B's IP address and host C's hardware address. As discussed previously, host A will update the address information from the ARP reply. Host C now acts as host B, and the attacker can now log into host A.

Defenses

Defending against ARP attacks is difficult. Changing the protocol in significant ways would break compatibility with all TCP/IP based systems on a network. Attempting to eliminate the dynamic nature of the protocol makes network administration a nightmare. However, there are some things that can be done to improve security on the local area network.

If false entries are inserted into the ARP and IP routing tables, there are two ways they can be deleted:

- Entries can be deleted manually using the `arp -d` *host_entry* command.
- The entries will also timeout and be deleted by the system.

RFC 826, which defines ARP, specifies that ARP cache entries should be deleted automatically after a reasonable period of time. The default timeout values for unsolicited ARP cache entries is five minutes. IP routing table entries timeout after 20 minutes. In Solaris software, these timeout intervals can be altered.

```
# ndd -set /dev/arp arp_cleanup_interval 60000
# ndd -set /dev/ip ip_ire_flush_interval 60000
```

The timeout interval is specified in milliseconds. 60000 milliseconds is one minute. Both these commands reduce the timeout period for the ARP cache and IP routing table. Entries will be deleted at a faster rate. This may slow down an ARP attack since bogus entries do not remain as long. These commands are available in the system init script provided in "Appendix A". The major side effect of this change is a greater number of ARP requests and replies. Do not use on congested networks.

Another alternative is to create static hardware address entries in the ARP cache. This solution is effective but breaks the dynamic nature of ARP, can increase maintenance costs, and may not be effective in most environments. A static entry in the ARP cache is a permanent mapping of an IP address to hardware address. These entries can be loaded at system boot time. Create a file containing IP addresses and the corresponding hardware addresses, similar to the following:

```
gort.eng.sun.com 08:00:20:ba:a3:c5
olympics.eng.sun.com 08:00:20:4d:6d:30
switchblade.eng.sun.com 08:00:20:b3:48:57
```

Load the contents of this using `arp -f` *file*. These entries are now marked as permanent entries in the cache and cannot be deleted by timeout nor overridden by unsolicited information. They can be deleted by using the `arp -d` *host_entry* command. If the network interface card is replaced on one of the systems in the static ARP listings, the ARP cache on all systems using static ARP entries must be updated or further communication will not be possible. This solution may not be appropriate in environments which frequently change equipment.

It is also possible to disable ARP completely for an interface. This means that the network interface will no longer send ARP requests nor process ARP replies. To disable ARP processing, use the `ifconfig` *interface* `-arp` command. Every system that disables ARP must have static ARP entries. Also, any system that might need to communicate with systems without ARP will need static ARP entries (such as routers). This solution is not recommended for most environments because of the high administrative costs. It may be effective with a small number of machines that need to communicate with each other and do not interact with other systems on the local network.

IP

The Internet Protocol (IP) is the lower level protocol that provides bulk data transport. It is connectionless and makes no provisions for reliable delivery. The configuration parameters discussed in this article are controlled by the Solaris IP driver.

IP Forwarding

IP forwarding is the process of routing packets between network interfaces on one system. A packet that arrives on one network interface and is addressed to a host on a different network is forwarded to the appropriate interface. Routers handle a majority of this work, but a computer with multiple network interfaces can do this as well.

A Solaris system with more than one configured network interface forwards IP datagrams between network interfaces. It functions as a router. This is the default action in Solaris Operating Environment to allow quick system configuration.

Systems with multiple interfaces can be configured to function as “multihomed” servers. A multihomed system has several network interfaces, each with a separate IP address. It is not intended to route or forward packets but handles network requests from multiple, directly-attached networks. For example, NFS operates quickly when the server is connected to the same network as its clients. A large NFS

server may serve clients on several networks. The server response is faster and the throughput is greater when the NFS server is directly attached to each client network it serves.

