realtimepublishers.com®

The Definitive Guide™ To

Enterprise Network Configuration and Change Management

VOYENCE™

Don Jones

Chapter 5: Network Configuration Management Technologies....85
SNMP....86
SNMP Theory of Operation....86
SNMP Use in Network Configuration Management....88
SNMP Considerations....88
Syslog....89
Syslog Theory of Operation....89
Syslog Use in Network Configuration Management....91
Syslog Considerations....92
SSH....93
SSH Theory of Operation....93
SSH Use in Network Configuration Management....94
SSH Considerations....94
TFTP....95
TFTP Theory of Operation....95
TFTP Use in Network Configuration Management....97
TFTP Considerations....99
Telnet....100
Telnet Theory of Operation....100
Telnet Use in Network Configuration Management....101
Telnet Considerations....101
RADIUS....102
RADIUS Theory of Operation....102
RADIUS Use in Network Configuration Management....103
RADIUS Considerations....104
TACACS and TACACS+....105
TACACS Theory of Operation....105
TACACS Use in Network Configuration Management....106
TACACS Considerations....106
Summary....106

Copyright Statement

Chapter 5: Network Configuration Management Technologies

Network configuration management relies heavily on several open-standard technologies. These technologies—such as TFTP and SSH—were designed with entirely different purposes in mind, but serve the needs of network configuration management quite well. However, before implementing any network configuration management solution, you should thoroughly understand the underlying technologies, how they work, and any implications—particularly from a security standpoint—that they bring to your environment.

Some of the technologies I'll discuss in this chapter are defined as open standards in various Internet Engineering Task Force (IETF) Request for Comment (RFC) documents. Where appropriate, I'll provide cross-references to these RFC documents, which you can access online at http://www.ietf.org. Note that RFCs are almost continually superseded by newer versions; be sure to carefully examine any RFCs to make sure you're reading the latest version.

In this chapter, I'll discuss seven of the most important technologies that provide a foundation for effective network configuration management. These include:

- SNMP
- Syslog
- SSH
- TFTP
- Telnet
- RADIUS
- TACACS

For each of these technologies, I'll provide a theory of operation along with a description of how the technology supports a network configuration management solution. I'll also discuss the considerations of each technology, such as other supporting technologies required in the environment, potential security issues, and potential management issues.

SNMP

SNMP is an open-standard management protocol, designed to provide monitoring and control capabilities in a multi-vendor network environment. Almost all modern, managed network devices support SNMP, along with many server OSs and other network elements.

Various aspects of SNMP are defined in several RFC documents, including 1089, 1098, 1157, 1187, 1215, 1227, 1228, 1270, 1303, 1351, 1352, 1353, 1442-1450 (which define SNMPv2), 1901-1910, 2011-2013, and so forth. The most recent RFCs deal with SNMPv3, and include 3410-3418.

SNMP Theory of Operation

SNMP has three basic elements:

- The actual protocol
- The Structure of Management Information (SMI)
- The Management Information Base (MIB)

SNMP defines two basic roles for network devices: a *management station*, which collects SNMP information and is the central point of management on the network, and *agents*, which represent individual SNMP-compatible devices.

SMI (described in RFC 1155) is a framework that explains the basic types of information that SNMP can manipulate. SMI doesn’t define any specific devices—such as routers or switches—but rather defines a basic skeleton for how those devices might be described. The most fundamental concept of that description is the Object Identifier (OID).

An OID is a tag that allows SNMP to refer to a particular managed object. OIDs are structured in a sort of hierarchy. The root of this hierarchy has no label and has three children named ccitt, iso, and joint-iso-ccitt. The most important of these, perhaps, is the ISO child, which defines a child named org. The United States Department of Defense (the original backer of what became the Internet) is listed under the org child, and has defined its own child named internet. This hierarchy, depicted in Figure 5.1, is referred to as *iso.org.dod.internet*.

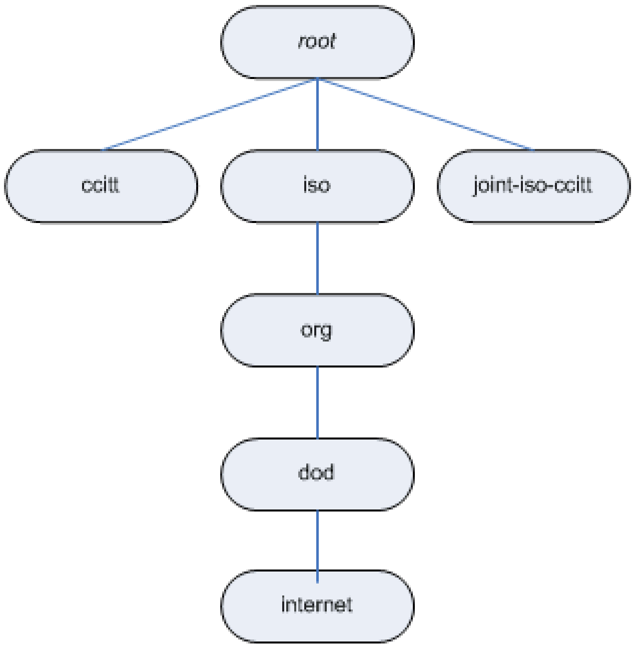

Figure 5.1: The iso.org.dod.internet OID hierarchy.

Iso.org.dod.internet is the core tree used for many SNMP object definitions. Control over this tree's children is managed by the Internet Activities Board (IAB), which reserves a portion of the hierarchy for vendors to define their own objects.

Using SMI as the skeleton, objects are defined at various levels of the object tree. A sub-tree that defines several devices and groups them together is referred to as an MIB. MIBs must be registered with an administrative authority, such as the IAB, which assigns a unique OID to the root of the MIB sub-tree, ensuring that various sub-trees remain unique and don't conflict with one another. For example, a common MIB is the Internet TCP/IP MIB, which is referred to as MIB-II. This MIB includes the objects that are associated with TCP/IP variables and SNMP variables.

Having access to an MIB tells an SNMP management solution—such as a network configuration management solution—about the devices that can be managed. Most network device vendors provide MIBs that describe and define their devices, thus making those devices accessible to SNMP. The primary information described in an MIB is *SNMP variables.* These variables represent all the changeable parameters of a managed device. For example, a router's MIB might define an SNMP variable for the router's name. Querying that variable will reveal the router's name, while *setting* that variable will change the router's name.

SNMP is an efficient way for network management solutions to retrieve data from managed devices and to modify configuration settings on those devices. SNMP itself is generic and can potentially work with any object; MIBs define device-specific parameters, describing to an SNMP management station which parameters are available for the device. For example, a Windows server could implement SNMP and provide an MIB describing how to remotely read and set Windows registry settings. The benefit of SNMP is that it is completely vendor- and technology-neutral—regardless of how a device is designed to be natively managed, an SNMP implementation can ensure that the device can be managed in the same way as any other SNMP device.

