Johannes 'heipei' Gilger

SSH Agent Forwarding considered harmful

26 Feb 2015

tl;dr: Don't use SSH `ForwardAgent`, it's stupid and insecure. Use `ProxyCommand` instead.

- Hackernews thread on this post
- r/netsec thread on this post

Introduction

Since I recently saw a post on Reddit netsec which sadly omitted what to use instead of ssh-agent, I felt it was time to write yet another discussion about the perils of what is a useless workflow at best and a dangerous habit at worst. I'll show a simpler, more secure and more powerful alternative in the form of SSH `ProxyCommand`.

The problem with SSH Agent Forwarding

SSH Agent Forwarding can be enabled by calling `ssh -A` or by setting the `AgentForward` flag in your config. It is meant as an easy way to connect to a host A with your SSH key and from there connect to another host B with that same key. This obviously is only needed if you cannot connect to host B directly from your workstation.

The problem is that while you're connected to host A, a forwarding socket will be set up so that the SSH client on host A can connect to the ssh-agent on your workstation to perform authentication on its behalf. This means that anyone with sufficient permission on host A will be able to use that socket to connect to and use your local ssh-agent. It could be the root user or anyone else who managed to compromise host A. The result is that the user would be able to impersonate you to any host as long as you're connected to host A.

You might say that host A only belongs to yourself, there is no other user on it, even less so someone with root access. But then again: Why take the chance? The probability of encountering a compromised machine increases with the number of hosts you connect to, and I know most people consider their workstation their most secure host.

You might also say that the window of compromise is small since it is only open while you're connected to host A. Again: Why take the risk? This is like having unprotected sex only for a short amount of time. And sometimes you *do* leave that SSH session open while you're out for lunch or something is processing in the background.

Then there is the last line of reasoning: Only using a dedicated key for forwarding to host A. This argument also becomes void if you consider that someone who originally only had access to host A can now also access host B. And furthermore, keeping track of which keys have been added to your local ssh-agent is a tedious task. SSH is very promiscuous when it comes to using SSH keys, and once you make use of another key it will happily add that to your current agent session.

Lastly, SSH Agent Forwarding only works for interactive sessions which gives it that very ghetto feeling of missing out on basically all the great features SSH has to offer. Ever copied a file to host A and then to host B? You're doing it wrong!

The alternative: ProxyCommand to the rescue

If you're still reading this, chances are that you're not aware of the awesome toolbox that is SSH (or at least OpenSSH, let's be honest).

OpenSSH has an option called `ProxyCommand`. It works by specifying a ProxyCommand to connect to host B. The config would look like this for our simple example:

```
1 Host hosta
2       User userfoo
3       Hostname 123.123.123.123
4
5 Host hostb
6       User userbar
7       Hostname 192.168.1.1
8       Port 2222
9       ProxyCommand ssh -q -W %h:%p hosta
```

This config tells your local SSH client to connect to host A via a direct connection. Now, if you type `ssh hostb` what it will do is connect to host A first and then forward your SSH client via the `ProxyCommand` to the host and port specified by the `-W` parameter, in this case I used `192.168.1.1` to underline that this is an *internal* host. Your local SSH client will then authenticate to host B as if you were connected to it directly.

Another observation here is that you don't need to memorize anything about the host, neither it's IP or obscure hostname, port or username when you can simply alias it with the `Host` line.

Legitimate uses of ForwardAgent

I've been trying to come up with legitimate reasons for using SSH agent forwarding but failed to do so. I've only recently discovered one case where agent forwarding is not only useful but also secure: Namespaces.

With Linux namespaces (filesystem, process, network namespaces) processes can be put into their own namespaces on the same machine. This is something that makes Docker work, but it can also be used standalone to create a number of different networking setups. If you run `sshd` in your separate network namespace, you can then ssh into that namespace to work from there and connect to the host only reachable from the namespace. Since you're connecting to your own machine, forwarding your ssh-agent does not increase the attack surface: It's still your machine.

