Global Information Assurance Certification Paper

Richard Cove, 26 November 2000

Secure Backups on Solaris Internet servers.

Introduction

Backing up Unix servers remotely, securely and cheaply usually means compromising one of the above principals. Traditional methods have included using rexec utilities, tar and commercial backup programs. The methods discussed here are geared towards a typical Internet gateway problem. That is a number of servers (say: web, firewall, and ftp) that have a small amount of data on each server these servers are often located in different security zones and a high level of trust between these servers is not desirable. Allowing unattended backups between machines means that a high level of trust must be allowed between the machines in question, at a minimum the backup server must have read access to the entire disk of the client being backed up. Often what happens in many situations, is the backups are done infrequently or not at all, as the security is difficult to maintain.

This paper discuses the use of ssh and ufsdump to more securely backup filesystems. Included are commands and scripts that can be applied a solaris system. The commands have been tested on solaris systems; the general principals should apply to most Unix systems.

Commercial backup solutions tend to be geared to backing up large amounts of data quickly on a lan or wan. Very little thought is given to the security of the backups, typically the server and clients must use rpc and require you to open up a large number of ports if done across a firewall.

Solaris is supplied with an easy to manage utility, ufsdump that is adequate for backing up filesytems where the tape is local. Ufsdump and its companion ufsrestore are capable programs that allow a system to be restored quickly and easily from a complete disaster using just the tape drive and booting the system from CDROM. Ufsdump does not however provide any tape management functionality and it is up to the operator to carefully document what is contained on the tape drive.

When the tape drive is not on the same system as the one being backed up a rhost trust between machines needs to be set up. The security implications of the r* commands are well documented. This trust is based on IP source address that is subject to being spoofed, and all communications, including passwords (if used) are in clear text.

Ssh provides a more secure replacement to the r* commands, by using encryption, PKI keys, and access control lists. Any unattended backup requires that entered passwords are to be entered, so the trust relationships must be set up carefully. To reduce risk it is recommended that the trust be limited to a specially created backup user. At a minimum this user must have read only access to the device that the raw file system is contained on and write access to the tape drive.

How to set up the backups.

Consider the scenarios that are required.

There are 4 variations of local / remote backups that can be performed. These are described in relation to the server that is running the backup script.

Scenario	Trust Required
Local Filesystem Local tape, ufsdump to tape.	Read access to local filesystem Write access to tape drive only
Local Filesystem to Remote tape	Script will ssh to remote tape machine
Remote Filesystem to Remote tape	Script will ssh to remote system, remote system will then ssh to the tape machine.
Remote Filesystem to local tape	Script will ssh to remote system, remote system will then ssh back to the (local) tape machine.

Set up access control

The access control is dependant on the security installed at you site. You will need to allow port 22 access from the local to the remote machine, after installing ssh you will need to set up the access control that for the machine to be logged into, typically the /etc/sshd_config and the /etc/hosts.allow files.
If your firewall allows you can use time based ACL's to only allow the machines access at defined times.

Set-up the backup user

On each machine create a special backup user, for the examples this user is backup and the home directory is /backup. Place this user in the sys group, this allows the backup user read access to the raw filesystem on most solaris systems. To check that the sys group has the read access perform for each partition that is to be backed up a "ls –l /dev/dsk/c0t0d0s?" the actual disk partition is pointed to by the symbolic link do a "ls –l" on this device to see the permissions attached to this device, if required add read access for sys group to this device, an example is below.

```
lrwxrwxrwx  root root  41 Sep 15 11:33 /dev/dsk/c0t0d0s0 -> / devices/pci@1f,4000/scsi@3/sd@0,0:a
brw-r-----  root sys 32,  0 Sep 15 11:33 /devices/pci@1f,4000/scsi@3/sd@0,0:a
```

Note that ufsdump does not need read access to each file on the filesystem only the raw device as described above

Set-up ssh access

Obtain and install ssh, the version described here is Version 1.2.30. Any current version of ssh acceptable, procedures may vary.
To create public and private keys perform the following procedure on a Unix server with SSH installed.
Run ssh-keygen, this will generate a public key (~/.ssh/identity.pub) and a private key (~/.ssh/identity) for the machine that it is run on. Do not enter a passphrase if you do not want to be prompted for passwords during the login process. Be careful not to allow the private key to be divulged to anyone.
Append the contents of the locally generated ~/.ssh/identity.pub to the ~./ssh/authorized_keys on the remote machine.

Test that a ssh login, without password can be performed from the local machine without any prompts, with a command “ssh remote.machine ls”, the first time you login you may be prompted for a confirmation that the host key is correct if this key is not automatically saved you will need to to manually add it to the ~.ssh/known_hosts file.
This procedure will need to be reversed if a 2-way trust is going to be set up.

Test the backups

Now we are ready to go, manually run the ufsdump commands to test the backups. During testing it is best to only dump a subset of the filesystem for speed.

