Fortified SSH: A Cost-Effective Way to Safeguard Your Network

Contents

Introduction

The threats to enterprise systems and data continue to escalate at an alarming rate. It's still relatively easy for hackers to break into corporate or government networks and cause irreversible damage. The risks aren't limited to data exposure, but also include threats to data integrity, access, and authentication.

One of the products at the forefront in the war to keep data secure is a small, robust, easy-to-configure software solution called Secure Shell, or SSH. SSH is a set of tools based on the SSH protocol. The main purpose of SSH is to securely transmit data over network connections using strong encryption and authentication methods. SSH is a replacement for nonsecure Telnet, FTP, X11, and Berkeley r-commands (rlogin, rcp, and rsh)—all of which transmit data in the clear.

SSH, in its many implementations, is currently estimated to have more than two million users in more than 80 countries. Increasingly, organizations are turning to SSH because it:

- Provides a secure client-server protocol that encrypts data during transmission over a network.
- Offers strong authentication methods to ensure that the client and server are communicating with trusted hosts.
- Prevents root access, which is typical of nonsecure network applications such as Telnet and FTP.
- Is transparent to end users.
- Includes, in commercial versions, high levels of technical support, maintenance, and attention to platform vulnerabilities. (Free, non-commercial implementations of SSH don't provide the same level of support as commercial versions.)

SSH is not a foolproof solution to all network-related security problems. But it does eliminate today's greatest security threats. This white paper provides an in-depth look at SSH—the features that give it such power, how it counters attacks, and when it doesn't work. There are real-world case studies and a section that compares commercial to non-commercial versions. By the time you're done reading, you'll know why organizations worldwide are switching to SSH.

Two Versions of SSH

Tatu Ylönen developed SSH1 in 1995. His initial impetus was to address the nonsecure nature of all traveling or stored data—passwords, e-mails, protocol interfaces, and more—given the relative ease of surreptitiously capturing data off the network.

After finding several limitations in the protocol, Ylönen went on to develop SSH2 in 1996. SSH1 and SSH2 are based on distinctly different protocols, and the two are not compatible. The Internet Engineering Task Force (IETF) is currently working to standardize the SSH2 protocol in order to guide its future development.

SSH1 and SSH2 share the following features:

- Client programs that perform remote logins, remote command execution, and secure file copying across a network.
- An extremely configurable SSH server.
- Several selectable encryption algorithms and authentication mechanisms.
- An SSH agent to cache keys for easy access.

SSH2 adds a number of new features to provide a stronger, more comprehensive product. These features include:

- Encryption ciphers, such as 3DES and AES.
- The use of sound cryptographic Message Authentication Code (MAC) algorithms for integrity checking.
- Support for public key certificates.

SSH1 and its numerous implementations are still deployed on the Internet today. This situation is rapidly changing, however, as SSH2 is now employed and certifiable under the FIPS 140-1 and 140-2 NIST/U.S. government cryptographic standards. Commercial entities are also switching over to SSH2 in large numbers to take advantage of its security enhancements and technical support.

SSH Features

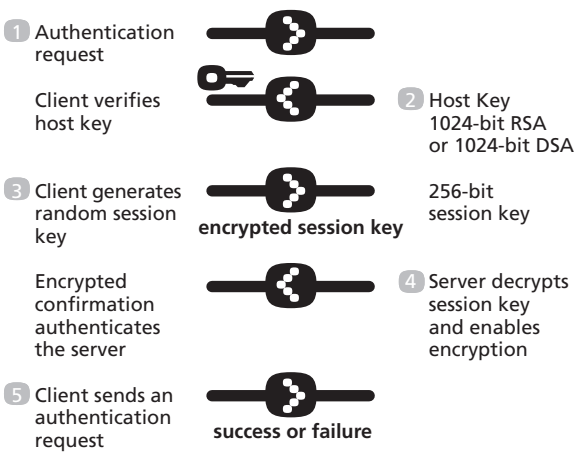
The features that make SSH such a powerful countermeasure in the fight to keep networks secure are listed below. These features are standard, unless otherwise noted, with SSH1, SSH2, OpenSSH, and commercial products such as Reflection® for Secure IT.

