# Verfahren zum Testen der IT-Sicherheit an Unternehmen

Vorstellung eines Social-Engineering Angriffsvektors

**Marcus Schilling**
Informatik und Kommunikationssysteme, M. Eng.
Hochschule Merseburg
Wintersemester 2017/2018

**Prof. Dr. Andre Döring**
Fachbereich Wirtschaftswissenschaften
Hochschule Merseburg

**Prof. Dr. Uwe Heuert**
Fachbereich Ingenieurswissenschaften
Hochschule Merseburg

# Inhaltsverzeichnis

# 1 Problematik

In den 70er Jahren des 20. Jahrhunderts wurde Technologie für Privatpersonen zugänglich, die sonst eher der Forschung oder anderen Einrichtungen, wie beispielsweise dem Militär vorbehalten waren. Bis zu diesem Zeitpunkt waren fortschrittliche Kommunikations- und Datenverarbeitungsanlagen so teuer, dass nur wenige, vertrauenswürdige Personen Zugang dazu bekommen konnten. So wurden entsprechende Protokolle und Netzwerkhardware eher auf einen stabilen und zuverlässigen Betrieb ausgelegt, statt auf Datensicherheitsaspekte, wie Authentifizierungsmechanismen oder ähnliches zu achten. Dazu trug auch der Irrglaube bei, PCs würden niemals interessant für Privatanwender, da sie keinen Nutzen davon hätten. Dieser Meinung waren auch prominente Computerhersteller, wie Ken Olsen, Präsident der Digital Equipment Corporation:

> Es gibt keinen Grund dafür, dass irgendjemand einen Computer bei sich zu Hause haben will. [33, S. 27]

Dennoch gab es zu der Zeit schon längst die Idee von selbst replizierenden Computerprogrammen, wie John von Neumann 1949 im Artikel „Theory of self-reproducing automata“ beschrieb. Bob Thomas, Mitarbeiter der BBN Technologies und Mitentwickler des ARPANET[1], gelang mit dem „Creeper System“ die Entwicklung eines (aus heutiger Sicht bezeichneten) Computerwurms, der sich über das ARPANET auf bestimmte, angeschlossene Geräte verbreiten konnte. Dieser war zwar nicht gefährlich, aber so erfolgreich, dass Ray Tomlinson[2] ein gleichartiges Programm („Reaper“) schrieb, welches diesen Wurm von betroffenen Computern löschte.

Erste Computer für Privatanwender, wie der Altair 8800, der Apple I und II, oder auch der Commodore 64 besaßen keinerlei Netzwerkhardware und waren eher für technikbegeisterte Bastler, die darauf programmierten oder spielten. Besonders technikaffine und ideenreiche Bastler, die versuchten, die Grenzen des Möglichen zu erreichen und bestenfalls nach oben zu verschieben, wurden als Hacker bezeichnet. Es folgte ein reger Austausch von selbstgeschriebener Software über Disketten oder Kassetten, das sogenannte „Pantoffelnetzwerk“ [19, S. 24]. Das Problem, dass sich Schadsoftware ungewollt verbreiten kann, besteht durch den Dateiaustausch auch schon hier.

Damals waren diese Programme eher als Scherz, oder auch als Proof-of-Concept für potenziell gefährliche Angriffsvektoren gedacht. Mit dem zunehmenden Ausbau von Infrastruktur, der zunehmenden Vernetzung von Computern und der damit einhergehenden Verlagerung von Diensten in das Internet, werden auch die Angriffsoberflächen und die potenzielle Beute immer größer. Während damals die Anzeige des Textes „I'm the Creeper: Catch me, if you can“ der ganze Schaden war, den das Programm anrichten

[1] Das ARPANET gilt als der Vorläufer des heutigen Internets. Es war das erste Computernetzwerk, welches auf eine hochverteilte Infrastruktur setzte und somit weniger Fehleranfällig sein sollte, als das damalige Telefonnetz. [33, S. 80–85]

[2] Raymond S. Tomlinson war Programmierer und gilt als Pionier des Internets und Erfinder der E-Mail. Er betrieb das erste Mailboxsystem im ARPANET, welches auch Kommunikation zwischen unterschiedlichen Computermodellen ermöglichte. [34]

konnte, ist Malware heutzutage in der Lage, Infrastruktur bestimmter, unsicherer Anlagen zu zerstören[3], Daten oder Identitäten zu stehlen[4], den Computer des Opfers für kriminelle Zwecke in Form von Botnetzen zu missbrauchen oder in anderer, vom Opfer unerwünschter Weise zu manipulieren.

Da ursprünglich aus dieser Szene stammend, ist der Begriff „Hacker“ durch das kriminelle Vorgehen solcher Angreifer („Cracker“) sehr in Verruf geraten. Eine Definition des Begriffes ist zwar schwer, jedoch ist folgende Abgrenzung für diese Arbeit wichtig:

**White Hat** Die ursprüngliche Definition von Hacker. Mitglieder der Hackerszene sehen sich stets als White Hats und fördern den kreativen Umgang mit Computern und Technologie.

**Black Hats** Kriminelle, die ihr Wissen für ihre eigenen Interessen ausnutzen und dabei Dritten einen Schaden zufügen, indem sie in ihre Systeme einbrechen und Daten stehlen. Diese werden dann Gegenstand von Erpressungen oder verkauft.

**Grey Hats** Eine Mischform. Grey Hats verschaffen sich auch illegalen Zugang zu Systemen Dritter, jedoch ohne die Absicht, Schaden anzurichten. Viele Grey Hats melden die gefundenen Schwachstellen dem Betreiber des anfälligen Systems und wollen damit vor Sicherheitslücken warnen oder die allgemeine Computersicherheit erhöhen. Dieses Vorgehen wird durch Bug Bounty Programme[5] zudem immer weiter von den betroffenen Firmen selbst unterstützt und zahlen für gemeldete Sicherheitslücken sogar freiwillig einen Geldbetrag an den Finder.

Durch die zunehmende Komplexität von Programmen, die über das Internet frei zugänglich sind und der daran angeschlossenen Hardware, die immer performanter, individueller und somit komplexer wird, sind die Wege, solche Systeme anzugreifen, stark gewachsen. Das Katz- und Mausspiel zwischen Creeper und Reaper damals trägt auch heute noch einen symbolträchtigen Charakter für die Verbreitung und Bekämpfung aktueller Malware und Angriffstechniken. Cracker, insbesondere Script-Kiddies[6] finden immer einfachere Wege, beispielsweise Viren-Toolkits, um immer mehr verschiedenartige Malware zu bauen. Antiviren-Hersteller versuchen, über Honeypots[7] und ähnliche Techniken, Malware aufzuspüren und zu analysieren, um diese später zuverlässig erkennen und abwehren zu können. Versierte Hacker jeglicher Art finden immer kreativere Wege, Sicherheitsmechanismen auszuhebeln oder Schwachstellen zu finden und auszunutzen („Zero-Day-Exploits“), die sich nicht mit Antiviren-Programmen erkennen lassen.

[3] Wie geschehen bei Stuxnet

[4] Von Identitätsdiebstahl spricht man dann, wenn man so viele Daten über ein Opfer in Erfahrung bringen konnte, dass der Angreifer sich mit diesen als das Opfer ausgeben kann.

[5] `https://www.bugcrowd.com/bug-bounty-list/`

[6] Script-Kiddies werden Personen mit wenig Know-How bezeichnet, die für ihre Angriffe ausschließlich vorgefertigte Skripte und Programme verwenden, ohne die zugrundeliegende Angriffstechnik zu kennen.

[7] Honeypots sind absichtlich anfällige Systeme, die sich als attraktives Ziel für Malware ausgeben, um sich einem Angriff auszusetzen und dabei aktuelle Angriffstechniken analysieren zu können.

## 1.1 Malware

Als Malware wird jegliche Software bezeichnet, die für einen Anwender unerwünschte oder gar schädliche Funktionen aufweist. Es gibt viele Unterkategorien von Schadsoftware, die nach ihrer Verbreitung und ihrem Schadenspotenzial klassifiziert wird. In dieser Arbeit wird hauptsächlich die Rede sein von

**Trojanern** Programme, die einen Mehrwert oder den Anschein eines bestimmten Nutzens für einen Nutzer darstellen, bei der Ausführung jedoch heimlich weitere Aktionen ausführt, von denen der Nutzer nichts weiß und die gegebenenfalls schädlich sind. Häufig dienen Trojaner nur dazu, weitere Schadsoftware zu installieren. Solche Trojaner werden wiederum als „Dropper“ bezeichnet. Diese Form von Malware verbreitet sich nicht selbst, sondern bietet sich dem Nutzer an mit dem Versprechen, für ihn nützlich zu sein. Trojaner werden häufig über E-Mails verbreitet.

**Würmern** Programme, die sich selbstständig über das Netzwerk unter Ausnutzung von Sicherheitslücken verbreiten können, um dort eine Schadroutine auszuführen und sich dann weiter auszubreiten. Sie benötigen keine Nutzerinteraktion zur Ausführung und arbeiten häufig vom Nutzer unbemerkt.

**Spyware** Programme, welche Dateien des Hosts, Nutzer- und Netzwerkaktivitäten oder Zugangsdaten ausspionieren können. Spyware wird häufig von Trojanern auf dem Hostsystem installiert.

Malware kann mithilfe von Social Engineering Techniken[8] wie E-Mails, USB-Sticks, falsche Downloads oder ähnliche Techniken, oder aber auch durch Ausnutzung von Schwachstellen von außen über ein Netzwerk oder das Internet auf einem Hostsystem installiert werden.

## 1.2 Penetrationstest

Unternehmen, die ihre Dienste über das Internet zur Verfügung stellen und somit einen Teil ihrer Infrastruktur dem Internet aussetzen müssen, sind damit auch stets potenziellen Angreifern ausgesetzt. Durch die teils unüberschaubar vielen Konfigurationsmöglichkeiten ihrer Server und der eingesetzten Software und durch potenzielle Programmierfehler in der Software selbst kann kein Schutz garantiert werden. Es ist daher sinnvoll, das Konzept des Bug Bounty Programmes weiterzudenken und selbst aktiv zu werden. Dafür kann man sich die Methoden eines potenziellen Angreifers aneignen und regelmäßig selbst nach Möglichkeiten suchen, in das eigene System einzudringen und die dabei entdeckten Schwachstellen zu beheben, bevor es ein bösartiger Angreifer kann.

[8] Social Engineering bezeichnet die zwischenmenschliche Manipulation, die Hilfsbereitschaft, Vertrauen oder Autoritätshörigkeit ausnutzt, um an Daten zu gelangen, die das Opfer gar nicht herausgeben dürfte. Bekannt wurde dieses Verfahren durch Kevin Mitnick, der Social Eningeering als „die bei weitem erfolgreichste Methode, um an ein Passwort zu gelangen und schlage rein technische Ansätze um Längen“.(Mitnick, The Art of Deception)

Dieses Vorgehen bezeichnet man als Penetrationstest oder verkürzt Pentest [23, S. 13–21]. Neben der Prävention von Angriffen, dienen Penetrationstests auch der Bestandsaufnahme der Sicherheitslage für CIOs[9] oder CISOs[10], oder einer regelmäßigen Prüfung und Qualitätssicherung der Sicherheitsrichtlinien. Zusammengefasst erhöht dies die Datensicherheit des Unternehmens, senkt das Risiko eines Vorfalls mit eventuell hohen Verlusten für das Unternehmen und senkt damit auf Dauer Kosten [23, S. 7].

Pentests können verschiedenste Richtlinienchecks abdecken und mit Absprache des Auftraggebers verschiedene Angriffsvektoren nutzen, um sich Zugang zu den zu testenden Systemen zu verschaffen. Um sich die benötigten Informationen zu verschaffen, können je nach Art des vereinbarten Tests entweder in einem White-Box-Tests alle Daten an den Pentester übergeben werden, oder er benutzt bei einem Black-Box oder Grey-Box-Test technische Hilfsmittel wie Netzwerk- oder Schwachstellenscanner. Um tiefer in das Unternehmensnetzwerk blicken zu können, welches im Normalfall von außen schwer einzusehen ist, sollten solche Scans von innerhalb des Unternehmensnetzwerkes gestartet werden, da die Kommunikation hier bereits hinter einer abdichtenden Firewall stattfindet.

Einer Studie des BSI zufolge[8] besitzen KMUs[11] in Deutschland gute Sicherheitsprodukte und gut konzeptionierte Netzwerke, haben aber weniger Verständnis für die Notwendigkeit von IT-Sicherheit bzw. überschätzen den Grad der Sicherheit, den ihre bisherigen Ansätze und Systeme bieten. Sie haben sich auch wenig Gedanken gemacht, wie bei einem Vorfall vorgegangen werden soll, besitzen also kein gutes Notfallmanagementkonzept. Lediglich Backup-Strategien sind meist gut durchdacht und umgesetzt [8, S. 31].

Diese Arbeit stellt eine Methode samt technischem Hilfsmittel vor, mit dem sich Netzwerkscans und die Ausnutzung von Schwachstellen von innerhalb des Unternehmensnetzwerkes ausführen und ansatzweise automatisieren lässt, um die Arbeit von Pentestern deutlich zu erleichtern bzw. die Testergebnisse zu verbessern. Dies erfolgt unter der Annahme, dass auch bösartige Angreifer in der Lage sind, sich Zugang zu dem Netzwerk zu verschaffen und von innen heraus Schaden anzurichten. Dies wird der BSI-Studie zufolge dadurch begünstigt, dass KMUs zu sehr auf Schutzmechanismen vertrauen, die Angriffe von außen abwehren können. Auf diese Weise wird die Umsetzung von Sicherheitsrichtlinien getestet, die bei Angriffen von außen und von innen greifen sollten.

Vorgestellt wird eine Software, die in der Lage ist, Netzwerkscans durchzuführen, mögliche Schwachstellen mithilfe von nmap[15] aufzuspüren und entsprechende Exploits[12] direkt innerhalb des Netzwerkes zur Ausführung zu bringen. Zudem wird eine Methode aufgezeigt, mit der das Toolkit mithilfe von Social-Engineering in Verbindung mit mangelnden Sicherheitskonzepten eine Software in ein zu testendes Unternehmen geschleust werden kann. Beide Konzepte sind unabhängig voneinander und beliebig austauschbar, sodass andere Testprogramme genutzt oder auch andere Einschleusungsmechanismen zur Anwendung kommen können.

[9] Chief Information Officer
[10] Chief Information Security Officer
[11] Kleine bis mittelständige Unternehmen
[12] Programme, die auf die Ausnutzung bestimmter Schwachstellen ausgelegt sind

## 1.3 Aktuelle Bedrohungslage durch Trojaner

Im Jahr 2017 erlebte Ransomware, gemeinhin auch als *Cryptotrojaner* bezeichnet, ein neues Allzeithoch ihrer Verbreitung. Neben ihrer typischen Verbreitung über Spammails, gesendet über Botsysteme wie Necurs[27] [13], nutzen sie erstmals auch aktiv Sicherheitslücken von Windowssystemen aus[22], die in geheimen Dokumenten der NSA aufgeführt waren und veröffentlicht wurden.[35][10] Öffentlichkeitswirksam wurde dieses Vorgehen mit der schlagartigen Verbreitung des WannaCry-Trojaners, der hierzulande mit der Inbesitznahme von Anzeigetafeln der deutschen Bahn bekannt wurde[12]. Einem Bericht von *Malwarebytes Labs* zufolge, der die aktuelle Bedrohungslage genau analysiert, sind derzeit ca. 70% der verbreiteten Malware Ransomware.[21]

Einmal auf einem System, können sie massiven Schaden anrichten. Cryptotrojaner wie Cerber[2], WannaCry[5], Petya[4], NotPetya[1], Jaff[9] oder Locky[3] verschlüsseln alle Ihnen zugänglichen Dateien und stehlen sie somit dem Besitzer, der sie nur wiederherstellen kann, wenn er ein Lösegeld per BitCoin zahlt, oder ein entsprechendes Backup zur Hand hat. Gerade Unternehmen können enorme Kosten durch Produktionsausfälle entstehen, bis sie die Daten wiederherstellen können. Sind Backups bereits älter, droht sogar der Verlust einiger Daten. Hinzu kommen Vertrauenseinbußen bei aktuellen oder potenziellen Kunden, deren Schaden sich nur schwer beziffern lässt.

### 1.3.1 Funktionsweise von Trojanern

Ein Trojanisches Pferd (auch Trojaner genannt) ist ein Programm, dessen implementierte Ist-Funktionalität nicht mit der angegebenen Soll- Funktionalität übereinstimmt. Es erfüllt zwar diese Soll-Funktionalität, besitzt jedoch eine darüber hinausgehende, beabsichtigte zusätzliche, verborgene Funktionalität. [14]

Trojaner können über jeden Weg auf einen Computer gelangen, über den Daten transportiert werden. Dazu gehören Drive-By-Downloads, Installationsroutinen anderer Programme, das Auslesen von Datenträgern und ähnliche Methoden. Die derzeit am häufigsten verwendete Methode der Verbreitung sind Spam-E-Mails mit einem infizierten Anhang, der sich meist als Rechnung oder Anwaltsschreiben in einem Microsoft-Office-Format ausgibt. Bei WannaCry wurde erstmals eine Sicherheitslücke in einem Microsoft-Dienst (SMB1) ausgenutzt, um sich auch ohne die Interaktion mit einem Nutzer verbreiten zu können und zur Ausführung zu kommen. Petya wiederum verbreitete sich als schadhafter Drive-By-Download eines gefälschten Updates einer häufig in Unternehmen eingesetzten Buchhaltungssoftware. Cracker hatten sich Zugang zu dem Unternehmen hinter der Software verschafft und die Updatedateien ausgetauscht.[17] [14]. Zudem nutzt es Passwörter im Arbeitsspeicher des infizierten Systems, um sich auf weitere Rechner des Netzwerkes zu verbreiten. Dies entspricht einer für Würmer typischen Funktionalität.

Sind Trojaner auf einen Computer gelangt, wird ihre Schadroutine nicht automatisch ausgeführt, sondern muss durch die Interaktion des Nutzers aufgerufen werden. Im Falle

[13] `https://intel.malwaretech.com/botnet/necurs`

[14] `https://www.golem.de/news/ransomware-petya-kampagne-nutzt-luecke-in-buchhaltungssoftware-1706-128631.html`

von Petya war diese Aktivität die Ausführung des Programmupdates für die Buchhaltungssoftware. Da diese Software als vertrauenswürdig eingestuft ist, gab es hier aber kaum eine Möglichkeit, den Angriff vorher schon erkennen zu können. Trojaner können alle möglichen Schadroutinen aufweisen. Häufig installieren sie weitere Backdoorprogramme, die bestimmte Ports des Rechners öffnen, über die dauerhafte Fernzugänge auf den infizierten Rechner ermöglicht werden. Häufig wird ein solcher auch als Teil eines Botnetzes „versklavt“ und soll an DDoS-Angriffen[15], der Verteilung von Spam oder ähnlichen Aktionen teilnehmen. Ebenso verbreitet ist auch die Methode, den Rechner als Wanze einzusetzen. Es werden Keylogger oder Sniffer installiert und Berichte an einen zentralen Computer gesendet. Meist dient dieses Vorgehen der Vorbereitung weiterer, gezielterer Angriffe mit potenziell höherer Ausbeute für den Angreifer und höherem Schaden für das betroffene Unternehmen.

Ein weiterer Funktionsbaustein vieler Trojaner ist die Verschleierung ihrer Aktivität. Je nach Zweck des Trojaners werden Zeitstempel erstellter Dateien gefälscht, Log-Dateien gelöscht oder aber die Schadroutine wird in virtuellen Maschinen oder bei Vorhandensein potenzieller Analysesoftware wie Wireshark gar nicht erst ausgeführt. Im Falle von Droppern, also Trojanern, deren einziger Zweck die Installation weiterer Schadsoftware ist, zerstört sich der Trojaner häufig selbst, um so wenig Aufmerksamkeit, wie möglich auf sich zu ziehen.

### 1.3.2 Gefährdungslage für Unternehmensnetzwerke

Sind Unternehmen das Ziel von Angriffen, so werden hier meist gezieltere Angriffe gefahren. Zum einen ist der potenzielle Wert erbeuteter Informationen wesentlich höher als bei ziellosen Angriffen auf Privatpersonen, zum anderen sind die Schutzmaßnahmen der Unternehmen normalerweise wesentlich besser und damit schwieriger zu Umgehen, als bei normalen PC-Nutzern. Häufig werden hier also Social-Engineering-Angriffe durchgeführt. Angreifer informieren sich vor der Durchführung ihres Angriffes über das Opfer und suchen nach Adressen, Namen und Position von Angestellten, Prozessabläufen oder anderen internen Informationen. Ziel ist es, die Mitarbeiter des Unternehmens zu überzeugen, den Schadcode auszuführen. So wird Malware innerhalb des Unternehmens ausgeführt, ohne vorher mühsam entsprechende Sicherheitsmaßnahmen umgehen zu müssen.

Solche Social-Engineering-Attacken können in Form entsprechender Mails mit scheinbar unternehmensinternem Absender, liegengebliebener USB-Sticks oder telefonisch durchgeführt werden. Mitarbeiter werden früher oder später aufgefordert, präparierte ausführbare Dateien auszuführen oder durch Herausgabe von Informationen bei der Vorbereitung des Angriffes zu helfen.

Schäden für Unternehmen sind zumeist der Diebstahl oder gar die Löschung wichtiger (Geschäfts-)Daten, Erpressung oder ein Vertrauensverlust seitens der Kunden. In jedem Fall ist ein finanzieller Verlust zu erwarten, der zum wirtschaftlichen Scheitern des Unternehmens führen kann.

