Networking and Name Resolution: Section 1 Transcript

Introduction

1/3

Welcome to Networking and Name Resolution. We will discuss networking and name resolution in this module while focusing on system configuration and how to find this important information on a UNIX system.

For the purpose of this module, we will primarily discuss various networking commands and what the outputs of the commands are displaying. Configuration files containing network settings will also be explained along with their locations. Domain name resolution is described along with the order in which the data sources are interrogated. These settings must be mastered in order to form a solid foundational framework of how UNIX systems have their networking configured.

Additionally, some commonly used networking services and applications will be introduced along with commands that have been marked for deprecation by the Linux community, as many system administrators still use them daily. We will also mention the commands replacing the deprecated Linux commands. We also discuss networking configurations which are out of the ordinary as items of potential interest from a forensics perspective. All systems that we access remotely are on a network and it is very important to know how they are setup and what networks they can communicate over.

Throughout this module, you'll be presented with opportunities to assess and apply what you've learned.

At the end of this module, you will be able to:

- Describe the local name resolution process on a UNIX host,
- Analyze the output of network-related utilities,
- List and identify ports on the provided system,
- Describe the purpose of name resolution,
- Describe Network File Systems,
- Analyze entries within a socket table, and
- Create a bash backdoor using xinetd on the provided system.

Pre-requisites

2/3

Some of the commands that you will need to research and become proficient with are: `dig`, `finger`, `getent`, `host`, `hostname`, `ifconfig`, `ip`, `netstat`, `nslookup`, `ping/traceroute`, `route`, `ss`, and `telnet`.

You should use Internet search engines and UNIX man pages to conduct your initial research of these and other commands. You will find additional information about these commands in the Resources for this module.

Bypass Exam Introduction

If you are already familiar with the subject matter presented in this module, you can choose to take a Bypass Exam to skip this module.

The Bypass Exam option provides a single opportunity to successfully demonstrate your competence with the material presented within the module. If you pass, you'll receive credit for completing the module, unlocking the content within, and you will be free to proceed to the next module. If you do not pass, you will need to successfully complete the module, including all exercises and the Module Exam, to receive credit.

Click the Next Section button to continue.

Networking and Name Resolution: Section 2 Transcript

Network Configuration 1/5

Networking is the last piece of the puzzle when characterizing activity on a machine, and network configuration is the set of items needed to successfully exchange information on a network; UNIX differs from Windows in how it configures its network. One of the most important first steps when configuring a UNIX machine for TCP/IP networking is to understand which pieces of information are global for the machine, and which pieces of information are specific to a single network interface, as defined by a device name, typically eth0, p2p1, or e1000; however, naming conventions vary across UNIX variants. For example the loopback interface (i.e., the special interface that the computer uses to communicate with itself) is usually named "lo" or "lo0."

The four most important configuration items are IP Address, Netmask, Hostname, and Default Gateway. Select the images displayed to learn about each configuration item.

- An **IP Address** is required for networking, and is configured per interface.
- A **netmask** is required for networking, and is configured per interface.
- **Hostname** is not required, but limits network capability if not used, and is usually configured globally, but can be configured on a per interface basis on some systems.
- **Default Gateway** is not required, but limits network capability if not used, and is usually configured globally, but can be configured locally if there is only one interface.

Network Configuration Files 2/5

As with many UNIX configurations, there is a difference between the startup configuration and the running configuration. Let's focus on startup configurations.

Different UNIX systems have network configurations in different locations. In older UNIX variants, the startup networking configuration consisted of the administrator embedding commands in a startup script such as `/etc/rc.local`; however, most modern UNIX systems have a dedicated service (usually called network) that is responsible for starting up network services based on a set of configuration files stored somewhere in the `/etc` directory. Please note that there is no magic in the network service; it ultimately runs the same commands that an administrator would have configured to run in `rc.local` in the old days.

