HOCHSCHULE MITTWEIDA
University of Applied Sciences

BACHELORARBEIT

Herr
Karl Büchner

Konzeption und prototypische Implementierung eines Tools zur automatischen Schwachstellenanalyse in einem Active Directory

Mittweida, 2022

Fakultät Angewandte Computer- & Biowissenschaften

BACHELORARBEIT

Konzeption und prototypische Implementierung eines Tools zur automatischen Schwachstellenanalyse in einem Active Directory

Autor:

Herr

Karl Büchner

Studiengang:

Angewandte Informatik (B.Sc.)

Seminargruppe:

IF19wI2-B

Erstprüfer:

Prof. Dr.-Ing. Uwe Schneider

Zweitprüfer:

Mag. rer. soc. oec. Dipl.-Ing. Marian Kogler

Einreichung:

Mittweida, den 16.09.2022

Verteidigung/Bewertung:

Mittweida, 2022

Faculty Applied Computer Sciences & Biosciences

BACHELOR THESIS

Conception and prototypical implementation of a tool for automatic vulnerability analysis in an Active Directory

Author:

Mr.

Karl Büchner

Course of studies:

Applied Computer Science (B.Sc.)

Seminar group:

IF19wI2-B

First examiner:

Prof. Dr.-Ing. Uwe Schneider

Second examiner:

Mag. rer. soc. oec. Dipl.-Ing. Marian Kogler

Submission:

Mittweida, den 16.09.2022

Defence/Evaluation:

Mittweida, 2022

Bibliografische Beschreibung:

Büchner, Karl:
Konzeption und prototypische Implementierung eines Tools zur automatischen Schwachstellenanalyse in einem Active Directory. (2022). Seitenzahl der Verzeichnisse: 17 Seiten, Seitenzahl des Inhaltes: 67 Seiten, Seitenzahl der Anhänge: 0 Seiten. Mittweida, Hochschule Mittweida (FH), Fakultät Angewandte Computer- & Biowissenschaften, Bachelorarbeit, 2022

In der Vorliegenden Arbeit wird der Zitationsstil: IEEE verwendet.

Satz: LaTeX 2_ε

Referat:

In der vorliegenden Bachelorarbeit wurde ein Prototyp einer Software konzipiert und implementiert, die es ermöglicht, eine Active Directory Domain nach potenziellen Schwachstellen zu durchsuchen und deren Ausnutzbarkeit zu testen. Dabei simuliert die Software das Vorgehen eines Penetrationstesters, um zu erörtern, ob ein solches Tool diesen ersetzen kann. Die Arbeit beschreibt das Vorgehen des Autors während der Programmierung dieser Software sowie die Funktionsweise des Programms. Durch die Entwicklung und Reflektion dieses Tools konnte gezeigt werden, dass menschliche Penetrationstester deutliche Vorteile gegenüber automatisierten Programmen aufweisen und solche Tools aufgrund einiger Faktoren nur unterstützend, aber nicht ersetzend, verwendet werden können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ACE	Access Controll Entity
AD DS	Active Directory Domain Service
AES	Advanced Encryption Standard
API	Application Programming Interface
CFB	Cipher Feedback
CLI	Command Line Interface
CNA	CVE Numbering Authority
CVE	Common Vulnerabilies and Exposures
DACL	Discretionary Access Control List
DCERPC	Distributed Computing Environment / Remote Procedure Call
ECB	Electronic Code Book
FEA	Full Extended Attribute
HAL	Hardware Abstraction Layer
HTTP	Hypertext Transfer Protocol
ID	Identification Digit
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
IT	Information Technology
JSON	JavaScript Object Notation
LM	Lan Manager
LSA	Local Security Authority
LSASS	Local Security Authority Subsystem Service
MVP	Minimum Viable Product
NetBios	Network Basic Input Output System
Nonce	Number used once
NSA	National Security Agency
NTLM	New Technology Lan Manager
OOB	Out Of Bounds
PDU	Protocol Data Unit
RCE	Remote Code Execution

RDP	Remote Desktop Protocol
RegEx	Regular Expression
SAM	Security Account Manager
SID	Security Identifier
SMB	Server Message Block
TCP	Transmission Control Protocol
UAC	User Account Control
VHD	Virtual Hard Disk
VSS	Volume Shadow Copy Service

1 Einleitung

Die moderne Welt ist ohne Internet und digitale Infrastruktur kaum vorzustellen, geschweige denn zu realisieren. Diese Welt bietet sowohl mit Blick auf Unterhaltung und persönlichen Nutzen als auch mit Blick auf die Wirtschaft viele Möglichkeiten und Chancen. Auch in der Arbeitswelt schreitet die Digitalisierung voran und es wird immer mehr in diese investiert [22], denn auch Unternehmen können Vorteile aus einer Vernetzung beziehen. So gaben in einer Unternehmensumfrage der IHK Berlin schon 2017 46% der befragten Unternehmen an, bereits von effizienteren Prozessen zu profitieren [29].

Doch die reine Infrastruktur, wie der Ausbau von Internetanschlüssen oder die Anschaffung von Computern in einem Unternehmen, ist nur die Oberfläche der Digitalisierung, da sie zwar die Möglichkeit schafft Prozesse und Standorte zu vernetzen, Informationen schnell auszutauschen und verfügbar zu machen, diese jedoch bei falscher oder fehlerhafter Umsetzung auch gefährdet. Es ist längst kein Geheimnis mehr, dass Cyberkriminalität das Potential hat, große Schäden an Unternehmen zu verursachen. Sei es durch Störung der Betriebsabläufe bis hin zu vollständigem Systemausfall, dem Diebstahl von Daten oder durch Erpressung von Geldern durch Verschlüsselung von Daten. So hat die Cyberkriminalität allein in Deutschland im Jahr 2020 Kosten von ungefähr sechs Milliarden Euro verursacht. [27]

Das Problem hierbei ist, dass es laufende Kosten verursacht, vorhandene Infrastruktur zu warten, zu sichern und vor allem aktuell zu halten, um sie möglichst gut gegen Angriffe bösartiger Parteien zu schützen. Leider sind nicht alle Unternehmen bereit, diese Kosten zu tragen. Die Schwierigkeit hierbei ist die stetige Veränderung der IT-Sicherheitswelt durch neu entdeckte Schwachstellen und der Weiterentwicklung verschiedener Schadprogramme, wodurch ein einmal gesichertes System in der folgenden Woche bereits wieder anfällig sein kann. Kleinere und neue Sicherheitslücken sind zwar gefährlich und beeinträchtigen die Sicherheit des Systems, jedoch nicht im selben Ausmaß wie alte, nicht-gepatchte Systeme. Diese können teilweise kritische Sicherheitslücken enthalten, welche seit mehreren Jahren bekannt sind und deren Ausnutzung in diesen Jahren perfektioniert wurde.

Ein Beispiel hierfür ist ein Erlebnis des Autors während des studentischen Pflichtpraktikums (Februar - Mai 2022) bei einem Penetrationstest. Hier wurde im Firmennetzwerk ein Computer gefunden, welcher mit dem Betriebssystem Microsoft Windows 7 (Veröffentlichung 2009, Ende des erweiterten Supports 2020 [47]) ausgestattet war. Dieser Computer war, aufgrund fehlender Updates, anfällig für den 2017 bekannt gewordenen, sogenannten EternalBlue Exploit, der dem Angreifer, einmal ausgenutzt, volle Rechte auf dem betroffenen Computer ermöglicht. Solche Maschinen bzw. Computer sind ein perfektes Einfallstor, um tiefer in ein Firmennetzwerk einzudringen, da diese meist über Verzeichnisdienste mit anderen Computern im Firmennetzwerk kommunizieren.

Oberste Priorität eines Unternehmens sollte also sein, zumindest bekannte, kritische Schwachstellen in ihrem System zu identifizieren und zu schließen, sodass ein Eindringen in das Netzwerk und die Verbreitung innerhalb diesem erschwert wird. Leider fehlen in vielen Unternehmen die Kompetenzen für eine solche Analyse des Systems und die Bereitschaft Geld für einen professionellen Penetrationstest auszugeben. Ein Indiz dafür, dass dieses Bewusstsein noch nicht ausreichend vorhanden ist, ist eine Studie von Bitkom Research aus dem Jahr 2021, wonach deutsche Unternehmen im Schnitt 7% ihres IT-Budgets für die IT-Sicherheit ausgeben [3, S. 13]. Im Gegensatz dazu wird vom Bundesamt für Sicherheit in der Informationstechnik (BSI) eine Investition von circa 20% empfohlen [9, S. 66]. Aufgrund der Diskrepanz zwischen der empfohlenen Investition und den tatsächlichen Investitionen, ist es nicht überraschend, dass die Zahl der erfassten Angriffe seit Jahren stetig steigt [14]. Daher soll im Rahmen dieser Arbeit ein Prototyp eines Programms konzipiert und implementiert werden, welches Active Directory Computernetzwerke automatisch auf einschlägige Sicherheitslücken untersucht. Dieses Programm soll sich ähnlich wie eine Malware verhalten. Untersucht werden soll dabei, ob eine solche Software einem Penetrationstester helfen oder ihn gar ersetzen kann.

Der Kontext des Active Directory wurde gewählt, da Microsoft Windows das meist genutzte Betriebssystem weltweit ist [61] und Active Directory einer der am weitesten verbreiteten Identität- und Zugriffsmanagementdienste ist [5], [38]. Zunächst werden die beiden Softwarekomponenten Windows und Active Directory umrissen und eine Zielstellung für die Entwicklung des Tools formuliert. Anschließend werden einige theoretische Grundlagen erläutert gefolgt von einer Vorstellung der Methodik beziehungsweise des Vorgehens in der Entwicklung des Tools und einer Reflektion.

1.1 Microsoft Windows

Windows ist ein Betriebssystem der Firma Microsoft. Im Rahmen dieser Arbeit ist mit Windows immer Microsoft Windows gemeint. Seinen Namen hat es ursprünglich von der damals revolutionären Idee der Fensterverwaltung (Fenster engl. *window*, plural *windows*, daher der Name Windows). Das erste Betriebssystem mit einer grafischen Oberfläche war zwar Mac OS auf dem Macintosh von Apple aus dem Jahr 1984, jedoch veröffentliche Microsoft im Folgejahr, 1985, das Betriebssystem Windows 1.0, welches ebenfalls eine grafische Oberfläche implementierte. [54] Windows 1.0 gehört zur, ab 1992 nicht mehr fortgeführten, 16-Bit Linie von Microsoft und hatte kaum Anwender [8]. Mit Windows NT 3.1 kam 1993 das erste Betriebssystem der NT-Linie auf den Markt. Diese Produktlinie wurde bis zur aktuell neusten Version, Windows 11, weiterentwickelt und ist somit seit fast 30 Jahren vertreten. Die heutige Form von Windows ist das mit Abstand meistgenutzte Betriebssystem für Personal-Computer und Workstations weltweit. [61]

Für Entwickler bietet das Betriebssystem eine API an, welche Betriebssystemfunktionen bereitstellt, die wiederum in eigene Programme eingebunden werden können, um mit dem Betriebssystem zu interagieren (z. B. um neue Dokumente zu erstellen oder alte zu löschen).

Die Bedienung des Betriebssystems und seiner Oberfläche ist anwenderorientiert, einfach und bietet viele Funktionen für Office-Tätigkeiten, aber auch für Computerspiele, Multimediaanwendungen, Dateiverwaltung und Internetnutzung. Es bietet eine Nutzerverwaltung an, wodurch mehrere Nutzer »eigene Bereiche« haben, auf die nur sie zugreifen können. Ebenso besteht die Möglichkeit mehrere Geräte in einem Netzwerk zusammenzuschließen, um sie gemeinsam zu verwalten und zum Beispiel gemeinsam genutzte Netzwerklaufwerke zu erstellen.

1.2 Active Directory

Active Directory ist ein Verzeichnisdienst, welcher ebenfalls von Microsoft entwickelt wurde. Neben anderen Funktionen wird es vor allem für die Authentifizierung von Nutzern in einem Netzwerk genutzt. Als Veranschaulichung gibt es in der Literaturquelle [21, S. 10 ff.], einen passenden Vergleich (zusammengefasst in eigenen Worten):

Angenommen man ist als Dienstleister beauftragt in einem großen Unternehmen an den Standorten neue Software zu installieren. Also fährt man an diese Standorte und meldet sich an der Rezeption an. Anschließend muss man hier seine persönlichen Infor-

mationen in einem Formular angeben und es an der Rezeption abgeben. Die Rezeption wiederum validiert, dass tatsächlich ein Auftrag vorliegt und händigt danach eine Karte zum Öffnen der Türen aus, die Zugang genau zu den Gebäuden gewährt, zu denen Zugang benötigt wird. Türen können auch nur mithilfe dieser Karte geöffnet werden. Versucht man ein anderes Gebäude zu betreten, wird der Zugang verwehrt und z. B. ein Security Mitarbeiter kann im System die Gebäude einsehen, zu welchen man Zutritt hat.

Übertragen auf einen Verzeichnisdienst, ist das Formular ein immer gleiches Set an Fragen, welches als Äquivalent zu einem Set benötigter Attribute und deren Werten angesehen werden kann. Dieses Set an Fragen reicht jedoch nicht für eine zuverlässige Verifizierung aus und somit wurde an anderer Stelle nachgefragt, ob die Identität valide ist. Ist sie das, so bekommt man eine eindeutige Schlüsselkarte, welche eine einzigartige Identität innerhalb des Systems darstellt und zentral gespeichert wird. Man kann innerhalb des Systems auch nur diese Karte verwenden um Zugänge gewährt zu bekommen, das heißt die eigene Identität muss in einer definierten Art und Weise dargestellt werden. Der Security Mitarbeiter steht in diesem Beispiel für eine Autorität im System, welche Rechte einzelner Identitäten im zentralen Speicher einsehen kann, ebenso wie das Lesegerät an den Türen, welches je nach Berechtigung die Tür öffnet oder verschlossen lässt. Über diese Identität lassen sich auch weitere Daten, wie zum Beispiel wer hat welches Gebäude wann betreten und verlassen, sammeln, um Änderungen im Nachhinein zurückverfolgen zu können.

Somit ist ein Verzeichnisdienst ein Dienst, der dafür sorgt, Nutzer, Ressourcen und Rechte zentral für mehrere Computer zu verwalten und durchzusetzen [21, S. 12]. Strukturiert wird ein solches Netzwerk hierarchisch. Die Gesamtheit des Netzwerkes nennt man einen Forest (engl. für Wald), welcher alle teilnehmenden Computer, Server und anderweitige Geräte, wie Drucker, umfasst. Ein Forest kann in verschiedene Domains unterteilt werden, um Untergruppen, wie zum Beispiel unterschiedliche Abteilungen, voneinander abzugrenzen. Dabei können Domains und Forests netzwerkübergreifend sein. Diese wiederum können in sich auch unterteilt werden. Mehrere hierarchische Domains bilden einen Tree (engl. für Baum). Jede Domain hat ihre eigenen Regeln, Anwendungen, Netzwerklaufwerke und Nutzer. Um diese zusammenzufassen werden »Organizational Units« (OU) verwendet (siehe ► Abbildung 1.1). Die standardmäßig vorhandenen OUs sind zum Beispiel User, Computer und Domain Controller.

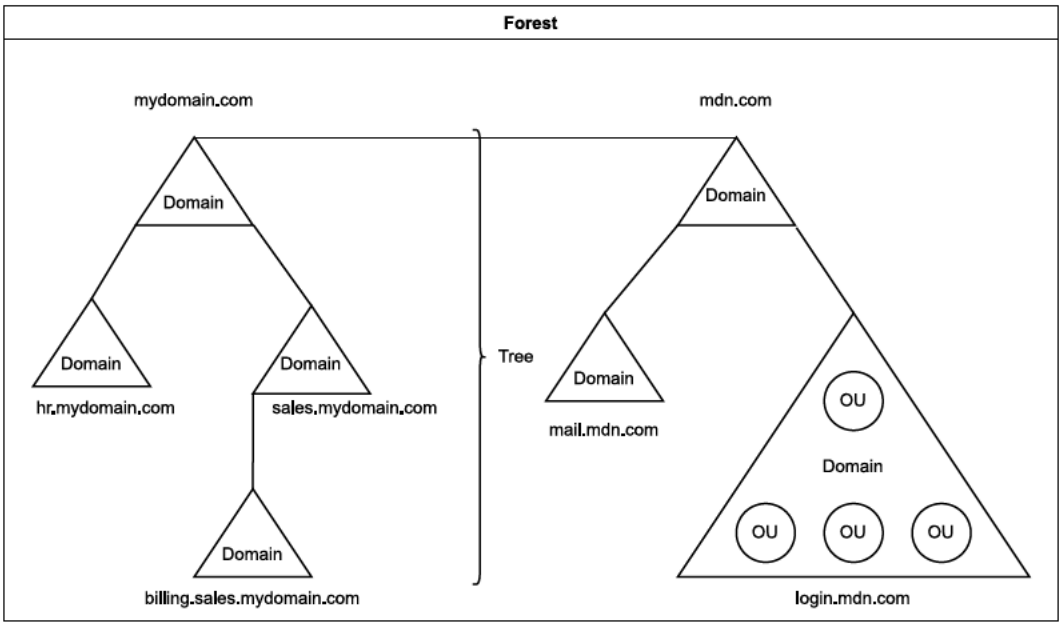

Abb. 1.1: Active Directory Forest Struktur. Eigene Abbildung.

Die zentrale Verwaltung wird durch einen oder mehrere Domain Controller realisiert, welche Computer mit einem Windows Server Betriebssystem sind, die den Active Directory Domain Service bereitstellen. Diese haben bei einem Angriff den höchsten Stellenwert, da hier alle Nutzer und Regeln gespeichert sind und der Domain Controller diese durchsetzt und durch das Netzwerk propagiert.

1.3 Malware

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) definiert Malware folgendermaßen:

> »Malware ist ein Kunstwort, das sich aus ›Malicious software‹ ableitet. Es bezeichnet Software, die mit dem Ziel entwickelt wurde, unerwünschte und meist schädliche Funktionen auf einem IT-System auszuführen. Dies geschieht in der Regel ohne Wissen des Benutzers.« [10]

Da es unterschiedliche Malwarearten gibt, welche unterschiedliche Vorgehensweisen, Infektionswege und Schadwirkungen haben, wird Malware grob in die Kategorien Trojaner, Wurm und Virus unterteilt.

Ein Trojaner ist eine Malware, welche sich tarnt, um ausgeführt zu werden. Tarnmöglichkeiten sind hierbei z. B. das Nachahmen eines validen Programmes oder das Verstecken in Dokumenten, bis diese geöffnet werden. Wird ein Trojaner gestartet, so

führt er, ohne Wissen des Nutzers, nicht erwünschte, schädliche Funktionen, wie z. B. das Mitlesen sämtlicher Tastendrücke, aus. Eine Malware der Trojaner-Klasse kann sich nicht eigenständig verbreiten und neue Systeme infizieren, sondern ist auf die Installation bzw. den Start durch einen Nutzer angewiesen. [11]

Ein Virus ist eine Schadroutine, jedoch kein eigenständiges Programm. Diese Routine wird in Dateien oder Programmen versteckt und aktiviert sich, wenn das infizierte Programm gestartete oder die infizierte Datei geöffnet wird. Ein Virus ist in der Lage andere Dateien auf dem Computer zu infizieren. Wird eine infizierte Datei heruntergeladen oder mittels eines anderen Mediums auf ein anderes System übertragen, so infizierte der Virus auch das System. [12], [20]

Ein Wurm ist, anders als ein Virus, ein komplettes eigenständiges Programm und benötigt kein Wirtsprogramm. Würmer besitzen eine eigene Verbreitungsroutine, um sich selbständig auf andere Systeme auszubreiten und dort ihre Schadfunktion auszuführen. [12]

Ein Beispiel für einen Wurm, der weltweit Windows Computer infizierte und verschlüsselte ist die WannaCry Malware. Diese verbreitete sich 2017 durch die EternalBlue genannte Schwachstelle in Windows 7 und befiel damit über 230.000 Computer. Einmal infiziert, verschlüsselte die WannaCry Software die Dateien des Computers und zeigte eine Lösegeldforderung zur Entschlüsselung der Daten. [33]

Die Schadsoftware Emotet ist ein Beispiel für einen Virus. Dieser verbreitet sich über infizierte E-Mail-Anhänge oder in Emails enthaltene Links. Dafür nutzt der Virus die Outlook-Harvesting Methode, bei der die Mails im Posteingang analysiert werden, um echt aussehende Antworten zu generieren, an die sich der Virus, in Form eines Dokumentes, anhängt. Wird die Datei geöffnet und somit der Virus aktiviert, agiert dieser als Dropper, das heißt es wird andere Malware nachgeladen. [13]

Ein Vertreter der Malwareklasse Trojaner ist die Pegasus Software des Herstellers NSO Group. Diese Software ist, einmal auf einem Gerät installiert, in der Lage sämtliche Kommunikation mitzuschneiden, Daten auszulesen und auch die Position bzw. den Aufenthaltsort des Gerätes zu überwachen. Zum Teil kann dieser Trojaner, ohne Interaktion vom Nutzer, auf einem Gerät installiert werden, verbreitet sich aber nicht selbstständig. [2], [1]

Die einzelnen Malwareklassen in reiner Form sind heute selten. Die meisten Malwares sind eine Kombination verschiedener Schadsoftwarearten und deren Funktionen. [31]

Ein Beispiel hierfür wäre der Virus Emotet, der sich ebenfalls wie ein Wurm verbreiten kann, indem er Passwörter in einem Netzwerk ausprobiert. [36]

1.4 Zielstellung und Anforderungen an die Software

Wie einleitend bereits erwähnt, soll das Ziel dieser Arbeit sein, ein Tool zu entwickeln, das automatisch das Netzwerk auf Sicherheitslücken untersucht. Der Fokus liegt dabei darauf, gefundene Sicherheitslücken auszunutzen, um sich im Netzwerk fortzubewegen und somit einer echten Malware funktional möglichst nahe zu kommen. Da dieses Tool ein eigenständiges Programm werden und sich automatisch durch ein Netzwerk bewegen soll, verhält es sich wie die Malwareklasse Wurm. Es wird daher in dieser Arbeit weiterführend als Wurm bezeichnet. Durch das Suchen und Ausnutzen von Sicherheitslücken ähnelt diese Malwareklasse in ihrer Strategie dem Vorgehen eines Penetrationstesters. Durch die angestrebte Funktionalität soll sich das Tool von verfügbaren, kommerziellen Schwachstellen-Scannern unterscheiden. Diese scannen meist von einem System aus das Netzwerk und gleichen gefundene Daten und Konfigurationen mit einer Datenbank ab. Der Wurm hingegen soll sich aktiv durch das Netzwerk bewegen und auf Schwachstellen sowie deren Ausnutzbarkeit testen. Unter Umständen ermöglicht dieses Vorgehen auch das Finden von fehlerhaften Konfigurationen, die noch nicht in einer Datenbank vorhanden sind, sondern durch die Ausnutzbarkeit einer Schwachstelle offengelegt werden.

Das primäre Ziel des Wurmes soll darin bestehen, sich durch die Active Directory Domain zu bewegen bis möglichst jeder Computer, einschließlich des Domain Controllers, infiziert ist und es der Wurm schafft, sich in den Kontext eines Domain Administrators zu versetzen oder dessen Anmeldedaten (Hash und/oder Passwort) zu erhalten. Dazu müssen folgende Anforderungen erfüllt werden:

Funktionale Anforderungen:

1. Auslesen von Nutzerdaten des lokalen Computers
2. Kategorisieren der erhaltenen Nutzerdaten in Nutzer, lokale Administratoren und Domain Administratoren
3. Erhöhung der eigenen Privilegien
4. Sammeln von Informationen über die Domain
    - Name des Domain Controllers
    - IP-Adresse des Domain Controllers
    - Name der Domain

5. Auslesen des Local Security Authority Subsystem Service (LSASS) und ggf. vorhandener Remote Desktop Protocol (RDP) Sessions
6. Scannen von Ports entfernter Computer im selben Netzwerk, jedoch mit Beschränkung auf festgelegte IP-Adressen
    - zwischen Zugriffen auf Ports desselben Computers mindestens 30 Sekunden warten
7. Erkennen von offenen Ports, die einem potenziell angreifbaren Dienst oder Service zugewiesen sind und somit auf eine Schwachstelle hindeuten können
8. Ausnutzen von Remote Code Execution Schwachstellen für eine Verbreitung auf andere Computer
9. Schaffen von Übertragungswegen für die Programmdatei mittels SMB-Shares
10. Melden aller erhaltenen Nutzerdaten sowie aller gefundenen Hinweise auf Schwachstellen und deren Ausnutzbarkeit bei einer zentralen Stelle

Nicht-funktionale Anforderungen:

1. Kompatibilität mit Windows 7 und Windows 10
2. Kompatibilität mit 32 Bit und 64 Bit Systemarchitekturen
3. Verhinderung einer Erkennung durch den Windows Defender als potenzielle Schadsoftware
4. Keine Schadwirkung auf Computern oder im Netzwerk

Um ein MVP (Minimum Viable Product) zu erhalten, müssen mindestens die funktionalen Anforderungen 1, 2, 4, 6, 7, 8, 9 und 10 sowie die nicht-funktionalen Anforderungen 1 und 3 grundlegend implementiert sein. Die Kompatibilität mit 32 Bit und 64 Bit Architekturen (nicht-funktionale Anforderung 2) soll angestrebt werden, ist aber optional, ebenso wie das Auslesen des LSASS und vorhandener RDP Sessions (funktionale Anforderung 5).

Da für jede Domain ein eigener Domain Controller benötigt wird und die Anzahl der gleichzeitig ausführbaren virtuellen Maschinen durch die Hardware des Autors limitiert ist, soll in der Entwicklung von einem Domain Controller ausgegangen werden, der eine Active Directory Domain verwaltet.

