

Secure Remote Access with the Solaris™ 9 Operating Environment

A Technical White Paper



Table of Contents

Executive Summary1
Introduction3
Network Threats5
Password Theft	5
Exploiting a Trust Relationship	6
Session Eavesdropping	6
Session Hijacking	6
Weak Authentication	6
Man-In-the-Middle Attacks	6
DNS Spoofing	6
Automated Network Sniffer	6
Insertion Attack	7
Solaris Secure Shell Protection8
Private Network Sessions	8
Strong Authentication	8
Session Integrity	9
Agent Forwarding	10
X-Windows Session Tunneling	10
Easy Integration	10
Similar Replacement Commands For Network Access Commands	10
Compressed Data Streams	11
Solaris Secure Shell Features12
Security Policy Considerations14
Specific Policy Statements	14
Port Forwarding	15
Gateway Ports	15
Authentication Requirements	15
Key Handling	16
Auditing	16
Unsafe Network Services	16
Conclusion17

Chapter 1

Executive Summary

More and more, remote computing is becoming a standard practice in today's networked world. Using remote computing technologies, users are able to gain access to IT resources from home as well as remote locations such as hotels and branch offices. System administrators are able to safely manage and maintain these systems from remote locations. While this can reduce administration costs and increase productivity, it also raises security concerns. Data privacy and user authentication may be compromised in remote computing sessions. Traditional tools such as `telnet`, `rsh`, and `rCP` provide little protection against the myriad types of security threats in remote situations.

The Solaris™ 9 Operating Environment (OE) provides a comprehensive set of tools and technologies to protect the IT environment and reduce the risks associated with remote access and administration. The Solaris™ Secure Shell command shell is a new way to encrypt all network traffic, provide stronger authentication, and monitor the integrity of the network session. It also provides equivalent replacements for familiar commands — `telnet`, `ftp`, `rlogin`, `rsh`, and `rCP`, as well as a tunneling mechanism for X-windows communications and other unsafe network services. Because the Solaris Secure Shell syntax is similar to existing commands, organizations can implement it with little user training. This can significantly lower the overall cost of training and deployment.

The Solaris Secure Shell helps protect against a number of network threats, such as session eavesdropping, password theft or misuse, session hijacking, and others. By offering strong authentication and technology to ensure session integrity, Solaris Secure Shell enables administrators to safely maintain systems in remote locations. The Solaris Secure Shell also provides a

way for users to secure X-Windows sessions over the Internet, allowing teleworkers to securely login and use productivity applications from home or other remote sites.

The Solaris Secure Shell is a powerful and flexible security tool that can be implemented relatively easily in most user environments. As a lightweight application, Solaris Secure Shell is typically easier to deploy than other applications. Administrators looking for a more comprehensive security solution may want to consider Internet Protocol Security (IPsec), which is also included as part of the Solaris 9 OE. IPsec provides strong encryption and authentication capabilities at the IP level — lower in the network stack than Solaris Secure Shell. IPsec is also transparent to end users and the applications they are using.

TABLE 1 offers a side-by-side comparison of these two security technologies:

TABLE 1 A comparison of IPsec and Solaris Secure Shell technology

Feature	IPsec	Solaris Secure Shell
Ease of configuration and maintenance	more difficult	less difficult
Centralized policy control	Yes	No
Transparent to applications	Yes	No
Transparent to users	Yes	Application commands, such as <code>rsh</code> , are similar (<code>scp</code>)
Network protections	Privacy, strong host-to-host authentication, VPN, integrity, automatic keying	Privacy, strong user and host authentication, session integrity
Secure X-Windows sessions	Yes	Yes

For all its many advantages, IPsec requires additional effort to implement and manage, and Solaris Secure Shell grants users some control over their configuration. IPsec requires kernel-level support, while Solaris Secure Shell is designed to work on existing systems with minimal changes. Properly implemented in the appropriate situation or environment, each offers strong security for remote and network computing tasks.

