

## Aufgabe 4

1.)

Der Security Account Manager (SAM) ist ein Systemdienst bei Windows Betriebssystemen, der Benutzer Passwörter, bzw. die daraus erzeugten LM oder NTLM Hashes, in einer Datenbankdatei speichert und den Validierungsprozess während der Anmeldung verwaltet.

Neben dem Passwort Hash enthält ein Eintrag in dieser Datenbankdatei auch noch Informationen zum Benutzer, etwa den Benutzernamen und die Zugehörigkeit zu einer Benutzergruppe.

2.)

Dateipfad des Dienstes:

%SystemRoot%/system32/lsass.exe

Dateipfad der Datenbankdatei:

%SystemRoot%/system32/config/SAM

Das Betriebssystem schützt den Zugriff auf diese Datei, indem durch interne Prozesse auf sie zugegriffen wird und sie somit für andere Zugriffe blockiert ist. Somit kann die Datei nicht geöffnet oder kopiert werden.

3.)

Bei einem Rainbow-Table-Angriff wird ausgenutzt, dass bei der Erstellung eines Passwort Hashes kein Salt verwendet wurde.

Ein ausgelesener Passwort-Hash wird dabei in einer Datenbank gesucht, die vorberechnete Hashes beliebiger Zeichenfolgen bis zu einer gewissen Länge enthält. Bei Passwörtern der Länge 6 ist die Größe dieser Datenbank ca 2,3 TByte, wenn alle Permutationen enthalten sind. Diese Datenbank nennt man Rainbow-Table.

NTLM Hashes sind anfällig, weil Sie keinen Salt verwenden.

4.)

Mimikatz ist ein Tool, das entwickelt wurde um zu demonstrieren wie eine Schwachstelle bei der Authentifizierung von Windows ausgenutzt werden kann, um zwischengespeicherte Anmeldedaten eines Windows Rechners abzugreifen. Heutzutage kann das Tool benutzt werden um unterschiedliche Arten von Sicherheitslücken nachzuweisen.

5.)

