# Using SSH with OneID

## Introduction

SSH is commonly used to provide secured access to servers. Your access is secured using a password or a private/public keypair. Now, OneID can be used to supplement SSH security by adding multi-factor authentication and remote access control.

## Benefits

In addition to traditional MFA, OneID also allows you to instantly change the level of authorization. The user can choose to lock his account, or change the assurance level of his account with a quick swipe on your mobile device.

If the mobile device is stolen, it can be disabled by the user. It will also auto-disable after 5 PIN attempts.

If the user has multiple OneID Remotes, any remote can be used to authorize the request. If a user adds a Remote, that Remote automatically works with all the SSH servers without any any additional reconfiguration or provisioning effort. So a new device can be added in seconds!

This is more secure than using RSA SecurID because there is no chance of a MITM attack (the OneID Remote app shows you independently of the main device what you are signing so a compromise on the main device doesn’t impact the display on the Remote and vice versa) and because the approval cannot be re-used (a SecurID token can be re-used or repurposed without your consent).

It is more convenient than SecurID because there is no transcription of a long code from one device to another. If a PIN is required, it is four digits.

OneID is also more secure than most any other 2FA because there is no single point of compromise. For example, say you use a virtual machine which has your private key and then you use VPN to log into a server, perhaps with a RSA SecurID token. If there is malware on your machine, all bets are off; there is a single point of compromise. With OneID, the attacker has to compromise your machine and your phone (and possibly also learn your PIN code). That is extremely difficult to do.

Also, suppose your private key is compromised by an attacker. Think of all the systems you have to change your public key on. With OneID this is not a problem because there are 3 public keys so there is never any need to re-issue public keys to any relying party because an attacker can’t get all three public keys.

Because OneID for SSH replaces the shell, the issues of PAM authentication do not impact your other security measures in place. OneID will work with your existing environment.

There is also a OneID PAM module that is available.

## Installation

OneID for SSH is easily installed on any Linux based environment. OneID provides packages for most of the common Linux distributions. After a package installation, adding OneID requires a small modification of the “authorized\_keys” file in the .ssh directory for each user that requires OneID access.

|  |
| --- |
| ~/.ssh/authorized\_keys |
| Before the key entry: ssh-rsa AAAAB3NzaC1yc2EAAAAD……add:command="/usr/bin/oneid"The entry should look like:command="/usr/bin/oneid" ssh-rsa AAAAB3NzaC1yc2EAAAAD…… |

The above will enable a user to set whatever level of assurance (LoA) he wants.

You can add the following arguments to the command line:

|  |  |
| --- | --- |
| -o | Require an out of band (OOB) authorization on every login |
| -p | Require an out of band + PIN authorization on every login |

So to force an OOB (or better):

|  |
| --- |
| command="/usr/bin/oneid -o" ssh-rsa AAAAB3NzaC1yc2EAAAAD…… |

This will ensure a minimum LoA before allowing a user to login. OneID will always use the most restrictive mode requested. If the User has a higher LoA set on his account than the LoA set for the server, that level will be used instead.

## Usage

After installation, the user should just sign up for a OneID account, install the mobile application for their platform, and SSH into the machine normally. They will be asked to add their device. After the initial enrollment, the user will be able to control access to their account from their OneID mobile application.

For a full demonstration of OneID, see the video at:

<http://xfer.bigforge.com/SSHOneID.mp4>