Solaris Kernel Tuning for Security

by By Ido Dubrawsky (idubraws@cisco.com)
last updated Dec. 20, 2000

Introduction

The Solaris kernel provides a great deal of user-configurable control over the system TCP/IP stack. Everything from cache table lifetimes to the number of TCP connections that the system can address are controllable. However, without understanding the underlying need for tuning these kernel parameters many system administrators choose to ignore them - thereby leaving their systems vulnerable to a resourceful assailant.

Solaris Kernel Tools

The only tool available to Solaris system administrators for tuning kernel parameters is ndd. Currently, ndd only supports the TCP/IP kernel drivers. It can be used to both show and set the values of parameters for these drivers.

Solaris Kernel Parameters

In general to show a particular parameter the command format is:

```
# ndd /dev/<driver> <parameter>
```

where <driver> is one of the following: ARP, IP, TCP, and UDP. To view all parameters for a particular driver the command is:

```
# ndd /dev/<driver> \?
```

To set a kernel parameter using ndd, the format of the command is:

```
# ndd -set /dev/<driver> <parameter> <value>
```

Unfortunately, changes to the Solaris kernel parameter values using ndd are not permanent. The values for these parameters return to default upon system reboot. To make these changes more permanent a system administrator needs to put these changes into a shell script that is run at system boot (one possible location would be /etc/init.d/inetinit or in a separate shell script). One of the primary problems with setting these parameters into a shell script is that the parameters are implementation-specific and may change from one Solaris release to another.

ARP

ARP (Address Resolution Protocol) is used to dynamically map layer-3 network addresses to data-link addresses. When one system wants to communicate with another system on a network it first sends an ARP packet to the broadcast address, FF:FF:FF:FF:FF:FF. The packet asks the simple question: "who has network address A?...tell network address B". Since all hosts on a network receive these broadcast packets, system A receives the ARP request and sends back a response. The originating host then uses the responses to its ARP broadcasts to build a table, or cache, mapping the 32-bit IP addresses to Layer-2 hardware, or MAC, addresses. A second table is maintained by the network layer. This table is built from information provided by the data-link layer and contains network-routing information for active connections. The network layer requests MAC addresses from the data-link layer and inserts these addresses into a network routing table. Network routing entries expire after 20 minutes.

When a network host prepares to communicate with another the IP layer checks the ARP cache first. If an entry for the network peer does not exist in the cache, an ARP request is broadcasted. ARP cache entries expire after five minutes.

The ARP cache is susceptible to two types of attacks: ARP cache poisoning and ARP spoofing. ARP cache poisoning involves "inserting" either a non-existent ARP address or an incorrect ARP address into a system's ARP cache. This results in a denial of service since the target system will send packets to the peer's IP address but the MAC address will be incorrect.
ARP spoofing can result in system compromise. Just like IP spoofing, ARP spoofing relies on, first, disabling a host on the network so that it cannot reply to any ARP request broadcasts. Once that is done the attacker can configure the disabled host's IP address on the attacking host. When the victim host tries to communicate with the disabled host the attacker's system responds to any ARP request broadcasts, thereby inserting its MAC address in the victim's ARP cache. Communication between the two hosts can then proceed normally.
It is very difficult to defend against ARP attacks. One defence against ARP attacks is to reduce the lifetime of cache entries. The cache lifetime is determined by the kernel parameter arp_cleanup_interval. The IP routing table entry lifetime is controlled by the kernel parameter ip_ire_flush_interval

```
# ndd -set /dev/arp arp_cleanup_interval <time>
# ndd -set /dev/ip ip_ire_flush_interval <time>
```

where <time> is in milliseconds. Reducing the ARP cache timeout interval and the IP-routing table timeout interval will slow down an attacker but not stop them.
Another option could be to create static ARP addresses for some systems. Static ARP cache entries are permanent and therefore do not expire. These entries can be deleted using the command arp -d. A third option would be to disable ARP processing on the system interface(s) altogether and add static ARP entries.

IP Parameters

The Solaris kernel also provides for control of various characteristics of the IP network protocol. This functionality is provided through several parameters:

```
ip_forwarding
ip_strict_dst_multihoming
ip_forward_directed_broadcasts
ip_forward_src_routed
```

IP forwarding involves routing IP packets between two interfaces on the same system. Typically this is a job that a router performs, however, a Solaris system can perform this task as well. By default a system installed with Solaris will perform IP forwarding. This can be disabled by setting the kernel parameter ip_forwarding to 0:

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

Another avenue of attack would be for an intruder to create packets that are destined onnly for networks that are connected to a multihomed server that does not forward IP packets. By setting the parameter ip_strict_dst_multihoming to 0 the system drops any packets that appear to originate from a network attached to another interface:

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

Directed broadcasts are packets that are sent from one system on a foreign network to all systems on another network. Directed broadcasts are the basis for the "smurf" attack where forged ICMP packets are sent from a host to the broadcast address of a remote network. The source address in the ICMP packets is forged to contain the address of the victim host. The systems on the remote network receive the ICMP packet and then reply back to the victim host thereby flooding the host with traffic. Any Solaris system that has IP forwarding enabled will forward directed broadcasts as well. To disable the forwarding of directed broadcasts set ip_forward_directed_broadcasts to 0:

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

When packets travel between one host and another on a network their path is determined by either dedicated routers or hosts providing routing services. However, IP has the ability to specify the route between the source and destination. This ability comes in one of two forms: strict and loose. With a strict source route, the sender specifies the address of every intermediate hop between it and the destination. With

loose source routing the sender only specifies some of the intermediate hops leaving routers free to choose any path between the two systems. Source routing may be used to bypass security measures in the network topology. There is no reason to see source-routed packets in a network. Any host that does allow IP-forwarding should silently drop source-routed packets by setting the Solaris kernel parameter ip_forward_src_routed to 0:

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

TCP Parameters

In September 1996, Phrack Magazine published an article titled "Project Neptune". This article described a type denial of service attack known as a SYN flood. The goal of this attack is not necessarily to break into a system but to render the system unusable from the perspective of the internet or intranet. The attack exploits the basic way a TCP connection works. When a system attempts to connect to a server using TCP the sender a TCP/IP packet to the destination with the SYN bit set. This SYN packet is then acknowledged by the destination with a packet with both the SYN and ACK bits set in the TCP header. The sender then replies to SYN-ACK packet from the destination by sending its an ACK packet back. This "3-way" handshake looks something like:

```
Sender(client)                          Destination(server)

   SYN ------------------------------->
(Sequence #: X)
      <-------------------------------  SYN-ACK
                                      (Sequence #: X+1, Sequence
#: Y)
   ACK ------------------------------->
(Sequence #: Y+1)
```

The abuse can occur when the destination host has responded to the sender with a SYN-ACK but does not receive an ACK back from the sending host. This then leaves the destination host connection in a "half-open" state. The source host then opens a new TCP connection with the destination host and repeats the process. This process continues until all possible TCP socket connections that the destination host can handle are in the "half-open" state. Once this happens, no further TCP SYN packets can be processed by the target until the "half-open" connections are removed from the TCP connection queue.

One way to determine if a Solaris system is under a TCP SYN attack would be to monitor the number of TCP connections in a SYN_RCVD state:

```
# netstat -an -f inet | grep SYN_RCVD | wc -l
```

This value can be compared to a baseline value taken when the machine is running under normal circumstances. Solaris provides another way to determine if a machine is under a TCP SYN attack. By running the command:

```
# netstat -s -P tcp
```

and inspecting the values of the parameters tcpTimRetransDrop and tcpListenDrop a TCP SYN attack can be identified. The parameter tcpTimRetransDrop shows the number of aborts since boot time due to abort time expirations. This value includes both the SYN requests as well as established TCP connections.

The parameter tcpListenDrop shows the number of SYN requests that have been refused since the system was booted because of a TCP queue backlog. There is a high probability that the system is under a TCP SYN attack if the tcpListenDrop value increases quickly along with the value of tcpTimRetransDrop.

To offset such an attack the administrator must do two things:

a) shorten the value of the abort timer, and
b) lengthen the TCP connection queue.

To shorten the abort timer the kernel parameter: tcp_ip_abort_cinterval can be used.

The value for this parameter is given in milliseconds. By default the abort timer interval is 180 seconds. To set the abort time to 60 seconds the system administrator can use the command:

```
# ndd -set /dev/tcp tcp_ip_abort_cinterval 60000
```

The kernel parameter tcp_conn_req_max_q0 controls the queue size for unestablished TCP connections in Solaris 2.6 and above (or in Solaris 2.5.1 w ith patch 103581-11). The default value for tcp_conn_req_max_q0 is 1024. To increase the queue size the following command can be used:

```
# ndd -set /dev/tcp tcp_conn_req_max_q0 2048
```

Another type of SYN attack involves exhausting the TCP established connection queue. This attack is not as desirable as the TCP SYN attack mentioned above because of the fact that the connection can be traced back to its source, however, it still presents a problem. Solaris 2.6 and above (as well as Solaris 2.5.1 with patch 103582-11) provide control over the size of the established TCP connection queue. This control is provided by the kernel parameter tcp_conn_req_max_q. By default it is set at 128. To increase the established TCP connection queue, the command is:

```
# ndd -set /dev/tcp tcp_conn_req_max_q <size>
```

where <size> is the total number of active, established, TCP connections allowed to the host. Increasing either the TCP queue for unestablished connections or the TCP queue for established connections will require more memory. Without sufficient memory the server's performance will be affected. Also, while this provides some measure of relief against TCP SYN attacks and TCP established connection exhaustion attacks these types of attacks depend on which side has more resources. If the attacker can produce more TCP connections (whether "half-open" or established) than the server can possibly handle this denial of service will succeed.

Conclusion

The Solaris kernel has many configurable parameters that are security related. These parameters can be adjusted to strengthen the security posture of a system. The parameters cover such things as ARP timeouts, IP forwarding of packets, IP source routing of packets, TCP connection queue sizes, and many other factors governing network connections. By tuning the kernel properly a system administrator can even prevent OS fingerprinting of a Solaris system as provided by such tools as queso and nmap.

Relevant Links

Solaris[tm] Operating Environment Network Settings for Security
Alex Noordergraf and Keith Watson

Solaris inetd.conf Pt. 1
Hal Flynn

Solaris inetd.conf Pt. 2
Hal Flynn