Security is a major concern in SNMP. Not only can SNMP be used to change device configurations—an obvious security red flag—but it can also be used to query information from devices, which can lead to security compromises. For example, an improperly secured router could be queried, via SNMP, and made to reveal information about how your network is put together. That information, which would include IP addresses and routes, could be used to architect an effective attack on your network.

I'll discuss security and SNMP in more detail in a moment.

SNMP Use in Network Configuration Management

SNMP has obvious value in a network configuration management solution. In earlier chapters, I described how many devices—such as routers and switches—allow the use of TFTP to dump their configuration information to a file. Other devices, however, might not; SNMP represents a universal constant for most manageable network devices, providing a common way for a network configuration management solution to retrieve configuration settings (which can be saved to a database) and to reconfigure a device, if necessary.

Another aspect of SNMP is *traps*. Traps are essentially event notifications sent from an SNMP agent to an SNMP management station whenever specific events occur. Such events might include an administrator logging off of the device or placing the device into configuration mode. These traps, if passed to a network configuration management solution, can serve as a trigger, notifying the solution that the device's configuration might have changed. The solution can then access the device's configuration, compare it with a previously-saved version, and take the appropriate action as defined by an administrator.

SNMP Considerations

SNMP is a fairly old protocol and recent revisions to it have shown the growing concern about network security. Originally, SNMP's basic security element was the *community string*. This string is a simple text name that defines a group to which SNMP devices could belong. In theory, devices would only respond to management stations possessing the same community string. However, the default community string, "public," is left unchanged on many devices, making it easy for attackers to guess the string and begin attacking SNMP-managed devices. Even changing the community string isn't a completely effective solution because it is fairly easy for an attacker to monitor network traffic for SNMP packets and analyze them to discover the community strings in use on the network.

SNMPv3—the current version of the protocol—includes enhancements that provide for data encryption of SNMP packets. That encryption originally used the Data Encryption Standard (DES) and was later upgraded to the more secure Triple-DES; it can optionally use the newer Advanced Encryption Standard (AES). SNMPv3 also includes authentication mechanisms to ensure that only authorized management stations can query and modify the configurations of managed devices. This authentication mechanism is based on MD5 and SHA hashes, which help protect device passwords from being discovered by an electronic eavesdropper on your network.

Why bother encrypting traffic on your firewalled internal network? See the sidebar "It's Your Internal Network—Why Encrypt?" later in this chapter.

SNMPv3, however, is still not universally implemented. Older versions of SNMP should be considered marginally safe only behind a firewall; even then, the threat of compromise—particularly if SNMP packets are passed across insecure wireless networks—is very real. To properly secure your SNMP environment, move to the more secure SNMPv3 on all agents and management stations. Regularly change your community strings using long, nonsense strings—such as "m5hfyf84ba"—that are difficult to guess.

> Many network devices that support SNMPv3 continue to also provide support for older versions of SNMP. Be sure to configure your devices to use only the latest version possible, which will help ensure the most secure implementation.

Another consideration, from the viewpoint of an SNMP-enabled network configuration management solution, is the existence of other SNMP management software. Although devices don't mind accepting queries from multiple management stations, devices will typically only send SNMP traps to a single location. To support multiple independent SNMP systems, one of them will need to receive all traps and be configured to pass the traps on to other systems, as necessary. Most enterprise management frameworks support this capability, allowing additional SNMP-capable systems to be "plugged in" to receive copies of SNMP traps received by a central trap-handling station.

Syslog

Syslog is essentially a remote logging protocol for network devices and servers. Rather than generating their own onboard log files (which many devices can still do), log messages are sent to a central logging server—the *Syslog server* or *collector*—and stored in a central database. Syslog server software—sometimes referred to as a *Syslog daemon*—is available for most major server OSs, including Linux, UNIX, Windows, and so forth.

> The Syslog protocol isn't defined in an RFC, but there is an informational RFC—number 3164—that describes the observed behavior of the protocol.

Syslog Theory of Operation

Syslog is a fairly simple technology. Devices simply send a properly formatted message to the Syslog server, which dutifully records the messages in its database. There are no standards for the content of Syslog messages, meaning individual device vendors generally make up their own log message formats. Syslog doesn't define any acknowledgement of the message, so it is possible for the message to be lost on the network (if, for example, the Syslog server is unavailable at the time).

> RFC 3165 defines a reliable delivery mechanism for Syslog messages, in part requiring devices to queue messages until receipt is acknowledged by the Syslog server.

Syslog operates over the User Datagram Protocol (UDP), which provides connectionless data transport. Syslog operates on UDP port 514 by default, although many administrators prefer to change this port so that bogus messages cannot readily be sent to the Syslog server by an attacker.

Syslog also includes the concept of a *relay*, which is simply a collector that forwards all, or select, Syslog messages to another collector. Relays can be used in remote sites to help aggregate message traffic and remove the need for multiple devices to rely on a WAN connection for sending Syslog messages. Figure 5.2 illustrates the use of devices, relays, and connectors in a large network.