[15] Distributed Denial of Service

Nicht zu unterschätzen ist die Gefahr, dass böswillige Mitarbeiter mithilfe ihres umfangreichen Wissens dem Unternehmen Schaden zufügen können. Gründe hierfür können unter anderem Rache für eine Kündigung, eine Art Vergeltung für aus Sicht des Angreifers (z.B. moralisch) falsche Unternehmsstrategien oder schlicht Bereicherungswille sein. Hinzu kommt, dass sich viele Unternehmen durch den Kauf von Firewalls und Antivirenprodukten gegen Gefahren von außerhalb des Netzwerkes sichern, nicht oder nur unzureichend aber gegen Gefahren von innerhalb des Netzwerkes. So besitzen Mitarbeiter unnötig hohe Rechte zum Abruf oder gar Schreiben von Dateien, Passwortrichtlinien sind nicht vorhanden, unzureichend oder werden nicht durchgesetzt, Konfigurationen von Servern sind nicht richtig eingestellt oder gar auf dem Herstellerstandard belassen und so weiter. Dies kann sich auch Schadsoftware zunutze machen, die hinter einer solchen Firewall zum Einsatz kommt. Diese kann wesentlich einfacher auf von außen unzugängliche Daten zugreifen oder Schwachstellen von Systemen ausnutzen, die von außen nicht sichtbar wären.

All dies kann im Schadensfall zum größten anzunehmenden Schaden des Unternehmens führen, dem Verlust nicht-öffentlicher Geschäftsdaten.

## 1.4 Sicherheit durch Penetration Testing

### 1.4.1 Ziele des Penetration Testing

Pentetration Tests oder kurz *Pentests* sollen dem entgegen wirken und können einen entscheidenden Beitrag zur Sicherheit leisten. Durch verschiedene Testverfahren können all die oben beschriebenen Probleme und mehr entdeckt und gelöst werden.

Das Ziel ist es, die Infrastruktur aus Sicht eines realen Angreifers zu betrachten und sich mit deren Methoden Zugang zum System zu verschaffen. Auf diese Weise sollen Schwachstellen gefunden, analysiert und behoben werden, bevor ein böswilliger Angriff darauf stattfinden kann. Auch Vulnerability-Scanner versprechen das Auffinden von Schwachstellen und geben häufig auch eine detaillierte Beschreibung der gefundenen Probleme an. Dabei sind sie einfach in der Handhabung und wesentlich günstiger, als ein Penetration Test. Der Vorteil eines ausgebildeten Pentesters ist aber die Qualität der Analyse[23]. Ihm stehen weitreichendere Methoden und fundiertes Fachwissen zur Verfügung, um alle möglichen Lücken zu verifizieren. Dies sorgt für eine gegen null tendierende False-Positive-Rate. False-Positives sind bekannte Probleme von Vulnerability-Scannern, die oftmals lediglich Programmversionen auf bekannte Schwachstellen prüfen, diese aber nicht überprüfen können. So können vom Scanner unbemerkt Gegenmaßnahmen getroffen worden sein, die eine Ausnutzung dieser Schwachstelle bereits unmöglich machen. Auch können gemeldete Konfigurationsfehler in bestimmten Kombinationen unproblematisch sein, die dennoch als Schwachstelle erkannt werden. So entstehen teils lange Listen von Schwachstellen, die eigentlich nicht existieren und einen hohen Arbeitsaufwand zur Prüfung und Behebung nach sich ziehen.

Pentester können zudem nicht automatisierbare Testmethoden oder gar Kombinationen daraus anwenden, die ganz andere Schwachstellen aufdecken. Ein Mehrwert bieten auch die von professionellen Pentestern erstellten Analysen, die eine bessere Einschätzung des Gefahrenpotentials bieten und die Behebung der Schwachstellen erheblich vereinfachen. Oftmals werden Vulnerability-Scanner dennoch als Hilfsmittel von Pentestern eingesetzt.

### 1.4.2 Arten von Penetration Tests

Pentests können die Systemsicherheit aus der Sichtweise eines realen Angreifers bewerten. So entsteht ein realistisches Bild von den Möglichkeiten, die ein Angreifer zur Verfügung hat, um Daten zu stehlen oder zu manipulieren.

Es kann jedoch auch von Vorteil sein, als Pentester andere Sichtweisen einzunehmen. So können Angriffe nicht nur von außen, sondern beispielsweise bei Insider-Angriffen auch von innerhalb des Systems mit einem gewissen Vorwissen über die Systeme und deren Schwachstellen stattfinden. Auch dies muss bei Penetration Tests berücksichtigt werden können.

Aus diesem Grund haben sich verschiedene Testtypen entwickelt, die Angriffe basierend auf unterschiedlichem Vorwissen, unterschiedlichen Zugriffsrechten und unterschiedlichen Vorgehensweisen testen, um möglichst alle Arten von Angriffen und Angreifern realitätsnah zu simulieren und dabei auffindbare Sicherheitsmängel zu beheben.

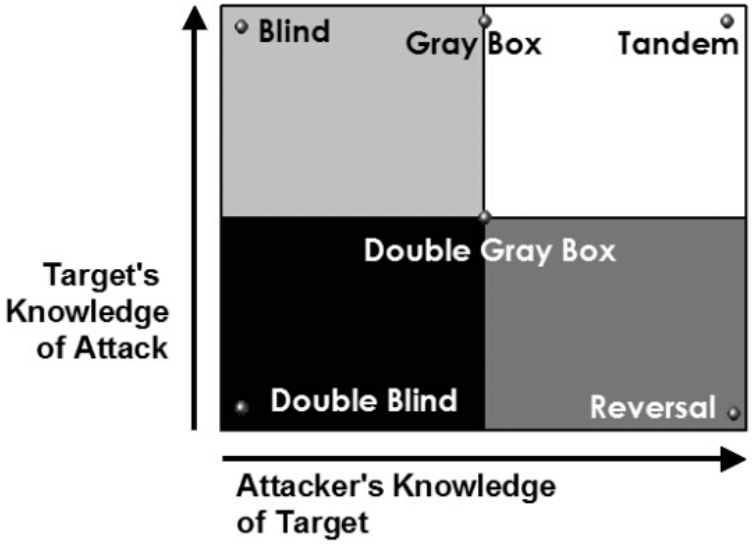


Abbildung 1: Die verschiedenen Arten von Pentests, basierend auf dem Vorwissen des Pentesters über das Unternehmen und dem Vorwissen des Unternehmens über die Vorgehensweise des Pentesters.

Es gibt 6 verschiedene Testarten, die üblicherweise zum Einsatz kommen [23, S. 20–22]:

1. Blind
2. Double Blind
3. Gray Boy
4. Double Gray Box
5. Tandem
6. Reversal

Bei einem *Blind*-Test testet der Analyst das Ziel, ohne die Infrastruktur oder dessen Abwehrmechanismen zu kennen. Das Ziel weiß von dem Angriff in allen Einzelheiten und kann sich darauf einstellen, wodurch hier die Fähigkeiten und Erfahrungen des Testers in Bezug auf Angriffstechniken und Geschwindigkeit stark im Vordergrund stehen, da von ihnen das Testergebnis stark abhängt. Diese Vorgehensweise entspricht der von „Ethical Hackern“, also White Hats. Weiß hingegen auch das Ziel nicht von diesem Angriff, so spricht man von einem *Double Blind*-Test. Auch hier testet der Analyst wieder das Ziel, ohne die Infrastruktur oder dessen Abwehrmechanismen zu kennen. Da das Ziel nichts von dem Angriff und der zum Einsatz kommenden Methodik weiß, werden hier zusätzlich zu den Fähigkeiten des Angreifers die Abwehrfähigkeit des Testzieles geprüft.

Das getestete Unternehmen muss den Angriff detektieren und abwehren können. Blind- und Double Blind-Tests werden auch als „Black Box Tests“ bezeichnet, da der Angreifer keinerlei Vorwissen über die zu testende Infrastruktur hat.

Wird davon ausgegangen, dass der Angreifer über ein gewisses Vorwissen über die Infrastruktur und deren Abwehrmechanismen verfügt, so kommt ein *Gray Box*-Test zum Einsatz. Es wird ein Testszenario erstellt, in dem ein Angreifer bestimmte Informationen zur Verfügung hat. Sei es, dass er ein Insider[16] ist oder ein Angreifer, der bereits bestimmte Informationen erbeuten konnte, beispielsweise von einem gestohlenen Laptop. Ein solcher Test bietet oft realistischere und vielfältige Angriffsmöglichkeiten für den Pentester. Dieser kann sich anhand der ihm zur Verfügung stehenden Informationen eine klügere Angriffsstrategie überlegen, als bei einem Black Box Test. Das Testergebnis ist hier stark von den vorher vermittelten Informationen abhängig. Es können daher verschiedene Angriffsszenarien mit verschiedenen Vorinformationen erdacht werden. Kennt das Testziel den Testbereich und Testzeitraum des Angreifers, nicht jedoch seine Teststrategie, so spricht man von einem *Double Gray Box Test*.

Bei sogenannten *White Box Tests* bekommt der Angreifer je nach Testbereich umfassendes Wissen über die Infrastruktur, die Kommunikationskanäle, Abwehrmechanismen, Verfahrensweisen oder gar den Quellcode für eingesetzte Programme im Unternehmen zur Verfügung gestellt. Solche Tests können keine realen Angreifer widerspiegeln, sollen jedoch alle Möglichkeiten des Pentesters für einen Angriff ausreizen. Dieser kann ein umfassendes Verständnis über das Unternehmen erarbeiten und darauf basierende Testmodelle entwerfen. Solche Tests erfordern einen hohen Einarbeitungsaufwand und sind stark abhängig von der Zeitspanne des Tests und der damit einhergehenden Gründlichkeit der Pentester. Bei einem solchen Modell sind die Pentester üblicherweise in den Sicherheitsprozess des Unternehmens eingebunden, sodass sie dem Unternehmen für eine längere Zeit zur Verfügung stehen oder gar direkt für das Unternehmen arbeiten.

Werden die Mitarbeiter des Testzieles ebenso umfassend in den Test eingebunden, spricht man von einem *Tandem* Test. Bei einem solchen wird jedoch nicht die Abwehrfähigkeit gegen unbekannte beziehungsweise unvorhergesehene Angriffe getestet. Diese werden bei *Reversal Test* geprüft, bei denen der Analyst auf sein vollständiges Wissen zurückgreifen kann, das Testziel jedoch bezüglich des Testbereiches, des Zeitpunktes des Tests und der eingesetzten Methodik im Unklaren gelassen wird.

### 1.4.3 Ablauf eines Penetration Tests

Es gibt keine offiziellen Standards für Penetration Tests, jedoch einige Standardisierungsansätze. So erstellte das BSI Ende des Jahres 2003 ein Durchführungskonzept für Penetrationstests, welches noch immer relevant ist und von vielen als Standard anerkannt wird.[6] Dieses Konzept teilt den Ablauf eines solchen Tests in fünf Phasen ein:

1. Vorbereitung

[16] Jemand mit Wissen über die Interna des Unternehmens. Ein Insider kennt beispielsweise bestimmte Prozessabläufe, Verfahren, Gewohnheiten der Mitarbeiter oder gar Zugangsdaten zu bestimmten Systemen. Typischerweise sind Insider aktuelle oder ehemalige Mitarbeiter des Unternehmens.

2. Informationsbeschaffung und -auswertung
3. Bewertung und Risikoanalyse
4. Aktive Eindringversuche und Validierung
5. Abschlussanalyse

Demnach Bedarf es bei der *Vorbereitung* einer engen Absprache mit dem Auftraggeber in Form eines Initialworkshops.[29] In diesem werden die Ziele des Tests und der Testbereich, auch „Scope“ genannt, festgelegt. Es werden Ansprechpartner bestimmt und die Aggressivität des Tests vereinbart, die stark von dem Testziel und dem damit verbundenen Testszenario abhängig ist. Das Testziel kann beispielsweise die Einhaltung bestimmter Vorschriften und Compliance-Richtlinien sein, Testszenarien legen mögliche Vorgehensweisen für Eindringversuche fest. So kann beispielsweise getestet werden, was der Verlust eines Laptops mit wichtigen Unternehmensdaten darauf auslösen kann. Wichtig ist zudem die Bekanntmachung kritischer Systeme, die durch den Tests keinesfalls gestört werden dürfen bzw. deren Verfügbarkeit nicht beeinträchtigt werden darf.

Für die *Informationsbeschaffung* werden passive und aktive Verfahren eingesetzt. Ziel ist hier, so viele Informationen über die Interna des Unternehmens, der Mitarbeiter, Richtlinien, Prozessabläufe und Infrastruktur zu erfahren, wie möglich. Aus diesen werden dann mögliche Schwachstellen extrahiert und Angriffsvektoren gebildet. Passive Methoden bestehen meist aus Internetrecherche, Besuch der Webseite des Unternehmens und ähnlichen Mitteln. Für eine aktive Informationsbeschaffung werden technische Hilfsmittel wie Port- und Vulnerabilityscanner und andere Software beispielsweise zur Analyse von Webdiensten des Unternehmens eingesetzt.

Basierend auf diesen Informationen werden in der *Bewertungsphase* mögliche Schwachstellen analysiert und deren Schadenspotenzial bewertet. Unter Einbeziehung der Testkriterien und einer gegebenenfalls notwendigen Absprache mit den Verantwortlichen des betroffenen Bereiches werden weitere Testszenarien und Angriffsvektoren bestimmt. Hier ist zu beachten, dass darauf ausgeführte Angriffe die Verfügbarkeit der Systeme beeinträchtigen und das operative Tagesgeschäft des Unternehmens gefährdet sein könnte.

Sind alle Punkte geklärt, so werden gefundene Schwachstellen in der Phase der *Aktiven Eindringversuche* validiert. Dadurch wird überprüft, ob sich diese ausnutzen lassen, um weitere Informationen oder höhere Rechte zu erlangen. Gerade durch Vulnerabilityscanner gefundene Schwachstellen stellen sich oft als false-positives[18][32] heraus und werden bereits durch technische Maßnahmen abgewehrt oder sind aus anderen Gründen nicht für Angriffe geeignet. Dafür kommen typischerweise Exploit-Codes zum Einsatz, die bestimmte Fehlfunktionen auslösen und Informationen freilegen oder zusätzliche Rechte gewähren können. Dies ist ein kritischer Schritt, da sich das Verhalten der Systeme durch triggern dieser Fehlfunktionen selten vorher genau definieren lässt. Es besteht hier daher eine erhöhte Gefahr eines Systemausfalles. Ließen sich Schwachstellen zugunsten der Pentester ausnutzen, so kann mit den erlangten Informationen und Rechten erneut eine Informationsbeschaffungsphase notwendig werden. Dieses Vorgehen wird „Pivoting“ genannt.

Sind alle Tests abgeschlossen, so müssen die Informationen in der *Abschlussanalyse* übersichtlich zusammengefasst werden. Hierfür ist eine genaue Dokumentation in allen vorherigen Phasen notwendig. Der Abschlussbericht muss für die zuständigen Leute der Führungsebene verständlich sein. Dafür müssen alle Ergebnisse in nicht-technischer Form erklärt und das angewendete Verfahren beschrieben werden. Zusätzlich müssen aber auch alle technischen Details des Angriffes für die zuständigen Techniker enthalten sein. Zum Abschlussbericht gehört üblicherweise auch eine Präsentation der Ergebnisse vor dem Auftraggeber.

# 2 Pentesting Toolkit

Das hier vorgestellte Werkzeug „Pentesting Toolkit“ soll den Prozess ab der Informationsbeschaffung bis hin zur Abschlussanalyse für den Pentester vereinfachen. Es liefert Funktionalitäten zur Beschaffung von Informationen aus einem zu testenden Unternehmensnetzwerk heraus und sammelt diese an einem zentralen Ort.

Dafür verwendet es verschiedene Programme, die ein Gesamtsystem bilden, welches sich über die Grenzen der Unternehmensinfrastruktur des Testziels hinaus erstreckt. So sammelt es Daten, die nur innerhalb des Netzwerkes erreichbar sind und versendet sie aus den geschützten Bereich des Unternehmensnetzwerkes hinaus an einen von den Pentestern kontrollierten Server. Dabei nutzt es aus, dass viele Unternehmen ihre Infrastruktur mit oft kommerziellen Werkzeugen nach außen hin gut sichern kann, jedoch Konfigurationsprobleme oder ein unzureichendes Rechtemanagement aufweist, sodass Insider-Angriffe oder Schadsoftware, die trotzdem hinein gelangt, leichtes Spiel haben.

Das Pentesting-Toolkit enthält ein Programm, welches ein zu solcher Schadsoftware ähnliches Verhalten aufweist. Es vereint Funktionalitäten einer Wanze[17] und die einer Ratte[18]. Da der primäre Zweck das Sammeln von Informationen für spätere Angriffe ist, reden wir nachfofolgend von einer Wanze. Diese soll auf einem oder mehreren Hosts des Unternehmens ausgeführt werden. In dieser Arbeit werden Mittel und Wege aufgeführt, mit denen die Wanze mit gängigen Methoden des Social-Engineering in das Unternehmensnetzwerk des Testzieles eingeschleust werden kann, die auch gängige Praktik für die Verbreitung von Trojanern aus der Wildnis sind.

Die Informationen, die diese Wanze sammelt, werden über das Internet an einen von den Pentestern kontrollierten Command&Control-Server (CNC-Server) gesendet, der ebenfalls Teil des Pentesting Toolkits ist. Dieser Umstand macht eine umfangreiche Absicherung der sensiblen Unternehmensdaten erforderlich. Eine Sicherheitsbetrachtung dazu ist in Kapitel 2.6 zu finden.

Der CNC-Server stellt den Pentestern alle eingegangenen Informationen übersichtlich und sortiert über eine Weboberfläche zur Verfügung. Zusätzlich bietet er eine Möglichkeit, die Wanzen fernzusteuern und somit gezielt Daten abzufragen, basierend auf zuvor erbeuteten Informationen. Dieses Vorgehen ist für alle Testszenarien, außer White Box Tests notwendig. Dafür werden eine Reihe möglicher Befehle in der Weboberfläche für jede einzelne Wanze aufgelistet, die dann direkt von der Wanze auf dem jeweiligen Hostsystem ausgeführt werden können.

Diese Befehle reichen von verwaltenden Tätigkeiten wie die Deaktivierung oder ein Update des Programmes bis hin zu Netzwerkscans und Angriffstechniken zur Infizierung weiterer Computer im jeweiligen Netzwerk des Hostcomputers der Wanze. Diese Funktionalitäten heben sich dann jedoch deutlich von den Fähigkeiten einer Wanze ab und gleichen eher denen einer Ratte.

[17] Wanzen sind in der Lage, Informationen eines Hostsystems wie Tastatureingaben, gespeicherte Dateien oder Kommunikation auslesen und an ein beliebiges Ziel versenden kann.

[18] Eine Ratte (RAT, Remote Access Tool) kann einen Computer, oft unbemerkt vom Nutzer, fernsteuern und so aktiv nach bestimmten Informationen suchen, Konfigurationen verändern oder anderen Schaden auf dem System selbst oder auf von ihm aus erreichbaren Hosts anrichten.

Zur Verwaltung all dieser Informationen wird eine SQL-Datenbank eingebunden, die über die Kommunikationsschnittstelle des CNC-Servers mit den Informationen der Wanzen versorgt wird und in verschiedenen Tabellen abspeichert.

Diese Kommunikationsschnittstelle fungiert als eigenständiges Programm und arbeitet in der aktuellen Implementierung teilweise unabhängig von der Weboberfläche, mit der die Pentester interagieren. Sie ist, solange keine Störung vorliegt, für alle Wanzen zu jeder Zeit erreichbar. Es werden jedoch nicht nur Informationen gesammelt. Da dieses Programm in Kontakt mit allen Wanzen steht, werden darüber auch die Befehle an die Wanzen übermittelt, die die Pentester über die Weboberfläche eingegeben haben.

## 2.1 Abgrenzung zu Metasploit

Alle bisher integrierten Funktionen der Wanzen ließen sich auch mithilfe des Metasploit-Framework umsetzen.[19] Das Pentesting-Toolkit (PT) verfolgt aber einen anderen Ansatz zur Durchführung des Angriffes, als Metasploit. Wanzen sollen durch einfaches Social-Engineering möglichst tief in ein Unternehmen eingeschleust werden. Ohne irgendeine Art der Nutzerinteraktion melden sich Wanzen bei einem Command&Control-Server und bauen somit einen Tunnel zum Internet auf. Die Stärke des PT liegt dabei in seiner geringen Größe und einfachen Systemanforderungen, die eine Ausführung ohne Installation und ohne Privilegien erlauben. Voraussetzung ist ein installiertes .Net-Framework, welches auf Windows-Hosts vorinstalliert ist und zumeist auch bleibt, da auch viele andere Anwendungen darauf aufbauen. Das PT dient als Unterstützung zur Benutzung des Metasploit-Framework. Es liefert einfacher Informationen über das zu testende Netzwerk und kann diese über eine Weboberfläche übersichtlich darstellen. Zudem lassen sich viele Wanzen zugleich steuern, ohne dass sich der Nutzer um Shellsessions oder ähnliches kümmern muss. Der Server und alle Wanzen versuchen stets automatisch, eine Verbindung aufrecht zu erhalten. Auf diese Weise können schneller Informationen über das Netzwerk gesammelt werden, um weitere, gezielte Angriffe zu erlauben.