What would the alternative look like? Well, you can ssh into your namespace, spawn a separate ssh-agent there and then add all your keys to that, typing each passphrase again. Or you could create a `ProxyCommand` to use your namespace as an intermediate host, either globally or per-host. But then you'd have to toggle that on and off if you ever want to connect to the host directly as opposed to through the namespace. Here, a simple `ssh -A` clearly wins in terms of convenience.

Conclusion

By now you should have some idea about why not to use SSH Agent Forwarding. If you do have a legitimate use-case which I haven't covered here, feel free to leave it in the comments. I'll follow up this post with a general breakdown of some of the awesome features that SSH has to offer. The `ProxyCommand` will play a key role, but there are lot's of other goodies hidden inside OpenSSH.

Updates - March 4 2015

After I posted this post on r/netsec it quickly got quite few replies and some interesting discussions developed. As I mentioned in the post I was open for suggestions where ssh-agent might be hard to replace and how to use it more securely.

One thing a number of people mentioned was using `ssh-agent -c` which will show a confirmation window each time some program wants to use the agent to authenticate somewhere. This is certainly a cool feature, though it will also trigger for local users (i.e. you) and thus might be annoying.

As for reasons to use SSH Agent Forwarding in the first place, there was one argument which I can relate to: Performance. If the hosts (jump-host and target host) are behind a slow uplink and you frequently have to get data from one host to the other it really sucks having to copy everything to your local workstation and then back up. Unless a dedicated key which resides on either of the hosts does the trick you're stuck with Agent Forwarding.

The other reason I heard was "convenience". As I'll show with my next post on SSH, Agent Forwarding is a lot less convenient in a number of ways.

A big problem about Agent Forwarding I forgot to mention is this by the way: If an attacker compromises Host A he can not only use your forwarded agent when you're connected

but he is actually able to eavesdrop on your ongoing session. So even if you had an additional layer of security on Host B (password, time-based token), the attacker could read and modify everything about your session. ProxyCommand will prevent this from happening since the SSH session to Host B terminates on your workstation!

References

- Hackernews thread on this post
- r/netsec thread on this post
- The perils of using an ssh-agent - Post which fails to mention ProxyCommand
- Good post on ssh-agent and ProxyCommand
- Putting vpnc into a Linux network namespace
- Linux network namespaces in operation

Comments

markhellewell • 4 years ago

Being able to interact with GitHub/etc. repositories while on remote servers, without having to manage private keys on those servers, is quite useful. Use HashKnownHosts to hinder discovery of target hosts for which agent auth might work. You can very easily use gpg-agent as your ssh-agent, and can switch to use of whitelisted private keys to reduce agent promiscuity.

5 • Reply • Share ›

heipei Mod markhellewell • 4 years ago

I never said that it could not be useful. But at the same time it can be really dangerous. Sure, if you have a dedicated key for only fetching from Github, and then only forward that key that would certainly be better.

1 • Reply • Share ›

Neil Ramsay • 4 years ago

This is a great write up, and definitely something I will start to use in the future. Do you know if there is a way to do this in Windows with PuTTY? The majority of people I work with are Windows users, and need clear and easy connection instructions.
Cheers.

1 • Reply • Share ›

heipei Mod Neil Ramsay • 4 years ago

Sorry, no idea about PutTTY. Have you googled it yet?

• Reply • Share ›

Yitz Gale Neil Ramsay • 3 years ago

Yes. It's on the Connection/Proxy tab in the New Session dialog. Here's how to set it up:

Let's say you want to connect via server A to server B. Enter the Host/Port data for server B, as seen from server A, on the main Session tab. Put your username on server B in Auto-login on the Connection/Data tab. On the Connection/Proxy tab, set Proxy type: Local, Proxy hostname/Port for server A. For the local proxy command, use plink, as follows (replace "usernameA" with your username on server A):

Johannes Gilger - @heipei

Atom Feed