Solaris™ Operating Environment Network Settings for Security: *Updated for Solaris 9 Operating Environment*

Alex Noordergraaf, Enterprise Server Products

Sun BluePrints™ OnLine—June 2003

http://www.sun.com/blueprints

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No. 816-5240-11
Revision 1.0 2/23/04
Edition: June 2003

Please
Recycle

Solaris™ Operating Environment Network Settings for Security: *Updated for Solaris 9 Operating Environment*

This article describes network settings available within the Solaris™ Operating Environment (Solaris OE) and recommends how to adjust network settings to strengthen the security posture of Solaris OE systems.

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

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 is addressed in the Sun BluePrints™ OnLine article, “Solaris Operating Environment Security: Updated for the Solaris 9 Operating Environment” published in July, 2002.

The information in this article is applicable to Solaris 2.5.1, 2.6, 7, 8 and 9 OE releases. Some evaluation is necessary prior to using the settings in this article with other Solaris OE releases.

The application of most of these network security settings 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 beneficial changes.

A free and publicly available security tool called the Solaris™ Security Toolkit (also known as JASS) can assist in configuring these network changes and other security related processes. Many Sun customer sites use this toolkit to configure security on their Sun systems. Additional information about this toolkit can be found at: `http://www.sun.com/security/jass/`.

The ndd Command

Several of the network settings discussed in this article are configured using the ndd command. It is used to examine and set kernel module parameters, namely the Transmission Control Protocol/Internet Protocol (TCP/IP) drivers. Most kernel parameters accessible through ndd can be modified 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 \?
# ndd /dev/udp \?
```

These commands list the parameters for the Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP), IP, TCP, and User Datagram Protocol (UDP) drivers. In this updated BluePrint OnLine article, the various drivers are listed in alphabetic order.

The Solaris 8 and 9 OE releases include support for the next version of the Internet Protocol suite (IPv6) and the Internet Protocol Security architecture (IPsec). These have additional drivers. A list of parameters for these drivers can be found with the following commands:

```
# ndd /dev/ip6 \?
# ndd /dev/icmp6 \?
# ndd /dev/tcp6 \?
# ndd /dev/udp6 \?
# ndd /dev/ipsecesp \?
# ndd /dev/ipsecah \?
```

The IPv6 parameters for the ICMP, IP, TCP, and UDP drivers are also listed in the standard (IPv4) parameter lists. This article does not discuss IPsec, but the parameters are listed here for completeness. Neither IPv6 nor IPsec support will be supported in any Solaris OE release earlier than Solaris 8 OE.

There are also network interface device drivers with parameters that can be adjusted using the ndd command. The following command will list the parameters for the hme (FastEthernet) device driver:

```
# ndd /dev/hme \?
```

The "\?" string 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.

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 /dev/arp arp_debug
0
```

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 this ndd command:

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

Notes on Parameter Changes

Previously, only some ndd parameter documentation was available from Sun. This has been a known problem. Since the release of the Solaris 8 OE, there is now documentation of selected tunable TCP/IP parameters. The book is the *Solaris Tunable Parameters Reference Manual* and is available on the docs.sun.com℠ web site. Most of the parameter information for the Solaris 9 OE is also applicable to previous releases.

Network parameters set with the ndd command apply to the currently running Solaris instance; parameter changes do not last past system reboots. Once a system is booted, the default parameters will be used. To provide a simple method of setting

the ndd network parameters mentioned in this article at Solaris boot time, a system init script has been created and is described in "Sample System nddconfig init Script."

Setting driver parameters involves making trade-offs. Most parameters involve changing the default Solaris OE configuration. The default settings are optimal for most situations. Adjusting parameters might affect normal system operation, so Sun does not encourage parameter changes.

All ndd parameter changes suggested in this article include a discussion of trade-offs, where appropriate. Some settings change the expected operation of systems; these are noted. Most of these recommended parameter changes are being actively used on production systems at customer sites.

Sun sometimes alters parameter names or adds additional parameters between releases of the Solaris OE. Most of the IPv4 parameters described in this article are used consistently across Solaris OE releases. When there are exceptions, the text for the parameter specifically mentions the OE differences.

