Published: Jun 26, 2020 Version: 1.0

FSA-2020-2 Targeted MitM Attacks Using Information Leakage in SSH Clients

Because of an information leak in the initial key exchange message of the SSH protocol, an attacker can detect if an SSH client using the default configuration stores a host key for the target server. Thus, an attacker can focus a man-in-the-middle attack on clients that connect to a server for the first time and avoid clients that would show a warning because of a changed host key. Clients that connect to a server for the first time, ask the user to confirm the fingerprint of the host key. Users that compare the shown fingerprint by a known value are safe. However, many users rely on trust on first use and accept host keys without verification.

The reason of the information leak is that the host key algorithms in the client key exchange message are ordered by preference and algorithms of stored host keys are preferred. SSH clients have to decide between leaking information in the key exchange message and triggering host key warnings by requesting unknown host keys. Two popular SSH clients OpenSSH client and PuTTY leak information in the default configuration and are thus susceptible to the aforementioned attack. We also checked the behavior of Dropbear SSH client, which turned out not to be vulnerable. However, this means that Dropbear SSH might trigger host key warnings even though the client stores a host key.

1 OpenSSH client

FZI-ID	FZI-2020-3
CVE	CVE-2020-14145
Manufacturer	OpenBSD Foundation
Product	OpenSSH client
Affected Version	5.7-8.3
Type	CWE-203 - Observable Discrepancy
Date Found	20.03.2020
Discovered By	Timon Hackenjos
Patch Status	Won't fix
Patch Version	-
CVSS Score	4.7
CVSS String	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:L/I:L/A:N

We investigated the vulnerability in OpenSSH 8.2 and 8.3 portable using a proof of concept tool that compares the host key algorithm list in key exchange messages. Tests showed that the host key algorithm list differs from the default list for all algorithm types that are not certificate-based, namely ECDSA, Ed25519 and RSA. This means that in the default configuration an attacker can identify users connecting to a server for the first time without false positives.

```
$ ./ssh-detect.py -r OpenSSH-8.2-init.pcap
Client Key Exchange Init 192.168.102.1:33580 -> 192.168.102.134:22

Host Key Algorithms:
ecdsa-sha2-nistp256-cert-v01@openssh.com
ecdsa-sha2-nistp384-cert-v01@openssh.com
ecdsa-sha2-nistp521-cert-v01@openssh.com
sk-ecdsa-sha2-nistp256-cert-v01@openssh.com
ssh-ed25519-cert-v01@openssh.com
sk-ssh-ed25519-cert-v01@openssh.com
rsa-sha2-512-cert-v01@openssh.com
rsa-sha2-256-cert-v01@openssh.com
ssh-rsa-cert-v01@openssh.com
ecdsa-sha2-nistp256
ecdsa-sha2-nistp384
ecdsa-sha2-nistp521
sk-ecdsa-sha2-nistp256@openssh.com
ssh-ed25519
sk-ssh-ed25519@openssh.com
rsa-sha2-512
rsa-sha2-256
ssh-rsa

Client Version: SSH-2.0-OpenSSH_8.2 (known)
Default algorithm list detected! Client doesn't store host key.
```

First connection to a server using OpenSSH client in version 8.2.

```
$ ./ssh-detect.py -r OpenSSH-8.2-known.pcap
Client Key Exchange Init 192.168.102.1:33662 -> 192.168.102.134:22

Host Key Algorithms:
ecdsa-sha2-nistp256-cert-v01@openssh.com
ecdsa-sha2-nistp384-cert-v01@openssh.com
ecdsa-sha2-nistp521-cert-v01@openssh.com
ecdsa-sha2-nistp256
ecdsa-sha2-nistp384
ecdsa-sha2-nistp521
sk-ecdsa-sha2-nistp256-cert-v01@openssh.com
ssh-ed25519-cert-v01@openssh.com
sk-ssh-ed25519-cert-v01@openssh.com
rsa-sha2-512-cert-v01@openssh.com
rsa-sha2-256-cert-v01@openssh.com
ssh-rsa-cert-v01@openssh.com
sk-ecdsa-sha2-nistp256@openssh.com
ssh-ed25519
sk-ssh-ed25519@openssh.com
rsa-sha2-512
rsa-sha2-256
ssh-rsa

Client Version: SSH-2.0-OpenSSH_8.2 (known)
Non-default algorithm list
```

Connection using OpenSSH client in version 8.2 with known ecdsa key.

2 PuTTY

FZI-ID	FZI-2020-5
CVE	CVE-2020-14002
Manufacturer	Simon Tatham
Product	PuTTY
Affected Version	0.68-0.73
Type	CWE-203 - Observable Discrepancy
Date Found	04.05.2020
Discovered By	Timon Hackenjos
Patch Status	Partially (disabled by default)
Patch Version	0.74
CVSS Score	4.7
CVSS String	CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:L/I:L/A:N

Using our proof of concept tool, we examined the behavior of PuTTY and found that it is vulnerable too. However, the host key algorithm list only differs if a host key of non-default type is stored on the client. Currently ssh-ed25519 is the default algorithm and thus users that store an ssh-ed25519 key and users that connect for the first time are indistinguishable for an attacker. Therefore, there is always a risk of triggering a warning during a man-in-the-middle attack.

```
$ ./ssh-detect.py -r putty-0.73-init.pcap
Client Key Exchange Init 192.168.249.130:49675 -> 192.168.249.131:22

Host Key Algorithms:
ssh-ed25519
ecdsa-sha2-nistp256
ecdsa-sha2-nistp384
ecdsa-sha2-nistp521
ssh-rsa
ssh-dss

Client Version: SSH-2.0-PuTTY_Release_0.73 (known)
Default algorithm list detected! Client doesn't store host key.
```

First connection to a server using PuTTY in version 0.73.

```
$ ./ssh-detect.py -r putty-0.73-known.pcap
Client Key Exchange Init 192.168.249.130:49676 -> 192.168.249.131:22

Host Key Algorithms:
ecdsa-sha2-nistp256
ssh-ed25519
ecdsa-sha2-nistp384
ecdsa-sha2-nistp521
ssh-rsa
ssh-dss

Client Version: SSH-2.0-PuTTY_Release_0.73 (known)
Non-default algorithm list
```

Connection using PuTTY in version 0.73 with known ECDSA key.

