A comprehensive guide to fixing slow SSH logins

The debug text that brought you here

Most of you are probably getting here just from frustratedly googling “slow ssh login”. Those of you who got a little froggier and tried doing an ssh -vv to get lots of debug output saw things hanging at **debug1: SSH2_MSG_SERVICE_ACCEPT received**, likely for long enough that you assumed the entire process was hung and ctrl-C’d out. If you’re patient enough, the process will generally *eventually* continue after the **debug1: SSH2_MSG_SERVICE_ACCEPT received** line, but it may take 30 seconds. Or even five minutes.

You might also have enabled debug logging on the server, and discovered that your hang occurs immediately after **debug1: KEX done [preauth]**
and before **debug1: userauth-request for user** in **/var/log/auth.log**.

I feel your frustration, dear reader. I have solved this problem after hours of screeching head-desking probably ten times over the years. There are a few fixes for this, with the most common – DNS – tending to drown out the rest. Which is why I keep screeching in frustration every few years; I *remember* the dreaded **debug1: SSH2_MSG_SERVICE_ACCEPT received** hang is something I’ve solved before, but I can only remember *some* of the fixes I’ve needed.

Anyway, here are all the fixes I’ve needed to deploy over the years, collected in one convenient place where I can find them again.

It’s usually DNS.

The most common cause of slow SSH login authentications is DNS. To fix this one, go to the SSH **server**, edit **/etc/ssh/sshd_config**, and set **UseDNS no**. You’ll need to restart the

service after changing sshd_config: /etc/init.d/ssh restart, systemctl restart ssh, etc as appropriate.

If it's not DNS, it's Avahi.

The next most common cause – which is *devilishly* difficult to find reference to online, and I hope this helps – is the never-to-be-sufficiently damned avahi daemon. To fix this one, go to the SSH **client**, edit **/etc/nsswitch.conf**, and change this line:

```
hosts:          files mdns4_minimal [NOTFOUND=return] dns
```

to:

```
hosts:          files dns
```

In *theory* maybe something might stop working without that mdns4_minimal option? But I haven't got the foggiest notion what that might be, because nothing ever seems broken for me after disabling it. No services need restarting after making this change, which again, must be made *on the client*.

You might *think* this isn't your problem. Maybe your slow logins only happen when SSHing to one particular server, even one particular server on your local network, even one particular server on your local network which has UseDNS no *and* which you don't *need* any DNS resolution to connect to in the first place. But yeah, it can still be this avahi crap. Yay.

When it's not Avahi… it's PAM.

This is another one that's really, really difficult to find references to online. Optional PAM modules can really screw you here. In my experience, you can't get away with simply disabling PAM login in /etc/ssh/sshd_config – if you do, you won't be able to log in at all.

What you need to do is go to the SSH **server**, edit **/etc/pam.d/common-session** and comment out the optional module that's causing you grief. In the past, that was **pam_ck_connector.so**. More recently, in Ubuntu 16.04, the culprit that bit me hard was **pam_systemd.so**. Get in there and comment that bugger out. No restarts required.

```
#session optional pam_systemd.so
```

GSSAPI, and ChallengeResponse.

I've seen a few seconds added to a pokey login from GSSAPIAuthentication, whatever that is. I feel slightly embarrassed about not knowing, but seriously, I have no clue. Ditto for ChallengeResponseAuthentication. All I can tell you is that neither cover standard interactive passwords, or standard public/private keypair authentication (the keys you keep in ~/ssh/authorized_keys).

If you aren't using them either, then disable them. If you're not using Active Directory authentication, might as well go ahead and nuke Kerberos while you're at it. Make these changes on the **server** in **/etc/ssh/sshd_config**, and restart the service.

```
ChallengeResponseAuthentication no
KerberosAuthentication no
GSSAPIAuthentication no
```

Host-based Authentication.

If you're actually using this, don't disable it. But let's get real with each other: you're not using it. I mean, I'm sure *somebody* out there is. But it's almost certainly not you. Get rid of it. This is also on the **server** in **/etc/ssh/sshd_config**, and also will require a service restart.

```
# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
```

Need frequent connections? Consider control sockets.

If you need to do *repetitive* ssh'ing into another host, you can speed up the repeated ssh commands enormously with the use of control sockets. The first time you SSH into the

host, you establish a socket. After that, the socket obviates the need for re-authentication.

```
ssh -M -S /path/to/socket -o ControlPersist=5m remotehost exit
```

This creates a new SSH socket at **/path/to/socket** which will survive for the next 5 minutes, after which it'll automatically expire. The **exit** at the end just causes the ssh connection you establish to immediately terminate (otherwise you'd have a perfectly normal ssh session going to **remotehost**).

After creating the control socket, you utilize it with **-S /path/to/socket** in any normal ssh command to **remotehost**.

The improvement in execution time for commands using that socket is *delicious*.

```
me@banshee:/~$ ssh -M -S /tmp/demo -o ControlPersist=5m eohippus exit

me@banshee:/~$ time ssh eohippus exit
real 0m0.660s
user 0m0.052s
sys 0m0.016s

me@banshee:/~$ time ssh -S /tmp/demo eohippus exit
real 0m0.050s
user 0m0.005s
sys 0m0.010s
```

Yep… from 660ms to *50*ms. SSH control sockets are pretty awesome.

PUBLISHED BY

Jim Salter

Mercenary sysadmin, open source advocate, and frotzer of the jim-jam.

View all posts by Jim Salter →

July 1, 2017 Jim Salter Debian, FreeBSD, Linux, Open Source, Ubuntu