In CentOS, startup network configuration files are found in `/etc/sysconfig` and the subdirectory `/etc/sysconfig/network-scripts`, while in a Solaris 10 system startup, network configuration files are stored either directly in `/etc` or in the subdirectory `/etc/inet`. While not displayed, FreeBSD stores its network startup configuration in `/etc/rc.conf`, while Debian Linux uses the `/etc/network directory`.

Other Network-Related Utilities 3/5

UNIX has a variety of utilities that can be used to view information about the networks that the computer resides in; many of these are similar to Windows.

There are many other tools other than the ones displayed, but for now, click each button displayed to learn how to gather information to help assess system integrity.

ifconfig: The ifconfig command can be used to view information about the running configuration of network interfaces and assign an address to a network interface. The ifconfig command is used to redefine an interface's address or other operating parameters.

There is a lot of information in this display, but notice that this view lets us tie together each network interface with its MAC address and its IP address.

From a system integrity perspective, the main item to look out for is subinterfacing. A subinterface would typically be labelled with a ":x" following the interface name. For instance, eth0:1 has its own entry when running ifconfig -a. A subinterface allows the system to have more than one IP address on a single network. If both IP addresses are in the same network, there is likely no concern; however, it is fairly unusual to have multiple networks on the same network interface.

In Linux, ip addr is an alternative to ifconfig.

arp: The arp command shows the IP and MAC addresses of all machines that the system has recently seen on its LAN. This can be useful for identifying machines with which your target has been communicating. It is possible, but unusual, to include static entries in the ARP table. These will be shown in the arp -an output, depending on the operating system. CentOS prints the string "PERM" on a static entry line, while Solaris 10 uses the flag "S." Static entries in the ARP table could be indicative of a malicious user trying to redirect traffic to another host in the network.

In Linux, ip neighbor is an alternative to arp.

ping/traceroute: A traceroute determines where routing problems appear within a network path, and ping tests the reachability of a host by measuring the time it takes for a message to be sent to and returned from a host. The primary use for ping and traceroute is to validate the network connection.

Netstat: The netstat command-line tool displays network connections, routing tables, network interfaces, and network protocol statistics. It is used for finding problems in the network and to determine the amount of traffic on the network as a performance measurement.

In Linux, ss is an alternative to netstat.

route: The route command displays the system's current routing table; the command `netstat -rn` does the same thing. Custom routes added into the routing table can be interesting, especially if they are not expected to be there; like with ARP entries, this may be indicative of a malicious user trying to redirect traffic. Usually the only entries in the table are the following:

- An entry for each LAN to which the system belongs,
- An entry for the link-local (169.254.x.x) network, and
- The default gateway, usually shown with the "G" flag.

In Linux, `ip route show` is an alternative to route.

UNIX Networking

Welcome to the UNIX Networking video. I will be your guide for this video presentation.

As the systems on which we are operating are being accessed over a network, it is important to analyze the network configuration and state on every system. This provides context about where we are operating and help develop target situational awareness.

Since the on-disk configuration and state may differ, we will examine both. This will give us a more complete picture. It will allow us to see what changes have been made since the system booted and what changes will be made on the next reboot.

In this presentation, we will review the tools used to view and modify network configuration and state on some UNIX systems.

As you run these commands on a live system, you should always be asking yourself, "What does this information tell me about the network in which this system lives?" and "What does it tell me about this system's role in its network?"

Let's get started!

Various UNIX and Linux operating systems store their network settings in different locations and formats. We will only look at how traditional Solaris (versions 10 and older) systems and the RedHat family of Linux distributions store their network configuration. Fortunately, this topic tends to be very well documented, so you can easily find resources that cover other systems, such as Solaris 11, the Debian family of Linux distributions, FreeBSD, OpenBSD, Linux with NetworkManager, and any other unfamiliar system you may encounter.

Let's start by looking at how Solaris 10 and older stores its persistent network configuration.

When Solaris boots, it determines names for the network interfaces, based on the driver names. Here we see the network interface e1000g0 and the loopback interface lo0.