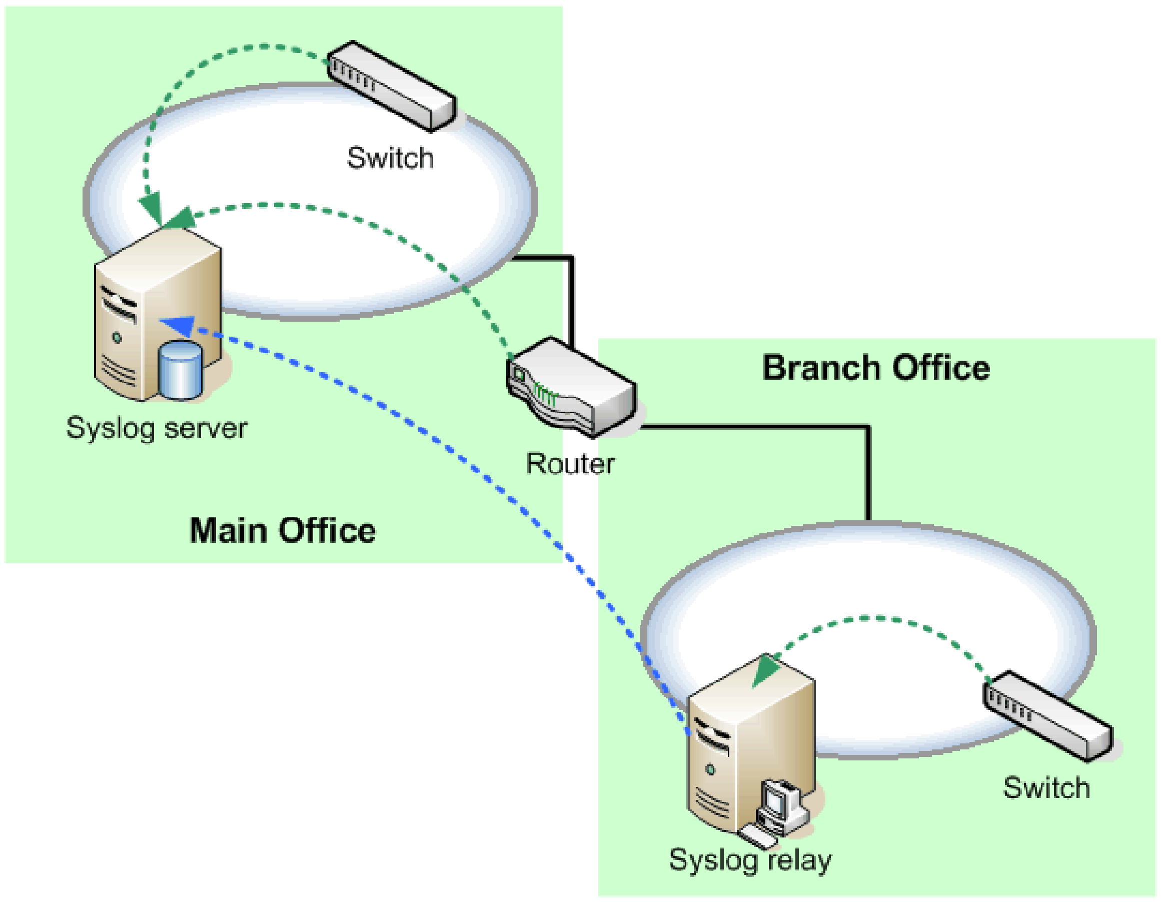

Figure 5.2: Syslog on a large distributed network.

One trick to Syslog is that messages cannot exceed 1024 bytes in length. This restriction is partly a result of the connectionless nature of UDP that makes the delivery of larger messages problematic.

Syslog packets consist of a few basic parts:

- The PRI part, which defines the type of message (kernel messages, system messages, security messages, and so forth). A severity indication (emergency, warning, notice, and so on) is also included.
- The HEADER part, which contains a timestamp and the hostname or IP address of the sending device.
- The MSG part, which consists of the remainder of the packet and can contain any valid text. As previously mentioned, there are no strict rules for formatting this section, although individual device vendors tend to pick a format appropriate for their devices and stick with it. The following shows an example packet:

  ```
  <34>Jun 14 13:27:45 device14 su: ‘su root’ failed for user4 on
  /dev
  ```

Syslog Use in Network Configuration Management

Syslog doesn’t seem to offer a direct role in network configuration management. After all, Syslog doesn’t provide access to device configurations, and the messages contained in a Syslog log file aren’t necessarily in a consistent format. However, like SNMP, Syslog can provide vital triggers that tell a configuration management solution that a device’s configuration might have changed, cueing the solution to examine the device configuration to determine what has happened.

Change management solutions can do so in one of two ways. The seemingly most straightforward way would be to simply perform a periodic scan of the Syslog database, looking for messages—such as an administrator placing a device into configuration mode—that might relate to a device configuration change. The configuration management solution would need to know which messages to look for—this task is readily done by simply examining the messages produced by the top manufacturers’ devices. The problem with this technique is that it introduces a delay—potentially a significant one—between the time the Syslog server receives the message and the time the configuration management solution scans for it.

A better technique is to utilize Syslog’s relay mechanism, turning the configuration management solution into a Syslog relay. If all devices forward Syslog messages to the configuration management solution, the solution can examine them in real-time and take whichever action is necessary. The solution can also relay the messages to your actual Syslog server for permanent logging. Figure 5.3 shows how the configuration management server can fit into the relay scheme.

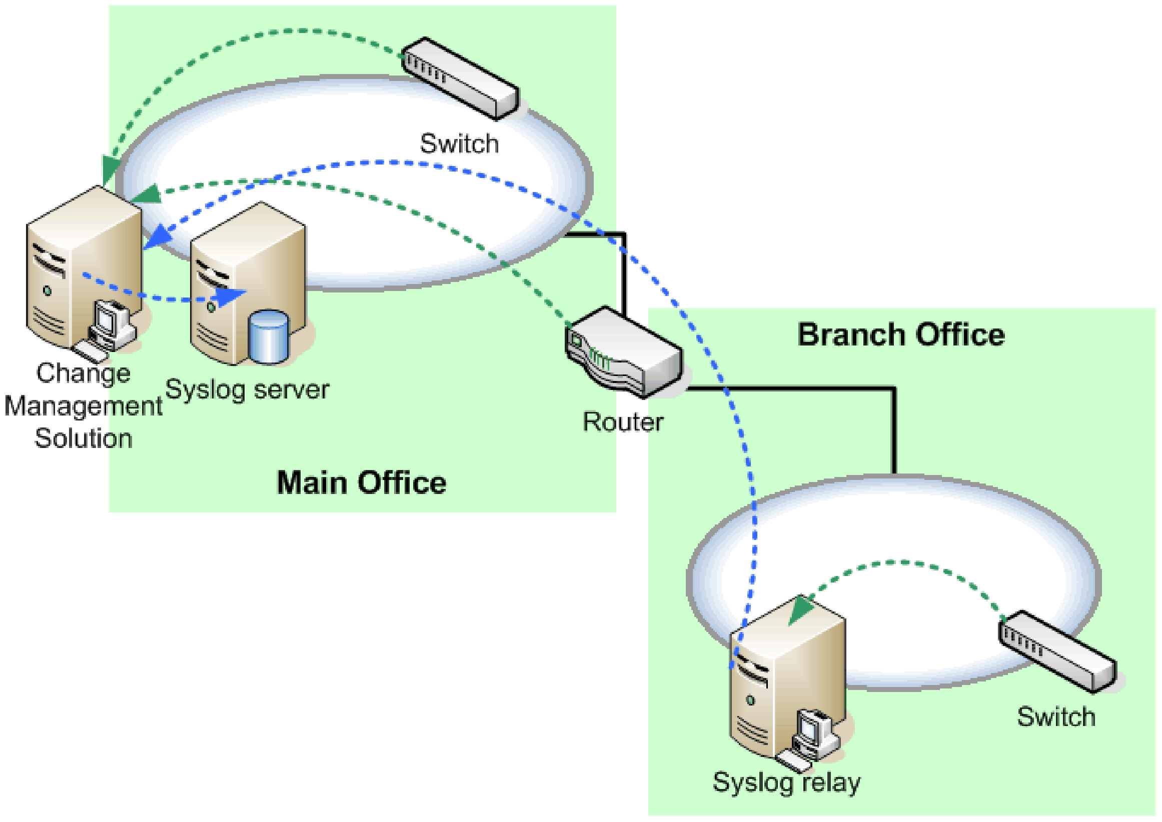

Figure 5.3: Using Syslog relay to examine device messages in real time.

Syslog Considerations

Syslog offers a few security considerations. First, messages are transmitted entirely in the clear; whether that presents a problem depends primarily on what information your devices include in their Syslog messages. Most Syslog messages are fairly useless to attackers, although these messages do occasionally provide clues about the network's infrastructure and design.

