Internet Security 2009W

Protokoll

Firewall

Manuel Mausz, Matr. Nr. 0728348
manuel-tu@mausz.at
Aldin Rizvanovic, Matr. Nr. 0756024
e0756024@student.tuwien.ac.at

Wien, am 25. November 2009

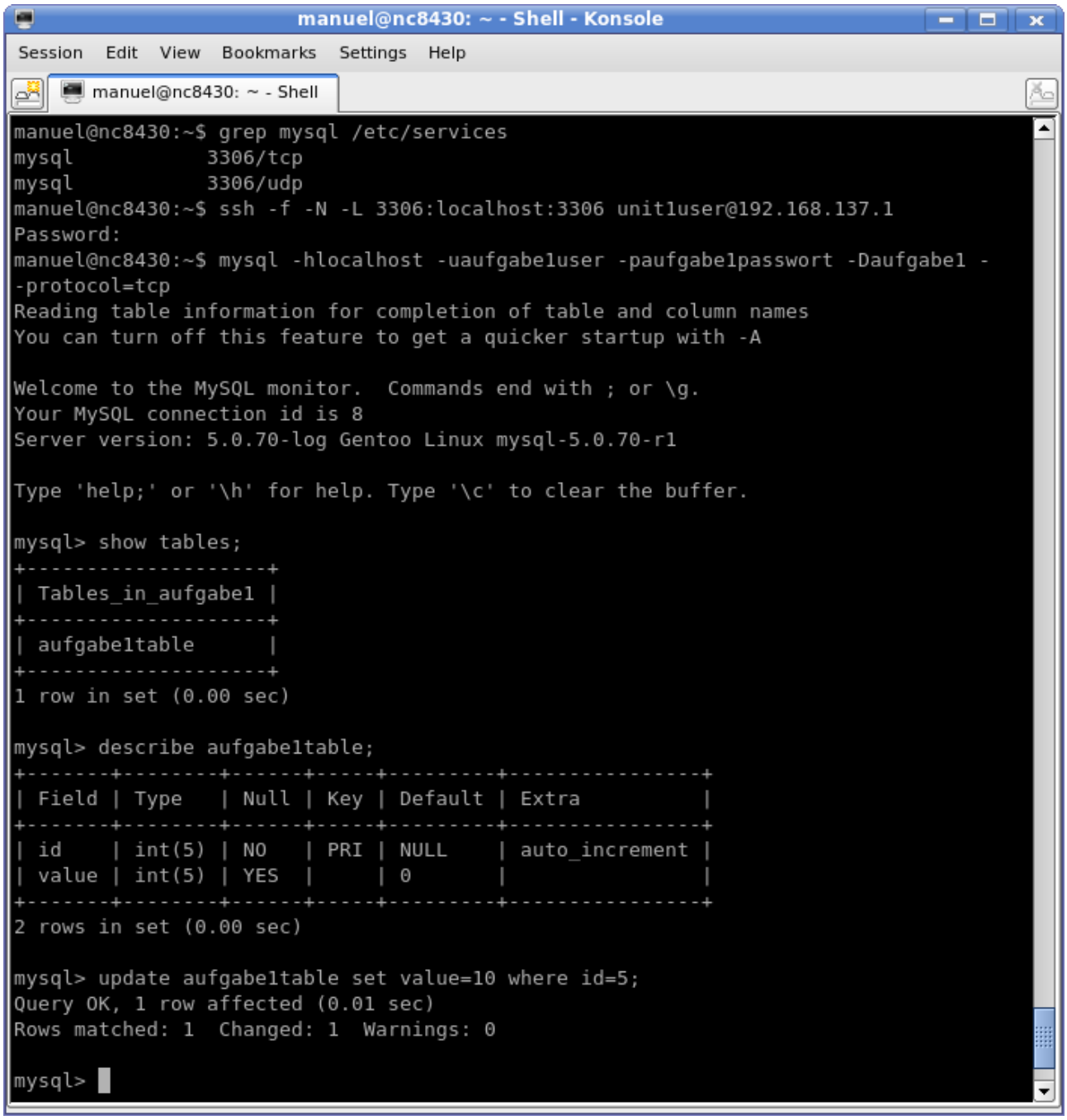
Inhaltsverzeichnis

Use SSH to access a blocked port

Die MySQL-Datenbank akzeptiert ausschließlich lokale Verbindungen. Über den zur Verfügung gestellten SSH Zugang und den SSH Client, sind wir in der Lage einen „*weitergeleiteten*“ Port zu öffnen. Sämtliche Verbindungen auf den lokal geöffneten Port werden dabei über SSH an den entfernten Rechner weitergeleitet und von diesem ausgeführt. Somit sind wir in der Lage auf unserem Rechner eine lokale Verbindung zur entfernten MySQL-Datenbank aufzubauen.