Systems that allow packet forwarding are targets for attackers as they provide access to other systems and networks. Some of these may not normally be accessible through routers. Private, non-routed networks can be served by a multihomed server. If IP forwarding is enabled on the server, this network is now reachable. Internal firewalls that protect access to a small set of systems can be bypassed by forwarding packets through a multihomed server that is directly attached to the protected internal network.

Packet forwarding is easily be disabled on a Solaris system. Simply creating a file named `/etc/notrouter` will disable IP forwarding at boot time. IP forwarding can also be switched on or off while the system is operating, using the `ndd` command. Use this command to disable IP forwarding:

```
# ndd -set /dev/ip ip_forwarding 0
```

An argument of 1 instead of 0 will enable IP forwarding. An attacker may attempt to compromise the system just to enable packet forwarding to gain access to normally inaccessible systems. This is another reason to make sure all servers are secure.

Strict Destination Multihoming

Strict destination multihoming prevents packet spoofing on nonforwarding multihomed systems. A Solaris system with IP forwarding disabled and strict destination multihoming enabled will ignore packets sent to an interface from which it did not arrive. This prevents attackers from creating packets destined for networks only connected to a multihomed server that does not forward packets. The system is aware of which interface on which a packet arrives. If a packet appears to be from a network attached to another interface, the packet is dropped.

This feature can be enabled on the Solaris Operating Environment. It is disabled by default. Use the following `ndd` command to enable it:

```
# ndd -set /dev/ip ip_strict_dst_multihoming 1
```

Add this command to the init scripts. Or alternatively, install the init script listed in ”Appendix A”.

Forwarding Directed Broadcasts

A directed broadcast is a unicast datagram from a system on a remote network addressed all systems on another network. Once the datagram reaches the router connected to the intended network, the datagram is forwarded to all systems as a data-link layer broadcast.

Directed broadcasts can be problematic due to the amount of network traffic generated by broadcasts and the ability to send a packet to all systems on a network. An attacker may take advantage of forwarded directed broadcasts to attack and probe systems. CERT Advisory CA-98.01 describes a denial of service attack called the “smurf” attack after its exploit program. It involves forged ICMP echo request packets sent to broadcast addresses. The source address in the forged packet is set to a target. The result is that the target and intermediate routing machine which forwards the directed broadcasts suffered from network congestion. One recommended action is to disable directed broadcast forwarding at all routers. Attackers may also send directed broadcasts to probe the network and determine which systems have exploitable vulnerabilities.

A Solaris system with IP forwarding enabled forwards directed broadcasts by default. You can disable it using the following `ndd` command:

```
# ndd -set /dev/ip ip_forward_directed_broadcasts 0
```

Add this command to the init scripts. Or alternatively, install the init script listed in ”Appendix A” of this article.

Routing

The process of routing involves examining a table of route information and making a decision about which interface to send datagrams. The routing table is the central point of information for each network host to determine where to send packets. Even a simple desktop system must determine whether the destination is on the local subnet (a direct route) or is reachable through a local router (an indirect route).

The routing table may need to be updated from time to time. There are several routing information protocols that are used to propagate routing information. The Solaris Operating Environment uses the in.routed and `in.rdisc` daemons to dynamically manage routing information. `in.routed` understands the Routing Information Protocol (RIP), version 1. `in.rdisc` implements ICMP Router Discovery. When a Solaris system is configured to forward packets as a router (IP forwarding enabled), these daemons advertise routing information to clients and

other routers and listen to other routers for information. As new information is received, these daemons update the routing table. This method of managing routing information is known as dynamic routing.

There are several problems with dynamic routing that attackers can use to create denial of service attacks or to view packet data from inaccessible systems. First, routing information can be forged. Routing information is typically sent via broadcast or multicast packets. An attacker can generate routing information packets claiming to be from a router and send them out to hosts or routers. These packets can direct hosts to send packets to a system that is not a router or to a busy router that cannot handle the increase in traffic. Most network administrators have a story about dynamic routing gone awry. Misconfigured routers generate their own denial of service problems. A more sophisticated attack involves directing packets through a multihomed system to examine the packet data as it flows across this system which now functions as a router. The attacker sends forged routing information packets to a router claiming a lower hop count to a destination network that the attacker cannot access. The target router then routes packets through the compromised system allowing the attacker to examine the traffic.