Die Auswahl der Exploits bzw. der zu testenden Sicherheitslücken erfolgte nach Verfügbarkeit von Quellcode und Dokumentationen. Ebenfalls wurde darauf geachtet mindestens eine Active Directory spezifische Schwachstelle zu implementieren. Es wurden zwei Schwachstellen ausgewählt, welche das Auslesen von Nutzerdaten ermöglichen sowie zwei andere, die zur Verbreitung verwendet werden können. Um den zeitlichen Rahmen für die Entwicklung des Tools einzuhalten, wurde die Anzahl der auszunutzenden Schwachstellen auf vier limitiert.

Folgende Sicherheitslücken sollen getestet und ggf. verwendet werden:

- CVE-2021-36934 (genannt SeriousSam oder auch HiveNightmare) um Nutzerdaten wie Passwort Hashes zu erhalten
- CVE-2020-1472 (genannt Zerologon, Active Directory spezifisch) um die Nutzerdaten auf dem Domain Controller zu erhalten
- CVE-2021-36936 (genannt PrintNightmare) für Verbreitung im Netzwerk
- MS17-010 (genannt EternalBlue, kombiniert mehrere Schwachstellen) für Verbreitung im Netzwerk

2 Grundlagen und Theorie

2.1 Windows und Active Directory Sicherheitsmechanismen

2.1.1 Windows Sicherheitsmechanismen

Windows 10 ist das beliebteste Betriebssystem weltweit [28] und implementiert eine Vielzahl an Funktionen, Algorithmen und Protokollen, um die Sicherheit sowohl lokal als auch in einem Netzwerk zu gewährleisten.

Zunächst bestehen Windows Versionen mit einem NT Kernel (ab Windows NT 3.1 bis zur aktuellen Version Windows 11) aus zwei Abstraktionsebenen, Kernel-Mode und User-Mode. Die Trennung dient dazu, den Betriebssystem-Kern vor Manipulation zu schützen, da dieser direkten Speicher- und Hardware Zugriff hat und innerhalb diesem über sämtliche andere Zugriffe entschieden wird. In diesem Modus werden ebenfalls die meisten Treiber ausgeführt. Alle Kernel-Mode Prozesse teilen sich einen virtuellen Adressraum, in dem jeder Kernel-Mode Prozess volle Rechte hat. Im User-Mode werden, wie der Name vermuten lässt, die Benutzerapplikationen ausgeführt. Diese werden voneinander isoliert, indem jeder Prozess seinen eigenen virtuellen Adressraum zugewiesen bekommt (durch den Kernel-Mode Code), auf den auch nur dieser Prozess zugreifen kann. Dateioperationen und Speicherzugriffe werden über die von Windows bereitgestellte API realisiert, welche als eine Art Gateway zum Kernel-Mode agiert. [64]

Grundlegend werden viele Aktionen an diversen Stellen mitgeschnitten, kategorisiert und als Events dem Windows Event Log Service übergeben [46, S. 114 f.]. Der Local Security Authority Server (LSA) kategorisiert unter anderem in Logon/Logoff, Object Access, Privilege Use, System und noch einige andere sicherheitsbezogene Kategorien, die er dem Event Log übergibt. Neben der Kategorie werden auch Auftrittszeitpunkt, der Nutzer in dessen Kontext das Event auftrat, die Quelle des Events, also welche Komponente das Event erzeugt hat, sowie das Ergebnis dieser Aktion gespeichert. [46, S. 114 f.]

Um die Nutzerdaten zu schützen, implementiert Windows das Prinzip der »Discretionary Access Control«, was bedeutet, dass je nach Identität des Anfragenden entschieden wird, ob dieser Zugriff auf ein Objekt (Ordner, Dokument, Programm etc.) bekommt oder nicht. Dafür gibt es spezielle Tokens, die gewisse Berechtigungen in sich tragen und einem Nutzer zugeordnet sind. Diese Tokens können auch sogenannte »Impersonation Tokens« sein, mit denen ein Prozess die Identität des Anfragenden ›annehmen‹ kann, um so auf die Ressourcen Zugriff zu bekommen, auf welche der Nutzer, der ›angenommen‹ wurde, Zugriff hat. [46, S. 130 f.]

Objekte, wie Dateien oder Ordner, haben jeweils einen Security Descriptor der in seinen Sicherheitsattributen den Besitzer speichert, sowie die Information, ob eine Discretionary Access Control List (DACL) für dieses Objekt vorhanden ist. Sollte eine DACL vorhanden sein, wird diese ebenfalls vom Security Descriptor gespeichert. In der DACL stehen Access Control Entities (ACEs), welche Nutzer oder Gruppen repräsentieren und entweder Zugriffe gewähren oder verbieten [46, S. 130 f.]. Wird ein Zugriff gewährt, so bekommt der Anfragende einen Zeiger auf ein ›Handle‹ für dieses Objekt, mit welcher er auf dieses zugreifen kann, jedoch nur mit den in der »Handle« gespeicherten Zugriffsmethoden [46, S. 131 f.].

Ein wichtiger Bestandteil der windowseigenen Schutzmechanismen ist, zumindest unter Windows 10 und aufwärts, der Windows Defender. Dieser ist ein mitgeliefertes Antiviren Programm, welches zum Beispiel einen Echtzeitschutz anbietet, bei dem automatisch neue und aufgerufene Programme vor ihrer Ausführung auf bekannten bösartige Funktionen überprüft und ggf. isoliert oder gelöscht werden. Dafür werden verschiedene Methoden zur Analyse angewandt. Einerseits die statische Analyse, bei der die Signaturen der Dateien mit bekannten Signaturen von Schadsoftware verglichen werden und andererseits eine dynamische Analyse, die durch gewisse Aktionen ausgelöst werden kann, die als möglicherweise schadhaft eingestuft werden. Solche Aktionen können gewisse Windows-API Aufrufe, wie zum Beispiel die Funktion `CreateProcess`, sein. Aber auch Ein- und Ausgabeoperationen, wie das Schreiben eines Dokuments, können einen Scan auslösen. Wird der Scan ausgelöst, so wird der Speicher des Prozesses analysiert und der Prozess ggf. terminiert, um eine Schadwirkung abzuwenden. [6], [63]

Somit ist der Windows Defender einer der zu bedenkenden Faktoren in der Entwicklung des Wurmes, da dieser das Potential hat, den Wurm vor oder während seiner Ausführung zu entdecken und zu entfernen. Das ist zwar in einem echten Szenario wünschenswert, aber vor dem Hintergrund eines gesamten Netzwerkscans nicht hilfreich. Außerdem sind

die meisten bösartige Schadprogramme, vor allem jene, die viel Schaden verursacht haben, meist professionell entwickelt und verschleiern ihren Schadcode. Für die Entwicklung einer erfolgreichen Malware ist dies unbedingt nötig, da diese eine Erkennung durch Antiviren-Software umgehen können muss, um ihr Ziel zu erreichen.

2.1.2 Active Directory Sicherheitsmechanismen

Die in Windows implementierten Sicherheitsmechanismen gelten auch im Kontext eines Active Directory Forests bzw. innerhalb der Domänen eines Forests. Da hier aber mehrere Computer mit verschiedenen Nutzern zu verwalten sind, benötigt man eine Netzwerkverbindung. Wie einleitend bereits erwähnt, sind Funktionen eines Verzeichnisdienstes unter anderem die Verwaltung und die Authentifizierung der Nutzern, die der Domain bzw. dem Netzwerk angehören. [7, S. 5]

Authentifizierung innerhalb einer Active Directory Domain funktioniert meist über das Kerberos Authentifizierungsprotokoll [21, S. 482 f.], wodurch zum Beispiel Single-Sign-On ermöglicht wird, sodass man sich nur am eigenen Computer mit Benutzername und Passwort anmelden muss und danach ohne weiter Eingaben des Passworts auf Netzwerkressourcen zugreifen kann. Realisiert wird diese Funktion durch den Erhalt eines ›Tickets‹ bei erfolgreicher Anmeldung. Dieses Ticket wird dann genutzt, um sich z. B. gegenüber der Netzwerkressource zu authentifizieren, auf die man zugreifen möchte. Diese Tickets haben eine zeitlich begrenzte Gültigkeit, können jedoch erneuert werden. Außerdem ermöglicht Kerberos eine Überprüfung der Identität des Kommunikationspartners. Trotz dessen das eine Kerberos Authentifizierung Vorteile, wie die Überprüfung der Identität, bietet, kann nach wie vor die NTLM Authentifizierung genutzt werden [25]. [23]

Wie bereits erwähnt, ist eine weitere Möglichkeit sich innerhalb einer Active Directory Domain zu authentifizieren das NTLM (NT LAN Manager) Protokoll, was vereinfacht die Authentifizierung über den NTLM Hash des Passwortes ist. Diese Authentifizierung wird aus Gründen der Abwärtskompatibilität angeboten um auch Clients Pre-Windows 2000 authentifizieren zu können. Es gibt zwei Versionen des NTLM Protokolls, NTLMv1 und NTLMv2. Die Unterschiede dieser beiden Protokollversionen liegen in den Felder, die übertragen werden müssen. NTLMv2 benötigt im Gegensatz zu v1 einen zusätzlichen Zeitstempel. Außerdem ist die Server Challenge, die genutzt wird um das Vorhandensein des Passworts zu überprüfen, bei NTLMv2 von variabler Länge, bei NTLMv1 immer 16 Byte. Von der Verwendung der NTLMv1 Authentifizierung wird abgeraten [40]. [56], [35], [66]

2.2 Analyse der Funktionsweise der Exploits

Im folgenden Abschnitt dieser Arbeit werden die im Wurm implementierten Exploits mit den zugehörigen Schwachstellen vorgestellt. Dabei wird auf die technische Grundlage der Exploits eingegangen und der Ablauf einer Attacke dargestellt.

Um Sicherheitslücken eindeutig identifizieren zu können wird die CVE-Nummerierung (Common Vulnerabilities and Exposures) verwendet. Diese werden von CNA's (CVE Numbering Authority) erstellt und zugewiesen. CNA´s sind meist Forschungseinrichtungen und Sicherheitsfirmen sowie Hersteller von Software. Ein Beispiel für eine CNA ist ist die Microsoft Corporation. [58] Dieser Abschnitt enthält für jede der verwendeten Sicherheitslücken die entsprechende CVE Nummer. Zur besseren Lesbarkeit wird in der vorliegenden Arbeit jedoch meist die umgangssprachliche Bezeichnung verwendet.

2.2.1 Hive Nightmare

Der Einstiegspunkt des Wurmes für eine Erhöhung seiner Privilegien ist die HiveNightmare oder auch SeriousSAM genannte Schwachstelle mit der CVE Nummer CVE-2021-36934 [50].

Diese Sicherheitslücke ermöglicht es normalen Nutzern die Dateien im Ordner »%windir%\system32\config« zu lesen, in dem unter anderem die System Hives SAM, SECURITY und SYSTEM gespeichert werden. Diese Dateien enthalten sensible Nutzerdaten, wie Nutzername und zugehörigen NTLM (NT Lan Manager) Hashes. Mithilfe dieser Dateien ist es möglich, seine eigenen Privilegien durch eine Pass-the-Hash Attacke zu erhöhen.

Damit diese Schwachstelle jedoch ausgenutzt werden kann, muss der Systemschutz des Computers aktiviert sein. Dies ermöglicht, dass der VSS (Volume Shadow Copy Service) Sicherungen dieser Dateien erstellt. Hintergrund davon ist, dass die entsprechenden Dateien vom System geöffnet sind während Windows ausgeführt wird. Somit kann auf diese Dateien nicht direkt zugriffen werden. Ist aber der Systemschutz aktiviert, gibt es wahrscheinlich eine Sicherungskopie dieser Dateien, welche nicht vom System geöffnet und somit nicht für andere Nutzer blockiert sind. Diese Sicherungspunkte müssen jedoch erstellt worden sein. Aus diesen können dann die Nutzerdaten extrahiert werden, selbst wenn man sich im Kontext eines unprivilegierten Nutzers befindet. Wurde der Systemschutz aktiviert, aber kein Sicherungspunkt erzeugt, ist die Schwachstelle nicht ausnutzbar. [19]

2.2.2 Zerologon

Der zweite, durch den Wurm verwendete, Exploit nutzt die Zerologon Schwachstelle aus. Diese ermöglicht eine Erhöhung der Privilegien durch Überschreiben des Maschinen-Accountpassworts des Domain Controllers. Dadurch ist der Angreifer in der Lage, alle auf dem Domain Controller vorhandenen Nutzerdaten auszulesen. Die Schwachstelle mit der CVE Nummer CVE-2020-1472 [49] liegt in der Netlogon Implementation, genauer in der Implementierung des Verschlüsselungsalgorithmus.

Das Netlogon Protokoll ist dafür zuständig, Anmeldungen in einem Active Directory zu realisieren und die entsprechenden Session Schlüssel für diese Anmeldung zu generieren [42, S. 11]. Hierfür nutzt es ein Verfahren mit Herausforderungen (Challenges), die sich Client und Server gegenseitig stellen. Dafür wird initial von beiden Parteien eine sog. Nonce (»Number used once«), bestehend aus acht zufälligen Bytes, an die andere Partei übertragen. Diese verrechnen dann jeweils die erhaltene Nonce mit der eigenen und einem gemeinsamen Geheimnis, was durch das Passwort des Computers, der sich verbinden möchte, repräsentiert wird. Diese Werte werden mit dem HMAC-SHA256 Algorithmus gehasht. Die ersten 16 Byte des Ergebnisses stellen den Session Schlüssel dar. Da beide die Nonce des jeweilig anderen erhalten haben und ihre eigene Nonce kennen, sind die Schlüssel auf beiden Seiten gleich und können für eine weitere Verschlüsselung der Kommunikation genutzt werden. Um den Client final zu verifizieren, muss dieser seine initial gesendete Nonce mit dem Session Schlüssel verschlüsseln und an den Server senden. Dieser entschlüsselt und überprüft diese Nachricht und kann so verifizieren, dass der Client den richtigen Schlüssel besitzt. [42, S. 102-110]

Für diese Verschlüsselung wird der AES-128 (Advanced Encryption Standard) Verschlüsselungsalgorithmus im CFB8 (CFB – Cipher Feedback) Modus verwendet [42, S. 106]. Der Vorteil des CFB8 Modus gegenüber z. B. dem ECB Modus (ECB – Electronic Code Book), liegt darin, dass er es ermöglicht, beliebig lange Daten, ohne ein Padding, zu verschlüsseln. Im ECB Modus hingegen müssen immer Blöcke von 16 Bytes verschlüsselt werden. Im CFB8 Modus werden immer nur 8 Bit (1 Byte) mit einem Byte aus dem Schlüssel verrechnet und dann mittels XOR auf das Byte der zu verschlüsselnden Daten angewandt, beginnend mit den vier Initialisierungsvektor-Bytes. Hier liegt auch das Problem des Algorithmus, denn in der Microsoft Implementation wird der Initialisierungsvektor für diese Verschlüsselung immer auf Null gesetzt [42, S. 107]. [18]

Mit diesem Wissen kann berechnet werden, dass in durchschnittlich einem von 256 Fällen die Verschlüsselung von Null-Bytes (`0x00`) (Klartext) mit einem Initialisierungsvek-

tor bestehend aus Null-Bytes in einem verschlüsselten Text resultiert, welcher nur aus Null-Bytes besteht. Ursache dafür ist, dass durch das AES Verfahren die Wahrscheinlichkeit Null-Byte als Ergebnis einer Verschlüsselungsoperation zu erhalten, bei 50% liegt. Somit liegt die Wahrscheinlichkeit für die ersten 8 Bit nur Nullen zu erhalten bei 0,0039% (1/256). Tritt dies ein, werden ab dieser Stelle nur Null-Bytes mit Null-Bytes über XOR verrechnet und es entsteht der verschlüsselte Text bestehend aus nur Nullen. [18]

Da die initiale Challenge nur aus Nullen besteht, ergibt, in durchschnittlich einem von rund 256 Fällen, die mit dem Session Schlüssel verschlüsselte Client Challenge einen Geheimtext aus nur Nullen. Wird 256-mal die Null-Nonce (Client Challenge) gesendet, direkt gefolgt von einem aus Nullen bestehenden `NetrServerAuthenticate3` Aufruf, ist die Authentifizierung daher wahrscheinlich erfolgreich (siehe ▶ Abbildung 2.1). [18]

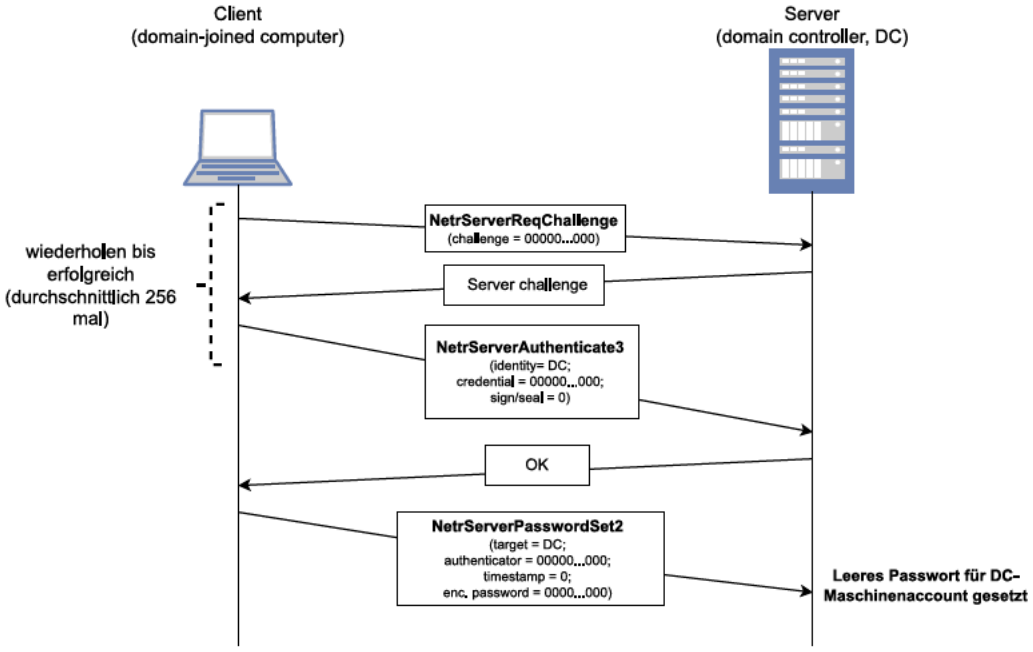

Abb. 2.1: Ablauf Zerologon. Eigene Abbildung, in Anlehnung an: [62].

Mit dieser Session auf dem Domain Controller lassen sich diverse Funktionen aufrufen. Eine davon ist die `NetrServerPasswordSet2` Methode, die das Passwort des Servers ändert und aufrufbar ist, ab dem Moment, wo der Netlogon Credential Computation Check erfolgreich ist [42, S. 128 f.].

Für einen erfolgreichen Aufruf dieser Funktion muss noch ein Hindernis überwunden werden, da in dieser Funktion zwei Werte gesetzt sein müssen. Zum einen ein 516 Byte großer Puffer, indem das neue Passwort und die Länge des neuen Passworts übertra-

gen werden. Dieser wird mit zufälligen Daten aufgefüllt, um die Größe von 516 Byte zu erreichen. Zum anderen muss die ursprüngliche von Client gesendete Nonce, erweitert um die aktuelle Zeit in Unix Epochenform und verschlüsselt mit dem Session Schlüssel, gesetzt werden. Das erste Feld Null zu setzen ist leicht zu erreichen. Als neues Passwort werden nur Null-Bytes gewählt, der Puffer mit Null-Bytes aufgefüllt und eine Passwortlänge von Null gesetzt. Dies verschlüsselt mit nur Null-Bytes, ergibt nur Nullen und ist somit, zumindest im Sinne des Exploits, valide. Hier findet serverseitig keine Validierung statt, sodass man das Active Directory Passwort einfach leer lassen kann und somit löscht. [18], [42, S. 108]

Das zweite Feld stellt eine größere Herausforderung dar, da es einen Zeitstempel enthält. Durch eine Validierung des Zeitstempels würde spätestens auffallen, dass die Verbindung nicht vertrauenswürdig ist. Da die Zeitangabe in Unix-Epochenformatierung die Sekunden seit Beginn der Unix Epoche (01.01.1970, 00 Uhr 00 Minuten 00 Sekunden [18], [42, S. 32 und S. 126 f.] angibt, kann diese nie komplett aus Nullen bestehen und somit auch keinen Geheimtext nur aus Null-Bytes erzeugen. An dieser Stelle würde auffallen, dass der richtige Session Schlüssel nicht vorhanden ist und deshalb keine Werte korrekt verschlüsselt werden können, die nicht nur aus Nullen bestehen. Aber auch hier gibt es keine serverseitige Validierung des Datums und der Uhrzeit, was weiterführend ermöglicht als Zeitstempel Sekunde Null der Unix-Epoche anzugeben. Das Setzen des Zeitstempels auf Null resultiert in einem aus nur Nullen bestehenden Klartext. Die Verschlüsselung dieses Textes erzeugt wieder einen nur aus Null-Bytes bestehenden verschlüsselten Text. Zwar liegt der Zeitstempel über 50 Jahre in der Vergangenheit, jedoch wird das nicht validiert und somit ist es möglich, das Passwort des Domain Controllers zu löschen. Anschließend kann, unter Nutzung des leeren Passworts, eine Verbindung mit vollen Rechten hergestellt werden, um Befehle auszuführen oder Daten auszulesen. [18]

2.2.3 EternalBlue

EternalBlue ist eine Remote Code Execution (RCE) Schwachstelle im Windows Betriebssystem. Entdeckt wurde sie von der NSA (National Security Agency, Nachrichtendienst der USA), die auch einen Exploit dafür entwickelte. Dieser wurde 2017 von einer Hackerguppierung Namens »The Shadow Brokers« veröffentlicht. [15]

Diese Schwachstelle befindet sich in der Implementation des SMBv1 Protokolls (Server Message Block) und ermöglicht die Ausführung von eigenem Code mit SYSTEM Berechtigung. SYSTEM Berechtigungen sind dabei die Privilegien des SYSTEM-Accounts, welcher vom Betriebssystem selbst und diversen Services genutzt wird und standardmäßig

alle Berechtigungen besitzt [37]. Microsoft veröffentlichte hierzu das Security Bulletin MS17-010 [4], in welchem verschiedene CVE Nummern angegeben werden, die unterschiedliche Sicherheitslücken bezeichnen und kombiniert zur Remote Code Execution führen. Die CVE-Nummern mit Bezug zur Remote Code Execution sind CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146 sowie CVE-2017-0148.

Die RCE wird durch das Kombinieren von drei Bugs erreicht. Diese drei verschiedenen Bugs werden im folgenden Abschnitt vorgestellt und erklärt, um so eine Vorstellung zu vermitteln, wie der Exploit funktioniert.

2.2.3.1 Bug 1 – Falsche NTFeaListSize Berechnung

FEA bedeutet »Full Extended Attribute« und ist eine Struktur, mit der zusätzliche Informationen, in Form von Key-Value Paaren, zu einem Dokument übertragen werden. [41, S. 44 f.]. `FEAs` werden in einer Struktur Namens `Os2FeaList` übertragen, diese wird auf dem Ziel-Computer jedoch in eine `NTFeaList` Struktur konvertiert, welche anschließend in den nonpaged Kernel Pool geschrieben wird [26]. In der `Os2FeaList` Struktur wird die Größe der `FEA` Liste, in einem Feld `SizeOfListInBytes` abgelegt, die transferiert wird. Dieses Feld wird auf dem Ziel-Computer von der Funktion `SrvOs2FeaListToNt` benutzt, um die Größe der `NTFeaList` zu berechnen, in die die Struktur umgewandelt wird. Diese Größe wird auch in diesem Feld gespeichert (der alte Wert wird überschrieben). Diese Funktion validiert weiterführend, dass nicht mehr `FEA` Einträge gesendet wurden als die `SizeOfListInBytes` angibt, sodass kein halber `FEA` Eintrag geschrieben wird. Sollte ein Eintrag nicht ganz in die angegebene Größe passen, berechnet die Funktion die Größe neu, sodass dieser Eintrag abgeschnitten wird (der angefragte Speicher wird kleiner). In dieser Funktion liegt der erste Bug. Werden mehr Einträge als (2^16)-1 (`0xffff`) Bytes und die `SizeOfListInBytes` ist größer als `0xffff` gesendet, so kommt ein Implementationsfehler in der Berechnung zum Tragen, welcher `SizeOfListInBytes` als ein WORD behandelt (2 Bytes), obwohl dieses Feld ein DWORD (Double Word, 4 Bytes) ist [26]. Wird beispielsweise initial eine `Os2FeaList` übertragen, deren `SizeOfListInBytes` einer Größe von `0x10000` entspricht und deren Größe umgewandelt und verkleinert `0xff5d` ergeben würde, entsteht dadurch, dass nur das LOWORD (die geringwertigen zwei Byte) geändert wird, eine Größe von `0x1ff5d` [26]. Somit wird, statt die Größe zu verringern, ein größerer Wert zurückgegeben als der Puffer eigentlich groß ist. Dadurch kommt es zu einem »Out of Bound« (außerhalb der Grenze/Beschränkung) Schreibvorgang (siehe ► Abbildung 2.2) [26]. Auf den Nutzen dieses Bugs wird später in diesem Abschnitt eingegangen.