Secure remote logins

Both Telnet and the common Berkeley r-commands (rsh, rlogin, and rcp) transmit usernames and password data in plaintext over the network, making them easy prey for third-party interceptions. Telnet sessions are also readable if someone is sniffing the line.

SSH is a substitute for both Telnet and rlogin. By running the SSH program instead of Telnet, you authenticate yourself to the remote computer's SSH server through an encrypted connection. The entire session is secure and the user won't be able to tell the difference because the encryption is transparent.

SSH CONNECTION PROTOCOL



Secure file transfers

File Transfer Protocol (FTP) is used for file transfers in mixed environments. As a standard component of every operating system, FTP is the tool most commonly used to transmit files larger than 5 MB that do not fit into e-mail. The advantages of FTP are that it works quickly and costs nothing.

But FTP is problematic because it doesn't encrypt data sent from one computer to another over the Internet. As a result, packets can be easily intercepted en route and read. Furthermore, passwords need to be programmed into scripts for automated sessions.

By comparison, SSH has some clear advantages. With SSH, you can easily and securely automate file transfers. SSH encrypts files prior to transmission and then decrypts them after they arrive at their final destination. In addition to providing ironclad user authentication, SSH doesn't require scripted passwords.

Secure remote administration

Telnet is a widely used protocol for the remote access and administration of UNIX servers. Like FTP, it comes with every UNIX box. The benefits of Telnet are that it's free, easy to use, and requires zero configuration. But its limitations are significant. It offers no security against eavesdropping or password theft, and provides only plaintext password authentication.

Again, SSH comes out on top. It offers secure remote access and administration of UNIX servers, including various user authentication methods. And when dealing with graphical X11 Window System applications, SSH automatically sends network traffic into a secure tunnel.

Secure remote-command execution

System administrators frequently need to run the same command on multiple computers. The rsh command can accomplish this, but like its other r-command brethren, it transmits commands in plaintext. SSH, on the other hand, encrypts all commands for travel across the network.

Keys and agents

Typically, if a user has accounts on several computers within a network, each account must have a unique password. Users are required to memorize and retype passwords—a tedious and risky process.

SSH's public-key authentication feature eliminates the need to memorize and retype. A key is a set of bits that allows an SSH client to say, "I am really me." Using the SSH program's ssh-keygen command, a user can create a key set. The private key, which must be kept secret, represents the user's identity for outgoing SSH connections. The public key symbolizes the user's identity for incoming connections to his or her account. Keys, along with SSH's authentication-agent program, make life easier for both users and administrators.

Access control

SSH allows users to grant other users access to their network accounts. In this way, specific functions can be performed without passwords or system-administrator privileges.

Port-forwarding

Port-forwarding, or tunneling, allows inherently nonsecure TCP/IP traffic to be forwarded through an SSH connection. It offers an easy, cost-effective way to protect POP3, SMTP, and HTTP connections, as well as FTP, news service, and other TCP-based services. Any TCP-based application can also be secured through an SSH tunnel. As a result, the applications gain privacy protection, integrity checking, authentication, and authorization.

SSH Protections

In this section, we'll look at some possible methods a hacker might use to gain access to your data in transit. We'll also describe how SSH counters these attacks.

Eavesdropping

By using packet-sniffer software, eavesdroppers can read unencrypted network traffic without the sender or receiver knowing it. Sniffers can collect unencrypted passwords, usernames, and any other data transferred via the network. Even if a password is encrypted, eavesdroppers can capture personal information simply by watching the unencrypted traffic that's generated after the password is entered.

SSH protects against eavesdropping by encrypting all data, making it unreadable to potential eavesdroppers.