It then looks for a hostname for each interface in hostname files in /etc. The network interface name is the last part of the file name.

It looks up these hostnames in the hosts table to determine the associated IP addresses.

On older Solaris systems, this lookup includes the ipnodes file. As of the last build of Solaris 10, ipnodes is no longer used

The system's canonical hostname is stored in the nodename file. This resolves the ambiguity when there are multiple interfaces which would have different names in their hostname files.

The netmask for each network to which the system is connected is listed in the netmasks file. The first field is the network name and the second is the netmask.

Most Solaris systems store the default gateway in the defaultrouter file.

Solaris systems with multiple network interfaces or more complicated routing tables may store their routing information in /etc/gateways instead. The /etc/notrouter file would then be present unless the system is a router.

The DNS servers used for DNS queries and the default domain for host queries without an FQDN are stored in the /etc/resolv.conf file.

Changing the persistent configuration is just a matter of editing the relevant files, which is mostly self-explanatory. These changes would take effect the next time the system boots.

To add a static route, you can add that to the /etc/gateways file, if it's in use.

Otherwise, you can add a static route to the /etc/inet/static_routes file.

Now let's view the current state of the network on this Solaris system.

The ifconfig command shows us that interface e1000g0 is up with address 192.168.11.14.

Its netmask is shown in hexadecimal. This is the same as 255.255.255.0.

When run as root, ifconfig also shows the MAC address.

The hostname command shows the system's current host name.

The netstat command can be used to show the routing table. Here we see the default gateway and other routes.

For DNS settings, the persistent configuration on-disk and active configuration are the same thing.

With the arp command, we can see the arp cache. There are entries for our gateway and other systems our machine has communicated with recently. Solaris also has an entry for the local system.

Using netstat to view the TCP socket table, we see listening sockets and established connections.

Netstat can also show the UDP socket table.

Now let's look at how RedHat and related Linux distributions store their network configuration.

When Linux boots, it detects the network cards and names its network interfaces. There are a few naming conventions it can use. This system is using the traditional naming scheme, with eth0 as the first ethernet interface and lo as the loopback interface.

The IP configuration is stored in an ifcfg file named for the interface. This file sets various parameters by defining shell variables which are used by scripts in this directory. Most variables are optional and may or may not appear in this file.

This is an ethernet interface and its name is also stored in the file.

The interface is set to be enabled when the system boots.

The MAC address is listed here. This is optional, but must be correct or the interface will not be enabled.

For BOOTPROTO, "static" or "none" means it is statically configured in this file. Otherwise, it may use "dhcp."

Since it is static in this case, the IP address is explicitly defined here.

The netmask is also here, in this case in decimal notation. It could instead give the netmask in CIDR notation by saying PREFIX=24.

Traditionally, the host name and default gateway were listed in the /etc/sysconfig/network file.

The default gateway may also be listed in the ifcfg file that we just looked at.

On systemd-based Linux systems, including RHEL 7 and newer versions of other Linux distributions, as well as all Debian-based systems, the hostname is stored alone in the /etc/hostname file.

The host name and IP address may be in the hosts table too, but this is not generally required.

Static routes are stored in a route file for the associated network interface. A default route could be stored there, but that would be unusual.

There are two different formats that could possibly be in use in this file. Only one of these two formats would be in use at any particular time.

Viewing the current state of the network on Linux is almost exactly the same as for Solaris, with just some minor differences in the command-line arguments.

The ifconfig command shows us that interface eth0 is up with address 192.168.11.13.

We can also see the netmask and MAC address.

The hostname command shows the system's current host name.

The netstat command can be used to show the routing table. Here we see the default gateway and other routes.

For DNS settings, the persistent configuration on-disk and active configuration are the same thing

With the arp command, we can see the arp cache.

With Linux's netstat command, we can view the TCP and UDP sockets with the "-t" and "-u" options, and we can also view the PIDs of the associated processes with the "-p" option.