Local Tape Local filesytem

```
ufsdump 0cuf /dev/rmt/0cn /etc/mail
```

Local Filesystem, Remote tape dump

```
ufsdump 0cuf - /etc/mail | \
ssh –l backup tapehost "dd bs=1024 of=/dev/rmt/0cn"
```

Remote Filesystem, Remote tape dump

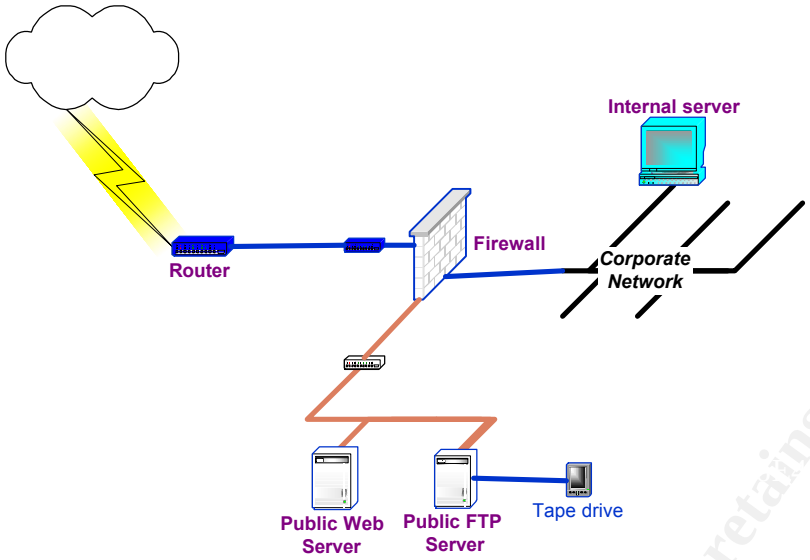
```
ssh -l backup remote_files_machine ufsdump 0f - /etc/mail | \
ssh -l backup tapehost "dd bs=1024 of=/dev/rmt/0cn"
```

Remote Filesystem, Local tape dump

```
ssh -n -l backup remotehost “ufsdump 0f - /etc/mail”  | \
`dd bs=1024 of=/dev/rmt/0cn`
```

Choose the machine to run the backups from

The machine that controls the backups does not need to have the tape on it. Choosing the correct machine can reduce the amount of trust required between machines.
An example (as shown in the diagram) would be: the firewall controls the backup with the tape drive located on a machine located in the dmz of the firewall.
This way there is no trust needed inbound to the firewall and the internal servers only need to trust the firewall.
This is not perfect but remember backing up servers over the network always involves a compromise.