The Solaris Secure Shell software in the Solaris 9 OE is easily integrated into most user environments. It is installed and “on” by default, and host keys are generated automatically. This enables users to quickly take advantage of the stronger security offered by the Solaris Secure Shell protocol. It works with both the SSH-1 and SSH-2 protocols, and is compatible with commercially available Solaris Secure Shell products. In addition, Solaris Secure Shell offers all the advantages that users expect from the Solaris environment, including internationalization compliance, extensive testing and quality assurance (QA), code auditing, and integration with Solaris Basic Security Module (BSM) auditing.

Enhancing the security of those tasks that require remote access can reduce the risk of intrusion, and result in higher service levels. By integrating these capabilities into the Solaris OE, acquisition, training, and implementation costs are lowered, resulting in an overall improvement in productivity.

The Solaris OE has a strong legacy of promoting and providing standards-based technologies for IT solutions. The Solaris Secure Shell implementation in the Solaris 9 OE is designed and built with adherence to Internet Engineering Task Force (IETF) standards.

This paper contains a description of network threats, and an overview of how Solaris Secure Shell can provide protection against these threats. This includes a description of Solaris Secure Shell features, capabilities, and its underlying technology. A discussion of security policy considerations is also included.

Chapter 2

Introduction

As more network users require greater access to remote systems, the risk of compromised accounts and systems increases. Without proper safeguards, login information (such as name and password) or transmitted data can be intercepted while travelling between the client and server. Once the attacker captures login information from one of these unsafe network protocols, that user's account is compromised. Since the network session is transferred in clear text, attackers can also eavesdrop on a session, take control of it (known as hijacking), or apply one of several other serious network attacks.

Solaris Secure Shell enables users and administrators to:

- Login to another host securely, even over an untrusted network
- Copy files securely between the two hosts
- Run commands securely on a remote host
- Tunnel X-windows network traffic safely

Solaris Secure Shell enables a user to securely access a remote host over a public or untrusted network. The software is built on a client-server architecture, with end-to-end encryption between the local client and the remote server. By executing a Solaris Secure Shell command, users connect to the Solaris Secure Shell daemon on a remote server. Authentication is provided by password and public keys, and network traffic is encrypted. This helps maintain privacy and prevents impersonation of users and servers (spoofing).

Solaris Secure Shell provides commands for remote login and file transfer. It can also be used as an on-demand virtual private network (VPN) to forward X-Window system traffic or individual network ports between the local and remote machines over the encrypted network link.

In the Solaris 9 OE, Solaris Secure Shell supports both SSH-1 and SSH-2 protocols. The SSH-1 protocol contained a number of problems and limitations, but is still widely deployed. The SSH-2 protocol rectifies some of the basic security design flaws of Version 1 and is more secure. For this reason, the SSH-1 protocol is included for backward compatibility, but it is not enabled by default. By supporting both protocols, the Solaris 9 OE facilitates migration efforts within organizations, their partners, and users.

Chapter 3

Network Threats

As network computing becomes an integral part of daily life, so does the potential for assets and information to be compromised by unauthorized users. As enterprises rely on the network for both internal and external relationships, the importance of security increases.

Yet, organizations everywhere face increasing risk of security attacks, from both internal and external sources. According to the 2001 CSI/FBI Computer Crime and Security Survey:

- 91% of the respondents detected employee abuse of Internet access privileges
- 85% reported security breaches within the last 12 months
- 76% reported that a likely source of attack would be disgruntled employees
- 40% detected system penetration from the outside
- 78% reported denial of service attacks
- 13% reported theft of transaction information (up from 8% in 2000)

Solaris Secure Shell can be effective against many security threats, and is an essential resource in a comprehensive effort to create a safe computing environment. It can also help prevent a number of network threats, which are described below.

Password Theft

When using conventional network access commands — telnet, ftp, and others — to perform remote computing operations, name and password data are sent in the clear across open networks to remote servers. Login information can be read, and later used for unauthorized access to computing resources.

Exploiting a Trust Relationship

By masquerading as another system, it is possible for an attacker to use spoofing to exploit a trust relationship, gaining entry to another system. Trusted nodes listed in the `.rhosts` file can be attacked by sending false hardware address information to a node, convincing it that packets from the attacking system are actually from a trusted system.

Session Eavesdropping

Using products or technology that are capable of listening to traffic as it passes by, an eavesdropper reads traffic without affecting it. When the traffic is unencrypted, information such as name, password, and network session can all be read by the eavesdropper.