## 2.2 Allgemeines Vorgehen

Das in dieser Arbeit vorgestellte PT ist darauf ausgelegt, nach einer beliebigen Form der Einschleusung in das Netzwerk möglichst unauffällig die Umgebung des ausführenden Hosts zu analysieren. Dazu gehören allgemeine Informationen zum aktuellen System, den ausgeführten Diensten darauf und den erreichbaren Hosts im Netzwerk. Es generiert daraus einen Bericht und sendet ihn an den Command&Control-Server (CNC-Server) des Pentesting-Toolkits. Zusätzlich ist es in der Lage, Kommandozeilenprogramme wie *Nmap* herunterzuladen, auf dem System zu installieren und zu steuern. Damit kann eine erweiterte Netzwerkanalyse stattfinden, die vollständig über die Nutzeroberfläche des CNC-Servers gesteuert werden kann. Die Ergebnisse sind sofort nach Beendigung des Tests sichtbar und können vom Pentester ausgewertet werden. Basierend auf diesen Ergebnissen können dann weitere Tests oder eine Verbreitung der Software im Netzwerk angestoßen werden. Auf diese Weise kann der Pentester immer tiefer in das Netzwerk

[19] `https://www.metasploit.com/`

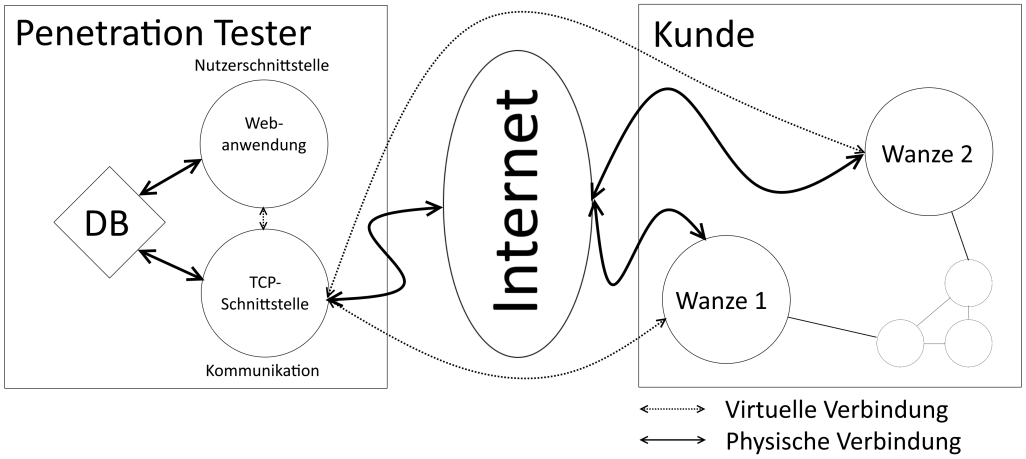


Abbildung 2: Struktureller Netzaufbau des Pentesting Toolkit

eindringen und Stück für Stück immer höherwertige Ziele analysieren, wie es das Pivoting vorsieht.

## 2.3 Aufbau

Der Aufbau des Programmes entspricht dem klassischen Client-Server Modell. Auf der Seite der Pentester steht der Command&Control-Server, welcher eine Datenbank zur Speicherung aller gesammelten Informationen, eine Webseite zur Steuerung der Wanzen sowie eine TCP-Schnittstelle zur Kommunikation mit den Wanzen ausführt. Datenbank, Weboberfläche und Kommunikationsschnittstelle können dabei auf drei unterschiedlichen Systemen ausgeführt werden, solange alle Programme Zugriff auf die selbe Instanz der Datenbank haben.

Auf der Seite der zu testenden Unternehmen werden die Wanzen ausgeführt. Diese versuchen, über eine festgelegte Adresse eine Verbindung zum CNC-Server herzustellen und Befehle entgegenzunehmen oder weiterzuvermitteln. Die Vernetzung des Unternehmens spielt dabei keine Rolle, jedoch braucht jede Wanze über ihr Hostsystem Zugang zum Internet. Dies könnte in späteren Versionen des Pentesting-Toolkit über ein Tunneling-System geändert werden. Dies betrachten wir im Ausblick in Kapitel 5.

## 2.4 Pentest-Client („Wanze“)

Die Wanzen vereinen Funktionen typischer Spyware, ergänzt um Mechanismen typischer RATs. Sie besitzen Methoden zum Sammeln von Informationen über das Hostsystem, den Anwender und das angeschlossene Netzwerk. Die Aktivitäten geschehen größtenteils vom Nutzer unbemerkt.

Wie für Malware typisch, besitzt es Methoden zur Verbreitung, Funktionen für den eigentlichen Angriff, sogenannte „Payloads“ und versucht, seine Aktivitäten weitestgehend

geheimzuhalten.

Die Verbreitung kann entweder trojaner-, oder wurmartig stattfinden.

### 2.4.1 Verbreitung

Wanzen des PT können auf zwei verschiedene Arten verbreitet werden. Bevor das Programm Zugang zu einem Netzwerk hat, ist es auf eine manuelle Installation angewiesen. Üblicherweise werden Trojaner per Social-Engineering Angriffen verteilt, da sie auf die Interaktion mit dem Nutzer angewiesen sind. Das heißt, dass sich das Programm als nützliches Werkzeug tarnt, welches der Nutzer gerne ausführen möchte, damit aber ebenso die Erlaubnis zur Installation des Trojaners erteilt. Eine andere Verbreitungsart wird von *Würmern* genutzt. Verbreitet sich Schadsoftware wurmartig weiter, so nutzt sie Sicherheitslücken in einem Netzwerkverbund aus, um die Installation auf anderen Systemen anzustoßen. Über ein Netzwerk kann es alle anderen Hosts, die von dieser Sicherheitslücke betroffen sind, befallen. Dort wird es ausgeführt und versucht wiederum, verbundene Systeme zu infizieren. Dieses Vorgehen ist nützlich, da hier eine sehr schnelle Verbreitung stattfinden kann. Die Effizienz hängt von der Vernetzung der Systeme untereinander, der Sicherheit der einzelnen Hosts und natürlich der Stärke des Wurms im Netzwerk ab. Das heißt, je mehr Möglichkeiten der Wurm hat, Systeme zu infizieren, desto weniger ist er von bestimmten Konfigurationen einzelner Systeme abhängig, die nicht in jedem Fall vorliegen müssen und desto schneller funktioniert die Verbreitung.

Diese Form der Verbreitung ist nicht von einer Nutzerinteraktion abhängig, setzt aber bekannte Sicherheitslücken im Netzwerk voraus. Zudem muss ein Einstiegspunkt in das Netzwerk gefunden werden. Einen solchen Einstiegspunkt eröffnet der USB Rubber Ducky.

**USB Rubber Ducky** Beispielhaft für diese Arbeit wird ein Social Engineering Angriff genutzt, der auf zusätzliche Hardware setzt. Zum Einschleusen des Programmes kommt der *USB Rubber Ducky*[20] zum Einsatz. Dies ist ein von *Hak5* entwickelter USB-Chip, der sich optisch nicht von einem normalen USB-Speicherstick unterscheiden lässt. Bei der Kopplung mit einem Computer verhält sich der Chip jedoch wie eine handelsübliche USB-Tastatur und ist aus Sicht des Betriebssystemes nicht von einem solchen zu unterscheiden. Da er selbst jedoch keine realen Tasten besitzt, kann der Chip selbstständig Nutzereingaben mit bis zu 1000 Wörtern pro Minute emulieren, wobei ein Wort ungefähr 5 Zeichen entspricht.[30] Diese werden vorher mit einer einfachen Programmiersprache („Ducky Script“) vorgegeben und in einem binären Format auf einer Micro-SD-Karte abgespeichert, die in den Stick eingeschoben werden kann. Sobald der Rubber Ducky an einem USB-Port angeschlossen wird, werden gegebenenfalls notwendige Treiber für USB-Tastaturen vom Betriebssystem installiert und der Stick beginnt sofort mit der vorgegebenen Eingabe.

Für diese Arbeit wird ein Script verwendet, welches eine versteckte Powershell unter Windows öffnen kann, durch die das Wanzenprogramm des Pentesting Toolkit her-

[20] https://www.hak5.org/gear/usb-rubber-ducky

untergeladen und ausgeführt wird. Die Wanze ist so einfach gehalten, dass sie keinen aufwändigen Installationsprozess bedarf. Voraussetzung ist jedoch ein installiertes .Net-Framework 3.0, welches auf den meisten Windows-Hosts jedoch bereits installiert ist. Der Rubber Ducky bedarf einem physischen Zugang zum System, welcher jedoch in den seltensten Fällen für wertvollere Systeme gegeben ist.

Daher gibt es noch eine andere Verbreitungsmethode, sofern sich bereits mindestens eine aktive Wanze in einem Unternehmensnetzwerk befindet.

**Verbreitung der Wanze mit Bordmitteln des Pentesting-Toolkit** Die andere Verbreitungsmethode ähnelt der eines Wurmes. Wie bereits in Kapitel 2.4.1 beschrieben wurde, bedarf es hierfür aber bekannter Sicherheitslücken. In einem Pentestszenario sind diese nicht gegeben, sondern sollen entdeckt werden. Das Pentesting Toolkit kann hier Unterstützung mittels Nmap leisten, welches von den Wanzen ausgeführt werden kann. Hiermit können Schwachstellen in umgebenden Systemen gefunden werden. Die erstellten Berichte sind über die Nutzeroberfläche des CNC-Servers einsehbar. Werden von den Pentestern nutzbare Sicherheitslücken entdeckt, so können sie mithilfe einer Metasploit-Framework-Instanz ausgenutzt werden. Die Pentester müssen hierfür also individuelle Exploits auswählen oder entwerfen. Die PT-Wanze kann leicht als Metasploit-Payload genutzt werden, wordurch sie unter Ausnutzung passender Sicherheitslücken und Befehle von Metasploit auf dem System installiert werden kann. Zudem soll es zukünftig möglich sein, den Netzwerkverkehr von Metasploit über die Wanzen im Netzwerk zu tunneln, um so auch von außen unzugängliche Hosts erreichen zu können. Auf diese Weise können die Wanzen selbst Exploits gegen benachbarte Hosts anwenden.

### 2.4.2 Payloads

Als Payloads werden in der Kommunikationstechnik allgemein Nutzdaten bezeichnet. Diese enthalten die reinen Informationen darüber, was Sender $A$ Empfänger $B$ mitteilen wollte. Analog dazu werden beim Pentesting damit die eigentlich beabsichtigten Befehle bezeichnet, die beispielsweise von einem Trojaner in das System eingeschleust und ausgeführt werden. Es ist also der eigentliche Schadcode, der dem Sammeln von vertraulichen Informationen, der Einrichtung eines permanenten Zuganges oder der Vorbereitung eines weiteren Angriffes dient.

Der Schadcode, der bei der Einschleusung der Wanze über den Ducky-Stick zum Einsatz kommt, führt eine Powershell aus, die die eigentliche Wanze herunterlädt und ausführt. Er ist in Ducky-Script verfasst und beschreibt die Tastatureingaben und die Wartezeiten zur nächsten Eingabe, die dem Betriebssystem vorgetäuscht werden. Wartebefehle stehen häufig nach der Eingabe der ENTER-Taste. Hier muss normalerweise auf die Verarbeitung des letzten Befehls durch das Betriebssystem gewartet werden.

```
WIN R
powershell -windowstyle hidden
ENTER
PAUSE 2000
$source = "https://shareserver.com/path/to/program";
$destination = "%TEMP%\BuggingDevice.exe";
Invoke-WebRequest $source -OutFile $destination;
start-process $destination;
exit;
ENTER
```

Abbildung 3: Der Payload des USB Rubber Ducky zum Download der PT-Wanze.

Dieser Code öffnet den windowseigenen „Ausführen“-Dialog mithilfe der Tastenkombination *WIN+R*.
Durch Eingabe des Befehls *powershell -windowstyle hidden*, bestätigt durch die *ENTER*-Taste, wird die Powershell ohne sichtbares Fenster geöffnet. Dafür wird dem Betriebssystem 2000 Millisekunden, also 2 Sekunden Zeit gelassen, bevor mit der nächsten Eingabe fortgefahren wird.
Es folgen 4 Zeilen Powershell-Code, der eine Datei von einem vorgegebenen Weblink herunterlädt und auf die Festplatte unter einem bestimmten Pfad schreibt. Ist dies geschehen, so soll die heruntergeladene Datei direkt ausgeführt und die Powershell-Sitzung beendet werden. Durch ein abschließendes *ENTER* wird die Befehlskette in Gang gesetzt.

Sobald das Wanzenprogramm ausgeführt wird, können weitere Payloads wie Nmap-Befehle oder andere Kommandozeilenprogramme, die unprivilegiert, also ohne erhöhte Rechte auf dem System funktionieren, ausgeführt werden. Wanzen des PT dienen somit hauptsächlich dem Sammeln von Informationen über das Netzwerk zur Vorbereitung eines Angriffes auf andere Unternehmensinfrastruktur. Es können jedoch auch beliebige andere Befehle ausgeführt werden. Ist es durch eine Sicherheitslücke möglich, erhöhte Rechte auf dem System zu gelangen, so können Befehle auch privilegiert ausgeführt werden.

Die Ausführung des Schadcodes ist abhängig von Modulen, die die Wanze vor dem ersten Aufruf installieren muss. Module können alle Funktionen erfüllen, deren Installation keinerlei Privilegien benötigt, oder aber durch Ausnutzung von Sicherheitslücken manuell durch den Pentester installiert werden können.

**Kommandozeile** Das Kommandozeilenmodul nutzt ausschließlich im .Net-Framework vorhandene Funktionalitäten, um beliebige Befehle über die Windows-Kommandozeile oder Powershell des Hostsystems der Wanze auszuführen. Dieses ist das einfachste, aber auch mächtigste Werkzeug. Befehle werden vor dem Nutzer versteckt ausgeführt, sofern durch sie keine Benutzeroberflächen geöffnet werden sollen. Pentester sollten darauf achten, nur Programme ohne Nutzeroberfläche zu starten oder diese zu unterdrücken. Es

können leicht einige Kommandos auf der Weboberfläche vorgegeben werden, sodass der Pentester Hilfestellung für die Ausführung der korrekten Kommandos mit den korrekten Optionen erhält.

**Nmap** Das Nmap-Modul erfordert mindestens den Download einiger Bibliotheken der offiziellen Nmap-Distribution, um deren Grundfunktionalitäten wie Host Discovery über die Konsole ausführen zu können. Diese Dateien werden vom CNC-Server zur Verfügung gestellt.

Nmap ist ein freies und quelloffenes Programm für Sicherheitsaudits und Netzwerkerkundung. Es dient dazu, andere Hosts im Netzwerk des ausführenden Rechners zu entdecken, identifizieren und Informationen zu möglichen Schwachstellen zu sammeln. Es ermöglicht umfangreiches Fingerprinting des Betriebssystems und darauf ausgeführter Software der entfernten Rechner. Dafür sendet es Netzwerkpakete an alle erreichbaren Hosts und wertet deren Antworten aus. Es kann auf allen gängigen Betriebssystemen ausgeführt werden und kommt größtenteils auch ohne privilegierten Systemzugriff aus.

Nmap wird auf den Zielrechnern als Kommandozeilentool installiert und kann über versteckte Kommandozeilenfenster für den Nutzer unsichtbar arbeiten. Der Pentester kann mithilfe des Pentesting-Toolkits nur indirekten Zugriff auf Nmap erhalten. Da keine direkt für den Pentester zugängliche Shell auf dem Zielrechner ausgeführt wird, werden einige vorgefertigte Befehle in den Pentestclient integriert, die über den Pentest-Server aufgerufen werden können. Es ist jedoch kein Problem, später weitere, individuelle Befehle hinzuzufügen. Dafür kann die Updatefunktion der Clients verwendet werden, die die neueste Programmversion des Pentestclients herunterlädt und ausführt.

Folgende Verwendungsszenarien von Nmap erscheinen für die Nutzung des Pentesting-Toolkits sinnvoll:

**Host Discovery** Die erste Aufgabe nach der erfolgreichen Übernahme eines Hosts ist, andere erreichbare Hosts des Netzwerkes aufzulisten. Hier wird geprüft, ob nun im Vergleich zu vorher neue Angriffsziele zur Verfügung stehen, die ihrerseits wieder für eine Installation und Ausführung von NMap infrage kommen.

Je nach Bedarf gibt es unterschiedliche Techniken, um Hosts im Netzwerk zu entdecken. Sie unterscheiden sich hauptsächlich in ihrer Zuverlässigkeit und Auffälligkeit. Ein einfacher *List scan (-sL)* ist in der Lage, Hosts aufzulisten, ohne tatsächlich Pakete zu ihnen zu senden. Zusätzlich wird eine Reverse DNS Resolution durchgeführt, um den Namen des Hosts zu bekommen. Dieser kann oft bereits Aufschluss über dessen Funktion liefern. Dadurch wird jedoch nicht geprüft, welche dieser Hosts gerade aktiv sind. [20, Kap. 15 - Host Discovery]

Ein *Ping scan / No port scan (-sn)* kann zusätzlich überprüfen, welche Hosts zum aktuellen Zeitpunkt aktiv sind. Dafür werden standardmäßig ICMP echo requests, sowie TCPSYN- an Port 443 und TCPACK-Pakete an Port 80 der jeweiligen Hosts gesendet. Diese Option ist intrusiver, da ein leicht erhöhter Netzwerkverkehr stattfindet, liefert aber wertvollere Informationen, als ein einfacher List scan. Dieses Vorgehen findet für jeden Host einzeln statt, es werden keine Broadcast-Pakete

gesendet, auf die viele Hosts auch nicht reagieren würden. [20, Kap. 15 - Host Discovery] Um Firewalls zu umgehen, sind erweiterte Optionen mit *-P** empfohlen. Diese werden nachfolgend beschrieben.

Mit *No ping (-Pn)* wird der gesamte Host Discovery Prozess übersprungen. Stattdessen werden alle im Netzwerk verfügbaren Adressen wie ein aktiver Host behandelt und stark intrusive Methoden wie Port Scan, OS- und Software Detection ausgeführt. Um auch den Port scan zu deaktivieren und ausschließlich eigene nmap Skripte (NSE) auszuführen, kann diese Option mit *-Sn* kombiniert werden. [20, Kap. 15 - Host Discovery]

*TCPSYN Ping (-PS<portlist>)* versucht, aktive Hosts mittels leerer TCPSYN-Pakete zu ermitteln. Werden keine speziellen Ports angegeben, so wird nur Port 80 geprüft. Ist der Host aktiv, so wird bei einem geschlossenen Port entweder ein RST- (Reset) oder bei einem offenen Port ein TCPACK-Paket (Acknowledge) gesendet, welches eine Bereitschaft zur Verbindung signalisiert. Im letzteren Fall wird der Verbindungsversuch durch Nmap sofort mit einem RST-Paket abgebrochen. Unabhängig von der Antwort wird der Host als aktiv gekennzeichnet. Dieser Vorgang kann auch als unprivilegierter User ausgeführt werden. [20, Kap. 15 - Host Discovery]

Ähnlich zu dem oben beschriebenen Vorgehen funktioniert ein *TCPACK Ping (-PA<portlist>)*. Statt TCPSYN- werden jedoch TCPACK-Pakete versendet, die normalerweise aufgrund einer nicht vorhandenen Verbindung sofort vom Hostsystem mittels eines TCPRST quittiert werden und sich somit als aktiv offenbaren. Der Sinn einer solchen Verbindung ist, schlecht konfigurierte, nicht statusbasierte Firewalls, die nur TCPSYN-Pakete blockieren, zu umgehen. Der Vorgang ist als unprivilegierter Nutzer nicht möglich und es findet ein Fallback auf TCPSYN ping statt. [20, Kap. 15 - Host Discovery]

Um die Wahrscheinlichkeit einer Umgehung moderner, statusbasierter Firewalls zu umgehen, zu maximieren, sollten sowohl SYN- als auch ACK-Pakete versendet werden, also beide Optionen *-PS -PA* aktiviert werden.

Zusätzlich zu den aufgeführten TCP-Tests können auch UDP-Pakete für ein *UDP ping (-PU<portlist>)* eingesetzt werden, um aktive Hosts zu erkennen. Hier ist jedoch zu beachten, dass Hosts anders auf UDP-Anfragen reagieren und sich die resultierenden Fehlermeldungen bei nicht vorhandenen Hosts ebenfalls von denen mittels TCP angefragten Hosts unterscheiden. Werden offene Ports mittels UDP getroffen, so reagieren Hosts häufig gar nicht. Bei geschlossenen Ports werden *ICMP Host unreachable* Pakete zurückgesendet. Andere ICMP-Fehler signalisieren eine nicht erreichbare Adresse. Aus diesem Grund ist es bei einem *UDP ping* wichtig, ungewöhnliche und somit höchstwahrscheinlich geschlossene Ports anzusprechen. Der Standardport dieser Operation lautet 40125. [20, Kap. 15 - Host Discovery]

**Weitere Nmap-Fähigkeiten** Nmap kann noch wesentlich komplexere Befehle ausführen. Dazu gehören Portscanning, Betriebssystem-, Diensteerkennung, Firewall und In-

trution Detection System Evasion und sogar einige Vulnerabilities können erkannt und ausgenutzt werden. Dies übersteigt jedoch die Komplexität der Musterimplementierung der Masterarbeit und wurde daher noch nicht integriert. Es können aber weitere Befehle definiert und einfach an Nmap weitergeleitet werden.

**Weitere Module** Es können beliebige Bibliotheken nachinstalliert werden, die einfache Aufrufe über die Windows-Kommandozeile unterstützen. Die Wanze kann ohne zusätzliche Konfiguration nur die Textergebnisse der Standardausgabe aller Kommandozeilenprogramme lesen und als Bericht an den CNC-Server weiterleiten.

Alle Module müssen derzeit noch manuell in alle Programmteile des Pentesting-Toolkit eingepflegt werden, um den Aufruf über die Weboberfläche, den Transport der korrekten Befehle mithilfe des Kommunikationsprotokolls und der Kommunikationsschnittstelle des Servers zu gewährleisten und schließlich die Verarbeitung durch die Wanze zu ermöglichen. Dies bedeutet für jedes einzelne Modul je nach Komplexität einen hohen Aufwand. Später soll das PT auf eine bessere Modularisierung aufbauen, um das Einpflegen neuer Funktionen stark zu vereinfachen. Siehe hierfür Kapitel 5.

