

Services from a SecurityPoint of View

Contributed by LE Webmaster
Wednesday, 26 January 2005

A network intruder will look for security weaknesses at every point in your network architecture. If you have adequately locked down the Physical, Data Link, Network, and Transport layers of your network, the wily hacker will simply move up to those protocols and services your network does expose to the Internet. These application-specific protocols are actually much easier to exploit, so many hackers start there and drop down to the Network or Transport level when they need to circumvent a protocol's security mechanisms. In this article, we'll go over each of the most commonly used Internet services, briefly examining each for their weaknesses and abuse potential. First, however, we'll discuss sockets and services in general, identifying typical service vulnerabilities so you can identify potential problems when you need to install services on your own network. In this article, we'll go over each of the most commonly used Internet services, briefly examining each for their weaknesses and abuse potential. First, however, we'll discuss sockets and services in general, identifying typical service vulnerabilities so you can identify potential problems when you need to install services on your own network.

Evaluating Socket-Based Services Which services are safe to allow through your firewall, which are not safe, and which ones do you do need to keep an eye on? When a new service becomes popular, or when you want to give your network clients a new Internet-based tool, what do you look for when you evaluate the service?

How Complex Is the Service? Complex services are easier to exploit than simple services. The Echo service, for example, simply transmits back to the client whatever the client sends to it. The Echo service is useful for debugging and for network connectivity testing, but it is difficult to see how the Echo service could be exploited to gain control of the computer running the service. Since the Echo service accepts data from the client, however, it must be programmed to correctly handle being fed too much data at once. The mail service, on the other hand, is a large, complex piece of software that accepts data (mail) from and returns data to the client, as well as reads and stores data and configuration information on the computer's hard drive. Many mail services (POP and IMAP, for example) require authentication before the client can use the service. SMTP, on the other hand, allows any connecting user to send mail as though it came from any user even a non-existent one. If the authentication mechanism can be subverted, the passwords hacked, or the service tricked into sending out private data (such as your password file), the hacker can use the service to get enough information to break into your computer through other means, such as FTP or Telnet.

How Might the Service Be Abused? Some services might be simple and innocuous in themselves, but can be turned to unexpected and detrimental uses. Chargen, for example, is a simple Unix service that sends out ASCII characters over and over. Chargen is a useful network programming and testing tool, because there are certain classes of networking problems that become evident when you can look at a stream of data spanning a whole range of binary representations. A communications channel that clears (or sets) the top two bits of every data word, for example, becomes obvious because the pattern of characters from Chargen will change as well. An unscrupulous hacker, however, might exploit this protocol by forging a SYN packet (connection request) that redirects the output of Chargen to another computer and port. This way the hacker can flood the target computer with data that doesn't even originate from his own computer!

What Information Does the Service Dispense? Some services may be simple, terse, and still dangerous to your network security. Finger, for example, was designed to help Unix users contact each other. A Finger request will tell you whatever there is an account for an individual on a computer, what that account name is, when the user last logged on, additional contact information for the user, and whatever else that user would like to tell the world. That information is certainly useful if you need to know a coworker's e-mail address or phone extension. It is also incredibly useful for a hacker who wants to verify the existence of an account, find a dormant one, or get clues to the identity and personality of an account holder. You won't find many computers on the modern Internet that

support the Finger protocol. The Whois service is another one that you might not want to run on your network due to the amount of information it can give to a network intruder. Services such as Whois and Finger are excellent timesaving tools for use within an organization, but you should never allow access to these services from beyond your internal network or your intranet.

How Much of a Dialog Does the Service Allow? A simple service with a regular interface is easier to secure than a complex service that uses an extensive dialogue with the client to provide its functionality. HTTP, for example (disregarding CGI, server applets, and Active Server Pages for the moment), is easy to secure because all the client can do is ask for a resource, and the server does not maintain data about the state of the connection between client requests (i.e. the protocol is stateless). A stateful protocol is more difficult to secure, especially if the protocol requires client authentication at the beginning of the session and allows for many requests, replies, and state changes after authentication. A stateful protocol must be tested for security holes for every state the client may place the server in. It is possible, for example, to hijack a Telnet session after an authorized client has established the connection and provided correct credentials. Also, the more dialog a service allows, the more dangerous the service is when compromised. If a hacker arranges the Finger service to run at the wrong permissions level (such as root or Administrator), the hacker would still just get account and contact information from it. An FTP session at the supervisor level, however, could send the hacker any file in the computer. A root Telnet session would allow the intruder to do anything at all, including running programs, shutting down or starting services, replacing operating system code, as well as creating and deleting accounts.