Syslog is constantly evolving. Work is being done to add authentication capabilities to Syslog and to digitally sign Syslog messages to help prevent spoofing.

A more serious security threat is a Denial of Service (DoS) attack on the Syslog server itself. Such an attack could prevent the server from accepting Syslog messages from devices, a condition that could be used to disguise an in-progress attack on those devices. This type of multi-horned attack relies on the fact that Syslog doesn't guarantee delivery; under-attack routers, for example, might be firing off critical messages at the Syslog server, which simply ignores them because it is being flooded with a DoS attack. The routers never realize that their messages aren't getting through. This scenario is the primary argument for changing Syslog to operate on a nonstandard port, thus making it marginally more difficult for an attacker to launch a DoS attack. Other precautions include configuring the Syslog server to only accept connections from known device IP addresses, which would require attackers to take additional steps (IP spoofing) in order to conduct an attack on the server.

SSH

It seems like SSH, Secure Shell, has been around forever; it certainly has been around for a long time in UNIX/Linux circles. Many network devices are built on a proprietary UNIX-like OS, so it makes sense that many would therefore incorporate SSH. SSH provides a way for one computer to log on to another, in order to execute commands on the remote computer. In terms of network management, SSH provides a secure means of logging on to network devices in order to control them and alter their configuration.

SSH isn't defined in an RFC; it has been a standard component of most UNIX implementations for years. You can read a description of it at http://www.employees.org/~satch/ssh/faq/ssh-faq-1.html. With regard to the IETF, SSH is defined in an Internet-Draft (actually several of them) that you can access at http://www.ietf.org/ids.by.wg/secsh.html.

SSH Theory of Operation

There are two versions of SSH in widespread use: SSH1 and SSH2. The two are drastically different in terms of their underlying workings:

- For encryption, SSH1 uses DES and SSH2 can use either DES or Triple-DES (3DES). SSH1 can also use lesser-known cipher algorithms such as IDEA and Blowfish; SSH2 supports Blowfish, Twofish, Arcfour, and Casr128-cbc.
- For authentication, SSH1 uses RSA encryption and SSH2 uses DSA.

Both versions authenticate using a variety of techniques:

- A password file (such as /etc/passwd in UNIX implementations)
- A user's public key (RSA or DSA, depending on the SSH version)
- Kerberos (SSH1 only)

Practically speaking, network devices will tend to use the first technique or combine SSH with RADIUS- or TACACS-based authentication.

SSH consists of several sub-protocols:

- SSH-TRANS—Provides server authentication, encryption, and integrity; this transport-layer protocol can also provide data compression.
- SSH-USERAUTH—Provides user-to-server authentication and runs over SSH-TRANS
- SSH-CONNECT—Runs over the SSH-USERAUTH protocol and multiplexes the encrypted tunnel into several logical communications channels

A key concept in SSH is that of the *host key*. Each server (or network device) accepting SSH connections should have a host key, which serves as the basis for identification. SSH sessions begin with a key exchange, and the host key allows a client to verify that the client is talking to the server it intended—thus preventing an attacker from spoofing the server and capturing a client connection. Of course, the client must have prior knowledge of the correct key for comparison purposes. In practice, clients generally obtain this prior knowledge from a local database that associates each host name with a public host key. Other techniques, including the use of public certification authorities, exist but are generally not practical in terms of a network device's internal SSH implementation.

SSH Use in Network Configuration Management

Many network devices incorporate SSH. Configuring a network device to use SSH requires several basic steps, including enabling the SSH transport support and creating the SSH encryption keys that will be used. The following example shows the IOS commands necessary to enable SSH in a router:

```
hostname hostname

aaa new-model
username user password 0 user

ip domain-name company.com

cry key generate rsa
ip ssh time-out 60
ip ssh authentication-retries 2

line vty 0 4
transport input SSH
```

Many devices can create a debug log of SSH output, which can help an administrator who is troubleshooting the protocol. The following sample if from a debug log from the SSH process:

```
00:23:20: SSH0: starting SSH control process
00:23:20: SSH0: sent protocol version id SSH-1.5-Cisco-1.25
00:23:20: SSH0: protocol version id is - SSH-1.5-1.2.26
00:23:20: SSH0: SSH_SMSG_PUBLIC_KEY msg
00:23:21: SSH0: SSH_CMSG_SESSION_KEY msg - length 112, type 0x03
00:23:21: SSH: RSA decrypt started
00:23:21: SSH: RSA decrypt finished
00:23:21: SSH: RSA decrypt started
00:23:21: SSH: RSA decrypt finished
00:23:21: SSH0: sending encryption confirmation
00:23:21: SSH0: keys exchanged and encryption on
00:23:21: SSH0: SSH_CMSG_USER message received
00:23:21: SSH0: authentication request for userid cisco
00:23:21: SSH0: SSH_SMSG_FAILURE message sent
```

Why use SSH in network configuration management? Unlike other protocols—such as Telnet—that don't offer any form of transport security, SSH provides remote command-line administration over an authenticated, encrypted connection. This encryption ensures that attackers can't learn precious network configuration information—such as passwords—by capturing and analyzing network traffic. Network configuration management solutions that can connect to network devices by using SSH are inherently more secure and less prone to reveal sensitive network configuration information to attackers.

SSH Considerations

Most security considerations around SSH relate to SSH1, which has some structural weaknesses that leave it open to a fairly broad number of attacks. Partly for this reason, SSH1 is no longer being developed and SSH2 is highly recommended. SSH1 is, however, a simpler protocol and is available on a wider variety of platforms, which is why it is still fairly common. SSH1 has more options for authentication and performs better than SSH2.

Another consideration for using SSH—either version—is actually a concern for any software that uses encryption. Many countries, most notably France, Russia, and Pakistan, require special permits to use the levels of encryption provided by either version of SSH. These considerations might make it inadvisable to utilize SSH in network operations in some countries.

Frankly, the most basic consideration for using SSH is the complexity—or at least the perceived complexity—of setting it up. As I've demonstrated, implementing SSH on a network device doesn't need to be complicated. *Any* level of data encryption is preferable to none.

It's Your Internal Network—Why Encrypt?

There is sort of a feeling among network administrators that most of their network devices are behind a firewall and that their office has locks or card keys or some other access control, so why bother encrypting traffic in this "safe" environment? The security concept of *defense in depth* demands more than just one level of protection.

For example, consider the almost ridiculous infection rates of recent Trojan-carrying viruses such as Bagel, NetSky, and MyDoom. Among their other capabilities, these viruses provide a means for an external attacker to use computers on your internal network for their own purposes. This capability can give an attacker an instant "in" to your supposedly "safe" network traffic, where they can capture and analyze traffic to determine the structure of your network, capture passwords, and so forth.