SSH Client Parameter: `-L [bind_address:]port:host:hostport`

Lösung des Beispiels:

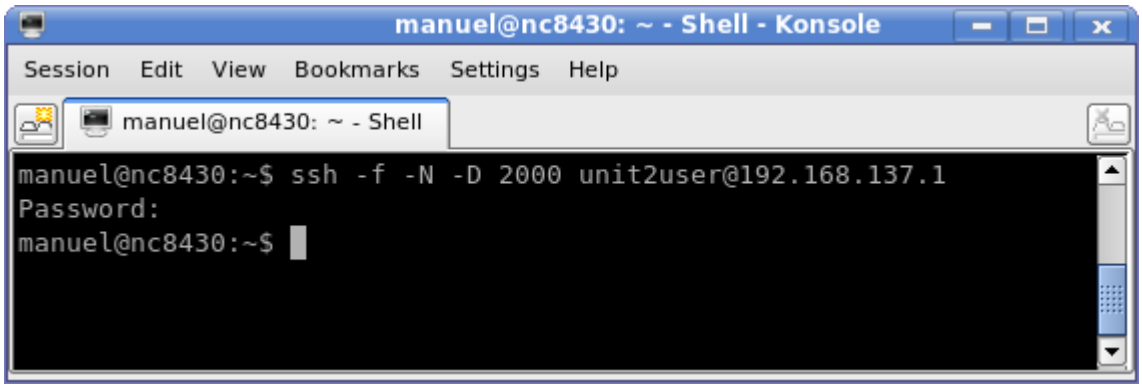


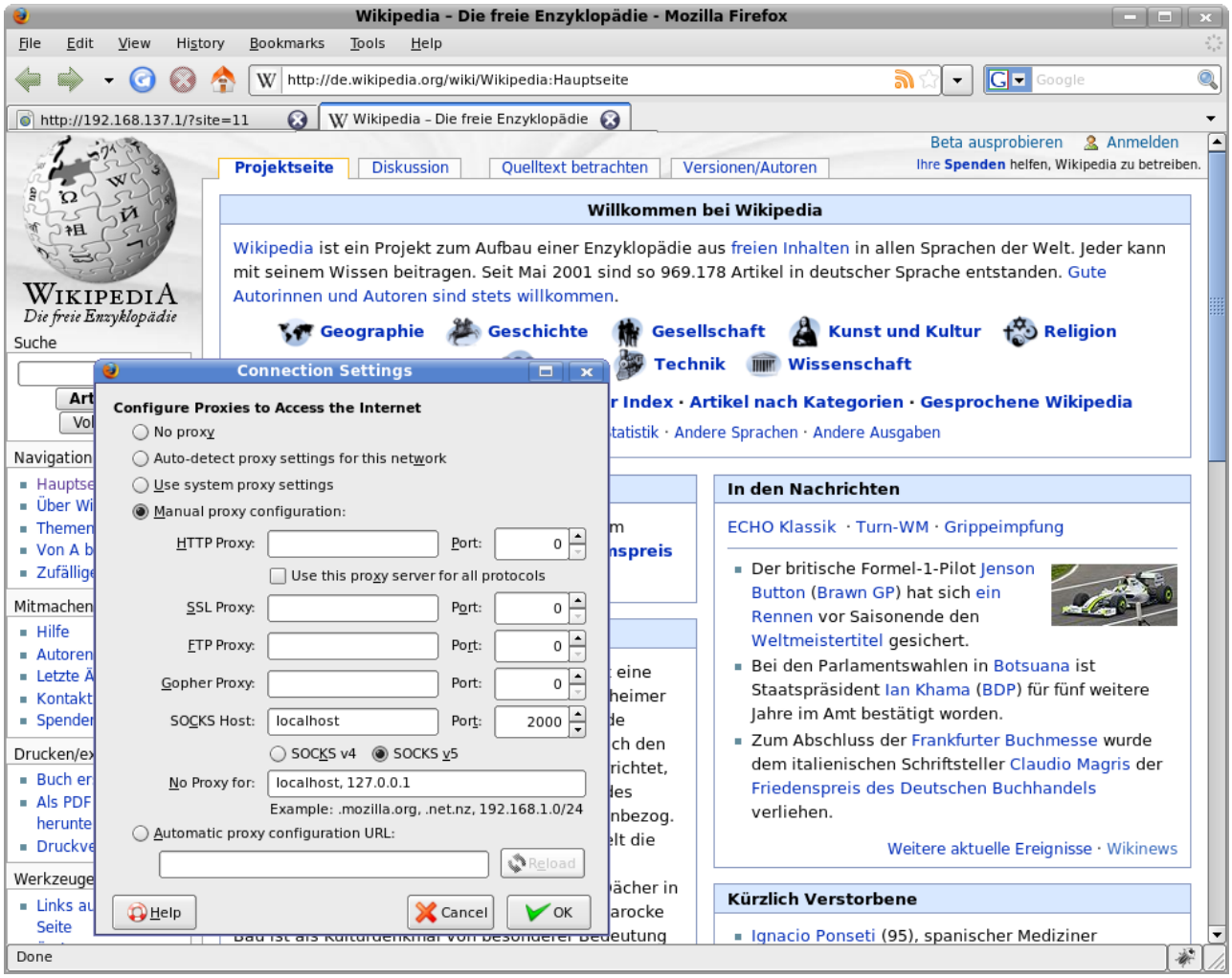
Use SSH to bypass firewall

In diesem Szenario sind abgesehen von SSH (Port 22) und DNS (Port 53) sämtliche Verbindungen gesperrt. Über den SSH Client und dessen Funktion „*dynamisches Portforwarding*“ lässt sich ein lokaler Port erzeugen, der als Proxy agiert und Verbindungen über die SSH Verbindung an den entfernten Rechner weiterleitet und Proxy-typisch von dort ausgeführt werden. Nach Eintragung des Proxys in unserem Browser, sind wir in der Lage die Firewall zu umgehen.

SSH Client Parameter: `-D [bind_address:]port`

Lösung des Beispiels:





Block SMTP using firewall rules

Bei diesem Task ist es unsere Aufgabe den Port für SMTP (Port 25) zuerst zu sperren und anschließend diese Sperre über den bereitgestellten SSH Zugang zu umgehen. Das Sperren des Ports geschieht über die im GNU/Linux-Kernel integriertes Framework bzw. Hooksystem „*netfilter*“, die Netzwerkpakete abfangen und manipulieren kann, sowie dessen Userspace-Programm „*iptables*“, das zur Konfiguration dient.