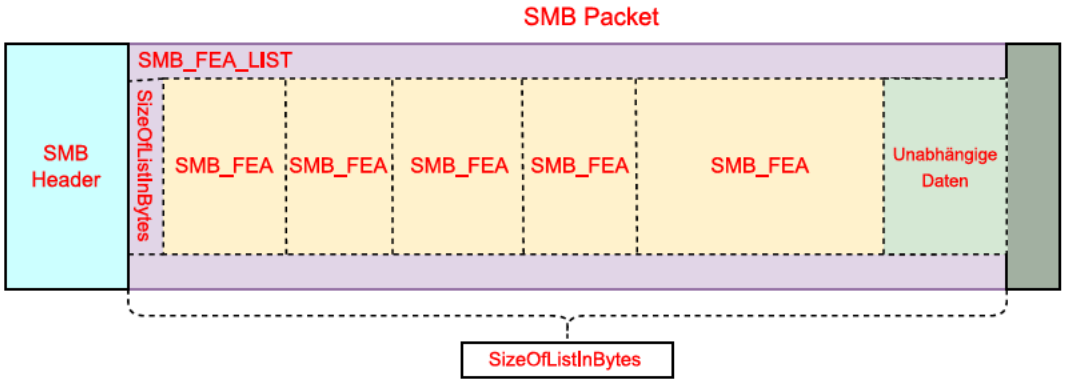

Abb. 2.2: Speicherallokation nach falscher `NTFeaList` Berechnung. Eigene Abbildung, in Anlehnung an: [26].

2.2.3.2 Bug 2 – Fehlende Methodenvalidierung

Dieser Bug ermöglicht die Ausnutzung des ersten Bugs. Es gibt im SMB Protokoll verschiedene Methoden, um Dateien zu übertragen. Diese Methoden heißen `SMB_COM_TRANSACTION2` und `SMB_COM_NT_TRANSACT` [41, S. 252 f. und S. 325 f.]. Beide bieten Subaufrufe `<Methodenname>_SECONDARY` an, bei denen mehrere Pakete nacheinander geschickt werden können. Mit dem `NT_TRANSACT` Kommando werden Dateien übertragen, welche die `MaxBufferSize` übersteigen. Diese werden dann aufgeteilt und mit den `_SECONDARY` Methoden nacheinander gesendet [41, S. 261 f. und S. 334 f.]. In der `TRANSACTION2` Methode wird der `MaxDataCount` Wert in einem WORD (2 Byte, USHORT), bei der `NT_TRANSACT` Methode in einem DWORD (4 Byte, ULONG) übertragen (siehe ▶ Abbildung 2.3).

Das Problem hierbei ist, dass die ursprüngliche Methode nicht gespeichert wird. Das heißt, es ist möglich diese beiden Methoden vermischt zu nutzen, ohne dass dies zu einem Serverfehler führt. Dabei wird die Konvertierung der Strukturen anhand der zuletzt benutzten Methode vollzogen. So kann man, indem man initial ein `NT_TRANSACT` sendet, gefolgt von `TRANSACTION2_SECONDARY`, Bug 1 ausnutzen, da in der `TRANSACTION2_SECONDARY` Methode `MaxDataCount` in Form eines WORDs angegeben wird und somit serverseitig das DWORD der ursprünglichen `NT_TRANSACT` Methode als WORD behandelt wird. [26]

NT_TRANSACT

```
SMB_Parameters
  {
  UCHAR WordCount;
  Words
    {
    UCHAR  MaxSetupCount;
    USHORT Reserved1;
    ULONG  TotalParameterCount;
    ULONG  TotalDataCount;
    ULONG  MaxParameterCount;
    ULONG  MaxDataCount;
    ULONG  ParameterCount;
    ULONG  ParameterOffset;
    ULONG  DataCount;
    ULONG  DataOffset;
    UCHAR  SetupCount;
    USHORT Function;
    USHORT Setup[SetupCount];
    }
```

NT_TRANSACTION2

```
SMB_Parameter
  {
  UCHAR WordCount;
  Words
    {
    USHORT TotalParameterCount;
    USHORT TotalDataCount;
    USHORT MaxParameterCount;
    USHORT MaxDataCount;
    UCHAR MaxSetupCount;
    UCHAR Reserved1;
    USHORT Flags;
    ULONG Timeout;
    USHORT Reserved2;
    USHORT ParameterCount;
    USHORT ParameterOffset;
    USHORT DataCount;
    USHORT DataOffset;
    }
```

Abb. 2.3: Auszüge Anfrage-Strukturen. Eigene Abbildung, in Anlehnung an: [41, S. 252 f. und S. 326].

2.2.3.3 Bug 3 – Falsche Daten Extraktion

Wird eine SMB Verbindung aufgebaut, so wird als erstes mittels `SMB_COM_SESSIONS_SETUP_ANDX` die Authentifizierung gestartet. Hier können zwei Modi verwendet werden, die dem Server in einem Feld namens `WordCount` (bei LM und NTLM 13, bei NTLMv2 12), der `SMB_Parameters` Struktur mitgeteilt werden. [26]

Erhält der Server dieses Paket, so werden die Strukturen `SMB_Data` und `SMB_Parameters` extrahiert und das Paket wird validiert. Dafür vergleicht die Validierungsfunktion verschiedenen Flags. In dieser Funktion existiert jedoch ein Fehler. Durch diesen wird, sollte eine spezielle Flag nicht gesetzt sein (`FLAGS2_EXTENDED_SECURITY`), ein LM bzw. NTLM Authentifizierungsversuch als NTLMv2 gewertet. Das führt weitergehend dazu, dass `ByteCount` von einem falschen Offset gelesen wird und somit durch Manipulation dieses Feldes, mit einem kleinen Paket ein großer Speicherbereich angefragt und belegt werden kann. [26]

2.2.3.4 Ablauf einer EternalBlue Attacke

Als erstes wird die Session aufgebaut und die ersten `FEA` Einträge werden mittels `NT_TRANSACT` gesendet, jedoch nicht alle, um die, in Bug 1 beschriebene, Konvertierung in `NTFEA` und damit den Schreibvorgang in den nonpaged Kernel Pool nicht zu starten. Dadurch wird für die Verbindung eine `SRVNET` Struktur abgelegt. Anschließend werden mit

der _SECONDARY Funktion alle bis auf den letzten FEA Eintrag gesendet, sodass der OOB (Out of Bounds) Schreibvorgang noch nicht ausgeführt wird. Jetzt müssen mehrere neue Verbindungen geöffnet werden, um mehr SRVNET Strukturen in den Speicher zu schreiben, in der Hoffnung, dass eine dieser Strukturen genau hinter der SRVNET Struktur der ersten Verbindung liegt. Nun wird mittels Bug 3 ein Speicherbereich reserviert, der der Größe der NTFeaList (deren Größe falsch berechnet wurde durch Bug 1) entspricht. Anschließend werden noch mehr Verbindungen geöffnet, auch hier in der Hoffnung, dass sich eines der dann erzeugten SRVNET Strukturen hinter dem reservierten Speicherbereich befindet und später durch den Overflow überschrieben werden kann. [26]

Danach wird die Verbindung, die den großen Speicherbereich mittels Bug 3 reserviert hat, beendet, um diesen Bereich freizugeben und so ein ›Loch‹ zu erzeugen. Anschließend wird, mit der TRANSACTION_2_SECONDARY Methode, der letzten FEA Eintrag gesendet, um so die Konvertierung der Os2Fea Einträge zu NTFEA Einträgen auszulösen. Dadurch, dass das ›Loch‹ der Größe entspricht die (geglaubt wird) zu benötigen, werden die NTFEA Einträge in dieses Loch hineingeschrieben. Durch Bug 1 wird aber weitergeschrieben als Speicher reserviert ist und es werden Header Felder der folgenden SRVNET Struktur überschrieben (siehe ▶ Abbildung 2.4). Die hier überschriebene Felder sind pSrvNetWskStruct und MDL [26]. pSrvNetWskStruct ist ein Zeiger auf eine Struktur, die unter anderem einen Zeiger zu einer Funktion beinhaltet, welche ausgeführt werden soll, sobald die Verbindung beendet wurde. MDL ist ein Feld, in dem gespeichert wird, wohin Daten, die innerhalb dieser Verbindung erhalten werden, geschrieben werden sollen. [26]

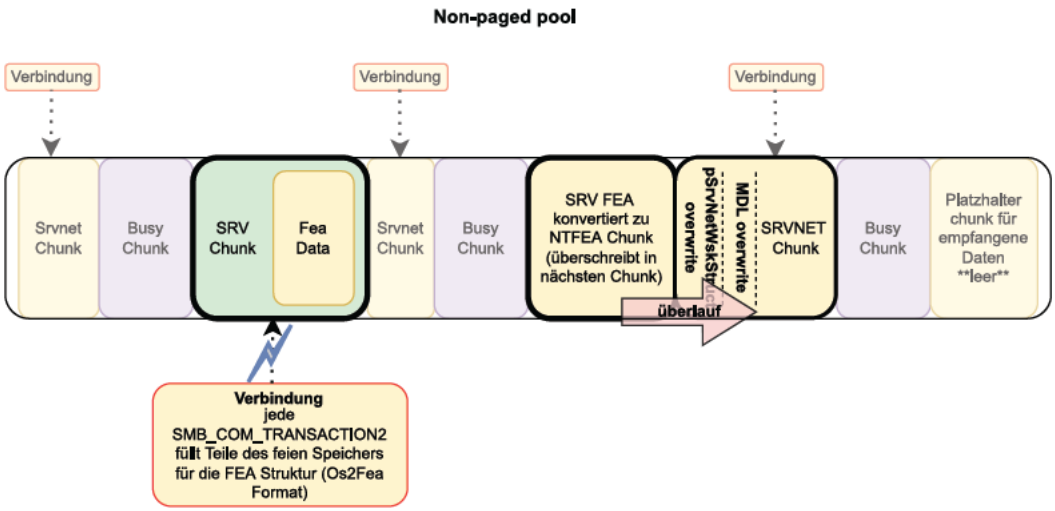

Abb. 2.4: NTFea Overflow. Eigene Abbildung, in Anlehnung an: [26].

Diese beiden Felder werden durch die Daten des letzten FEA Eintrages so überschrieben, dass sie in die HAL (Hardware Abstraction Layer) zeigen. Um diese Zeiger auf etwas verweisen zu lassen, werden in die Verbindung, deren SRVNET Struktur überschrieben wurde, eine manipulierte SRVNET Struktur und Shellcode gesendet und somit in die HAL geschrieben. Unter Shellcode wird eine Abfolge von Prozessorbefehlen verstanden, die in den Speicher eines Computers geschrieben und ausgeführt werden. Ziel dieser Befehle ist meistens, wie auch im Rahmen dieser Arbeit, das Erstellen einer Konsoleninstanz (Shell), über die Befehle ausgeführt werden können. Daher der Name Shellcode. [53]. Der Zeiger für die Funktion, welche nach Beendigung der Verbindung aufgerufen wird, wird in der gefälschten SRVNET Struktur so gesetzt, dass er auf den danach gesendeten Shellcode verweist. [26]

Zum Abschluss müssen alle Verbindungen beendet werden. Somit führt der Prozessor alle Funktionen aus, die in den Strukturen enthalten sind, auf die die Zeiger in den zugehörigen SRVNET Strukturen verweisen. So auch die Funktion, auf die in der gefälschten SRVNET Struktur gezeigt wird (Shellcode). Somit wird der Shellcode ausgeführt und die Remote Code Execution war erfolgreich. [26]

2.2.4 BlueKeep

In diesem Abschnitt wird der zweite genutzte Remote Code Execution Exploit, Bluekeep (CVE-2019-0708 [48]), vorgestellt. Er basiert auf einem Fehler im Windows Remote Desktop Protocol (RDP), welches für die Bildschirmübertragung und Fernsteuerung von entfernten Rechnern genutzt wird.

Eine Verwendung von BlueeKeep war ursprünglich nicht vorgesehen, jedoch kam es während der Implementierung von EternalBlue zu Schwierigkeiten, sodass auf diesen Exploit ausgewichen wurde. Im Abschnitt Entwicklungstagebuch wird auf die Problematik des EternalBlue Exploits genauer eingegangen.

Um die Funktionsweise dieses Exploits verstehen zu können, benötigt man das Verständnis der folgenden beiden Pakete.

2.2.4.1 Refresh Rect PDU

Dieses Paket, wird genutzt, um bei einer Bildschirmübertragung die Nachricht zu übermitteln, welche Bildschirmareale neu gezeichnet werden müssen [45, S. 222]. Innerhalb dieser Funktion können mehrere T_Rectangle16 Strukturen übergeben werden, welche jeweils vier Felder mit einer Größe von 2 Byte beinhalten. Damit werden die Koordinaten

der Rechtecke, die neu gezeichnet werden sollen, übertragen. Es können pro Struktur 8 Byte eigene Daten gesendet werden. Die Anzahl der Rechtecke, die pro Paket übertragen werden können, ist durch das Feld `numberOfAreas`, mit der Größe 1 Byte (Maximum `0xff` = 255d), limitiert [45, S. 223]. Erhält der Server ein solches Paket, werden die Rechtecke entpackt. Beim Entpacken werden pro Rechteck stets `0x828` Bytes reserviert und die Daten dort hineingeschrieben. Durch diese Größe werden die entpackten Strukturen zu Adressen von je `0x1000` ausgerichtet. Wenn z. B. an der Adresse `0x54321000` die erste Struktur mit einer Größe von `0x828` geschrieben wird, so wird die folgende Struktur an die Adresse `0x54322000` geschrieben. Die geschriebenen Strukturen enthalten jeweils an einem Offset von `0x2c` (also z. B. an der Adresse `0x5432102c`) immer die durch den Client manipulierbaren Daten. [68]

Da man dieses Paket beliebig oft senden kann und in jedem Paket `0xff` Rechtecke enthalten sein können, kann man dadurch seine Daten über einen sehr großen Adressraum im Kernel Speicher verteilen. [67]

2.2.4.2 Client Name Request PDU

Mit dieser Funktion lassen sich große Mengen Daten in den Kernel Pool schreiben. Eigentlich ist dieses Paket dafür gedacht, den Client Name abzufragen bzw. zu übertragen, um anschließend auf das Dateisystem des Clients zugreifen zu können. [67]

Für die Übertragung des Namens gibt es zwei Felder, `ComputerNameLen` und `ComputerName`, welche beide vom Client beliebig gesetzt werden können. Der Inhalt des `ComputerName` Feldes wird auf der Serverseite in den Kernel Pool geschrieben. Das Feld `ComputerNameLen` gibt dabei an, wie viel Speicher benötigt wird und bestimmt daher die Größe der Speicherreservierung. Somit bietet dieses Paket, durch eigenständiges Setzen dieser beiden Felder, die Möglichkeit beliebig viele Daten in den Kernel Pool zu schreiben. [43, S. 30]

2.2.4.3 Ablauf einer BlueKeep Attacke

Das RDP Protokoll funktioniert über verschiedene Kanäle, über die Daten übertragen werden. Während des Session Aufbaus wird, unter anderem, der Kanal `MS_T120` angelegt, der eigentlich nur für interne Zwecke genutzt wird. Dieser Kanal ist für diesen Exploit relevant, da RDP auch die Möglichkeit bietet, im Session Aufbau einen oder mehrere Kanäle selbst zu definieren. Dabei wird nicht validiert, ob dieser Kanal schon existiert. [68]

Für jeden Kanal wird im `ChannelPointerTable` des Servers ein Zeiger abgelegt, der auf das zugehörige Kanal-Objekt verweist. Definiert man nun einen Kanal im Session Aufbau und nennt diesen auch `MS_T120`, so werden zwei Pointer zum selben Objekt in dieser Tabelle abgelegt. Diesem definierten Kanal kann man über einen `MCS Channel Join Request` beitreten und dort Daten hineinsenden. Setzt man innerhalb dieser Daten an einer gewissen Stelle eine zwei, so wird das auf der Serverseite als Schließen der Verbindung gewertet und einer der beiden `ChannelPointerTable` Einträge wird gelöscht und somit der Speicherbereich freigegeben. Der zweite Zeiger bleibt jedoch bestehen. [68]

Um den freigewordenen Speicher neu mit eigenen Daten beschreiben zu können, wird ein `Client Name Request PDU` verwendet und entsprechend präpariert. Hierfür muss das Feld `ComputerNameLen` auf `0x98` gesetzt werden, um eine Speicherreservierung von `0x8c` Bytes (Größe des `MS_T120` Objektes) zu erhalten. Dabei wird einkalkuliert, dass nicht nur die reinen Daten in den Speicher geschrieben werden, sondern auch der `Client Name Request PDU` Header mit einer Größe von 16 (`0x10`) Byte sowie 32 (`0x20`) Byte für interne Funktionen berücksichtigt werden müssen. Der Header und der Speicherplatz für interne Funktionen sind vom Client nicht manipulierbar, werden aber zwangsläufig geschrieben. Die Größe der in den Speicher geschriebenen Daten setzt sich nach folgender Formel zusammen: (`0x20` (interner Nutzen) + `0x10` (Header) + `0x98` (eigene Daten)) = `0xc8`. Dieses Paket wird mehrfach gesendet, um die Wahrscheinlichkeit zu erhöhen, dass der Inhalt eines dieser Pakete in den Speicherbereich geschrieben wird, wo vorher das `MS_T120` Objekt stand und auf den der im `ChannelPointerTable` verbliebene Zeiger verweist. Mit diesem Request wird in den freien, ehemals durch das `MS_T120` Kanalobjekt belegten, Speicher ein manipuliertes `MS_T120` Kanalobjekt geschrieben, dass unter Anderem den Stage 1 Shellcode enthält sowie die Adresse eines Zeigers auf den Stage 0 Shellcode, welcher durch die Manipulation anderer Felder aufgerufen wird. [68]

In diesem Exploit wird der Shellcode in verschiedenen Stufen (Stages) übertragen. Das dient dafür, den gesamten Shellcode ausführen zu können, auch wenn an der in der Ausführung erreichbaren Stelle nur wenig Daten geschrieben werden können. Dabei werden die einzelnen Stages so geschrieben, dass sie die jeweils folgende Stage ausführen.

In der auf der Serverseite verarbeitenden Funktion gibt es eine Route, in der ein Funktionszeiger aus dem Kanalobjekt aufgerufen bzw. ausgeführt wird. Um diese Stelle zu erreichen, müssen im manipulierten `Client Name Request PDU` verschiedenen Felder richtig gesetzt werden. Das Ziel ist es, einige Überprüfungen zu täuschen und den Aufruf des Zeigers zu erreichen, der in diesem gefälschten `MS_T120` Objekt an einem Offset von

0xc8 steht. Dieser Zeiger wird so gesetzt, dass er auf einen Offset von 0x30 in einem der Speicherbereich zeigt, der durch die Refresh Rect PDU´s geschrieben wurde. Diese Adresse verweist auf die zweiten 4 Byte der 8 Byte, welche im Refresh Rect PDU übertragen wurden. Der Wert an dieser Adresse wiederum verweist auf den Offset von 0x2c, also 4 Byte vorher. Hier steht der Stage 0 Shellcode des Exploits (bestehend aus den ersten 4 Byte der Refresh Rect PDU, die zwei Anweisungen darstellen). [68]

Zum besseren Verständnis des folgenden Abschnitts wird in diesem Absatz kurz die Funktion von Prozessorregistern angerissen. Shellcode besteht, wie bereits erwähnt, aus Prozessoranweisungen. Prozessoren arbeiten mit Registern, in denen sie die Daten, mit denen sie rechnen, speichern. Dabei kann ein Register, z. B. das eax Register, nochmals in zwei kleinere Register (al und ah) unterteilt werden, wobei al die niederwertigen Bytes enthält und ah die höherwertigen. Analog dazu kann das ebx Register unterteilt werden. Auf die Register wird durch ihren Namen zugegriffen. Ein Beispiel hierfür ist der Befehl mov eax, 100. Dieser schreibt den Wert 100 in Binärdarstellung in das Register eax. Ergebnisse von Berechnungen werden in das erste genannte Register geschrieben.

Die erste Anweisung (00c3) steht für add bl, al, wodurch die Adresse des Stage 1 Shellcodes, der im gefälschten MS_T120 Objekt steht, berechnet und in das ebx Register gespeichert wird. Das funktioniert, da durch den aufgerufenen Funktionszeiger eine Adresse (im eax Register) aufgerufen wird, die auf 0x30 endet. Somit steht in al der Wert 0x30. Im ebx Register steht die Adresse des gefälschten MS_T120 Objektes und dadurch entsteht durch die Berechnung die Adresse des gefälschten MS_T120 Objekts an einem Offset von 0x30, was der Start der vom Client definierbaren Daten ist. Hier steht Stage 1 Shellcode. Die zweiten 2 Byte (ffe3) stehen für die Anweisung jmp ebx, wodurch an genau die berechnete Stelle gesprungen und die Ausführung fortgesetzt wird. [68]

Als letztes muss der Stage 2 Shellcode, ebenfalls mittels Client Name Request PDU, übertragen werden. Der Stage 1 Shellcode im gefälschten MS_T120 Objekt erfüllt in diesem Kontext eine Funktion als »egg hunter«. Das heißt er sucht über verschiedene Register die Adresse des Stage 2 Shellcodes und führt diesen dann aus. [68]

Nachdem alle Shellcode Stages übertragen wurden, muss die Verbindung beendet werden. Dadurch wird aus dem ChannelPointerTable der verbliebene Zeiger auf das, inzwischen gefälschte, Kanalobjekt gelesen. Durch die Manipulation des referenzierten Objektes wird die Route, die zur Ausführung des Zeigers auf den Stage 0 Shellcode führt, erreicht. Der Stage 0 Shellcode führt dann Stage 1 aus, welche Stage 2 lokalisiert und ausführt. Stage 2 ist dann der tatsächliche Payload und baut, in der Implementation des

Wurmes, eine Shell auf, die über ein TCP-Verbindung mit dem angreifenden Computer kommuniziert und Befehle ausführt. [68]

2.2.5 PrintNightmare

Die dritte Remote Code Execution Schwachstelle, deren Exploit im Wurm implementiert wurde, ist die PrintNightmare genannte Schwachstelle mit der CVE Nummer CVE-2021-34527 [51]. Sie ist, je nach Konfiguration, lokal und auf entfernten Computern ausnutzbar. In der Implementation des Wurmes lag der Fokus auf Angriffe gegen entfernte Computer.

Um diese Schwachstelle auszunutzen, wird die Funktion `RpcAddPrinterDriverEx` des Print System Remote Protocols verwendet, welche dafür gedacht ist, für neue Drucker Treiber auf einem Druckserver zu installieren [44, S. 266]. Diese Funktion soll nur von authentifizierten Administratoren aufgerufen werden können, jedoch ist es durch Manipulation der Aufrufparameter möglich, einen präparierten Treiber als normaler authentifizierter Benutzer zu installieren, welcher anschließend im SYSTEM Kontext ausgeführt wird. [69]

Um diese Funktion als normaler Nutzer erfolgreich aufrufen zu können, müssen jedoch einige Anforderungen erfüllt sein. Zuerst wird ein authentifizierter Nutzer innerhalb der Domain benötigt und auf dem Zielcomputer muss der Printspooler, also der Druckservice, aktiviert sein und Verbindungen annehmen. Außerdem wird für den Angriff eines entfernten Computers ein Netzwerklaufwerk benötigt, auf welches der angegriffene Computer Zugriff hat. Hier wird die Treiberdatei, die geladen werden soll, gespeichert. Weiterführend muss von den folgenden Voraussetzungen mindestens eine erfüllt sein, um ein Ausnutzen dieser Schwachstelle zu ermöglichen. Eine dieser Bedingungen ist, sollte man einen Domain Controller angreifen, dass der genutzte Nutzer zur Gruppe »Pre-Windows 2000 Compatible Access« gehört. Das ist meist gegeben, da zu dieser Gruppe standardmäßig »Authenticated Users« gehören, also alle angemeldeten Domain Nutzer. Eine weitere Möglichkeit ist, dass auf dem Zielcomputer die `Point and Print warning on install` und die `Point and Print warning on update` Optionen deaktiviert sind und somit keine Bestätigung durch einen privilegierten Nutzer zum Hinzufügen eines Treibers benötigt wird. Ist auf dem Ziel-Computer UAC (User Account Control) deaktiviert, so muss keine Administratorenbestätigung zur Installation eines Treibers eingeholt werden. Dies ermöglicht ebenfalls die Ausnutzung dieser Schwachstelle. [69]

Sind die Voraussetzungen gegeben, so kann man die `RpcAddPrinterDriverEx` Funktion aufrufen und ihr den Pfad zur Treiber .dll (Dynamic Link Library) Datei übergeben. Dynamic Link Libraries sind Dateien, die zusätzliche Funktionen bereitstellen und modular

eingebunden werden können. In diesem Kontext werden durch die DLL die Druckertreiber-Funktionen installiert. Diese werden dann geladen und im SYSTEM Kontext ausgeführt. Innerhalb des Treibers kann man jede mögliche Funktionalität implementieren. Der Wurm übergibt einen Treiber, der eine über TCP zurück zum angreifenden Computer kommunizierende Konsole erzeugt. Diese empfängt Befehle, die genutzt werden, um den Wurm auf diesem Computer direkt in einem mit hohen Berechtigungen versehenen Kontext auszuführen. Der Pfad zum Treiber kann sowohl auf das Dateisystem des angegriffenen Computers, als auch auf ein Netzwerkordner verweisen.

3 Methodik und Vorgehensweise

3.1 Geplante Vorgehensweise

In diesem Abschnitt wird darauf eingegangen, wie die Entwicklung des Wurmes geplant war und wie die Umsetzung einiger Aspekte geschehen sollte.

Um während der Entwicklung möglichst zielgerichtet arbeiten zu können und dabei nicht den Überblick zu verlieren, wurde im ersten Schritt ein einfacher Programm-Ablauf-Plan erstellt (siehe ▶ Abbildung 3.1), welcher den Ablauf der grundsätzlichen Schritte, die der Wurm machen soll, darstellt. Dies dient als Grundlage für weitere Überlegungen.