DNS and IP spoofing

Domain Name System spoofing is what happens when an attacker tricks a DNS server into trusting a host that it shouldn't. This allows someone to access your site's e-mail or direct users to the wrong web pages. According to Information Security magazine, one in every three organizations with an Internet presence is vulnerable to DNS spoofing.

IP spoofing is a technique that grants access to a computer by sending a message with the IP address of a trusted host. This takes a little work on the attacker's end, but it is an effective hacking method.

SSH wards off such attacks by cryptographically verifying the identity of the server. For every session, the SSH client validates the server's host key against a local list of available keys that are associated with server names and addresses. If the keys do not match, then an immediate warning is issued.

Connection hijacking

When an attacker disconnects users from their TCP connections, it's called connection hijacking or the spoofing of TCP packets. Attackers must be "active"—meaning that they can listen to network traffic as well as inject their own traffic into the transmission—in order to perform a connection hijacking.

When hijacking a connection, an attacker sits on the network, sniffing for packets transmitted from the client to the server. The attacker then obtains the hosts' IP addresses and relative port numbers, which allow him to spoof TCP/IP packets. This kind of attack can be devastating, regardless of how strong the authentication method is.

SSH is unable to prevent hijackings because of an inherent flaw in the TCP layer. SSH can, however, render the hijacking ineffective through its integrity-checking process. If a session is modified during transmission, SSH will shut down the connection immediately—without using the nefarious data. SSH2 uses the cryptographically strong hash functions MD5 and SHA-1 for integrity checking.

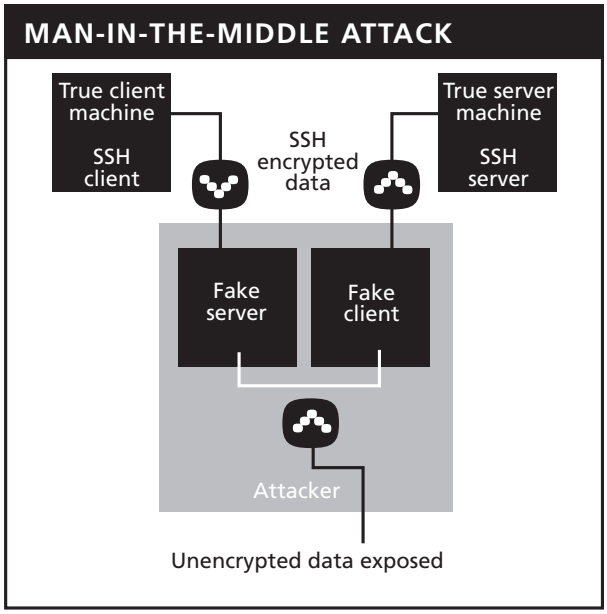
Man-in-the-middle attacks

A man-in-the-middle attack occurs when a third party poses as both ends of a communication. During a session, an attacker will sit between the SSH client and the server and convince each of the two hosts that he is the other host.

After a client has been authenticated and given access by a server, the attacker can learn which port and sequence numbers are being used to transmit data. With this information, the attacker can intercept traffic and read, capture, or delete data. The hacker shares the session key with the legitimate user, fooling both the client and the server into thinking they are connected to one another.

There are two ways SSH can protect against man-in-the-middle attacks:

- The first defense is SSH's server-host authentication. Because the attacker does not have the server's private host key, he would have to break into the server host to pull off the impersonation. For this protection to be totally effective, the client must check the server-supplied public host key against its list of known hosts.
- The second defense SSH provides is stronger authentication for the client. Passwords are vulnerable, but public keys and certificates are essentially immune to these types of attacks.



Insertion attacks

In an insertion attack, also known as a replay attack, a hacker identifies a packet containing a specific command and then resends the packet at another time during the session in order to replay the command. Like TCP connection hijacking, insertion attacks introduce altered data into the network stream. But in the case of insertion attacks, the insertion is encrypted and sent to either the client or server to decrypt.

SSH2 uses cryptographically strong integrity checks, namely SHA-1 and MD5, to repel insertion attacks.