Defense in depth suggests that you shouldn't trust your firewalls or door locks—instead, assume that every level of security will be penetrated by a determined attacker. Using SSH to encrypt network device management makes sense, then, as a last line of defense when the physical network is compromised.

TFTP

The Internet's File Transfer Protocol (FTP) is perhaps one of the top two application protocols used (the Web's HTTP is number one). Trivial FTP (TFTP) is less well-known but plays an important role in network device management, particularly in network configuration management systems.

📖 TFTP is formally defined in RFC 1350 (and certain aspects of using uniform resource identifiers with TFTP are defined in RFC 3617). RFC 1350 actually defines TFTPv2; the protocol was originally defined in RFC 783. RFCs 2347-2349 define a set of extensions to TFTP; RFC 2090 defines multicast options for TFTP.

TFTP Theory of Operation

Perhaps the biggest difference between FTP and TFTP is the underlying transport protocol. FTP uses the connection-oriented TCP, and TFTP relies on the connectionless UDP. TFTP is considerably less feature-rich than FTP to keep both TFTP client and server software uncomplicated and easy to use. TFTP is pretty much limited to reading and writing files; it cannot list directories, and the core protocol does not have any mechanism for encryption or user authentication (although some implementations, and proposed extensions to the protocol, do provide these features).

Unlike many UDP-based protocols, TFTP takes steps to ensure packet delivery. However, because UDP doesn't provide any underlying means for reliable delivery, TFTP must do so. It's worth a quick review of how regular FTP, over TCP, operates, so that TFTP's method can be better understood.

In TCP, the sender's protocol stack sends packets of data in sequential order. The receiving computer reassembles these packets. The recipient maintains a *sliding window*, as illustrated in Figure 5.4. As illustrated, this recipient is missing one packet from the sequence, indicating a network failure of some kind. Because the sliding window is full of packets, the recipient will stop acknowledging new packets and will ask the sender to retransmit the missing packet.

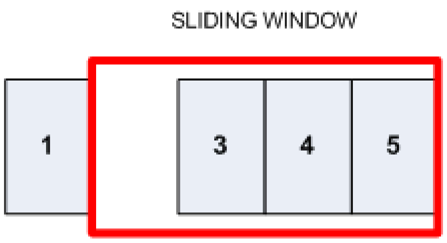

Figure 5.4: Sliding window on recipient's TCP/IP stack.

Once the missing packet is retransmitted and received, the sliding window will advance, and the client will begin accepting new packets, as Figure 5.5 shows.

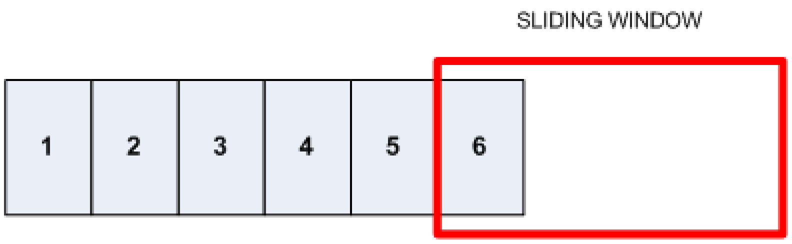

Figure 5.5: Sliding window ready to accept new packets.

The underlying TCP is doing all the work. The benefit of this method is that the sender doesn't need to keep track of individual packets that have been received, and the recipient doesn't need to acknowledge each packet received. With UDP, however, none of this underlying architecture exists. Instead, TFTP is designed to transmit one packet from sender to recipient, then wait for the recipient to confirm receipt. If the recipient doesn't do so promptly, the sender simply retransmits the packet until either a confirmation is sent back or the TFTP session times out and the sender gives up. Figure 5.6 illustrates the interaction between a TFTP server and client.

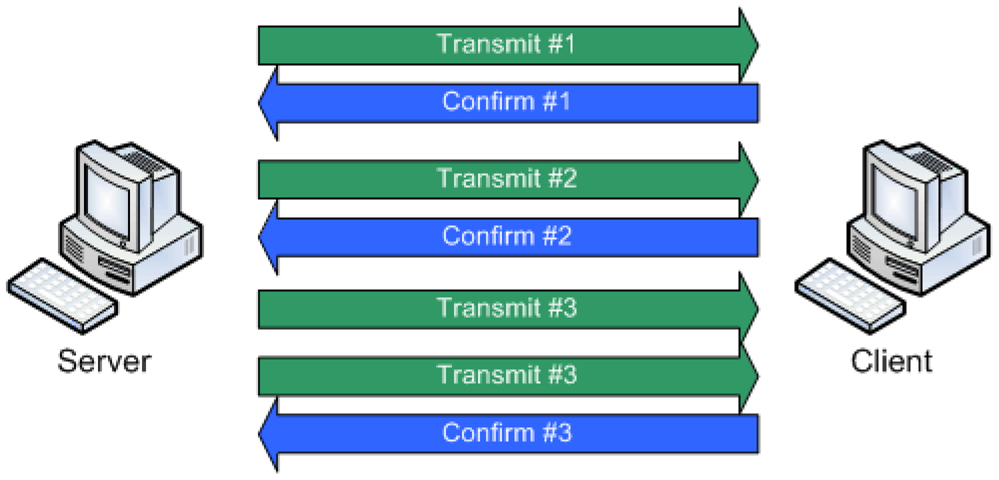

Figure 5.6: Packet exchange in TFTP.

TFTP supports only five types of packets:

- Read request
- Write request
- Data
- Acknowledgement
- Error

Any error on either end of the connection results in that end sending an error packet and immediately terminating the session without waiting for acknowledgement of the error. The other party's only option is to start over; TFTP does not provide for any resume capability (newer implementations of FTP provide this feature). Any data packet that is less than 512 bytes is considered the last packet in the transfer; after that packet is acknowledged, the session is terminated automatically. As you can see, TFTP is an extremely simple protocol designed to provide minimal file transfer functionality.

TFTP Use in Network Configuration Management

TFTP is a major player in network device management because it represents the most efficient way for network devices to dump their configuration files and to read new configuration files. Almost all managed network devices can act as a TFTP client, writing and reading configurations to and from a TFTP server. A network configuration management solution can work in conjunction with a TFTP server (or, more commonly, act as a TFTP server) to move configuration data to and from network devices.

For example, the following IOS commands will force a router to read a configuration file from a TFTP server, then implement that configuration file:

```
Escape character is '^]'.
User Access Verification
Password: ****
router> enable
Password: ****
router# config network
Host or network configuration file [host]?
Address of remote host [255.255.255.255]? 123.123.123.123
Name of configuration file [router-confg]? router-confg
Configure using router-confg from 123.123.123.123? [confirm]
Loading router-confg from 123.123.123.123 (via BRI0): !
[OK - 1265/32723 bytes]
router# write
router# exit
```

Network configuration management solutions can automate this process, logging on to devices via SSH or Telnet and commanding them to write their configuration files to a designated TFTP server (which might be running as part of the configuration management software). The solution can then pull the configuration data into its database, compare it with prior configurations, analyze it for security issues, and perform whatever other operations are necessary. To roll back a device's configuration, the solution can again access the device via Telnet or SSH and command it to retrieve a configuration file from the designated TFTP server (which again might be the configuration management software itself).