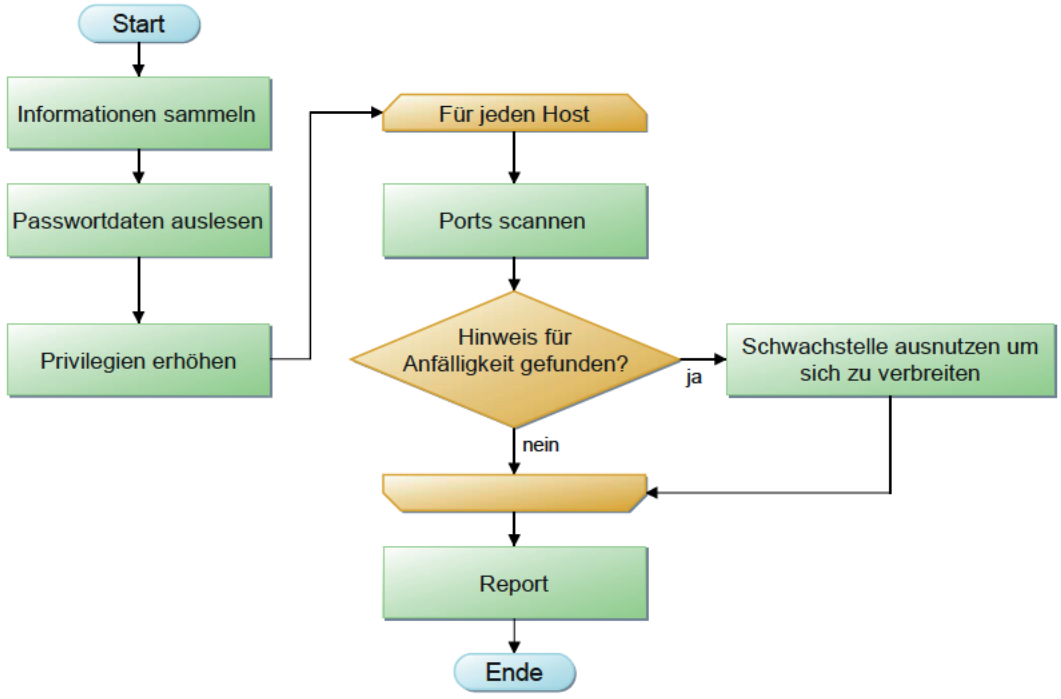

Abb. 3.1: Vereinfachter Programmablauf. Eigene Abbildung.

Als Programmiersprache für den Wurm wurde Python gewählt. Sie ist simpel zu verwenden und bietet dennoch viele Möglichkeiten zur Manipulation von Paketen. Außerdem sind viele zusätzliche Bibliotheken verfügbar und viele Antiviren Programme erkennen Python-Schadsoftware nicht.

Für die Entwicklung wird die Entwicklungsumgebung Visual Studio Code gewählt, da sie Module für Syntax-Highlighting für verschiedene Programmiersprachen, ein integriertes Terminal mit mehreren Reitern, eine einfache Verwaltung mehrere Ordner und eine Integration der Versionsverwaltungssoftware Git anbietet.

Als Vorgehensmodell für die Entwicklung soll das Prototyping angewandt werden, da nicht alle Voraussetzungen, wie z. B. die Formatierung der Argumente, zum erfolgreichen Ausführen der Exploits, bekannt sind. Hierbei sollen die jeweiligen Exploits erst alleinstehend ausgeführt werden, um sie zu testen. Anschließend sollen, vorausgesetzt der Test war erfolgreich, die Exploits in den Wurm integriert werden. Dabei sollen die Exploits zuerst mit festgeschriebenen Werten aus dem manuellen Test innerhalb des Wurmes arbeiten, damit sichergestellt werden kann, dass es innerhalb des Wurmes keine Kreuzwirkung zwischen verschiedenen Funktionen gibt. Abschließend soll der Wurm den Exploits automatisch die benötigten Argumente übergeben.

Zuerst soll eine Recherche zu vorhandenen Exploits für die angestrebten Sicherheitslücken stattfinden und möglichst für jede Schwachstelle ein Exploit gefunden werden. Hier muss besonders auf die Benutzbarkeit im Rahmen eines automatischen Programms geachtet werden, da der Wurm später keine Möglichkeit haben soll, ihm interaktiv neue Befehle oder Informationen zu geben.

Nebenbei sollte das erste Gerüst des Wurmes implementiert werden. Dazu zählen das Sammeln von Informationen über die Domain (z. B. Name(n) und IP-Adresse(n) des oder der Domain Controller), sowie über den Nutzer, in dessen Kontext der Wurm gestartet wird (Ist er Administrator? Wenn ja, lokal oder Domain? etc.). Auch die Netzwerkumgebung soll gescannt werden, um z. B. vorhandene Netzwerklaufwerke zu finden und zu verbinden.

Anschließend soll versucht werden, auf Grundlage der HiveNightmare Schwachstelle im Nutzerkontext die lokale SAM auszulesen und mit den gewonnenen Passwort Hashes sich selbst in den Kontext eines Administrators zu versetzen, also die eigenen Privilegien zu erhöhen.

Wurde dies erreicht, werden die Exploits implementiert, um eine Verbreitung des Wurmes zu realisieren und weiter Nutzerdaten zu erhalten. Da manche Exploits Nutzerdaten, wie den NTLM-Hash eines Domain-Benutzers, benötigen, sollte als erster Exploit der Zerologon Exploit implementiert werden, um anschließend die SAM des Domain Controllers auszulesen und somit alle vorhandenen Nutzer, auch die nicht lokal vorhandenen, mit

ihren Passwort Hashes zu erlangen, die weiterführend genutzt werden können. Sind alle Exploits implementiert, wird als letzter Schritt ein Server erstellt, der als zentrale Sammelstelle für die Resultate des Wurmes auf den verschiedenen Computern des Netzwerkes fungiert. Die entsprechende Meldefunktion wird im Wurm hinzugefügt.

Um den Wurm testen zu können, wird ein kleines Testnetzwerk mit mindestens drei Systemen benötigt (Windows 7, Windows 10 und ein Windows Server 2012 R2). Realisiert wird dieses Netzwerk durch virtuelle Maschinen. Windows 7 und Windows 10 wurden aufgrund der Anforderung und der entsprechenden Anfälligkeiten gewählt. Windows Server 2012 R2 wurde als Server gewählt, da Ziel dieser Arbeit ist, vor allem ältere Systeme zu Testen und Microsoft für Windows Server 2012 R2, im Gegensatz zu Windows Server 2016, eine VHD (Virtual Hard Disk) Datei bereitstellt, die das aufsetzen einer virtuellen Maschine deutlich beschleunigt.

Um den Python Code in eine ausführbare Datei zu packen, die keinen installierten Python Interpreter auf dem Zielcomputer benötigt, wird das Tool Pyinstaller verwendet. Diese wird im Abschnitt Projektdetails genauer erklärt.

3.2 Zusammenfassung der Anforderungen

Das zu entwickelnde Tool soll die Eigenschaften einer Wurm-Malware nachbilden und folgende Anforderungen erfüllen:

Kompatibilität (nicht-funktionale Anforderung)
Das Tool soll ein eigenständiges Programm sein, welches sowohl auf Windows 7 als auch Windows 10 Computern in einer Active Directory Domain funktionsfähig ist und dabei 32 Bit und 64 Bit Architekturen unterstützt. Die Kompatibilität mit 64 Bit und 32 Bit Architekturen ist dabei optional.

Verschleierung der eigenen Funktionen (nicht-funktionale Anforderung)
Das Tool muss seine Funktionen so verschleiern, dass diese nicht vom Windows Defender als potenziell gefährlich markiert und somit die Ausführung des Tools unterbrochen oder verhindert wird.

Auslesen und Kategorisieren von Nutzerdaten sowie Erhöhung der Privilegien (funktionale Anforderungen)
Das Tool soll in der Lage sein Nutzerdaten des lokalen Computers und des Domain Controllers auszulesen und versuchen, sich mithilfe dieser in einen Kontext höherer

Berechtigung zu versetzen. Ziel ist es, in den Kontext eines Domain Administrators versetzt zu werden. Um das zu erreichen, müssen die erhaltenen Nutzerdaten kategorisiert werden. Für das Auslesen von Nutzerdaten sollen die Exploits Zerologon und HiveNightmare implementiert werden.

Scannen der Netzwerkumgebung und Erkennen potenziell angreifbarer Services (funktionale Anforderung)
Das Tool soll seine Netzwerkumgebung, deren Grenzen vor der Ausführung festgelegt werden, scannen und dabei gefundene Hinweise auf potenziell schwachstellenbehaftete Services speichern. Die Erkennung dieser Services soll anhand der geöffneten Ports geschehen. Der Netzwerkscan soll sehr langsam vorgehen, um keine eventuell vorhandenen Abwehrmechanismen wie IDS (Intrusion Detection System) und IPS (Intrusion Prevention System) zu aktivieren.

Ausnutzen von Sicherheitslücken zur Verbreitung und Schaffung von Übertragungswegen für Treiber- und Programmdateien (funktionale Anforderung)
Das Tool soll die Verbreitung realisieren, indem es Hinweisen auf potenziell angreifbare Dienste nachgeht und versucht, die entsprechenden RCE Schwachstellen auszunutzen. Dafür muss es in der Lage sein Netzwerkordner zu finden, lokal zu verbinden sowie deren Zugriffsberechtigungen abzufragen, um anschließend darüber Programm- und Treiberdateien zu übertragen. Die zu benutzenden Exploits für die Verbreitung sind EternalBlue und PrintNightmare.

Melden gesammelter Informationen und Daten (funktionale Anforderung)
Das Tool soll jede gefundene potenzielle Schwachstelle an eine zentrale Stelle melden, ebenso wie den Erfolg oder Misserfolg der Ausnutzung. Die erhaltenen Nutzerdaten und der Fundort sollen ebenfalls weitergegeben werden. Außerdem soll das Tool Informationen speziell zum Domain Controller sammeln, speichern und melden.

Auslesen von LSASS und RDP Sessions (optionale funktionale Anforderung)
Das Tool soll eine Funktionalität zum Auslesen des LSASS sowie ggf. vorhandener RDP Sessions implementieren, um damit Nutzerdaten zu erweitern. Diese Funktion ist optional für ein MVP.

Ausbleiben einer Schadwirkung (optionale funktionale Anforderung)
Das Tool darf keine Schadwirkung entfalten, um es bei einem echten Penetrationstest verwenden zu können. Da im Rahmen dieser Arbeit nur ein Prototyp entwickelt werden soll, ist diese Anforderung optional. Es soll jedoch keine Schadwirkung bewusst implementiert werden.

3.3 Entwicklungstagebuch

In diesem Abschnitt wird chronologisch beschrieben, wie die Programmierung des Wurmes ablief, welche Probleme dabei aufgetreten sind und wie diese gelöst wurden. Außerdem werden Unterschiede im Ablauf der Entwicklung gegenüber dem geplanten Ablauf beschrieben und begründet.

Um den Vorbereitungsaufwand so gering wie möglich zu halten, sollte das Testnetzwerk mit der Mächtigkeit des Wurmes wachsen. Somit wurde initial nur eine Maschine benötigt, auf welcher das Auslesen der Nutzerdaten sowie das Erhöhen der Privilegien implementiert und getestet werden konnte.

Der Ablauf der Entwicklung kann grob in fünf Phasen untergliedert werden (siehe ▶ Abbildung 3.2).

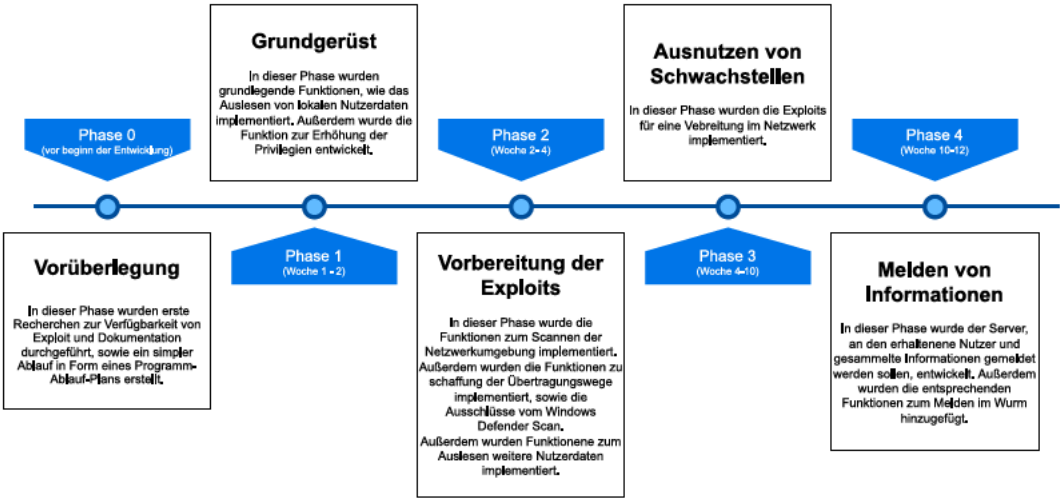

Abb. 3.2: Phasen der Entwicklung. Eigene Abbildung.

3.3.1 Das Grundgerüst

Zu Beginn wurde eine virtuelle Maschine mit Windows 10 aufgesetzt und präpariert, um für HiveNightmare anfällig zu sein. Damit die Anfälligkeit überprüft werden konnte, wurde als erstes ein HiveNightmare Exploit gesucht. Auf GitHub konnten zwei Varianten gefunden werden. Der eine Exploit wurde in der Sprache C# programmiert und war als gebaute, ausführbare Datei vorliegend, der andere in Form eines Python Skriptes.

Der als gebaute Version vorliegende Exploit wurde heruntergeladen und, nach Deaktivierung des Windows Defenders, gestartet. Dieser produzierte drei Dateien,

`SAM-<Datum>`, `SECURITY-<Datum>` und `SYSTEM-<Datum>`. Damit war erwiesen, dass die Konfiguration korrekt ist, um für HiveNightmare anfällig zu sein.

War der Windows Defender aktiviert, wurde dieser Exploit von ihm erkannt und blockiert. Somit musste das Python Skript verwendet werden, da dieses über die Konsole problemlos aufgerufen werden konnte. Nachdem die Funktion des Skriptes in den Code des Wurmes übertragen wurde, erkannte der Windows Defender diesen als potenziell gefährlich und blockierte seine Ausführung. Da die ›gefährliche‹ Aktion nur ein Kopiervorgang ist und das Skript ohne festgeschriebene Werte nicht erkannt wurde, musste der Windows Defender die Logik anhand des Pfades, welcher als String in der Wurm-Datei enthalten war, erkannt haben. Um dieses Problem zu lösen, wurde dieser String verschleiert (siehe ► Unterabschnitt 3.4.3) und der Wurm neu gepackt. Das führte dazu, dass der Windows Defender zwar die ausführbare Datei nicht mehr als gefährlich erkannte, aber den Start des Kopiervorgangs als verdächtige Aktion detektierte und die Ausführung stoppte. Zum einfacheren Testen der weiteren Verschleierung wurde das Exploit Skript zu einem eigenständigen Programm gepackt. Nachdem dieses einzelne Skript das erste Mal gepackt wurde, ohne dass eine Änderung am Code oder der Verschleierung vorgenommen wurde, erkannte der Defender weder die Programmdatei noch die Ausführung. Da kein Weg gefunden werden konnte den Code im Wurm direkt zu implementieren, ohne dass seine Ausführung dadurch beeinträchtigt wird, wurde der gepackte Python Exploit mit in den Wurm eingebettet und während der Ausführung des Wurmes gestartet. Um die produzierten Dateien einfacher finden zu können, wird dem Exploit im Startaufruf ein Ordner übergeben, in den die entsprechenden Dateien (SAM, SYSTEM, SECURITY) kopiert werden sollen. Diese Dateien enthalten die angestrebten Daten, welche jedoch noch extrahiert werden müssen. Zur Extraktion dieser Daten wird das `secretsdump.py` Skript der Impacket Bibliothek verwendet.

In der Funktion zum Auslesen der Nutzerdaten offenbarte sich ein Problem, welches im Laufe der Entwicklung noch öfter auftreten sollte. Das Skript zum Auslesen dieser Daten ist darauf ausgelegt in der Kommandozeile mit Argumenten aufgerufen zu werden. Somit musste das Skript angepasst werden, um es für eine automatische Nutzung zu präparieren. Im Fall des `secretsdump.py` Skriptes war es einfach zu realisieren, da hier die relevanten Funktionen in einer Klasse `DumpSecrets` implementiert sind, welches aus diesem Skript importiert werden kann. Hier offenbarte sich ein weiteres Problem mit CLI Skripten in einem automatisierten Umfeld.

Das `secretsdump.py` Skript erhält die benötigten Informationen wie IP-Adresse, NetBios Computername oder, in diesem Fall, die Pfade zu den entsprechenden Hive Dateien über

Kommandozeilen-Argumente, was in der Implementierung des Wurmes nicht möglich ist.

Das Übernehmen der Kommandozeilen-Argumente wird in diesem Skript mit der Python Bibliothek argparse implementiert, welche mittels eines `ArgumentParser` Objekts ein `options` Objekt erstellt, in dem alle Argumente gespeichert werden. Die möglichen Argumente werden dem `ArgumentParser` Objekt über die Funktion `add_argument` mitgeteilt und das `options` Objekt durch den Aufruf der `parse_args` Funktion erhalten. Dieses `options` Objekt wird dem `DumpSecrets` Objekt, welches die Funktionen zum Auslesen der Daten enthält, in der Initialisierung übergeben. Somit muss für jeden unterschiedlichen Aufruf (z. B. LOCAL SAM Dump und Domain Controller SAM Dump), dass `options` Objekt anders initialisiert sein. Die vorhandenen Felder bleiben gleich. Folglich musste ein Skript angelegt werden, welches die unterschiedlichen Initialisierungen des `DumpSecrets` Objekts mit den entsprechenden Argumenten realisiert. Dieses Skript wurde `credential_dumps.py` genannt und beinhaltet alle Funktionen, die zum Auslesen von Nutzerdaten verwendet werden.

Im `credential_dumps.py` Skript wird global der `ArgumentParser` erstellt und bekommt die möglichen Argumente mitgeteilt. In den einzelnen Funktionen des Skriptes, die die verschiedenen Anwendungen widerspiegeln (`dump_dc` für den Domain Controller, `get_hive_dump` für die lokalen Hives), wird jeweils ein `options` Objekt, durch den `parse_args` Aufruf, initialisiert. Nachdem diese das Objekt initialisiert haben, sind die meisten Optionen (bis auf einige Standard-Werte) leer, da es keine Kommandozeilen-Argumente gibt. Die Felder des `options` Objektes kann man im Nachhinein manipulieren und so die benötigten Daten in das Objekt schreiben. Anschließend wird das `DumpSecrets` Objekt mit diesen `options` initialisieren und die Funktion zum Auslesen gestartet. Nachdem das umgesetzt wurde, war der Eintrittspunkt programmiert und der Wurm hatte seine ersten Nutzerdaten erhalten, mit denen weitergearbeitet werden konnte.

Um die Entwicklung und das Debugging zu vereinfachen, werden die Exploits vorerst vor ihre Nutzung von einem lokalen HTTP-Server heruntergeladen. Perspektivisch sollen diese aber im Wurm eingebettet und bei Bedarf entpackt werden.

3.3.2 Erhöhung der Privilegien

Als nächster Schritt müssen die Privilegien erweitert werden, um mehr Berechtigungen auf dem infizierten Computer zu haben. Unter anderem um folgend das Tool Mimikatz zu laden und auszuführen, welches ermöglicht ggf. vorhandene Remote Desktop / Terminal Service Anmeldedaten aus dem laufenden Local Security Authority Subsystem Service

(LSASS) und dem RDP Service zu extrahieren. Der Unterschied hierbei zur SAM ist, dass auf älteren Systemen evtl. Klartextpasswörter in diesen Prozessen gespeichert sind, welche ausgelesen werden können. Wenn das Auslesen von Klartextpasswörtern nicht möglich ist, können die gesammelten Nutzerdaten mit Mimikatz um z. B. Kerberos Passwörter erweitert werden. Außerdem kann, unter der Voraussetzung, dass ein Administrator eine RDP Session auf diesem Computer hat, der Administrator-Passwort-Hash oder sogar das Klartext Passwort ausgelesen werden. Das Problem hierbei ist, dass Mimikatz ein bekanntes Tool ist, welches vom Windows Defender sofort erkannt und blockiert wird. Zur Umgehung dieses Problems werden Administrator Privilegien benötigt, damit man einen Scan Ausschluss für bestimmte Dateien, Ordner und Prozesse im Windows Defender hinzufügen kann, um anschließend Mimikatz ungestört nutzen zu können.

Um sich selbst in den Administrator Kontext zu versetzen, wird ein Python-Nachbau des, in der Windows Sys-Internal Suite [60] enthaltenen, Tools PsExec verwendet. Dieses Tool ermöglicht es, sofern die Zugriffskontrolle so konfiguriert ist, sich mit dem NTLM Hash eines Administrators zu authentifizieren und eine Konsole mit SYSTEM Berechtigung zu erhalten bzw. einen Befehl in diesem Kontext auszuführen.

Auch das `psexec.py` Skript ist aus der Impacket Bibliothek und eigentlich auf eine CLI Anwendung ausgelegt. Es beinhaltet jedoch ebenfalls alle wichtigen Funktionen in einer Klasse `PSEXEC`, sodass diese einfach aus dem Skript importiert, mit den relevanten Daten initialisiert (Zielsystem, Nutzername, Passwort-Hash, Befehl) und anschließend mit einer klasseneigenen `run` Funktion gestartet werden kann. So wird über PsExec, mit dem übergebenen Befehl, der Wurm erneut aufgerufen. Dies Mal wird dieser im SYSTEM Kontext gestartet. Im Aufruf des Wurmes im SYSTEM Kontext bekommt der neu aufgerufene Wurm die Prozess ID der Nutzer-Kontext Instanz als Argument übergeben und beendet diesen Prozess. Dadurch wird sichergestellt, dass nur eine Instanz des Wurmes auf dem Computer ausgeführt wird.

Während der Entwicklung der Erhöhung von Privilegien, wurde ein Umstand, bezüglich des Übergebens von Argument, nicht berücksichtigt, der zu einem Fehler führte. Dies äußerte sich darin, dass nach der Fertigstellung der Privilegien Erhöhung der Wurm in der Ausführung mit dem Fehler »Invalid Argument«, ohne genauere Beschreibung oder Informationen zur Herkunft des Fehlers, abbrach. Nachdem alle Funktionsaufrufe ohne Erfolg überprüft und einzeln getestet wurden, konnte der Fehler auf das `credential_dumps.py` Skript zurückgeführt werden. Da hier alle Funktionen einzeln funktionierten, musste der Fehler in diesem Skript liegen. Aufgrund dessen, dass außerhalb der Funktionen nur das `ArgumentParser` Objekt initialisiert wurde, war die Fehlerursache klar. Durch das

Hinzufügen des `--pid` Parameters zur Terminierung der Nutzer-Kontext Instanz des Wurmes im Hauptskript des Wurmes war global ein Argument hinzugefügt wurden, der in der `ArgumentParser` Initialisierung im `credential_dumps.py` Skript nicht existierte. Somit wurde bei Aufruf einer Funktion aus diesem Skript (wodurch das Initialisieren des `options` Objekts ausgelöst wird) bemerkt, dass ein nicht definiertes Argument übergeben wurde, was zum »Invalid Argument« Fehler führte. Nachdem der Fehler lokalisiert wurde, war eine Lösung leicht umzusetzen. Zu der `ArgumentParser` Definition im `credential_dumps.py` Skript wurde ein Parameter `--pid` hinzugefügt. Dadurch war dem `ArgumentParser` dieser Parameter bekannt und es wurde kein Fehler ausgelöst. Da das `secretsdump.py` Skript diese Option vorher nicht kannte und daher nicht darauf zugreift, blieb die Funktionalität davon unberührt. Nachdem dieses Problem gelöst wurde, konnte die Erhöhung der Privilegien erfolgreich implementiert werden.

Da der Wurm keine Persistenz implementiert oder eine Ablage für seine Daten hat, muss die System-Kontext Instanz des Wurmes die Nutzerdaten erneut aus der SAM auslesen. Dies wird auch über den HiveNightmare Exploit realisiert, obwohl theoretisch die Berechtigungen zum normalen Auslesen vorhanden sind, da hierfür schon eine Funktion existiert, die wiederverwendet werden kann.

Um die Privilegien Erhöhung mit einer Test-Maschine implementieren zu können, wurde die Option `LocalAccountTokenFilterPolicy` im Windows Registry Pfad `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System` auf den Wert eins gesetzt, um so mit einem lokalen Administratoraccount eine Verbindung zu sich selbst herstellen zu können. Dies wäre ohne diese Option nicht möglich, da Windows lokalen Administratoren in einer Domain nicht die benötigten SYSTEM Berechtigungen auf Shares gewährt. [34] Auf die Folgen dieses Umstandes wird zu einem späteren Zeitpunkt erneut eingegangen.