Now let's imagine that we have to move our Solaris and Linux ops VMs to a different network which uses a different IP scheme, MAC address scheme, domain name, and host name convention.

Let's start with the Solaris system. First, we'll use commands to make the needed changes to the active state. Then, we will update the configuration files to make the changes persistent.

The ifconfig command is used to change the IP address or netmask.

We can also bring the interface down.

If we need to spoof the MAC address to fit into some scheme in the new network, we can also do that with ifconfig.

Let's bring the interface back up and see the new MAC address in place.

On Solaris, ifconfig can also be used to enable DHCP on a network interface. This will start dhcpagent for that interface.

We can also turn DHCP off.

The host name is changed with the hostname command.

The route command is used to change default gateway. We also have to delete the old default.

The route command can also be used to add or delete a static route.

The arp command can be used to add or delete static ARP entries.

We need to update the resolv.conf file to change the DNS settings.

A successful DNS query shows that we have correctly configured our network settings

Next, we must edit the network configuration files so that the new settings will be in effect upon reboot.

We'll change the IP address in /etc/hosts and the hostname in /etc/hosts, as well as in the nodename and hostname files.

We can also use the hostname file to pass arguments to the ifconfig command to set the MAC address.

If we needed to set e1000g0 to use DHCP, we would create a dhcp.e1000g0 file for it.

We'll add the netmask for this network to the netmasks file.

The default gateway should be updated in the defaultrouter or gateways file, as appropriate.

We'll also update any static routes in the static_routes or gateways file, if necessary.

Now let's make the same changes on the Linux system. The commands are very similar to Solaris.

The ifconfig command is used to change the IP address or netmask.

We could also use ifdown to bring the interface down, but that would clear out all settings on the interface.

The ifconfig command can be used to spoof the MAC address.

We can see the new MAC address is in place. If we had used ifup to bring the interface back up, it would have read the settings from the on-disk configuration files which is how one would normally enable DHCP, if needed. The DHCP client could also be called directly. Most Linux systems use either dhclient or dhcpcd.

We can release the IP or bring down the interface to turn off DHCP.

The host name is changed with the hostname command.

We'll use the route command to change the default gateway, add a static route, or delete a static route, if needed.

The arp command can be used to add or delete static ARP entries.

If the ifconfig, route, arp, and netstat commands aren't available on a particular Linux system, the ip and ss commands can be used instead.

We need to update the resolv.conf file to change the DNS settings.

We can ping our Solaris system, so our network connectivity looks good.

Finally, we must edit the network configuration files so that the new settings will be in effect upon reboot.

We'll start by updating the IP address in the ifcfg file.

To change the MAC address, we cannot change the HWADDR variable, as this must match the actual hardware. We can use the MACADDR variable to do that.

If we were changing the interface to use DHCP, we would change the BOOTPROTO value to dhcp and remove the IPADDR, NETMASK or PREFIX, and GATEWAY variables.

We'll update the default gateway and host name if they're in the network file.

We'll also update the host name if it appears in the /etc/hostname or /etc/hosts files.

If needed, we'll update static routes in the route file.

Note that some Linux systems may have multiple network profiles configured and store a copy of some configuration files in another location. Those should be updated as well.

This concludes our video demonstrating how to view and manipulate UNIX network configuration and state. This should be helpful to you on the upcoming Networking Milestone exam.

Exercise Introduction

5/5

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Networking and Name Resolution: Section 3 Transcript

Name Resolution 1/6

Name resolution is the process of associating a name to a numeric value, both locally and via the Domain Name System (DNS). UNIX takes a very broad view of the concept of name resolution. There are a large number of parameters in both UNIX operations and networking that involve both a numerical ID and a human-readable name. Networking presents a number of additional such mappings including TCP and UDP port numbers and network names.

UNIX systems typically use a file called `/etc/nsswitch.conf` to define how the system should perform this name resolution.