### 2.4.3 Verschleierung

Wanzen nutzen bisher zumindest einfache Methoden zur Verschleierung ihrer Arbeit. Dieses Verhalten ist typisch für Trojaner, da ihr Handeln so lange wie möglich unentdeckt bleiben soll.

Je nach Einsatz der Wanze gibt es unterschiedliche Zielgruppen, vor denen ihr Handeln verschleiert werden soll:

**Endanwender** Hostsysteme von nichttechnischen Arbeitskräften müssen den Anschein wahren, dass alles wie gewohnt funktioniert. Es sollten keine zusätzlichen Fenster auftauchen, keine Fehlermeldungen generiert und keine Beeinträchtigungen des Systems verursacht werden, die zu einer erheblichen Verlangsamung oder gar Systemausfällen führen. Dies führt meist schnell zu einer Untersuchung durch geschultes, technisches Personal, das Schadsoftware schnell erkennen würde.

**Schutzsysteme auf dem Zielhost** Es ist davon auszugehen, dass alle Systeme innerhalb von Unternehmensnetzwerken über ein installiertes Antivirenprogramm verfügen. Dieses kann auch versteckte Prozesse kenntlich machen, die Schaden anrichten. Dazu nutzen sie entweder eine Mustererkennung, die Dateien gezielt auf enthaltene, bekannte Viren prüfen, oder aber heuristische Methoden zur Bewertung des Verhaltens ausgeführter Prozesse auf potenziell schädliche Aktivitäten.[11] Dazu gehört replizierendes Verhalten, also das Schreiben von Kopien des eigenen Programmcodes beispielsweise in andere Dateien, oder auch inversive Eingriffe in das System wie das Löschen von Dateien oder Verändern von Systemeinstellungen. Ebenso verdächtig sind Internetverbindungen zu bekannten vertrauensunwürdigen Servern und Netzwerken. Damit Malware unbemerkt agieren kann, muss sie sich verhalten, wie bekannte und vertrauenswürdige Programme.

**IT-Administratoren/Intrusion Detection Systeme** Jedes größere Unternehmen, mindestens aber jedes Unternehmen, welches sensible Daten verarbeitet, verfügt über fachkundiges Personal in der IT-Abteilung. Es ist davon auszugehen, dass diese Systeme einsetzen, die offensichtlich abnormales Verhalten innerhalb des Unternehmensnetzwerkes erkennen und melden. Das IT-Personal kann Fehler in Logfiles aufspüren, die Firewalls, Netzwerkrouter, zentral eingesetzte Software oder ähnliche Systeme erstellen, die durch die Malware beeinflusst werden.

Ohne genaue Kenntnis der Verfahren des Unternehmens ist kaum vorherzusehen, welche Analysemethoden zum Einsatz kommen, wie regulativ eingesetzte Sicherheitssoftware konfiguriert ist und wie sensibel Mitarbeiter auf anormales Verhalten reagieren. Es ist nicht möglich, eine Software zu schreiben, die alle möglichen Entdeckungsrisiken umgehen kann. Das Pentesting Toolkit kann daher nur versuchen, Aufgaben mit geringstmöglichem Aufwand zu bearbeiten und die Ressourcennutzung des Unternehmens damit so klein wie möglich zu halten.

Folgende Punkte wurden bei der Entwicklung besonders berücksichtigt, um eine schnelle Erkennung des PT zu verhindern:

1. Die Infektion des Hosts erscheint durch das Ducky Toolkit aus Sicht möglicher Antivirensoftware als legitime Nutzeraktion.

2. Die Ausführung des Programmes bedarf keiner gesonderten Installation und ist auf den meisten Windows Hostsystemen lauffähig. Dies bedeutet allerdings, dass das Programm zunächst keinen Neustart des Systems überstehen kann, ohne dass der Nutzer darüber gewarnt werden könnte.

3. Das Programm zeigt dem Nutzer keine sichtbare Programmoberfläche, Warnhinweise oder ähnliche Anhaltspunkte der Aktivität.

4. Das Programm läuft ressourcenschonend. Es werden für alle Aktivitäten wenig Ressourcen beansprucht. Die Zeiten für Boot und Shutdown des Systems werden nicht signifikant beeinträchtigt, solange der Pentester keine Befehle auf dem Host ausführen lässt, die Gegenteiliges bewirken. Die Aggressivität von Standardbefehlen wird für den Anwender deutlich gekennzeichnet.

5. Es ist eine Verschlüsselung des Netzwerkverkehrs vorgesehen.

6. Es werden TCP-Verbindungen genutzt, die einigen IT-Administratoren vertrauenswürdiger erscheinen, als Kommunikation mittels UDP[21].

## 2.5 Pentest-Server

Der Server des Pentesting-Toolkit (CNC-Server) dient als zentrale Steuereinheit und bildet damit das Herzstück des Programmes. Alle Befehle, gesammelten Informationen und Ereignisse gehen von ihm aus oder werden an ihn weitergeleitet.

[21] User Datagram Protocol

Der CNC-Server ist angebunden an eine Datenbank zur Verwaltung aller aktiven Wanzen sowie deren gesammelten Informationen, wie Logs und Scanergebnissen. Zudem verfügt er über eine einfache Schnittstelle zu einem externen Programm, welches die Kommunikation zu allen Wanzen aufrecht erhält. Zur Visualisierung und Steuerung sämtlicher Abläufe stellt er eine Weboberfläche zur Verfügung.

### 2.5.1 Benutzeroberfläche

Über die Weboberfläche können authentifizierte Nutzer sämtliche Informationen über die aktiven Wanzen einsehen. Diese werden übersichtlich in einer Tabelle dargestellt.

Zu diesen Informationen gehören:

- Name (falls einer festgelegt wurde)
- Hostname
- IP-Adresse
- Zeitpunkt des letzten Kontaktes
- Aktuelle Verfügbarkeit
- Zugang zu Detailinformationen zu Scanberichten und ähnlichem

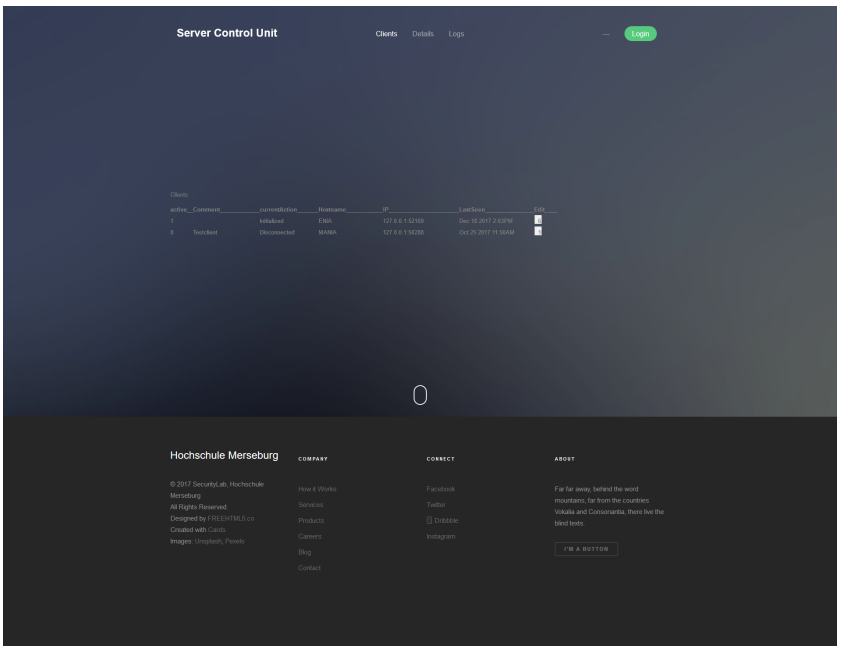


Abbildung 4: Die Landingpage für autorisierte Nutzer mit Übersicht der aktuell 2 bekannten Wanzen.

Auf der Detailseite, die für jede verfügbare Wanze verfügbar ist, können zudem Befehle direkt an diesen speziellen Host gesendet werden. Dies ist natürlich nur möglich, wenn eine Verbindung zu dieser Wanze hergestellt werden kann. Die TCP-Schnittstelle zu den Wanzen kann diesen Befehl jedoch speichern und absenden, sobald die Verbindung wieder hergestellt werden kann, auch wenn der Nutzer zu diesem Zeitpunkt nicht mehr auf der Weboberfläche angemeldet wurde. Dafür gibt es eine extra Tabelle in der Datenbank, die solche Aufträge speichern kann (siehe Dokumentation der Datenbank in Kapitel 3.2).

Derzeit werden nur wenige Befehle unterstützt, das Programm kann jedoch einfach um mehr Funktionalitäten erweitert werden. Siehe dazu die Dokumentation zur Implementierung zusätzlicher Payloads in Kapitel **??**, wie die Befehle in der Kommunikation, im Client und der Server-Applikation implementiert werden müssen.

Die derzeitig unterstützten Befehle sind:

**Ping** Ein Ping sendet ein kleines Paket an den Host, welches ein Heartbeat-Paket anfordert. Dieses enthält keinerlei Daten und dient nur dazu, die Verbindung und Reaktionsfähigkeit der Wanze prüfen zu können. Pings werden automatisch alle 60 Sekunden gesendet, können aber auch manuell angestoßen werden.

**Schlafmodus** Dieser Befehl unterbricht alle laufenden Prozesse, die auf dem Host durch die Wanze ausgeführt werden. Lediglich die zur Erhaltung der Kommunikation notwendigen Prozesse werden weiterhin ausgeführt.

**Deaktivierung** Eine Deaktivierung der Wanze führt dazu, dass alle Aktivitäten der Wanze eingestellt werden. Sie ist dadurch nicht mehr erreichbar und wird keinerlei Prozesse mehr ausführen können. Dieser Befehl entfernt zusätzlich alle durch die Wanze angelegten Dateien und kann nicht mehr rückgängig gemacht werden.

**Nmap-Befehle** Es gibt eine Reihe von Befehlen, die Nmap auf verschiedene Arten aufrufen. Diese Befehle sind abhängig von Nmap selbst und werden von der Wanze an die Kommandozeile von Nmap übergeben. Neben einer Reihe von vordefinierten Befehlen, die direkt in CNC-Server und Wanze eingepflegt wurden, können auch manuell Kommandos als Argumente an die Befehlszeile übergeben werden. Zu den eingebauten Befehlen gehören verschiedene Methoden zur Erkundung von Hosts („Host-Discovery“) und Portscans. Es sollte für den Nutzer jeweils dargestellt werden, wie viel Aufmerksamkeit diese Befehle erregen könnten, wobei eine solche Anzeige eher als Richtwert zu verstehen wäre. Die Liste an Befehlen ist erweiterbar mit allen Befehlen, die Nmap unterstützt. Es kann jedoch sein, dass der Host der Wanze nicht in der Lage ist, alle Befehle (zum Beispiel aufgrund fehlender Treiber oder sonstiger Restriktionen) auszuführen.

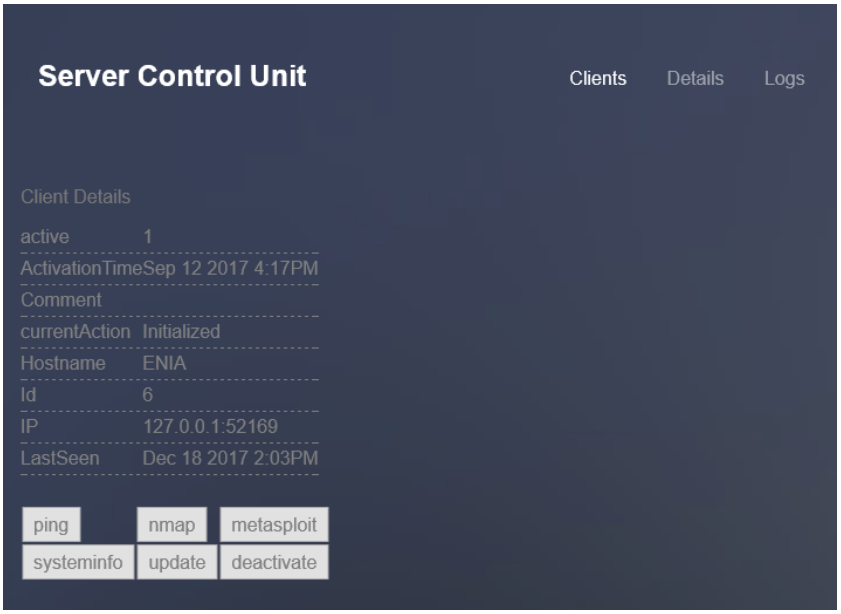


Abbildung 5: Die Detailansicht einer Wanze mit den zugehörigen, unterstützten Befehlen.

Zusätzlich zu den aufgeführten Befehlen sind für den Nutzer unzugängliche Befehle interpretiert, die der Server ausschließlich bei Bedarf sendet. Dazu gehört beispielsweise die Update-Funktion, die eine Wanze zur Installation einer neuen Version zwingt, bevor weitere Kommunikation mit dem Server stattfinden kann. Auch Befehle, die für einen geregelten Kommunikationsablauf notwendig sind, können nicht manuell vom Nutzer angestoßen werden. Dies würde die Anwendung unnötig verkomplizieren und bringe keinerlei Vorteile.

Schließlich gibt es noch eine Unterseite mit allen gesammelten Berichten und Informationen der Wanzen. Diese ist über den Reiter „Logs“ erreichbar und listet wieder eine Tabelle mit allen wichtigen Details der Logs.

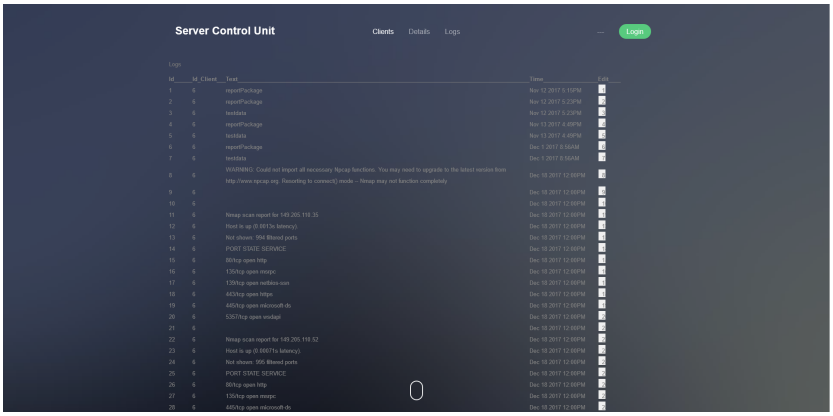


Abbildung 6: Übersichtsseite aller bisher eingegangenen Berichte jeder einzelnen Wanze.

### 2.5.2 Steuerungsschnittstelle

**Kommunikation zwischen Wanzen und CNC-Server** Die Kommunikation zwischen den Wanzen und dem CNC-Server basiert auf einem einfachen TCP-Protokoll. UDP wird aktuell (noch) nicht unterstützt, da einige Unternehmen UDP-Pakete aufgrund seines schlechten Images generell sperren. UDP kommt für Peer-to-Peer (P2P) Netze zum Einsatz und wird üblicherweise für VoIP und Filesharing eingesetzt. Durch solcherlei Anwendungen ist es in den Verruf eines unsicheren Protokolls geraten. Der Netzwerkaufbau des Pentesting-Toolkits setzt auf ein Client-Server-Modell, wobei der Server direkt mit dem Internet verbunden ist und eine feste Adresse hat. TCP ist hier als Protokoll daher angemessen.

Der Ablauf der Kommunikation folgt einem festen Schema und geht stets von den Wanzen aus. Es ist davon auszugehen, dass Unternehmensnetzwerke durch eine Firewall geschützt sind, die unbekannte Verbindungen von außen blockieren. Nur Hosts im Netzwerk, die Zugang zum Internet haben, können Pakete nach außen schicken. Die Firewall speichert daraufhin die Verbindungsinformationen und lässt erst dann Pakete von außen herein, die als Antwort auf die gespeicherte Anfrage zu verstehen sind. Dafür müssen die IP-Adresse, Portnummer und im Falle von TCP auf Sequenznummern der Pakete stimmen[22]. Dieses Verfahren wird „Stateful Packet Inspection“ genannt und ist seit der Firewall-1 von CheckPoint Software Technologies Ltd. aus dem Jahre 1994[31] gängiger Standard.

Es ist für den CNC-Server daher nicht möglich, eine Verbindung zu den Wanzen innnerhalb eines durch eine solche Firewall geschützten Netzwerkes aufzubauen. Wanzen können jedoch durch die Firewall hindurch eine Verbindung zum Server herstellen, der daraufhin über diese Verbindung auch Daten an die Wanzen versenden kann.

Die Kommunikation entspricht immer dem folgenden Schema:

**1. Verbindungsaufbau** Sobald eine Wanze aktiv wird, schickt sie ein TCP-Paket mit ih-

[22] `https://www.checkpoint.com/smb/help/utm1/8.2/7081.htm`

rem Hostnamen, Versionsinformationen der Wanze und einem Authentifizierungsschlüssel.

**2. Prüfen der Verbindungsdaten** Der Server prüft daraufhin alle Daten auf Korrektheit sowie die Versionsinformationen. Entspricht diese nicht dem neuesten Stand, so wird ein Update erzwungen.

**2.1 Update der Wanze** Die Wanze startet erneut eine Powershell und installiert sich auf die selbe Weise mit neuen Programmdateien. Die Kommunikation beginnt nach dem Update wieder beim ersten Schritt.

**2.2 Bestätigung der Verbindung** Sind alle Daten korrekt, so sendet der CNC-Server eine einfache Bestätigung des Verbindungsaufbaus an die Wanze.

**2.3 Ablehnung der Verbindung** Sind nicht alle Daten korrekt, so sendet der Server ein entsprechendes Paket. Dies deaktiviert die Wanze.

**3. Stehende Verbindung** Es werden keine weiteren Daten gesendet, bis der Nutzer eine Aktion über die Weboberfläche auslöst. Die Verbindung wird offen gehalten.

**3.1 Herzschlag** Solange eine Verbindung besteht, senden sowohl Wanzen als auch Server sogenannte Heartbeat-Pakete. Diese signalisieren eine weiterhin bestehende Bereitschaft zur Interaktion und dienen zusätzlich zum Testen der Verbindung.

**4. Aufgabe** Löst der Nutzer eine Aufgabe aus, so wird diese zusammen mit Details zur Aufgabe an die Wanze gesendet.

**4.1 Bestätigung** Die Wanze bestätigt die begonnene Ausführung der Aufgabe.

**4.2 Ablehnung** Kann die Wanze den Befehl nicht interpretieren oder ist es ihr aus anderen Gründen nicht möglich, diese Aufgabe auszuführen, so teilt sie dies dem Server mit.

**5. Bearbeitung/Fertigstellung der Aufgabe**

**5.1 Bericht** Ist die Aufgabe erfüllt, so werden die Ergebnisse in Form eines Berichtes an den Server gesendet. Die Kommunikation wird an Punkt 3 fortgeführt.

**5.2 Abbruch der Aufgabe** Kann die Aufgabe aufgrund eines kritischen Fehlers nicht fortgesetzt werden, so wird dies dem CNC-Server mit einer Fehlerbeschreibung mitgeteilt.

**Datenbankschnittstelle** Der CNC-Server verfügt über eine MSSQL-Datenbank, in der sich Details aller bekannten Wanzen, alle Scanberichte und gesendeten Aufgaben befinden. Zugriff auf diese Datenbank haben der PHP-Webserver, sowie das C#-Programm, welches die dauerhafte TCP-Schnittstelle zu den Wanzen darstellt.

Die TCP-Schnittstelle nimmt dabei alle Daten der Wanzen entgegen und schreibt sie in die Datenbank, während die Weboberfläche vorrangig lesenden Zugriff auf die

Datenbank hat, um alle Informationen darzustellen. Eine Ausnahme bildet die Aufgabentabelle. Da der Nutzer Aufgaben über die Weboberfläche erstellt, wird diese Tabelle auch über diese mit Daten gespeist. Gesendet werden die Aufgaben jedoch von der C#-TCP-Schnittstelle, weshalb diese in periodischen Abständen neue Aufgaben aus dieser Tabelle abfragt.

Eine genaue Auflistung aller Datenbankoperationen ist der Dokumentation zur Datenbank in Kapitel 3.2 zu entnehmen.

## 2.6 Sicherheitsaspekte

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) beschreibt in seinem IT-Grundschutzkatalog für Webanwendungen[7] einige Sicherungsmaßnahmen, um einen nach aktuellen Standards als sicher bewerteten Betrieb einer solchen Webanwendung zu gewährleisten.

Ziel ist der sichere Betrieb der Webanwendung, der Hardware und die Wahrung der Vertraulichkeit verarbeiteter Daten des Kunden.

Für die Umsetzung dieses Sicherheitskonzeptes ist eine Aufstellung der Gefährdungslage, der Anforderungen an die Webanwendung und die Umsetzung zugehöriger Schutzmaßnahmen erforderlich. Dafür werden die vom BSI aufgeführten Punkte auf das Pentesting-Toolkit adaptiert.

### 2.6.1 Gefährdungslage

#### Allgemeine Bedrohungen

1. Die Webanwendung verarbeitet streng vertrauliche Daten eines Unternehmens, die es durch verschiedene technische Hilfsmittel und Verfahren erlangt. Diese werden von Clientprogrammen aus dem Bereich des Unternehmensnetzwerkes heraus über ein öffentliches Übertragungsmedium an die Webapplikation gesendet. Offenbar besteht hier die Gefahr der **Verletzung der Vertraulichkeit** der Daten.

