Automatic SSH public key fingerprint retrieval and publication in DNSSEC

Pascal Cuylaerts &
Marc Buijsman

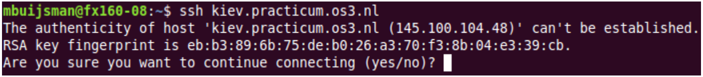
UNIVERSITY OF AMSTERDAM

2 February 2011

Overview

- Introduction
- Research
- Mechanism design
- Proof of concept
- Conclusion
- Demo

Introduction



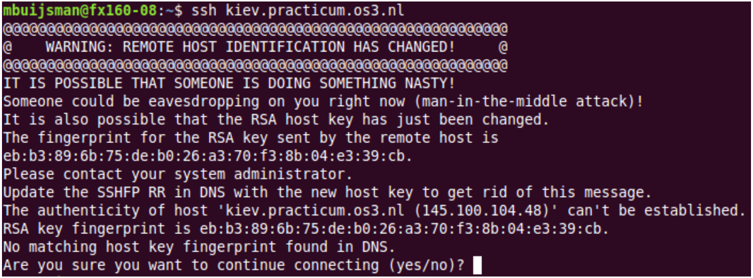
```
mbuijsman@fx160-08:~$ ssh kiev.practicum.os3.nl
The authenticity of host 'kiev.practicum.os3.nl (145.100.104.48)' can't be established.
RSA key fingerprint is eb:b3:89:6b:75:de:b0:26:a3:70:f3:8b:04:e3:39:cb.
Are you sure you want to continue connecting (yes/no)?
```

- First time SSH connection
  - □ Public key fingerprint (MD5 hash)
- Must do manual check
  - □ Inconvenient
  - □ Prone to human error and laziness
- Could use DNSSEC instead
  - □ No need to remember fingerprint
  - □ Key can be validated automatically

Introduction

- DNS has SSHFP resource records
  - SHA1 hash of both RSA and DSA public keys
  - `@ IN SSHFP 1 1 4249AA3FCF054089F9817DDBCDA89096F08C971E`
  - `@ IN SSHFP 2 1 A72B1B577E5822FD69F59703D2745C8EFD3949A5`
- DNSSEC signed records can be validated
- OpenSSH patch to do this automatically
- Can be warned if fingerprints don't match
  - Just like `known_hosts`, but then in DNS

Introduction



```
mbuijsman@fx160-08:~$ ssh kiev.practicum.os3.nl
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    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 the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
eb:b3:89:6b:75:de:b0:26:a3:70:f3:8b:04:e3:39:cb.
Please contact your system administrator.
Update the SSHFP RR in DNS with the new host key to get rid of this message.
The authenticity of host 'kiev.practicum.os3.nl (145.100.104.48)' can't be established.
RSA key fingerprint is eb:b3:89:6b:75:de:b0:26:a3:70:f3:8b:04:e3:39:cb.
No matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)?
```

Introduction

- DNS is accessible by anyone
  - One DNS versus many `known_hosts` files
- Correct fingerprint (FP) must be published
  - People will think: DNSSEC validated, so FP valid
  - Malicious FP is big vulnerability
- Retrieving FP manually is safest
  - Easy for only one machine
  - But cumbersome for many machines...
- Automation desirable
  - But how to do this securely?

Research - Research question

How can SSH public key fingerprints be automatically collected from remote machines and published in DNSSEC in a secure way?

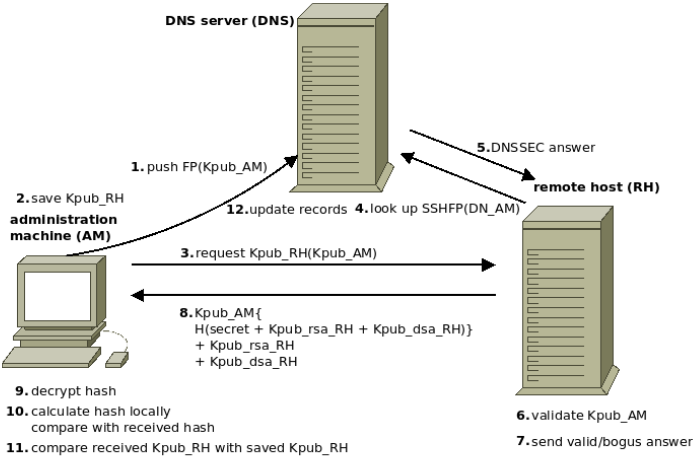
Research

- Need to authenticate many machines
  - Public keys cannot be used
- Securing channel without pre-shared information?
  - Man-in-the-middle detection
  - Risk reduced to first connection
  - LAN is considered fairly secure
- Never 100% secure
- Authentication desired
  - Remote host must proof its identity
  - Public/private key pair not trusted
  - Need something else: pre-shared secret