By default, a Solaris system uses system daemons to dynamically manage routing information. Static routing can be used to prevent malicious remote routing changes. The Solaris Operating Environment defines a default route by creating a file named `/etc/defaultrouter` containing the IP address of the router for local subnet. Other static route can be defined using the “route” command. See the man page on “route” for more information. Static routing works in environments with a single router to other networks. Networks with redundant routers need to use dynamic routing so that systems can switch to a working router should one fail. Solaris systems functioning as network routers should continue to use dynamic routing.

Forwarding Source Routed Packets

A source routed packet contains a specific path that datagram should follow to its final destination. Normally, routing decisions are handled by routers. They maintain information on available routes and dynamically update them as new route information is received. Source routed packets bypass routing decisions made by routers to define their own path to the final destination.

There is not much need for source routing in most networks. Properly configured routers make better decisions about routing. Source routed packets may be an indication of nefarious activity on the network. An attacker may attempt to use source routed packets to bypass specific routers or internal firewalls or try to avoid a known network intrusion detection system by routing packets around it. Source routed packets are rare. Silently dropping them should affect few if any applications.

A Solaris system with IP forwarding enabled forwards source routed packets by default. It can be disabled with this ndd command:

```
# ndd -set /dev/ip ip_forward_src_routed 0
```

Add this command to the start-up scripts. Or alternatively, install the init script listed in ”Appendix A” of this article.

ICMP

The Internet Control Message Protocol (ICMP) provides a mechanism to report errors and request information. The configuration parameters discussed here are managed in IP driver.

Broadcasts

ICMP broadcasts are always troublesome. There are rules that prevent “broadcast storms” by governing when ICMP error messages should not be generated. However, a significant number of replies to a ICMP broadcast from all systems on a network could cause significant network performance degradation. An attacker may use broadcast ICMP requests to create a denial of service attack. It is best to disable the ability to respond to ICMP broadcasts. Solaris software has three ICMP broadcast parameters, as described in the following sections.

Echo Request Broadcast

An echo request is a common network diagnostic created with the ping command. Echo requests can be sent to broadcast addresses. All systems configured to respond to broadcasted echo requests will send an echo reply. That can be a large number of packets. Even more devastating is the ability to increase the payload size of the packet. The receiving system will return all of the data contained in the payload. Extremely large payloads will be fragmented across several packets, thus further increasing network traffic. Disable responding to echo request broadcasts with this ndd command:

```
# ndd -set /dev/ip ip_respond_to_echo_broadcast 0
```

Add this command to the system start-up scripts. It is also included in the init script in ”Appendix A”.

Timestamp Request Broadcast

Timestamp requests are often used to synchronize clocks between two systems. Individual timestamp requests are normal, but there is no need for a system to respond to a broadcasted request. Use this ndd command to disable it:

```
# ndd -set /dev/ip ip_respond_to_timestamp_broadcast 0
```

Add this command to the system start-up scripts. It is also included in the init script in ”Appendix A”.

Address Mask Broadcast

An address mask request is used to determine the netmask for a network. It can be sent by diskless systems, such as printers or X-terminals, while booting. This type of request is typically broadcast. Use the following ndd command to disable responding to address mask broadcasts:

```
# ndd -set /dev/ip ip_respond_to_address_mask_broadcast 0
```

In Solaris software, this option is actually disabled by default. Most environments will never need to enable it. It is mentioned here for completeness and to bring attention to its existence. It is also included in the init script in ”Appendix A”.

Redirect Errors

Redirect Errors are used by a router to inform a host sending data to use a different router. Both routers involved in the redirection must be connected to the same subnet. The sending host will then install a new host routing entry in the routing table for the destination host. Unlike ARP entries, these will not time out and be deleted. Most systems check the redirect message for errors and potential problems prior to modifying the routing table.