Complete script

```
This is a complete script that implements the principles described above save the
script into a directory that is readable to the backup user as backup_ssh, add execute
permissions.  Then add an entry into the backup cron to run as required.
usage: backup_ssh level filesystem...
eg:
20 0 * * * * /usr/backup/backup_ssh 0 / /var /opt other_system:/var


#!/bin/ksh
# Backup script using ufsdump and ssh for remote backups
#
# Script should produce no output and results will be mailed to $Backupmail
#   There are 4 possibilities
#
#  1. Local Filesystem Local tape, ufsdump to tape
#  2. Local Filesystem to Remote tape, script ssh to tape machine
#  3. Remote Filesystem to Remote tape, script will ssh to remote system,
#       remote system will then ssh to the tape machine
#  4. Remote Filesystem to local tape, script will ssh to remote system,
#       remote system will then ssh back to the (local) tape machine
#
# Create a ssh trust between all machines that are to be logged into
#   1.   Create a RSA key with ssh-keygen on each machine, give a null passphrase.
#   2.   Add the ~/.ssh/identity.pub key into the machine that is to be logged into
#   3.   Backing up a remote filesystem will require a 2 way ssh trust relationship
#
# If backup user is not root the user will need to be in the sys group or the
#   raw filesystem will need to be read access for that user
# Revision History
#
# The machine that the script is running on
Thishost=`uname -n`
#
# The machine that has the tape drive
if [ "${Tapehost}" = "" ] ; then
    Tapehost="tape_host"
fi
#
# Tape device name, use the no rewind device
Tape=/dev/rmt/0cn
#
# Who do I send mail to
Backupmail="root"
#
# UID for the user to run the ssh as
# this user must have permission to read from the raw filesystem
# Put the backup user into the "sys" group
BackupUID=backup
#
# SSH identity file
IDENTITY=/backup/.ssh/identity
Backuperr=0
Output="/tmp/backup.tmp"
# location of ssh
SSH="/usr/local/bin/ssh"
UFSDUMP="/usr/sbin/ufsdump"
#
my_dump () {
    # Function my_dump, parameters required:
    #   $1 is Hostname where the partition lives
    #   $2 is Disk Partition to dump
    FS=$2
    if [ ${Tapehost} = $Thishost -a ${Thishost} = ${1} ]; then
        echo Local dump to local tape: Level ${Level} filesystem ${FS}
        echo "${UFSDUMP} ${Level}f ${Tape} ${FS}"
        ${UFSDUMP} ${Level}f ${Tape} ${FS}
        # set error flag if any problems
        Ufsstat=$?
        if [ $Ufsstat != 0 ] ; then Backuperr=$Ufsstat ; fi
            elif [ ${Thishost} = ${1} -a ${Tapehost} != $Thishost ]; then
            echo Local Filesystem, Remote tape dump: Level ${Level}, \
                filesystem ${1}:${2}, Tape ${Tapehost}:${Tape}
            ${UFSDUMP} ${Level}f - ${2} | ${SSH} -l ${BackupUID} ${Tapehost} \
```

```
            "dd bs=1024 of=${Tape}"
        # set error flag if any problems
        Backuperr=$?
        #Ufsstat=$?
        #if [ $Ufsstat != 0 ] ; then Backuperr=$Ufsstat ; fi
            elif [ ${Thishost} != ${1} -a ${Tapehost} != $Thishost ]; then
        echo Remote Filesystem, Remote tape dump: Level ${Level}, \
                    filesystem ${1}:${2}, Tape ${Tapehost}:${Tape}
        ${SSH} -l ${BackupUID} -i ${IDENTITY} ${1} ${UFSDUMP} ${Level}f - ${2} | \
        ${SSH} -l ${BackupUID} -i ${IDENTITY} ${Tapehost} "dd bs=1024 of=${Tape}"
        # set error flag if any problems
        Backuperr=$?
    else
        echo Remote Filesystem, Local tape dump: Level ${Level}, \
             filesystem ${1}:${2}, Tape ${Tapehost}:${Tape}
        ${SSH} -n -i ${IDENTITY} -l ${BackupUID} ${1} ${UFSDUMP} ${Level}f - ${2} | \
            `dd bs=1024 of=${Tape}`
        # set error flag if any problems
        Backuperr=$?
    fi
}
#
tape_command () {
    # Run a command on the appropriate tape drive
    # and set the Backuperr according to the result
    if [ ${Tapehost} = $Thishost ]; then
        echo "Local tape drive: Tape, $Tape Command, $1"
        mt -f $Tape $1
        Backuperr=$?
    else
        echo "Remote tape drive: Tape, $Tape Command, $1"
        ${SSH} -l ${BackupUID} -i ${IDENTITY} ${Tapehost} "mt -f $Tape $1"
        Backuperr=$?
    fi
}
##############################
# Main script starts here
##############################
if [ $# -lt 2 ] ; then        # check for parameters
        echo ""
        echo "ufsdumps up at level (0 to 9) the specified filesystems to ${Tape} "
        echo " mails results to ${Backupmail}"
        echo "usage: `basename $0` level filesystem... "
        exit 0
fi
rm /tmp/backup.tmp

# Set the dump level from the first parameter.
Level=$1
shift 1

# Check to see if there is a tape in the drive
tape_command status >>  $Output 2>&1
if [ $Backuperr = 0 ] ; then
    for i in $* ; do
    # for each paramater on the command line.
    # the parameters should be in the form of "filesystem" or "hostname:filesystem"
        # Parse the parameter to split the hostmame from the files-system
        DUMPhost=`echo $i | cut -d : -f1`
        # Check for a local filesystem if local the
        # DUMPhost will = the original parameter
        if [ "${DUMPhost}" = ${i} ] ; then
            DUMPhost=${Thishost}
        fi
        FILEsystem=`echo $i | cut -d : -f2`
        # Finally we run the dump
        my_dump ${DUMPhost} ${FILEsystem} >> $Output 2>&1
    done
fi
#
echo Backup script done >> $Output 2>&1
#
# Mail the administrator with the results and eject the tape if
# backup was successful
#
if [ $Backuperr = 0 ] ; then        # if we detected any errors mail admin staff
    # Send mail notification.
```

```
    mailx -s "${Thishost} - backup complete no errors" $Backupmail < $Output
    tape_command offline >> $Output 2>&1
else
    #Send error notification.
    mailx -s "${Thishost} - backup error " $Backupmail < $Output
    tape_command rewind >> $Output 2>&1
fi
#
```

Suominen Kimmo “Getting started with SSH” 20 August 1999
URL: http://www.tac.nyc.ny.us/~kim/ssh/ (22 Sept 2000)

Ylönen Tatu “SSH FAQ” Revision 1.3 - 21 June 2000
URL: http://www.tigerlair.com/ssh/faq/ssh-faq.html (22 Sept 2000)

Galvin Peter “Enter the secure shell” 30 March 2000 URL:
http://www.sunworld.com/sunworldonline/swol-02-1998/swol-02-security_p.html
(22 Sept 2000)

Sun Microsystems “Info Docs document 20812 Primary Ufs Boot Disk Emergency Backup and Restore” URL: http://sunsolve.sun.com

W. Curtis Preston, Curtis W. Preston, Gigi Estabrook (Editor) “UNIX Backup and Recovery” 1st edition (December 15, 1999) O'Reilly & Associates