Why use TFTP instead of SNMP queries? SNMP generally allows only a single variable to be read at a time, meaning the configuration management software would need to issue potentially thousands of queries to pull a device's entire configuration. With TFTP, the entire configuration can be conveniently written to a single file in one easy operation.

TFTP Considerations

TFTP is lacking in even the most rudimentary security capabilities. Potential dangers include:

- Data is transmitted in clear text, making it subject to capture and analysis. Because no network should truly be considered secure (due in part to the prevalence of Trojan horse viruses that provide attackers with a presence on otherwise safe networks), this concern is important. Device configuration files contain incredibly useful data upon which an attacker can construct attacks.
- There is no verification of the client or server in TFTP, opening the possibility for packet spoofing. Bogus configuration data could be fed to a network configuration management solution through packet spoofing, or a router could be fooled into sending configuration data to a bogus TFTP server.

Although more secure implementations of TFTP have been proposed and produced, the TFTP implementations commonly available in network devices support only the basic, insecure TFTP standard. Unfortunately, there is not much that can be done to improve this situation. You could propose a scenario in which device management is done over an independent, more secure physical network; however, such scenarios are expensive, complicated, and impractical. What you can do is try to minimize the security exposure of TFTP:

- Implement solid fist-level defenses, including firewalls, multiple levels of virus scanning, and so forth. If you can better ensure the security of your physical network, the number of potential attacks on TFTP is significantly reduced.
- Ensure that configuration data is stored securely. Capturing configuration files from TFTP while in-transit requires a persistent presence on your network; if you can reduce the likelihood of that occurring, then attacking the stored configuration files is the only remaining vulnerability. Network configuration management solutions, for example, should store configuration data in a secured, encrypted database.
- Regularly change device passwords and configure devices to require a password even for read-only access to their configuration files. Frequent password changes and secure read-only access will limit the opportunities for an attacker to obtain device configuration files.
- Try to restrict the availability or usefulness of packet capture software. For example, in a fully switched network, packet capture software is almost useless if the switches don’t provide a promiscuous port—which is a switch configuration that you can enforce.

Telnet

Telnet is sort of the grandfather of remote management tools, although it didn't start that way. It was originally intended as a terminal emulator, allowing relatively "dumb" terminal workstations to connect to a more powerful mainframe. Today, Telnet provides remote command-line access for most UNIX-based systems and for most managed network devices.

Telnet has been around since the beginning of the network that would eventually become today's Internet, having been first defined in RFC 97 back in 1971. The most recent RFCs dealing with Telnet include 2941-2953 for Telnet encryption and authentication, 1571-1572 for various Telnet options, and so forth. Searching the RFC index at http://www.ietf.org/iesg/1rfc_index.txt for the word "Telnet" yields a surprising number of hits, underscoring the foundational nature of this protocol.

Telnet Theory of Operation

Telnet is perhaps one of the simplest protocols associated with the Internet, and is certainly one of the older ones, having been first defined in 1971. Essentially, a Telnet client sends your keystrokes to a remote computer, which processes them however it needs to. The output from the remote computer is sent to your Telnet client, where it is displayed for you to read. You can think of Telnet as a sort of remote keyboard, electronically attached via TCP/IP to a distant computer.

A large number of other Internet protocols are based on Telnet. The SMTP and POP3 protocols, in fact, simply use Telnet sessions with a defined, automated format for interaction between server and client. You can even use a Telnet session to, for example, conduct a manual SMTP session by having the client connect to an SMTP server on port 25 (the default SMTP port). Knowing that Telnet underlies other protocols makes it easier to understand how those protocols work and what operational or security issues they might contain as a result of being based on Telnet.

Telnet operation is simple: Character codes are sent and received. There is a bit of complexity involved in newer options such as flow control, the ability to send extended characters (those beyond ASCII 128), and so forth; for the most part, however, these features won't affect a network configuration management implementation.

You might need to know some specifics when connecting to older devices; Telnet is designed to emulate several popular terminal devices, such as VT-100, TTY, 5250, and so forth; older network devices might require a Telnet connection for a specific emulation mode—check their documentation for details. Newer devices often support a range of emulation modes and will negotiate one with the Telnet client.

Due in part to its age—Telnet was created at a time when only a few hundred people had access to the network that would become the Internet—Telnet doesn't incorporate much in the way of security. Certainly, the remote system (such as a network device) can require you to enter a username and password to gain access, but Telnet passes that information—in the form of keystrokes—in clear text.

There isn't really an official "secure Telnet," per se; SSH was, in fact, developed to overcome the security vulnerabilities in Telnet and can be considered an official "secure Telnet" protocol. Implementations of Telnet exist that utilize SSL/TLS to encrypt the data channel (or some other cryptographic means of encrypting data), but these implementations aren't widespread or generally supported by a broad range of network devices.

> Search Google for "secure telnet" and you'll get several thousand hits; most of what you'll see, however, are pages referring to SSH by the name "secure Telnet." The two protocols share a similar purpose and work in much the same way, but SSH provides encryption and authentication. SSH was built from the foundation established by Telnet, so it's fair enough to call it "secure Telnet." In fact, several RFCs refer to it by that name, although there's a lively debate among purists to call it SSH instead.
>
> You'll also find references to a Cisco offering named "secure Telnet," which is, in fact, a gateway service that provides Cisco support engineers with secure access to Cisco CallManager products inside your network. It isn't really a generic "secure Telnet" that you can deploy for your own use.

Telnet Use in Network Configuration Management

Telnet is a part of daily life for most network administrators—most of whom, should be using SSH, as I'll discuss next—and can be a big part of network configuration management. Configuration management solutions can use Telnet to log on to network devices, request a configuration dump via TFTP (or upload a new configuration via TFTP), and much more. These tasks can be performed equally well with SSH in most cases.

Telnet Considerations

Simply put, there is no such thing as secure Telnet, unless you're referring to SSH, which I've discussed already. The complete lack of any security in the basic Telnet protocol is reason enough not to use it unless you have absolutely no choice. Although you might believe that your network is secure and safe behind a firewall and a bevy of virus scanners, there is *always* the possibility that someone will gain access to your network despite those measures.

Take the Nimda worm as an example: When it first arrived on the scene, it infected thousands of computers before antivirus vendors realized it even existed and created a definition for it so that their products would start catching it. It was probably weeks later before everyone *installed* that definition. Had Nimda contained a clever enough "backdoor" capability, plenty of supposedly secure networks would have been opened to the worm's creators. The fact that backdoor viruses haven't become common (until recently, with MyDoom) is due more to luck than any actual technological hurdle.