2. Die Webanwendung erlaubt Zugriff auf Datenverarbeitungsgeräte, die nicht zur Anwendung gehören. Durch diesen Zugriff ist eine Manipulation dieser Geräte, daran angeschlossener Geräte und auf Daten möglich, die auf diesen Geräten gespeichert oder verwaltet werden. Dieser Zugriff ist abhängig von den Sicherheitsrichtlinien des Unternehmens, deren korrekte Umsetzung jedoch nicht gewährleistet ist. Hier besteht eine Gefahr der **Verletzung der Integrität von Daten und Geräten** des Unternehmens bzw. seiner Kunden.

3. Durch die Manipulationsmöglichkeiten von Unternehmensgeräten besteht eine Gefährdung der **Verfügbarkeit von Systemen**. Unter Umständen erhält die Webanwendung Zugriff auf Systeme, die relevant für das Alltagsgeschäft des Kunden sind. Eine Verletzung der Verfügbarkeit solcher Systeme könnte existenzbedrohende Folgen haben.

**Spezifische Bedrohungen** Unter den spezifischen Bedrohungen sind Angriffspunkte in der Webanwendung zu verstehen, die zu den oben genannten Problemen führen können.

1. **Mängel in der Programmierung** Durch Fehler in der Programmierung können Angreifer oder auch normale Anwender Fehler auslösen oder gezielt provozieren, die ihnen Zugriff auf die Webanwendung und damit auf die geschützten Systeme und Informationen verschaffen.

2. **Umgehen der Autorisierung** Alle vertraulichen Daten sind nur einem kleinen, vertrauenswürdigen Personenkreis zugänglich. Die Webanwendung muss sicherstellen, dass nur der spezifizierte Personenkreis Zugang zu den Daten erhält. Dafür müssen diese Personen ein Authentifizierungsverfahren durchlaufen, um ihre Vertrauenswürdigkeit nachweisen zu können. Dies muss so gestaltet sein, dass es unmöglich ist, Zugriff auf den geschützten Bereich zu bekommen, ohne die notwendigen Rechte zu besitzen bzw. ohne dieses Authentifizierungsverfahren durchlaufen zu haben.

3. **Fehlerhafte Ein- und Ausgabedaten** Durch fehlerhafte Datenverarbeitungsmechanismen können unerwartete Konsequenzen für die Zuverlässigkeit des Programmes entstehen. Durch Code-Injection-Angriffe, die mithilfe unerwarteter Ein- und Ausgabedaten durchgeführt werden, können fremde Befehle zur Ausführung kommen und alle Schutzziele erheblich gefährden. Alle Nutzereingaben und alle automatisch erzeugten Befehlsdaten, die gelesen, gespeichert, oder ausgeführt werden, bedürfen einer sorgfältigen Prüfung, um solche Angriffe zu verhindern. Besonders die Datenbank und alle Argumentdaten für durch Wanzen auszuführende Befehle sind gefährdet. Beachtet werden muss auch, dass solche Daten auch nach der Erzeugung, zum Beispiel während der Übertragung manipuliert werden könnten.

4. **Unzureichende Fehlerbehandlung** Die Anwendung ist in erheblichem Maße abhängig von externen Ressourcen, wie einer stabilen Internetverbindung, Hostsysteme, deren Verfügbarkeitsgrad unbekannt ist, der Datenbank, Nutzereingaben und vielem mehr. Treten unerwartete Ereignisse wie eine fehlerhafte Eingabe, plötzliche Unerreichbarkeit der Ressourcen oder ähnliche auf, so darf dies nicht zu undefinierten Systemzuständen führen. Das Programm muss für jeden anzunehmenden Vorfall eine Fehlerbehandlungsroutine aufweisen. Undefinierte Systemzustände führen zwangsläufig zum Absturz der Anwendung und damit zu einer Einschränkung der Verfügbarkeit. Zudem können Vertraulichkeit und Integrität der Daten beeinträchtigt werden.

5. **Unzureichende Protokollierung** Sicherheitsrelevante Fehler und Ereignisse müssen von allen Anwendungen protokkolliert werden, sodass Ursachen zu einem späteren Zeitpunkt noch nachvollziehbar sind. Dies stellt die Möglichkeit zu einer schnellen Fehlerbehebung sicher.

6. **Missbrauch durch automatisierte Nutzung** Es muss sichergestellt werden, dass kein Teilprogramm des PT automatisiert genutzt und zweckentfremdet werden kann.

Durch eine automatisierte Nutzung können Vorgänge in kürzester Zeit sehr oft wiederholt werden und auf diese Weise auf Wiederholung basierende Angriffe durchgeführt werden. Beispiele hierfür sind Brute-Force-Angriffe zum Brechen von Passwörtern oder einer Übertragungsverschlüsselung und Denial of Service Angriffe.

**7. Unzureichendes Session-Management** Sind Methoden des Sessionmanagements nicht korrekt umgesetzt, so könnte ein Angreifer in den Besitz der Session eines autorisierten Nutzers in der Webanwendung gelangen. Dieser hätte dadurch Zugriff auf alle vertraulichen Informationen der durch das Toolkit getesteten Unternehmen.

### 2.6.2 Anforderungen

Die Anforderungen legen fest, welche Gegenmaßnahmen für die oben beschriebenen Bedrohungen getroffen werden müssen und ob speziellen Eigenschaften in der Umsetzung gegeben sein müssen.

**A1.1 - Authentisierung bei Webanwendungen** Der Zugriff auf geschützte Bereiche darf nur durch authentisierte Personen erfolgen. Hierfür muss ein Nutzer manuell freigeschaltet werden, der sich dann mit einer Kombination aus Benutzernamen und Passwort authentisieren kann. Derzeit scheint ein gesondertes Rechtemanagement nicht erforderlich, da die Webanwendung nur einem sehr kleinen Personenkreis zugänglich sein wird. Alle authentisierten Personen haben daher Vollzugriff auf alle Programmfunktionen. Dafür wird eine zentrale Authentisierungskomponente verwendet und eine Passwortrichtlinie festgelegt. Das Passwort kann nicht durch die Anwendung gespeichert werden.

**A1.1.1 - Gesonderte Authentisierung** Besondere Programmfunktionen bedürfen einer erneuten Authentisierung, um sicherzustellen, dass eine Websession nicht gestohlen wurde. Besondere Programmfunktionen sind jene, die potenziell hohen Schaden für die Anwendung oder den Kunden bedeuten können. Dazu zählen:

- Abschalten von Wanzen
- Aufruf integrierter Angriffsroutinen gegen Unternehmenshosts

**A1.2 - Authentisierung bei Wanzen** Wanzen nehmen Befehle des CNC-Servers entgegen und können diese unter Umständen direkt in einer Kommandozeile des Hostsystems, welches innerhalb eines streng geschützten Bereiches liegt, zur Ausführung bringen. Hier muss strengstens sichergestellt werden, dass diese Befehle von einem authentifizierten Nutzer gesendet und auf dem Übertragungsweg nicht manipuliert wurden. Ersteres wird durch den Punkt *A1.1* sichergestellt. Um eine spätere Manipulation ausschließen zu können, muss eine ausreichend sichere Übertragungsverschlüsselung in Verbindung mit einem Zertifizierungsverfahren angewendet werden, sodass Wanzen sicherstellen können, dass der Befehl von dem ihnen bekannten CNC-Server gesendet wurde. Das dafür angewendete Zertifizierungsverfahren setzt vorraus, dass das Zertifikat

1. Niemandem zugänglich,
2. Jederzeit widerrufbar,
3. im Gefahrenfall jederzeit austauschbar ist.

Kann eine Wanze die Authentizität eines Programmbefehls nicht sicherstellen, so darf sie keinerlei Funktion auslösen. Eine Wanze darf keinem anderen Zertifikat als dem ihr bekannten Zertifikat des CNC-Servers vertrauen, eine Änderung des Zertifikats nur durch kompletten Austausch des Wanzenprogrammes erfolgen. Zur Sicherung des eingepflegten Zertifikates der Wanze darf daher auch ein Update des Wanzenprogrammes nur durch den ihr bekannten, authentisierten CNC-Server erfolgen. Für diesen Austausch muss ein anderes Zertifikat genutzt werden, als das für den regulären Betrieb notwendige. Für höchste Sicherheit darf dieses auch nicht durch die normalen Serveranwendungen zugänglich sein, um den Diebstahl dieses Zertifikates zu erschweren.

**A2 - Sichere Datenübertragung** Sämtliche Kommunikationswege sind als nicht vertrauenswürdig anzusehen. Werden Daten auf andere Hosts (über ein Firmennetzwerk oder über das Internet) übertragen, bedarf es einer ausreichenden kryptografischen Sicherung vor der Übertragung, um die Vertraulichkeit dieser Daten zu schützen. Sensible Daten dürfen weiterhin nicht unverschlüsselt auf Datenbanken oder Dateisystemen abgelegt werden.

**A2.1 - Prüfung der Herkunft von Daten** Die Herkunft von Daten muss zweifelsfrei festgestellt werden. Dies setzt eine vollständige Umsetzung aller Anforderungen unter *A1.x.x* voraus. Daten, deren Herkunft nicht zweifelsfrei feststellbar ist, müssen zwingend verworfen werden, bevor sie interpretiert werden könnten.

**A2.2 - Prüfung der Echtheit von Daten** Alle Datenpakete müssen auf Echtheit überprüft werden. Pakete, deren Absender oder Inhalt manipuliert wurde, oder Replay-Pakete[23] müssen schnellstmöglich verworfen werden.

**A3 - Zugriffskontrolle** Alle Bereiche der Webseite, der Wanzen und der Datenbank müssen bei Aufruf die Authentizität des jeweiligen aufrufenden Nutzers/Systems prüfen. Für die Webanwendung reicht die Kontrolle der Gültigkeit der Websession, außer in den unter Punkt *A1.1.1* aufgeführten Fällen. Wanzen müssen stets durch das mitgesendete Zertifikat des CNC-Servers die Echtheit der Anforderungen prüfen können. Die Datenbank darf nur autorisierten Anwendungen zur Verfügung stehen. Schlägt diese Prüfung fehl, dürfen keinerlei Daten ausgegeben und keinerlei Funktionen ausgeführt werden, die nur berechtigten Personen und Systemen zugänglich sind.

[23] Pakete, die abgefangen und zu einem späteren Zeitpunkt erneut, manchmal auch mehrfach nochmals gesendet werden, um den authentischen Befehl zu einem für den Angreifer günstigen Zeitpunkt nochmals auszuführen.

Der Server sollte zudem nicht die Rechte besitzen, Dateien des Hostsystems zu manipulieren, um das Einschleusen von Malware, beispielsweise durch Code-Injection-Angriffe, zu verhindern.

**A4 - Sicheres Session-Management** Session-IDs müssen zufällig (d.h. mit nach aktuellen Standards ausreichender Entropie) erzeugt und vergeben werden. Die Übertragung der ID muss auf sicherem Wege stattfinden. Es ist empfehlenswert, Zugriffe auf die Webanwendung des CNC-Servers nur lokal zu gestatten, um einen Großteil sessionbasierter Probleme zu vermeiden. Jede Session muss durch den Nutzer explizit beendbar sein, um eine erneute Authentisierung für folgende Programmaufrufe erforderlich zu machen. Beendet ein Nutzer die Session nicht, so muss diese bei festgestellter Inaktivität des Nutzers automatisch beendet werden.

**A5 - Kontrollierte Einbindung von Daten** Sämtliche Teile der Anwendung lehnen eine Dateiübertragung oder -speicherung ab. Es werden ausschließlich Textdaten in serialisierter Form übertragen und verarbeitet. Jede Art der Verarbeitung muss gegen Code-Injection-Angriffe abgesichert werden. Einer gesonderten Absicherung bedarf es daher für Serverbefehle, die direkt durch die Wanzen in Form von Kommandozeilenparametern zur Ausführung kommen. Siehe hierzu die Sicherungsmaßnahmen unter Punkt *A1.2*.

**A6 - Protokollierung sicherheitsrelevanter Ereignisse** Sämtliche Programme müssen eine Funktion zur Protokollierung besitzen, sodass Ereignisse und Aktionen stets nachvollzogen werden können. Diese müssen einen ausreichend genauen Zeitstempel besitzen, um die genaue Reihenfolge der Ereignisse feststellen zu können. Alle Protokolle werden für berechtigte Personen lesbar zentral in der Serverdatenbank gespeichert. Clients senden Log-Daten verschlüsselt an den CNC-Server.

**A7 - Schnelle Versorgung mit Patches und Updates** Werden Schwachstellen für integrierte Bibliotheken oder Codebausteine entdeckt, müssen alle laufenden Instanzen durch eine aktualisierte Version ersetzbar sein. Hier müssen vor allem die laufenden Wanzenprogramme so lange deaktiviert werden können, bis eine aktualisierte Version vorliegt und benutzt werden kann.

**A8 - Schutz vor automatisierter Nutzung** Folgende Probleme können durch unberechtigte, automatisierte Nutzung entstehen:

1. Massenhafte Anmeldeversuche an der Weboberfläche
2. Replay-Angriffe auf an Wanzen versendete Befehle
3. Spam mit gefälschten Scan- und Logberichten

Anmeldungeversuche am Webserver müssen sowohl zeitlich als auch numerisch begrenzt werden. Das bedeutet, dass nicht mehr als eine bestimmte Anzahl an Anmeldeversuchen pro Minute eingehen dürfen, als auch ein Limit der Gesamtanmeldeversuche nicht überschritten werden darf, bevor Gegenmaßnahmen eingeleitet werden. Die Gegenmaßnahmen dürfen nicht zu einem Ausschluss des realen Nutzers aus dem System führen.

### 2.6.3 Umsetzung

Die Komplexität einiger oben genannter Anforderungen übersteigt den Rahmen der Masterarbeit deutlich. Es wurden daher bisher nicht alle Konzepte umgesetzt. Alle beschriebenen Anforderungen sind aber spätestens dann von Bedeutung, wenn das Pentesting Toolkit real eingesetzt werden soll. Nachfolgend werden die Umsetzungsstrategien aller bisher umgesetzten Punkte beschrieben:

**A5 - Kontrollierte Einbindung von Daten** Sämtliche Verarbeitungsroutinen, die Daten von Außen verwenden, benutzen eine genaue Spezifikation, welche Daten sie erwarten. Entsprechen die Daten nicht dem erwarteten Format, werden sie ignoriert bzw. eine Fehlermeldung generiert.

- Alle Daten, deren Definitionsbereich eingrenzbar ist, nutzen stets den engstmöglichen Definitionsbereich. So werden alle numerischen Werte stets in den entsprechenden Datentyp umgewandelt, auch wenn sie als Zeichenketten übergeben werden. Diese Umwandlung findet vor jeder sonstigen Verarbeitung der Daten statt und wird stets fehlerresistent ausgeführt.
  Dies gilt vorrangig für die REST-API aller Unterseiten der Webanwendung, die abhängig von der Client-ID eine Tabelle generieren und diese ID dafür an die SQL-Datenbank weiterreichen.
- Im Kommunikationsprotokoll werden soweit möglich ausschließlich Aufzählungstypen verwendet, die die Argumente für die Ausführung von Befehlen darstellen. Entsprechen die Daten keinem gültigen Wert der Aufzählung, werden sie verworfen. Das Kommunikationsprotokoll wird von den Wanzen und der TCP-Schnittstelle des CNC-Servers verwendet.
- Alle Datenbankaufrufe sind mittels Prepared Statements vor SQL-Injection-Angriffen geschützt. Injizierte SQL-Befehle können damit niemals die Datenebene verlassen, können also niemals als Befehl interpretiert und ausgeführt werden. Die Datenbank wird von der Webanwendung und der TCP-Schnittstelle des CNC-Servers verwendet.

**A6 - Protokollierung sicherheitsrelevanter Ereignisse** Alle Anwendungen haben eine von jedem Codeabschnitt aus erreichbare Funktion, um Fehler zu protokollieren. Die Konsolenanwendungen geben diese Fehler bisher in Textform auf dem jeweiligen Host aus. Hier muss noch eine sichere Senderoutine eingebaut werden, damit diese Fehlermeldungen auf der Weboberfläche für die Pentester einsehbar sind. Diese Daten sind teilweise äußerst sensibel, sodass hier zuerst eine zuverlässige Implementierung von *A2 - Sichere Datenübertragung* erfolgen muss. Bei jeder Fehlerbehandlungsroutine sollte abgewägt werden, ob ein Log dieses Fehlers notwendig sein könnte.

Die Loggingfunktionen besitzen auch eine Klassifizierung der Ereignisse:

1. **Debug** - Der Fehler beeinträchtigt die Funktionsweise des Programmes nicht, könnte aber auf potenzielle Programmierfehler hinweisen.

2. **Normal** - Der Fehler beeinträchtigt die Funktionsweise des Programmes kaum und könnte auf Programmierfehler, oder eine Fehlbedienung des Programmes hinweisen.
3. **Schwer** - Der Fehler führt zu einer spürbaren Beeinträchtigung des Programmes, sodass einige Funktionen nicht mehr ausgeführt werden können.
4. **Kritisch** - Der Fehler führt zum Versagen der Anwendung.

Diese Klassifizierung der Fehler führt zu einer deutlich erleichterten Fehlersuche und kann schwere Fehler deutlich hervorheben, sodass diese nicht durch eine Vielzahl kleiner Fehler überdeckt werden können.

**A7 - Schnelle Versorgung mit Patches oder Updates** Alle Serverkomponenten laufen auf einem von den Pentestern und Entwicklern der Anwendung kontrollierten System. Es stellt für die Funktionsfähigkeit des Pentesting Toolkits kein Problem dar, wenn Serverkomponenten kurzzeitig nicht erreichbar sind. Programmteile des Servers können daher einfach ersetzt werden.

Werden Fehler entdeckt, die die Funktionsweise der Wanzen beeinträchtigen oder gar die ausführenden Hostsysteme gefährden könnten, müssen die Wanzen schnellstmöglich deaktiviert und ausgetauscht werden. Die Serverkomponente sieht dafür einen Updatemechanismus vor, der die Wanzen sofort bei der Kontaktaufnahme zu einem Update zwingen kann und bis dahin jegliche weitere Kommunikation verweigert. Dieser Updatemechanismus kann jedoch noch nicht vom Nutzer ausgelöst werden, da eine Implementierung in den Wanzen ohne vollständige Implementierung der Anforderung *A2 - Sichere Datenübertragung* ein erhebliches Risiko für das Unternehmen darstellen könnte. Bisher könnte der Wanze ein falscher Updatelink oder ein gefälschtes Programmpaket angeboten werden, deren Ausführung mit erheblichen Risiken verbunden wäre. Wanzen reagieren daher noch nicht auf Updateanweisungen. Dies muss vor einem Produktiveinsatz unbedingt geändert werden.

Der Server ist allerdings bereits in der Lage, Wanzen komplett zu deaktivieren. Sie reagieren darauf mit einer eigenständigen Abschaltung und werden damit nie wieder erreichbar sein.

### 2.6.4 Risiken für einsetzende Unternehmen

**Risiken durch den Einsatz der Wanzen** Der Einsatz der Wanzen des Pentestin Toolkit muss mit äußerster Sorgfalt erfolgen. Jede zusätzliche Software, die in einem sensiblen Bereich des Unternehmens läuft, kann potenziell hohen Schaden anrichten oder gar eine zusätzliche Angriffsoberfläche darstellen, über die Zugriff auf dahinterliegende Systeme erlangt werden kann.

Um die Angriffsfläche zu minimieren, läuft das Pentesting-Toolkit mit minimalen Rechten auf den Hostsystemen. So wird die Möglichkeit eines Privilegue Escalation Angriffes über das PT durch einen Dritten wirksam verhindert. Selbst, wenn es einem Angreifer gelingt, Bugs in den Wanzen des PT auszunutzen, ist er dennoch nicht in der Lage, mehr Rechte, als ein normaler Nutzer zu erhalten. Ein Vorteil der eingesetzten Sprache

C# ist, dass die Speicherverwaltung vollständig vom Compiler übernommen wird[24]. Es ist dadurch kaum möglich, Pufferüberläufe in normalem Code mit Benutzerrechten zu verursachen. Auf die Implementierung mittels Zeiger in einem „unsafe“-Kontext wurde verzichtet.

Zu beachten sind jedoch Risiken, die auch durch eine Ausführung im Benutzerkontext entstehen können. Wanzen erlauben die Ausführung beliebigen Codes auf der Kommandozeile des Hostsystems. Diese Befehle werden von außen als einfache Datenpakete an die Wanzen übergeben. Es ist daher unbedingt notwendig, die Sicherheitsanforderung *A2 - Sichere Datenübertragung* aus Kapitel 2.6.2 vollständig umzusetzen. Wanzen dürfen Befehle von außen nur dann Ausführen, wenn sie zweifelsfrei verifizieren konnten, dass der CNC-Server der Absender ist. Nur so kann die Ausführung fremden Codes effektiv verhindert werden. Im Benutzerkontext denkbare Risiken sind die Veränderung von Konfigurationsdateien, die Installation weiterer Malware, die Löschung von Benutzerdaten oder eine unberechtigte Nutzung von Ressourcen und damit einhergehender Verlust der Verfügbarkeit und Integrität von Daten des Unternehmens.

Wird bekannt, dass ein Programmfehler in den Wanzen vorliegt, so gibt es zwei Möglichkeiten, das Unternehmen vor dadurch auftretende Gefährdungen zu schützen. Je nach Schwere des Fehlers können die Pentester alle Wanzen mit einer bestimmten Programmversion zu einem Update auf eine neuere Version zwingen, sobald ein Update vorliegt. Bei schwerwiegenden Fehlern kann ein weiterer Betrieb sofort durch eine Selbstdeaktivierung der Wanzen verhindert werden, wie in Kapitel 2.6.3 beschrieben. Hier ist stets die Sicherheit des Unternehmens vorrangig zu den bisher erbrachten Leistungen der Pentester in Form der Verbreitung der Wanzen im Unternehmensnetzwerk und der hieraus zukünftig möglichen Vorteile.