How Programmable or Configurable is the Service? The more configurable a service, the easier it is to get the configuration wrong. The more programmable the service, the more likely bugs are to pop up, creating loopholes for network intruders to jump through. So, Exchange Server (which has more configuration options than you can shake a stick at) and Internet Information Server (or any other web server that allows you to run arbitrary scripts to generate web pages) are more likely to contain security weaknesses than simpler services.

What Sort of Authentication Does the Service Use? Any service that requires authentication from the client is a potential security risk for two reasons: the authentication protocol may be weak, and users tend to use the same account names and passwords across multiple services. POP is one example of weak authentication. The username and password are sent from the client to the server unencrypted, and the password is sent complete and unsalted. In POP, the server asks for the username and password, and the client just sends them. Compare this to MAPI (used by Microsoft Exchange), which uses a challenge-and-response protocol. With MAPI, the server requests the username and password, but also sends a value for the client to salt (prepend to) the password before the client hashes (scrambles) the password. The username and hashed password are then sent to the server. The server can compare the hash to a known hashed password to determine if the client should have access to the service. No eavesdropping computer can determine what the password is from the hash, and the same hash cannot be used more than once because the server changes the hash every time. Password hashing works by storing an encrypted version of a password rather than storing the password itself. The encryption algorithm is similar to a public-key protocol in that it can only be used to encrypt the password, not to decrypt it. Retrieving a stored password then doesn't reveal the password because the hash cannot be reversed. Challenge-response protocols are used to ensure that a hacker can't use a password hash. In a challenge-response protocol, a random number is transmitted by the server to the client. The client then encrypts the random number using the password hash and transmits the result back to the server, which uses its copy of the hash to decrypt the random number. If the decrypted random number matches the original random number, then the client has proven that it knows the hash and the server can trust it. On the wire, only a random number and permuted random number have been transmitted, both of which are worthless and cannot be reused. The purpose of a challenge-response protocol is to keep the hacker from intercepting the passwords as they travel from the client to the server. If the hacker can't intercept the password, he may just try to guess it. This is another area in which many protocols fail. A

properly implemented protocol will detect an unusual number (three or greater) of failed password attempts, after which it will not allow any more logon attempts to that username or from that client. A weak protocol will allow as many attempts as the hacker can perform, and a clever hacker can write a program to perform hundreds of attempts per second, determining the true password by brute force. Windows will by default lock out any account but the Administrator account when there are too many failed password attempts. It is easier to remember one password than a half-dozen, so many computer users use exactly the same password for all of their computer accounts. This means that if a network intruder penetrates one computer and captures the password list for that computer, one or more passwords and usernames from that computer are very likely to also work in another computer that is otherwise secure. Your password-protected service gives the hacker a double resource—if the hacker can find a password that works on that service, she'll try it elsewhere, and if she can find a password elsewhere, she'll try it on that service as well.

Your Network Profile

One thing that you should realize as you consider which services you will expose to the Internet is that the more services you choose to expose, the less secure your network will be. Each protocol you decide to allow may have a low probability of being compromised, but those probabilities are additive and it only takes one hole to negate all your security efforts. So remember, less is more in this case, fewer services exposed is more security for your network. In addition, hackers use the services you expose to profile your servers and they use that information to select which attacks to run in an attempt to penetrate your network. If you expose all of the ports that a default Windows server exposes, then the hacker is much more likely to pull out his Windows hacking scripts rather than his Linux hacking scripts, for example.

Common Internet Services

The following services (and their Port and Protocol types) are covered: DNS (53 UDP) FTP (20 and 21 TCP) HTTP (80 TCP) IMAP (143 TCP) NTP (123 UDP) POP (110 TCP) SMTP (25 TCP) HTTPS (443 TCP) DNS (53 UDP)

While this protocol is a prime target for network intruders, you can't disable it without disabling all your network clients as well. You should protect your DNS servers by blocking zone transfer packets or by using a DNS proxy service. The following bulleted items summarize this protocol's security profile (we'll provide a similar bulleted list for each protocol we discuss):

- Complexity-Complex
- Abuse Potential-High
- Information Sensitivity-Minimal
- Dialog-Minimal
- Programmability and Configurability-High

File Transfer Protocol (20 and 21 TCP)