3.3.3 Mimikatz und Windows Defender

Nachdem der Wurm im System Kontext gestartet werden konnte, musste als nächstes implementiert werden, dass er ein Umfeld schafft, in dem er Mimikatz laden und ausführen kann. Dafür muss, wie bereits erwähnt, zuerst der Windows Defender so eingestellt werden, dass er die ausführbare Datei nicht an ihrer Signatur erkennt. Um das zu erreichen, wird zuerst ein Ordner erstellt und anschließend werden über Powershell Kommandos (siehe Kommandos unten) sowohl der Ordnername als auch der (geänderte) Dateiname von Mimikatz vom Echtzeit-Scan ausgeschlossen. Danach kann Mimikatz in diesen Ordner geladen (bzw. entpackt) werden.

```
subprocess.run(["powershell","-Command",'Add-MpPreference
-ExclusionPath "C:\\windows\\system32\\mytempdata"'])

subprocess.run(["powershell","-Command",'Add-MpPreference
-ExclusionPath "C:\\windows\\system32\\mytempdata\\
<Mimikatz Executable Name>"'])

subprocess.run(["powershell","-Command",'Add-MpPreference
-ExclusionExtension "exe"'])
```

Nachdem die Exklusionen erstellt und die Funktionalität zum Laden und Starten von Mimikatz implementiert wurden, offenbarte sich ein weiteres Problem. Mimikatz bietet diverse Module mit entsprechenden Kommandos in der Form `<Modulname>::<Kommando>` an. Diese müssen entweder (im interaktiven Modus) nacheinander in der Mimikatz Konsole ausgeführt oder (in direkter Ausführung) Mimikatz beim Start als Argumente übergeben werden. Windows Defender erkennt hierbei den Befehl

```
C:\windows\system32\cmd.exe <Mimikatz Executable Name>
privilege::debug <weiter Befehle>
```

und blockiert die weitere Ausführung. Auch das Hinzufügen einer Exklusion für den Prozessname `cmd.exe` und dem Pfad `C:\Windows\System32\` führte nicht zum Erfolg.

Durch Testen verschiedener Befehlskombinationen kristallisierte sich heraus, dass Windows Defender nur den `privilege::debug` Befehl erkennt und blockiert, welcher aber benötigt wird um Mimikatz Debug Privilegien zu gewähren, damit das Tool anschließend in laufende Prozesse einhaken und z. B. die LSA auslesen kann. Da anscheinend die Analyse nur diesen Befehl erkennt, ist die Lösung des Problems, den Kommandonamen zu ändern. Mimikatz ist ein quelloffenes Tool und somit kann jeder sich den Quellcode herunterladen, anpassen und selbst neu kompilieren.

Die Änderung des Kommando- und Modulnamens erfolgte auf folgende Weise.

Zunächst wurde das Git-Repository geklont und der C-Quellcode bezüglich der Modulnamen analysiert. Um ein Modul und die zugehörige aufgerufene Funktion zu finden, iteriert Mimikatz über alle ihm bekannten Module bis er das gesuchte Modul gefunden hat und iteriert dort über die vorhandenen Funktionen. Jedes Modul ist in zwei Strukturen

gespeichert. Einmal die KUHL_M Struktur, welche den Modulnamen, eine kurze Beschreibung und eine Liste von KUHL_M_C Strukturen beinhaltet. Die KUHL_M_C Struktur wird zum Speichern der Modulkommandos genutzt. Diese speichert den Kommandonamen, eine Beschreibung dieses Kommandos sowie einen Zeiger auf die interne Funktion, die bei diesem Kommando aufgerufen werden soll.

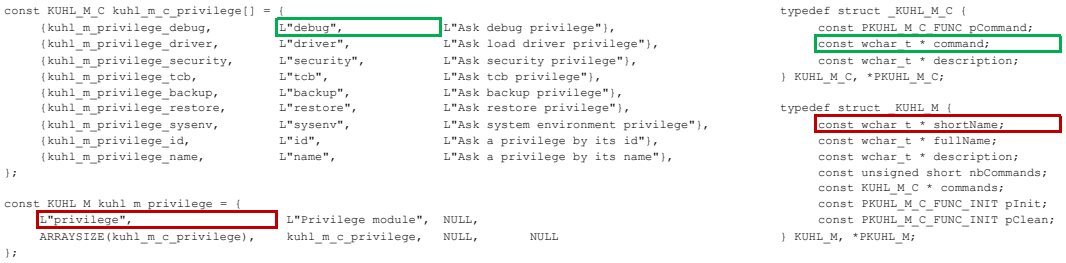

Abb. 3.3: Mimikatz Kommando- und Modulstruktur. Eigene Abbildung.

Somit musste, um den Befehl `Privilege::debug` zu ändern, der Modulname (`wchar_t * shortName`) in der KUHL_M Struktur geändert werden. Außerdem muss der Kommandoname im `wchar_t * command` Feld der KUHL_M_C Struktur (siehe ▶ Abbildung 3.3) angepasst werden. Danach musste der Code neu kompiliert werden, um die Änderungen zu übernehmen. Nachdem Mimikatz neu kompiliert wurde und mit einem anderen Befehl Namen für `privilege::debug` aufgerufen werden konnte, erkannte Windows Defender diesen Aufruf nicht mehr und blockierte die Ausführung nicht. Somit war Mimikatz einsatzbereit. Als nächster Entwicklungsschritt wurde nun das automatische Ausführen von Mimikatz implementiert und die Ausgabe mittels einer Methode `update_user_or_create_new` verwertet. Diese, wie der Name vermuten lässt, überprüft für jeden erhaltenen Nutzer, ob zu dem erhaltenen Nutzername bereits ein `User` Objekt vorhanden ist und aktualisiert dieses ggf. oder erzeugt ein neues `User` Objekt, falls ein Nutzer noch nicht in der Liste der verfügbaren Nutzer enthalten ist.

Nachdem dies implementiert wurde, konnte der Wurm alle auf dem Computer vorhandenen Nutzer, mindestens mit ihrem Passwort-NTLM-Hash, auslesen. Da aber nicht zwangsläufig alle Domainnutzer lokal vorhanden sind, sollte als nächstes versucht werden, die SAM des Domain Controllers mittels einer Zerologon Attacke auszulesen.

3.3.4 Domain Controller Nutzerdaten auslesen

An dieser Stelle wurde der Punkt erreicht, an dem das Testnetzwerk um einen Domain Controller wachsen musste. Hierfür wurde eine virtuelle Maschine mit dem Windows Server 2012 R2 aufgesetzt, auf welchem anschließend der Active Directory Domain Service

(AD DS) gestartet wurde.

Danach mussten die Funktionen zum Sammeln der Informationen implementiert werden. Diese waren ursprünglich als Grundgerüst geplant, wurde jedoch aufgeschoben, da sie in der Anfangsphase der Entwicklung noch nicht benötigt wurden. Um auch hier die Informationen gut speichern und weitergeben zu können, wurde eine Domain Controller Klasse `DomainController` für den Wurm erstellt, die direkt bei Start des Wurmes instanziiert wird und in ihrer Initialisierungsfunktion diverse Befehle zum Sammeln der domainspezifischen Informationen ausführt. Hier wird unter anderem der NetBios Name des Domain Controllers erfragt, welcher wiederum in die IP-Adressen (IPv4 und IPv6) aufgelöst werden kann. Außerdem wird der Domain Name erfragt und gespeichert. Diese Informationen wurden durch das Ausführen von Windows CLI Kommandos (z. B. `nltest /dclist:<Domain Name>`, zum Erhalt des/der Domain Controller NetBios Namen) gesammelt. Die Ausgaben dieser Kommandos stellen deutlich mehr Text und Informationen bereit, als eigentlich benötigt werden. Da die Ausgaben dieser Kommandos immer gleich formatiert sind, lassen sie sich aber z. B. anhand von Leerzeichen oder Zeilenumbrüchen gut in Substrings unterteilen, um die relevanten Informationen zu extrahieren. Zur Erleichterung des Auslesens der IP-Adressen des Domain Controllers, wurden zusätzlich auch hier wieder simple reguläre Ausdrücke verwendet (siehe ▶ Abbildung 3.4).

```
def __init__(self):
  ip_4_regex = r'.*\..*\.*\.*\.*\.*'
  ip_6_regex = r'.*\:.*\:.*\:.*\:.*\:.*\:.*\:.*'
  try:
     self.dc_name = str(subprocess.Popen('nltest /dclist:'+self.domain_name, stdout=subprocess.PIPE).stdout.read()\
        .splitlines()[1]).split(".")[0].replace(" ","")[2:]
     lookupresult = subprocess.getoutput("nslookup "+ self.dc_name).splitlines()
     for i in range(len(lookupresult)):
        if self.dc_name[2:] in lookupresult[i] and "Address" in lookupresult[i+1]:
           if re.match(ip_4_regex,lookupresult[i+1]):
              self.dc_ip_4 = lookupresult[i+1].replace("Address:  ","").replace("Addresses: ","")
           if re.match(ip_6_regex,lookupresult[i+1]):
              self.dc_ip_6 = lookupresult[i+1].replace("Address:  ","").replace("Addresses: ","")
           try:
              ip = lookupresult[i+2].replace("\t","").replace(" ","")
              if re.match(ip_4_regex,ip):
                 self.dc_ip_4 = ip
              if re.match(ip_6_regex,ip):
                 self.dc_ip_6 = ip
           except:
              continue
  except:
     pass
```

Abb. 3.4: Initialisierungsfunktion der `DomainController` Klasse. Eigene Abbildung.

Nachdem das Sammeln der Informationen über den Domain Controller implementiert wurde, musste ein Exploit für die Zerologon Schwachstelle gefunden werden. Hierfür gibt es eine Python Implementation, die die Impacket Bibliothek nutzt [52]. Der Exploit Code ist hier in einer Funktion `perform_attack` enthalten, die aus dem Skript importiert werden konnte. Diese sendet, wie in Abschnitt (siehe ▶ Unterabschnitt 2.2.2) beschrieben,

Pakete, deren Inhalt nur aus Null-Werten besteht. Dieser Vorgang wird bis zu 2000 mal oder bis die Authentifizierung erfolgreich war (durchschnittlich 256 mal) wiederholt und löscht bei Erfolg das Passwort des Computer Accounts des Domain Controllers. Anschließend kann mit dem `secretsdump.py` Skript auch die Domain Controller SAM ausgelesen werden, da sich das Skript via SMB und ohne Passwort als der Maschinenaccount anmelden kann und somit Zugriff auf alle lokalen Ressourcen hat [59].

Nachdem diese Funktion entwickelt wurde, waren alle Exploits zum Auslesen von Nutzer- und Anmeldedaten implementiert. Das Installieren des ursprünglichen Passworts nach dem Auslesen der Nutzerdaten wurde vernachlässigt, um zuerst alle für den MVP geforderten Funktionen zu implementieren.

3.3.5 Die Verbreitung

Nachdem das Auslesen der Nutzerdaten implementiert war, musste nun der letzte und wichtigste Schritt des Wurmes entwickelt werden, die Verbreitung. Hierfür war in der Vorüberlegung geplant, die Schwachstellen EternalBlue und PrintNightmare auszunutzen.

Angefangen wurde mit der Suche nach einem EternalBlue Python Exploit, wozu auch mehrere GitHub Repositorien existieren [65]. Um diese testen zu können, musste das Testnetzwerk um eine Maschine erweitert werden, die für einen EternalBlue Exploit anfällig ist. Diese Anforderung stellte ein größeres Hindernis dar, als erwartet, da die meisten online verfügbaren Systemabbilder und VirtualBox Dateien schon die entsprechenden Updates, um diese Lücke zu schließen, installiert hatten. Über das Internetarchive konnte ein Evaluationsabbild gefunden werden, bei dem das neuste Update auf das Jahr 2012 datiert war und diese Maschine somit die Schwachstelle beinhaltete [39]. Nachdem diese Maschine aufgesetzt und gestartet wurde, konnte sie zur Domain hinzugefügt und SMBv1 aktiviert werden.

Um zu testen, ob diese Maschine tatsächlich über die EternalBlue Schwachstelle angreifbar ist, wurde zuerst mit dem Metasploit Framework ein Angriff gestartet. Dieser war erfolgreich und es wurde eine Meterpreter Session auf der Maschine erzeugt. Somit konnte diese Schwachstelle ausgenutzt werden.

Anschließen wurden die verschiedenen Exploits getestet. Schon beim Testen der Python Exploits zeigte sich eine große Schwierigkeit dieser Implementationen. Die Skripts verursachten fast jedes Mal einen Crash der virtuellen Maschine. Getestet wurde mit diversen

Shellcodevarianten, die alle einen rückkommunizierenden Prozess mit einer cmd.exe Instanz erzeugen sollten. Auf dem angreifenden Computer wurde jeweils ein Port geöffnet und überwacht, zu dem sich dieser Prozess verbinden sollte. In manchen Fällen wurde für einen kurzen Moment eine Verbindung aufgebaut. Diese brach aber sofort wieder ab, da die virtuelle Maschine abstürzte. Somit konnten keine Befehle übertragen werden und damit auch der Wurm nicht kopiert werden.

Da anscheinend der Shellcode ausgeführt wurde, dieser nur nicht in der Lage war, vor dem Absturz eine vollständige Verbindung aufzubauen, war ein Ansatz für die Lösung des Problems das Erstellen eines Autostart-Eintrages statt eines rückkommunizierenden Prozesses. Dieser sollte bewirken, dass nach dem Neustart durch den Absturz, der Wurm von einem Netzwerklaufwerk gestartet wird.

Für den Autostart-Eintrag wurde mit der Programmiersprache C++ ein kleines Programm geschrieben, welches vorerst in der Wurzel des Dateisystem (`C:\`) eine Textdatei erstellen sollte. Nachdem dieses Programm entwickelt wurde, sollte dessen Shellcode über den EternalBlue Exploit übertragen werden. Leider musste hier festgestellt werden, dass die Zeit vor dem Absturz nicht ausreicht, um einen Dateisystemvorgang durchzuführen, was sich darin äußerte, dass keine Datei erstellt wurde. Da es an dieser Stelle keine weiteren Optionen gab, diesen Exploit zu verwenden, wurde dessen Implementierung abgebrochen und lediglich der Scan nach potenziell angreifbaren Zielen implementiert. Um das Projekt übersichtlich zu halten, wurde diese Logik in ein separates Skript `exploit_manager.py` ausgelagert.

Der zweite geplante Exploit, der zur Verbreitung genutzt werden sollte, war der PrintNightmare Exploit. Auch hierfür waren auf GitHub verschiedene Exploits zu finden, die jedoch alle auf den von cube0x0 [17] erstellten Exploit verwiesen. Hierfür musste ebenfalls eine Maschine erstellt werden, die für diese Schwachstelle anfällig ist. Da hier, wie im Abschnitt »PrintNighmare« (siehe ► Unterabschnitt 2.2.5) beschrieben, einige Voraussetzungen erfüllt sein müssen und dieser Exploit auch für neuere Versionen des Windows Betriebssystems funktionieren sollte, wurde hierfür eine virtuelle Maschine mit Windows 10 aufgesetzt, die ihr letztes Update im zweiten Halbjahr 2020 erhalten hatte und die PrintNightmare Schwachstelle in sich trägt. Nachdem die erforderlichen Registry Einträge gesetzt wurden, die das Ausnutzen der Schwachstelle ermöglichen, konnten die Exploits getestet werden. Hierfür wurde ein präparierter Treiber verwendet, der einen rückkommunizierenden Prozess mit einer cmd.exe Instanz erzeugt. Da alle gefundenen Exploits manuell erfolgreich ausgeführt werden konnten, fiel die Wahl auf den ›originalen‹ Exploit von cube0x0. In dieser Implementation wurde der Angriff in der `Main` Methode

gestartet, welche ein DCERPC (Distributed Computing Environment / Remote Procedure Call) Verbindungsobjekt, den Pfad des aktuell installierten Treibers und den (Netzwerk-) Pfad des zu installierenden Treibers übergeben bekam. Die DCERPC Verbindung wurde mit der Funktion connect aufgebaut, die eine Authentifizierung über NTLM-Hashes ermöglichte. Der aktuell installierte Treiber wurde über die Funktion `getDriver` erfragt, die ebenfalls eine DCERPC Verbindung übergeben bekam.

Diese drei Funktionen wurden im `exploit_manager.py` Skript importiert und genutzt. Zuerst wird versucht mit allen bekannten Nutzern, zu denen auch ein NTLM Hash bekannt ist, eine DCERPC Verbindung zum Opfersystem aufzubauen. Wenn dies erfolgreich ist, wird das Verbindungsobjekt der `getDriver` Funktion übergeben, um den momentan installierten Treiber abzufragen.

Um den Wurm auf dem entfernten Rechner starten zu können, muss das entfernte System auf die Programmdatei des Wurmes zugreifen können. Zur Realisierung dieser Aufgabe, wurden zwei Klassen erstellt. Die `NetworkManagement` Klasse, die in der Netzwerkumgebung nach SMB-Shares (im Netzwerk freigegebenen Ordnern) suchen und diese speichern sollte. Sowie die Klasse `SmbShare`, welche die gefundenen SMB-Shares, darstellen und deren Name, Host und einige weiter Informationen über den Netzwerkordner speichern sollte. Die `SmbShare` Objekte der gefundenen SMB-Shares wurden im `NetworkManagement` Objekt in einer Liste gespeichert.

In der Initialisierung des Networkmanagers wird automatisch jeder eingetragene Zielcomputer nach Netzwerkshares angefragt, um so die verfügbaren Shares zu finden. Bevor mit der Ausführung der PrintNightmare Funktionen begonnen wird, werden zuerst alle verfügbaren Shares auf dem Computer eingehängt und die Zugriffsrechte erfragt. Anschließend werden die beiden oben beschriebenen Funktionen, `connect` und `getDriver`, ausgeführt. Nachdem der Treiberpfad vom entfernten Computer erhalten wurde, wird ein Netzwerkordner ausgewählt, auf welches dann die zu installierende Treiberdatei geschrieben wird. Auch diese Datei wurde vorerst von einem Server heruntergeladen, perspektivisch wird diese aber im Wurm eingebettet.

Da der entfernte Rechner auf diesen Netzwerkordner Zugriff haben muss, wird die Liste der `SmbShare` Objekte nach einem SMB-Share durchsucht, auf den jeder Nutzer Zugriff hat. Anschließend wird die `Main` Funktion des PrintNightmare Exploits gestartet, welche diesen Treiber installieren soll. Nachdem das implementiert und getestet wurde, indem manuell ein Port abgehört wurde, musste der Wurm selbst einen Port abhören und Befehle in diese Verbindung senden. Dies wird `exploit_manager.py` Skript in einer

Funktion `setup_listener` implementiert, welche einen Socket erstellt und auf eingehende Verbindungen wartet. Der Funktion zum Abhören des Ports wird der Netzwerkpfad des Wurmes übergeben. Hier wird, nach dem selben Prinzip wie für den Treiber, ein Share gewählt und die Wurm-Programmdatei dorthin kopiert. Empfängt der Socket eine Verbindung, sendet er zwei verkettete Befehle, mit denen der Wurm kopiert und ausgeführt wird.

```
copy <Programmdatei Netzwerkpfad> C:\ && C:\worm.exe
```

Um einen Port abzuhören und gleichzeitig die Exploit Funktionen ausführen zu können, musste Nebenläufigkeit für diese Funktionen implementiert werden. Um das zu erreichen, wurde die Funktion zum Abhören eines Ports in einen separaten Thread ausgelagert. Dadurch wurde verhindert, dass das Abhören eines Ports die Ausführung der Exploitfunktionen blockiert.

Nachdem beide Teile, Exploit und Abhören, implementiert wurden, mussten diese getestet werden. Dabei war der Exploit immer erfolgreich, jedoch wurde nie eine Verbindung aufgebaut und der Wurm beendete sich. Die Schwierigkeit hier lag darin, dass dieser Exploit erst gestartet wird, nachdem die Privilegien erhöht wurden, somit der User-Kontext Wurm terminiert wurde und daher in der Konsole keine Ausgaben oder Fehlermeldungen zu sehen sind. Über das Schreiben von Dokumenten an gewissen Stellen im Wurm, wurde diese Problematik angegangen. Es wurde festgestellt, dass der Port-überwachende Thread mit einem Time Out Fehler endet. Die Ursache war, das Sockets eine Timeout-Zeit besitzen, das heißt nach einer gewissen Zeit ohne Verbindung schließt sich der Socket mit dem ›Timed out‹-Fehler. Diese Timeout-Zeit kann individuell gesetzt werden und lag in diesem Fall bei 240 Sekunden (4 Minuten). Der Grund dafür ist, dass dieser Thread nicht endlos ausgeführt werden soll und die Verbindung vom angegriffenen Computer in der Regel recht schnell geht. Also wurde die Fehlersuche fortgesetzt und versucht, den Socket mit einem unendlichen Time Out zu erstellen. Auch hier kam keine Verbindung zu Stande. Als Ursache für diesen Fehler konnte schlussendlich die Firewall gefunden werden. Diese blockierte alle Verbindungen, da der Wurm keine Erlaubnis durch die Firewall hat. Somit musste eine Firewall Regel erstellt werden, die dem Wurm erlaubt, eingehende Verbindungen zu akzeptieren. Das entsprechende Powershell Kommando wurde mit an der Stelle implementiert, wo auch die Defender Scan Ausschlüsse erstellt werden. Nach der Erstellung dieser Regel, wurde über den Port eine Verbindung aufgenommen und es konnten Befehle übertragen werden. Das Kopieren und Ausführen des Wurmes funktionierte daraufhin problemlos. Somit war der erste Verbreitungsweg, Ausnutzen der

PrintNightmare Schwachstelle, implementiert. Um aber für eine Verbreitung nicht von allein einer Schwachstelle abhängig zu sein, wurde statt EternalBlue noch ein BlueKeep Exploit implementiert. Gewählt wurde dieser Exploit, da er ein Pre-Authentication Exploit, und somit nicht auf Nutzerdaten angewiesen ist. Auch für diesen Exploit gibt es verschiedene Repositorien. Die Wahl fiel hier auf den am besten dokumentierten und am leichtesten zu Implementierenden Exploit von RICSecLab [30].

Die größte Herausforderung für diesen Exploit war es, die benötigten Bibliotheken zu installieren, um den gefundenen Exploit überhaupt verwenden zu können. Dieser benötigt eine Bibliothek, die nicht über den Paketmanager von Python installiert werden kann, sondern selbst gebaut und installiert werden muss. Hier gab es diverse Probleme. Das Erste war, dass während des Bauens der Bibliothek die »C Runtime Library« nicht gefunden werden konnte. Dies konnte durch das Installieren der »C/C++ Buildtools« gelöst werden. Nachdem diese verfügbar waren, konnte die Bibliothek erfolgreich kompiliert und installiert werden. Der nächste Fehler lag darin, dass die kompilierte Version eine andere als benötigt war und somit manche Funktionsaufrufe Fehler in der installierten Bibliothek verursachten. Da im Repository des BlueKeep Exploits keine Version angegeben, die von dieser Bibliothek benötigt wurde, musste der Code der letzten Updates dieser Bibliothek, in Bezug auf den durch den Exploit verursachten Fehler, analysiert werden. Das Ergebnis war, dass die Version 1.0.0 der pyrdp Bibliothek [32] benötigt wird. Durch die Versionsverwaltung Git war das Zurücksetzen der Bibliothek auf diese Version und eine erneute Installation einfach umzusetzen. Nachdem die richtige Version installiert war, verursachte die Nutzung im Wurm nach wie vor Fehler, die manuelle Ausführung funktionierte jedoch. Der Fehler im Wurm lag in einer nicht vorhandenen Konfigurationsdatei, bei der nicht einsehbar war, woher diese Konfigurationsdatei bei Ausführung in der Konsole gelesen wird. Dieses Problem wurde gelöst, indem in der Konsole die Konfiguration gespeichert wurde, danach ausgelesen und eine Funktion zum Bibliothekscode hinzugefügt wurde. Diese Funktion gibt bei Aufruf die Default-Werte zurück, sollte keine Konfiguration zu finden sein. Nachdem nun die Bibliothek zum dritten Mal gebaut und installiert wurde, konnte der Exploit erfolgreich ausgeführt werden.

Die Übertragung des Exploit Codes in den Wurm konnte leicht umgesetzt werden, da auch hier die relevanten Funktionen alle in einer Klasse `Exploit` enthalten sind, die im `exploit_manager.py` Skript importiert werden konnte. Ein Problem gab es jedoch mit dieser Implementation, welche zwar die Funktionalität des Exploits nicht beeinträchtigte, aber auch nicht vollständig gelöst werden konnte. In der genutzten Implementation wird ein `reactor` aus der twisted Python Bibliothek verwendet, der asynchron die Verbindung aufnimmt und die entsprechenden Pakete sendet. Nur beendet sich dieser nicht, nach-

dem alle Pakete gesendet wurden, sondern muss über einen `stop` Aufruf angehalten werden. Das Problem dabei war, das keine Möglichkeit gefunden wurde, festzustellen, wann alle Pakete gesendet wurden. Um zumindest die weitere Ausführung des Wurmes nicht zu blockieren, wurde der `start` Aufruf des Exploits in einen weiteren Thread ausgelagert. Nach dem Start des Exploit- und des abhörenden Threads, wurde nur auf das zurückkehren des Abhörenden Threads gewartet, bevor die Ausführung der Wurm-Funktionen fortgesetzt wurde. Bei Beendigung des Wurmes wird der laufende Exploit Thread automatisch terminiert.

Damit waren alle Exploits implementiert und die Test-Funktionen komplett. Als letzten zu implementierenden Teil blieb noch das Melden der erhaltenen Nutzerdaten, der infizierten Computer und der dafür genutzten Schwachstellen. Da während der Entwicklung für Debugging Zwecke immer wieder Daten, wie die durch Zerologon erhaltene Nutzerdaten, in Dateien auf den Desktop geschrieben wurden, mussten diese Schreibvorgänge nun in Anfragen zum Report Server umgewandelt werden.

3.3.6 Report Funktion und Report Server

Für den Report Server wurde die Python Bibliothek Flask verwendet. Da nur Informationen vom Wurm zum Server gesendet werden müssen, bietet dieser nur POST Endpunkte an. In den POST Anfragen werden die Informationen im JSON Format übertragen.

