



Published on Free Software Magazine (<http://www.freesoftwaremagazine.com>)

Protect your server with Deny Hosts

Limiting brute force based dictionary attacks

By Ken Leyba

Requiring system accessibility via the Internet poses several problems for system administrators. One problem is allowing access by authorized users with the least amount of complexity on the client computer while keeping the system and its services safe from intruders. Common services that may be provided include web server, File Transfer Protocol (FTP) server, and Secure Shell (SSH) server. Each of these services can require different methods of security to ensure only authorized users have access. This article explains how to secure Secure Shell with the DenyHosts program and the prerequisites of configuring the SSH daemon and TCP Wrappers.

Secure Shell

Secure Shell (or SSH) is used to log into a remote machine to execute commands on the remote machine. SSH can also be used to setup secure tunnels for X11 connections, for example to run remote graphical applications that reside on the server machine. SSH was originally developed to replace insecure applications like rsh, rlogin, telnet, and others.

Most major GNU/Linux distributions have the OpenSSH [1] daemon installed by default. OpenSSH is the free software implementation of the secure shell protocol. The OpenSSH server executable (or daemon) is `/usr/sbin/sshd` and the configuration file is `/etc/ssh/sshd_config`. The `sshd` configuration file contains keyword argument pairs. Two configuration options of note are `PORT` and `PermitRootLogin`. The `PORT` keyword specifies which port number that the `sshd` daemon listens on. Changing the argument of this keyword from the default 22 to another unused port number allows security through obscurity. The downside to changing this argument is that all your users will need to know the port number to use, and how to configure their `ssh` clients to connect to this port. The other keyword, `PermitRootLogin`, should have its argument changed from the default `yes` to `no`. Changing this argument will prevent the root superuser account from logging in directly via `ssh`. Instead, a normal user would have to login and use the `su` (/substitute user/) command to become root. Changing this argument will have no effect on the root user logging into the console of the server.

TCP Wrappers

TCP Wrappers [2] are used as network based Access Control List for services like FTP and SSH. The `libwrap` library, part of the TCP Wrappers package, provides support for the TCP Wrappers functionality. Most major GNU/Linux distributions have TCP Wrappers support built into services like SSH by using the `libwrap` library. This can be seen by using the `ldd` command (used to print shared library dependencies) against the `sshd` executable.

```
# ldd /usr/sbin/sshd | grep libwrap
libwrap.so.0 => /lib/libwrap.so.0 (0xb7f74000)
#
```

Protect your server with Deny Hosts

TCP Wrappers uses two files for access control, `/etc/hosts.allow` and `/etc/hosts.deny`. These files utilize keywords and arguments or options for determining access. The `/etc/hosts.allow` file is processed first, stopping at the first daemon/client match in the file and allows access. If no matches are found then `/etc/hosts.deny` is processed and the first daemon/client match denies access. If no match is found then access is allowed by default. See the man page `hosts_access(5)` for more information. This article's example system is running Debian GNU/Linux 4.0, the default installed configuration is used.

```
# man 5 hosts_access
```

Even with Secure Shell and TCP Wrappers, other basic security practices need to be in place. Services like SSH and FTP are vulnerable to dictionary attacks. Dictionary attacks are brute force attacks using multiple user name and password combinations to try and access a public system. Possibly compromised systems are used to attack services, sometimes thousands of attempts in a single day. Every entity needs good policies and procedures in place for defining user names that are not common, for example guest, admin and test. There also needs to be a good password policy using a minimum number of characters and combinations of uppercase, lowercase, numeric and special characters.

DenyHosts

From the DenyHosts [3] web site:

DenyHosts is a Python script that analyzes the `sshd` server log messages to determine what hosts are attempting to hack into your system. It also determines what user accounts are being targeted. It keeps track of the frequency of attempts from each host. Additionally, upon discovering a repeated attack host, the `/etc/hosts.deny` file is updated to prevent future break-in attempts from that host. An email report can be sent to a system admin.