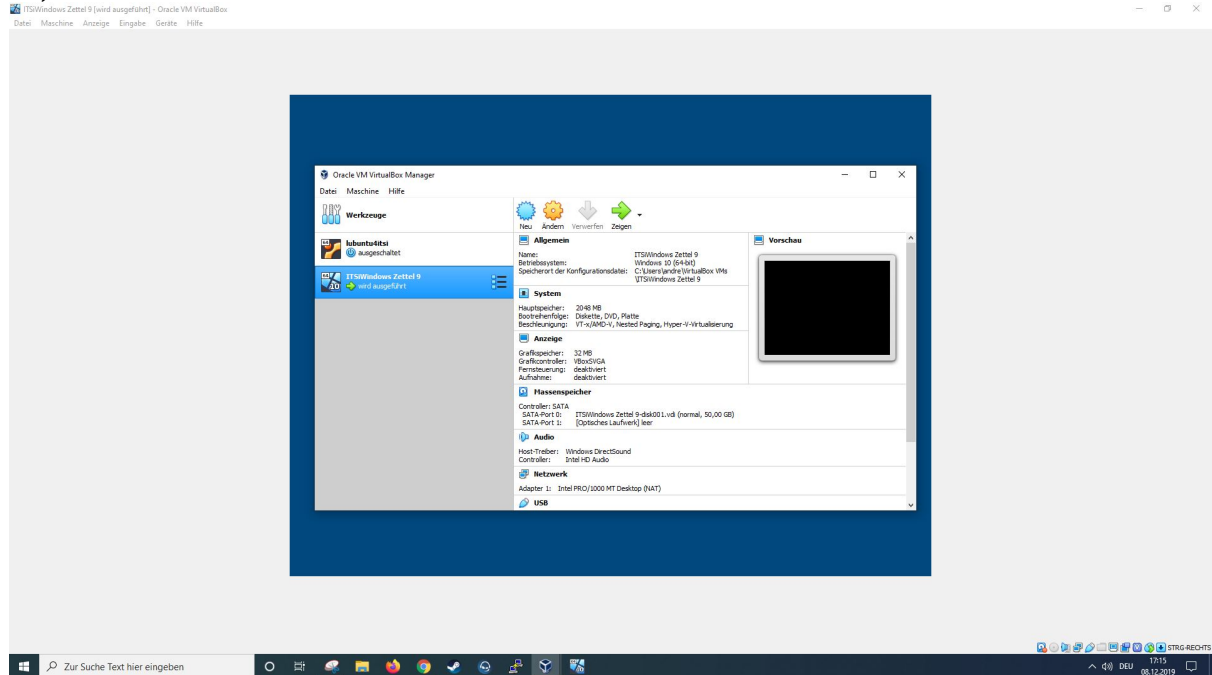


Fig. 5.1: VM wurde importiert und gestartet

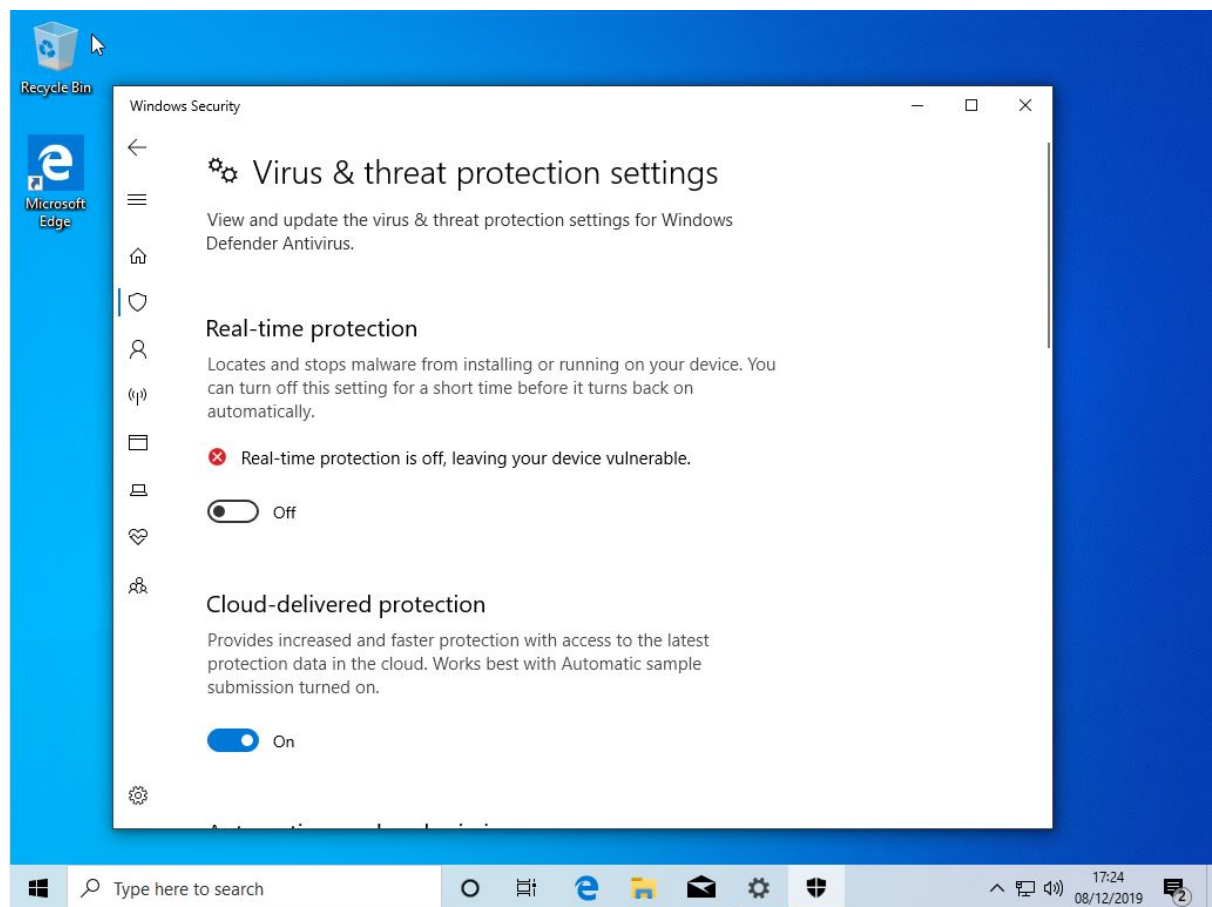


Fig. 5.2: Viruserkennung sollte ausgeschaltet werden bevor mimikatz runtergeladen werden kann, hier wurde zuerst