SSH features

Things waiting to happen when you run a ssh server

structure

- speach
  30 minutes
- demonstration
  20 minutes
- discussion
  20+ minutes

Things that can happen

- MiM attacks (main topic)
- authentication token guessing/stealing
  password bruteforcing,
  RSA key theft
- timing attacks
- protocol attacks

Things less interesting

- buffer overflows
- etc.

Man in the Middle (MiM)

- common attack against public key systems
- attacker spoofs public key of communicating parties
- with SSH this means: sending fake hostkeys to client and negotiating a known session key to sniff communication
- easy with LANs and WLANs

Hostkeys?

- integral part of SSHv1 and SSHv2
- used with SSHv1 to encrypt secret session key
- used with SSHv2 to sign the session key negotiation (DH)
- transferred in both cases at beginning of conversation
- accepting a wrong hostkeys is like using telnet

Wrong Hostkey? MiM!

- for each host connected to, the hostkey is saved
- upon each connect, ssh client tries to find corresponding hostkey

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    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 RSA1 host key has just been changed.
The fingerprint for the RSA1 key sent by the remote host is
f3:cd:d9:fa:c4:c8:b2:3b:68:c5:38:4e:d4:b1:42:4f.
Please contact your system administrator.
```

The known hosts problem I

- ssh client records host-keys of hosts already connected to
- there are 3 different key-types (OpenSSH)
  ssh-rsa1 (SSHv1)
  ssh-rsa (SSHv2)
  ssh-dss (SSHv2)
- for host to be known, the host AND keytype have to match
- => lets us produce key misses via unknown keytypes!

The known hosts problem II

```
Enabling compatibility mode for protocol 2.0
The authenticity of host 'lucifer (192.168.0.2)' can't be established.
DSA key fingerprint is ab:8a:18:15:67:04:18:34:ec:c9:ee:9b:89:b0:da:e6.
Are you sure you want to continue connecting (yes/no)?
```

The known hosts problem III