Installation

Installation of DenyHosts in our Debian GNU/Linux system is straightforward. For other distributions and package managers, see the operating system documentation. Using the `apt-get` command, install the `denyhosts` package, which is part of most major distributions.

```
# apt-get install denyhosts
```

The package manager installs the main DenyHosts Python script `/usr/sbin/denyhosts` and the configuration file as `/etc/denyhosts.conf`. The first time DenyHosts is run, it will create a work directory `/var/lib/denyhosts/`. Note that this work directory is specific to Debian GNU/Linux and by default is `/usr/share/denyhosts/data/` (each GNU/Linux distribution may vary). The work directory holds several files which are the data collected by DenyHosts. The files are in human readable format and can be edited manually if needed. If not installed by default, the dependency Python language will be installed by the package manager.

In addition, a startup script is installed as `/etc/init.d/denyhosts`. This script will start DenyHosts in daemon mode and run as a process. Optionally, DenyHosts can run as a cron job. To start DenyHosts with the default configuration and the prerequisites as earlier:

```
# /etc/init.d/denyhosts start
```

Protect your server with Deny Hosts

After the work directory is created, DenyHosts processes the SSH daemon log file, in this case `/var/log/secure`, and determines which hosts have attempted to gain access to the server and failed. To stop the DenyHosts daemon use the stop argument:

```
# /etc/init.d/denyhosts stop
```

If configuration files or user maintained files, explained in the following sections, are changed the daemon can be restarted:

```
# /etc/init.d/denyhosts restart
```

Configuration

There are several configuration file parameters that determine when and which hosts are added to `/etc/hosts.deny` and how DenyHosts operates. The DenyHosts configuration file is well documented and lists each option and their parameter values. The file has four sections: required settings, optional settings, daemon specific settings and daemon synchronization.

The required settings specify the server specific file locations and threshold levels. `SECURE_LOG` defines the location of the sshd log file. `HOSTS_DENY` specifies the file and location of the TCP Wrappers access control list. `PURGE_DENY` is a time setting that will purge hosts that are older than this settings parameter, an integer followed by [m]inutes, [h]ours, [d]ays, [w]eeks or [y]ears. By default purge is disabled and `PURGE_DENY` must have a value to be enabled. In this example the purge time will be four weeks (4w), that is DenyHosts will purge host entries older than four weeks. `PURGE_THRESHOLD` defines the maximum times a host will be purged; by default the parameter is '0', in which the host can be purged or added indefinitely.

```
SECURE_LOG = /var/log/auth.log
HOSTS_DENY = /etc/hosts.deny
PURGE_DENY = 4w
PURGE_THRESHOLD = 0
BLOCK_SERVICE = sshd
```

In order for entries to the file `/etc/hosts.deny` to be effective, the service must be wrapped by `tcpd`, the access control facility of TCP Wrappers, or have `libwrap` support built in. The `tcpd` program has not been discussed because `sshd` has `libwrap` support in this example. For those distributions that do not have `libwrap` support, you will need to reference the distributions documentation [4]. The `BLOCK_SERVICE` setting lists the service(s) that should be blocked. The remaining required settings are left at their default. For more information, see the configuration file settings and comments.

The next section of the configuration file defines the optional settings which include SMTP information for sending e-mail reports. The optional section also includes settings for resetting failed login attempts. These time set values can reset failed login attempts to zero when a host exceeds the age reset value between failed attempts.

The `ADMIN_EMAIL` parameter defines the address, or addresses if delimited by commas, that receive the e-mail reports. If the local machine will send the reports then the `SMTP_HOST` is `localhost` and the `SMTP_PORT` is `25`. Customization of the "From:" address is accomplished with the `SMTP_FROM` parameter. The "Subject:" line can be modified with the `SMTP_SUBJECT` parameter, which is useful if multiple reports are being sent from different machines. If using an SMTP server that requires authentication, the `SMTP_USERNAME` and `SMTP_PASSWORD` settings are set. The remaining settings include the time definitions to reset failed login attempts to zero. These settings include `AGE_RESET_VALID`,