When SSH Has No Defense

SSH has no defense once an attacker has gained root access to a machine. At that point, the attacker can destabilize SSH from within and render security nonexistent. The following examples illustrate the types of attacks and situations that can undermine your network's security, despite the presence of SSH.

Password cracking

Although SSH encrypts passwords while they're transmitted over the network, a password is an authentication type that's easily compromised. For example, an attacker can launch a brute-force "dictionary attack"—using a program that employs a list of dictionary words—in an effort to log on as an administrator.

The obvious counter to this kind of attack is to come up with a difficult password. It's generally recommended that users create passwords containing a combination of upper/lowercase letters and numbers that only the administrator knows. With SSH, servers can also be configured to disallow the use of passwords; public keys can be used instead, so weak passwords are not a problem.

IP and TCP attacks

SSH works on top of the transmission-control protocol and the Internet protocol, which makes it vulnerable to attacks against the weaknesses inherent in both. The attacks occur at the network level, so SSH can't protect against them. They include denial of service and routing of data to locations other than those intended. Lower-level techniques, such as IPSec, can thwart attacks at the TCP/IP level.

Traffic analysis

Even though SSH prevents snoopers from reading network traffic, there is still quite a bit of information an attacker can garner through simple observation. The amount of traffic, the destination, and the timing can all be used by a third party to identify transactions, backup schedules, or the best time of day to launch a denial-of-service attack. SSH traffic is easy to spot and is customarily destined to well-known port 22.

One way to mitigate this risk is to produce a constant stream of traffic, whether the network is active or not. This prevents prying eyes from being able to pick out the true traffic.

Covert channels

Covert channels transmit data in an unanticipated and unnoticed manner. For example, employees whose e-mail use is restricted might communicate with one another by including messages in their home directories. A system administrator wouldn't expect this sort of communication and would have a difficult time eliminating it.

SSH does nothing to counter covert channels. It can, however, be used as a covert channel itself—with even/odd packet lengths simulating the dots and dashes of Morse code.

Five Real-World Case Studies

How well does SSH fit into a responsible, comprehensive program of enterprise security? Information security technologies (InfoSec) have traditionally been identified with the preservation of three elements: confidentiality, integrity, and availability. Failure to maintain any of these three elements has economic, political, and social consequences.

Because information has value, it is an attractive target for criminals. Based on InfoSec principles, computers are designed, implemented, and operated to ensure that only authorized persons can access confidential data, modify or destroy information assets, or deny access to information when necessary. But the traditional InfoSec design may not be enough to counter today's computer threats.

In order to reduce loss and meet a standard of due care, InfoSec practitioners must now be concerned with the additional elements of authenticity, possession, and utility. The first three case studies illustrate SSH in action, supporting InfoSec goals. The last two case studies show what happened when SSH wasn't used. All are based on actual customer experiences.

Case 1: Remote access and administration

First we look at a mid-sized corporation that has five offices with Internet connections, leased lines, and a mixed environment of UNIX and Windows servers supporting a small group of administrators and a larger group of users. As one might expect, the corporate servers need periodic maintenance and cleanup.

Without SSH, the administrators manage the hosts by using nonsecure UNIX utilities (rlogin, rcp, and FTP). These utilities are not secure because they are easily hacked and permit denial-of-service attacks that defeat confidentiality, integrity, and authentication. The Windows server, on the other hand, is managed using nonsecure access to the Windows Terminal Server functionality. The corporation is using a leased line for each office so that passwords and data do not have to travel unprotected.

By using SSH, the corporation can securely manage and access its UNIX and Windows servers, transfer files, and decrease the number of leased lines required since traffic is able to travel securely and unexposed, even over the Internet. In addition, the firewalls between the offices are easy to configure and can serve as separate authentication hosts for users. This method saves a significant amount of time and money. More importantly, SSH protects the confidentiality, integrity, and availability of information from unauthorized disclosure, modification, destruction, or use. In most cases, this reduces legal liability and loss.