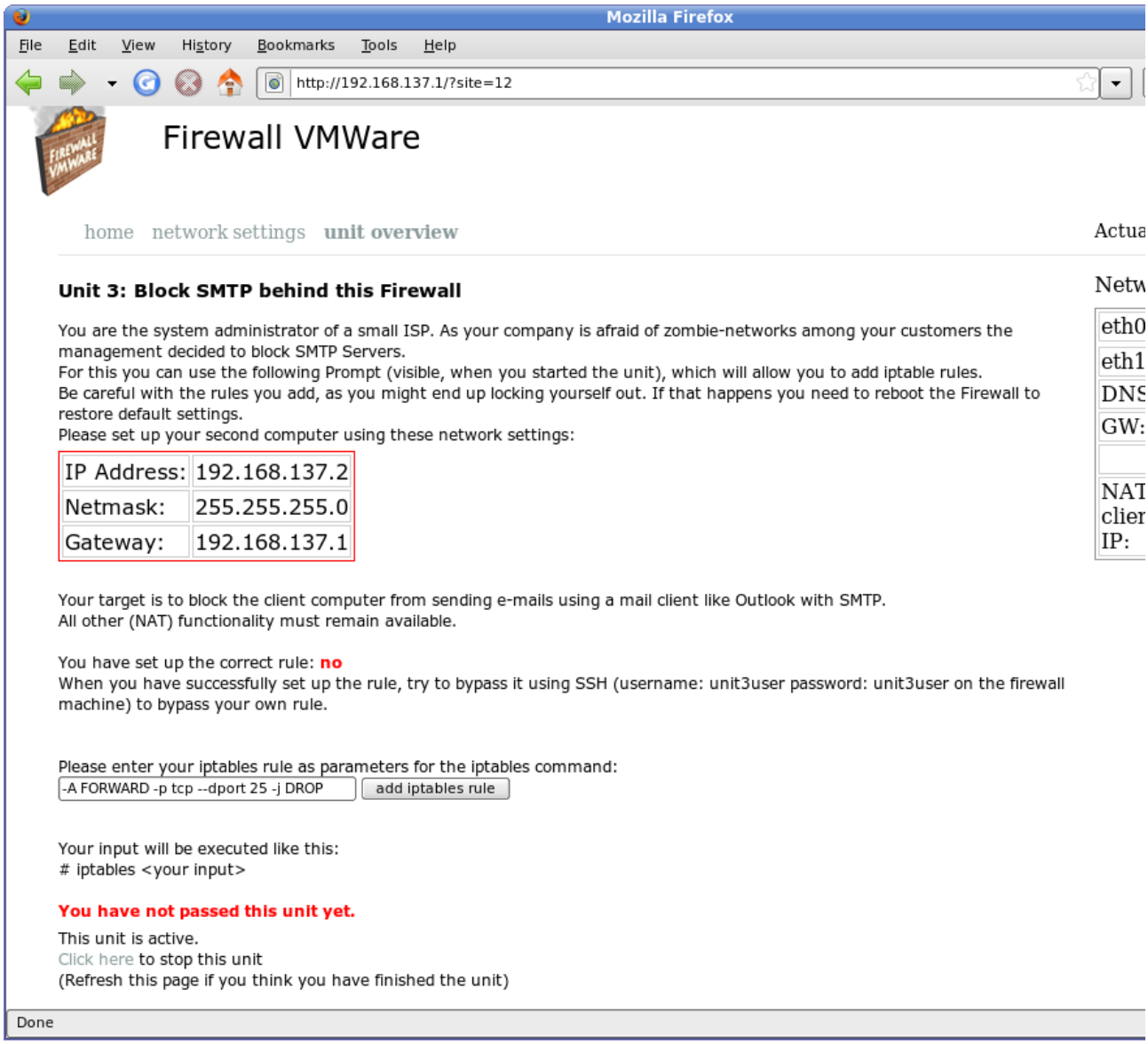
netfilter/iptables besteht im wesentlichen aus Tabellen, die einem Zweck dienen. Diese Tabelle bestehen aus einzelnen Ketten, wie beispielsweise für ein-, ausgehende und weitergeleitete Pakete. In diese Ketten können Regeln eingefügt werden, die bei Auslösung eine Aktion bzw. einen Sprung ausführen. Die Ketten werden sequentiell abgearbeitet.

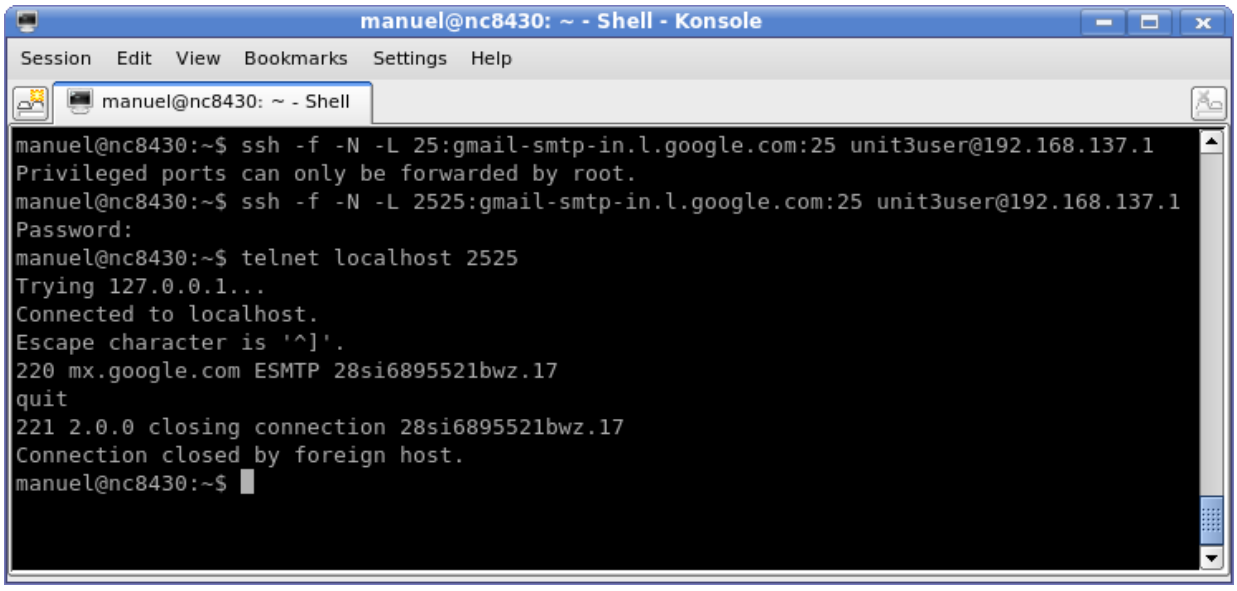
Zur Blockierung des Ports für das Protokoll TCP (SMTP verwendet ausschließlich TCP) haben wir folgenden Befehl verwendet: `iptables -t filter -A FORWARD -p tcp -dport 25 -j DROP`