FTP is a useful command-line protocol for transferring files over the Internet. FTP is often used to remotely update web content on HTTP servers. For this reason, among others, it may be necessary to allow FTP traffic through your firewall. FTP's development predates the development of firewalls and it is therefore a little more difficult to pass through a firewall than newer protocols such as HTTP. When a client opens a connection with the service (using port 21, the command channel), the server opens a second connection to the client (using port 20, the data channel). If the firewall is using IP translation to hide the client computers behind a single public IP address, the data channel connection attempt will fail unless special measures are taken in the firewall to identify and pass through the incoming data channel. You should be exceedingly careful in configuring FTP security because FTP establishes a dialog with the client in which the client can browse files on the FTP server and download them, and because FTP authentication is made using operating system usernames and passwords. Even if someone accesses the FTP server as the "anonymous user," the user can gain access to critical operating system files if you have set up file and directory security incorrectly, (especially if you have established symbolic links that allow the anonymous user out of the typical safety sandbox). When you set up an FTP server for access that is external to your network, do not use the same account names and passwords on the FTP server as are used for LAN logon. Here's a rundown of the security characteristics of FTP:

- Complexity-Complex
- Abuse Potential-High
- Information Sensitivity-Medium
- Dialog-Complex
- Programmability and Configurability-High

HTTP (80 TCP)

The Web uses the HTTP protocol to transfer text, video, sound, and even programs over the Internet. Initially, web servers were very simple

(merely sending out to a client whatever page the client requested), but the exploding World Wide Web demands more and more features from web servers. Now a web server is a complex piece of software, with many configuration options, a complicated dialog, and infinite programmability. The hacker exploitation of HTTP can go both ways a hacker may try to exploit your website using HTTP, and a hacker website may contain dangerous web page components such as malicious ActiveX controls or Java applets. A client computer on your network, can do absolutely anything any other program on that computer can do. You should require that on your network only those ActiveX controls that have been digitally signed by organizations you trust will be downloaded. You can use the Internet Explorer Administration Kit to lock down this Internet Explorer setting. If you can get away with it, disable ActiveX controls entirely. Java is a little safer. Make sure that all of the computers in your network are configured not to allow Java applets access to hardware resources unless they are digitally signed by organizations you trust. On the server side, be extremely careful with remote web administration software. Most of the website hacking done by Internet vandals has been accomplished by exploiting security holes in remote website management tools. Scrutinize server-side applets and CGI scripts. Do not make script directories browsable. Do not allow arbitrary scripts to be uploaded. Do not allow scripts to be executed from directories that contain other web data. If you can, maintain web page usernames and passwords separately from operating system usernames and passwords. Log web access, and look for unusual patterns (excessive 404 errors, etc.).

Security characteristics of HTTP:-

Complexity-Complex- Abuse Potential-High- Information Sensitivity-Medium- Dialog-High- Programmability and Configurability-High

IMAP (143 TCP) This is the protocol used by network clients to retrieve mail from servers that are configured to retain-mail on the server rather than transfer it to the client. The protocol itself, while more complex than POP, is slightly more secure (passwords aren't sent in the clear, at least).

Security characteristics of IMAP:-

Complexity-Simple- Abuse Potential-Medium- Information Sensitivity-Medium- Dialog-Low- Programmability and Configurability-Low

NTP (123 UDP) This is the protocol used by network devices, including firewalls, to reliably update and synchronize their time and date settings. Many implementations of NTP have proven vulnerable to buffer overrun exploit that allows remote root access. This is the perfect example of why even the simplest services should be inspected for security regularly.

Security characteristics of NTP:-

Complexity-Simple- Abuse Potential-Medium- Information Sensitivity-Medium- Dialog-Low- Programmability and Configurability-Low

POP3 (110 TCP) The Post Office Protocol allows clients to check their e-mail over the LAN or over the Internet. POP is easy to configure and use, but the protocol is a little too simple it doesn't encrypt usernames or passwords. Avoid allowing access to internal mail accounts from outside the firewall using POP and if you do, do not allow POP account names and passwords to be the same as LAN usernames and passwords. Consider using IMAP instead.