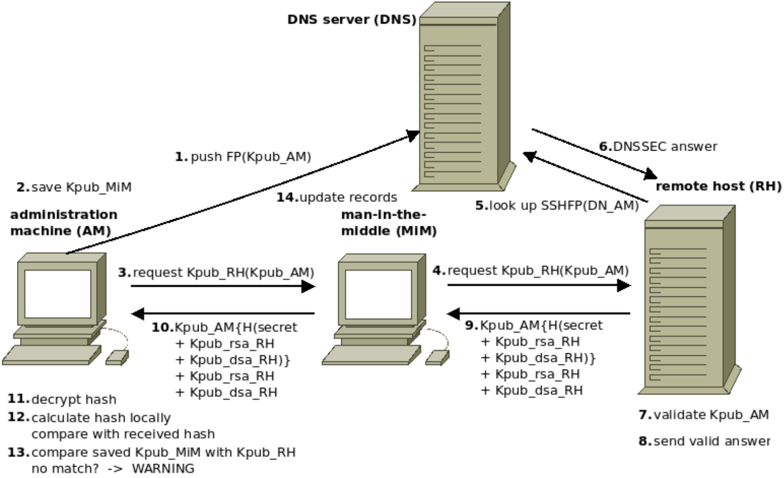
Research

- Administrator knows the secrets
  - File should be password protected
- Secret should be relatively strong
  - System UUID
  - Motherboard serial + product name
- Remote machine can look this up
  - Only with root permissions

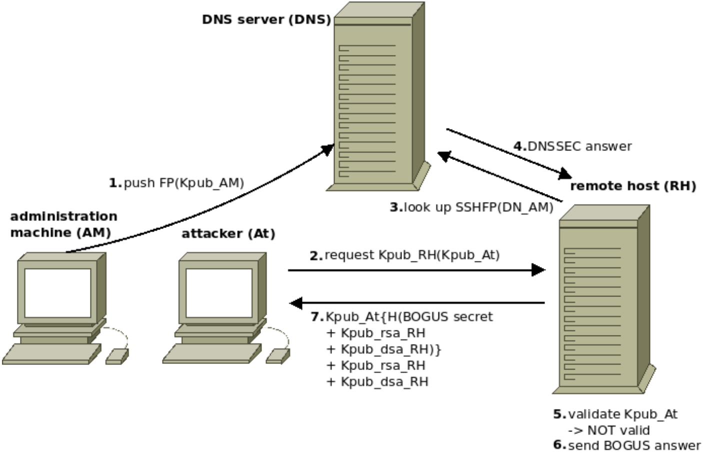
Mechanism design


DNS server (DNS)
1. push FP(Kpub_AM)
2. save Kpub_RH
administration machine (AM)
12. update records
4. look up SSHFP(DN_AM)
5. DNSSEC answer
remote host (RH)
3. request Kpub_RH(Kpub_AM)
8. Kpub_AM{
H(secret + Kpub_rsa_RH + Kpub_dsa_RH)}
+ Kpub_rsa_RH
+ Kpub_dsa_RH
9. decrypt hash
10. calculate hash locally
compare with received hash
11. compare received Kpub_RH with saved Kpub_RH
6. validate Kpub_AM
7. send valid/bogus answer

Mechanism design - MITM



Mechanism design - direct attacker



Proof of concept - components

- Aministration machine
  - Python application
  - dependencies (argparse, M2Crypto, libssh2, nsupdate)
  - Python interface for libssh2 C library
  - configuration file
  - encrypted secrets file
  - shared (with DNS) key file
- Remote host
  - Python application
  - dependency (argparse, M2Crypto, libunbound)
  - configuration file
  - restricted user account
  - edited sudoers file

Proof of concept - components

- DNS server
  - SSHFP records for administration machine
  - edited `named.conf`
    - allow for dynamic updates (`nsupdate`)
    - shared (with AM) key in `named.conf`

Conclusion

How can SSH public key fingerprints be automatically collected from remote machines and published in DNSSEC in a secure way?

- Need shared information to authenticate of remote hosts
  - Necessary to ensure correctness of fingerprint
- Our scheme ensures authenticity and integrity
- Automation possible with our applications

Proof of concept - demo

Demo

Q&A