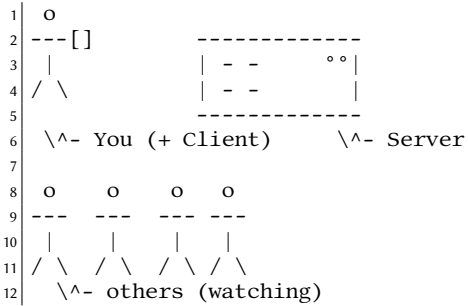
A short Update on (Open)SSH Security

André Niemann, becon GmbH
$vorname.$nachname@becon.de

Denog 10

Why and How SSH



(Why do you like) OpenSSH?

> “das es klappt, es geht halt”
>
> — OpenSSH User, IRC

(Why do you like) OpenSSH?

> “das es klappt, es geht halt”
>
> — OpenSSH User, IRC

> “-NL ist die Magic, die man immer braucht ”
>
> — another OpenSSH User, IRC

(Why do you like) OpenSSH?

"das es klappt, es geht halt"

— OpenSSH User, IRC

"-NL ist die Magic, die man immer braucht "

— another OpenSSH User, IRC

"SSH funktioniert einfach(, nicht wie TLS)"

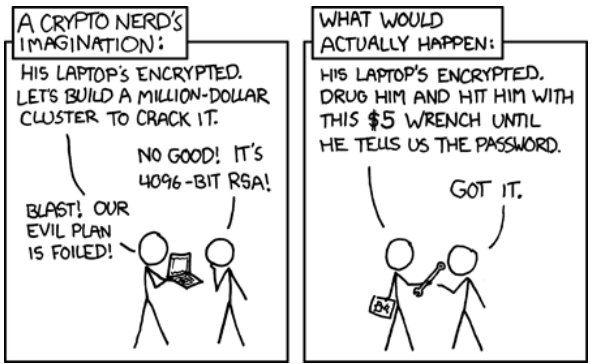
— bei ner Limo

Built-in (security) Features

- modular support for crypto primitives
- Rekeying for longer sessions
- both sides authenticated
- modular protocol (Trans, User, Conn)
- privilege Seperation

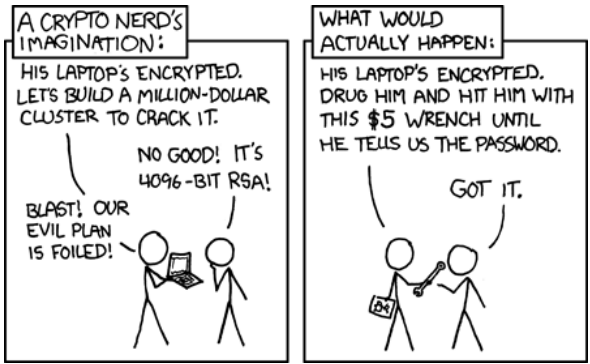
ssh -Q cipher

- 3des-cbc
- blowfish-cbc
- cast128-cbc
- arcfour
- arcfour256
- aes128-cbc
- aes128-ctr
- aes256-ctr
- aes128-gcm@openssh.com
- aes256-gcm@openssh.com
- chacha20-poly1305@openssh.com



ssh -Q cipher-auth

- aes128-gcm@openssh.com
- aes256-gcm@openssh.com
- chacha20-poly1305@openssh.com

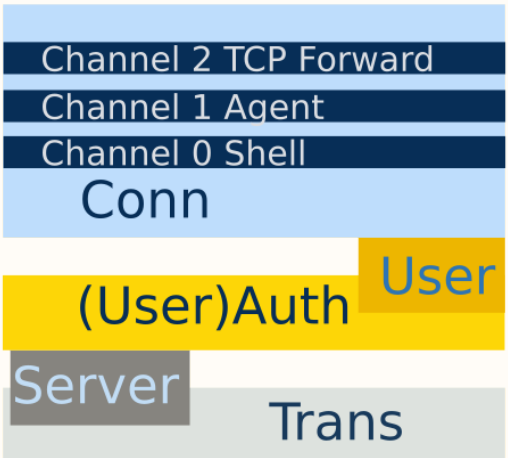


Rekeying

RekeyLimit

Specifies the maximum amount of data that may be transmitted before the session key is renegotiated, optionally followed a maximum amount of time that may pass before the session key is renegotiated. The first argument is specified in bytes and may have a suffix of [..], respectively. The default is between ‘1G’ and ‘4G’, depending on the cipher. The optional second value is specified in seconds [..]. The default value for RekeyLimit is default none, which means that rekeying is performed after the cipher’s default amount of data has been sent or received and no time based rekeying is done.

Protocol recap



Scenario: Server authentication

Hostkey Validation?!

- TOFU
- 'local Database' (/.ssh/known_hosts)
- GlobalKnownHostsFile (/etc/ssh/ssh_known_hosts)
- SSHFP Records (dnssec)
- UpdateHostKeys (man 5 ssh_config)
- SSH CA -> (AUTHORIZED_KEYS FILE FORMAT section)

Scenario: Working remotely

- ssh user@host

Scenario: Working remotely

» ssh user@host

☹ Password, sucks.

