

# SSH

- Secure SHell is a popular software approach to network security
  - operates at the application layer
  
- offers transparent encryption, authentication and integrity of data
  
- provides command line commands:
  - `ssh`
  - `sftp`
  - `scp`
  - `slogin`

# SSH architecture



# SSH Protocol

- provides Authentication
  - reliably determines someones identity using public/private key
  
- provides encryption
  - scrambles data as it passes across the network
  
- provides integrity
  - guarentees data travels across the network unaltered

## SSH Port Forwarding

- the ssh tools: `ssh`, `slogin`, `scp` and `sftp` are useful enough to warrant investment in time with ssh
- however the port forwarding capability ensures that ssh enters non command line networking!
- any port can be forwarded across ssh

## SSH Port Forwarding

- certain protocols transmit usernames/passwords in plaintext or using weak passwords
  - imap, pop3 and vnc, X windows
- ssh can be used to harden these very useful protocols

## SSH examples

- connecting to a remote machine

```
$ ssh moppsy.comp.glam.ac.uk  
Password:  
  
Linux moppsy i686 GNU/Linux  
Last login: Tue Feb  8 10:47:44  
  
fred@moppsy:~/ $ exit
```

## SSH examples

- using a command line ftp equivalent

```
$ sftp moppsy.comp.glam.ac.uk  
Password:  
sftp> dir  
sftp> get foo.ps  
sftp> quit
```

## X Windows Port forwarding

- GNU/Linux allows graphical applications to be run remotely
  - remote desktop per application
    - as well as per desktop (using vnc)

```
$ ssh -X moppsy.comp.glam.ac.uk
Password:

Linux moppsy i686 GNU/Linux
Last login: Tue Feb  8 10:47:44

fred@moppsy:~/ $ xterm
fred@moppsy:~/ $ exit
```

- ssh is forwarding all X traffic across port 22



## Browsing the web through an ssh connection

- suppose we want read the web pages of `mcgreg.comp.glam.ac.uk` securely

- ```
$ ssh -g -A -X -N -T -L2001:localhost:80 mcgreg.comp.glam.ac.uk
```

- which means create a secure link between port 2001 on localhost and port 80 on `mcgreg.comp.glam.ac.uk`

- ```
$ telnet localhost 2001
get index.html
Escquit
```

- or `http://localhost:2001/index.html`

## SSH through an untrusted proxy ssh server

- on your local machine you type:

```
$ ssh -g -A -X -N -T -L2001:trusted.com:22 untrusted.proxy.com
```

- which says open a secure connection starting at port 2001 on our local machine
  - which provides a connection between untrusted.proxy.com and trusted.com on port 22
  - the flags turn all port forwarding capability

## SSH through an untrusted proxy ssh server

- and in another terminal window type:

```
$ ssh -v -g -A -X -p 2001 localhost
```

- which now opens up a connection between your keyboard and localhost:2001
  - effectively giving you a secure encrypted connection to trusted.com:22

## Laboratory work

- try out all the examples presented in todays lecture

## Python and SSH

- type in the following code and analyse what it does!

# Python and SSH

```
#!/usr/bin/python

# secure shell pipe module

import os
import sys
from socket import *

localPortNo=8000
maxTries=10

# createTCPSocketSSH - creates a secure TCP socket between
#                      localhost:localPort and
#                      remoteHostname:remotePort

def createTCPSocketSSH (remoteHostname, remotePort=22,
                        localPort=-1):
```

# Python and SSH

```
global localPortNo
if localPort == -1:
    localPort = localPortNo
    localPortNo = localPortNo+1
tryNo = 1
while 1:
    command = "ssh -f -g -A -X -N -T -L%d:localhost:%d %s\n" %
              (localPort, remotePort, remoteHostname)
    print command
    result = os.system(command)
    if result == 0:
        break
    localPortNo = localPortNo+1
    tryNo = tryNo + 1
    if tryNo == maxTries:
        os.exit(1)
```

# Python and SSH

```
# create a TCP socket which connects to our ssh pipe
s = socket(AF_INET, SOCK_STREAM)
s.connect(("localhost", localPort))
return s
```

```
s = createTCPSocketSSH("mcgreg.comp.glam.ac.uk", 80)
s.send("get index.html\n")
print s.recv(1024)
```