Order of Lookup 2/6

The nsswitch configuration file determines the way in which a process looks up databases containing information regarding hosts, users, and more. The order of look up is decided by the line sequence for the databases. Each entry in `/etc/nsswitch.conf` contains a database name and a list of sources. Each source has one or more service specifications, and optional actions to perform if a certain result occurs.

For example, `/etc/host.conf` specifies how host names on a network are resolved. All possible options, such as configuration keywords, are on separate lines in the file. The keywords you will notice are: order, trim, multi, nospoof, spoof and reorder.

This being said, the `getent` command gets entries from several important databases. It uses `/etc/nsswitch.conf` to determine the search order of databases, including the password and group databases which store user information. The `getent` command is also used to search for information such as hosts, users, groups, and email aliases. In this example, `getent` is fetching password details for a user named leo.

DNS Lookup Tools 3/6

An important line of interest to us in the file `/etc/nsswitch.conf` is the "hosts" line. By default this file controls activities on the system such as converting port numbers to service names. The hosts line specifies the order in which various mechanisms are consulted when the system needs to map a hostname to an IP address, or vice versa. The default text for this line is `hosts: files dns`, which indicates that the system first consults its local files (namely `/etc/hosts`) to try to resolve the hostname. If unsuccessful, it then tries to do resolution through the DNS system, consulting the servers listed in `/etc/resolv.conf`.

There are a variety of useful tools available in UNIX to assist in troubleshooting name resolution. The standard DNS diagnostic tools that are deployed on most systems are `dig` and `nslookup`;

both allow the user to interact with a DNS server. The common use for these tools is to look up the IP address associated with a hostname, and to invoke the command with the target hostname as a single argument performs a DNS lookup for that hostname (for example, `dig example.com`). The host command performs similarly.

Note that all these tools are specifically DNS troubleshooting tools; therefore, they do not use the /etc/nsswitch.conf file to determine how they should resolve names, instead they start directly with a DNS lookup. These tools respect the nameserver ordering in the `/etc/resolv.conf` file, unless specifically configured to use a different nameserver than the default.

In order to test the full UNIX name resolution process, one can use the `getent` command, specifically using the syntax: `getent ahosts example.com`

This command references `/etc/nsswitch.conf` when making a decision about name resolution, and can provide a better picture of what the machine is actually doing when it resolves a destination. Let's take a few moments to explore each tool in more detail.

DNS Lookup Tools

4/6

Click the tabs to learn more about each tool. When you have finished reviewing each concept, click the Next Slide button to continue with the module.

- **host**- The host command is a simple utility for performing DNS lookups. It is normally used to convert names to IP addresses and vice versa. When no arguments or options are given, host displays a short summary of its command line arguments and options.
- **dig**- The domain information groper, commonly referred to as dig, is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use, and clarity of output. The host dns lookup tool has less functionality than dig.
- **nslookup**- The nslookup tool is a program used to query Internet domain name servers. The nslookup tool has two modes: interactive and non-interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and requested information for a host or domain.

Configuration Files

5/6

The hosts file assists in addressing network nodes in a network. It is a part of the IP implementation, and translates human-readable hostnames into IP addresses.

In some systems, the contents of the hosts file is preferred to other name resolution methods, such as the DNS, but many systems implement name service switches, such as `/etc/nsswitch.conf`, to provide customization. Unlike remote DNS resolvers, `/etc/resolv.conf`, the hosts file is under the direct control of the local computer's administrator.

Furthermore, `/etc/resolv.conf` is a set of routines that allow access to the DNS. It is human readable and is a trusted source of DNS information.

Exercise Introduction

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Networking and Name Resolution: Section 4 Transcript

Network File Systems 1/4

Remote file systems allow a server to share directories and files with clients over a network. With the Network File System (NFS), users and programs can access files on remote systems as if they were stored locally.