Folgende Endpunkte wurden implementiert:

- /init – Benachrichtigung über Start des Wurmes und Übergabe der IP-Adressen, die getestet werden
- /portscan – Ergebnisse der Portscans
- /domaincontroller – Informationen über den Domain Controller
- /userdata – gesammelte Nutzerdaten
- /exploit – Informationen zu Exploits
- /error – Informationen zu Fehlern im Wurm

Wenn der Wurm startet, wird der /init Endpunkt aufgerufen, um diesem die zu testenden IP-Adressen zu übergeben. Der Report Server speichert diese Adressen in einzelnen Objekten (mehr dazu in den Projektdetails ► Unterunterabschnitt 3.4.1.5), damit folgend Informationen für jeden Host einzeln gespeichert werden können. Den anderen Endpunkten wird jeweils die IP-Adresse übergeben, zu der die gesendeten Daten gehören, damit

diese zugeordnet werden können. Für jeden Host bzw. jede IP-Adresse werden Schwachstellenindikatoren und deren Ausnutzbarkeit separat gespeichert, da die Indikatoren auch falsch-positiv sein können.

Die Funktion zum Melden innerhalb des Wurmes wurde anders implementiert als im Abschnitt Vorgehensweise vorgesehen. Ursprünglich war geplant, dass alle gesammelten Daten am Ende der Ausführung gemeldet werden. Problematisch ist dabei folgendes. Sollte der Wurm in seiner Ausführung zu einem Zeitpunkt unterbrochen werden, an dem noch keine Daten gemeldet wurden, sind die bis dahin gesammelten Daten verloren. Daher wurde die Meldefunktion so implementiert, dass Daten sofort gemeldet werden, nachdem sie erhalten wurden. Dadurch wird auch nachvollziehbar, an welcher Stelle die Ausführung des Wurmes unterbrochen wurde.

Für das Melden von Exploitinformationen wird folgende JSON Struktur genutzt, die im Wurm erstellt und im Körper der POST Anfrage übertragen wird.

```
{"exploit": "<ExploitName, z. B. eternalblue>", "host":"<IP-Adresse>",
"indicator_found": <true/false>, "exploited":<true/false>}
```

Die Schlüssel `indicator_found` und `exploited` müssen nicht beide gesetzt sein, sondern können einzeln übertragen werden. Somit kann zuerst der Hinweis auf eine Schwachstelle gemeldet werden, bevor ihre Ausnutzbarkeit getestet wird. Das Ergebnis wird nach dem Test gemeldet. Dafür wurden zusätzliche Funktionen in der `NetworkManagement` Klasse implementiert.

Um Informationen zu melden, die im Wurm als Objekte gehalten werden, wird die json Bibliothek von Python benutzt. Diese ermöglicht es, Objekte mithilfe einer Encoder Klasse direkt in ein JSON Format zu überführen. Serverseitig kann dieses JSON wieder in dasselbe Objekt übertragen und gespeichert werden. Diese Funktionalität wurde für die `User` Objekte und das `DomainController` Objekt implementiert. Erhält der Server ein `User` Objekt von einem Host, welches bereits vorhanden ist, vergleicht er diese und erweitert ggf. die gespeicherten Informationen. Nutzer werden immer den Hosts zugeordnet, auf denen sie gefunden wurden.

Die Ergebnisse der Portscans werden in folgender Form übertragen:

```
{"host":"<IP-Adresse>", "open_ports":[<Liste der offenen Ports>]}
```

Dabei können für jeden Host mehrfach Informationen gesendet werden. Dies ermöglicht das Melden einzelner offener Ports vor Abschluss des gesamten Scans. Serverseitig wird die Liste der offenen Ports um die neuen erhaltenen Ports erweitert. Dabei wird darauf geachtet keine Einträge doppelt zu speichern.

Die JSON-Struktur zum Melden der Fehler wurde wie folgt gewählt:

```
{"error": "<Python Fehlerbeschreibung Beschreibung>", "location":
"<Ort des Fehlers im Wurm>", "host": "<IP-Adresse des Computers
auf dem der Fehler aufgetreten ist>"}
```

Die Zusammenfassung, die vom Server erstellt wird, wurde vorerst als Ausgabe einer Textdatei implementiert, die nach den einzelnen Hosts sortiert ist und für jeden die entsprechenden Informationen enthält.

Damit war auch die letzte geforderte Funktionalität implementiert und der Wurm wurde in die Lage versetzt, seine gesammelten Informationen einem Server (einer zentralen Stelle) zu melden.

3.4 Projektdetails

In diesem Abschnitt werden einige Details des entwickelten Tools und des Projektes vorgestellt und genauer erklärt, um den Aufbau des Projektes und des Tools näher zu bringen.

3.4.1 Projektstruktur

Wie bereits im Entwicklungstagebuch erwähnt, wurde die Logik des Wurmes auf unterschiedliche Skripte verteilt, um so die Übersichtlichkeit des Projektes zu verbessern. Dabei wurde als Hauptordner ein Ordner `Worm_Files` erstellt, in welchem die Unterordner `WormData` und `WormBinaries`, sowie `support_files` erstellt wurden. Der `support_files` Ordner enthält alle Exploits und Skripte, die nicht selbst geschrieben wurden, sondern in anderen Skripten verwendet werden. Der `WormBinaries` Ordner enthält die Treiber, die im Wurm eingebettet werden und Ordner `WormData` die ausführbaren Dateien wie Mimikatz, funnycat_b64x2 in ► Abbildung 3.5, und den HiveNightmare Exploit, HiveRead.exe in ► Abbildung 3.5. Die selbst programmierten Skripte `worm.py`, `credential_dump.py`, `user_manager.py`, `network_manager.py` und `exploit_manager.py` sind im Hauptordner `Worm_Files` enthalten und nutzen unter anderem die Skripte im `support_files` Ordner

(siehe ► Abbildung 3.5). Auf die selbst programmierten Skripte und deren genaue Funktion wird im Folgenden eingegangen.

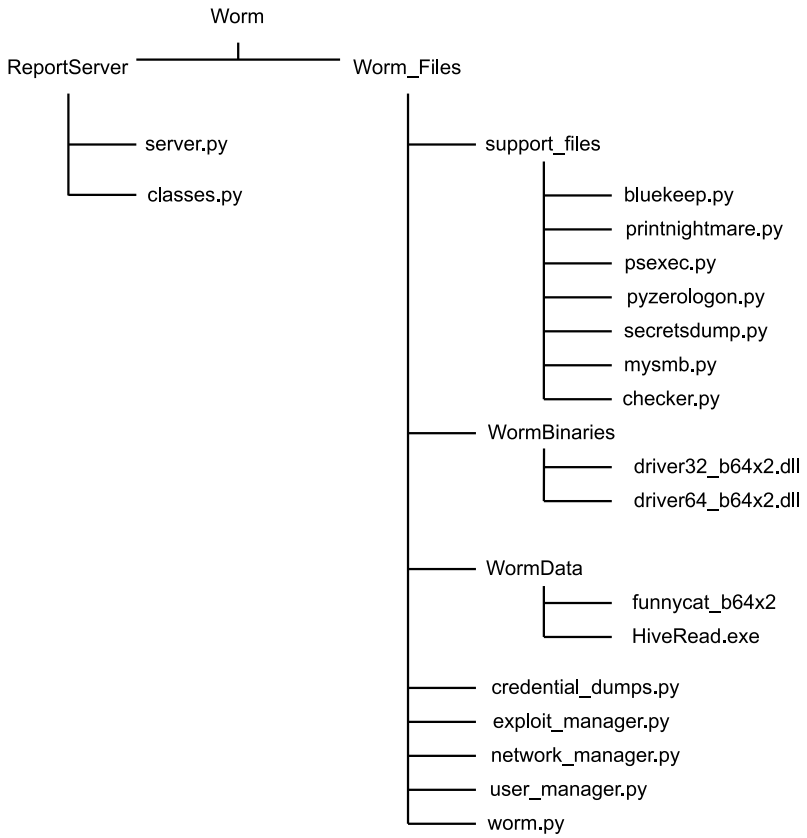

Abb. 3.5: Projektstruktur. Eigene Abbildung.

3.4.1.1 worm.py Skript

Das `worm.py` Skript ist das Hauptskript des Wurmes. Dieses Skript wird dem Pyinstaller zum Analysieren und Packen übergeben (siehe ► Unterabschnitt 3.4.2).

Der Ablauf dieses Skriptes ist, bis auf kleine Änderungen und Erweiterungen, analog zum Programm-Ablauf-Plan aus dem Abschnitt »Geplante Vorgehensweise« (siehe ► Abschnitt 3.1). Ein Unterschied zum Programm-Ablauf-Plan ist, dass das Melden von Informationen nicht erst am Ende der Ausführung geschieht, sondern immer direkt nach dem Erhalt der Information. Außerdem wurde geändert, dass nicht nur nach dem Auslesen der lokalen Nutzerdaten eine Erhöhung der Privilegien versucht wird. Nach einem Zerologon Angriff wird in der vorliegenden Version ebenfalls versucht die eigenen Privilegien zu erhöhen, sollte der Wurm noch nicht in einem SYSTEM Kontext ausgeführt werden.

Zu Anfang werden im `worm.py` Skript die Manager-Klassen, also `UserManager` und `NetworkManger`, sowie die `DomainController` Klasse instanziiert. Die Definition der `DomainController` Klasse ist im `worm.py` Skript enthalten. Diese Klasse wird dafür genutzt, durch das Ausführen von Befehlen, Informationen wie den NetBios Namen, die IP-Adresse und den Domain Namen zu erhalten und zu speichern. Das Sammeln dieser Informationen geschieht, ebenso wie das Melden dieser Informationen, automatisch in der Initialisierungsfunktion der Klasse. Ebenfalls wird in diesem Skript eine Liste definiert, die die zu untersuchenden IP-Adressen enthält und der Initialisierungsfunktion der `NetworkManger` Klasse übergeben wird.

Nach der Instanziierung aller Objekte wird überprüft, ob der Wurm mit dem Parameter `--pid` aufgerufen wurde und ggf. der Nutzer Kontext Prozess über die, als Argument übergebene, Prozess-ID terminiert. Außerdem wird hier auch der Arbeitsordner des Wurmes abgefragt (mehr dazu in Abschnitt Pyinstaller (siehe ► Unterabschnitt 3.4.2). Weiterführend wird überprüft, ob man sich in einem SYSTEM Kontext befindet. Sollte dies nicht der Fall sein, wird versucht mittels des HiveNightmare Exploits Nutzerdaten lokal auszulesen und sich damit in den Kontext höherer Berechtigung zu versetzen.

Danach werden die Hosts auf Hinweise nach Schwachstellen untersucht. Sollte bei einem Host ein Hinweis entdeckt werden, wird dieser über den `NetworkManager` gemeldet und danach versucht, die Schwachstelle auszunutzen. Konnte gegen den Domain Controller eine erfolgreiche Zerologon Attacke ausgeführt werden, wird, wie bereits erwähnt, erneut überprüft, ob sich der Wurm in einem System Kontext befindet. Sollte dies nicht der Fall sein, wird erneut eine Erhöhung der Privilegien versucht.

3.4.1.2 UserManagement (user_manager.py)

Um die erhaltenen Nutzerdaten zu speichern, wurde, wie bereits erwähnt, eine `UserManagement` Klasse erstellt. Die Definition hierfür ist Teil des `user_manager.py` Skriptes, ebenso wie die zugehörige User Klasse und die Funktion zum Überprüfen, ob der Nutzer, in dessen Kontext der Wurm ausgeführt wird, ein Administrator ist. Diese beiden Klassen sind in ► Abbildung 3.6 (siehe nächste Seite) veranschaulicht.

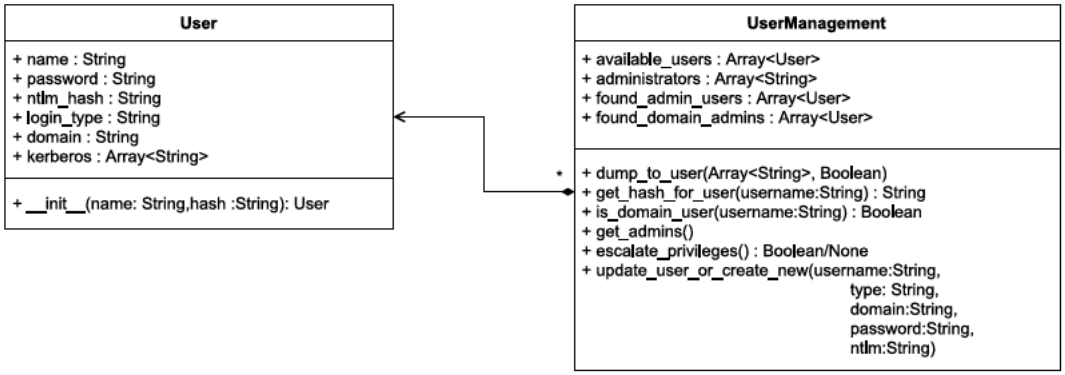

Abb. 3.6: `UserManagement` und `User` Klassendiagramme. Eigene Abbildung.

Die User Klasse beinhaltet Felder für alle relevanten Informationen wie z. B. Nutzername, NTLM Hash, Domainzugehörigkeit, Passwort und weitere, die für eine Weiterentwicklung interessant wären, z. B. der Login Typ oder das Kerberos Passwort (`login_type, kerberos`). Die `UserManagement` Klasse beinhaltet eine Liste für alle erhaltenen Nutzer sowie eine Liste für alle Nutzernamen von Administratoren. Da die Nutzernamen der Administratoren als Ergebnis eines Befehls erhalten werden, sind in der Liste `administrators` nur die Nutzernamen enthalten. Außerdem gibt es zwei weitere Listen, `found_admin_users` für lokale Administratoren und `found_domain_admins` für Domain Administratoren, in denen die kategorisierten `User` Objekte gespeichert werden.

Weiterführend beinhaltet die Klasse einige Funktionen für das Bearbeiten und Abfragen der gespeicherten Nutzer. Die Funktion `dump_to_user` erhält hierbei eine Liste aus Strings, wobei jeder Eintrag, einer Zeile in der Ausgabe von Mimikatz oder des `DumpSecrets` Objekts entspricht. Über diese Liste wird iteriert und die erhaltenen Informationen werden in ein oder mehrere User Objekte übertragen, die dann wiederum der Liste der erhaltenen Nutzer angehangen werden. Weiterführend gibt es eine Funktion `get_hash_for_user`, die einen Nutzernamen übergeben bekommt und über die Liste der vorhandenen Nutzer iteriert bis sie den entsprechenden Nutzer gefunden hat, um dann dessen NTLM Hash zurückzugeben. Diese Funktion wird für die Erhöhung der Privilegien verwendet, da PsExec einen Administrator-Account benötigt, um Befehle im SYSTEM Kontext auszuführen und in der Administratoren Liste, wie bereits erwähnt, nur Nutzernamen enthalten sind. Außerdem beinhaltet die `UserManagement` Klasse eine Funktion `update_user_or_create_new`, die z. B. nach dem Auslesen der Domain Controller SAM verwendet wird, um die Daten für einen Nutzer zu erweitern bzw. einen neuen zu erstellen, sollte es zu den erhaltenen Daten noch kein User Objekt geben. Die letzte Funktion dieser Klasse ist die `escalate_privileges` Funktion, die die PSEXEC Klasse mit einem Administrator-Account aufruft, um den Wurm im SYSTEM Kontext zu starten.

Die Entscheidung diese Funktion im `UserManagement` zu implementieren, beruht darauf, dass diese Funktion nur mit Nutzerdaten umgeht und auch kein direkter Exploit ist, da hier nur die Ergebnisse eines anderen Exploit, HiveNightmare oder Zerologon, verwendet werden.

Die `UserManagement` Klasse implementiert ebenfalls die Reporting Funktion für die Nutzerdaten. Jedes Mal, wenn die Funktionen `dump_to_user` oder `update_user_or_create_new` aufgerufen werden, erfolgt automatisch eine Meldung der dabei entstandenen oder veränderten `User` Objekte.

3.4.1.3 NetworkManagement (network_manager.py)

Das `network_manager.py` Skript beinhaltet ebenfalls zwei Klassendefinitionen, die genutzt werden, um Informationen zu speichern. Analog zur `UserManagement` und User Klasse, werden in diesem Skript eine `NetworkManagement` Klasse und eine `SmbShare` Klasse definiert (siehe ▶ Abbildung 3.7). Wird ein `NetworkManager` Objekt instanziiert, so wird der Initialisierungsfunktion die Liste der zu testenden Hosts übergeben. Diese ruft eine Funktion `collect_network_shares` auf, die jeden der übergebenen Hosts nach ihren freigegebenen Shares abfragt und aus der Antwort für jedes erhaltene Share ein `SmbShare` Objekt mit dem Namen und dem Host des Shares erstellt. Außerdem sind hier drei weitere Funktionen implementiert, die die Arbeit mit den Shares erleichtern.

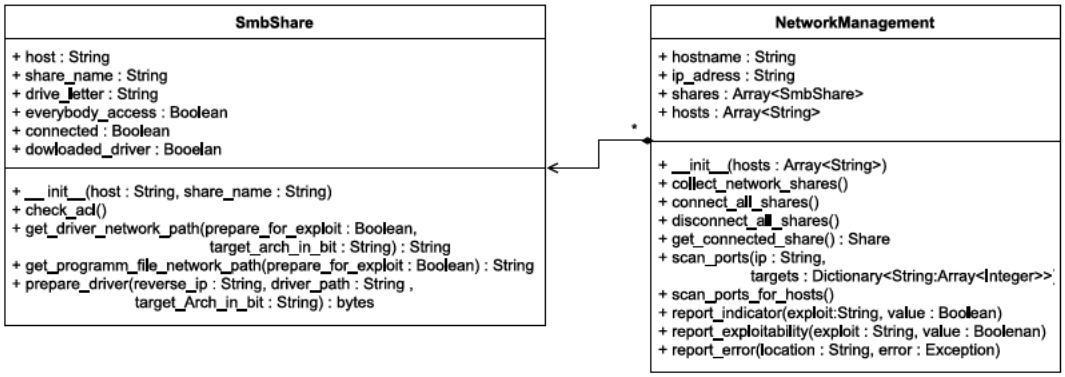

Abb. 3.7: `NetworkManager` und `SmbShare` Klassendiagramme. Eigene Abbildung.

Die erste Funktion ist die `connect_all_shares`, die vor dem Start der Exploits aufgerufen wird, um die erhaltenen Netzwerkshares lokal einzubinden und deren Zugriffsrechte zu erfragen. Wie bereits im Entwicklungstagebuch beschrieben, werden vor allem jene Shares gesucht, die für jeden Nutzer, der der Everyone Gruppe angehört (alle authentifizierten Nutzer, sowie Gast Accounts), zugreifbar sind. Daher wird auch nur diese Option in Form eines Boolean `everyone_access` im `SmbShare` Objekt gespeichert. Ebenfalls wird inner-

halb dieser Funktion gespeichert, ob das Einbinden dieses Shares erfolgreich war und wenn, dann welchen Laufwerkbuchstaben dieser Share erhalten hat. Diese Information wird benötigt, um später die für die Exploits und Verbreitung benötigten Dateien auf diesen Share zu kopieren. Damit zusammenhängend beinhaltet diese Klasse auch eine Funktion `get_connected_shares`, die aus allen verbundenen Shares einen Share zurückgibt, auf das alle Nutzer Zugriff gewährt bekommen. Die letzte Funktion des `NetworkManagers` ist die `disconnect_all_shares` Funktion, die aufgerufen wird, nachdem alle Exploits ausgeführt wurden. Diese entfernt alle in der Funktion `connect_all_shares` verbundenen Shares wieder vom Computer.

Die `SmbShare` Klasse beinhaltet ebenfalls Funktionen, die es vereinfachen die einzelnen Shares für die Exploit zu verwenden. Für die verschiedenen Exploits gibt es hier die Funktion `get_driver_network_path`, welche die Treiberdatei für den PrintNightmare Exploit in diesen Netzwerkordner kopiert und den Pfad dorthin in der Form zurückgibt, wie sie für den PrintNightmare Exploit benötigt wird. Außerdem wird hier in der Treiberdatei die IP-Adresse überschrieben, zu der der rückkommunizierende Prozess sich verbinden soll. Dafür gibt es eine Hilfsfunktion `prepare_driver`. Äquivalent dazu ist die Funktion `get_programm_file_network_path` dafür vorgesehen, die ausführbare Datei des Wurmes in diesen Netzwerkordner zu verschieben und den Pfad zurückzugeben. Um die Zugriffrechte auf einen Netzwerkordner abzufragen, enthält die Klasse `SmbShare` die Funktion `check_acl`, in welcher die Access Control List (ACL) des Ordners abgefragt und bezüglich der Zugriffsrechte analysiert wird.

Außerdem ist im `NetworkManagement` die Funktion `scan_ports_for_hosts` implementiert, die in der Initialisierungsfunktion einem Thread übergeben wird. Diese Funktion ruft nacheinander für jeden Host die Funktion `scan_ports` auf. Das erstellte Thread Objekt wird im `NetworkManager` gespeichert. Mit einer weiteren Funktion `wait_for_scan_thread` wird am Ende der Wurm-Ausführung auf die Beendigung des Threads gewartet. Werden offene Ports gefunden, erfolgt sofort eine Meldung an den Report Server.

Weiterführend beinhaltet das `network_manager.py` Skript auch die Funktion zum Dekodieren der Treiberdatei, da diese kodiert im Wurm eingebettet werden muss, um nicht vom Windows Defender erkannt und blockiert zu werden. Eine genauere Beschreibung dazu folgt im Abschnitt »Verschleierung der Inhalte des Wurmes« (siehe ► Unterabschnitt 3.4.3).

Zuletzt sind in der `NetworkManagement` Klasse die Funktionen zum Melden von Exploit Informationen, Nutzer und Domaindaten enthalten. Die Funktion `report_indicator`

wird genutzt, um Hinweise auf Schwachstellen bei entfernten Rechnern zu melden. Dafür bekommt diese die IP-Adresse des Hosts, auf dem der Hinweis gefunden wurde, übergeben und meldet diese entsprechend dem Report Server über den Endpunkt `/exploit`. Analog dazu wird die Funktion `report_exploitability` genutzt, um die Ausnutzbarkeit eines Exploits zu melden. Dabei muss dieser Funktion ebenfalls übergeben werden, ob eine Schwachstelle erfolgreich ausgenutzt werden konnte. Auch die Funktion zum Melden von Fehlern (`report_error`) wurde hier implementiert und wird von anderen Skripten im Fehlerfall mit Ort des Fehlers und dem Python Exception Objekt aufgerufen. Gemeldet wird ebenfalls die IP des Hosts auf dem der Fehler auftrat.

3.4.1.4 exploit_manager.py Skript

Im `exploit_manager.py` Skript sind die Funktionen zum Testen und Ausnutzen der verschiedenen Sicherheitslücken sowie zum Hinzufügen der Ausschlüsse vom Echtzeit-Scan des Windows Defenders enthalten.

Bevor diese Befehle zum Hinzufügen der Ausschlüsse ausgeführt werden, wird überprüft, ob man sich in einem höheren Kontext als ein normaler Nutzer befindet, um nicht unnötig Fehler zu produzieren. Weiterführend werden in diesem Skript die meisten der in den `support_files` gespeicherten Skripts verwendet, indem die für die Angriffe relevanten Funktionen oder Klassen aus diesen importiert werden. Beispiele hierfür sind die `perform_attack` Funktion, die aus dem Zerologon Exploit Skript (`pyzerologon.py`) importiert und in der Funktion `zerologon` verwendet wird oder die `Exploit` Klasse aus dem `bluekeep.py` Skript, die in der Funktion `bluekeep` verwendet wird.

Außerdem ist in diesem Skript das Scannen der Ports implementiert und die Tests, ob Maschinen Hinweise auf eine potenzielle Schwachstelle geben. Beispiele hierfür wären die `check_bluekeep` und die `check_print_nightmare` Funktionen, die spezielle Ports prüfen, die bestimmten Windows Services zugeordnet sind. In diesem Fall der Port 515 in der Funktion `check_print_nightmare`, da Port 515 der Port des Druckservers von Windows ist. Die Funktion `check_bluekeep` überprüft Port 3389, welcher der Port für den Remote Desktop Protocol Server von Windows ist.

Zuletzt ist in diesem Skript auch die Funktion zum Abhören eines Ports und Übertragen von Kommandos implementiert, die von der `printnighmare` Funktion sowie von der `bluekeep` Funktion genutzt wird.

3.4.1.5 Report Server

Für den Report Server wurde ein eigener Ordner `ReportServer` erstellt (siehe ▶ Abbildung 3.5), der denselben Eltern-Ordner wie der `Worm_Files` Ordner hat. Dies hat den Hintergrund, dass der Report Server ein eigenständiges Programm ist.

Dieser `ReportServer` Ordner enthält, neben dem Hauptskript `server.py`, noch die Datei `classes.py`. In dieser Datei werden die Klassen definiert, die der Server benötig. Teil davon sind die Klassen User und `DomainController`. Aber auch servereigene Klassen werden hier definiert. Ein Beispiel dafür ist die Klasse Host, welche genutzt wird, um die Ergebnisse der einzelnen Scans zu speichern. Dafür besitzt diese Klasse die Felder `ip_address`, `open_ports`, `vulnerability_indications`, `vulnerability_exploited`, `dumped_users`, `dc` und `occured_errors` (siehe ▶ Abbildung 3.8). Dabei sind die Felder `open_ports` und `dumped_users` Listen, in denen Integers (für Ports) und erhaltene Nutzerobjekte gespeichert werden.