Security characteristics of POP3:-

Complexity-Simple- Abuse Potential-Medium- Information Sensitivity-Medium- Dialog-Minimal- Programmability and Configurability-Low

SMTP (25 TCP) Most of the mail exchanged over the Internet is done using the Simple Mail Transport Protocol (SMTP). This protocol simply accepts mail in a simple dialog (without checking the authority or even the identity of the sender). Although the protocol is simple, the software that processes the mail (once it's received) is often not so simple. Many SMTP packages have complex configuration options and forwarding rules, and, if incorrectly configured, these can adversely affect network performance or crash the mail server when large amounts of mail are being processed. Also, the lack of sender authorization leaves SMTP open to spam attacks and e-mail flooding. Unfortunately, if you want to receive Internet mail you need to support SMTP. You should choose mail server software that is as bulletproof as possible and use care when configuring it, paying attention to details like available hard disk space, network bandwidth, and so on. Install a server-based virus scanner to sanitize e-mail attachments as well.

Security characteristics of the SMTP protocol:- Complexity-Complex- Abuse

Potential-High- Information Sensitivity-Medium- Dialog-Minimal- Programmability and Configurability-HighHTTPS (443 TCP)HTTP transfers (on port 80) traverse the Internet unencrypted, so web traffic using it should not be trusted. Internet commerce requires trust, however, as do private communications and secure Intranet use. Fortunately, the Secure Socket Layer provides a way for HTTP traffic to be encrypted between the client and the server. Other protocols may use SSL as well to encrypt their session data, but the network clients and servers must be written to allow this option. SSH is one other such service, as is SFTP (FTP over SSL). Security characteristics of HTTPS:- Complexity-Simple- Abuse Potential-Low- Information Sensitivity-Medium- Dialog-Minimal- Programmability and Configurability-LowOther Common Services:DNS is one service you must allow through your firewall in one manner or another, because without it your network clients won't be able to find anything. There are many more services you may want to support on your network, or that you may elect to block depending on the needs of your network users. Each has its strengths and vulnerabilities, Note that nearly all services are available for all platforms through the use of third-party applications. We've divided the services based on the standard configurations of these operating systems.

Standard Unix Services:SSH (22 TCP)Chargen (19 UDP and TCP)Daytime (13 UDP)Discard (9 UDP and TCP)Echo (7 UDP)Finger (79 TCP)NFS (2049 TCP and UDP)Quote (17 UDP)RPC (Unix) (111 UDP)RSH (514 TCP)Platform Neutral Services:Telnet (23 TCP)TFTP (69 UDP)BootP/DHCP (67 and 68 UDP)LDAP (389 TCP and UDP)SNMP (161 UDP)VNC (5800+, 5900+ TCP)Standard Unix ServicesUnix has a bevy of simple legacy services from early in the development of the Internet—when security was less of an issue and it was useful to have a service that cycled through the character set, so you could see if one of the routers between your computer and that host was chopping off the top bit of all the bytes in the network packet and thereby garbling your e-mail. Since then, the world has standardized on eight-bit bytes and hackers have learned to exploit unchecked buffer copies; so while the majority of these protocols are harmless, it makes no sense to leave them exposed at the firewall.

Chargen (19 UDP and TCP)Chargen continuously sends out the printable ASCII characters. It is useful for testing network applications. Any service that could be stopped or swamped by a stream of ASCII characters is broken anyway and shouldn't be let past your firewall. It is extremely unlikely that a network intruder could use Chargen to break into your system.

Daytime (13 UDP)This service sends the date and time at the server to the client. It would take a very clever hacker to find a security weakness in this protocol.

Discard (9 UDP and TCP)This protocol throws away any data sent to it. It is useful for developing network tools. It is as secure as it is useless.

Echo (7 UDP)Echo repeats to the connected client whatever the connected client sends to it. It is useful for testing network applications. It is extremely unlikely that a network intruder could use Echo to break into your system, as long as Echo properly manages its input buffers.

Finger (79 TCP)The Finger service was designed to help network users communicate by providing system information, such as the last time a user checked their e-mail, and real-world data, such as the user's office hours, telephone number, or current projects. Unfortunately, this data is as good as gold to hackers looking for potential account names and passwords. Also, some hackers will even go so far as to call an office pretending to be the help desk staff and trick users into giving up their account names and passwords over the phone.

NFS (2049 TCP and UDP)NFS is the Unix equivalent of NetBIOS; it gives LAN clients access to file server storage. If you need to allow remote clients access to NFS resources, establish an encrypted tunnel to do it, don't just open up the NFS ports. Improperly configured NFS servers can be easily exploited to gain root access, and it's easy to spoof NFS legitimate connections.

Quote (17 UDP)This protocol merely sends any connecting client a random selection from a file full of quotes. Quote provides little leverage for abuse.