```
RSA key fingerprint c1:12:f4:27:5f:ef:21:89:c0:33:fa:1a:57:20:e8:5f.
The authenticity of host '192.168.0.2 (192.168.0.2)' can't be establish
but keys of different type are already known for this host.
DSA key fingerprint is 9f:d3:fc:99:64:b0:93:a2:81:66:55:93:7d:ea:ed:e8.
Are you sure you want to continue connecting (yes/no)?
```

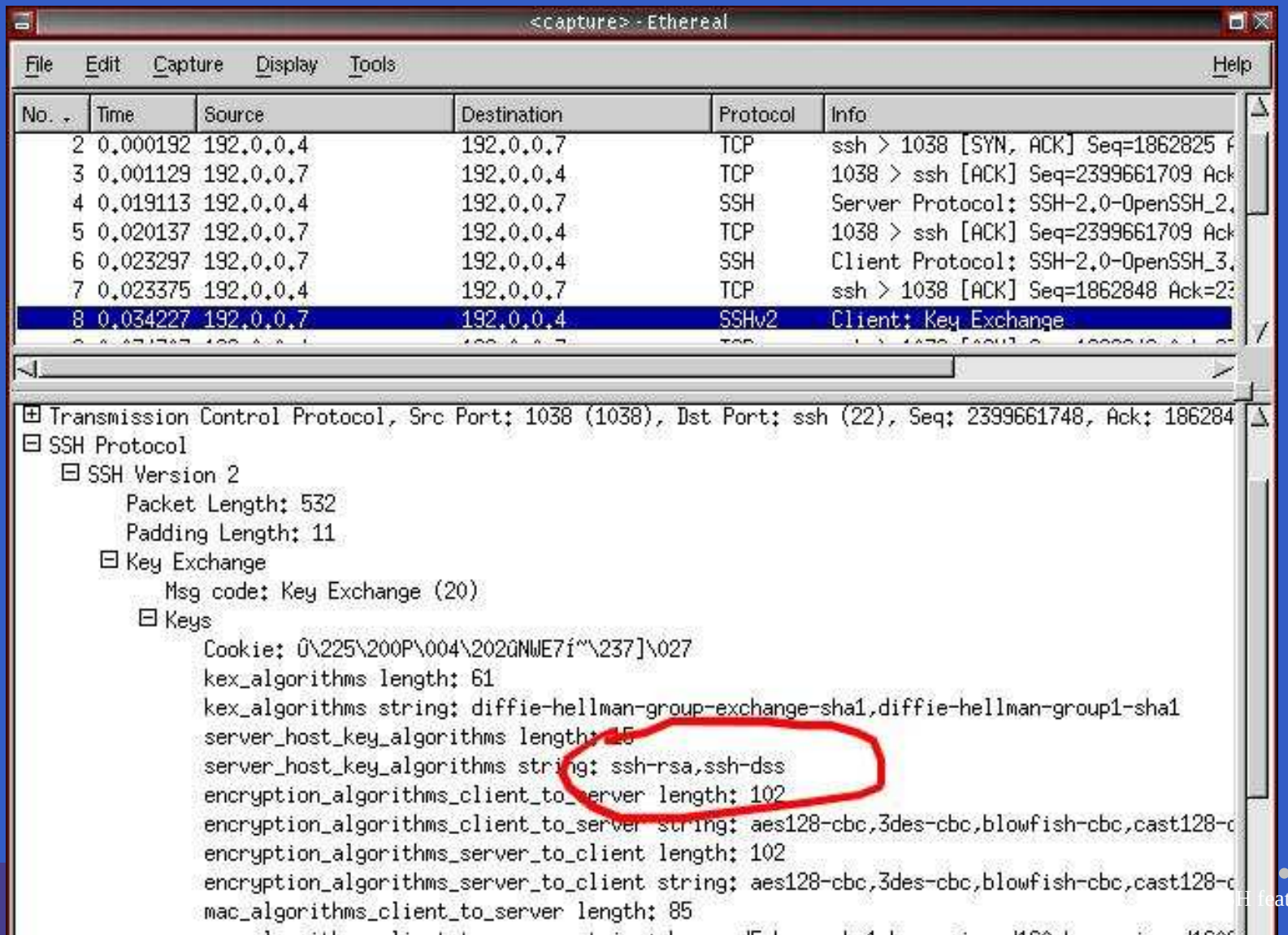
hostkey selection in SSHv1

- SSH server banner shows client which SSH versions are supported
- SSH-1.99-OpenSSH_2.2.0p1
- there is only one key-type: RSA
- if SSH client used to have SSHv1 key in known_hosts file, prompt with a SSHv2 only banner