The point is that Telnet is an incredibly unsafe way to be transmitting device administration credentials, including passwords, across your network. It's an insecure way to pass *any* kind of configuration data, and you should try not to use it. Most network devices support SSH as a more secure alternative, and most network configuration management solutions are just as capable of using SSH as they are Telnet to access your devices.

RADIUS

RADIUS was originally created to serve two purposes. First, it was intended to provide a centralized user authentication mechanism for remote access concentrators. These concentrators would accept dial-up connections from users, then pass the users' credentials to a RADIUS server, and the RADIUS server would send back a message saying whether the user was allowed to log on. Second, RADIUS provided a centralized logging—or *accounting*, to use the RADIUS term—database. The same concentrators could send messages indicating how long users stayed dialed in, for example, so that user-based billing records could be created. Today, RADIUS is still used for much the same purpose (although dial-up might be replaced by VPN-based remote access concentrators in many cases) and it can provide a similar role for network management.

First defined in RFC 2058-2059 in 1997, RADIUS has seen a number of enhancements and revisions since then, documented in RFC 2138-2139, 2548 (which covers Microsoft-specific RADIUS attributes), 2618-2621, and so on. Search the RFC index at http://www.ietf.org/iesg/1rfc_index.txt for a complete list of RFCs dealing with RADIUS.

RADIUS Theory of Operation

RADIUS is an extensible protocol that relies on UDP for communications. Keep in mind that RADIUS provides three distinct functions: authorization, authentication, and accounting. These functions can be implemented on a single server or, as Figure 5.7 shows or on different servers.

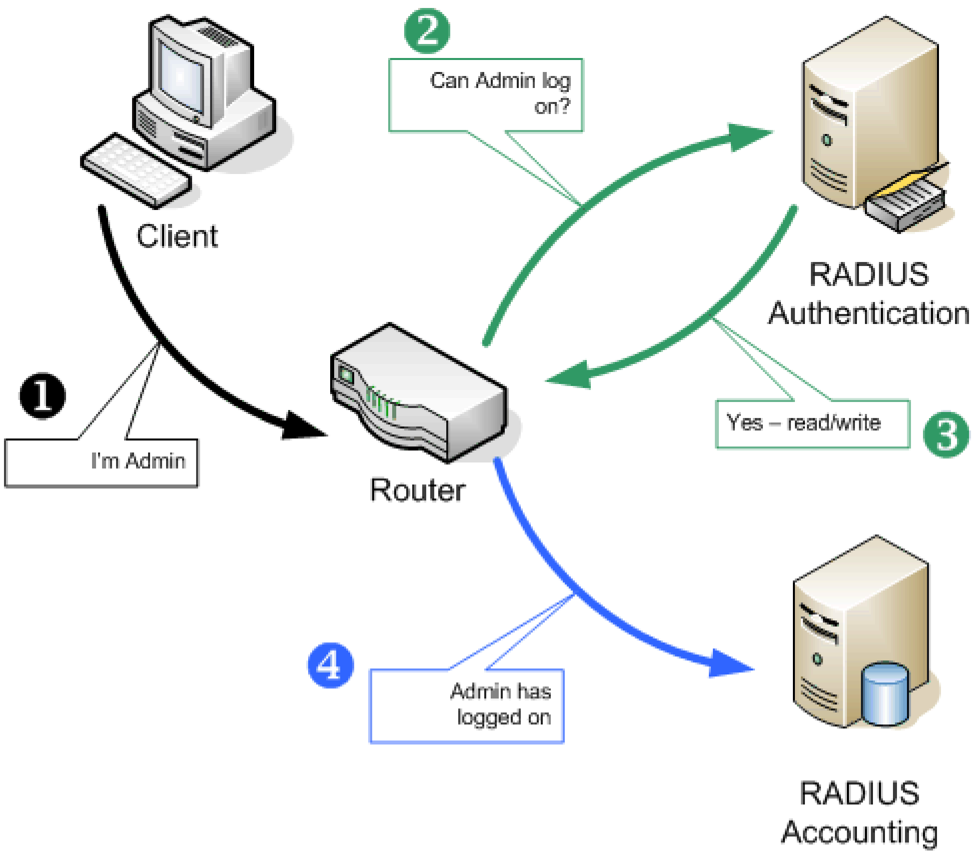

Figure 5.7: RADIUS operations.

The following list explains what is happening in Figure 5.7:

1. A router administrator uses Telnet or SSH to connect to a router, and provides authentication credentials.
2. The router passes those credentials over a secure connection to the RADIUS authentication server.
3. The authentication server responds to the router; in this case, the server indicates that the user is authorized. The sever can also return information describing the user's access permissions, which in this case are read/write.
4. The router can fire additional traffic to a RADIUS accounting server, which keeps a log of user activity—similar in some ways to Syslog.

RADIUS is considered extensible because it supports the use of vendor-specific attributes. The router, for example, could include information about its model number or about the manner in which the administrator is connecting—Telnet or SSH. Those attributes can be included in the RADIUS server's authorization consideration. For example, administrators might only be authenticated if they connect via SSH.

A major benefit of RADIUS is that it centralizes the account database. RADIUS doesn't implement its own account database in most instances; instead, it uses an exiting database—such as an enterprise LDAP directory or UNIX password file—for authentication. Because multiple devices can utilize a single RADIUS server for authentication and authorization you no longer need to maintain complex permission lists individually on each network device you manage.

Communications between the RADIUS server and RADIUS clients—such as routers—are secured through symmetric encryption. The RADIUS server defines a password, and clients must know the same password. This password is used as an encryption and decryption key for traffic between the server and client. Many RADIUS implementations also support IPSec for encryption, including public-key encryption, but few network devices provide similar support.

RADIUS might at some point be superseded by newer technologies. One promising contender is DIAMETER, which has been in the IETF working group phase for some time now. DIAMETER includes server failover specifications, use of TCP (instead of UDP) for more reliable client-server communications, and support for IPSec transmission-level security.

RADIUS Use in Network Configuration Management

RADIUS plays an obvious role in network configuration management, in that it can make it much easier for a network manager, or network management solution, to access a variety of devices. Rather than configuring each of your network devices with an individual password to be used by the network configuration management solution, you configure each network device to use RADIUS, allowing the network configuration management solution to use a single, universal set of credentials. This technique also makes it more practical to change those credentials on a regular basis because the credentials are centralized (not individually defined on dozens of devices).

Like Syslog, RADIUS can play a less-obvious role in network configuration management. Many network configuration management solutions can either scan RADIUS accounting logs or act as a RADIUS accounting server. This capability enables the solution to read or intercept RADIUS accounting messages, using them as a trigger to detect events that might potentially result in a device configuration change. For example, an accounting message that indicates a device has entered configuration mode is a clue that the configuration might have changed; the solution then knows to access the device, dump its configuration, and analyze that configuration for changes.