Protect your server with Deny Hosts

AGE_RESET_ROOT, AGE_RESET_INVALID and AGE_RESET_RESTRICTED. All of these settings can remain at their default settings. Otherwise, adjust the values as needed using the same format at the PURGE_DENY parameter.

```
ADMIN_EMAIL = joeadmin@interstellar.local, operatorjane@interstellar.local
SMTP_FROM = DenyHosts <nobody@localhost>
SMTP_SUBJECT = DenyHosts Report: develsystem.interstellar.local
```

Daemon specific settings are in the next section of the configuration file. When DenyHosts is started from a cron script, the `--purge` command line option and `PURGE_DENY` setting must be set in order to purge host entries. When started in daemon mode the `--purge` command line option is not needed, but the `PURGE_DENY` and `DAEMON_PURGE` settings are required. The `DAEMON_LOG` settings specifies the log file for DenyHosts when running in daemon mode. By default the log file is `/var/log/denyhosts`. The log file will contain the settings and parameters of the configuration file during startup and the daemon mode information. To understand what DenyHosts is doing, it is good practice to view this log file.

The `DAEMON_SLEEP` setting indicates the amount of time that the daemon will wait before scanning the file specified by the `SECURE_LOG` parameter. The default setting for this wait time is 30 seconds. The `DAEMON_PURGE` setting indicates how often DenyHosts should run the purge mechanism to remove entries in the `HOSTS_DENY` file. The default setting is 1 hour.

The last section in the configuration file is for daemon synchronization. DenyHosts has the ability to download and upload, from a central repository, host data from servers around the world. This feature proactively populates the `HOST_DENY` file so no login attempts can be made from known intruding IP addresses. The daemon synchronization mode is disabled by default.

Optional Files

There are two optional user maintained files for DenyHosts operation in the work directory. The file `restricted-usernames` contains user names, one per line, that are restricted. Restricted means exceeding the value of `DENY_THRESHOLD_RESTRICTED` in the configuration file, which is 1 by default. There are two helper scripts that are located in `/usr/share/doc/denyhosts/examples/scripts/`. The script `restricted_from_passwd.py` scans the `/etc/passwd` file (the list of user accounts), and displays users that have restricted shells. For example the `ftp` user account has `/bin/false` for the default shell. The `ftp` account should be added to `restricted-usernames`. The script `restricted_from_invalid.py` will parse the `users-invalid` file in the work directory and display the most frequently attacked user names. Both scripts require you to manually add the output to the `restricted-usernames` file.

The `allowed-hosts` file is optional and is maintained by the administrator: it allows the specification of hosts that are allowed to login. If, for example, there are legitimate hosts that a user mistyped a user name or password, it would be more work to reset the user password and purge the host from the deny file, or manually adding it to the `/etc/hosts.allow` file. Remember, `sshd` will first process the `/etc/hosts.allow` file, and if a match is made login is permitted. The format of the allowed hosts file permits IP addresses, wild-cards and ranges. If there is a subnet of lab machines, the setting could be `10.70.0.*`. Another example is a range of IP addresses such as `10.20.0.[100-175]`, or a single IP address, `10.1.0.236`. The two optional files are read at startup; so if any changes are made, DenyHosts must be restarted (see starting and restarting the daemon earlier in this article).

Conclusion

DenyHosts is a simple method of limiting the effects of brute force attacks on a GNU/Linux server that has services open on the Internet. The default configuration is sufficient for normal operation. However, it can be customized and it has optional features not enabled by default. With the prerequisite service and libwrap support, DenyHosts adds an extra layer of security in the security process.

Bibliography

[1] [Open SSH](#) [2] [TCP Wrappers](#) [3] [DenyHosts](#) [4] [ITSO TCP Wrappers](#)

Biography

[Ken Leyba](#) (/user/5507" title="View user profile.): Ken has been working in the IT field since the early 80's, first as a hardware tech whose oscilloscope was always by his side, and currently as a system administrator. Supporting both Windows and Linux, Windows keeps him consistently busy while Linux keeps his job fun.

Copyright information

This article is made available under the "Attribution-NonCommercial" Creative Commons License 3.0 available from <http://creativecommons.org/licenses/by-nc/3.0/>.

Source URL:

http://www.freesoftwaremagazine.com/articles/protect_your_server_with_deny_host