Case 2: Secure transfer of financial data

Next we consider the financial community and its need for secure data movement. Banks perform transactions and maintain many critical databases that need to be synchronized after the close of each business day. They initiate batch transfers (many of which are performed on a random 24-hour timetable) to ensure data integrity and confidentiality. This must be done without error or interference, and within a robust, authenticated environment.

Without SSH, each bank server needs its own network encryption devices. There may be quite a number of these to manage—and connections remain nonsecure on an end-to-end basis. Think of the disadvantages of naked network security. If data encryption is done with external network encryption devices on both ends of the connection, the data lines don't need to be secure, just reliable. Furthermore, both ends of the connection from the server to the encryption device are nonsecure, making them easy targets for eavesdropping.

Another problem is that during data transfers, scripts need to have passwords coded into them to authenticate to the remote server. This makes it difficult to hide the connection passwords and even more difficult to change the passwords periodically—which must be done to maintain security. Lastly, connection passwords may be known to several people. That means they are not personal or random enough to guard against a security breach.

One reasonable solution would be to enable end-to-end security with SSH using public key infrastructure (PKI) or security token-based user authentication. With SSH, a batch-transfer connection is secured end to end, with no server-encryption devices or script-connection passwords needed. In addition, SSH provides strong user authentication as well as support for enterprise authentication methods such as RSASecurID and PKI. The result is tighter security with much less administration.

Case 3: Business partner communications

Now we study a software company and its required communications. The company develops and markets its software in tandem with several development partners, distributors, and resellers. Research, development, sales, and marketing data are shared with these partners over the Internet during product development and prior to product releases. Through a web site, partners browse and download product material and updates. E-mail is used for mass mailings and to send large quantities of material, such as data sheet drafts or pricing information, while FTP is occasionally used to send CD images.

Due to the proprietary nature of the data being exchanged, a strict security policy is required and network traffic must be encrypted. This means that each partner must have a unique way to communicate with the various virtual private network (VPN) devices deployed by the parties involved. Network managers know that this is not a trivial problem.

Sending data back and forth using FTP is demonstrably nonsecure. In addition, each partner in this case uses a different VPN device with a different type of access (and discordant handshakes) to its internal network. This causes significant network overhead and complicates user-authentication and key-management problems.

With SSH, however, all data transfers can be protected, ensuring that product details are not leaked to the public. Every partner and user can be personally identified because the access tokens are unique. This limits risk and reduces vulnerability. By standardizing with SSH in research and development communication, the potential for problems with partner-access management is reduced dramatically. Access is to individuals instead of to entire networks, which segregates any losses to identifiable units.

Case 4: Compromised server invites password sniffer

A corporation is conducting e-commerce and has several servers in the same network. Because of a misconfigured firewall, one server was directly accessible from the Internet. This server was scanned and compromised by someone using an exploitable defect in its web server. The hacker implanted a RAT (Remote Access Trojan) with a sniffer to collect passwords from all network users connecting to this server.

When the sniffer was discovered it became evident that it had been running for at least a month, collecting virtually all user account-names and passwords. The compromised server was immediately shut down, every user account was locked, and users were forced to change their passwords. The other servers in the network were then searched for any signs of compromise.

The hacker was never caught, and he successfully hacked other companies as well. It is not known whether he worked alone or with others. The compromised server was shut down for a week. Fortunately for the corporation, no other host was hacked. Users lost a few hours of working time to change their passwords, and their ability to work was impaired for a week. This single security breach cost significantly more than it would have cost to implement the proper security measures in the first place.

Case 5: Suspicion of data theft by a competitor

In this example, a company finds that the only cost-effective way to exchange massive amounts of data with development partners in other countries is via the Internet. Although the disparate pieces of data possess some value, the bulk of the value was considered to come later in the project, after product development. Administrators believed that standard communications tools, such as e-mail and FTP, provided sufficient security measures and that no encryption was needed.