Das Pentesting-Toolkit dient dem Sammeln von Informationen über das Vorhandensein und die Möglichkeiten zur Ausnutzung von Schwachstellen. Dafür werden sensible Daten zum internen Aufbau des Unternehmensnetzwerkes gesammelt und über das Internet an den CNC-Server übermittelt. Diese Daten können Einbruchsversuche Dritter unterstützen, wenn sie in deren Hände fallen. Eine solche Übertragung darf nach einer korrekten Implementierung der Sicherheitsanforderung *A2 - Sichere Datenübertragung* aus Kapitel 2.6.2 durchgeführt werden. Die auf diesen Informationen basierende Ausführung von Exploits auf den Hostsystemen kann ebenfalls zu erheblichen Problemen führen. Zu erwartende Probleme sind Systemabstürze sowie Verlust der Verfügbarkeit und Integrität von Unternehmensdaten, die damit einhergehen können. Ein Einsatz von Exploits ist daher stets genau mit dem Unternehmen abzusprechen, siehe Kapitel 1.4.3.

**Risiken durch den Einsatz des CNC-Servers** Auch der Einsatz des CNC-Servers und seinen Datenbanken birgt Risiken für das Unternehmen, auch wenn diese Systeme außerhalb des Einflussbereiches des Unternehmensnetzwerkes laufen. Wanzen speichern äußerst sensible Daten zur internen Architektur des Unternehmensnetzwerkes, die durch die Wanzen automatisch generiert und versendet werden. Die Daten werden durch die Wanzen nicht persistent gespeichert, aber möglicherweise im Arbeitsspeicher gehalten. Zudem werden sie über einen unsicheren Kanal, das allgemeine Internet, übertragen.

Damit diese nicht abgefangen und von unauthorisierten Dritten gelesen und für ihre Zwecke missbraucht werden können, muss auch hier vor einem Einsatz eine vollständige Implementierung der Anforderung *A2 - Sichere Datenübertragung* aus Kapitel 2.6.2 sichergestellt werden.

Eine externe Eingabe von Daten und Befehlen birgt auch Risiken zum Missbrauch gegen das Unternehmen. Befehle könnten manipuliert, vervielfältigt oder mit gefälschtem Absender eingegeben und ausgeführt werden. Um dies zu vermeiden, muss die Herkunft und die Unveränderheit des Befehls geprüft werden. Auch an dieser Stelle können Gegenmaßnahmen nur mithilfe der vollständigen Implementierung der Anforderung *A2 - Sichere Datenübertragung* aus Kapitel 2.6.2 umgesetzt werden.

# 3 Technische Dokumentation

## 3.1 Pentest-Clients („Wanzen“)

### 3.1.1 Aufgaben der Clients

Die Clients (oder auch „Wanzen“) des Pentesting-Toolkits werden nach Beauftragung durch den Verantwortlichen eines zu testenden Unternehmens auf einen oder mehrere Workstations (oder den Teil davon, die im Kompetenzbereich des Auftraggebers liegt) geschleust.

Von dort aus haben Sie die Aufgabe, die Rechte des ausführenden Nutzers und die Lage der Workstation im Netzwerk auszunutzen, um Pentesting-Aufgaben zu übernehmen. Die Steuerung erfolgt über einen externen, zentralen Steuerungsserver. Die Berichterstattung erfolgt ebenfalls an diesen Server.

### 3.1.2 Programmbestandteile

Folgende Schlüsselelemente sind in der Wanzenanwendung enthalten:

**Program.cs:** Die Mainfunktion. Sie lädt alle wichtigen Funktionen für die Bearbeitung eingehender Pakete und stellt die Verbindung zum Command&Control-Server (CNC-Server) her. Diese Verbindung wird periodisch überprüft. Die Endlosschleife wird durchbrochen, sobald die Anwendung beendet werden soll. Dies kann durch einen Befehl des CNC-Servers oder durch Ausschalten des Hostsystems geschehen. In diesem Fall darf die Anwendung den Shutdown-Prozess des Hostsystems nicht behindern, um nicht aufzufallen. Zudem ist hier das Logging-System enthalten, welches für alle anderen Programmteile erreichbar ist.

**ServerInterface.cs:** Das ServerInterface stellt alle Funktionen zur Verfügung, um mit dem CNC-Server zu kommunizieren und eingehende Pakete zu verarbeiten und entsprechende Reaktionen auszulösen. Der Aufruf der Module zum Pentesting findet ebenfalls hier an zentraler Stelle im Rahmen der Paketverarbeitung statt (Funktion „HandlePackage“).

**Sender.cs:** Enthält die Funktionen zum Senden von Daten direkt an den CNC-Server.

**Listener.cs:** Enthält die Funktionen, die notwendig sind, um einen TCP-Listener auf dem Hostsystem zu platzieren. Da vorab nicht bekannt ist, welche Ports verfügbar sind, werden nacheinander alle Ports ab Portnummer 31415 bis 65432 probiert, bis ein TCP-Listener darauf platziert werden kann. Ist keiner dieser Ports verfügbar, bleibt die Wanze inaktiv.

**CnCReader.cs:** Diese Klasse enthält Funktionen zur Deserialisierung von Paketen, die nach dem Schnittstellenstandard des Pentesting-Toolkits serialisiert und gesendet wurden. Die ausgelesenen Pakete werden direkt an das ServerInterface zur weiteren Verarbeitung weitergeleitet.

### 3.1.3 Programmablauf

Für den Start des Programmes wird die Main-Funktion komplett durchlaufen. Damit das Programm dann nicht sofort beendet wird, wird eine Endlosschleife ausgeführt, die die Aufrechterhaltung der TCP-Verbindung zum CNC-Server sicherstellt. In der MainFUnktion werden alle für den korrekten Ablauf des Programmes benötigten Funktionen und Unterprogramme geladen. Der Ablauf wird nachfolgend in chronologischer Reihenfolge beschrieben:

**Trap** Bei Programmstart wird als erstes ein Trap[24] eingerichtet, der das Beenden des Programmes in jeglicher Form abfängt und für eine korrekte Schließung aller Verbindungen sorgt und die Entladung aller Ressourcen sicherstellt. Das Beenden des Programmes dauerte bei ersten Tests dadurch bis zu 5 Sekunden.

**ServerInterface** Das ServerInterface wird geladen, welches sofort versucht, einen Port für den TCP-Listener zu reservieren. Dafür wird als erstes die CNC-Reader-Klasse instanziiert, die die Auswahl eines geeigneten Ports übernimmt, den TCP-Listener erstellt und sofort auf eingehende Verbindungen wartet. Es sollte an dieser Stelle jedoch noch keine Verbindung von außen erstellt werden, da der CNC-Server die Adresse des Hostsystems der Wanze noch nicht kennt.

**Verbindungsaufbau zum Server** Sobald der TCP-Listener erfolgreich platziert wurde, wird nun versucht, eine Verbindung zum CNC-Server aufzubauen. Der Verbindungsaufbau verfolgt ein festes Protokoll (siehe Kapitel 3.1.5). Ist die Verbindung zum Server erfolgreich, wird auf eingehende Aufgaben gewartet und die Verbindung jede Minute geprüft und gegebenenfalls wieder aufgebaut.

**Abarbeitung von Paketen** Pakete werden von den Funktionen in der Datei CNCReader.cs eingelesen und gemäß den in der Kommunikationsschnittstelle beschriebenen Spezifikationen interpretiert. Es gibt fünf verschiedene Kommandos, auf die Wanzen reagieren. Sie sind an dem vorangestellten *C* zu erkennen.

- **CConnectionAccepted** teilt die erfolgreiche Anmeldung der Wanze am Server mit.
- **CConnectionRefused** teilt die Ablehnung der Anmeldung am Server mit. Hier wird eine Fehlermeldung als String an das Paket angehängt.
- **CExecutePayload** ist die Anweisung, eine bestimmte Aufgabe durchzuführen. Als zusätzliche Argumente sind ein Element der Aufzählung „Payload“ der Kommunikationsschnittstelle sowie zusätzliche Kommandos für die Befehlszeile erforderlich. Je nach Kommando können auch weitere Aufzählungen übergeben werden, die für die jeweilige Aufgabe in der Wanze implementiert sind. Dies spart Daten in der Übertragung besonders bei komplexen

[24] Ein Programmcode, der bei bestimmten Events ausgelöst wird. Dieser wird ausgeführt, wenn der Nutzer das Programm schließen möchte.

Befehlsketten. Die Wanze beantwortet eine solche Aufforderung mit einem „SCommandAccepted“- oder einem „SCommandRejected“-Paket.

- **CRequestHealthReport** ist eine einfache Aufforderung, einen erneuten Ping an den Server durchzuführen, also ein „SHeartbeat“-Paket zum Zeichen der andauernden Erreichbarkeit der Wanze zu versenden.

- **CUpdateRequired** ist Teil des Sicherheitskonzeptes und teilt der Wanze mit, dass der Server keiner weiteren Kommunikation zustimmt, bis die Wanze die aktuelle Version ihrer selbst vom Server geladen hat. Kann sie das Update aus irgendeinem Grund nicht durchführen, so ist die Wanze ab diesem Zeitpunkt nutzlos, um mögliche Gefahren für das Unternehmen durch gefundene Schwachstellen im Programm abzuwenden.

### 3.1.4 Programmstatus

Die Wanze besitzt 4 Zustände. Die Kommunikation mit dem CNC-Server ist abhängig vom Zustand. Sie geben Aufschluss darüber, ob eine Verbindung mit dem CNC-Server hergestellt werden konnte, oder nicht (**Disconnected, Connected**). Ist eine Verbindung zum Server verfügbar, so heißt das jedoch nicht, dass die Wanze vom Server zugelassen wurde. Ist dies der Fall, so wechselt sie in den Zustand „Initialized“. In diesem Zustand ist davon auszugehen, dass durch die Wanze an den CNC-Server gesendete Pakete auch verarbeitet werden. Zeitgleich ist dies der Ruhezustand, in dem sie Aufgaben des Servers entgegennehmen kann. Der Wechsel in diesen Zustand wird dem Server implizit mit der Übersendung eines Reports mitgeteilt. Hat die Wanze eine Aufgabe angenommen, so kann sie währenddessen keine weiteren Aufgaben annehmen. Sie wechselt daher für die Bearbeitung einer Aufgabe in den Status „Working“ und teilt dies dem CNC-Server implizit mit einer Bestätigung der Aufgabe mit. In der Weboberfläche sind alle derzeitigen Zustände der Wanzen zu sehen.

### 3.1.5 Kommunikation

Der Ablauf der Kommunikation wird vom Kommunikationsprogramm des CNC-Servers festgelegt. Die zulässigen Befehle und Argumente sind über die Bibliothek-Datei, der Kommunikationsschnittstelle, die alle Wanzen und der CNC-Server selbst einbinden müssen, abrufbar. Für Details siehe die entsprechende technische Dokumentation zum Kommunikationsprotokoll in Kapitel 3.3.

Der Ablauf der Kommunikation wird nachfolgend chronologisch beschrieben. Alle Daten, die die Wanze selbst senden muss, sind fett hervorgehoben:

1. **Anmeldung der Wanze beim CNC-Server**
2. *Bestätigung/Ablehnung der Anmeldung*
3. **Statusmeldung der Wanze**

4. *Zuteilen einer Aufgabe durch den Nutzer*
5. **Statusmeldung der Wanze**
6. **Aufgabenreport**
7. **Statusmeldung der Wanze**
8. Schließen der Verbindung

**Anmeldung am CNC-Server** Für die Anmeldung wird ein normaler Verbindungsversuch zu der bekannten Adresse des CNC-Servers unternommen. Dabei werden Authentifizierungsdaten und die Portnummer des platzierten TCP-Listeners übergeben, an die alle zukünftigen Antworten und Aufgaben des Servers gesendet werden sollen.

Derzeit bestehen die Authentifizierungsdaten aus dem Klartextstring „Some authentification data“. Im Rahmen eines produktiven Einsatzes wird hier das den Wanzen eigene Zertifikat codiert und übermittelt, welches zudem verschlüsselt übertragen wird, wie in der Sicherheitsbetrachtung der Masterarbeit beschrieben ist.

Konnte die Nachricht erfolgreich übermittelt werden, so geht die Wanze ab hier davon aus, dass eine Verbindung zum Server besteht und wechselt ihren Status zu „Connected“. Dies versetzt die Wanze in einen Wartezustand, bis die Verbindungsanfrage durch den CNC-Server bestätigt oder abgelehnt wird. Kommt keine Reaktion, so wird der Verbindungsaufbau durch die Endlosschleife der Main-Funktion periodisch wiederholt.

**Statusmeldungen der Wanze** Es gibt keine expliziten Pakete für die Mitteilung des aktuellen Zustandes. Jedoch können die Heartbeat-Pakete dahingehend erweitert oder zusätzliche Pakete eingefügt werden, für die bereits der Platzhalter „SHealthreport“ verfügbar ist. Alle Statusmeldungen werden dem Server implizit mitgeteilt. Der Server überwacht selbst den Zustand der Verbindung zur Wanze und schließt dabei auf die Zustände *Disconnected* und *Connected*. Wird der Wanze eine Aufgabe übermittelt, so bestätigt sie dies mit einem „SCommandAccepted“-Paket, wodurch sie offensichtlich in den Working-Zustand übergegangen ist. Kann der Befehl aus irgendeinem Grund (üblicherweise nicht unterstützte oder nicht lesbare Pakete) nicht ausgeführt werden, so wird dies dem Server mit einem „SCommandRejected“-Paket mitgeteilt.

Wurde eine Aufgabe bestätigt, so kann sie auf 2 Arten beendet werden. Durch einen fehlerbedingten Abbruch, oder durch Fertigstellung der Aufgabe. Im Normalfall wird das Ergebnis der Aufgabe mit der Übersendung eines „SReport“-Paket mitgeteilt. Im Fehlerfall wird ein „SCommandAborted“-Paket samt Fehlerinformationen versendet. In beiden Fällen kann der CNC-Server davon ausgehen, dass sich die Wanze wieder im *Initialized*-Zustand befindet.

### 3.1.6 Reports

Reports werden bei Abschluss einer jeden Aufgabe an den CNC-Server gesendet. Der Report bildet sich bei allen bisher implementierten Aufgaben aus der Standardausgabe

und der Fehlerausgabe des Kommandozeilenprogrammes. Ein Report besteht aus einem String-Array, wobei jedes Element eine Zeile der Ausgabe beherbergt.

Ein *SReport*-Paket besteht nur aus den üblichen Paketdaten, ergänzt um das String-Array. Da Wanzen nur eine Aufgabe zur selben Zeit übernehmen können, weiß der Server stets, auf welche Aufgabe sich der aktuelle Report bezieht.

## 3.2 Pentestserver

### 3.2.1 Aufgaben des Servers

Der „Control Server“ dient der Steuerung der Wanze sowie der Auswertung der von ihr gesendeten Daten. Beides wird über ein Webinterface bzw. über eine externe Applikation, bspw. eine App für mobile Systeme, realisiert. Im Grunde handelt es sich hier also um einen Vermittlungsserver, der Befehle der Steuerungsinterfaces der Pentester entgegennimmt und an die Wanzen weiterleitet. Umgekehrt kann er von den Wanzen kontaktiert werden und Nachrichten und Scanergebnisse entgegennehmen.

Der Server bietet eine Übersicht aller aktiven (und aufgespürten) Clients und liefert diverse Informationen wie Einsatzort (so genau, wie möglich), letzter Zeitpunkt einer erfolgreichen Verbindung und zur Verfügung stehende Befehle und Versionsinformationen.

Zur Verfügung stehende Befehle eines jeden Clients sind eine Updatefunktion für die Wanze, das Herunterladen und Ausführen einer Datei, Netzwerkscans und weitere Pentestingfunktionen. Diese müssen nicht zwingend innerhalb dieser Arbeit eingepflegt, sondern können später auch modular hinzugefügt werden.

### 3.2.2 Aufbau des Servers

Der Server wird öffentlich zugänglich sein, um von jedem Unternehmensnetzwerk aus zugreifbar zu sein. Da er Forschungszwecken dient und später eventuell im „Institut für angewandte Informationssicherheit“[25] Verwendung finden könnte, sollte er mit einem digitalen Zertifikat der Hochschule Merseburg ausgestattet sein.

Der Server ist in **zwei Module** unterteilt. Das erste Modul stellt den Webservice dar, der von allen eingesetzten Wanzen kontaktiert werden kann. Das zweite Modul fungiert als menschliche Schnittstelle und stellt alle Informationen für den Nutzer übersichtlich dar.

Für die Weboberfläche wird die Skriptsprache PHP zum Einsatz kommen, da wir an einer sicheren Umsetzung von Webseiten mit eben dieser Sprache innerhalb des SecurityLabs der Hochschule Merseburg forschen. Zudem ist diese frei verfügbar und kann auf jedem System ohne Probleme ausgeführt werden. Die Serverschnittstelle wird hingegen mit C# implementiert. Dies ist vorerst notwendig, um eine einfache (De-) Serialisierung der Nachrichtenpakete umzusetzen. Die Kommunikation zwischen den beiden Teilen des Servers ist jedoch nicht an einen Host gebunden und kann auch über ein Netzwerk stattfinden, solange beide Programmteile Zugang zur selben Datenbank haben.

Beide Module benötigen Zugriff auf eine gemeinsame Datenbank, welche Informationen zu den jeweiligen Clients wie Systeminformationen, Verbindungsdaten, Logs und Scanergebnisse speichert.

### 3.2.3 Kommunikations- und Koordinationsprogramm / TCP-Server

Der TCP-Server bildet das Herzstück der Anwendung. Über sie läuft die komplette Kommunikation zwischen Server und Wanzen. Zudem liegt es nahe, auch die gesam-

[25] `https://www.ifais.de/`

te Programmlogik und Ablaufsteuerung hier zu integrieren, damit die Kommunikation erheblich beschleunigt werden kann. Der Anwender hat für dieses Programm keine direkten Interaktions- oder Konfigurationsmöglichkeiten. Es dient der Aufrechterhaltung der Kommunikation zwischen Wanzen und Pentestern. Dies geschieht komplett automatisiert.

**Programmbestandteile** Folgende Schlüsselelemente sind in der Serveranwendung enthalten:

- **Program.cs:** Main-Funktion zum initialen Start aller Module und zum geregelten Beenden des Programmes. Beinhaltet auch das Log-Modul, auf das alle anderen Programmteile zugreifen können.
- **Database.cs:** Datenbankfunktionen, die alle verwendeten SQL-Statements beinhalten und von anderen Programmteilen aufgerufen werden können. Auf diese Weise können sichere SQL-Statements, Argumentchecks und Ergebnisse der Abfragen an zentraler Stelle verwaltet werden.
- **OpenConnections.cs:** Verwaltung aller Client-Verbindungen mit allen Informationen, die zum Senden und Empfangen von Daten mit diesem Client erforderlich sind.
- **ServerInterface.cs:** TCP-Schnittstelle mit allen die direkte Kommunikation betreffenden Funktionen.

#### Programmablauf

**1. Programmstart** Für den Start des Programmes wird die Main-Funktion komplett durchlaufen. Damit das Programm dann nicht sofort beendet wird, wird auf die Beendigung aller geladenen Endlosschleifen gewartet. Diese müssen demzufolge eine Abbruchbedingung haben, sodass ohne lange Verzögerung auf Wunsch des Nutzers beendet werden können. Der Ablauf wird nachfolgend in chronologischer Reihenfolge beschrieben:

**Trap** Bei Programmstart wird als erstes ein Trap[26] eingerichtet, der das Beenden des Programmes in jeglicher Form abfängt und für eine korrekte Schließung aller Verbindungen sorgt und die Entladung aller Ressourcen sicherstellt. Das Beenden des Programmes dauerte bei ersten Tests dadurch bis zu 5 Sekunden.

**Datenbank** Eine Verbindung zur Datenbank wird getestet. Die Datenbank basiert derzeit auf einer Datenkbankdatei („Pentesting_DB.mdf“). Die Verbindung wird stets nur so lange offen gehalten, wie für Operationen benötigt. Kann keine Verbindung zur Datenbank hergestellt werden, so wird das Programm mit entsprechender Fehlermeldung geschlossen.

[26] Ein Programmcode, der bei bestimmten Events ausgelöst wird. Dieser wird ausgeführt, wenn der Nutzer das Programm schließen möchte.

**TCP-Schnittstelle** Als nächstes wird versucht, den Port 31337 zu reservieren. Dieser ist allen Wanzen bekannt und sie werden versuchen, sich dorthin zu verbinden. Auf diesem Port wird nach Verbindungsversuchen gelauscht. Für jede eingehende Verbindung wird sofort ein neuer Thread geöffnet, in dem die Verbindung verwaltet, gehalten und weiter abgehört wird. Auf dem ursprünglichen Port wird damit sofort wieder nach neuen Verbindungen gelauscht.

**Connection Health** Es wird ein Thread gestartet, der im Minutentakt alle aktiven Verbindungen auf ihre Funktionsfähigkeit hin überprüft. Dies geschieht mit Ping-Paketen, auf die die Wanzen mit Heartbeat-Paketen antworten sollen. Sie tun dies nach einer solchen Aufforderung, oder aber auch im Minutentakt automatisch. Antworten Wanzen drei mal hintereinander nicht, so wird die Verbindung als getrennt angesehen und die Wanze muss sich für weitere Kommunikation erneut anmelden.