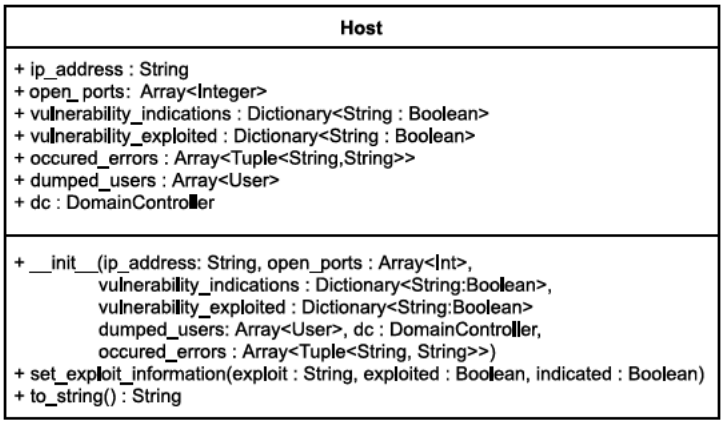

Abb. 3.8: Host Klassendiagramm. Eigene Abbildung.

Die beiden Felder `vulnerability_indications` und `vulnerability_exploited` sind Dictionaries der Form `String:Boolean`, wobei String die Bezeichnung einer Schwachstelle ist und der Boolean Wert angibt, ob ein Hinweis auf diese gefunden wurde bzw. ob diese ausgenutzt wurde. Das Feld `dc` speichert die Domain Controller Informationen, sollte der Host der Domain Controller sein.

Das Hauptskript `server.py` definiert Funktionen, die erhaltene Informationen in eine JSON-Datei speichern und aus dieser Lesen um die Hosts mit neuen Informationen zu erweitern. Außerdem werden hier sämtliche Endpunkte mit den entsprechenden HTTP-Methoden definiert und die dazugehörigen Funktionen zur Verarbeitung der einzelnen Informationen implementiert.

Das Report-Dokument wird bei Beendigung des Servers aus der JSON-Datei generiert und im selben Ordner abgelegt.

3.4.2 Pyinstaller

Wie im vorherigen Abschnitt dargestellt, ist die Logik des Wurmes auf mehrere Python Skripte aufgeteilt. Um diese zu einer ausführbaren Datei zusammenzufügen, wird das Tool PyInstaller im onefile Modus (Argument `--onefile`) verwendet, was rekursiv die einzelnen Skripte, genutzte Bibliotheken und die benötigten Dynamic Link Libraries (.dll) sammelt und in ein Verzeichnis packt, welches bei Start (der generierten ausführbaren Datei) durch den Bootloader in einen temporären Ordner entpackt wird, der nach Beendigung des Programms wieder entfernt wird [16]. Das hat den Vorteil, dass auf dem Zielcomputer kein Python Interpreter installiert sein muss, da auch dieser mit in der ausführbaren Datei enthalten ist.

Im Quellcode des Wurmes wird an verschiedenen Stellen der Pfad des temporären Ordners abgerufen, um z. B. die kodierten, eingebetteten Dateien, wie die Treiberdateien für den PrintNightmare Exploit, zu dekodieren und an eine andere Stelle zu schreiben. Der Pfad dieses Ordners wird während des Entpackungsvorgangs vom Bootloader in die `sys._MEIPASS` Variable geschrieben, nachdem das `frozen` Attribut gesetzt wurde [16, S. 25]. Das Modul `sys` ermöglicht es, auf Variablen, die vom Python Interpreter genutzt werden, zuzugreifen. In diesem Fall die `_MEIPASS` Variable. [57]

```
basedir = sys._MEIPASS
```

Pyinstaller bietet ebenfalls eine UPX Integration, indem man dem Befehl zum Packen der Skripte ein Argument `--upx-dir <Pfad zum UPX Ordner>` anfügt. UPX ist eine Software mit der sich ausführbare Dateien komprimieren lassen, sodass ihr Platzbedarf auf der Festplatte geringer ist. [55]

3.4.3 Verschleierung der Inhalte des Wurmes

Durch die statische Analyse des Windows Defenders wurde unter anderem die Funktion des HiveNightmare Exploits erkannt und somit die Ausführung des Programms nicht gestattet. Dies war ebenfalls der Fall, wenn die Treiberdateien, die einen rückkommunizierenden Prozess mit einer Konsoleninstanz erzeugen, unverschleiert in die Wurmdatei gepackt wurden.

Um dies zu umgehen, müssen die Dateiinhalte so angepasst werden, dass sich deren Signatur ändert und die verdächtigen Werte, wie z. B. Strings, nicht mehr einfach auslesbar und erkennbar sind, sondern erst während der Ausführung zusammengesetzt werden. Im Fall des HiveNightmare Exploits wurde der Pfad erkannt, an welchem die Kopien der SAM, SYSTEM und SECURITY Hives gespeichert sind und musste verschleiert werden. Dafür wurde der String in zwei Substrings unterteilt, die mit der ROT13 Kodierung kodiert wurden. Dadurch wurde aus:

```
Backup_path = f"\\\\\\\\\?\\GLOBALROOT\\Device\\
HarddiskVolumeShadowCopy{i}\\Windows\\System32\\config\\{config_file}"
```

Folgendes:

```
string1 = '\\\\?\\TYBONYEBBG\\Qrivpr\\UneqqvfxIbyhzrFunqbjPbcl'

string2 = '\\Jvaqbjf\\Flfgrz32\\pbasvt\\'

backup_location = f''+codecs.decode(string1,'rot_13')+f'{i}'+
codecs.decode(string2,'rot_13')+f'{config_file}'
```

Diese Unterteilung reichte aus, um die Erkennung durch den Windows Defender zu umgehen.

Die Kodierung der Treiberdateien wurde ähnlich umgesetzt. Da ROT13 eine String Kodierung ist, die Buchstaben verschiebt, kann sie nicht für Binärdateien verwendet werden. Hierfür wurde die Base64 Kodierung gewählt, welche eine binäre Kodierung unterstützt. Mittels eines Python Skriptes wurde die Treiberdatei im Binärformat eingelesen, base64 kodiert und wieder in eine Datei geschrieben. Hier reichte eine einfache Kodierung jedoch nicht aus, um den Defender Scan bestehen zu können. Dies wurde gelöst, indem die bereits kodierte Datei erneut im Binärformat eingelesen wurde und erneut mittels des base64 Verfahrens kodiert wurde. Nachdem die Treiberdatei doppelt kodiert war, konnte sie problemlos im Wurm eingebettet werden, ohne dass dieser dadurch vom Windows Defender als gefährlich eingestuft wurde. Im Wurm wurden die entsprechenden Funktionen zum Dekodieren der Treiberdateien implementiert (siehe Code-Auszug aus der Entpackungsfunktion).

```
basedir = sys._MEIPASS

driver =
open(basedir+"\\driver"+target_arch_in_bit+"_b64x2.dll","rb").read()

for i in range(0,2):
  driver = base64.b64decode(driver)
return driver
```

4 Diskussion

Das Ziel dieser Arbeit war es, den Prototyp eines Software-Tools zu entwickeln, welcher sich wie eine Malware der Malwareklasse Wurm verhält, dabei jedoch keine Schadwirkung besitzt. Das Software-Tool soll in der Lage sein, seine Netzwerkumgebung zu scannen und Hinweise auf potentielle Schwachstellen zu erkennen. Anschließend sollen diese Schwachstellen ausgenutzt werden, um sich im Netzwerk fortzubewegen oder Nutzerdaten zu erhalten. Weiterführend sollen die gefundenen Hinweise, die ausgenutzten Schwachstellen und die erhaltenen Daten für eine Auswertung an eine zentrale Stelle geschickt werden. Dabei ist es wichtig, dass die Software nicht vom Windows Defender als schädlich erkannt und blockiert wird.

Die Entwicklung eines Prototyps war erfolgreich, jedoch fehlen für ein ausgereiftes Tool, welches einen echten Penetrationstester ersetzen bzw. unterstützen kann, noch viele Fähigkeiten. In diesem Kapitel wird auf die Möglichkeiten, aber auch auf die Schwächen sowie fehlenden Fähigkeiten des entwickelten Tools eingegangen.

4.1 Die Möglichkeiten des Tools

Die Software beinhaltet grundsätzlich alle geforderten Funktionalitäten für den Diebstahl von Nutzerdaten und der Verbreitung im Netzwerk. Der entwickelte Prototyp des Wurms wird nicht vom Windows Defender erkannt. Ist ein Computer anfällig für den HiveNightmare Exploit, so ist der Wurm in der Lage, die SAM, SECURITY und SYSTEM Hive auszulesen, daraus NTLM Hashes der Nutzer zu extrahieren und zu melden sowie diese für eine spätere Verwendung zu speichern. Er ist ebenfalls in der Lage, die erhaltenen Nutzer anhand ihrer SID (Security Identifier) grob in Domain Administratoren, lokale Administratoren und Nutzer (sowohl Domain als auch lokal) zu kategorisieren. Diese Logik könnte dahingehend erweitert werden, lokale Nutzer und Domain Nutzer zu unterscheiden. Der Grundstein für diese Funktionalität ist bereits, durch ein Feld `domain` in der `User` Klasse, gelegt. In diesem Feld wird der Domainname gespeichert, sollte einer vorhanden sein. Dies ist jedoch im momentanen Entwicklungsstand noch sehr unzuverlässig.

In der ab hier folgenden Beschreibung wird von einer erfolgreichen Erhöhung der Privilegien ausgegangen, welche eine der größten Schwierigkeiten des Wurmes ist. Die Details der Problematik werden im Abschnitt »Die Probleme und Schwierigkeiten des Tools« (siehe ► Abschnitt 4.2) genauer beleuchtet.

Nach erfolgreicher Privilegienerhöhung, werden Scan-Ausschlüsse des Windows Defenders für alle ausführbaren Dateien (Endung .exe), ein Ordner `mytempdata` im `System32` Ordner und eine Datei in diesem Ordner erstellt. Außerdem wird eine Firewall Regel hinzugefügt und aktiviert, die es dem Wurm ermöglicht Ports zu öffnen und abzuhören. Die erstellten Windows Defender Scan-Ausschlüsse ermöglichen es, Mimikatz zu entpacken und in den erstellten Ordner zu schreiben, da die Datei nicht mehr vom Windows Defender gescannt werden kann. Somit ist die Software ebenfalls in der Lage den laufenden LSASS Prozess und ggf. vorhandene RDP Prozesse nach Klartextpasswörtern oder anderen wertvollen Daten, wie Passwort Hashes, zu durchsuchen.

Außerdem implementiert der Wurm einen Zerologon Angriff auf den Domain Controller mit anschließendem Auslesen der SAM, jedoch zum aktuellen Entwicklungsstand nur über den Maschinenaccount des Domain Controllers, wodurch nur die lokalen Daten des Domain Controllers abgerufen werden können. Hier könnte perspektivisch ein zweiter Schritt implementiert werden, bei dem sich mit den erhaltenen Hashes auf den Domain Controller verbunden wird, um dann die Domain Datenbank auszulesen und alle Domain Nutzerdaten zu erhalten.

Die Exploits zur Weiterverbreitung im Netzwerk sind ebenfalls funktionsfähig, solange die Firewall Regel erstellt werden kann, da ohne die Firewall Regel, das Übertragen von Kommandos zum Kopieren und zum Starten des Wurmes auf dem angegriffenen Computer nicht möglich ist. Ebenfalls Voraussetzung für ein erfolgreiches Ausnutzen der Schwachstellen ist, dass auf mindestens ein Netzwerklaufwerk jeder Nutzer Zugriff gewährt bekommt. Dieses Netzwerklaufwerk wird benötigt, um die Programmdatei sowie den Treiber für den PrintNightmare Exploit zu übertragen. Auch hier müssten zusätzliche Funktionen implementiert werden, um die Zuverlässigkeit des Wurmes zu verbessern. Diese könnten die Funktionalität dahingehend erweitern, dass nicht nur nach für jeden lesbare Netzwerkordnern gesucht wird, sondern auch danach, welche Nutzerdaten bereits erhalten wurden bzw. ob der freigegebene Ordner seinen Ursprung auf dem Zielcomputer hat. Dadurch gäbe es eine größere Auswahl an Netzwerkordnern, die für die Übertragung von Dateien für die Exploits verwendet werden können.

Der BlueKeep Exploit funktioniert ebenfalls, solange ein Netzwerkordner gefunden wer-

den konnte, über den die Programmdatei des Wurmes übertragen werden kann. In der Ausführung dieses Exploits gibt es ein technisches Detail, welches die Funktionalität zwar nicht beeinträchtigt, jedoch in einem Endprodukt nicht vorkommen sollte. Wie im Entwicklungstagebuch bereits erwähnt, werden die Pakete, die die RCE ermöglichen, über ein `reactor` Objekt asynchron übertragen. Da dieser sich nicht beendet, nachdem alle Pakete gesendet wurden, wurde der Startaufruf in einen Thread ausgelagert, um die Ausführung der weiteren Wurm-Logik nicht zu blockieren. Auf diesen Thread wird in der aktuellen Version nicht gewartet und er wird auch nicht aktiv beendet, sondern terminiert mit dem Wurm. Um dies zu lösen, könnte der Exploit umgeschrieben werden, sodass die Pakete nicht über einen `reactor` gesendet werden und die Exploit Funktion zurückkehrt, wenn alle Pakete gesendet wurden.

Der Netzwerk-Scan des Wurmes ist den Anforderungen entsprechend implementiert und scannt die Ports von entfernten Rechnern mit einem zeitlichen Abstand zwischen 30 und 60 Sekunden. Der Port Scan könnte dahingehend erweitert werden, dass eine Service- und Betriebssystemerkennung implementiert wird. Ebenfalls könnte die Nebenläufigkeit des Scans verbessert werden. In der aktuellen Version wird ein Thread gestartet, der nacheinander die Hosts scannt. Dabei wäre jedoch zu beachten, nicht alle Hosts gleichzeitig, sondern versetzt zu scannen, um die Aktivierung von IDS und IPS zu verhindern.

Die Meldefunktion der erhaltenen Daten und der gefundenen Schwachstellen funktioniert ebenfalls. Die gemeldeten Informationen könnten noch verfeinert, vorverarbeitet und erweitert werden. Die in den Anforderungen formulierten Informationen werden bereits jetzt schon gemeldet.

Somit sind alle funktionalen Anforderungen erfüllt. Die nicht-funktionalen Anforderungen 1 und 3 sind ebenfalls erfüllt. Das Ausbleiben einer Schadwirkung (nicht-funktionale Anforderung 4) konnte nicht erreicht werden, ebenso wie eine Kompatibilität mit 32 Bit Systemen (nicht-funktionale Anforderung 2). Da die nicht erreichten Anforderungen keine Voraussetzung für ein MVP sind, erreicht das entwickelte Tool den Status eines MVPs (Minimum Viable Product).

4.2 Die Probleme und Schwierigkeiten des Tools

Wie bereits im vorherigen Abschnitt angerissen, ist die größte Schwierigkeit des Wurmes die Erhöhung der Privilegien. Aufgefallen ist dies erst gegen Ende der Entwicklung durch den Test auf einer komplett neu aufgesetzten virtuellen Maschine. In der Startphase der Entwicklung wurde, wie bereits erwähnt, so begonnen, dass

möglichst wenige Testmaschinen benötigt werden. Somit mussten die lokalen Daten ausreichen, um PsExec testen zu können. Damit dies möglich ist, wurde eine Registry Option (`LocalAccountTokenFilterPolicy`) geändert, die es erlaubt, mit einem lokalen Administrator-Account eine vollwertige Administratorenverbindung zu einem Share aufzubauen. Ohne diese Option ist es nicht möglich, sich mit einem lokalen Administratoraccount mittels PsExec in einen SYSTEM Kontext zu versetzen. Da der eingerichtete Zustand der virtuellen Maschine gesichert wurde, um sie nicht für jeden Test neu aufsetzen zu müssen, wurde während der Entwicklung nicht berücksichtigt, dass diese Option gesetzt ist. Die Folgen für die Funktionalität des Wurmes sind nicht zu vernachlässigen, da das Ausbleiben der Privilegienerhöhung viele neue Schwierigkeiten schafft, die nicht gelöst werden konnten.

Durch die fehlende Erhöhung der Privilegien ist der Wurm nicht in der Lage Mimikatz zu entpacken, da er keine Scan-Ausschlüsse für den Windows Defender erstellen kann. Das hat zur Folge, dass er auch nicht den laufenden LSASS Prozess oder RDP Prozesse nach Nutzerdaten durchsuchen kann. Dies allein würde kein Problem darstellen, da auch mittels einer Zerologon Attacke Nutzerdaten erhalten werden können, die für eine Erhöhung der Privilegien genutzt werden könnten. Jedoch entsteht durch die fehlende Administrator-Berechtigung auch hier ein Problem. Dadurch, dass der Windows Defender Ausschluss für die Wurm-Datei nicht erstellt werden kann, erkennt der Windows Defender die Zerologon Attacke anhand des Verhaltens des Wurmes und terminiert ihn. Der Grund der Erkennung ist nicht offensichtlich. Ein Ansatz, um dieses Problem eventuell zu lösen, wäre das Verzögern der Pakete, da diese aktuell ohne Verzögerung versendet werden.

Die letzte Option ist das Verbreiten auf einen anderen Computer. Durch die Verbreitung wird der Wurm in einem SYSTEM Kontext gestartet und kann somit Windows Defender Ausschlüsse erstellen. Von dem infizierten Computer aus könnte dann eine erneute Zerologon Attacke versucht werden. Jedoch ist auch diese Funktionalität vom Ausbleiben der Privilegienerhöhung betroffen, da ohne die nötigen Berechtigungen keine Firewall Regel erstellt werden kann. Dadurch ist es dem Wurm nicht möglich einen Port des angreifenden Computers abzuhören und Kommandos zum angegriffenen Computer zu senden. Diese Problematik könnte angegangen werden, indem die präparierte Treiberdatei dahingehend geändert wird, dass die Befehle schon darin enthalten sind und den Netzwerkordner nach demselben Verfahren ändert, wie die IP-Adresse, zu der eine Verbindung aufgebaut werden soll. Analog dazu müsste der Shellcode des BlueKeep Exploits angepasst werden, sodass dieser ebenfalls die auszuführenden Befehle in sich trägt.

Die Anforderung der Unterstützung von sowohl 64 Bit als auch 32 Bit Architekturen

konnte nicht vollständig umgesetzt werden. An einigen Stellen im Wurm sind erste Schritte dafür implementiert, werden jedoch zur Zeit nicht genutzt.

Die letzte Abweichung von den Anforderungen ist, dass der Wurm eine Schadwirkung verursacht. Zwar wurde keine Schadwirkung bewusst implementiert, jedoch wird durch einen erfolgreichen Zerologon Angriff das Passwort des Active Directory Objektes des Computers geändert, das von diesem lokal gespeicherte aber nicht. Somit ist nach einer erfolgreichen Attacke die Verbindung von diesem Computer zur Domain gestört und kann nur durch manuelle Aktionen wiederhergestellt werden. Dieses Problem ließe sich lösen, indem man den bereits angesprochenen Extraschritt nach einer Zerologon Attacke implementiert, wodurch alle Nutzer auf dem Domain Controller und das Klartext-Passwort der Maschine erhalten werden. Dieses Passwort könnte wieder installiert werden, nachdem die Nutzerdaten ausgelesen wurden. Auf diese Weise könnte die unerwünschte Nebenwirkung umgangen werden.

4.3 Absicherung gegen den Wurm

Das Ziel des Wurmes ist es, ein Netzwerk auf Sicherheitslücken zu untersuchen und diese ggf. auszunutzen. In diesem Abschnitt wird darauf eingegangen, wie ein Netzwerk gegen die verschiedenen Angriffe des Wurmes gesichert werden kann. Dabei wird der Umstand vernachlässigt, dass der Wurm, wie diese Lücken im lokalen Netzwerk, abgesehen von Updates, geschlossen werden können.

Alle Sicherheitslücken lassen sich inzwischen sehr einfach und zuverlässig über ein Update schließen, da alle in dieser Arbeit benutzten Exploits mindestens ein Jahr alt sind und für jede dieser Lücken bereits Sicherheitsupdates veröffentlicht wurden. Dennoch wird folgend genauer erläutert, wie man diese Lücken im lokalen Netzwerk, abgesehen von Updates, schließen kann.

Der HiveNightmare Exploit funktioniert auf Grundlage von fehlerhaften Zugriffsberechtigungen auf die Volume Shadow Copies in `%windir%\system32\config`. Diese existieren nur, wenn Systemschutz aktiviert ist. Somit könnte man den Systemschutz deaktivieren, was den Nachteil hat, dass man keine Sicherungspunkte des Systems erstellen kann. Alternativ können mithilfe des Befehls »`icacls`« DACLs (Discretionary Access Control Lists) geändert werden. Das heißt die Zugriffsberechtigungen auf Dateien und Ordner können angepasst werden. Durch die Option `/Vererbungsebene` kann die Vererbung der Berechtigungen vom Eltern-Ordner u.a. aktiviert werden. Mit dem Befehl:

```
Icacls /Vererbungsebene:e %windir%\system32\config\*.*
```

erben alle Kind-Dateien des `config` Ordners die Zugriffsberechtigung des `config` Ordners, die nur für einen Administrator einen Zugriff erlauben. [24]

Die PrintNightmare Schwachstelle nutzt den Print Spooler Service. Somit wäre eine Option, diesen zu deaktivieren. Der Nachteil ist, dass die Geräte auf denen der PrintSpooler deaktiviert ist, auch keine Druckaufträge bearbeiten können. Muss von einem Gerät aus jedoch gedruckt werden, so kann diese Option nicht verwendet werden. Alternativ kann der Druckserver innerhalb des Print Spoolers deaktiviert werden, sodass keine anderen Computer auf den Print Spooler zugreifen können und somit auch keine Treiber installiert werden können. Die letzte, auch auf Druckservern anwendbare, Alternative ist das Überprüfen bzw. Setzen der Registry Schlüssel `HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows NT\Printers\PointAndPrint\NoWarningNoElevation` und `\UpdatePromptSettings` auf jeweils Null. Dadurch wird eine Administratorbestätigung für das Hinzufügen eines Treibers benötigt und die Attacke somit erschwert. Zuletzt kann man die Ports 445 und 135 filtern oder schließen, da diese für RPC (Remote Procedure Call – Port 135) und SMB (Port 445) genutzt werden. Sind diese geschlossen, kann der Aufruf zum Installieren des Treibers nicht getätigt werden und es fehlt ebenfalls ein Übertragungsweg für die Treiber-Datei.

Die Zerologon Sicherheitslücke kann nur durch ein Update der entsprechenden Systeme geschlossen werden.

Da EternalBlue Implementationsfehler des SMBv1 Protokolls nutzt, ist hier eine Option das Deaktivieren von SMBv1. Alternativ zu SMBv1 können SMBv2 oder v3 verwendet werden. Da dies nicht in allen Netzwerken einfach umzusetzen ist, ist auch hier die sicherste Möglichkeit das Installieren der entsprechenden Sicherheitsupdates.

Um das Ausnutzen der BlueKeep Sicherheitslücke zumindest deutlich zu erschweren, kann für RDP Network Level Authentication aktiviert werden. Dies verhindert das Verbinden mit einem RDP Kanal ohne vorherige Authentifizierung, jedoch nicht die RCE an sich.

Abschließend lässt sich sagen, dass ein Netzwerk am besten gegen diesen Wurm gesichert werden kann, indem auf allen Computern die entsprechenden Updates in-

stalliert werden. Zwar kann das Ausnutzen einiger Sicherheitslücken durch manuelle Einstellungen erschwert oder verhindert werden, jedoch sollte das Ziel sein, alle Sicherheitslücken und nicht nur die in diesem Wurm getesteten, zu schließen. Der beste und zuverlässigste Weg dafür sind Sicherheitsupdates des Herstellers.

5 Fazit

5.1 Funktionalität der Software

Die moderne Welt wird immer digitalisierter und vernetzter, jedoch wächst die Investitionsbereitschaft in IT-Sicherheit nicht äquivalent dazu, was die Sicherheit der Infrastruktur beeinträchtigt. Deshalb sollte im Rahmen dieser Arbeit ein Prototyp eines Tools entwickelt werden, dessen Funktionsweise der Arbeit eines Penetrationstesters nachempfunden ist, um damit Netzwerktests zu vereinfachen und somit zu vergünstigen. Das Tool sollte sich von den bereits existierenden Schwachstellen-Scannern dadurch unterscheiden, dass nicht nur Hinweise auf Schwachstellen gesucht werden, sondern diese auch ausgenutzt werden sollen. Um diese Vorgaben zu erreichen, wurde ein Tool entwickelt, welches sich analog zur Malwareklasse Wurm verhält, um so das Vorgehen eines Penetrationstesters zu simulieren.

Das Ergebnis der Entwicklung dieses Tools war eine Software, die fast alle geforderten Funktionalitäten implementiert, diese aber nur unter sehr speziellen Voraussetzungen ausführen kann. Die fehlenden Anforderungen sind die Kompatibilität mit 32 und 64 Bit Architekturen sowie das Ausbleiben einer Schadwirkung. Da diese aber nicht für ein MVP gefordert sind, kann der entwickelte Prototyp als MVP bezeichnet werden. Trotzdem ist ein erfolgreicher Einsatz der Software zum Testen eines Netzwerkes unwahrscheinlich und dieser Scan bei Erfolg sehr unvollständig, da das Tool nur einige wenige Schwachstellen testen kann. Das entwickelte Programm verdeutlicht den großen Vorteil eines menschlichen Penetrationstesters: Flexibilität und Anpassungsfähigkeit an verschiedene Gegebenheiten.

Perspektivisch besitzt das Tool das Potenzial, einen Penetrationstester durch einen grundlegenden Netzwerkscan zu unterstützen und seine Arbeit zu erleichtern. Jedoch muss, um einen solchen Stand zu erreichen, noch viel Entwicklungsarbeit geleistet werden, um vorhandene Funktionalitäten im Sinne der Zuverlässigkeit zu verbessern. Ebenfalls müssen deutlich mehr Sicherheitslücken getestet werden können.

