Importance of SSHFP
And
Configuring SSHFP for Network Devices

Muhammad Moinur Rahman
moin@bofh.im

SSHFP

```
root@ns:~ # ssh root@pdr.bofh.network
The authenticity of host 'pdr.bofh.network (2604:6800:0:162:0:1:0:1)'
can't be established.
ECDSA key fingerprint is
SHA256:AlzRr/ZNFC9fjs89jYAD1o2dFDs4vu3gUVUD7gI2QBk.
No matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)?
```

- Have you ever logged in that host or device ?
- Have you ever checked the public key ?
- Do you know the SHA256 Fingerprint of the ECDSA Public Key?
- 99.9% Sysadmin or Network admin never checks it @ console
- Victim of MITM Attack

SSHFP

- First came up in 2006
- Defined in RFC 4255
- Creates a DNS record of type SSHFP
- Distributed by DNS, Verified by DNS lookup and secured by DNSSEC

Previous Usage

- Distributed known_hosts file
- Easily modify
- No access, no verification
- Maybe use a bastion host

known_hosts file

```
pdr.bofh.network ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdH
AyNTYAAABBBLGgYTGJQyJFOGAn3C31QG1aw/LQDl+x
q+M1pvM6RAW8vGvuaMsS5c23PjcR0zEfsjhKCV33vY
hXf7hxshcDZVI=
```

- Host
- Algorithm
- Public Key

Why SSHFP?

- SSH Public Key size are large to distribute via DNS
- So need the fingerprint
- Fingerprint method
  - Base64
  - sha1/sha256 checksum

What we need?

- Fingerprints of the host
- Proper SSHFP DNS records
- Zone must be DNSSEC signed
- A DNSSEC validator DNS server or ldns support in ssh client
- A SSH client capable to verify DNSSEC validated fingerprint

Fingerprints of the host?

```
root@pdr:~ # ssh-keygen -r pdr.bofh.network

pdr.bofh.network IN SSHFP 1 1 2cfb54c336799bf601a17a6b2723d096ed23ce55

pdr.bofh.network IN SSHFP 1 2
95a39495ec59ad717c0990bfe1f3c8ddd9e2b1201065e0ca5fc381cbc7ea8d8b

pdr.bofh.network IN SSHFP 2 1 90eb07d57e037aacbec479ed3a4c6a1264edf4f0

pdr.bofh.network IN SSHFP 2 2
eccdc4d93bb9af3eb4791e30f66aa327d9f0c9c388f5e950159ec285bb746783

pdr.bofh.network IN SSHFP 3 1 491ff6cd714362c5bae223fc2c942dbf78c3ba0e

pdr.bofh.network IN SSHFP 3 2
597d44db5b0e28f787ad5b1535e8b201e509373f8a8e711c71c950556ecdf799

pdr.bofh.network IN SSHFP 4 1 c63d32f031cc3fc7fe867baf2d0cd23d4ef25213

pdr.bofh.network IN SSHFP 4 2
15d249791d60bc46ec400c3a16a66b1cca191b2d30464707e8e5ba204dea1433
```

Dissecting Fingerprints

- Hostname
- Record CLASS (Internet)
- Record Type (SSHFP)
- Algorithm
  - 1 – RSA
  - 2 – DSA
  - 3 – ECDSA
  - 4 – Ed25519
- Fingerprint TYPE
  - SHA1
  - SHA256
- Key

Algorithm

- Most modern OS will show all 4
- Old OS might show only RSA and DSA
- ECDSA and Ed25519 are modern Cryptographic algorithm
- Ed25519 added in RFC7479
- For ECDSA and Ed25519 we need OpenSSH=>6.7