Receiving Redirect Errors

An attacker may forge redirect errors to hosts to install bogus routes. This may create a denial of service attack since the different router may not be a router at all. There are some rules governing valid redirect errors, all of which can be spoofed easily. There are several attack tools do this. Use this ndd command to ignore ICMP redirect errors:

```
# ndd -set /dev/ip ip_ignore_redirect 1
```

Most environments with a single default router will have no need of accepting redirects. Add this command to the system init scripts. It is also included in the init script in ”Appendix A”.

Sending Redirect Errors

Only routers need to redirect errors, never hosts or even multihomed systems. Disable the sending of redirect errors with this ndd command:

```
# ndd -set /dev/ip ip_send_redirects 0
```

Add this command to the system start-up scripts. It is also included in the init script in ”Appendix A”.

Timestamp Requests

As mentioned previously, ICMP timestamp broadcasts are unnecessary in most environments. The Solaris software also has the ability to disable unicast timestamp requests as well. This ndd command does this:

```
# ndd -set /dev/ip ip_respond_to_timestamp 0
```

Add this script to the system start-up scripts. It is also included in the init script in ”Appendix A”.

This command prevents the system from responding to timestamp requests. Some remote UNIX® systems using the rdate command will no longer be able to retrieve the time. The Solaris rdate command uses the TCP time service provided by inetd and is not affected by remote systems that do not respond to ICMP timestamp

requests. Solaris 2.6 and 7 releases include a better method for time synchronization across multiple systems using the Network Time Protocol (NTP) system. Refer to the `xntpd` man page for more details.

TCP

The Transmission Control Protocol (TCP) provides connection-based, reliable data transport. It uses a lower protocol, IP, to manage delivery of datagrams. TCP handles connection management and has provisions for reliable delivery of data. The network configuration options described here are managed in the Solaris TCP driver.

SYN Flood Attacks

The 1996, Issue 48 of the electronic journal *Phrack* contained an article, titled *Project Neptune*, describing a network denial of service attack against TCP called “SYN flooding”. This attack makes a system respond very slowly to incoming network connections or not at all. A web site may appear to be down because it cannot establish connections for incoming browser requests. The *Phrack* article also contained source code to a program to generate SYN floods against remote systems. Soon after publication, several large ISPs and web sites were victims of these types of network attacks. Attackers launched attacks from their dial-up modem connections to the Internet that brought down sites with much faster connections to the network. Often it was difficult to trace the attack back to the source. Outbound and current connections were unaffected.

The Transmission Control Protocol (TCP) is part of the TCP/IP network protocol suite and is connection-oriented. Prior to exchanging data using TCP, two systems must create a connection. Connection establishment is a three step process in TCP, often called the “three-way handshake”. During this handshake, destination port information is exchanged and the two systems synchronize sequence numbers. (The “SYN” name refers to this synchronization step.)

The handshake works in the following manner:

1. A client sends a TCP segment to a server with the SYN flag set in the header and an initial sequence number (ISN) and port number.

2. The server returns a segment to the client with the SYN flag set, an acknowledgement (or ACK flag), the original ISN + 1, and its own ISN

3. The client sends a segment with the ACK flag set and the server’s ISN + 1.

A connection is now established and data can be exchanged.

The ISNs are used to provide reliability to the TCP protocol. The sequence numbers are incremented and sent with each outgoing packet. This allows the remote system to put packets in the proper order. If a packet is missing from the sequence, it can be detected and retransmitted.

The SYN flood attack takes advantage of a weakness in TCP. When a server receives the first SYN segment, it sends a SYN / ACK segment to the client address listed in the SYN segment. However, if that client is unreachable, the server will resend the SYN / ACK segment until a time limit is reached. (ICMP errors returned by the IP layer are ignored by the TCP layer.) If an attacking host sends many SYN segments for unreachable hosts, the server spends much time and system resources attempting to establish connections. Eventually, the server will reach its limit. Incoming connections still in the handshake phase are part of the backlog queue for the specified port. These queues are typically small. Once the queue is full, no further incoming SYN segments can be processed. Either the system will no longer respond for that port or the initial response becomes very sluggish. Systems with many network services could exhaust system memory because of the high number of uncompleted connections in the backlog queues.