Session Hijacking

When an attacker is capable of not only listening to network traffic but also inserting information, a session is susceptible to hijacking — which redirects it away from a legitimate end point. Solaris Secure Shell cannot prevent session hijacking, but can detect hijacking attempts.

Weak Authentication

Standard remote access tools, such as `telnet` and `ftp`, use conventional name and password data for login authentication. This system is inherently weak, because this data can be sniffed, guessed, or even observed while looking over the user's shoulder. Also, `r`-commands, such as `rlogin` and `rsh`, rely on information residing at the server, including `.rhosts` files, the source hostname, IP addresses, and privileged source ports. This information can help attackers to determine how to access target networks using packet spoofing.

Man-In-the-Middle Attacks

This type of attack is characterized by a person situated between the client and server, intercepting all traffic. Both client and server believe they are connected directly to each other, but instead are connected to a “man in the middle.” The attacker uses the intercepted login information to connect to the expected node, and is free to store, alter, or delete data on the target hosts.

DNS Spoofing

DNS spoofing is a term used for what happens when a DNS server accepts and uses incorrect information from a host that has no authority to give that information. This is accomplished by altering the cache of the name servers using forged DNS data. Spoofing attacks cause serious security problems to servers that rely on DNS information — for example, remote access services, such as the `r`-commands, that use host information for authentication.

Automated Network Sniffer

Automated network sniffers can be programmed to watch and record specific data from the overall network traffic. For example, a sniffer may be set up to capture root password data to a certain server, or financial information related to a specified person or company.

Insertion Attack

Unlike attacks where information is read or saved for later use, an insertion attack inserts data into the clear text data stream bound for either the client or server. This may be as harmless as disrupting the data stream, or it may insert potentially destructive commands.

Chapter 4

Solaris Secure Shell Protection

A number of features enhance security when connecting to remote servers and executing remote commands. These capabilities are described below.

Private Network Sessions

Strong encryption protects all information, including name, password, and data, from unauthorized access. This protection can be described as “end to end,” since it encrypts traffic between client and server as the data traverses local LANs and WAN segments, and can prevent password theft and session eavesdropping.

Traffic is encrypted using symmetric key encryption (3DES, Blowfish, AES). The keys for bulk encryption are exchanged using public key cryptography (Diffie-Hellman in SSH-2). Host and user keys can be either RSA or DSA. Cryptographic keys are securely negotiated for a particular session.

Strong Authentication

Connections involve two authentication mechanisms: A client process verifies the authentication of the server, and the server verifies the identity of the user who is requesting access. Client and server authentication processes negotiate to determine which authentication mechanism to use, based on their configurations. The strong authentication mechanism can prevent the exploitation of a trust relationship and man-in-the-middle attacks.

The SSH-2 protocol can be configured with a list of both required and allowed authentication techniques. By default, the SSH-2 protocol allows only password and public-key authentication. The SSH-2 protocol does not support `.rhosts` authentication, though separate control is available for root authentication. For maximum security, it's best to disable `.rhosts` access entirely. In the Solaris 9 OE, `.rhosts` authorization is off by default, though it is on for `rlogin` and `rsh`.

The client process helps ensure that the server is authentic. This guards against the connection being redirected to a different node. Man-in-the middle attacks — where an attacker sits in the middle pretending to be the client on one side and the server on the other — are also prevented.

The SSH-2 protocol associates keys with individual ports, enabling multiple keys per host. For example, SSH-2 uses multiple keys per session, with separate keys for incoming and outgoing traffic.

Unlike protocols like `ftp` and `rlogin` which send both name and password information in clear text, Solaris Secure Shell encrypts this data before it is sent to the server. The client authentication mechanism within Solaris Secure Shell also supports authentication technologies more secure than name and password, such as per-user public keys. One-time password solutions are also available from third-party providers.

Solaris Secure Shell can be used to protect X-Windows sessions. This includes both host-based and key-based authentication methods. Using X-Windows forwarding, Solaris Secure Shell provides secure transparent authentication and key transfer for X-Windows sessions.