`-t filter`	...definiert die Tabelle, wobei die Tabelle „*filter*“ standardmäßig genutzt wird und der Parameter somit weggelassen werden kann
`-A FORWARD`	...hängt die Regel an die Kette „*FORWARD*“ an (append)
`-p tcp`	...unsere Regel behandelt nur Pakete des TCP-Protokolls
`--dport 25`	...definiert den Zielport, den unsere Regel behandelt. Dieser muss 25 sein
`-j DROP`	...wenn alle Bedingungen zutreffen, springe zu diesem Ziel. *DROP* ist ein vordefiniertes Ziel, wodurch unser Paket nicht weiter vom System behandelt wird.

Teil 2 des Task wird wie Aufgabe #1 gelöst.

Lösung des Beispiels:





Configure a NAT

Ähnlich wie im Beispiel zuvor sollen wir mithife von netfilter/iptables arbeiten und ein funktionierendes NAT (Network Address Translation) konfigurieren und dieses testen. Um dies zu realisieren, besitzt netfilter die Tabelle NAT, die die Adressen vor dem Routing entsprechend umschreiben. Zudem besitzt die Tabelle Ketten, die vor und nach einer Entscheidung betreffend dem Routing betrachtet werden.

Um ein NAT zu konfigurieren benötigen wir 2 iptables Befehle:

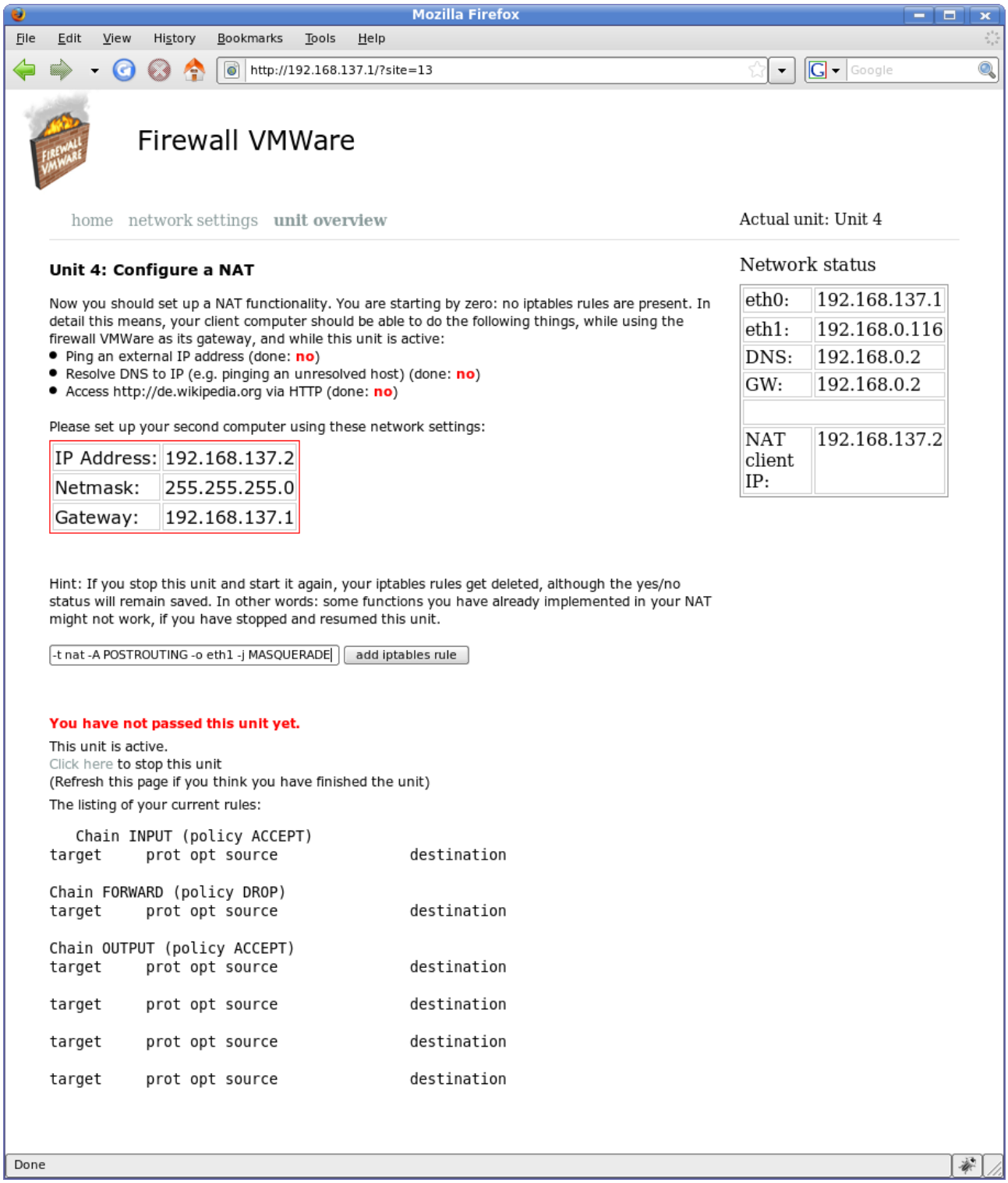
`iptables -A FORWARD -j ACCEPT`

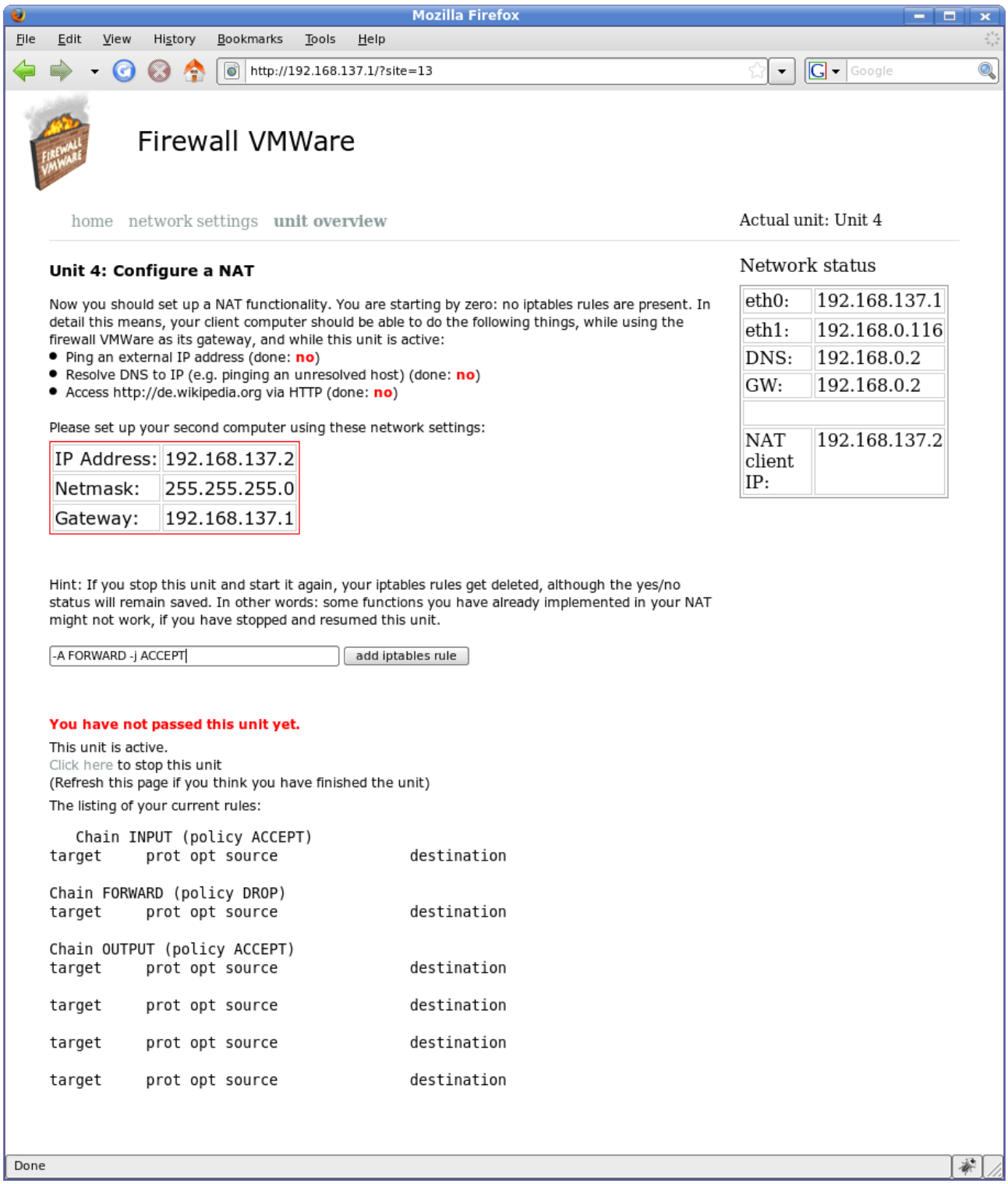
> Dieser Befehl fügt eine Regel an das Ende der Kette „*FORWARD*“ der Tabelle „*filter*“ (default) ein, der das Weiterleiten von Paketen von und zu allen Netzwerkinterfaces erlaubt. Hat man zusätzlich Netzwerkinterfaces, die Pakete nicht weiterleiten sollen, kann man diese Regel durch 2 spezifischere Regeln ersetzen (-i ethX und -o ethY).
>
> Anmerkung: Üblicherweise ist das Weiterleiten von Pakete im Linux Kernel deaktiviert. Dies muss über den Kernel-Paramater *net.ipv4.ip_forwar*d aktiviert werden, ist in dieser Testumgebung für uns aber bereits durchgeführt worden.