Configuring most network devices to use RADIUS is a fairly straightforward task. For example, to configure a router, you would simply use the following commands in global configuration mode:

```
Radius-server host hostname ip-address
  auth-port port1 acct-port port2
Radius-server key string
```

These commands specify the RADIUS server hostname, IP address, and, optionally, alternative ports for both authentication and accounting traffic. The second command specifies the RADIUS server password.

RADIUS Considerations

Like Syslog, the RADIUS protocol relies on connectionless UDP traffic. An inherent security flaw in any UDP traffic is that the traffic is not acknowledged at the transport level. In other words, the sender—in this case, a network device—simply sends the traffic and then forgets about it. In the case of authentication or authorization traffic, a response is expected from the RADIUS server, and the device won’t continue until that response is received; thus, no security concerns exist.

In the case of accounting traffic, however, the device is not expecting a response from the RADIUS accounting server. It would be possible for an attacker to perform a DoS attack against the accounting server, potentially preventing that server from accepting and logging messages from one or more network devices. An attacker could thus disguise attacks directly against those network devices. There is little protection against this type of attack; DoS attacks are inherent to TCP/IP traffic and difficult to defend against. The best defense is to simply ensure that such an attack can’t take place on your network and to use an intrusion detection system to watch for such attacks and alert you to them.

Because the RADIUS protocol calls for the encryption of authentication traffic, RADIUS is fairly immune to the type of password-harvesting attacks that can be used against Telnet traffic. However, keep in mind that RADIUS does not protect the *entire* path of password traffic. Figure 5.8 illustrates the various points of vulnerability.

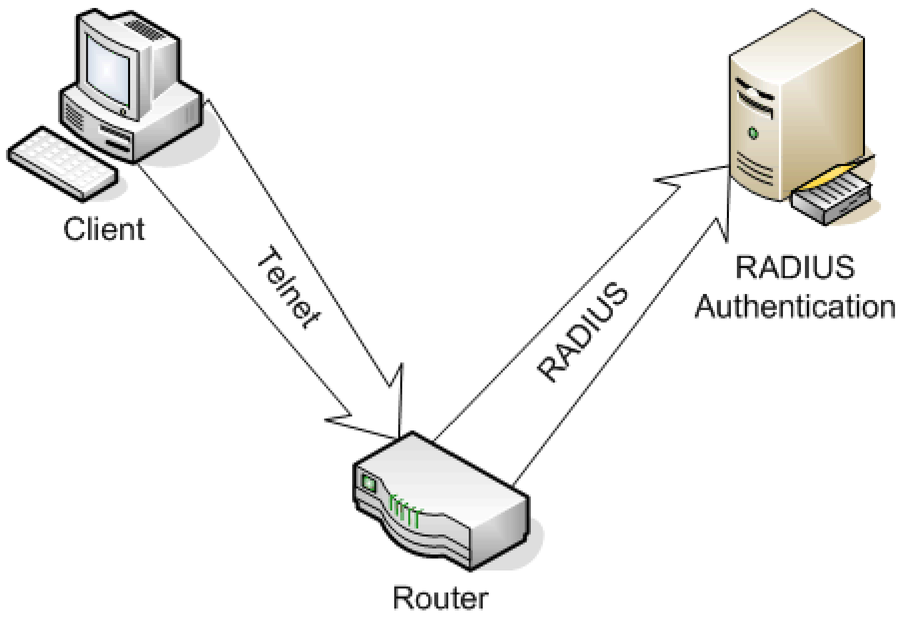

Figure 5.8: Potential password-harvesting opportunities.

In this example, an administrator is logging on to the router via Telnet, which is an unencrypted protocol. Although the router-to-RADIUS traffic containing the administrator's password is encrypted, the Telnet session over which the administrator typed that password is *not* encrypted and is subject to packet capture and analysis. A better solution would be to use SSH instead of Telnet, thus providing an encrypted channel for each level of traffic carrying the password.

TACACS and TACACS+

TACACS is actually an older technology than RADIUS, although the two serve similar functions. Today, TACACS—in the form of the enhanced TACACS+—tends to be seen primarily in Cisco devices. Cisco is one of the few major companies with a commitment to TACACS, and most Cisco platforms support TACACS natively.

TACACS is first mentioned in RFC 927, which appeared in 1984, and has been included in several RFCs since, including RFC 1492, which is perhaps the most comprehensive RFC on TACACS.

TACACS is not an IETF standard; RFC 1492 is an informational RFC that describes the protocol; the actual standard definition is protected by a copyright. The primary implementation of TACACS seen today was developed by Cisco and is used almost exclusively by the company's network devices. TACACS+ is not an IETF standard, meaning it's not defined in an RFC.

TACACS Theory of Operation

TACACS (or TACACS+, which for the purposes of this discussion are the same) works in pretty much the same fashion as RADIUS. There are differences in the actual interaction between clients and servers, and TACACS doesn't support the same degree of extensibility (through vendor-specific attributes) that RADIUS does. However, both protocols provide essentially the same authentication, authorization, and accounting services for network devices.

> ✎ Let me be clear about terminology: Although TACACS is often used as a generic term, the actual TACACS protocol is dated and not often seen anymore. A later version, Extended TACACS or XTACACS, is also out of date. What you'll primarily see these days is TACACS+, which is a completely new protocol that is not compatible with either TACACS or XTACACS. However, all three serve the same *function,* so it is common to refer to them, generically, as TACACS.

TACACS Use in Network Configuration Management

TACACS has a similar role to RADIUS in network configuration management, with the exception that TACACS is primarily supported in Cisco devices. In an all-Cisco environment, TACACS might be preferable to RADIUS simply because Cisco provides so many TACACS components, including the TACACS server software. In mixed environments, RADIUS generally represents a more universal solution.

Most network configuration management solutions (at least, third-party ones) try to support as many vendors' devices as possible, so you might find it more difficult to find a solution that supports TACACS directly. Instead, most solutions will support RADIUS accounting, again making RADIUS a more universal component of network configuration management.

TACACS Considerations

TACACS+ uses a different security model and different security mechanisms than prior versions of the TACACS protocol; because TACACS+ is what you'll mainly see in production environments, I'll focus on it for security considerations.

TACACS+ uses connection-oriented TCP traffic, making it a bit more secure than RADIUS (or TACACS) when it comes to accounting traffic. Because TCP ensures reliable packet delivery, a device "knows" when the TACACS+ accounting server hasn't received its messages. What the device does about it depends on the device; at the very least, the device should attempt to resend the packet. TACACS+ does encrypt all traffic between client and server, protecting authentication traffic from packet capture and analysis attacks.

Summary

All of these technologies can play an important role in network configuration management by providing a configuration management solution with access to real-time event notification and device configurations and secure access to devices. However, each of these technologies and protocols present specific operational and security concerns that you must be aware of if they're in use on your network.

In the next chapter, I'll describe some of the tools that can play a role in effective network configuration management. These tools can range from change request-tracking software to documentation and resolution-management utilities. I'll provide checklists for the most important features in each tool category, enabling you to evaluate tools for use in your own environment.