Solaris Secure Shell also uses personal key agents. These programs cache a user's private keys, and respond to authentication-related queries from Solaris Secure Shell clients. Once a user enters a pass phrase, the private key is decrypted and stored in memory by the agent. The personal key agent handles all key-related operations, eliminating the need to retype the pass phrase. Note that the clients use the personal key agents without directly accessing the private keys. If Solaris Secure Shell needs to sign an authenticator, it sends the agent a signing request containing the authenticator data, and designates which key to use. The agent performs the cryptographic operation and returns the signature. This technique replaces the need to directly expose the keys, and is more secure than handing them out to clients.

Session Integrity

Solaris Secure Shell can prevent session hijacking by performing integrity checks. These checks verify that the data has not been altered, and that it comes from the other end of the connection as expected. While TCP/IP can perform integrity checking, it does so only on a packet-by-packet basis. In contrast, Solaris Secure Shell offers integrity checking for the entire data stream. This helps ensure that the traffic is received as it was sent: in order and with no duplication. Session hijacking and insertion attacks can also be prevented. Note that nothing is added or deleted from the session traffic.

- The SSH-1 protocol uses CRC-32 to check each packet in the data stream. This is a non-cryptographic hash function, and does not protect against an insertion attack. While later versions of the protocol offer some protection against this type of attack, there is still a significant security risk. Because CRC-32 is considered a weak integrity checking solution, this is one of the reasons SSH-1 is disabled by default in the Solaris 9 OE.
- The SSH-2 protocol uses integrity checking based on cryptographic technologies, including keyed hash algorithms based on MD5 (128-bit) and SHA-1 (160-bit).

Agent Forwarding

Agent forwarding can provide single sign-on functionality among supported applications and services. An authentication agent, running in the user's laptop or local workstation, can be used to hold the user's RSA authentication keys. Solaris Secure Shell automatically forwards the connection to the authentication agent over any connection, and there is no need to store the RSA authentication keys on a machine in the network (except the user's own local machine). The authentication protocols never reveal the keys; they can only be used to verify that the user's agent has a certain key.

X-Windows Session Tunneling

X-Windows protocol connections, as with many other remote connections, pass unencrypted data between client and server. X-Windows protocol connections can be routed through a Solaris Secure Shell connection, providing stronger authentication and security. Also known as X-Windows forwarding, X-Windows session tunneling can be used to overcome firewall-related issues, by permitting X-Windows protocol connections to securely pass through the firewall.

X-Windows forwarding works by redirecting X client communications setting the DISPLAY environment variable to the X-Windows proxy set up on the Solaris Secure Shell server. The proxy acts like an X-Windows server, and instructs the Solaris Secure Shell client to behave as an X-Windows client. The client and server cooperate to pass X-Windows protocol data back and forth over the Solaris Secure Shell connection. So the X-Windows client application appears on the client screen just as if it had connected directly to the X-Windows server.

Note – Note that by default, the X-Windows forwarding functionality is disabled in the Solaris 9 OE to help ensure a higher level of security.

Easy Integration

Solaris Secure Shell is easily integrated into most user environments. In the Solaris OE, it is installed and “on” by default, and host keys are generated automatically on the first system boot after installation (if they do not already exist). This enables users to quickly take advantage of the stronger security offered by the Solaris Secure Shell protocol.

Similar Replacement Commands For Network Access Commands

Solaris Secure Shell uses similar commands, with similar command line options, to those with which most users are familiar. It is backward compatible with r-commands. While Solaris Secure Shell commands and r-commands can coexist on the same node, the r-commands are not secure. Table 2 summarizes the Solaris Secure Shell commands and their functions.

TABLE 2 A summary of Solaris Secure Shell commands

Conventional Command	Solaris Secure Shell Equivalent
rsh	ssh Executes a remote shell.
rcp	scp Copies files between hosts on a network, using ssh for data transfer.
rlogin	slogin Performs a remote login
ftp	sftp Performs a file transfer from a remote host. Note that sftp is not the FTP protocol over SSH protocol. It is a completely separate protocol being defined by the SECSSH working group at IETF. While the protocol is different, the user interface on the client is similar to ftp(1).