**Task Lookup** Es wird ein Thread gestartet, der die Datenbank in regelmäßigen Abständen nach Aufgaben durchsucht, die an Clients weitergeleitet werden sollen. Dieser Thread dient der Kommunikation des Webserver zum TCP-Server. Dieses Verfahren sorgt für eine lose Verbindung zwischen den Serveranwendungen und ermöglicht damit eine sehr einfache und freie Installation des Pentesting-Servers. Für einen korrekten Betrieb müssen also beide Programmteile Zugriff auf die selbe Datenbankinstanz haben, können aber ansonsten völlig frei sogar auf verschiedenen Hostsystemen betrieben werden. Wann die Aufgabe an die jeweilige Wanze zugestellt wird, entscheidet allein das Kommunikationsprogramm.

**2. Kommunikationsschleife**

**Verbindungsanfrage** Neue Verbindungsanfragen werden entgegengenommen und in einen neuen Thread ausgelagert, der diese Verbindung in Dauerschleife auf neue Daten abfragt, bis die Verbindung geschlossen wird.

**Zentrale Paketverwaltung** Alle eingehenden Daten werden an die zentrale Paketverwaltung übergeben. Sie liest eingehende Daten und löst Aktionen (zum Beispiel Datenbankzugriffe) und Antwortpakete entsprechend dem Kommunikationsprotokoll aus. Dieses wird im nächsten Unterkapitel detailliert dargestellt. Dieser Programmteil wird durch die lokale Clientverwaltung unterstützt, in der der Status aller seit Programmstart bekannten Clients gespeichert ist.

**Verbindungsabbau** Verbindungen werden verworfen, sobald der Client auf der anderen Seite nicht mehr reagiert. Die bisher bekannten Informationen und Verbindungsdaten über diesen Client bleiben gespeichert.

**3. Task- und Health-Schleife**

**Connection Health** Alle 60 Sekunden werden alle als aktiv markierten Verbindungen getestet. Hat ein Client in der Zwischenzeit keinerlei Daten versendet, so wird ein Testpaket an den Client gesendet. Ist das Versenden erfolgreich und kommt eine entsprechende Antwort zurück, so wird die Verbindung weiterhin als aktiv

markiert. Kann nach 3 Versuchen keine Aktivität festgestellt werden, so gilt die Verbindung als getrennt und der Client muss bei Bedarf eine neue Verbindung aufbauen.

**Tasks** Alle 60 Sekunden wird die Datenbank für alle Clients, die derzeit als aktiv markiert sind, nach Aufgaben abgefragt. Sind Aufgaben vorhanden, so werden entsprechende Pakete an die jeweiligen Clients versendet und der Datenbankeintrag entsprechend aktualisiert, um die Aufgabe nicht erneut zu senden.

**4. Beendigung des Programmes**

**Trap** Wird der Trap zur Beendigung des Programmes ausgelöst, so wird eine intern globale Exit-Variable auf true gesetzt. Alle Endlosschleifen des Programmes müssen diese Variable abfragen und ihre Arbeit abbrechen, sobald sie auf true gesetzt ist. Benutzen Schleifen lange Wartefunktionen, so müssen diese so gestaltet werden, dass sie sich dennoch innerhalb einer Sekunde durch einen Programmabbruch beenden lassen. Es ist daher nicht zulässig, *Wait()*-Befehle länger als eine Sekunde andauern zu lassen. Stattdessen werden bspw. 60 *Wait()*-Befehle mit je einer Sekunde Wartezeit aufgerufen, wobei jedesmal nach einem Programmabbruch getestet wird. Das Programm wird beendet, sobald alle Aktivitäten beendet worden sind.

**Kommunikation** Die Kommunikation folgt einem festen Ablauf. Nur so kann eine einfache Verwaltung theoretisch unbegrenzt vieler Clients erfolgen. Die jeweils benötigten Daten und der genaue Aufbau der Paketdaten sind der Dokumentation des Kommunikationsprotokolls in Kapitel 3.3 zu entnehmen.
Die vom CNC-Server zu sendenden Daten sind nachfolgend Fett gedruckt.

1. *Anmeldung der Wanze beim CNC-Server*
2. **Bestätigung/Ablehnung der Anmeldung**
3. *Statusmeldung der Wanze*
4. **Zuteilen einer Aufgabe durch den Nutzer**
5. *Statusmeldung der Wanze*
6. *Aufgabenreport*
7. *Statusmeldung der Wanze*
8. Schließen der Verbindung

Nach Fertigstellung einer Aufgabe und Übersendung eines Reports kann erneut eine Aufgabe übernommen werden. Die Verbindung wird von beiden Seiten solange wie möglich offen gehalten. Nicht aufgelistet sind hier Heartbeat-Pakete, die stets außer der Reihe gesendet oder angefragt werden können, um die Funktionsfähigkeit der Verbindung zu testen.

### 3.2.4 Webserver

Über den Webserver ist der Pentester in der Lage, sämtliche Informationen des Wanzenverbundes einzusehen und das Verhalten aller Wanzen zu kontrollieren.

#### Programmablauf

**Anmeldung** Jeder Nutzer, der sich mit dem Webserver verbindet, wird zunächst auf die Landingpage umgeleitet. Diese enthält allgemeine Informationen zum Projekt und allen Ansprechpartnern und was sonst benötigt werden könnte. Hier gibt es auch die Möglichkeit, sich über den Login-Button und entsprechenden Authentifizierungsdaten anzumelden. Nur bestimmte Nutzer haben Zugang zu der zentralen Wanzensteuerung. Der Login ist derzeit noch nicht implementiert.

**Wanzenübersicht** Die Landingpage für angemeldete Benutzer enthält zusätzlich eine Tabelle, in der alle bekannten Wanzen seit Serverstart mit allen wichtigen Details gelistet sind. Hier kann sich der Pentester einen ersten Überblick verschaffen und auswählen, von welcher Wanze er detailliertere Informationen benötigt.

Angezeigt werden

- Indikator, ob die Wanze derzeit aktiv und ansprechbar ist
- Hostname
- Beschreibung
- Zuletzt gemeldeter Status (Nicht authentifiziert, Verfügbar, Beschäftigt, ...)
- IP-Adresse
- Zeitpunkt des letzten Kontaktes
- Link zur Detailansicht

Diese Tabelle bietet bis auf die Auswahl eines Clients keine Interaktion, aktualisiert sich aber alle 5 Sekunden selbst, ohne die Seite neu laden zu müssen.

**Wanzendetailansicht** Alle Aufgaben und Interaktionsmöglichkeiten im Zusammenhang mit Wanzen stehen in der Detailansicht zur Verfügung. Hier sind nochmals alle Verfügbaren Informationen zu der Wanze aufgeführt. Zusätzlich zu den in der Gesamtübersicht verfügbaren Informationen sind noch

- Id-Nummer
- Aktivierungszeit der Wanze

aufgelistet. Die zugehörige Tabelle hat ein anderes Layout, aber die selbe Rendermethode, sodass auch diese sich alle 5 Sekunden selbstständig aktualisieren kann.

In der Wanzen-Detailansicht werden auch alle zur Verfügung stehenden Aufgaben angezeigt, die der Nutzer zuweisen kann. Je nach Aufgabentyp können auch verschiedene Argumente, wie zusätzliche Befehle oder Parameter eingegeben werden. Es obliegt dem Ersteller zusätzlicher Aufgabenmodule, wie diese auf dieser Seite eingebunden werden. Es ist auf ein konsistentes Design zu achten. Für eine zusätzliche Befehlszeile steht bereits ein entsprechendes Aufgabenfeld für die Kommandozeile zur Verfügung.

**Erstellung einer Aufgabe** Derzeit sind einige Beispielimplementationen von Aufgaben auswählbar. Hier sind verfügbar:

- Ausführen eines Pings
- Ausführen eines Host-Lookup mittels Nmap
- Ausführen des Windows Befehlszeilenprozessors (CMD)
- Update der Wanze
- Deaktivierung der Wanze

Geplant sind weitere Aufgaben sowie detailliertere Konfigurationsmöglichkeiten für jede Aufgabe. So soll Nmap eine Reihe von Auswahlmöglichkeiten bekommen, welche Aufgaben genau ausgeführt werden sollen. Diese könnten beispielsweise nach aufsteigender Aggressivität[27] sortiert sein.

Wurden Aufgaben an eine Wanze übergeben, so erstellt sie nach Fertigstellung einen Bericht und sendet diesen an den CNC-Server zurück. Ab diesem Zeitpunk ist dieser Bericht für den Pentester im Log-Bereich einsehbar.

**Log-Bereich** Der Log-Bereich zeigt alle Log-Berichte aller Wanzen an, die bisher in die Datenbank eingetragen wurden. Diese Daten überleben also auch einen Neustart der Anwendung, solange die Datenbank nicht gelöscht wird. Auch hier wird eine selbstaktualisierende Tabelle generiert, die alle wichtigen Daten enthält. Angezeigt werden

- ID des Berichts
- ID des Clients, der diesen Bericht erstellt hat
- Inhalt des Berichtes in Textform
- Zeitpunkt des Berichts (Absendedatum der Wanze)
- Schaltfläche, um den vollständigen Bericht in einer eigenen Ansicht anzuzeigen

[27] Aggressivität bedeutet hier, dass eine höhere Entdeckungsgefahr für die Wanzenaktivitäten im Netzwerk mit der Ausführung der Aufgabe einhergeht. Dies gilt für weitreichende Portscans, die eine hohe Netzwerklast bedeuten, oder intensive Scans eines einzigen Hosts, der dies als Angriff blockieren könnte.

### 3.2.5 Datenbank

Die Datenbank speichert alle Informationen, die von den Wanzen generiert werden. Sie soll als Anlaufstelle dienen, um Sicherheitslücken zu extrahieren und auf den jeweiligen Client angepasste Exploits erstellen zu können. Nachfolgend werden alle bisher vorhandenen Tabellen beschrieben.

**Clients** Die Clienttabelle speichert alle Metainformationen der Clients. Diese sind wichtig, um eine Verbindung aufrecht erhalten und seinen Wert anhand von Informationen zum Zustand des Clients, seiner erwarteten künftigen Erreichbarkeit und seiner Position im Unternehmen abschätzen zu können.

Gespeichert werden:

- Indikator, ob die Wanze derzeit erreichbar ist
- ID
- Hostname
- Gemeldeter Status (Offline, Online, Beschäftigt, Unbekannt)
- Beschreibung / Kommentar
- Letzte bekannte IP-Adresse
- Zeitpunkt des ersten Kontaktes
- Zeitpunkt des letzten Kontaktes

**Logs** In der Log-Tabelle sind alle von den Wanzen generierten Berichte gespeichert. Diese liefern wichtige Anhaltspunkte zu möglichen Schwachstellen auf Systemen im Netzwerk. Die Berichte sind abhängig von den ausgeführten Programmen. Die Wanzen liefern die Standardausgabe und die Fehlerausgabe der jeweiligen Befehlszeilenprogramme als Bericht aus.

Gespeichert werden:

- ID
- ID der Wanze, die den Bericht erstellt hat
- Textinhalt des Berichtes
- Zeitpunkt der Erstellung des Berichtes

**Tasks** Die Task-Tabelle dient als universale Schnittstelle zwischen der Server-Webanwendung und dem TCP-Interface. Hier werden alle Aufgaben gespeichert, die die Wanzen zugewiesen bekommen, inklusive deren Verarbeitungsstatus. Das Kommunikationsprogramm wird alle nicht zugewiesenen Aufgaben an Wanzen weiterleiten, sobald er Kontakt zu Ihnen aufbauen kann und die Aufgabe dann als „Zugewiesen“ ausweisen.

Gespeichert werden:

- ID
- ID der Wanze, die die Aufgabe bearbeiten soll
- Bearbeitungsstatus (Nicht zugewiesen (-1), zugewiesen (0), abgeschlossen (1))
- Name der Aufgabe[28]
- Argumente (zusätzliche Daten, Unterprogramme, Kommandozeilenargumente)[29]

[28] siehe Dokumentation zum Kommunikationsprotokoll
[29] siehe Dokumentation zum Kommunikationsprotokoll in Kapitel 3.3

## 3.3 Kommunikationsprotokoll

### 3.3.1 Aufgaben des Kommunikationsprotokolls

Das Kommunikationsprotokoll vereinheitlicht die Kommunikation auf Sender- und Empfängerseite der Programme des Pentesting-Toolkits in beide Richtungen. Dies betrifft die Kommunikation zwischen CNC-Server und aller Wanzen.

In der Schnittstelle ist das Aussehen eines jeden einzelnen Paketes (nur der Softwarelayer, alle darunter liegenden Ebenen sind durch den TCP/IP-Standard genau festgelegt), welches über das Internet an einen anderen Teil des Pentesting-Toolkits gesendet werden soll. Hier finden sich vorrangig Aufzählungstypen für mögliche Kommandos und Argumente, um die Programmierung zu vereinfachen und die zu übertragenden Datenmengen zu minimieren. Auf String-Inhalte soll damit weitestgehend verzichtet werden, da die Wartung und Implementierung auf beiden Seiten der Kommunikation erheblich erschwert wird.

Die daraus generierte Bibliothek heißt „PentestPackageInterchange.dll“ und wird von CNC-Server und Wanzen implementiert.

### 3.3.2 Aufbau der Schnittstelle

Die Schnittstelle beherbergt 3 Klassen für unterschiedliche Aspekte der Kommunikation:

- **DataPackage** beschriebt den Aufbau, die Serialisierung und die Deserialisierung eines jeden Paketes zum Austausch von Daten über das Internet.

- **Commands** ist eine Aufzählung aller möglichen Kommandos, die zwischen Server und Wanzen übertragen werden können und eine bestimmte Handlung veranlassen können. Ursprünglich waren hier zwei Klassen aufgeführt: Eine Klasse für alle Kommandos, die der Server verarbeiten kann und eine, die alle möglichen Kommandos für Wanzen beinhaltete. Dies führte jedoch zu Problemen und schlechterer Lesbarkeit des Codes, weshalb alle Elemente dieser Aufzählungen in eine einzige Klasse verschmolzen wurden. Um die Elemente trotzdem voneinander Unterscheiden zu können, beginnen alle Kommandos für den Server mit einem „S“ und alle Kommandos für die Wanzen (Clients) mit einem „C“.
  Zusätzlich zu den eigentlichen Kommandos sind auch Aufzählungen zu den verschiedenen Argumenten enthalten. Dazu gehören beispielsweise die verfügbaren Payloads für den „CExecutePayload“-Befehl, oder auch Tasks und Reporttypen.

- **ProgramState** ist eine Aufzählung aller möglichen Programmzustände der Wanzen. Diese ist für einen möglichen Austausch aktueller Programmzustände gedacht gewesen, dient nun aber lediglich der korrekten Benennung der Zustände auf beiden Seiten, da die Zustände nicht mehr direkt namentlich ausgetauscht werden müssen.

### 3.3.3 Programmbestandteile

**DataPackage** DataPackage ist die Klasse, die Inhalt und Aussehen der Datenpakete, die zwischen den Programmen des Pentesting-Toolkits ausgetauscht werden, bestimmt.

Durch die Einbindung dieser Klasse in alle kommunizierenden Programmteile ist ein einheitliches Kommunikationsschema sichergestellt, welches für die Datenverarbeitung auf beiden Seiten essentiell ist und die Arbeit für Programmierer erleichtert.

Zur Vereinfachung der zu sendenden Daten werden für viele Befehle und Informationen Aufzählungen zur Verfügung gestellt, die durch wenige zusätzliche Informationen ergänzt werden müssen.

Allen Datenpaketen gemein sind die Informationen über

- Hostnamen des Absenders,
- Zeit der Erstellung des Paketes,
- Pakettyp / Kommando und
- Zusätzliche Informationen als String-Array.

Die benötigten zusätzlichen Daten sind den nachfolgenden Erläuterungen zu den einzelnen Kommandos zu entnehmen. Die DataPackage-Klasse stellt zusätzlich Methoden bereit, um die Datenpaket-Objekte in Byte-Arrays zu serialisieren, sie also übertragungsfähig zu machen und zu deserialisieren, sie also nach der Übertragung wieder als Datenpaket-Objekt interpretieren zu können.

**Command** Die in der Aufzählung Command verfügbaren Elemente waren namensgebend für die übergeordnete Klasse „Commands“. Mit der Idee, auch häufig vorkommende Argumente in Aufzählungen zu kapseln, wurden hier weitere Aufzählungstypen hinzugefügt.

Die Kommandos können entweder für den Server, oder die jeweiligen Clients des Pentesting-Toolkits sein. Es gibt aber keine Elemente, die von beiden Seiten bearbeitet werden können. Für welche Seite der jeweilige Befehl bestimmt ist, ist an dem vorangestellten „S“ (Server) oder „C“ (Client/Wanze) zu erkennen.

Auf alle Kommandos folgt ein String-Array mit zusätzlichen Argumenten. Werden keine Argumente benötigt, so bleibt dieses Array leer.

Wanzen müssen die Verarbeitungsfunktionen für die folgenden Pakete besitzen. Wie sie das tun, ist der technischen Dokumentation in Kapitel 3.1 für die Wanzen zu entnehmen.

**CConnectionAccepted** Die Anmeldung der Wanze wurde vom Server akzeptiert. Die Wanze kann mit eingehenden Befehlen rechnen und von der Wanze gesendete Pakete werden verarbeitet.

Argumente:

- Keine

**CConnectionRefused** Die Anmeldung der Wanze wurde vom Server nicht akzeptiert. Die Wanze kann nicht mit eingehenden Befehlen rechnen und von der Wanze gesendete Pakete werden nicht verarbeitet.
Argumente:

- Grund für die Ablehnung (String).

Gründe für eine Ablehnung sind ein ungültiges Zertifikat, eine veraltete Programmversion oder eine bereits bestehende Verbindung mit dem (scheinbar) selben Client. Clients werden anhand des Hostnamens ihres Hostsystems unterschieden.

**CExecutePayload** Die Wanze soll eines der verfügbaren Pentestmodule unter den zusätzlich angegebenen Bedingungen ausführen.
Argumente:

- PayloadType (Aufzählung)
- Condition (Aufzählung), abhängig vom PayloadType
- AdditionalArguments (string)

Der Inhalt dieser Pakete ist stark abhängig von den auszuführenden Payloads. Die bisher integrierten Payloads CommandLine und Nmap haben noch keine speziellen Unterbefehle (Conditions) und brauchen immer Kommandozeilenbefehle (AdditionalArguments), die sie direkt an den Befehl anhängen und somit dem auszuführenden Programm weitergeben. Zukünftig können für Nmap bestimmte Befehle jedoch vorgegeben werden, sodass die Condition „SilentHostDiscovery“ vordefinierte Befehle in die Kommandozeile von Nmap schreibt, die die aktiven Hosts im Netzwerk offenlegt, ohne dass der Pentester die hierfür benötigten Argumente selbst nochmals übergeben muss.

**CRequestHealthReport** Der Server prüft, ob die eigenen Pakete noch bei den Wanzen ankommen und fordert für diesen Fall eine Bestätigung an. Die Wanze muss darauf mit einem „SHeartbeat“-Paket antworten, da die Verbindung nach dreimalig ausbleibender Antwort serverseitig getrennt wird.
Argumente:

- Keine

**CUpdateRequired** Die Anmeldung der Wanze wurde oder wird vom Server nicht mehr akzeptiert, da die Programmversion der Wanze veraltet ist. Sie muss die aktuelle Programmversion herunterladen.
Argumente:

- Update-URL (string)

Der Command&Control-Server reagiert auf die folgenden Befehle:

**SCommandAccepted** Die vom Server zuvor angeforderte Ausführung eines Payloads wurde entgegengenommen und kann ausgeführt werden. Der Payload ist also in-

stalliert und hat bisher keine Fehler verursacht. Die Wanze ist in den Working-Zustand übergegangen.
Argumente:

- Keine

**SCommandRejected** Die vom Server zuvor angeforderte Ausführung eines Payloads wurde entgegengenommen, kann aber nicht ausgeführt werden. Der Payload ist entweder nicht verfügbar und kann nicht automatisch installiert werden, verursachte sofort Fehler oder aber die Wanze befindet sich noch im Working-Zustand.
Argumente:

- Fehlermeldung (string)

**SCommandAborted** Die dem Server zuvor bestätigte Ausführung eines Payloads musste abgebrochen werden. Grund dafür kann das Herunterfahren des Hostsystems oder ein kritischer Fehler des ausgeführten Programmes sein.
Argumente:

- Fehlermeldung (string)

**SHeartbeat** Die Wanze testet den zum Server ausgehenden Kanal und teilt gleichzeitig ihre weiter bestehende Kommunikationsfähigkeit an.
Argumente:

- Keine

**SInitialConnection** Die Wanze möchte eine neue Verbindung zum Server aufbauen. Dies kann aufgrund einer Neuinstallation der Wanze oder durch den Wiederaufbau nach einem Verbindungsverlust geschehen. Es ist zu prüfen, ob die Wanze authentisch ist und zugelassen werden kann.
Argumente:

- Zertifikat (string)
- Listener-Port der Wanze (string)

**SReport** Das Ergebnis eines zuvor ausgeführten Payloads. Enthält alle Informationen, die das zugrundeliegende Programm ausgegeben hat sowie nicht kritische Fehler und Hinweise.
Argumente:

- Bericht - erste Zeile (string)
- ...
- Bericht - letzte Zeile (string)

**PayloadType** PayloadType enthält alle verfügbaren Payloads, die von den Wanzen ausgeführt werden können. Dies ist ein Argument für das Kommando „CExecutePayload“. Derzeit gibt es die Payloads

- Nmap
- Shell
- Systeminformationen

Abhängig von den Kommandos sind zusätzliche Argumente für die Ausführung notwendig. Shell benötigt einen String, der direkt an den Windows-Befehlszeilenprozessor weitergegeben wird.
Nmap benötigt Befehlszeilenargumente, die den Nmap-Befehl auf der Kommandozeile definieren. Wird hier kein Argument übergeben, so führt Nmap einen *Silent Host Discovery* durch.