Somit ist ein Schwachstellenscanner in Kombination mit einem Penetrationstester deut-

lich effektiver, perspektivisch kann dieses Tool einen Penetrationstest jedoch unterstützen.

5.2 Die Entwicklung

Das Tool wurde mit der Programmiersprache Python in der Entwicklungsumgebung Visual Studio Code entwickelt und in einem Netzwerk aus virtuellen Maschinen getestet. Die Wahl der Programmiersprache wurde gut getroffen, da viele Funktionalitäten bereits in Form von Bibliotheken oder öffentlich zugänglichen Skripten vorhanden waren und kombiniert werden konnten.

Unterstützend für die Übersichtlichkeit war die Projektstruktur, in der die verschiedenen Funktionalitäten in einzelne Skripte und Klassen ausgelagert wurden. Visual Studio Code ermöglichte die Verwaltung dieser Projektstruktur und trug somit dazu bei, den Fokus auf die Entwicklung zu legen und nicht auf die Verwaltung des Projektes. Ebenfalls hilfreich war die Versionsverwaltungssoftware Git, mit welcher Änderungen am Quellcode schrittweise gespeichert werden. Git ermöglicht somit das Zurücksetzen von Änderungen, die nicht zielführend sind. Dadurch wurden Fehler leicht rückgängig gemacht.

Das größte Hindernis in der Entwicklung war die Hardwarelimitierung des Autors, da für mehrere virtuelle Maschinen viel Rechenleistung benötigt wurde, was zu Performance-Einbußen innerhalb der einzelnen Maschinen führte und somit Tests des Tools zeitaufwändig machte. Ebenfalls problematisch war das Erstellen von Sicherungspunkten von virtuellen Maschinen, nachdem Einstellungen verändert wurden, die die Funktionalität des Wurmes veränderten. Dadurch entstanden falsche Annahmen über die Fähigkeiten des Tools, die ohne diese Einstellung nicht vorhanden waren. Andererseits war das Erstellen von Sicherungspunkten wichtig, um Maschinen nach einem Angriff auf diese Sicherungspunkte zurücksetzen zu können. Das war nötig, um immer dieselben Grundvoraussetzungen für einen Exploit zu gewährleisten und das Ergebnis somit nicht zu verfälschen.

Ein weiteres Hindernis für die Entwicklung des Wurmes war der Debugging Prozess. Hier war es ab dem Zeitpunkt der Privilegienerhöhung nicht möglich, Konsolenausgaben der Software zu erhalten. Das erschwert die Fehlersuche, war aber ein nebensächliches Problem, da es andere Wege zum Erhalt der Ausgaben und Fehlermeldungen gab, die genutzt wurden. Dennoch hätte dieses Problem leicht durch eine frühere Implementierung des Report Servers umgangen werden können, da Fehler diesem hätten gemeldet werden können.

Zusammenfassend lässt sich sagen, dass die Entwicklung gut geplant war und die Wahl der Werkzeuge gut getroffen wurde. Jedoch verlangsamte die zur Verfügung stehende Hardware die Entwicklung. Außerdem ist es empfehlenswert, die getätigten Einstellungsänderungen in virtuellen Maschinen zu vermerken und die Sicherungspunkte gut überlegt zu erstellen.

5.3 Ausblick

Wie bereits in den vorherigen Abschnitten erwähnt, ist der Wurm zwar funktionsfähig und erfüllt alle Anforderungen an ein MVP. Von diesem Stand der Entwicklung ausgehend gibt es jedoch viele verbesserungswürdige und fehlende Funktionen, deren Verbesserung bzw. Implementation den Nutzen des Tools deutlich erweitern würde.

Der wichtigste dieser Punkte ist die Verbesserung des Zerologon Exploits dahingehend, dass dieser nicht mehr von Windows Defender erkannt wird, sollte kein Scan-Ausschluss erstellt worden sein. Damit könnte erreicht werden, dass ohne den HiveNightmare Exploit trotzdem Nutzerdaten erhalten werden und anschließend für eine nachträgliche Erhöhung der Privilegien genutzt werden können. Außerdem könnte die Nutzerdaten-verarbeitende Funktion dahingehend erweitert werden, Kerberos Daten speichern, verwalten und nutzen zu können.

Eine weitere relevante Verbesserung wäre das Erweitern des PrintNightmare Treibers und des BlueKeep Shellcodes dahingehend, dass die auszuführenden Befehle nicht von einem anderen Computer gesendet werden müssen. Dies würde die Möglichkeit eröffnen, ohne Administratorberechtigungen auf dem angreifenden Computer RCE Schwachstellen auszunutzen, da kein Port in der Firewall freigegeben werden müsste. Außerdem könnten die Funktionen zum Arbeiten mit SMB Shares dahingehend erweitert werden, dass sie die Zugriffsberechtigungen auf ein Share genauer analysieren und mit bereits erhaltenen Nutzern vergleichen. Damit würde die Möglichkeit geschaffen werden, mehr Shares für eine Verbreitung zu nutzen.

Wichtig für einen tatsächlichen Nutzen im Rahmen eines Penetrationstestes wäre das Implementieren einer Funktion, die das Domain Controller Passwort wieder auf den Stand vor der Attacke setzt. Damit würde die einzige Schadwirkung des Wurmes entfernt werden. Ebenfalls wichtig für einen tatsächlichen Nutzen bei einem Penetrationstest wäre das Implementieren weiterer Sicherheitslücken, die getestet werden sollen, um so die Aussagekraft des Testes zu verbessern. Um mehr Computer testen zu können, könnte ebenfalls die Kompatibilität mit den verschiedenen System Architekturen umge-

setzt werden.

Abschließend kann festgestellt werden, dass das Toll an viele Stellen verbessert werden kann, um den Nutzen deutlich zu erhöhen.

Stichwortverzeichnis

Q

R

S

T

U

V

W

Z

Literaturverzeichnis

[1] C. Baars, F. Flade, G. Mascolo und NDR/WDR, *Spähsoftware: Wie "Pegasus" aufs Handy kommt*, tagesschau, Hrsg., 2021. Adresse: https://www.tagesschau.de/investigativ/ndr-wdr/spaeh-software-pegasus-smartphone-101.html (Zugriff am: 04. 09. 2022).

[2] C. Baars, F. Flade, G. Mascolo und NDR/WDR, *Trojaner "Pegasus": Wie autoritäre Staaten ihre Gegner ausspähen*, tagesschau, Hrsg., 2021. Adresse: https://www.tagesschau.de/investigativ/ndr-wdr/spaeh-software-pegasus-projekt-101.html (Zugriff am: 04. 09. 2022).

[3] A. Berg und S. Selen, *Wirtschaftsschutz 2021*, bitkom, Hrsg., 2021. Adresse: https://www.bitkom.org/sites/default/files/2021-08/bitkom-slides-wirtschaftsschutz-cybercrime-05-08-2021.pdf (Zugriff am: 14. 07. 2022).

[4] BetaFred und M. Dressman, *Microsoft-Sicherheitsbulletin MS17-010 - Kritisch*, Microsoft Corporation, Hrsg., 2021. Adresse: https://docs.microsoft.com/de-de/security-updates/securitybulletins/2017/ms17-010 (Zugriff am: 19. 08. 2022).

[5] R. Bhargava, *Why Google IAM is Afraid of AD*, JumpCloud Inc., Hrsg., 2017. Adresse: https://jumpcloud.com/blog/google-identity-management-active-directory (Zugriff am: 04. 09. 2022).

[6] C. Billinis, *Bypassing Windows Defender Runtime Scanning*, WithSecure, Hrsg., 2020. Adresse: https://labs.withsecure.com/blog/bypassing-windows-defender-runtime-scanning/ (Zugriff am: 28. 07. 2022).

[7] A. Binduf, H. O. Alamoudi, H. Balahmar, S. Alshamrani, H. Al-Omar und N. Nagy, "Active Directory and Related Aspects of Security," *2018 21st Saudi Computer Society National Computer Conference (NCC)*, S. 4474–4479, 2018. DOI: 10.1109/NCG.2018.8593188.

[8] D. Borchers, *25 Jahre Windows: Krieg und Frieden*, Heise Medien, Hrsg., 2010. Adresse: https://www.heise.de/newsticker/meldung/25-Jahre-Windows-Krieg-und-Frieden-1139493.html (Zugriff am: 09. 09. 2022).

[9] Bundesamt für Sicherheit in der Informationstechnik, “Die Lage der IT-Sicherheit in Deutschland 2021,” 2021. Adresse: https://www.bmi.bund.de/SharedDocs/downloads/DE/publikationen/themen/it-digitalpolitik/bsi-lagebericht-cybersicherheit-2021.pdf?__blob=publicationFile&v=4 (Zugriff am: 14. 07. 2022).

[10] Bundesamt für Sicherheit in der Informationstechnik, *Malware*, BSI, Hrsg., 2021. Adresse: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Gefaehrdungen/Malware/malware_node.html (Zugriff am: 21. 07. 2022).

[11] Bundesamt für Sicherheit in der Informationstechnik, *Trojaner - wie erkenne ich getarnte Schadprogramme?* BSI, Hrsg., 2021. Adresse: https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Methoden-der-Cyber-Kriminalitaet/Schadprogramme/Trojaner/trojaner_node.html (Zugriff am: 21. 07. 2022).

[12] Bundesamt für Sicherheit in der Informationstechnik, *Viren und Würmer*, BSI, Hrsg., 2021. Adresse: https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Methoden-der-Cyber-Kriminalitaet/Schadprogramme/Viren-und-Wuermer/viren-und-wuermer_node.html (Zugriff am: 21. 07. 2022).

[13] Bundesamt für Sicherheit in der Informationstechnik, *Maßnahmen zum Schutz vor Emotet und gefährlichen E-Mails im Allgemeinen*, BSI, Hrsg., 2022. Adresse: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Gefaehrdungen/Malware/Emotet/emotet_node.html (Zugriff am: 06. 09. 2022).

[14] Bundeskriminalamt, *Polizeilich erfasste Fälle von Cyberkriminalität im engeren Sinne in Deutschland von 2007 bis 2021*, BKA, Hrsg., 2022. Adresse: https://de.statista.com/statistik/daten/studie/295265/umfrage/polizeilich-erfasste-faelle-von-cyberkriminalitaet-im-engeren-sinne-in-deutschland/ (Zugriff am: 14. 07. 2022).

[15] C. Burdova, *Was ist EternalBlue und warum ist der Exploit MS17-010 immer noch relevant?* Avast, Hrsg., 2020. Adresse: https://www.avast.com/de-de/c-eternalblue (Zugriff am: 14. 08. 2022).

[16] D. Cortesi, *PyInstaller Documentation: Release 5.3+ga54d89e1*, 2022. Adresse: https://buildmedia.readthedocs.org/media/pdf/pyinstaller/latest/pyinstaller.pdf (Zugriff am: 19. 08. 2022).

[17] cube0x0, *cube0x0 / CVE-2021-1675: CVE-2021-1675 / CVE-2021-34527*, GitHub, Hrsg., 2021. Adresse: https://github.com/cube0x0/CVE-2021-1675 (Zugriff am: 25. 08. 2022).

[18] P. Duckling, *Zerologon - hacking Windows servers with a bunch of zeros*, Naked Security, Hrsg., 2020. Adresse: https://nakedsecurity.sophos.com/2020/09/17/zerologon-hacking-windows-servers-with-a-bunch-of-zeros/ (Zugriff am: 19. 08. 2022).

[19] P. Duckling, *Windows "HiveNightmare" bug could leak passwords - here's what to do!* Naked Security, Hrsg., 2021. Adresse: https://nakedsecurity.sophos.com/2021/07/21/windows-hivenightmare-bug-could-leak-passwords-heres-what-to-do/ (Zugriff am: 03. 09. 2022).

[20] J. Fechner, *Emotet Virus erkennen und sich davor schützen*, THE BRISTOL GROUP Deutschland GmbH, Hrsg., 2020. Adresse: https://www.bristol.de/emotet-virus/ (Zugriff am: 12. 09. 2022).

[21] D. Francis, *Mastering Active Directory: Deploy and Secure Infrastructures with Active Directory, Windows Server 2016, and PowerShell*, 2nd ed. Birmingham: Packt Publishing Limited, 2019.

[22] Gartner, *Prognose zu den weltweiten IT-Ausgaben von 2012 bis 2023*, Gartner, Hrsg., 2022. Adresse: https://de.statista.com/statistik/daten/studie/703143/umfrage/weltweite-it-ausgaben/ (Zugriff am: 14. 07. 2022).

[23] J. Gerend, dknappettmsft, K. Downie u. a., *Kerberos Authentication Overview*, Microsoft Corporation, Hrsg., 2022. Adresse: https://docs.microsoft.com/de-de/windows-server/security/kerberos/kerberos-authentication-overview (Zugriff am: 28. 07. 2022).

[24] J. Gerend, dknappettmsft, K. Toliver u. a., *icacls*, Microsoft Corporation, Hrsg., 2022. Adresse: https://docs.microsoft.com/de-de/windows-server/administration/windows-commands/icacls (Zugriff am: 03. 09. 2022).

[25] J. Gerend, A. Yaport-Garcia, dknappettmsft u. a., *NTLM Overview*, 2022. Adresse: https://docs.microsoft.com/de-de/windows-server/security/kerberos/ntlm-overview (Zugriff am: 13. 09. 2022).

[26] N. Grossman, *EternalBlue - Everything There Is To Know*, Check Point Software Technologies LTD, Hrsg., 2017. Adresse: https://research.checkpoint.com/2017/eternalblue-everything-know/ (Zugriff am: 19. 08. 2022).

[27] Harris Poll, *Verursachte Kosten aufgrund von Cyberkriminalität in ausgewählten Ländern im Jahr 2020*, NortonLifeLock, Hrsg., 2021. Adresse: https://de.statista.com/statistik/daten/studie/1243469/umfrage/kosten-durch-cyberkriminalitaet-in-ausgewaehlten-laendern/ (Zugriff am: 14. 07. 2022).

[28] M. Humpa, *Nutzer klammern sich an Windows 10: Dagegen hat Windows 11 aktuell keine Chance*, Chip.de, Hrsg., 2022. Adresse: https://www.chip.de/news/Nutzer-klammern-sich-an-Windows-10-Dagegen-hat-Windows-11-aktuell-keine-Chance_182738752.html (Zugriff am: 12. 09. 2022).

[29] Industrie- und Handelskammer zu Berlin, *Thema des Monats: Studie zur Digitalisierung in der Arbeitswelt*, IHK, Hrsg., 2017. Adresse: https://www.ihk.de/berlin/politische-positionen-und-statistiken-channel/arbeitsmarkt-beschaeftigung/fachkraeftesicherung/digitalisierung-der-arbeitswelt/tdm-7-8-2017-digitalisierung-arbeitswelt-3768630 (Zugriff am: 14. 07. 2022).

[30] R. Kimura, Y. Koike, N4NU und Y. Takesako, *RICSecLab / CVE-2019-0708: CVE-2019-0708 (BlueKeep) pre-auth RCE POC on Windows7*, GitHub, Hrsg., 2022. Adresse: https://github.com/RICSecLab/CVE-2019-0708 (Zugriff am: 27. 08. 2022).

[31] S. Knoppik, “Angst und Erpressung: Malware verstehen und abwehren,” *Mitteldeutsche Zeitung*, 4.12.2012. Adresse: https://www.mz.de/varia/angst-und-erpressung-malware-verstehen-und-abwehren-2166280 (Zugriff am: 12. 09. 2022).

[32] F. Labelle, O. Bilodeau, É. Gonzalez u. a., *GoSecure / pyrdp: PyRDP*, GitHub, Hrsg., 2022. Adresse: https://github.com/GoSecure/pyrdp (Zugriff am: 26. 08. 2022).

[33] N. Latto, *Was ist WannaCry?* Avast Software s.r.o., Hrsg., 2020. Adresse: https://www.avast.com/de-de/c-wannacry (Zugriff am: 21. 07. 2022).

[34] H. Liang und olprod, *Beschreibung der Benutzerkontensteuerung und Remoteeinschränkungen in Windows Vista*, Microsoft Corporation, Hrsg., 2022. Adresse: https://docs.microsoft.com/de-de/troubleshoot/windows-server/windows-security/user-account-control-and-remote-restriction (Zugriff am: 30. 08. 2022).

[35] P. Loshin und M. Cobb, *Kerberos*, TechTarget, Hrsg., 2021. Adresse: https://www.techtarget.com/searchsecurity/definition/Kerberos (Zugriff am: 28. 07. 2022).

[36] Malwarebytes, *Emotet-Schadsoftware – eine Einführung in den Banking-Trojaner*, Malwarebytes, Hrsg., 2019. Adresse: https://de.malwarebytes.com/emotet/ (Zugriff am: 12. 09. 2022).

[37] P. Matarazzo, *Lokale Konten (Windows 10) - Windows security*, Microsoft Corporation, Hrsg., 2022. Adresse: https://docs.microsoft.com/de-DE/windows/security/identity-protection/access-control/local-accounts (Zugriff am: 13. 09. 2022).

[38] T. Matthews, *Active Directory: a New Product for a New Millennium*, Exabeam, Hrsg., 2019. Adresse: https://www.exabeam.com/information-security/cybersecurity-calendar-active-directory/ (Zugriff am: 04. 09. 2022).

[39] Microsoft Corporation, *Windows 7 Professional SP1 (32 bit and 64 bit ISOs)*, 2011. Adresse: https://archive.org/details/win-7-pro-32-64-iso (Zugriff am: 12. 09. 2022).

[40] Microsoft Corporation, *Sicherheitsleitfaden für NTLMv1 und LM-Netzwerkauthentifizierung*, 2017. Adresse: https://support.microsoft.com/de-de/topic/sicherheitsleitfaden-f%C3%BCr-ntlmv1-und-lm-netzwerkauthentifizierung-da2168b6-4a31-0088-fb03-f081acde6e73 (Zugriff am: 13. 09. 2022).

[41] Microsoft Corporation, *[MS-CIFS] - v20201001: Common Internet File System (CIFS) Protocol*, 2020. Adresse: https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-CIFS/%5bMS-CIFS%5d.pdf (Zugriff am: 19. 08. 2022).

[42] Microsoft Corporation, *[MS-NRPC] - v20200826: Netlogon Remote Protocol*, 2020. Adresse: https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-NRPC/%5bMS-NRPC%5d-200826.pdf (Zugriff am: 18. 08. 2022).

[43] Microsoft Corporation, *[MS-RDPEFS] - v20210625: Remote Desktop Protocol: File System Virtual Channel Extension*, 2021. Adresse: https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-RDPEFS/%5bMS-RDPEFS%5d.pdf (Zugriff am: 19. 08. 2022).

[44] Microsoft Corporation, *[MS-RPRN] - v20211006: Print System Remote Protocol*, 2021. Adresse: https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-RPRN/%5bMS-RPRN%5d.pdf (Zugriff am: 19. 08. 2022).

[45] Microsoft Corporation, *[MS-RDPBCGR] - v20220903: Remote Desktop Protocol: Basic Connectivity and Graphics Remoting*, 2022. Adresse: https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-RDPBCGR/%5bMS-RDPBCGR%5d.pdf (Zugriff am: 19. 08. 2022).

[46] Microsoft Corporation, *Microsoft Common Criteria Security Target: Microsoft Windows Common Criteria Evaluation: Version: 0.03*, 2022. Adresse: https://download.microsoft.com/download/a/5/6/a5650848-e86a-4554-bb13-1ad6ff2d45d2/Windows%2010%202004%20GP%20OS%20Security%20Target.pdf (Zugriff am: 23. 07. 2022).

[47] Microsoft Corporation, *Windows 7 - Lifecycle*, Microsoft Corporation, Hrsg., 2022. Adresse: https://docs.microsoft.com/de-de/lifecycle/products/windows-7 (Zugriff am: 14. 07. 2022).

[48] Microsoft Security Response Center, *Remote Desktop Services Remote Code Execution Vulnerability: CVE-2019-0708*, Microsoft Corporation, Hrsg., 2019. Adresse: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2019-0708 (Zugriff am: 19. 08. 2022).

[49] Microsoft Security Response Center, *Netlogon Elevation of Privilege Vulnerability: CVE-2020-1472*, Microsoft Corporation, Hrsg., 2020. Adresse: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2020-1472 (Zugriff am: 19. 08. 2022).

[50] Microsoft Security Response Center, *Windows Elevation of Privilege Vulnerability: CVE-2021-36934*, Microsoft Corporation, Hrsg., 2021. Adresse: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-36934 (Zugriff am: 03. 09. 2022).

[51] Microsoft Security Response Center, *Windows Print Spooler Remote Code Execution Vulnerability: CVE-2021-34527*, Microsoft Corporation, Hrsg., 2021. Adresse: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34527 (Zugriff am: 19. 08. 2022).

[52] D.-j. Mollema, *dirkjanm / CVE-2020-1472: CVE-2020-1472 POC*, GitHub, Hrsg., 2020. Adresse: https://github.com/dirkjanm/CVE-2020-1472 (Zugriff am: 23. 08. 2022).

[53] J. Morrison, *What Is a Shellcode?* 2022. Adresse: https://www.easytechjunkie.com/what-is-a-shellcode.htm (Zugriff am: 13. 09. 2022).

[54] I. Nadler, *Die Geschichte von Windows: Unser Fenster zur digitalen Welt*, 2020. Adresse: https://news.microsoft.com/de-de/features/windows-geschichte/ (Zugriff am: 09. 09. 2022).

[55] M. F. Oberhumer, L. Molnár und J. F. Reiser, *UPX: the Ultimate Packer for eXecutables*, GitHub, Hrsg., 2020. Adresse: https://upx.github.io/ (Zugriff am: 30. 08. 2022).

[56] K. Pollack, *NTLM v1 and v2 vs Kerberos*, CalCom, Hrsg., 2021. Adresse: https://www.calcomsoftware.com/ntlm-v1-and-v2-vs-kerberos/ (Zugriff am: 13. 09. 2022).

[57] Python Software Foundation, *sys — System-specific parameters and functions: Python 3.10.6 documentation*, Python Software Foundation, Hrsg., 2022. Adresse: https://docs.python.org/3/library/sys.html (Zugriff am: 05. 09. 2022).

[58] Red Hat, *Was bedeutet CVE?* Red Hat, Inc., Hrsg., 2020. Adresse: https://www.redhat.com/de/topics/security/what-is-cve (Zugriff am: 01. 09. 2022).

[59] J. Ricketts, L. O'Connor, J. Flores, aaburnley, A. Buck und B. Winter, *Secure on-premises computer accounts*, Microsoft Corporation, Hrsg., 2022. Adresse: https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/service-accounts-computer (Zugriff am: 25. 08. 2022).

[60] M. Russinovich, *Sysinternals Suite*, Microsoft Corporation, Hrsg., 2022. Adresse: https://docs.microsoft.com/de-de/sysinternals/downloads/sysinternals-suite (Zugriff am: 30. 08. 2022).

[61] StatCounter, *Desktop Operating System Market Share Worldwide*, StatCounter, Hrsg., 2022. Adresse: https://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-202207-202207-bar (Zugriff am: 14. 07. 2022).

[62] Trend Micro, *What Is Zerologon?* Trend Micro Incorporated, Hrsg., 2022. Adresse: https://www.trendmicro.com/en_au/what-is/zerologon.html (Zugriff am: 05. 09. 2022).

[63] D. Vangel, *Microsoft Defender Antivirus in Windows*, Microsoft Corporation, Hrsg., 2022. Adresse: https://docs.microsoft.com/de-de/microsoft-365/security/defender-endpoint/microsoft-defender-antivirus-windows?view=o365-worldwide (Zugriff am: 05. 09. 2022).

[64] A. Viviano, *User mode and kernel mode*, Microsoft Corporation, Hrsg., 2021. Adresse: https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode (Zugriff am: 24. 07. 2022).

[65] W. Wangwarunyoo und C. Clauss, *worawit / MS17-010: MS17-010*, GitHub, Hrsg., 2018. Adresse: https://github.com/worawit/MS17-010 (Zugriff am: 25. 08. 2022).

[66] Whitehat, *Active Directory Hacking: Angriffe mit mimikatz - Pass-the-Hash (PtH)*, Whitehat, Hrsg., 2021. Adresse: https://www.whitehat.de/active-directory-hacking-angriffe-mit-mimikatz/pass-the-hash-pth (Zugriff am: 28. 07. 2022).

[67] T. Yan und J. Chen, *Exploitation of Windows CVE-2019-0708 (BlueKeep): Three Ways to Write Data into Kernel with RDP PDU*, Palo Alto Networks, Inc., Hrsg., 2019. Adresse: https://unit42.paloaltonetworks.com/exploitation-of-windows-cve-2019-0708-bluekeep-three-ways-to-write-data-into-the-kernel-with-rdp-pdu/ (Zugriff am: 19. 08. 2022).

[68] T. Yan und J. Chen, *Exploitation of Windows RDP Vulnerability CVE-2019-0708 (BlueKeep): Get RCE with System Privilege Using Refresh Rect PDU and RDPDR Client Name Request PDU*, Palo Alto Networks, Inc., Hrsg., 2020. Adresse: https://unit42.paloaltonetworks.com/cve-2019-0708-bluekeep/ (Zugriff am: 19. 08. 2022).

[69] S. Zarinkhou, H. Nachmias, O. Biderman und D. Vazgiel, *Demystifying the Print-Nightmare vulnerability*, Sygnia, Hrsg., 2021. Adresse: https://blog.sygnia.co/demystifying-the-print-nightmare-vulnerability (Zugriff am: 19. 08. 2022).

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit – »Konzeption und prototypische Implementierung eines Tools zur automatischen Schwachstellenanalyse in einem Active Directory« – selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

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

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 16.09.2022

Karl Büchner