Compressed Data Streams

Solaris Secure Shell features data compression, meaning that traffic can be compressed automatically before it leaves the client, and uncompressed after it is received and decrypted at the server. Compression is used to reduce overall network traffic and speed transmission times in both dial-up and LAN-based connections. However, compression introduces a level of overhead into the overall session, which can be overcome with fast processors. Users without fast processors may want to operate without compressed data streams or at least, reduce the level of compression.

Solaris Secure Shell allows users to adjust the compression level to achieve optimum performance. For example, systems containing fast processors and connecting over a slow line may need to use a higher level of compression, while slower processors using a fast connection may want to use less compression, or none at all.

Chapter 5

Solaris Secure Shell Features

In addition to the features cited above, the Solaris Secure Shell implementation in the Solaris 9 OE offers many additional features, including:

- *Interoperability with SSH-1 and SSH-2 protocols:* While these two protocols behave in a seemingly similar way, they are different in how they provide secure connectivity. Due to complications with licensing and other factors, many environments still use the SSH-1 protocol, even though it is widely considered less secure. The Solaris 9 OE implementation of Solaris Secure Shell works with both protocols.
- *Interface Compatibility With Commercial SSH and OpenSSH:* A number of commercially available Solaris Secure Shell implementations exist, including SSH. An open source version, OpenSSH, is available from the OpenBSD team. The Solaris 9 OE implementation of Solaris Secure Shell has been tested for interoperability at the protocol layer with other secure shell implementations.
- *Code Audited for Security:* The Solaris 9 OE version of Solaris Secure Shell has been code audited to detect security and logic flaws that may interfere with application integrity or otherwise prevent interoperability and proper operation.

- *Built-in User Safeguards:* Properly implemented, Solaris Secure Shell offers additional security for many network uses. In the Solaris 9 OE, it offers users and administrators protection against inadvertently exposing system assets to unauthorized usage. For example, if a user connects to a Solaris Secure Shell server where the host key has changed, a warning can be printed along with the fingerprint of the new key. This fingerprint can be verified against a known value to ensure that the Solaris Secure Shell server has not been tampered with. Warnings are configurable, and may be set up in such a way that the client will refuse a connection if the server key has changed.
- *Internationalization:* The Solaris Secure Shell is designed to transparently handle various cultural and linguistic conventions without additional modifications.
- *Port Forwarding:* Solaris Secure Shell can transparently encrypt and decrypt TCP/IP traffic from other applications — such as Telnet, IMAP, SMTP, and NNTP — on other TCP ports. Note that port forwarding does not work with protocols that are not built on TCP. This includes UDP-based technologies such as DNS, DHCP, and NetBIOS, as well as Novell IPX/SPX and AppleTalk. While NFS runs on TCP, it is a kernel service and cannot be tunneled.
- *Solaris OE Audit Enabled:* Solaris Secure Shell in the Solaris 9 OE supports kernel-level auditing. The SunSHIELDTM Basic Security Module (BSM) enables event logging down to the system call level.
- *Uses /dev/urandom for Entropy Collection:* A new feature of the Solaris 9 OE is a pseudo random number generator (PRNG). Cryptographic algorithms and protocols require a source of random bytes, also known as entropy. An entropy collection is a pool of random bytes that can be used for cryptographic operations. Entropy is used to generate data encryption keys, to help prevent spoofing attacks, and otherwise frustrate cryptoanalysis. /dev/urandom is a source for random bytes generated by a kernel PRNG. /dev/urandom provides a constantly filled entropy pool without imposing significant system overhead. Solaris Secure Shell draws from this pool, without pausing, and uses the byte strings as input for its own random number generation system. A pool of readily available random numbers can help provide stronger security.

Chapter 6

Security Policy Considerations

Many organizations have relied on firewalls to provide the bulk of their perimeter security needs. Solaris Secure Shell offers a number of security enhancements for remote computing tasks outside the security perimeter. For this reason, there are a number of security policy implications that must be considered before implementing Solaris Secure Shell.

Specific Policy Statements

As with any IT tool, the organization's security policy should make specific policy statements regarding the use of Solaris Secure Shell. This should cover expected uses, prohibited uses, and procedures to be followed in case of unexpected results.

Solaris Secure Shell was designed as a secure replacement for unsafe network commands, such as `rlogin`, `rsh`, `rcp`, `telnet`, and `ftp`. The way in which Solaris Secure Shell is configured should reflect the organization's security policy.