In response to this attack, the Solaris 2.5.1 kernel TCP connection queue was changed and patches were issued. Previously, the size of the connection queue defined the size of the backlog queue. Now, there are two queues. There is still a queue for established connections. The new queue is for unestablished connections where the handshake process is incomplete. SYN flood attacks affect this queue. When an attack occurs and the unestablished connection queue fills, an algorithm drops the oldest SYN segments first and allows the legitimate connections to complete. Patch 103582-11 (and up) adds this new queue system to Solaris 2.5.1 release. Solaris 2.6 and 7 releases already have it. When a system is under attack, the following message will appear in the logs:

```
Mar 8 19:24:01 example unix: WARNING: High TCP connect timeout
rate! System (port 80) may be under a SYN flood attack!
```

This indicates that the system is handling the attack as designed.

The sizes of the new queues are adjustable. Busy web servers may need to increase the size of the unestablished connection queue. The default size of the queue is 1024. Use the following ndd command to increase it to 4096:

```
# ndd -set /dev/tcp tcp_conn_req_max_q0 4096
```

Add this command to the system init scripts or use the script listed in ”Appendix A” of this article. Any time a kernel queue is increased in size, there must be adequate system memory to handle the increase.

Connection Exhaustion Attacks

While SYN flood attacks work on the principle of attacking the TCP three-way handshake, connection exhaustion attacks work on established connections. These attacks are not common because the connections can be traced back to the source in most cases, unlike SYN flood attacks. Most operating systems have a limit on the number of established connections that can be maintained whether set by kernel parameter or available physical memory. Once this limit is reached, no new connections can be established. The active connections must be completed and closed before new connections can be established. For most web servers, this limit is never reached due the fact that HTTP connections are typically short-lived. An attacker can open many connections to a server and hold them open for long periods of time, effectively pushing the server closer to its connection limit. A web server will close connections that have completed and accept new connections. An attacker who continually and quickly requests new connections will eventually hold all of the available connections. Normal users of the web server will receive messages indicating that the web server is not responding. This another denial of service attack.

Some defense against this type attack can be provided by tuning kernel and application parameters. This is not a complete solution since it is basically a battle of resources. Whoever has the most resources (systems, memory, etc.) will most likely win. An attacker can spread the connection attacks out to multiple systems to increase the total connection requests. However, some application and kernel adjustments can be made to reduce the effectiveness of such attacks. Most web servers have a parameter that sets the connection timeout value. For example, Apache 1.3.9 has a value named “Timeout” that sets the maximum time connection can be established. Once this time limit is reached, the server closes the connection. Setting this value to a lower value shortens the timeout period, and long-time connections are closed more quickly. Additionally, Solaris versions 2.5.1 (with at least patch 103582-11), 2.6, and 7 have a common parameter to adjust the maximum number of established network connections. The default value is 128. Use the following ndd command to increase it to 1024:

```
# ndd -set /dev/tcp tcp_conn_req_max_q 1024
```

Add this command to the system init scripts or use the script listed in ”Appendix A” of this article. Be aware that increasing the number of established connections will increase the amount of memory needed to process all connections. Server performance may suffer. Additional system memory may be required.

Decreasing the connection time and increasing the maximum number of established connections should be sufficient to ride out most connection exhaustion attacks. It may still be possible to create an effective denial of service even with the changes. However, the attacker must devote significant resources to be effective.

IP Spoofing Attacks

Predictable ISNs make it possible for attackers to compromise systems. The TCP three-way handshake discussed previously involves two systems synchronizing sequence numbers prior to data exchange. For each new connection most systems use ISNs that are fixed, predictable increments of a counter. An attacker uses this knowledge to create a three-way handshake using a predicted ISN to establish a connection and execute a command.