The original remote file system for UNIX was the NFS. It is still in wide use today and is implemented using Remote Procedure Call (RPC) services such as:

- `nfs` - The core of NFS, this service performs the file transfers and read/write actions necessary to share files,
- `mountd` - Used to allow a remote computer to mount part of the filesystem on its own system, and thus access it as if it were local,
- `nlockmgr` - Handles file locks to ensure files stay in sync, even over the network.

In general, NFS is very fast but until recently it was considered to be relatively insecure for use on public networks. Since NFS was written for UNIX, it tends to be a very good choice for connecting multiple UNIX systems. While other operating systems like Windows do implement NFS, making everything work together nicely can be tricky.

Since you now know the basics of NFS, let's look at another network file system, Samba.

Network File Systems 2/4

Microsoft also has a network file system, which is powered by the Server Message Block Protocol (SMB). UNIX machines can participate in SMB-based network file systems using a package called Samba, which implements the SMB protocol for UNIX. Samba is often considered slower but more secure than NFS.

Samba is a suite of UNIX applications that speak the SMB protocol. Many operating systems, including Windows and OS/2, use SMB to perform client-server networking. By supporting this protocol, Samba allows UNIX servers to get in on the action, communicating with the same networking protocol as Microsoft Windows products. Thus, a Samba-enabled UNIX machine can masquerade as a server on a Microsoft network and offer the following services:

- Share one or more filesystems,
- Share printers installed on both the server and its clients,
- Assist clients with Network Neighborhood browsing,
- Authenticate clients logging onto a Windows domain, and
- Provide or assist with WINS name server resolution.

Socket Table

From a system integrity perspective, the chief interest for the study of networking in UNIX is looking at the system's socket table. A socket is simply one endpoint of a connection. The system maintains a table of all the sockets it has open. The table can be viewed with the `netstat` command; there are a variety of options to `netstat`, which vary from system to system. We recommend using the following options for the most effective information:

- `netstat -nat` (Linux), and
- `netstat -an -P tcp` (Solaris).

The two tables have a lot of information for the user.

- Proto displays the protocol in use, usually TCP or UDP. Generally only TCP connections are of interest, since UDP is connectionless.
- Recv-Q and Send-Q show how much data is waiting to be read by applications or to be sent over the wire, respectively.
- Local Address displays the address (and port) on which this box is listening for connections. This may be a specific address (commonly the loopback address, which means only applications internal to the box can connect), or it can be listed as 0.0.0.0, meaning that the box is listening on that port globally.
- Foreign Address shows the address on the other end of the connection, this is typically 0.0.0.0 for listeners but a specific IP for established connections.
- State shows a multitude of TCP states exist, but the most common ones to see here are LISTEN and ESTABLISHED. Occasionally TIME_WAIT might be seen.
- Swind contains the offered window size as reported by the remote peer.
- Rwind displays the advertised window size being transmitted to the remote peer.

The socket table is usually the fastest way to tell what network services are running on the box, instead of having to search through the process list and guess which processes may be communicating on the network. They can also be good indicators of suspicious activity on the box, such as a shell running on the network (there's not a good reason for this type of activity), unusual listening ports, or listening processes running from unusual locations, like a user's home directory.

Section Completed

Networking and Name Resolution: Section 5 Transcript

Listening Ports 1/5

Listening ports are communication endpoints for applications and processes. As a listener, these ports are servers. Solaris listens to a lot of default ports, however, a default install of Solaris is not locked down. How many of these services do you wager have remote security vulnerabilities in them?

Some systems, such as Linux, are fairly conservative in their out-of-the-box state and tend not to have a lot of processes listening on the network. Others, such as Solaris, tend to "leak like a sieve."

For more conservative machines without many default listeners, there are still a number of services that are not an immediate security concern. For example:

- SSH - Secure Shell (SSH) is frequently enabled by default for remote access
- SMTP - Most systems use Simple Mail Transfer Protocol (SMTP) for internal purposes
- HTTP/HTTPS - HyperText Transfer Protocol / HyperText Transfer Protocol Secure (HTTP/HTTPS) may be concerning if the machine is not a web server, but many applications permit web management. This is often configured to run on unusual high ports, e.g., 8080, 8888.
- RPC - It may be configured on some machines. It is not a joy to see this service (some of the worst vulnerabilities occur in RPC, but it's not necessarily indicative of a malicious actor.

