How to secure your Linux cloud server

Janne Ruostemaa (https://upcloud.com/blog/author/raiou/) Staff

Integrations (https://upcloud.com/tutorials-category/integrations/) 0

One of the first things you should do after deploying a new cloud server is to make sure it will stay secure. Linux offers a multitude of options to help prevent unauthorized access and harden your system. In this how-to guide, you can find some commonly recommended steps in order to protect your cloud server.

Encrypt communications

When connecting to your cloud server all traffic will pass through the public network, that anyone could be eavesdropping on, unless you take measures to secure your communication. Avoid using any unencrypted transfer protocols such as Telnet and FTP, or anything that would send passwords or other sensitive information as plain text. Instead, you should use SSH (Secure Shell), SCP (Secure Copy), SFTP (SSH File Transfer Protocol) or rsync for all your remote control and file transfer needs.

The **SSH** protocol offers a secure encrypted channel over the public network to allow remote login and other network services to operate securely. The most commonly used implementation of this protocol is OpenSSH which is included in most Unix based operating systems like the majority of Linux distributions and OS X, in a Windows environment the PuTTY SSH client is a popular alternative. Check out our article for **Connecting to Your Server (https://www.upcloud.com/support/connecting-to-your-server/)** to learn more.

Secure Copy or SCP is a built-in feature of OpenSSH which allows simple file transfer over encrypted network connection. The SCP uses SSH for data transfer and provides the same authentication and level of security as SSH. Below are two examples of a single file copy to and from a remote server.

```
# Copy the file "foo.txt" from the local host to a remote host
scp foo.txt <username>@<remotehost>:/some/remote/directory
# Copy the file "foo.txt" from a remote host to the local host
scp <username>@<remotehost>:foo.txt /some/local/directory
```

SFTP is an other command line utility included in OpenSSH and should be installed on most Unix systems. Like SCP, SFTP uses SSH to securely transfer files over a network. Windows users can get the same functionality using **WinSCP (https://winscp.net/eng/index.php)** (Windows Secure Copy) which like its name suggests implements SCP and also SFTP functionality.

rsync is another utility commonly found on Unix systems. It offers file transfer over encrypted channels to keep the copies of a file on two computers synchronised. The program uses SSH to make the initial connection between the two systems and then invokes rsync on the remote host to determine which parts of the file being synced need to be copied over.

User account security policies

After logging in to your newly deployed cloud server for the first time, creating a new user account for yourself and enabling sudo access control, are some important tasks to start with. Sudo, which stands for “superuser do,” allows you to perform actions that would otherwise require the root account. This lets you avoid logging in as root on daily basis, instead use sudo privileges to execute root level commands when required.

Using sudo is considered good practice for security, and it’s usually installed in most Linux distributions by default. To get most out of what sudo offers, and to set up a secure user access, follow our guide for **Managing Linux User Account Security (https://www.upcloud.com/support/managing-linux-user-account-security/)**.

Monitoring login authentication

The reality in today’s internet is that your server security will be tested by malicious parties, sooner rather than later, hoping to find a poorly secured entrance. If your server has been running for even a day, you’ve most likely already had failed login attempts originating from IP addresses other than your own. Majority of Linux distributions keep logs for authentication from the moment they are booted up for the first time. Different systems might store the logs under different names, for example with Ubuntu and other Debian based servers you can view these logs using the following command

```
cat /var/log/auth.log | grep 'ssh.*Invalid'
```

On CentOS and other Red Hat variants use this instead

```
cat /var/log/secure | grep 'ssh.*Invalid'
```

The output will list dates and times of when invalid login attempts occurred, which user accounts were used, and from which IP addresses the connections came from. Even a large number of failed logins is nothing to be frightened about, though it shows how common practice this kind of behaviour is.

In contrast, check your successful log in times using the command below.

```
last
```

This will print the latest few login times, dates and the IP addresses the connections originated from. If you’ve recently used the web Console at your UpCloud Control Panel, you’ll see those login times marked with *tty1*, other remote control connections such as SSH show *pts/0* instead,

where the number is a connection identifier when you have had multiple SSH sessions open at the same time.