This is a sophisticated attack that involves exploiting a trust relationship between two systems. Typically, a remote shell command (`rsh`) is attempted due to the trust configuration of a `.rhosts` file. An interesting thing to note is that this attack is carried out with the attacker unable to see the packets returned from the target host. This is due to the fact that the attacker is not on the same local network and the packets will be destined for the spoofed host. For this example, assume host A trusts host B. An attacker on host C (on a different network) wants to execute a command on host A. The first step in this attack is to disable host B. This can be done using the SYN flood attack described earlier. The attacker then establishes a TCP connection (or several connections to judge network delays) to the target host to sample the ISN used. This will be used to predict the next ISN.

The attacker uses the following steps in the TCP three-way handshake:

1. The attacker creates a TCP segment with the SYN flag set and an arbitrary ISN. The source address is set to trusted host, and it is sent to the target system.

2. The target system returns a segment to the trusted system with the SYN and ACK flags set, the attacker ISN + 1, and its own ISN. The attacker cannot see this packet.

3. The attacker waits a period of time to allow the SYN / ACK segment to be sent and then sends a segment with the ACK flag set and the predicted ISN + 1.

If the attacker predicts the target's ISN accurately, then the remote shell daemon (`in.rshd`) will believe it has a valid connection to the trusted host. The attacker can now execute a command on the remote system.

RFC 1498 defines a better method for generating initial sequence numbers to prevent IP spoofing attacks. Using the procedure defined in the RFC, each connection has a unique and seemingly random ISN. A system using this technique is now a very difficult target for an attacker attempting to predict the ISN.

There are several settings available on Solaris systems: the predictable method (0), an improved method with random increment value (1), and the RFC 1948 method (2). The default method for all revisions of Solaris is 1. Solaris 2.6 and 7 releases have

all of these methods. The Solaris 2.5.1 release only has methods 0 and 1. Solaris 2.6 and 7 releases should be modified to use method 2. To do this, edit the `/etc/default/inetinit` file and change the following line:

```
TCP_STRONG_ISS=1
```

to

```
TCP_STRONG_ISS=2
```

Reboot the system after this change. Unfortunately, Solaris 2.5.1 software does not offer the RFC 1948 method, and there are no plans to backport it.

Adding Privileged Ports

The Solaris Operating Environment and other UNIX Platform variants restrict access to ports less than 1024. Ports 0-1023 are called reserved or privileged ports because only a process with superuser privilege can acquire them. Solaris 2.5.1, 2.6, and 7 releases provide a method to extend the privileged port range beyond 1023. Additionally, Solaris 2.6 and 7 releases have a mechanism to add additional, individual privileged ports.

Some services operate with superuser privilege but outside the privileged port range. The NFS server process (`nfsd`) attaches to port 2049. Unfortunately, an attacker without superuser privilege may start a server process on a system that normally does not operate as an NFS server. This nonprivileged process may offer a false NFS service to unsuspecting clients. There are other services and applications that operate outside the standard privileged port range as well.

The privilege port range and even individual ports can be configured in Solaris 2.5.1, 2.6, and 7 releases to allow the privileged port range to extend using the `tcp_smallest_nonpriv_port` parameter in the TCP driver. It is used to specify the smallest nonprivileged port number. Use the following `ndd` command to extend the privileged port range to 2050 to protect the NFS server port:

```
# ndd -set /dev/tcp tcp_smallest_nonpriv_port 2050
```

Add this command to the system init scripts.

With Solaris 2.6 and 7 releases, you can specify additional privileged ports. The current list of privileged ports can be listed using the following `ndd` command:

```
# ndd /dev/tcp tcp_extra_priv_ports
2049
4045
```

This is the output from a standard Solaris 7 system that shows that the NFS server port (2049) and the NFS lock manager port (4045) are already protected as privileged ports. These are protected in the Solaris 2.6 release as well. Privileged ports can be added using the `tcp_extra_priv_ports_add` parameter in the TCP driver. To add one, use the following `ndd` command:

```
# ndd -set /dev/tcp tcp_extra_priv_ports_add 6112
```

Port 6112 will be added to the list of privileged ports. Add this script to the start-up scripts. Use the `tcp_extra_priv_ports_del` parameter to remove previously configured ports.