- this is the case for SSHv1 only servers

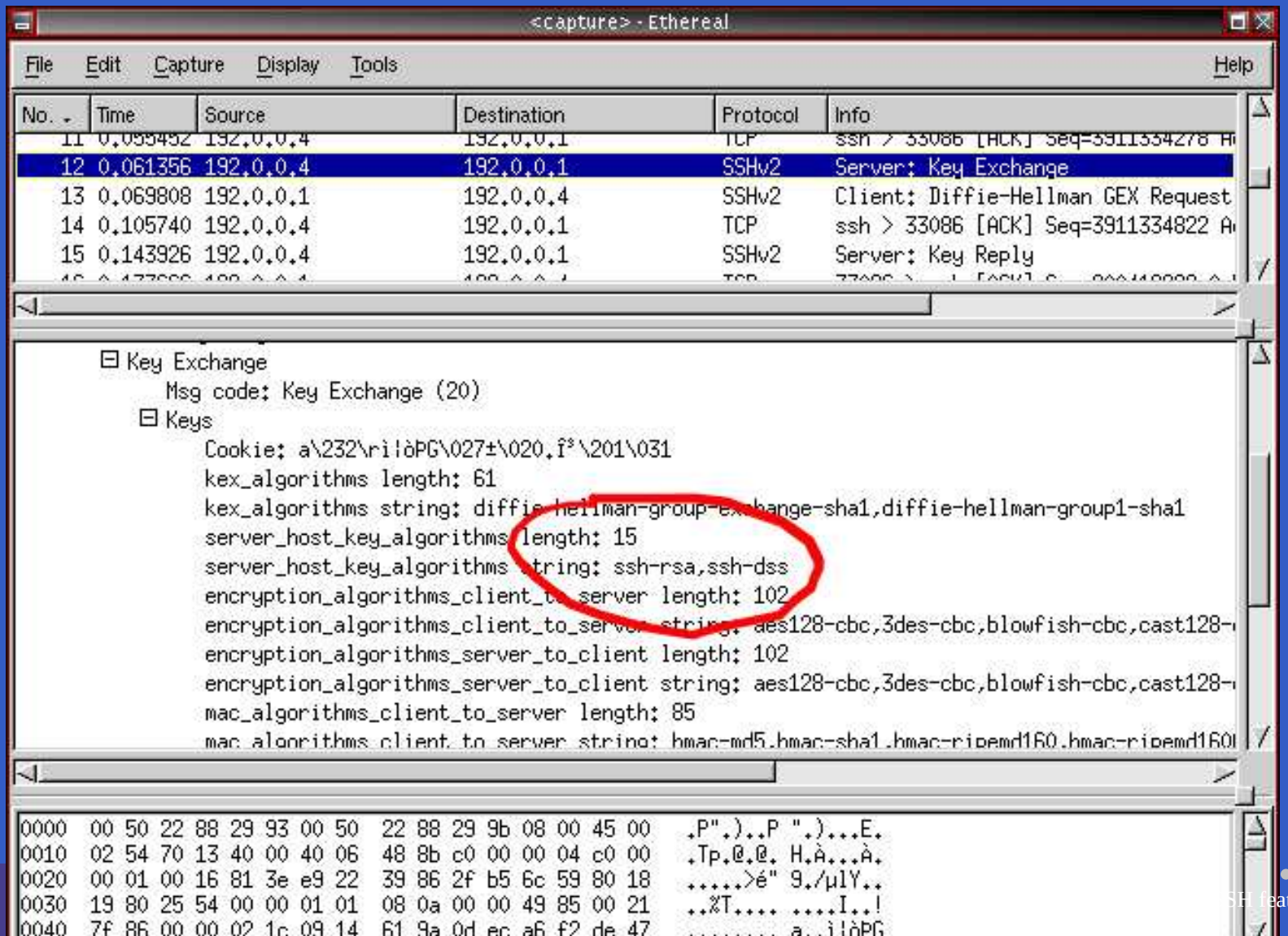
hostkey selection in SSHv2

- client and server send list of supported algorithms
- common: ssh-rsa and ssh-dss
- algorithm is selected as follows (RFC): ”The first algorithm on the client list that satisfies the requirements and is also supported by the server MUST be choosen.”
- Client: rsa, dss Server: dss => choose dss

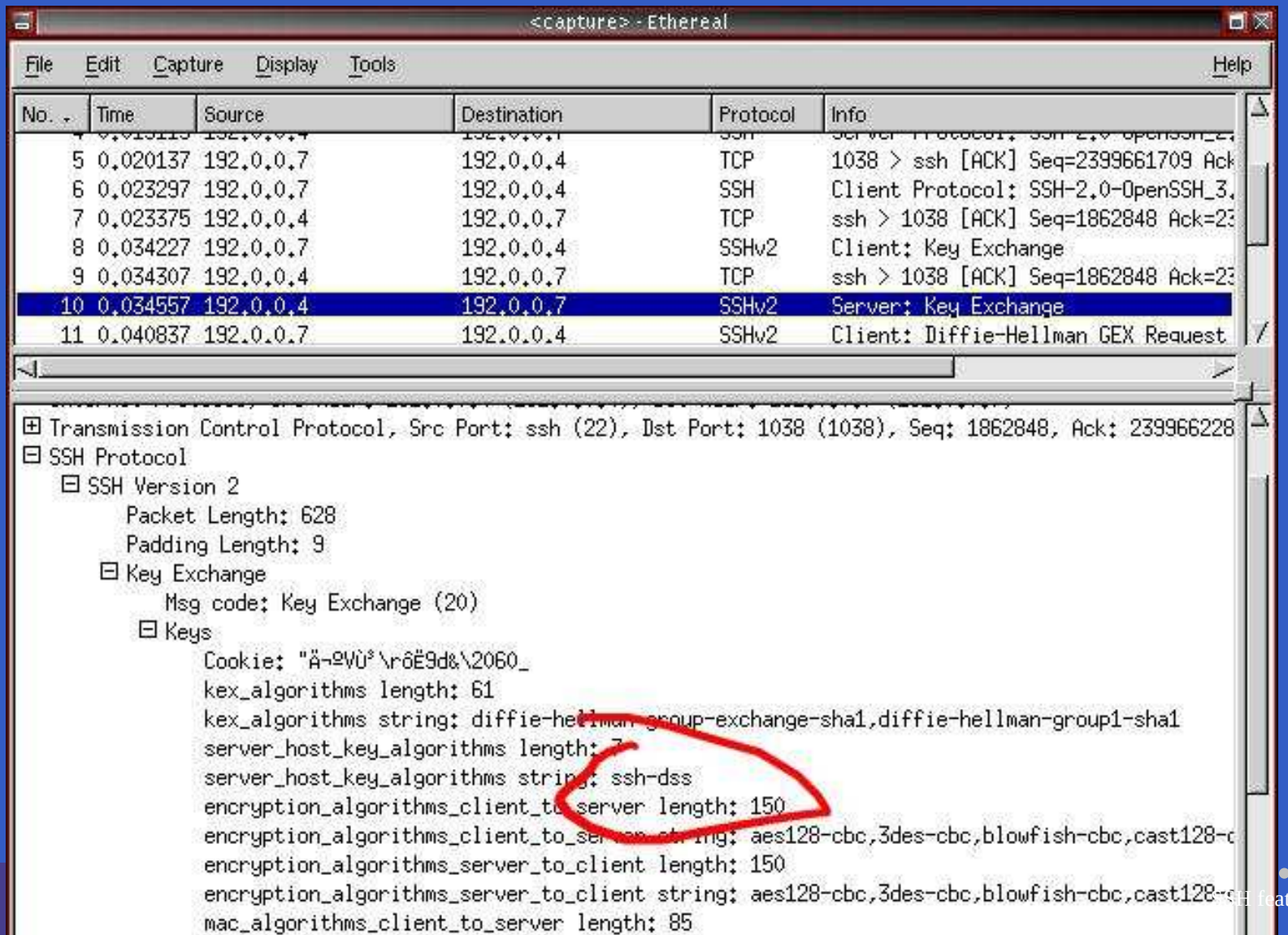
Client > Server Algorithm negotiation



Server > Client Algorithm negotiation



MiM > Client Algorithm negotiation



hostkey choosing I