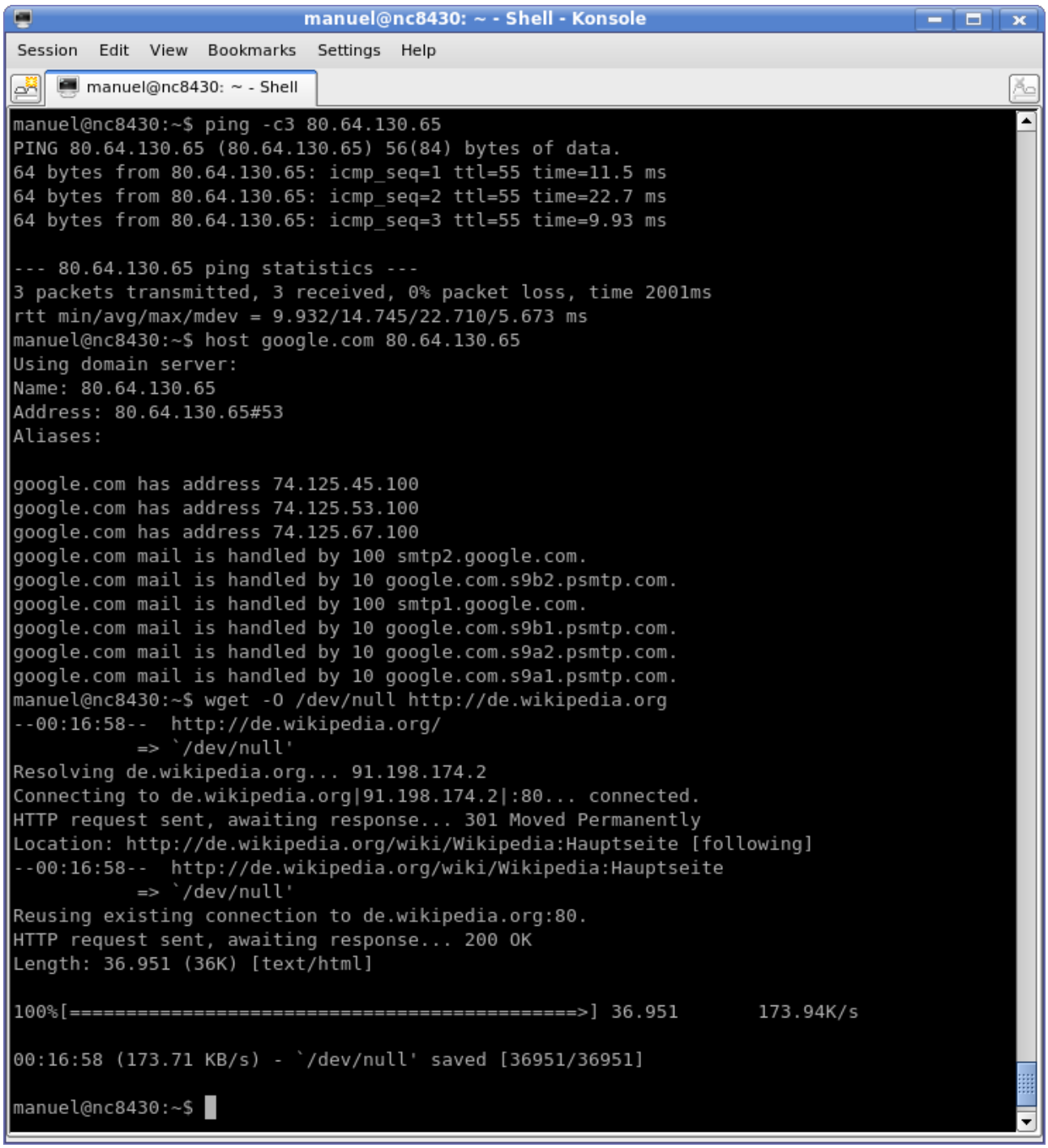
`iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE`