Distributing via DNS

```
root@pdr:~ # drill -D pdr.bofh.network sshfp
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 60914
;; flags: qr rd ra ad ; QUERY: 1, ANSWER: 9, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;; pdr.bofh.network.  IN  SSHFP
;; ANSWER SECTION:
pdr.bofh.network. 3599   IN  SSHFP  2 1 90eb07d57e037aacbec479ed3a4c6a1264edf4f0
pdr.bofh.network. 3599   IN  SSHFP  3 2 597d44db5b0e28f787ad5b1535e8b201e509373f8a8e711c71c950556ecdf799
pdr.bofh.network. 3599   IN  SSHFP  3 1 491ff6cd714362c5bae223fc2c942dbf78c3ba0e
pdr.bofh.network. 3599   IN  SSHFP  4 2 15d249791d60bc46ec400c3a16a66b1cca191b2d30464707e8e5ba204dea1433
pdr.bofh.network. 3599   IN  SSHFP  1 2 95a39495ec59ad717c0990bfe1f3c8ddd9e2b1201065e0ca5fc381cbc7ea8d8b
pdr.bofh.network. 3599   IN  SSHFP  4 1 c63d32f031cc3fc7fe867baf2d0cd23d4ef25213
pdr.bofh.network. 3599   IN  SSHFP  1 1 2cfb54c336799bf601a17a6b2723d096ed23ce55
pdr.bofh.network. 3599   IN  SSHFP  2 2 eccdc4d93bb9af3eb4791e30f66aa327d9f0c9c388f5e950159ec285bb746783
pdr.bofh.network. 3599   IN  RRSIG  SSHFP 8 3 3600 20171114225917 20171031212226 15678 dzcrd.net.
QBTRbj8xtyEN/9WH/PL39n1mC0XOKmGDj5TzgP/Kkvo7ac3wPwZ92dEcVnKyi1H2e8wP6532NIMjmuveyundnavCCbstOUfFyN17fBuEtHQTJzLmp8XQ7JxDgXbzbp6bvaKrck/XBFbRfk895oI9+Spg09fYkfseN4axEVoscQY=
;; AUTHORITY SECTION:
;; ADDITIONAL SECTION:
;; Query time: 1750 msec
;; EDNS: version 0; flags: do ; udp: 512
;; SERVER: 127.0.0.1
;; WHEN: Wed Nov  1 21:35:20 2017
;; MSG SIZE  rcvd: 530
```

SSHFP Validation with ssh client

```
root@ns:~ # ldd /usr/bin/ssh
/usr/bin/ssh:
libprivatessh.so.5 => /usr/lib/libprivatessh.so.5 (0x80084f000)
libgssapi.so.10 => /usr/lib/libgssapi.so.10 (0x800aee000)
libcrypto.so.8 => /lib/libcrypto.so.8 (0x800e00000)
libc.so.7 => /lib/libc.so.7 (0x801269000)
libprivateldns.so.5 => /usr/lib/libprivateldns.so.5 (0x801621000)
libcrypt.so.5 => /lib/libcrypt.so.5 (0x80187f000)
libz.so.6 => /lib/libz.so.6 (0x801a9e000)
```

If ldns is supported ..

- Need to maintain a trust-anchor
- Run `unbound-anchor`
- Add to */etc/resolv.conf*

```
anchor /etc/unbound/root.key [*BSD]
anchor /var/lib/unbound/root.key [Linux]
```

Else ..

- Run a validating resolver like Unbound

```
server:
        auto-trust-anchor-file:
"/etc/unbound/root.key"
```

Test

- root@pdr:~$ ssh root@pdr.bofh.network
  The authenticity of host 'pdr.bofh.network (2003:51:6012:110::9)' can't be established.
  ECDSA key fingerprint is SHA256:T09/p/ZSubnkraG3oslDMehfIiLRe6UiVn1dGZvtjZE.
  Are you sure you want to continue connecting (yes/no)?
  ^C
- root@pdr:~$ ssh -o VerifyHostKeyDNS=yes root@pdr.bofh.network
  root@pdr:~$

Test Successful

- DNSSEC Validation Works
- SSH login without .known_host works

SSH/SSHFP Demystified

- SSH Creates Public/Private Keys
  - For Linux/Unix under /etc/ssh/ssh_host_<ALGORITHM>.*
  - Normally during first run of sshd service
  - This part is automatic and hidden
- ssh-keygen creates fingerprints for those keys

SSHFP for Network Equipments

- Network Equipments don’t have ssh-keygen command
- Equipments comes with limited subset of ssh
- Configure ssh service for devices preferably version 2
- Create the public/private keys normally RSA only
- Gather the public key connecting through console or direct point-to-point connectivity
- Generate the Fingerprints for SSHFP records