- need to produce key miss via unknwon key type
- how do we know which algorithm to choose?
- connect to real server upon each MiM connection
- look for supported hostkeys (pre-connect)
- peek at client stream
- look for supported hostkeys (peeking)
- choose the right one

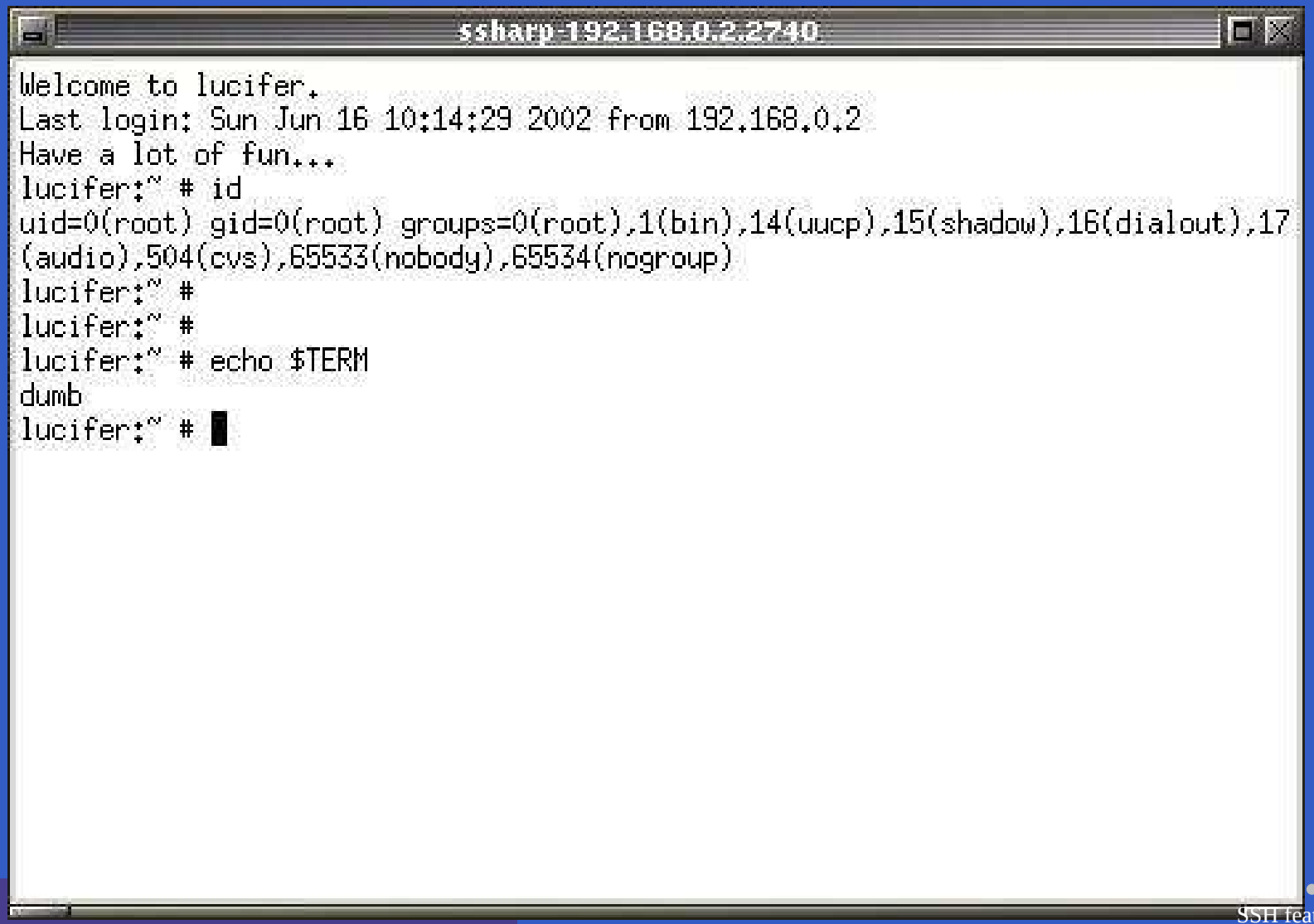
hostkey choosing II

client	server	RFC	MiM
rsa	rsa,dss	rsa	no way
dss	rsa,dss	dss	no way
rsa,dss	rsa	rsa	dss via pre-connect
rsa,dss	dss	dss	rsa via pre-connect
rsa,dss	rsa,dss	rsa	dss via peeking
dss,rsa	dss,rsa	dss	rsa via peeking
dss,rsa	rsa,dss	dss	rsa via peeking

Implementation

- reuse existing OpenSSH code
- patch server to accpet any login/password and to start special shell *ssharpclient* on a pty
- *ssharpclient* logs into remote-host => yields shell in pty
- optionally: slip *ssharpclient* through screen-like program => hunt for SSH

Attackers view



How to protect?

- *unknwon hostkey* messages should make you scared
- use RSA authentication
- do not use SSH1

References

- SSH Timing attacks
  http://www.openwall.com/advisories/
  http://www.openwall.com/presentations/
- ettercap SSH sniffer
  http://ettercap.sourceforge.net
- SSHarp
  http://stealth.openwall.net/SSH