25 DAYS UNTIL THE SCHEDULED RELEASE OF OPENBSD 4.6!

SECURITY SUGGESTIONS FOR OBSD 4.5

After the initial OpenBSD install you will want to update all of your sources to stable and then rebuild. Next you might want to batten down the hatches a little more.

What follows are recommendations only. Some people will view such measures as too paranoid or not extreme enough. Using or not using the below suggestions is up to you.

OBSD 4.5 PATCHES

» 008: Reliability Fix : Oct 5, '09
i386 only

» 007: Reliability Fix : Jul 29, '09
All architectures

» 006: Reliability Fix : Jun 24, '09
All architectures

» 005: Reliability Fix : Apr 24, '09
All architectures

» 004: Reliability Fix : Apr 24, '09
All architectures

» 003: Reliability Fix : Apr 24, '09
i386 only

» 002: Reliability Fix : Apr 11, '09
All architectures

» 001: Reliability Fix : Apr 8, '09
All architectures

... more

SECURITY LEVELS

Security levels essentially set how the kernel will handle system security. There are four security levels: -1, 0, 1, 2. Secureleve two being the most secure level. Securelevels are set from the `/etc/rc.securelevel` file.

Each level briefly explained

`securelevel -1` : There's no additional kernel security and many of the normal security features, such as permissions, are functional. Use this level for machines not in production use.

`securelevel 0` : When OpenBSD first boots up securelevel 0 is used. If this level is set in your rc.securelevel file securelevel 1 will actually be used when the boot process is finished. There are no added features of securelevel 0.

`securelevel 1` : OpenBSD's default securelevel. Writing to /dev/mem and /dev/kmem won't work. Raw disk devices are read-only. Schg and sappnd flags cannot be removed. Kernel modules cannot be loaded or unloaded 'on the fly'.

`securelevel 2` : Includes all securelevel 1 features plus: Limited setting of the system clock. pfctl cannot change PF or NAT rules. DDB kernel debugger sysclt values cannot be changed.

The end of the boot process will show what security level you are at.

Or at a prompt

```
# sysctl kern.securelevel
```

To adjust to a higher security level at the command prompt

```
# sysctl kern.securelevel=2
```

You cannot adjust from a higher security level to a lower security level at the command prompt. To lower security levels you will have to reboot.

USING KERNEL FLAGS

Setting kernel flags is like setting permissions but with an added twist. With the setting of some flags, not even root can make changes. Changes can only be made by booting into a lower securelevel or booting into single user mode.

Common used flags

sappnd : Can only be set or removed by root. Files set with this flag can be added to but not removed or edited. Good for log files. This flag cannot be removed with the system running in securelevel 1 or greater.

schg : Can only be set or removed by root. Files set with this flag cannot be changed, moved or replaced. This flag cannot be removed with the system running in securelevel 1 or greater.

uappnd : Can be set or removed by user or root. Files can be added to but not edited or removed by the average user (prevents accidental removal). The user or root may remove this flag at any time.

Using kernel flags can become addicting. Just make sure you know the overall outcome of using flags and realize that improper use may cause some serious system problems.

Checking to see if a file has a flag set

```
# ls -lo /bsd
-rw-r--r--  1 root  wheel  schg 5358488 Mar 30 11:47 /bsd
```

The schg text is the evidence of a flag being set.

Two popular flag settings

Disallowing changes to the kernel

```
# chflags schg /bsd
```

Disallowing changes to the binaries

```
# chflags -R schg /bin
```

You might want to set a sappnd flag to root's history file. If there is a remote root compromise of the system then looking over the tamperproof history file will help in tracing the intruder's movements.

Also, setting the sappnd flag to a user's history file will also prevent the old script kiddie trick of covering their tracks by sending shell history output to /dev/null via a soft link.

Removing a flag

Removing a flag set to the kernel file

```
# chflags noschg /bsd
```

You must be in securelevel 0 or -1 to remove this flag.

KERNEL FLAGS FOR THE PARANOID

What follows are some kernel flag suggestions for the paranoid. I recommend these changes only after you are done setting up your OBSD server.

Flag settings for the kernel and configuration files

```
# chflags schg /bsd
# chflags schg /etc/changelist
# chflags schg /etc/daily
# chflags schg /etc/inetd.conf
# chflags schg /etc/netstart
# chflags schg /etc/pf.conf
# chflags schg /etc/rc
# chflags schg /etc/rc.conf
# chflags schg /etc/rc.local
# chflags schg /etc/rc.securelevel
# chflags schg /etc/rc.shutdown
# chflags schg /etc/security
# chflags schg /etc/mtree/special
```

Flag settings for system binaries

```
# chflags -R schg /bin
# chflags -R schg /sbin
# chflags -R schg /usr/bin
# chflags -R schg /usr/libexec
# chflags -R schg /usr/sbin
```

ENCRYPT THE SWAP PARTITION

Encrypting your swap partition is mainly done to prevent any local user from potentially abusing the system.