While your cloud server should still be secure thanks to the Linux default security implementations, you should not rest easy and just hope it stays that way. There are some powerful tools available for reducing the failed login attempts and protecting from simple password brute forcing.

Fail2ban is one such intrusion prevention framework, which works together with a packet-control system or firewall installed on your server. It is commonly used to block connection attempts after a certain number of failed tries, effectively giving the user a time-out before their are allowed to try again. Read our guide to installing Fail2ban on Linux cloud servers with **CentOS (https://www.upcloud.com/support/installing-fail2ban-on-centos-7/)**, **Debian (https://www.upcloud.com/support/installing-fail2ban-on-debian-8-0/)** or **Ubuntu (https://www.upcloud.com/support/installing-fail2ban-on-ubuntu-14-04/)** to learn more.

Use SSH-keys instead of passwords

Passwords are the default way to authenticate to almost everything, and while secure to a point they can often be guessed using brute-forcing or dictionary lists by simply trying multiple variations of common passwords. Secure and difficult to guess passwords can then again get troublesome to remember and are easily mistyped.

Another option is to use SSH-keys for authentication by generating a pair of long, practically impossible to break, key codes. From these keys, a so-called public key can be safely passed on to your server, while keeping the private key securely on your own computer.

The public key can only be used to identify the user who has the private part of the pair.

The private key must be kept safe, ensuring that only you have access to it.

Check out our guide to **Using SSH-keys For Authentication (https://www.upcloud.com/support/using-ssh-keys-for-authentication/)** to learn how to implement it on your Linux cloud server.

Setup a firewall

Common solutions for any networked computer security is to set limitations to which connections are allowed. This can be done by using a firewall, a network security system, that monitors and controls the incoming and outgoing network traffic based on predetermined security rules.

The UpCloud control panel offers an easy to configure firewall that acts as a first line defence to secure your cloud server. The UpCloud firewall works server specifically, but you can copy firewall settings between your servers. You also have the option to configure the firewall using one of the premade setups available at the firewall rules settings. The premade rules are a simple starting point for further customization. You can read more about the **UpCloud Firewall (https://www.upcloud.com/support/firewall/)** at its own article.

An other option on a Linux server is to use the built-in solution called iptables, which is included in most distributions. On CentOS and other Red Hat variants iptables often comes with some preconfigured rules, while Ubuntu and Debian servers don't implement any restrictions by default. To learn more about iptables, check out our introductory guide to configuring iptables on your Linux server of either **CentOS (https://www.upcloud.com/support/configuring-iptables-on-centos-6-5)**, **Debian (https://www.upcloud.com/support/configuring-iptables-on-debian-8-0/)** or **Ubuntu (https://www.upcloud.com/support/configuring-iptables-on-ubuntu-14-04/)**.

Update your system

Make sure to regularly check for updates on your Linux server. New vulnerabilities are found from time to time and software companies release fixes to keep your system up to date and secure.

Ubuntu server users can do this with the following

```
sudo apt-get update && sudo apt-get upgrade
```

This is the simple method to update the packages already installed on your server, but it's not allowed to add or remove packages even if they've become obsolete. You can also use the advanced "smart upgrade" with

```
sudo apt-get update && sudo apt-get dist-upgrade
```

The command checks package relations and aims to upgrade the most important packages at the expense of less important ones if necessary.

Debian also includes the *apt-get* but recommends using *aptitude* instead. Enter the following command to upgrade your system.

```
sudo aptitude update && sudo aptitude full-upgrade
```

CentOS servers can be updated with a simple command shown below

```
sudo yum update
```

Yum does include the upgrade command as well, but it might also remove some packages it deems obsolete even if you were still using them, so the update command is generally safer in most cases.

Remember to update other software outside the package manager as well, for example, if you use content management software (CMS) like WordPress or Joomla. Make sure to keep your platform up to date and remove any unnecessary plugins, as outdated web apps are often targeted by attackers.