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

ARP

The Address Resolution Protocol (ARP) is used to map 32-bit IPv4 addresses to the address scheme used by the data-link layer. The data-link layer, sometimes referred to as the network link layer, consists of the operating system device driver and corresponding network interface card. This layer is responsible for dealing with the physical transport media. Sun network devices use a system-wide hardware address, sometimes referred to as the Media Access Control (MAC) address. This means that a Sun system with multiple Ethernet interfaces will, by default, have the same hardware address for each interface.

A Sun Quad FastEthernet™ card has a unique hardware address assigned to each of its four interfaces. It is also possible to configure the card to use the hardware address from the cards programmable read only memory (PROM). Refer to the Sun Quad FastEthernet card documentation for more information.

It should be noted that many operating systems, including the Solaris OE, allow the hardware or MAC address, of a network interface, to be defined through software. By explicitly setting the hardware address of a network interface in software, the vendor-defined hardware address will be overridden.

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 the network administrator. It is this dynamic nature of ARP that causes security issues.

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

ARP operates by broadcasting an address request and collecting the response to create its map 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 send it 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 might respond with a reply, called an *ARP reply*, which contains the host IP address and hardware address. The received response is used to build a table of IP addresses and the corresponding hardware addresses.

In the Solaris OE kernel, there are two tables that maintain the addresses. One table, maintained by the ARP layer, is called the *ARP cache*. It provides a layer of efficiency to the protocol. For instance, 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, an ARP request is sent, and the corresponding reply is processed. The Solaris OE also adds unsolicited address information to the ARP cache. These unsolicited address entries are special because they were not directly requested. These unsolicited 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 contains information supplied by requests to the ARP layer. By default, an entry will expire 20 minutes after it was added to the table.

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 OE 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 gratuitous ARP information will automatically update the hardware address information.

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 delivery 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 can 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 can prevent one system from exchanging packets with another. This makes the system appear to be *off the network*. During a spoofing attack, one system masquerades as another.

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. On the local system, an attacker who has administrative control of the system can insert bogus address information into 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 might use the *publish* feature of the ARP layer to broadcast incorrect information about other systems. If two ARP replies are received, the last one will be used. It might be the correct one, or it may not. This situation can spread discord throughout systems on the local network and be difficult to diagnose.

ARP spoofing attacks are more serious because they are 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 and gain entry to other systems. This attack involves sending false hardware address information to a target system that the system will use to update its ARP tables. Once the false information is implanted, the attacking system changes its IP address and attempts a connection to the target.

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.

ARP 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 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.
- Entries will 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 are five minutes for all releases of the Solaris OE. IP routing table entries timeout after 20 minutes.

In Solaris OE versions 8 and newer, the following timeout intervals can be altered.

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

In Solaris OE versions 2.5.1 through 7, the `ip_ire_arp_interval` parameter is named `ip_ire_flush_interval`.

The timeout interval is specified in milliseconds. One minute equals 60000 milliseconds. Both these commands reduce the timeout period for the ARP cache and IP routing table. Entries will be deleted at a faster rate. This might slow down an ARP attack since bogus entries do not remain as long. These commands are available in the system `init` script provided in “Sample System `nddconfig init` Script” on page 26. The major side effect of this change is a greater number of ARP requests and replies will be sent. It might not be prudent to use on congested networks.

Another alternative is to manually create hardware address entries in the ARP cache. This solution can protect against some ARP attacks 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 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 the `arp -f` <*file*> command where *file* contains a table of hostnames and hardware addresses. These entries are now marked as permanent entries in the cache and will not be deleted by timeout. They can, however, be overridden by unsolicited information. In addition, they can still be deleted by using the `arp -d` <*host_entry*> command where *host_entry* is the host name to delete. This solution might not be appropriate in environments that frequently change equipment.

Note – Permanent ARP entries are only permanent in that they will not be timed-out. They can be overwritten by ARP information received over the network.

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 might 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.