For example, should:

- Connections be allowed from within the firewall to a user's home machine?
- Password or two-factor password authentication be used?
- Users be allowed to install their own Solaris Secure Shell servers?
- Connections be allowed to nonauthorized Solaris Secure Shell servers?
- Connections be allowed if a key has changed?
- TCP or X-Window connections be tunneled?

Other procedures that should be considered:

- What should administrators do when changing or adding server keys?

- Which types of automated processes should use Solaris Secure Shell, and which ones should not?
- What should users do if they are trying to connect to a server where the key has been changed?

The Solaris 9 OE version of Solaris Secure Shell is highly configurable — system capabilities can be made to match organizational security policies. As with any new tool or capability, the security policy should inform users and administrators of acceptable and expected usage.

The Solaris 9 OE version of Solaris Secure Shell is highly configurable — system capabilities can be made to match organizational security policies. As with any new tool or capability, the security policy should inform users and administrators of acceptable and expected usage.

Port Forwarding

Beyond enabling or disabling this feature, there is very little control over the traffic — it provides “all or nothing” capabilities. Once forwarding is enabled, the client may forward any port to any place on the remote side. Host-based firewalls at the Solaris Secure Shell server can be used to provide additional connection and traffic controls.

Because traffic on the Solaris Secure Shell connection is encrypted, neither a firewall nor an intrusion detection system can detect when something abnormal is happening. In this type of situation, firewalls and intrusion detection should be reconfigured to detect abnormal traffic.

Gateway Ports

Gateway ports work in conjunction with port forwarding. Port forwarding typically allows only the local host to send data to the other side of a Solaris Secure Shell connection. If a gateway port is enabled, any other machine may connect to this port and have its data forwarded. In effect, a tunnel from one network to another network is created. This is generally considered a security risk. By default, gateway ports are disabled in the Solaris 9 version of Solaris Secure Shell.

Authentication Requirements

The Solaris 9 OE version of the Solaris Secure Shell supports multiple forms of authentication, including traditional login and password, two-factor, and host-based. Each method has different requirements and benefits:

- Passwords fit well into existing structures. Organizations with a large number of internal users to support may want to consider staying with passwords to reduce training costs.
- Two-factor authentication requires something the user knows, (such as a pass phrase) and something the user has (such as a private key) before being authenticated. Two-factor authentication offers enhanced security, along with higher maintenance costs. Access to critical areas such as assets on the management network should use two-factor authentication. In these situations, sites with remote users or those needing to automate jobs should consider using key-based authentication.
- Host-based authentication, which depends on a `.rhosts` file, is extremely unsafe and easily abused, but provides the most convenience.

Key Handling

Solaris Secure Shell does not offer key management functionality. In general, all key operations must be performed manually. Each individual server has its own key, which clients must verify on first use, on a case-by-case basis. At this time, there is no certificate process provided by a Public Key Infrastructure (PKI).

As mentioned above, changing host keys will have an impact on clients — a procedure should be implemented and a policy distributed on what actions to take when faced with new server keys.

Auditing

The Solaris 9 OE provides BSM auditing capabilities, which are supported by the Solaris Secure Shell application. Once Solaris Secure Shell is implemented, user actions such as login and remote administrative activity will be logged. Machine-initiated processes, such as a cron job that uses Solaris Secure Shell, will also be logged.

Unsafe Network Services

The Solaris 9 OE ships with r-command services still activated. Users are not likely to switch to Solaris Secure Shell unless these are turned off. System administrators should consider disabling unsafe network services such as telnetd, ftpd, rlogind, and rshd after deploying the Solaris 9 OE.