3 Mitigation

The vulnerability can be fixed by using the default host key algorithm list regardless of stored host keys. However, this fix introduces another problem. The client will prefer new host key algorithms even if another host key for the target server is already stored on the client. The consequence is that the client shows a warning that the host key of the server changed. The warning itself is not dangerous, provided the user compares the fingerprint of the new host key or forces the use of the existing host key. However, triggering lots of warnings might tempt a user to accept warnings without verification and thus increase the susceptibility for man-in-the-middle attacks instead of reducing it. On the protocol level, both problems could be solved by requiring the server to commit itself to its host keys before the client chooses an algorithm.

Users can mitigate the vulnerability by always comparing the fingerprint of the host key with a known good value. However, there are situations where it is not possible to get the fingerprint of the host key in advance.

3.1 OpenSSH

The developers of OpenSSH are not planning to change the behavior of OpenSSH regarding this issue, because of the aforementioned drawbacks. However, there are some configuration options that can be used to mitigate the vulnerability. OpenSSH provides alternative ways to validate host keys, namely SSHFP records and host certificates. These should be used if DNSSEC or a PKI are available.

Otherwise, by setting the option *HostKeyAlgorithms* explicitly (without '+', '-' or '^') the adaptive order of the host key algorithms can be disabled including the aforementioned drawback. OpenSSH offers the option *UpdateHostKeys* that can reduce the impact of this change. By enabling this option, the client will receive additional host keys from the server after authentication, given that the server supports the protocol extension.

```
UpdateHostKeys yes
```

Enable transfer of additional host keys.

To combine these options, a user would enable *UpdateHostKeys* and set the option *HostKeyAlgorithms* after connecting to each server at least once. Warnings might still be triggered for servers that do not support the protocol extension, however the warning can be circumvented manually. If the client shows a warning that the host key of the server changed and the client stores another host key, the user can force the use of the existing host key by setting *HostKeyAlgorithms* to the respective algorithm. Then, the user should transfer the host key of the preferred type using *UpdateHostKeys* or manually.

```
UpdateHostKeys yes
HostKeyAlgorithms ecdsa-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384-
cert-v01@openssh.com,ecdsa-sha2-nistp521-cert-v01@openssh.com,sk-ecdsa-sha2-
nistp256-cert-v01@openssh.com,ssh-ed25519-cert-v01@openssh.com,sk-ssh-ed25519-
cert-v01@openssh.com,rsa-sha2-512-cert-v01@openssh.com,rsa-sha2-256-cert-
v01@openssh.com,ssh-rsa-cert-v01@openssh.com,ecdsa-sha2-nistp256,ecdsa-sha2-
nistp384,ecdsa-sha2-nistp521,sk-ecdsa-sha2-nistp256@openssh.com,ssh-ed25519,sk-
ssh-ed25519@openssh.com,rsa-sha2-512,rsa-sha2-256,ssh-rsa
```

Enable transfer of additional host keys and disable dynamic order of host key algorithms.

```
user@client:~$ ssh user@192.168.249.131
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:EQZ...
Please contact your system administrator.
Add correct host key in /home/user/.ssh/known_hosts to get rid of this message.
Offending RSA key in /home/user/.ssh/known_hosts:1
ECDSA host key for 192.168.249.131 has changed and you have requested strict
checking.
Host key verification failed.
user@client:~$ ssh -o HostKeyAlgorithms="ssh-rsa" user@192.168.249.131
Last login: Tue Jun  9 06:39:31 2020 from 192.168.249.1
user@server:~$ cat /etc/ssh/ssh_host_ecdsa_key.pub
ecdsa-sha2-nistp256 AAA...
user@server:~$ exit
user@client:~$ echo "192.168.249.131 ecdsa-sha2-nistp256 AAA..." >
~/.ssh/known_hosts
user@client:~$ ssh user@192.168.249.131
Last login: Tue Jun  9 06:40:23 2020 from 192.168.249.1
```

Forcing the use of an existing host key and transferring a host key manually.

3.2 PuTTY

The preferred order of host key algorithms can be configured in the host keys config panel in PuTTY, though stored host keys are always preferred. Version 0.74 adds an option to disable the dynamic order of host key algorithms. The option is disabled by default. Because PuTTY users are only susceptible if they connect to a server for the first time and the server does not support the default algorithm, legacy servers should also be updated to support Ed25519 host keys.

4 Disclosure Timeline

- 26.03.2020: Report of the vulnerability to OpenSSH
- 01.04.2020: Acknowledgment of receipt
- 30.04.2020: OpenSSH developers mention that PuTTY is vulnerable too
- 11.05.2020: Report of the vulnerability to PuTTY
- 13.05.2020: Acknowledgement of receipt
- 26.06.2020: Publication