ICMP

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

Broadcasts

ICMP broadcasts are, at times, troublesome. A significant number of replies to a ICMP broadcast from all systems on a network could cause significant network performance degradation. An attacker might use ICMP broadcast requests to initiate a denial of service attack. It is best to disable the ability to respond to ICMP broadcasts. Internal ICMP rules prevent *broadcast storms* by governing when error messages should not be generated. The Solaris OE has several 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. Use the following `ndd` command to disable response to echo request broadcasts:

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

Add this command to the system startup scripts. It is also included in the `init` script described in “Sample System `nddconfig` `init` Script” on page 26.”

Echo Request Multicast

This option is similar to the broadcast echo request in IPv4, and it is subject to the same attacks described in the previous section. This option determines whether to respond to ICMP echo requests (pings) sent to a multicast IP address. An attacker might try to create a denial of service (DOS) attack on subnets or multicast groups by sending many multicast echo requests to which all systems in the multicast group would respond. This also provides information on systems that are available on the network. The default value is 1 (true). A Solaris 9 OE system can be instructed to ignore echo requests sent to a multicast address with the following `ndd` command:

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

IPv6 does not have broadcast packets. It uses multicast packets instead. This is equivalent to the IPv4 multicast echo request, so all the same attacks apply. A Solaris 8 OE or Solaris 9 OE with IPv6 enabled interfaces can be instructed to ignore multicast echo requests with the following `ndd` command:

```
# ndd -set /dev/ip6 ip6_respond_to_echo_multicast 0
```

Add this command to the system start-up scripts. It is also included in the `init` script described in “Sample System `nddconfig` `init` Script” on page 26.”

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. Although these timestamp requests can be used to set a system’s clock, they are not related to the Network Time Protocol (NTP). To synchronize system clock’s throughout an enterprise, the recommended mechanism is NTP. Use the `ndd` command to disable responses to timestamp broadcasts.

```
# 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 described in “Sample System `nddconfig` `init` Script” on page 26.”

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. These requests are ignored by default and that configuration can be verified with the following `ndd` command:

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

This setting is also included in the `init` script described in “Sample System `nddconfig` `init` Script” on page 26.”

Redirect Errors

Redirect errors are used by a router to inform a host sending data to forward the packets to 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 might forge redirect errors to install bogus routes. This could initiate a denial of service attack if the newly specified router is not a router at all. There are rules governing valid redirect errors, all of which can be spoofed easily. Use this `ndd` command to ignore IPv4 ICMP redirect errors:

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

Similarly, for IPv6, the system can be instructed to ignore redirects with this command:

```
# ndd -set /dev/ip ip6_ignore_redirect 1
```

Most environments with a single default router for each subnet will not need to accept redirects. Add this command to the system start-up scripts. It is also included in the `init` script described in “Sample System `nddconfig` `init` Script” on page 26.”

Sending Redirect Errors

Only routers need to send redirect errors, not hosts or multihomed systems. Disable the sending of IPv4 redirect errors with this `ndd` command:

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

Similarly, for IPv6, it is also possible to disable the generation of redirect errors with this `ndd` command:

```
# ndd -set /dev/ip ip6_send_redirects 0
```

Add this command to the system start-up scripts. It is also included in the `init` script described in “Sample System `nddconfig` `init` Script” on page 26”.

Timestamp Requests

As mentioned previously, ICMP timestamp broadcasts are unnecessary in most environments. The Solaris OE software has the ability to disable unicast timestamp request responses. Disabling this setting prevents the system from responding to timestamp requests. Some UNIX® systems using the `rdate` command will no longer be able to retrieve the time remotely. The Solaris OE `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. The following `ndd` command disables a Solaris OE systems from responding to unicast timestamp requests:

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

Add this command to the system start-up scripts. It is also included in the `init` script described in “Sample System `nddconfig` `init` Script” on page 26”.

The Solaris 2.6, 7, 8, and 9 OE 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 additional details.

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 OE IP driver.

IP Forwarding

IP forwarding is the process of routing packets between network interfaces on one system. A packet arriving on one network interface and 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 OE system with more than one configured network interface forwards IP datagrams between network interfaces. It functions as a router. This is the default Solaris OE behavior.

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 processes network requests from multiple, directly attached networks. A large NFS server can serve clients on several networks. The server response is faster and the throughput is greater when the NFS server is directly attached to each network of clients it serves.

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

Packet forwarding can easily be disabled on a Solaris OE system. Simply creating a file named `/etc/notrouter` will disable IPv4 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 for IPv4:

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

Similarly, the following command will disable forwarding of IPv6 packets:

```
# ndd -set /dev/ip6 ip6_forwarding 0
```

An attacker might compromise a system to enable packet forwarding; thereby, gaining access to normally inaccessible systems. This is another reason to make sure all servers are secure.

Since the release of the Solaris 8 OE there is an additional capability to enable IPv4 forwarding on an interface-by-interface basis. This provides greater flexibility in determining which interfaces will forward packets for the purposes of creating IP-in-IP tunnels. The following `ndd` commands will enable IPv4 IP forwarding on the `hme1` and `hme2` interfaces while disabling it on `hme0`:

```
# ndd -set /dev/ip hme0:ip_forwarding 0
# ndd -set /dev/ip hme1:ip_forwarding 1
# ndd -set /dev/ip hme2:ip_forwarding 1
```

Strict Destination Multihoming

Strict destination multihoming prevents packet spoofing on non-routing multihomed systems. A Solaris OE system with IP forwarding disabled and strict destination multihoming enabled will ignore packets coming into an interface that has addresses belonging to a network that is connected only to a different interface. This prevents attackers from creating packets destined for networks connected only to a multihomed server that does not forward packets. The system is aware of which interface the packet arrives on and 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 OE. It is disabled by default. Use the following `ndd` command to enable it for IPv4:

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

Similarly, for IPv6, strict destination multihoming can also be enabled through the following command:

```
# ndd -set /dev/ip ip6_strict_dst_multihoming 1
```

Add this command to the system startup scripts. Or alternatively, install the `init` script described in “Sample System `nddconfig` `init` Script” on page 26.”

Forwarding Directed Broadcasts

A directed broadcast is a unicast datagram from a system on a remote network addressed to 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 might 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. In this forged packet, the source address will be defined as a victim system or router. The result is that the victim and intermediate routing systems that forwarded the forged packet will suffer from network congestion. One recommended action is to disable directed broadcast forwarding at all routers. Attackers might also send directed broadcasts to probe the network and determine which systems have exploitable vulnerabilities.

When IP forwarding is enabled on a Solaris OE system, directed broadcasts will be forwarded by default.

Disable it by using the following `ndd` command:

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

Add this command to the system startup scripts. Or alternatively, install the `init` script described in “Sample System `nddconfig` `init` Script” on page 26.

Routing

The process of routing involves examining a table of route information and making a decision about which interface to send datagrams based on the destination IP address. 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 is periodically updated. Several routing information protocols exist to propagate routing information between systems and routers. The Solaris OE includes the `in.routed` and `in.rdisc` daemons to dynamically manage routing information. The `in.routed` daemon implements Routing Information Protocol (RIP), while the `in.rdisc` daemon implements ICMP Router Discovery. When a Solaris OE system is configured to forward packets as a router (IP forwarding enabled), by default 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.

Note – With the release of Solaris 9 OE (12/02) and RIPv2 compliance, a new MD5 authentication mechanism for RIP is supported in Solaris. For more information on how to use this new capability, refer to the `rdisc` and `in.routed` man pages. This new capability is based on RFC 2082 and provides more robust security capabilities then RIPv1. However, even with this authentication mechanism, the RIP packets are vulnerable to some attacks. Refer to Internet-Draft “RIPv2 authentication flaws.”

There are several problems with dynamic routing that attackers can use to initiate denial of service attacks or 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. 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 that now functions as a router. The attacker sends forged routing information packets to a router claiming a lower *hop count* metric 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 OE system uses system daemons to dynamically manage routing information. Static routing can be used to prevent malicious remote routing changes. The Solaris OE defines a default route during startup based on the IP address of the router for the local subnet contained in `/etc/defaultrouter`. Define other static routes by using the `route` command. See the `route` man page for additional information. Static routing works in environments with a single router on each subnet. Networks with redundant routers may need to use dynamic routing so that systems can switch routers should one fail. A Solaris OE system functioning as a network router should continue to use dynamic routing.

Forwarding Source-Routed Packets

A source-routed packet specifies a routing path to follow. 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 define their own paths and bypass routing decisions made by routers.

There is little need for source routing in most networks. Properly configured routers make better routing decisions. Source-routed packets are frequently an indication of nefarious activity. An attacker might 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, legitimate applications.

When IP forwarding is enabled on a Solaris OE system, source-routed packets will be forwarded by default. It can be disabled for IPv4 with this `ndd` command:

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

Similarly, for IPv6, source-routed packets can be disabled through the use of this `ndd` command:

```
# ndd -set /dev/ip6 ip6_forward_src_routed 0
```

Add this command to the system startup scripts. Or alternatively, install the `init` script described in “Sample System `nddconfig` `init` Script” on page 26.

TCP

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

SYN Flood Attacks

In 1996, Issue 48 of the electronic journal *Phrack* contained an article, "Project Neptune," describing a network denial of service attack against TCP called SYN flooding. This attack makes a system respond very slowly (or not at all) to incoming network connections. A web site can appear to be down because it cannot establish connections for incoming browser requests. The *Phrack* article also contained source code to a program for initiating SYN flood attacks against remote systems. Soon after publication, several large Internet Service Providers (ISP) and web sites were victims of these network attacks. Attackers launched attacks from their dial-up modem connections to the Internet, which brought down sites with much faster connections to the network. Often it was difficult to trace the attack back to the source.

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, 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 starting with the agreed upon sequence number.

The sequence numbers 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 the following weakness in the TCP protocol handshake. 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 maximum of partially open connections. These incoming connections still in the handshake phase are part of the backlog queue for the specified port. In older versions of Solaris OE, the backlog queue was small. Once the queue is full, no further incoming SYN segments can be processed. Either the system will no longer respond for the specified 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 OE 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 the 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 later) adds this new queue system to the Solaris 2.5.1 OE release. The Solaris 2.6, 7, 8, and 9 OE releases have it incorporated. When a system is under attack, this 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 message indicates that the system is handling the attack as designed.

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

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

Add this command to the system startup scripts, or use the script described in “Sample System nddconfig init Script” on page 26. 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 attack 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 a kernel parameter or available physical memory. Once this limit is reached, no new connections are established. The active connections must be completed and closed before new connections are 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 is another denial of service attack.

One defense against this type of 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 the battle. 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, the Apache 1.3.9 web server has a configuration parameter named `Timeout` (in `/etc/apache/http.conf` of the Solaris 8 OE) that sets the maximum time a 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. Additionally, the Solaris 2.5.1 (with patch 103582-11 or later), 2.6, 7, 8, and 9 OE releases have a common parameter to adjust the maximum number of established network connections. The default value is 128. Use this `ndd` command to increase the default value to 1024:

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

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 successful.

IP Spoofing Attacks

Predictable ISNs make it possible for attackers to compromise some 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 have fixed and predictable counter increments. An attacker uses this knowledge to create a three-way handshake by predicting the required 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. This attack is carried out with the attacker unable to see the packets returned from the target host. It 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 the 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 1948 defines a better method for generating ISNs to prevent IP spoofing attacks. Using the procedure defined in this RFC, each connection has a unique and seemingly random ISN. A system using this technique is now a difficult target for an attacker attempting to predict the ISN.

There are several settings available on Solaris OE 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 the Solaris OE is 1. The 2.6, 7, 8, and 9 releases have all of these methods. The Solaris 2.5.1 OE release only has methods 0 and 1. Solaris 2.6, 7, 8, and 9 OE releases should be modified to use method 2.

There are two mechanisms to implement this change. The first option is to edit the `/etc/default/inetinit` file and change this line:

```
TCP_STRONG_ISS=1
```

to

```
TCP_STRONG_ISS=2
```

Reboot the system after this change.

The second mechanism is to enable this method while a system is in operation. Use the following command:

```
# ndd -set /dev/tcp tcp_strong_iss 2
```

Note – This method can only be set by using the `ndd -set` command.

Unfortunately, the Solaris 2.5.1 OE software does not offer the RFC 1948 method, and there are no plans to backport it. There might be a minor performance penalty for using the RFC 1948 method.

TCP Reverse Source Routing

As previously discussed, source-routed packets define a specific routing path instead of allowing network routers to make such decisions. Systems should be configured to not forward source-routed packets even when IP forwarding is enabled.

Additionally, the Solaris OE can be configured to ignore the reverse route on incoming TCP source-routed packets. Normally, the reverse routing path is copied into all packets destined for the system from which they were received. With TCP reverse source routing disabled, source-routed packets are processed normally, except that the reverse route information is removed from all response packets. It is available in Solaris 8 OE and newer OS releases.

This feature is disabled by default and that configuration can be verified with the `ndd` command:

```
# ndd /dev/tcp tcp_rev_src_routes
0
```

Ignoring the reverse route prevents an attacker from spoofing another system during the TCP handshake process. It is also included in the `init` script in “Sample System `nddconfig init` Script” on page 26.

Common TCP and UDP Parameters

There are parameters common to both the TCP and UDP drivers. These parameters implement concepts that are similar and independent of the protocol.

The Solaris OE and other UNIX variants restrict access to network socket port numbers less than 1024. Ports 1–1023 are considered reserved and require superuser privilege to acquire them. The range of these privilege ports can be increased. Specific ports can also be marked as privileged.

The Solaris OE also provides a mechanism to define the range of dynamically assigned ports. These ports are commonly referred to as *ephemeral* because they are typically short-lived and primarily exist for outbound network connections. The upper and lower bound of this port range can be adjusted.

Adding Privileged Ports

The Solaris 2.5.1, 2.6, 7, 8, and 9 OE releases provide a method to extend the privileged port range beyond 1023 for both the TCP and UDP drivers. Additionally, the Solaris 2.6, 7, 8, and 9 OE releases have a mechanism to add additional, individual privileged ports.

Some services operate with superuser privilege outside the privileged port range. The NFS server process (`nfsd`) attaches to port 2049. Unfortunately, an attacker without superuser privilege can start a server process on a system that normally does not operate as an NFS server. This nonprivileged process can 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 is extended using the `tcp_smallest_nonpriv_port` parameter in the TCP and UDP drivers. It is used to specify the smallest nonprivileged port number. Use the following `ndd` command to extend the privileged port range to 4096 for both the TCP and UDP drivers:

```
# ndd -set /dev/tcp tcp_smallest_nonpriv_port 4097
# ndd -set /dev/udp udp_smallest_nonpriv_port 4097
```

Add this command to the system `init` scripts to enable this behavior at system start.

It is also possible to specify additional privileged ports. The current list of privileged ports can be viewed using these `ndd` commands:

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

This output shows that the NFS server port (2049) and the NFS lock manager port (4045) are already protected as privileged ports. These two ports are the default additional privileged ports for the Solaris 2.6, 7, 8, and 9 OE releases.

Adding privileged TCP or UDP ports involves similar but separate parameter names. Add TCP privileged ports using the `tcp_extra_priv_ports_add` parameter for the TCP driver. Add UDP privileged ports using the `udp_extra_priv_ports_add` parameter for the UDP driver. For example, to add TCP and UDP port numbers to the privileged list use this `ndd` command:

```
# ndd -set /dev/tcp tcp_extra_priv_ports_add 7007
# ndd -set /dev/udp udp_extra_priv_ports_add 7009
```

TCP port 7007 and UDP port 7009 are now part of the list of additional privileged ports.

It is also possible to delete defined additional privileged ports. Use the `tcp_extra_priv_ports_del` or `udp_extra_priv_ports_del` parameters to remove previously configured ports for the appropriate driver.

Extending the privileged port range can break applications. Prior to configuring additional privileged ports, determine which server processes run with superuser privilege outside of the privileged port range. Remember, that some services can 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.

Changing the Ephemeral Port Range

The Solaris 2.5.1, 2.6, 7, 8, and 9 OE releases provide a method to change the ephemeral port range for both the TCP and UDP drivers. The upper and lower range can be altered.

The following ndd commands show the range values for the TCP and UDP drivers:

```
# ndd /dev/tcp tcp_smallest_anon_port
32768
# ndd /dev/tcp tcp_largest_anon_port
65535
# ndd /dev/udp udp_smallest_anon_port
32768
# ndd /dev/udp udp_largest_anon_port
65535
```

Alter the ephemeral port ranges by specifying the smallest and largest port number for both the TCP and the UDP drivers.

Adjusting these values can be useful, particularly in firewall environments. Define a smaller range to simplify firewall rules for specific applications. Take care when defining a small range, because the ability to establish outbound network connections might be limited.

Sample System `nddconfig init` Script

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

Download the `nddconfig` system init script at:

```
http://www.sun.com/blueprints/tools/
```

About the Author

Alex Noordergraaf has over 10 years experience in the areas of computer and network security. As the Security Architect of the Enterprise Server Products (ESP) group at Sun Microsystems, he is responsible for providing technical leadership to define the security of Sun's next generation servers while addressing security for current products. He is the driving force behind the very popular freeware Solaris Security Toolkit. Prior to his role in ESP, he was a Senior Staff Engineer in the Enterprise Engineering (EE) group of Sun Microsystems, where he developed, documented, and published security best practices through the Sun BluePrints program. Published topics include: Sun Fire™ Midframe 15K system security, secure N-tier environments, Solaris OE minimization, Solaris OE network settings, and Solaris OE security. He has co-authored two Sun BluePrint books *Jumpstart Technology - Effective Use in the Solaris Operating Environment* and *Enterprise Security Solaris Operating Environment, Security Journal.*

Prior to his role in EE, he was a Senior Security Architect with Sun Professional Services where he worked with many Fortune 500 companies on projects that included security assessments, architecture development, architectural reviews, and policy / procedure review and development. He developed and delivered an enterprise security assessment methodology and training curriculum to be used worldwide by SunPS. His customers included major telecommunication firms, financial institutions, ISPs, and ASPs. Before joining Sun, Alex was an independent contractor specializing in network security. His clients included BTG, Inc. and Thinking Machines Corporation.

Note – Alex Noordergraaf would like to acknowledge Keith Watson's contributions to previous versions of this article. In 2002, Keith Watson left Sun and returned to the Center for Education and Research in Information Assurance and Security (CERIAS) at Purdue University.

Related Resources

- 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-1995-01.html`.
- CERT. *"smurf" IP Denial-of-Service Attacks*, CERT Advisory CA-98.01 `http://www.cert.org/advisories/CA-1998-01.html`.
- CERT. *TCP SYN Flooding and IP Spoofing Attacks*, CERT Advisory CA-96.21 `http://www.cert.org/advisories/CA-1996-21.html`.
- daemon9. *IP-spoofing Demystified*, Phrack 48, file 14.
- daemon9. *Project Neptune*, Phrack 48, file 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`.
- Noordergraaf, Alex and Brunette, Glenn. "The Solaris Security Toolkit - Installation, Configuration, and Usage Guide: Updated for version 0.3," Sun BluePrints OnLine, June 2001, `http://sun.com/blueprints/0601/jass_config_install-v03.pdf`.
- Noordergraaf, Alex and Brunette, Glenn. "The Solaris Security Toolkit - Quick Start: Updated for version 0.3," Sun BluePrints OnLine, June 2001, `http://sun.com/blueprints/0601/jass_quick_start-v03.pdf`.
- Noordergraaf, Alex. "Minimizing the Solaris Operating Environment for Security: Updated for Solaris 9 Operating Environment," Sun BluePrints OnLine, November 2002, `http://sun.com/blueprints/1102/816-5241.html`.
- Noordergraaf, Alex and Watson, Keith. "Solaris Operating Environment Security: Updated for the Solaris 9 Operating Environment," Sun BluePrints OnLine, December 2002, `http://www.sun.com/solutions/blueprints/1202/816-5242.pdf`.
- 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.
- Sun Microsystems, *Solaris Tunable Parameters Reference Manual*, December 2001.

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`