Chapter 7

Conclusion

As the demand on open networks for remote access has grown, the risks of compromised systems and accounts has kept pace. Network tools such as Solaris Secure Shell have been developed to counter the threats of password theft, session hijacking, and other network attacks. The Solaris 9 OE version of Solaris Secure Shell offers a number of benefits that can enhance overall system security, providing users with:

- *Strong Network Session Protection:* All traffic, including data and login information, is encrypted for privacy across both public and private networks. Integrity checks help assure the user that the traffic has not been tampered with.
- *Commands With a Simple, Familiar Interface:* Solaris Secure Shell offers very similar commands and controls to the commands being used today. This minimizes training requirements.
- *Stronger Host and User Authentication Mechanisms:* Password, public key, and one-time password systems can be used as required, while r-commands, telnet, and ftp use only simple password controls.
- *Support for Legacy Protocols:* The security of many TCP services, as well as X-Windows, can be enhanced through the tunneling capabilities of Solaris Secure Shell.
- *Flexibility in Deployment:* Solaris Secure Shell is a powerful asset for protecting network resources. It is a flexible tool to be used in meeting an organization's security policy requirements. By reducing the risk of intrusion and raising the level of data privacy, improved data integrity may help increase overall service levels. Because these capabilities are integrated into the Solaris 9 OE, the cost of acquisition, training, and implementation is reduced.

Chapter 8

Additional Resources

- Building and Deploying OpenSSH for the Solaris Operating Environment, by Keith Watson and Jason Reid (www.sun.com/blueprints/0701/openSSH.pdf)
- Configuring OpenSSH for the Solaris Operating Environment, by Jason Reid (www.sun.com/blueprints/0102/configssh.pdf)
- The Solaris Secure Shell working group of the Internet Engineering Task Force (IETF) (<http://www.ietf.org/html.charters/secsh-charter.html>)
- The Sun Fingerprint Database (<http://sunsolve.Sun.COM/pub-cgi/show.pl?target=content/content7>)
- Solaris 9 System Administrator AnswerBook (docs.sun.comSM)

Copyright 2002 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Solaris, docs.sun.com, SunSHIELD, and Solaris Secure Shell are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2002 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, docs.sun.com, SunSHIELD, Solaris, et Solaris Secure Shell sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



Please
Recycle



Adobe PostScript

Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, CA 94303-4900 USA Phone 800 786-7638 or +1 512 434-1577 Web sun.com



We make the net work.

Sun Worldwide Sales Offices: Africa (North, West and Central) +33-13-067-4680, Argentina +5411-4317-5600, Australia +61-2-9844-5000, Austria +43-1-60563-0, Belgium +32-2-704-8000, Brazil +55-11-5187-2100, Canada +905-477-6745, Chile +56-2-3724500, Colombia +571-629-2323, Commonwealth of Independent States +7-502-935-8411, Czech Republic +420-2-3300-9311, Denmark +45 4556 5000, Egypt +202-570-9442, Estonia +372-6-308-900, Finland +358-9-525-561, France +33-134-03-00-00, Germany +49-89-46008-0, Greece +30-1-618-8111, Hungary +36-1-489-8900, Iceland +354-563-3010, India-Bangalore +91-80-2298989/2295454; New Delhi +91-11-6106000; Mumbai +91-22-697-8111, Ireland +353-1-8055-666, Israel +972-9-9710500, Italy +39-02-641511, Japan +81-3-5717-5000, Kazakhstan +7-3272-466774, Korea +822-2193-5114, Latvia +371-750-3700, Lithuania +370-729-8468, Luxembourg +352-49 11 33 1, Malaysia +603-21161888, Mexico +52-5-258-6100, The Netherlands +00-31-33-45-15-000, New Zealand-Auckland +64-9-976-6800; Wellington +64-4-462-0780, Norway +47 23 36 96 00, People's Republic of China-Beijing +86-10-6803-5588; Chengdu +86-28-619-9333; Guangzhou +86-20-8755-5900; Shanghai +86-21-6466-1228; Hong Kong +852-2202-6688, Poland +48-22-8747800, Portugal +351-21-4134000, Russia +7-502-935-8411, Singapore +65-6438-1888, Slovak Republic +421-2-4342-94-85, South Africa +27 11 256-6300, Spain +34-91-596-9900, Sweden +46-8-631-10-00, Switzerland-German 41-1-908-90-00; French 41-22-999-0444, Taiwan +886-2-8732-9933, Thailand +662-344-6888, Turkey +90-212-335-22-00, United Arab Emirates +9714-3366333, United Kingdom +44-1-276-20444, United States +1-800-555-9SUN or +1-650-960-1300, Venezuela +58-2-905-3800