> Dieser Befehl fügt eine Regel an das Ende der Kette „*POSTROUTING*“ der Tabelle „*nat*“ ein, wobei alle Pakete die das Netzwerkinterface „*eth1*“ verlassen, maskieren werden. Beim Maskieren trägt die Firewall bzw. der Router seine eigene Adresse als Quelladresse ein.

Lösung des Beispiels:




Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.137.1/?site=13
Google
Firewall VMWare
home network settings unit overview
Actual unit: Unit 4
Unit 4: Configure a NAT
Now you should set up a NAT functionality. You are starting by zero: no iptables rules are present. In detail this means, your client computer should be able to do the following things, while using the firewall VMWare as its gateway, and while this unit is active:
Ping an external IP address (done: no)
Resolve DNS to IP (e.g. pinging an unresolved host) (done: no)
Access http://de.wikipedia.org via HTTP (done: no)
Please set up your second computer using these network settings:
IP Address: 192.168.137.2
Netmask: 255.255.255.0
Gateway: 192.168.137.1
Network status
eth0: 192.168.137.1
eth1: 192.168.0.116
DNS: 192.168.0.2
GW: 192.168.0.2
NAT client IP: 192.168.137.2
Hint: If you stop this unit and start it again, your iptables rules get deleted, although the yes/no status will remain saved. In other words: some functions you have already implemented in your NAT might not work, if you have stopped and resumed this unit.
-A FORWARD -j ACCEPT
add iptables rule
You have not passed this unit yet.
This unit is active.
Click here to stop this unit
(Refresh this page if you think you have finished the unit)
The listing of your current rules:
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy DROP)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
target prot opt source destination
target prot opt source destination
target prot opt source destination
Done


manuel@nc8430: ~ - Shell - Konsole
Session Edit View Bookmarks Settings Help
manuel@nc8430: ~ - Shell
manuel@nc8430:~$ ping -c3 80.64.130.65
PING 80.64.130.65 (80.64.130.65) 56(84) bytes of data.
64 bytes from 80.64.130.65: icmp_seq=1 ttl=55 time=11.5 ms
64 bytes from 80.64.130.65: icmp_seq=2 ttl=55 time=22.7 ms
64 bytes from 80.64.130.65: icmp_seq=3 ttl=55 time=9.93 ms
--- 80.64.130.65 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 9.932/14.745/22.710/5.673 ms
manuel@nc8430:~$ host google.com 80.64.130.65
Using domain server:
Name: 80.64.130.65
Address: 80.64.130.65#53
Aliases:
google.com has address 74.125.45.100
google.com has address 74.125.53.100
google.com has address 74.125.67.100
google.com mail is handled by 100 smtp2.google.com.
google.com mail is handled by 10 google.com.s9b2.psmtp.com.
google.com mail is handled by 100 smtp1.google.com.
google.com mail is handled by 10 google.com.s9b1.psmtp.com.
google.com mail is handled by 10 google.com.s9a2.psmtp.com.
google.com mail is handled by 10 google.com.s9a1.psmtp.com.
manuel@nc8430:~$ wget -O /dev/null http://de.wikipedia.org
--00:16:58-- http://de.wikipedia.org/
=> `/dev/null'
Resolving de.wikipedia.org... 91.198.174.2
Connecting to de.wikipedia.org|91.198.174.2|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://de.wikipedia.org/wiki/Wikipedia:Hauptseite [following]
--00:16:58-- http://de.wikipedia.org/wiki/Wikipedia:Hauptseite
=> `/dev/null'
Reusing existing connection to de.wikipedia.org:80.
HTTP request sent, awaiting response... 200 OK
Length: 36.951 (36K) [text/html]
100%[==========>] 36.951 173.94K/s
00:16:58 (173.71 KB/s) - `/dev/null' saved [36951/36951]
manuel@nc8430:~$

Screenshot: Alle Aufgabe gelöst