By default OpenBSD 4.5 will encrypt the swap partition. To turn this on for OpenBSD versions 3.7 and below:

- step 1 - Enable this feature without a reboot
- step 2 - Edit the sysctl config file, so that after a reboot the swap partition will be encrypted

1. As root change the kernel state variable

```
# sysctl vm.swapencrypt.enable=1
```

2. Edit `/etc/sysctl.conf` from

```
#vm.swapencrypt.enable=1
```

to:

```
vm.swapencrypt.enable=1
```

and to check if the kernel state is set:

```
# sysctl vm.swapencrypt.enable
```

DISABLE INETD

On a default install inetd is enabled. On my OpenBSD server at home I only run sshd, ntpd, syslogd, and httpd. None of which run off of inetd. But for the paranoid disabling inetd will usually cause no problems.

Disable inetd by editing the `/etc/rc.conf` file from

```
inetd=YES
```

to:

```
inetd=NO
```

and to stop inetd without a reboot:

```
# kill `cat /var/run/inetd.pid`
```

Note: It isn't inetd that has had past security problems but rather the services it controls.

SSH OVER TELNET

Telnet will not be running on a default OpenBSD install. I'm not sure there are any good arguments to running the telnet service. As most know the telnet login process uses plain text authentication, which makes sniffing a practical attack to gaining illegal remote access to a system. Then next on the menu would be performing a local exploit.

Ssh not only encrypts the login (authentication) process but the entire ssh session is encrypted.

Almost all Linux distros and BSD flavors include the OpenSSH server and client. And for Windows, Putty would be the equivalent to a free client.

To disable telnet in OBSD 3.9 and below (4.5 does not have a telnet entry):

Edit the `/etc/inetd.conf` file from

```
telnet
```

to:

```
#telnet
```

SFTP OVER FTP

Sftp will be running on a default install. Sftp will prevent the problem of sniffing ftp passwords which are transmitted in plain text.

You might be surprised how easy it is to use sftp. Almost all Linux distros and BSD flavors come with a sftp client. And for Windows there is the freeware program WinSCP.

Note: There are performance issues when using sftp. You will notice transfer speeds to be slower than ftp speeds. This can be 'blamed' on the fact that sftp communication is encrypted thus adding to the transfer time.

MOUNTING PARTITIONS

The way partitions are mounted can greatly affect system security. How partitions are mounted at boot time is controlled by the `fstab` file. Two examples of a `/etc/fstab` file with security in mind:

1. The following layout shows an average paranoid setup

```
/dev/wd0a / ffs rw 1 1
/dev/wd0h /home ffs rw,nodev,nosuid 1 2
/dev/wd0d /tmp ffs rw,nodev,nosuid,noexec 1 2
/dev/wd0g /usr ffs ro,nodev 1 2
/dev/wd0e /var ffs rw,nodev,nosuid,noexec 1 2
```

The difference between the two, the below has the root (`/`) partition set to `read-only` and the `/home` partition set to `noexec`.

2. More paranoia added to the mix with a dash of less usability

```
/dev/wd0a / ffs ro 1 1
/dev/wd0h /home ffs rw,nodev,nosuid,noexec 1 2
/dev/wd0d /tmp ffs rw,nodev,nosuid,noexec 1 2
/dev/wd0g /usr ffs ro,nodev 1 2
/dev/wd0e /var ffs rw,nodev,nosuid,noexec 1 2
```

USING RM WITH THE -P OPTION

Most of the Linux distros ship with a nice file wiping utility called `shred`. Using the command `rm` with the -P option will overwrite regular files 3 times before deleting them.

Using `rm` with -P option

```
# rm -P filename
```

Or add a command alias for BASH

```
# echo "alias rm='rm -P' " >> .bash_profile
```

INCREASE MINIMUM PASSWORD LENGTH

The default minimum length for OpenBSD login passwords is 6 characters. To increase this to 10 characters, simply edit the `/etc/login.conf` file.

Edit the `/etc/login.conf` file

```
default:\
    :path=/usr/bin /bin /usr/sbin /sbin /usr/X11R6/bin /usr/local/bin:\
    :umask=022:\
    :datasize-max=512M:\
    :datasize-cur=512M:\
    :maxproc-max=128:\
    :maxproc-cur=64:\
    :openfiles-cur=128:\
    :stacksize-cur=4M:\
    :localcipher=blowfish,6:\
    :ypcipher=old:\
    :tc=auth-defaults:\
    :minpasswordlen=10:\
    :tc=auth-ftp-defaults:
```

Adding the `:minpasswordlen=10:\` line under the `default` class.

Note: login.conf does not have to be converted (cap_mkdb) to a database file, unlike FreeBSD.

If you notice any errors, please let me know.

OTHER OPENBSD TUTORIALS

- Installation - demonstration of a FTP installation
- Tips & Tricks - suggestions and configurations
- Patching - patching and kernel building
- Updating - updating with CVSup