RPC (Unix) (111 UDP)Remote Procedure Call is a protocol that allows two computers to coordinate executing software. A program on one computer can use RPC to transfer the execution of a subroutine to another

computer, and have the result returned via RPC to the first. RPC is a fragile service, and most operating systems cannot handle arbitrary data being sent to an RPC port. RPC is best used in trusted LAN environments, and you should not let RPC traffic through your firewall. RSH (514 TCP)

The Remote Shell protocol makes up for deficiencies in Telnet. There are always dangers when you allow remote command-line access to computers through your firewall, and RSH is better than Telnet because RSH at least protects the passwords as they are exchanged between the client and the server, but SSH is better than either because it encrypts the session data as well. Use SSH for remote console access and block Telnet and RSH. SSH (22

TCP) The Secure Shell protocol makes up for deficiencies in RSH and Telnet. SSH uses SSL encryption to securely authenticate communications and encrypt session data. You can configure it to allow usernames and pass-phrases or cryptographic key exchange (or both) to authenticate the user, public keys to specify authorized client computers, and a range of algorithms to encrypt the session data with. If you are going to allow any kind of remote control through your firewall this is a good choice. Platform Neutral

Services There are some services that you may find running on both Windows computers and Unix computers because their universal utility. Not all of them must be exposed to the Internet, however, because many of them are primarily used in a LAN environment; DHCP is one such example. Others—Gopher, for example—have been superceded by later protocols. Some others are current and often used in an Internet setting but must be treated with care; LDAP is one such.

Telnet (23 TCP) Telnet is extremely simple it's just a connection opened to a command line interpreter. Whatever you type is sent to the interpreter on the server, and the interpreter's response is returned to you. The data traffic is not encrypted, and when you log on, the username and password are readable by any computer on any intermediate LAN. Do not allow Telnet access to computers inside your firewall. If you require command-line access, use a more secure protocol such as SSH. TFTP (69

UDP) TFTP is used with BootP and DHCP to allow diskless workstations to load their operating system and other configuration over the LAN. TFTP does not have the two-channel problem that FTP has (and therefore it inter operates well with a firewall), but there is little reason to allow TFTP through a firewall when you already have FTP and HTTP for file distribution. Also, hackers have developed tools for using unprotected TFTP servers as pirated software dumping grounds, so you should use a more secure file transfer protocol through your firewall. BootP/DHCP (67 and 68 UDP)

BootP was developed as a simple mechanism for allowing simple network terminals to load their operating system from a server over the LAN. Over time it has expanded to provide for centralized control of many aspects of a computer's identity and behavior on the network, including allocating IP addresses, configuring gateway, DNS, and router settings, dispensing NetBIOS names, as well as downloading operating system files. The greatest danger from BootP and DHCP is from a network intruder impersonating a DHCP server on your network and there by misconfiguring the DHCP clients. As long as you do not allow DHCP to pass your firewall, you should be able to use DHCP internally without

problems. LDAP (389 TCP and UDP) The Lightweight Directory Access Protocol is a flexible and distributed way of maintaining contact information (including usernames and passwords) over the Internet. Several Internet services use LDAP to maintain user information rather than relying on the operating system user accounts. This is more secure because it separates operating system functionality from service functionality, and a hacker who gets a service password will not necessarily be able to log on to the server with it.

If you want to maintain contact information on your network to facilitate communication with people in your organization, you should consider using LDAP instead of Finger and Whois. SNMP (161 UDP)

The Simple Network Management Protocol is a useful tool for remotely managing network devices such as routers, servers, hubs, clients, and terminal servers. You can use it to enable and disable ports, measure bandwidth, reboot devices, and gather statistics. However, it should be used to manage your network only, not to allow hackers to watch every aspect of the data flow on your network. Block SNMP traffic at your firewall VNC (5800, 5900

VNCTCP) Virtual Network Computing (VNC) allows for remote control of computers' desktops, much like Terminal Services does for Windows. Unlike

Terminal Services however, VNC is cross-platform clients exist for Windows, Unix, and many other operating systems besides. While the authentication between the VNC server and the client is encrypted the session information (keystrokes and window contents exchanged) are not. If you require remote control of computers inside your network from the outside, consider requiring other security mechanisms as well such as a VPN tunnel between the client and the server network. Last words Security is not a static thing, it's a continually evolving process. You can't just plug in a firewall and expect it to solve your security problem forever. Attacks change, methods become obsolete, and so do firewalls. To obtain true security, you have to maintain constant vigilance