Configuring Router

```
R1#conf t
R1(config)#ip domain-name bofh.network
R1(config)#crypto key generate rsa modulus 4096
R1(config)#crypto key generate rsa modulus 4096
R1(config)#crypto key generate rsa modulus 4096
R1#show ip ssh
SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
Minimum expected Diffie Hellman key size : 1024 bits
IOS Keys in SECSH format(ssh-rsa, base64 encoded):
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC1pb6dZ01nHBPZUZXcTxf7swq3NoNc6Geiz9C006YS
gCwz47msZn7YGIlPtrhw+GGT2jZo+34xSXekVYV9Cie+e8NI4UgD7b+aXma6aNPGfDN338jAmWBN7SDB
sZvLJnjDYndNqRtYXby8uhvzWfGNtPwIZCXgbX+nOTmCR9Ap0iFF4zxzPppwPko41xdyAaOc4qros8aR
ocLP8hrEeEz9R7WWZZ5ui/+ya3wa/SFjOZIcnv/7BKl1E5z+CM0OneDUfktw6lEd3hYGr347sz5I4Obj
ia3HJ1D3lQKtRcT8zqavdlDyaYZEvYbg4eMp6aSMjJ79BOhhGgiFJ/Mxw7isMikfd6iKisdU+pd3XxQN
mNlDywfuRKl9RYfYAVNG3PvOAqzECplh4s5+jl1+SWeKIg3ah7iFv7ddEVdBtaOrCsVxaOJuIMYPioTH
aoDNmeOpfZJa/VIqker78bTlIgIGALybnnniFYhAb9ztYJv8g14pmu7cZol3lGg9pa/Qi6sdCAVyizdj
0wsS/s8Sl8VhwoZpxEUs0XQIHEsKP47XpJdmKAWG3vYopWYz7JCRmpwl8+oRCHLL35zV4Em+BO4aH/gt
+nRQuEaVFBwaHGbe0A77d4ABIHLWAvuFb4wr+oOGoD57kDolrFYq7IfW/d261+DMr11Ip0kJEncXIgZc
```

Extracting Key

- Connect with Console
- Connect directly with point to point LAN
- Use show ip ssh
  - Copy the string starting with “ssh-rsa” and ending till last line
- For remote connection add the key with prompt and verify the keys
- Save the file in a Linux/Unix host with filename like hostname_rsa_key.pub

Using the script

```
#!/usr/local/bin/bash

HOST="${1}"

if [[ "$1" == "-h" || "$1" == "--help" ]]

then

  echo "Usage: sshfpgen <hostname>"

fi

for pubkey in *_key.pub

do

      echo "$HOST IN SSHFP 1 1 $(cut -f2 -d ' ' "$pubkey" | base64 --decode | shasum  | cut -f 1 -d ' ')"

      echo "$HOST IN SSHFP 1 2 $(cut -f2 -d ' ' "$pubkey" | base64 --decode | shasum -a 256  | cut -f 1 -d ' ')"

done
```

Running the script

```
root@ns:~ # ./sshfpgen r1.bofh.network

r1.bofh.network IN SSHFP 1 1 96898fe4604ed5e7b1a1c80374b1200fae3f4adb

r1.bofh.network IN SSHFP 1 2 ebfca263706e4e12c7189ae377014f30467af6e6243d3b8a7e5f763d2813023b
```

Adding to DNS

- Following records were added into the DNS

```
r1.bofh.network IN SSHFP 1 1 96898fe4604ed5e7b1a1c80374b1200fae3f4adb

r1.bofh.network IN SSHFP 1 2 ebfca263706e4e12c7189ae377014f30467af6e6243d3b8a7e5f763d2813023b
```

- DNS Query Test

```
$ dig SSHFP +noadditional +noquestion +nocomments +nocmd +nostats r1.bofh.network

r1.bofh.network. 3600  IN  SSHFP 1 1 96898fe4604ed5e7b1a1c80374b1200fae3f4adb

r1.bofh.network. 3600 IN  SSHFP 1 2 ebfca263706e4e12c7189ae377014f30467af6e6243d3b8a7e5f763d2813023b
```

Connectivity Test

- Connecting without DNSSEC validation

```
$ ssh VerifyHostKeyDNS=yes r1.bofh.network

The authenticity of host 'r1.bofh.network (192.168.100.100)' can't be established.

RSA key fingerprint is SHA256:y2STsQ4RA/8durhpic+pb6UjcKwz7+bUaKX3C40yOGk.

Matching host key fingerprint found in DNS.

Are you sure you want to continue connecting (yes/no)?
```

Connectivity Test

- Connecting with DNSSEC validation

```
$ ssh VerifyHostKeyDNS=yes r1.bofh.network

Username:
```

- Also no records will be added in .known_host file

Live Demonstration & Questions