Minimize vulnerabilities

An important part of securing a cloud server is to not leave open any unnecessary network services that are listening for incoming connections. A newly deployed Linux system usually only has SSH port 22 open. You can test your own server by scanning for open ports using network tool named *Nmap*. The program isn't included in many distributions by default, but you can install it simply with one of the following commands on Ubuntu and Debian or CentOS respectively.

```
sudo apt-get install nmap
```

```
sudo yum install nmap
```

With the program installed, try running a test scan on the localhost using

```
nmap -v -sT localhost
```

The printout will list port numbers and services associated with them that are currently open for local connections. Next, use the same command, but scan for your server's public IP instead. This can be performed from any computer with internet access and Nmap installed using the following

```
nmap -v -sT <public IP>
```

If you had more than just SSH appear in the localhost scan, they most likely do not show up in the public IP list. One example of such services is the SMTP email server included in Debian.

Any other services open to the public network should be paid close attention to. Make sure you know what services you have running and how secure their connection methods are. Disable any services you know you don't need.

Scan for malware regularly

Linux systems are generally less likely to be infected by malicious software as open source scrutiny and diverse end-user configurations make finding and exploiting vulnerabilities difficult. Your primary defence should be preventative effort to stop unauthorized access, but it can't be your only security measure. While you might not think anything on your system is out of the ordinary, harmful program could be running unnoticed for a long time before causing alarming traffic or system damage. Therefore it's important that you scan your cloud server for malware regularly, just to make sure it hasn't been infected.

Aside from the variety of malware, another type of malicious software to look out for are rootkits, which are a collection of programs designed to gain access to a computer or parts of its OS that are usually restricted while at the same time hiding their presence. The rootkits are often used by an attacker after gaining root access on their target system. Even though rootkits try to mask their existence there are tools made specifically for detecting known rootkit variants.

Read our started guide for scanning malware on your Linux server running either **CentOS (https://www.upcloud.com/support/scanning-centos-server-for-malware/)**, **Debian (https://www.upcloud.com/support/scanning-debian-8-0-server-for-malware/)** or **Ubuntu (https://www.upcloud.com/support/scanning-ubuntu-14-04-server-for-malware/)**.

Implement Intrusion Detection System

Checking your system with malware scanners and the likes are still mostly scheduled tasks performed every now and then. This gives any malware time between scans to go about their business unnoticed possibly even for an extended period of time. The solution for the downtime between malware sweeps is to set up an intrusion detection system (IDS), that constantly keeps an eye on your cloud server and its network traffic.

Snort (https://www.snort.org/) is a popular choice for network based intrusion detection system (NIDS), it's open source, actively developed, and light weight enough to be installed on even the smallest of cloud servers. Check out our guides for installing Snort on **CentOS (https://www.upcloud.com/support/installing-snort-on-centos/)**, **Debian (https://www.upcloud.com/support/installing-snort-on-debian/)** or **Ubuntu (https://www.upcloud.com/support/installing-snort-on-ubuntu/)**.

The other type of intrusion detection system is host base (HIDS), which analyses system behaviour and configuration status to detect potential security breaches, compromises, modifications to critical system files, common rootkits, and malicious processes.

OSSEC (http://www.ossec.net/) is a good example of an open source HIDS that performs log analysis, integrity checking, rootkit detection, real-time alerting and active response. OSSEC is available for most operating systems including most common Linux distributions. It's intended to be configured to on server-client basis, where very light clients are installed on the critical systems, that then send their reports to the OSSEC server for analysis. This is ideal for users with multiple cloud servers for centralized security monitoring.

Janne Ruostemaa (https://upcloud.com/blog/author/raiou/)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Click here to leave a comment

Name* Email* Website

Post Comment

In the capital city of Finland, you will find our headquarters, and our first data (https://upcloud.com) center. This is where we handle most of our development and innovation.

Email
General hello@upcloud.com (mailto:hello@upcloud.com)
Sales sales@upcloud.com (mailto:sales@upcloud.com)
Support support@upcloud.com (mailto:support@upcloud.com)

Encrypt communications

User account security policies

Monitoring login authentication

How finalwebsites builds lasting partnerships on mutual values

Use SSH-keys instead of passwords

Setup a firewall

Update your system

Minimize vulnerabilities

Scan for malware regularly

Implement Intrusion Detection System