During the product-development cycle, it became apparent that the company's direct competitor was hitting the same milestones, only just a bit later. Even though the subject company introduced its innovations first, the fact that its competitor offered similar innovations shortly afterward seemed to have a negative impact on sales. No proof of industrial espionage was found, but it was suspected that one or more former employees or partners may have been involved in the theft of product data. As destructive as insider threats can be, the damage caused by a combined insider-outsider team has the potential to be even worse. The subject company was probably the target of this type of coordinated attack.

In the end, no one was sure if the data had been stolen or not. It was clear that the company had left itself vulnerable to industrial espionage and that the product-development theft had the potential to negatively affect the company's stock price. The corporation took measures to protect its data against all types of espionage—whether by outside parties or current or former employees.

Weighing the Costs of SSH

According to conventional wisdom, implementing security measures to save money is illogical since there is little or no additional revenue to be gained. This theory has proven to be false, as well as risky, in practice.

Generally, the biggest IT expense is personnel costs, but a serious breach of data security at the network level means time lost throughout the organization as a whole. The areas of IT, legal, risk, public relations, and management, among others, all suffer.

The following hypothetical example of the costs to set up a basic service (like a web site or database) should help clarify the issues surrounding any system that processes, stores, or exchanges valuable business data.

- Low-end servers typically cost around $2,300.
- For software licenses, add another $2,000.
- To set up the server (design and installation), add another $2,000 to $2,500, minimum.
- After installation, maintenance for the service typically averages $200 to $250 per month.

The cost of setup in this example is roughly $10,000, including all hardware, software, and maintenance for the first year.

A commercial SSH server license costs less than 10 percent of the system setup price given here, which is an extremely low-end scenario. But if the system lacks SSH protection and gets hacked, administrators will be forced to redo work, change passwords, and take a number of other reactive measures to secure the system.

At $50 per hour, per administrator, the price of SSH translates to less than 10 hours of administration work to secure the system. And this comparison doesn't take into account the revenue lost while the service is offline and being repaired, or the inevitable costs associated with any resulting loss of reputation.

Not All Options Are Created Equal

SSH has been securing networked connections for nearly 10 years, and there are a number of implementations available. Some versions come with server operating systems or are available as free downloads from the Internet. OpenSSH is an example of the latter. You can also buy third-party, fully supported commercial versions, such as Reflection for Secure IT. This section explores both options.

OpenSSH

The benefits of OpenSSH include a large user base, support for most UNIX platforms, and the fact that it's free. But even with these benefits, OpenSSH comes with significant risks that administrators must consider when choosing an SSH provider:

- Platform selection is limited and lacks native Windows support.
- There is little or no quality control. Several critical updates need to be installed every year, and even that is risky.
- There is no single point of reference for updates.
- There is no reliable support, no responsible party to go to with questions, and always the possibility of a hostile reply in response to the simplest of newsgroup queries.
- Functionality is technically limited. For example, PKI lacks user authentication, which is available with supported SSH products. OpenSSH is behind the curve by more than five major product enhancements and will not catch up with the supported technology.
- There are no liability or patent guarantees, leaving companies vulnerable to lawsuits over copyrights and patents.

Reflection for Secure IT

Introduced in 1996, Reflection for Secure IT has become the choice of many network administrators. Most of Reflection for Secure IT's customers are from the government, high-end banking, insurance, and telecommunications sectors—all of which demand strong network security.

The benefits of Reflection for Secure IT include:

- High-quality software.
- Reduced administrative overhead (when compared with other options).
- Low total cost of ownership.
- Full support and maintenance, including a professional staff to answer questions, provide custom design, and solve problems as needed.
- Timely software updates.
- The opportunity to do business in new ways.

Comparing maintenance costs: Reflection for Secure IT vs. OpenSSH

The table below compares maintenance costs provided by Reflection for Secure IT versus an OpenSSH solution. As you can see, despite the zero maintenance fee nature of OpenSSH, real maintenance costs are significantly lower when using the Reflection for Secure IT option.