**ReportType** ReportTypes sind derzeit noch nicht implementiert und dienen bisher als Platzhalter zur Unterscheidung von Berichten der Wanzen zwischen SystemLogs, Scanberichten und Fehlerberichten.

**SystemLogs** SystemLogs beinhalten alle Informationen, die durch die Wanzenanwendung selbst geloggt werden. Dazu gehören Debuginformationen wie Verbindungsauf- und -abbau, erhaltene Nachrichten und Anforderungen und die gesendeten Antwortpakete, allesamt versehen mit einem Zeitstempel.

**Scanberichte** Scanberichte enthalten alle Informationen, die die ausgeführten Kommandozeilenprogramme als Ergebnis präsentieren. Je nach Programm sind hier auch einige Hinweise für den Nutzer oder Debuginformationen vorhanden.

**Fehlerberichte** Fehlerberichte werden von Wanzen verschickt, wenn ein ausgeführtes Programm einen kritischen Fehler wirft. Dies ist meist mit der Angabe eines Fehlers verbunden, der an den CNC-Server weitergeleitet wird.

**ProgramState** ProgramState war ursprünglich zur Übertragung des Programmstatus der Wanze an den Server gedacht. Wie der Dokumentation der Wanzen in Kapitel 3.1 zu entnehmen ist, ist dies nicht mehr notwendig, da der Server anhand des Verhaltens der Wanze stets selbstständig auf den aktuellen Status schließen kann.

## 3.4 Hinzufügen neuer Payloads

Um einen neuen Payload hinzuzufügen, müssen alle Teilprogramme entsprechend angepasst werden, solange keine bessere Modularisierung umgesetzt wird.

**Kommunikationsprotokoll - „PentestPackageInterchange“**

1. Das neue Kommando muss als Element in die Aufzählung des Kommunikationsprotokolls eingetragen werden.
2. Die Weboberfläche verwendet eine identische Version der Schnittstelle, allerdings in PHP. In der „Commands.php“-Datei der API muss das gleiche Kommando an gleicher Stelle eingetragen werden. Die den Aufzählungselementen zugewiesenen Zahlen müssen übereinstimmen!

**Weboberfläche - „PentestingServer“**

1. Der neue Befehl muss eine Eingabemöglichkeit für den Benutzer bereitstellen. Diese werden in der Detailansicht einer jeden Wanze angezeigt, die in der Datei „editClient.php“ beschrieben ist. Hier können Knöpfe, verbunden mit Texteingabefeldern, Auswahlboxen oder ähnliche Elemente verwendet werden.
2. Im PHP-Bereich der selben Datei (oben) muss eine Submit-Action definiert werden. Hierfür ist bereits eine case-Auswahl vorhanden, die um den neuen Befehl und die Verarbeitung zusätzlicher Nutzereingaben ergänzt werden müssen, um einen korrekten Eintrag in die Tasks-Tabelle der Datenbank zu schreiben.

**Kommunikationsprogramm - „CNCServer“**

1. Das CNCServer-Programm fragt die Tabelle periodisch nach neuen Aufgaben ab. Hier muss eine Umwnadlung des Datenbankeintrages in einen entsprechenden Befehl umgewandelt werden, der als DataPackage an die Wanze gesendet werden kann.
2. In der Datei „Serverinterface.cs“ gibt es die Funktion „StartlookingupTasks()“. Diese enthält eine case-Auswertung aller möglichen Aufgabentypen der Tabelle Tasks. Diese muss um den neuen Befehl und eine entsprechende Umwandlung in ein Datapackage ergänzt werden.

**Wanze - „BuggingDevice“**

Schließlich muss die Routine des neuen Payloads in die Wanze selbst integriert werden.

1. In der Datei „ServerInterface.cs“ gibt es die Funktion „HandlePackage()“ mit einer Case-Auswertung aller möglichen Befehle, die die Wanze über DataPackages annehmen kann. Diese muss um den neuen Befehl ergänzt werden.
2. In der Verarbeitungsroutine für diesen neuen Befehl sind Argumente auszulesen und für die Initialisierung des Payloads zu verarbeiten.

3. Schließlich muss der neue Payload aufgerufen werden.
4. Handelt es sich nicht um ein gewöhnliches Kommandozeilenprogramm, welches über die integrierte Funktion „startCmdProgram()“ aufgerufen werden kann, die lediglich eine ausführbare Datei mit einem string von Parametern aufruft, so muss die Erstellung eines Berichtes ebenfalls an dieser Stelle integriert werden.

# 4 Abwehr des vorgestellten Angriffes

## 4.1 Voraussetzungen für den erfolgreichen Angriff

Die dargestellte Methode des Einschleusens der Wanzen basiert auf einer Reihe von Sicherheitsmängeln, die häufig anzutreffen sind.

1. Das Personal testet einen unbekannten USB-Stick an einem Firmenrechner oder PCs sind entsperrt und unbeaufsichtigt.
2. Der Zugang zu USB-Ports ist möglich, es gibt keine mechanische Sicherung der USB-Ports.
3. Plug&Play ist für USB-Geräte aktiviert.
4. Es gibt kein Schutzsystem, welches automatisierte Tastatureingaben erkennt und abfängt.
5. Verschiedene Hosts sind miteinander verbunden und dürfen kommunizieren.

Jeder dieser aufgeführten Punkte ist essentiell für den Betrieb des Pentesting Toolkit. Sobald eine Lücke geschlossen wird, wird eine Infektion mit einer Wanze des PT bereits wirkungsvoll verhindert.

## 4.2 Geeignete Gegenmaßnahmen

Folgende Abwehrmaßnahmen sind daher zu treffen:

1. Unbekannte USB-Sticks dürfen nicht an produktive Systeme angeschlossen werden.
2. USB-Ports sind, soweit möglich, für Unbefugte zu versperren. Ein Computer kann dafür in einen Schrank eingeschlossen werden.
3. Über die Gruppen-Richtlinien unter Windows kann die Nutzung von USB Ports, oder aber auch die automatische Installation unbekannter Geräte verhindert werden.
4. Der Zugriff von Hosts innerhalb des Unternehmensnetzwerkes wird durch Firewalls reglementiert. Es ist genau festzulegen, welche Hosts Zugriff auf andere Hosts benötigen, welche Dienste sie nutzen dürfen und welche Netzwerkpfade bzw. Dateien sie lesen oder gar überschreiben dürfen.

## 4.3 Automatisierte Abwehr

Es ist nicht immer möglich, alle oben genannten Regeln einzuhalten. USB-Sticks sind durchaus gängige Mittel zum Informationsaustausch, auch wenn es andere Möglichkeiten

gibt. Einige Hersteller[30] bieten daher ein automatisiertes Erkennungssystem für Angriffe, die auf dem USB Rubber Ducky oder ähnlicher Hardware basieren.

Diese suchen nach einer Reihe verdächtiger Indizien, die stark für ein automatisiertes Eingabesystem sprechen.

**Eingabegeschwindigkeit** Die einfachste Variante ist, die Geschwindigkeit der Nutzereingaben zu messen. Der USB Rubber Ducky tippt mit bis zu 1000 Wörtern pro Minute[16], mehr als jeder Mensch tippen könnte. Der Rekord für die schnellste Eingabegeschwindigkeit liegt etwa bei 174 Wörtern pro Minute.[28]

Es ist allerdings schwierig, nur anhand dieses Kriteriums eine Unterscheidung zwischen Nutzer und Maschine zu treffen. Die Eingabegeschwindigkeit des USB Rubber Ducky kann schließlich beliebig gedrosselt werden.

**Zugriff auf Tastaturtreiber** Ein zweites Kriterium ist die Installation neuer Treiber. Wird erkannt, dass der Systemtreiber für Tastaturen („kbdhid.sys“) geladen wird, könnte diese Aktion sofort blockiert werden. Dies erfordert allerdings einen höheren Aufwand, sollte tatsächlich mal eine neue Tastatur angeschlossen werden.

**Typische Befehle eines Angreifers** DigitalGuardian prüft daher weiterhin, ob über die neue Tastatur für Angriffe typische Befehle ausgeführt werden sollen. Dazu gehören solche, die Systeminformationen abfragen, wie

- dxdiag
- systeminfo
- winver
- msinfo32
- eventvwr
- inetcpl.cpl
- ipconfig
- perfmon
- resmon
- ...

oder Systemeinstellungen ändern. Dafür können auch Skripte mit Notepad für die Powershell, die Powershell selbst, die Kommandozeile oder auch der Registrierungseditor geladen werden.

[30] Zum Beispiel die „Advanced Threat Protection Engine“ von DigitalGuardian.[13]

**Einschätzung** Alle Kriterien zusammen, also die Eingabegeschwindigkeit, der kürzlich erfolgte Anschluss einer neuen Tastatur und die Eingabe typischer Befehle ergeben eine zuverlässige Entscheidung, ob es sich um einen legitimen Nutzer, oder einen aktiven Angriff handelt.

Einen einfacheren Ansatz verfolgt die Windows 10 - App „Disguised-Keyboard Detector“ von Penteract.[26] Diese sperrt den Computer immer dann, wenn eine neue Tastatur angeschlossen wird, was die Eingabe eines Passwortes erfordert. Zudem wird ein Warnhinweis für den Nutzer angezeigt, damit dieser die Entsperrung erst nach der Prüfung des neu angeschlossenen Gerätes vornimmt.

# 5 Ausblick

Das derzeit entwickelte Pentesting Toolkit erlaubt bisher die Verwaltung beliebig vieler Wanzen, die einen direkten Internetzugang haben und mit dem Server kommunizieren können. Diese können bereits beliebige Befehle auf einer Kommandozeile ausführen, den Host Discovery von Nmap nutzen und die Ergebnisse aller Operationen zurück an den Server senden. Dies erleichtert den Pentestern ihre Arbeit, da sie sich nicht mehr um SSH-Sessions zum Remote-Zugriff kümmern müssen und Aufgaben relativ leicht an viele Wanzen in kurzer Zeit deligieren können. Alle Informationen werden dabei an einem zentralen Ort gespeichert und sind für alle berechtigten Mitarbeiter zugänglich.

Der Einsatz der Software kann allerdings nur experimentell, nicht produktiv erfolgen, da sicherheitsrelevante Funktionen noch nicht abgesichert sind.

**Tunneling/Pivoting** Werden diese Funktionen implementiert, bietet das Pentesting Toolkit großes Potenzial für eine Weiterentwicklung. So könnte das Pentesting-Toolkit auch dafür eingesetzt werden, IP-Pakete und Befehle von außen an Hosts innerhalb des Netzwerkes weiterzuvermitteln. Auf diese Weise würde ein parallel-Netzwerk aufgespannt, durch das Hosts kontaktiert werden können, die von außen eigentlich nicht erreichbar sind. Diese Vorgehensweise ist als „Pivoting“ bekannt und wurde bereits in Kapitel 1.4.3 genannt. Eine Umsetzung hierfür würde jedoch einige Peer-to-Peer-Funktionen benötigen, die noch nicht implementiert wurden. In dieser Arbeit wurde vorerst auf dieses Feature verzichtet, da diese Funktionalität bereits unter dem Namen „Portfwd“[25] im Meterpreter enthalten ist.

**Modularisierung des Pentesting-Toolkits** Eine bessere Modularisierung des Programmes, speziell der verfügbaren Exploits, würde die Integration neuer Exploits vereinfachen. Notwendig dafür wäre eine einfache Beschreibungssprache für die Integration neuer Eingabemöglichkeiten in die Weboberfläche des Steuerungsservers, neuer Programmbefehle in das Kommunikationsprotokoll und die Installation benötigter Dateien, Treiber oder Befehle zur Ausführung der Exploits in die Wanzen. Bisher müssen all diese Änderungen manuell in die jeweiligen Programmteile eingepflegt werden, was einen hohen Einarbeitungsaufwand erfordert. Zudem müssen mit jeder Programmänderung alle Programme des Pentesting-Toolkits neu kompiliert werden.

# Literatur

[1] AO Kaspersky Lab. *New Petya / NotPetya / ExPetr ransomware outbreak. Yesterday, a global ransomware outbreak began, and it looks to be as big as the WannaCry story that broke not so long ago.* Hrsg. von AO Kaspersky Lab. 27.06.2017. URL: https://www.kaspersky.com/blog/new-ransomware-epidemics/17314/ (besucht am 19. 12. 2017).

[2] AVAST Software s.r.o. *Cerber Ransomware.* Hrsg. von AVAST Software s.r.o. 2016. URL: https://www.avast.com/c-cerber (besucht am 19. 12. 2017).

[3] AVAST Software s.r.o. *Locky ransomware.* 2016. URL: https://www.avast.com/c-locky (besucht am 19. 12. 2017).

[4] AVAST Software s.r.o. *Petya.* Hrsg. von AVAST Software s.r.o. 2016. URL: https://www.avast.com/c-petya (besucht am 19. 12. 2017).

[5] AVAST Software s.r.o. *WannaCry.* 2016. URL: https://www.avast.com/c-wannacry (besucht am 19. 12. 2017).

[6] Bundesamt für Sicherheit in der Informationstechnik. *Durchführungskonzept für Penetrationstests.* Hrsg. von Bundesamt für Sicherheit in der Informationstechnik. URL: https://www.bsi.bund.de/DE/Publikationen/Studien/Pentest/index_htm.html (besucht am 07. 11. 2017).

[7] Bundesamt für Sicherheit in der Informationstechnik. *IT Grundschutzkompendium. APP.3.1 Webanwendungen.* 2017. URL: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKompendium/bausteine/APP/APP_3_1_Webanwendungen.html (besucht am 19. 12. 2017).

[8] Bundesamt für Sicherheit in der Informationstechnik. *Studie zur IT-Sicherheit in kleinen und mittleren Unternehmen. Grad der Sensibilisierung des Mittelstandes in Deutschland.* Unter Mitarb. von secunet Security Networks AG. 2011.

[9] Check Point Research Team. *JAFF. A New Ransomware is in town, and it's widely spread by the infamous Necurs Botnet.* Hrsg. von Check Point Software Technologies Ltd. 11.05.2017. URL: https://blog.checkpoint.com/2017/05/11/jaff-new-ransomware-town-widely-spread-infamous-necurs-botnet/ (besucht am 19. 12. 2017).

[10] CNET Security. „Hacked NSA tools could put some windows users at risk. A hacking group says it leaked government malware designed to break into Windows computers. But Microsoft says its up-to-date software is safe." In: (15. Apr. 2017). URL: https://www.cnet.com/news/hacked-nsa-tools-put-windows-users-at-possible-risk/ (besucht am 19. 12. 2017).

[11] Andrew Lee David Harley. *Heuristic Analysis. Detecting Unknown Viruses.* 2009. URL: https://www.welivesecurity.com/media_files/white-papers/Heuristic_Analysis.pdf (besucht am 27. 11. 2017).

[12] Der Tagesspiegel. „BKA übernimmt Ermittlungen nach weltweiter Cyber-Attacke. Ein Hackerangriff mit der Schadsoftware "Wanna Cry" hat zehntausende Rechner in 99 Ländern lahmgelegt. Betroffen waren auch Russlands Innenministerium und die Deutsche Bahn. Bundesamt fordert zu Updates auf.“ In: *Der Tagesspiegel* 2017 (13. Mai 2017). URL: http://www.tagesspiegel.de/weltspiegel/wanna-cry-bka-uebernimmt-ermittlungen-nach-weltweiter-cyber-attacke/19797084.html (besucht am 19. 12. 2017).

[13] Digital Guardian. *Detecting A Rubber Ducky USB Attack With Digital Guardian Advanced Threat Protection (Video Demo)*. 2016. URL: https://digitalguardian.com/blog/detecting-rubber-ducky-usb-attack-digital-guardian-advanced-threat-protection-video-demo (besucht am 19. 12. 2017).

[14] Claudia Eckert. *IT-Sicherheit. Konzepte - Verfahren - Protokolle*. ger. 9., aktualisierte Aufl. Berlin: De Gruyter Oldenbourg, 2014. 990 S. ISBN: 978-3-486-77848-9. DOI: 10.1515/9783486859164. URL: http://www.degruyter.com/search?f_0=isbnissn&q_0=9783486859164&searchTitles=true.

[15] Gordon Lyon. *Nmap*. URL: https://nmap.org/ (besucht am 19. 12. 2017).

[16] Hak5 Gear. *USB Rubber Ducky*. 2017. URL: https://hakshop.com/products/usb-rubber-ducky-deluxe (besucht am 19. 12. 2017).

[17] Hauke Gierow. „Petya-Kampagne nutzt Lücke in Buchhaltungssoftware“. In: *Golem.de* 2017 (28. Juni 2017). URL: https://www.golem.de/news/ransomware-petya-kampagne-nutzt-luecke-in-buchhaltungssoftware-1706-128631.html (besucht am 19. 12. 2017).

[18] IT Governance Blog. *Vulnerability scans & false positives: The importance of sanitising input*. 2012. URL: https://www.itgovernance.co.uk/blog/vulnerability-scans-false-positives-the-importance-of-sanitising-input/.

[19] Sebastian Kübeck. *Web-Sicherheit. Wie Sie Ihre Webanwendungen sicher vor Angriffen schützen*. ger. 1. Aufl. Heidelberg: mitp, 2011. 331 S. ISBN: 9783826690242.

[20] Gordon Lyon. *Nmap network scanning. Official Nmap project guide to network discovery and security scanning*. eng. Zero-day release: May 2008. Sunnyvale, CA: Insecure.Com LLC, 2010. 434 S. ISBN: 978-0-9799587-1-7.

[21] Malwarebytes Labs. *Cybercrime tactics and techniques Q2 2017*. Unter Mitarb. von Adam Kujawa u. a. 2017. URL: https://www.malwarebytes.com/pdf/white-papers/CybercrimeTacticsAndTechniques-Q2-2017.pdf (besucht am 02. 08. 2017).

[22] Martin Holland und Axel Kannenberg. „WannaCry: Angriff mit Ransomware legt weltweit Zehntausende Rechner lahm“. In: *Heise Online* (12. Mai 2017). URL: https://www.heise.de/newsticker/meldung/WannaCry-Angriff-mit-Ransomware-legt-weltweit-Zehntausende-Rechner-lahm-3713235.html (besucht am 19. 12. 2017).

[23] Michael Messner. *Hacking mit Metasploit. Das umfassende Handbuch zu Penetration Testing und Metasploit*. ger. 2., aktualisierte. u. erw. Aufl. Safari Tech Books Online. Heidelberg: dpunkt-Verl., 2015. 588 S. ISBN: 9783864902246. URL: http://proquest.safaribooksonline.com/9781457194221.

[24] Microsoft Docs. *Unsafe Code*. 2017. URL: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/unsafe-code (besucht am 19. 12. 2017).

[25] Offensive Security. *Metasploit Dokumentation - Portfwd*. 2017. URL: https://www.offensive-security.com/metasploit-unleashed/portfwd/ (besucht am 19. 12. 2017).

[26] Penteract LLC. *Penteract Disguised-Keyboard-Detector*. 2017. URL: https://www.microsoft.com/de-de/store/p/penteract-disguised-keyboard-detector/9p57qf7pgm4m?rtc=1 (besucht am 19. 12. 2017).

[27] Proofpoint. „Necurs Botnet Returns With Updated Locky Ransomware In Tow“. In: *Proofpoint.com* (22. Juni 2016). URL: https://www.proofpoint.com/us/threat-insight/post/necurs-botnet-returns-with-updated-locky-ransomware-in-tow (besucht am 19. 12. 2017).

[28] Recordsetter. *Fastest Endurance Typing Performance*. Hrsg. von Recordsetter. 2011. URL: https://recordsetter.com/world-record/endurance-typing-performance/14115?autoplay=false (besucht am 19. 12. 2017).

[29] Enno Rey, Michael Thumann und Dominick Baier. *Mehr IT-Sicherheit durch Pen-Tests. Optimierung der IT-Sicherheit durch gelenktes "Hacking" ; von der Planung über die Vertragsgestaltung zur Realisierung*. ger. 1. Aufl. Edition <Kes>. Thumann, Michael (VerfasserIn) Baier, Dominick (VerfasserIn). Wiesbaden: Vieweg, 2005. 218 S. ISBN: 978-3528058395.

[30] Ahmed Sabbir und Wolfgang Stuerzlinger. „Analysis of Text Entry Performance Metrics“. In: *Proceedings of the IEEE Toronto International Conference–Science and Technology for Humanity (TIC-STH '09)*, S. 100–105.

[31] Shwed und IL) Gil (Jerusalem. „System for securing inbound and outbound data packet flow in a computer network“. 5,606,668. Checkpoint Software Technologies Ltd. 15.12.1993.

[32] Symantec Corporation. *Tech Note: False Positives with Vulnerability Scanning*. 7.02.2011. URL: https://support.symantec.com/en_US/article.TECH148894.html (besucht am 19. 12. 2017).

[33] Andrew S. Tanenbaum, David J. Wetherall und Katharina Pieper. *Computernetzwerke*. ger. 5., aktualisierte Aufl. Bd. 4137. it - Informatik. Wetherall, David J. (VerfasserIn). München: Pearson Higher Education, 2012. 1032 S. ISBN: 9783868941371.

[34] Wikipedia, Hrsg. *Ray Tomlinson*. Wikimedia Foundation, Inc. 2017. URL: https://en.wikipedia.org/wiki/Ray_Tomlinson (besucht am 25. 07. 2017).

[35] Zack Whittaker. „NSA's arsenal of Windows hacking tools has leaked. The NSA used th eWindows hacking tools to target several banks.“ In: *ZDNET Zero Day* 2017 (14. Apr. 2017). URL: `http://www.zdnet.com/article/shadow-brokers-latest-file-drop-shows-nsa-targeted-windows-pcs-banks/` (besucht am 19. 12. 2017).

# Abbildungsverzeichnis