Extending the privileged port range can break applications. Plan prior to configuring additional privileged ports. Determine which server processes run with superuser privilege outside of the privileged port range. Bear in mind that some services may run as normal user processes. Extending the range or including a port inappropriately will prevent the server from acquiring the network port needed to operate. Whenever possible, add specific ports to the privileged port list instead of changing the range of privileged ports.

Summary

The application of some of these network security settings will require planning and testing but should be applicable to most computing environments. Being cognizant of the known network attacks will hopefully provide the needed leverage to apply changes. A free, publicly available tool called Titan can assist you in configuring these network changes and other security related processes. Many Sun customer sites use this tool to configure security on their Sun systems. Sun Professional Services has security architects to assist customers in creating a secure environment using the network security settings discussed in this article. You can download the tool from `http://www.fish.com/titan/`

Bibliography

Bellovin, Steven. *Defending Against Sequence Number Attacks*, RFC 1948, AT&T Research, Murray Hill, NJ, May 1996

CERT. *IP Spoofing Attacks and Hijacked Terminal Connections*, CERT Advisory CA-95.01
`http://www.cert.org/advisories/CA-95.01.IP.spoofing.attacks.and.hijacked.terminal.connections.html`

CERT. *"smurf" IP Denial-of-Service Attacks*, CERT Advisory CA-98.01
`http://www.cert.org/advisories/CA-98.01.smurf.html`

CERT. *TCP SYN Flooding and IP Spoofing Attacks*, CERT Advisory CA-96.21
`http://www.cert.org/advisories/CA-96.21.tcp_syn_flooding.html`

daemon9. *IP-spoofing Demystified*, Phrack 48, file 14
`http://www.phrack.com/search.phtml?view&article=p48-14`

daemon9. *Project Neptune*, Phrack 48, file 13
`http://www.phrack.com/search.phtml?view&article=p48-13`

Graff, Mark. Sun Microsystems Security Bulletin: #00136, 1996,
`http://sunsolve.Sun.COM/pub-cgi/retrieve.pl?doctype=coll&doc=secbull/136&type=0&nav=sec.sba`

Morris, R. T. *A Weakness in the 4.2BSD UNIX TCP/IP Software*, CSTR 117, 1985, AT&T Bell Laboratories.

Plummer, Dave. *An Ethernet Address Resolution Protocol*, RFC 826, Network Information Center, SRI International, Menlo Park, CA., November 1982.

Stevens, W. Richard. *TCP/IP Illustrated, Volume 1*, 1995. Addison-Wesley.

Appendix A

This shell script implements most all of the `ndd` commands mentioned in the article. Make any variable adjustments before using. Follow the instructions in the comments of the script to install it.

- nddconfig

This shell script can be downloaded at http://www.sun.com/blueprints/tools/

Author's Bio: Keith Watson

Keith Watson has spent the past two years at Sun developing an enterprise network security auditing tool suite named the Sun Enterprise™ Network Security Service (http://www.sun.com/software/communitysource/senss/). He currently works for the SunPS Global Enterprise Security Service (GESS) consulting practice. Prior to joining Sun, he was part of the Computer Operations, Audit, and Security Technologies (COAST) laboratory at Purdue University.

Author's Bio: Alexander Noordergraaf

Alexander Noordergraaf has over eight years of experience in the area of Computer and Network Security. As a Senior Security Architect for SunPS Global Enterprise Security Service (GESS), he has worked with many Fortune 500 companies on projects that include Security Assessments, Architecture Development, Architectural Reviews, and Policy/Procedure review and development. His customers have included major telecommunication firms, financial institutions, and ISPs.