This comparison of server maintenance costs assumes the cost of a UNIX administrator to be $50 per hour. So when calculating the cost of maintenance with the OpenSSH solution, we multiplied the maintenance time per server (1.5 hours) by the number of updates (6) by the number of servers (8) by $50 per hour. This gave us the final annual figure of $3,600.

Not included in the OpenSSH maintenance-cost calculation is the fact that every change may break or damage something else. Keeping the maintenance solution simple yields less work and fewer problems down the road.

	Reflection for Secure IT	OpenSSH
NUMBER OF SERVERS	8	8
NUMBER OF UPDATES	2	6
WORKLOAD	1 hour/server 16 man hours = $800	1.5 hours/server 72 man hours = $3,600
SOFTWARE MAINTENANCE FEE	$800	$0
TOTAL COST	$1,600	$3,600
PROS	• Fewer updates • Less work • Better support • Maintenance notifications that do not require a technical degree	• No maintenance fees • “Techies” prefer to support the open-source community
CONS	• Maintenance fees	• More updates • More work • Inferior support • Multiple parties provide technical notifications

How to Choose a Security Solution

For most software, the total cost of ownership has little to do with sticker price and everything to do with the amount of work needed to get the system up, running, and maintained. Factors to look for when choosing a security solution for your network include:

- 24x7 support.
- Training sessions and courses.
- Quality assurance.
- FIPS certification by the U.S. government (NIST).
- Broad platform support.
- The ability to customize solutions.
- Protection from legal liability. (Is your company exposed if you suffer a breach of security?)
- Vulnerability notification.
- Intellectual property rights warranty. (Make sure that you will be using legal software, which can help prevent lawsuits.)

As stated earlier, the trouble with using the options included within operating systems, such as FTP and Telnet, is that they offer little or no security.

SSH freeware options provide no support agreements and updates often lack proper quality-control checks, leaving systems vulnerable to attack. Since there is no seller, the responsibility for software vulnerabilities falls solely on the user. Similarly, most of the SSH implementations that come with operating systems do not offer adequate vendor support, and may have been built from older code with open vulnerabilities.

From UNIX Administration to Multipurpose Security-Protocol Suite

Initially, the SSH protocol suite became the security tool of choice for almost every UNIX server. SSH developers, such as AttachmateWRQ, expanded their platform support to include other platforms, including Windows, and new functionality. This robust security technology is capable of handling a range of issues—not just administration and basic file transfer. Security problems involving wireless connections, PDAs, and specialized security architectures can all be addressed by SSH.

We believe that fortified (continuously improved and supported) commercial versions of SSH help meet the lion's share of today's security goals. In a nutshell, SSH benefits include secure communication, robust authentication, better access control, easier firewall configuration, and access to UNIX and Windows servers with one tool. Commercial versions of SSH—especially FIPS-certified SSH—clearly reduce risk, meet a standard of due care, and achieve the highest security standards.

About AttachmateWRQ

The leader in universal host access and integration—with a combined 40 years of software experience—AttachmateWRQ helps you maximize the value of your existing IT investments as you advance long-term business strategies. More than 40,000 customers, representing over 16 million desktops worldwide, use AttachmateWRQ products and services to extend, manage, and secure their enterprise assets. Learn more at **www.attachmatewrq.com**.

Corporate Headquarters
1500 Dexter Avenue North
Seattle, Washington 98109

TEL 206 217 7500
800 872 2829
FAX 206 217 7515

EMEA Headquarters
The Netherlands
TEL +31 71 368 1100
FAX +31 71 368 1181

Asia Pacific Headquarters
Australia
TEL +61 3 9825 2300
FAX +61 3 9825 2399

Latin America Headquarters
Mexico
TEL +52 55 5658.7755
FAX +52 55 5658.6393

Attachmatewrq.com

E-Mail:
info@attachmatewrq.com

05-0123.0905