Make sure to look at the local address on listening services. If the service is listening on localhost (127.0.0.1), it is not a security concern, since other machines cannot access the service.

Typical Running Services 2/5

When looking at network traffic, how do you know what is normal and what is suspicious? You need to know what different machines do and have a baseline of what is normal. Use `/etc/inet/services` to list ports and the services that use them. Comparing and contrasting traffic with the baseline will help you determine if there are services on the network that should not be running.

If you're not using a service, shut it off. Don't leave unnecessary services running as that leaves you more vulnerable.

xinetd/inetd Service 3/5

The Internet service daemon (originally called inetd, then replaced with xinetd in Linux) is a useful, but often vulnerable and misunderstood service available in most UNIX systems. Solaris 10+ uses a system called Service Management Facility (SMF), and keeps inetd as a legacy reference file. The daemon's purpose is to save resources by limiting the number of processes that must be running on a particular server at any one time. It works by listening on the appropriate ports itself, instead of using the actual services. When a connection is made to a port that xinetd is listening on, the service that should be attached to that port is started. Let's see how the daemon works for File Transfer Protocol (FTP).

Upon system startup, xinetd opens a listening socket on port 21, the FTP port. A client then connects to the computer on port 21 and xinetd starts the ftpd service and passes the client connection to it.

The benefit of using the xinetd system is that all of the services provided by a server do not have to be running at boot. For instance, servers commonly provide services like ssh, ftp, and more, which if run traditionally, would all use system resources, even when the services were not being used. The xinetd daemon allows these resources to be saved until they are needed. The configuration for xinetd is relatively simple. The main configuration for the xinetd daemon itself is found in `/etc/xinetd.conf`. Each service to be managed by xinetd has a configuration file in the `/etc/xinetd.d/` directory, the syntax for both of which can be found in `xinetd.conf`. The use of xinetd will be explored in a class exercise, specifically, how the xinetd daemon can be forced to run non-network aware programs over the network to expose a dangerous, hard-to-discover backdoor on the system.

RPC is a framework through which processes can take advantage of functionality provided by another process, most often on another computer across the network. Perhaps the best-known use of RPC is the implementation of NFS. If a system supports RPC, it will have a few processes associated with RPC running:

- portmapper (always on port 111) - on a fixed port, lets clients lookup the ports for the rest of the RPC services.
- status - the RPC information service

On modern systems, RPC is usually handled by the xinetd service.

Backdoors 4/5

A backdoor in a computer system (or cryptosystem, or algorithm) is a method of bypassing normal authentication, securing illegal remote access to a computer, obtaining access to plaintext, and so on, while attempting to remain undetected. The backdoor may take the form of an installed program or through a rootkit.

Exercise Introduction 5/5

It is time for an Exercise. In order to successfully complete the Exercise, you are expected to use your notes and any available resources presented throughout the course, in addition to conducting your own Internet research.

Networking and Name Resolution: Section 6 Transcript

Summary

You have completed the Networking and Name Resolution module. In this module, we reviewed name resolution, network-related utilities, network file systems, socket tables, and backdoors.

You should now be able to:

- Describe the local name resolution process on a UNIX host,
- Analyze the output of network-related utilities,
- List and identify ports on the provided system,
- Describe the purpose of name resolution,
- Describe Network File Systems,
- Analyze entries within a socket table, and
- Create a bash backdoor using xinetd on the provided system.

In addition to these topics, as you traversed through this module, you were expected to conduct research to learn about a variety of UNIX commands.

To receive credit and advance to the next module, you must achieve a passing score on the Module Exam. Click the Next Section button to begin the Module Exam.