Scenario: Working remotely

- » ssh user@host
- ☹ Password, sucks.
- » ssh-keygen
- » ssh-copy-id || cat .ssh/authorized_keys < id.pub
- » ssh user@host [-i id.pub]

Scenario: Working remotely

- » ssh user@host
- ☹ Password, sucks.
- » ssh-keygen
- » ssh-copy-id || cat .ssh/authorized_keys < id.pub
- » ssh user@host [-i id.pub]
- ☹ again pass, still sucks, but in a more secure way

Client Security - Keys

$ file rsa4key

- rsa4key: PEM RSA private key

» ssh keygen -o (or newer OpenSSH or ed25519 keys)

$ file newformat_id

- newformat_id: OpenSSH private key

Client Security - SSH Agent

- ssh-agent / gpg-agent

```
$ env | grep -i auth
SSH_AUTH_SOCK=/tmp/ssh-srFyhRtSTS3V/agent.2185

agent-security:
$ ls -l /tmp/ssh-srFyhRtSTS3V/agent.2185
srw------- 1 andre andre 0 May 23 15:23 /tmp/ssh-srFyhRtSTS3V/agent.2185
```

- ssh-add -c key (type [y][e][s])
- -l
- -D / -d key
- -x (to lock)

/.ssh/config

```
Host *
    KexAlgorithms curve25519-sha256@libssh.org,diffie-hellman-group-exchange
        -sha256
    Ciphers chacha20-poly1305@openssh.com, aes256-gcm@openssh.com

Host box
    VisualHostKey yes
    Identityfile legacyKey
    Ciphers +3des-cbc
    HostKeyAlgorithms +ssh-dss

Host Jump.box
    User andre
    ForwardAgent yes
    Port 4242
    DynamicForward 33334
```

Scenario: SSH CA

- id_rsa
- id_ed25519
- id_??-cert ?

Scenario: SSH CA

- id_rsa
- id_ed25519
- id_??-cert ?

rfc4251: Section 4.1

o The client has a local database that associates each host name (as typed by the user) with the corresponding public host key. [..]
o The host name-to-key association is certified by a trusted certification authority (CA). [..]

Scenario: SSH CA

- id_rsa
- id_ed25519
- id_??-cert ?

rfc4251: Section 4.1

o The client has a local database that associates each host name (as typed by the user) with the corresponding public host key. [..]
o The host name-to-key association is certified by a trusted certification authority (CA). [..]

The SSH protocol currently supports a simple public key authentication mechanism. Unlike other public key implementations, SSH eschews the use of X.509 certificates and uses raw keys.

SSH Key signing/CA

- ssh-keygen -f ssh-ca -b 4096
- echo ”cert-authority $(cat ssh-ca.pub)≫ /.ssh/authorized_keys
- ssh-keygen -s signing-key -I key-identifier -h -n hostname -V +52w host-key

SSH Key signing/CA 2

- automation (fetch with ansible, delegate_to signign host, deploy)
- there are 'tools'.

SSH Key signing/CA 2

- automation (fetch with ansible, delegate_to signign host, deploy)
- there are 'tools'.

Issues?

- Vendorsupport?
- revocation, decommissioning
- renew (for short lived)
- function User (sign-in commands)
- binding user-key

shd_config

- Rootlogin (prohibit-password)
- AllowUsers/Groups
- Match blocks
- PubkeyAcceptedKeyTypes
- AuthenticationMethods publickey,password publickey,publickey

Scenario: Pubkey + 2FA

```
AuthenticationMethods publickey,password
```

```
local ~ \$ ssh root@box
Authenticated with partial success.
root@box's password:
Welcome to box
```

Hardening

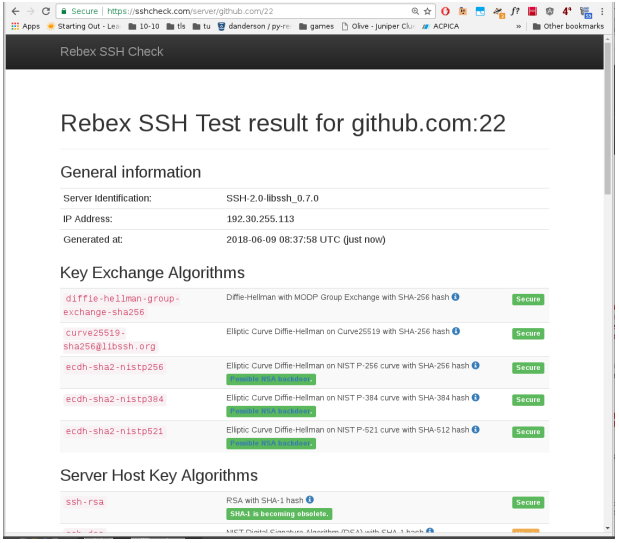
- no CBC, RC4 Cipher
- no old Hash-Alogs (< sha256)
- no pass-auth
- keep private keys private (no github)
- >= rsa2048
- authorized_keys in central place

Hardening

- no CBC, RC4 Cipher
- no old Hash-Alogs (< sha256)
- no pass-auth
- keep private keys private (no github)
- >= rsa2048
- authorized_keys in central place

- mozilla wiki (https://wiki.mozilla.org/Security/Guidelines/OpenSSH)
- BSI TR-02102-4

Check it - https://sshcheck.com



What else is left?

- talk about infrastructure tools!

What else is left?

- talk about infrastructure tools!
- review configs some times

What else is left?

- talk about infrastructure tools!
- review configs some times
- check ciphers as well (ssh + TLS)

What else is left?

- talk about infrastructure tools!
- review configs some times
- check ciphers as well (ssh + TLS)
- man 5 ssh_config