nur Echtzeit Schutz deaktiviert, später auch alle anderen Schutzmechanismen, nachdem der Download von mimikatz nicht funktioniert hat (siehe Fig. 6.1)

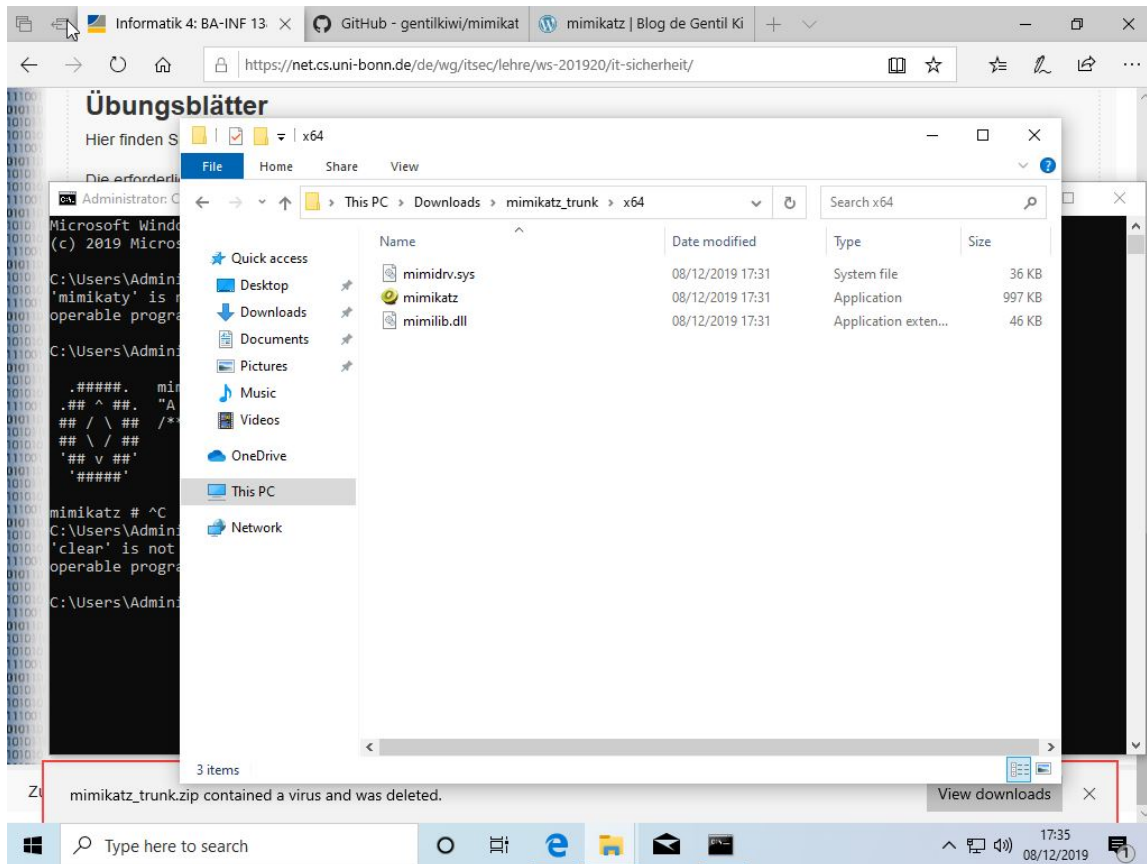


Fig. 5.3: Aktuelle mimikatz Version heruntergeladen

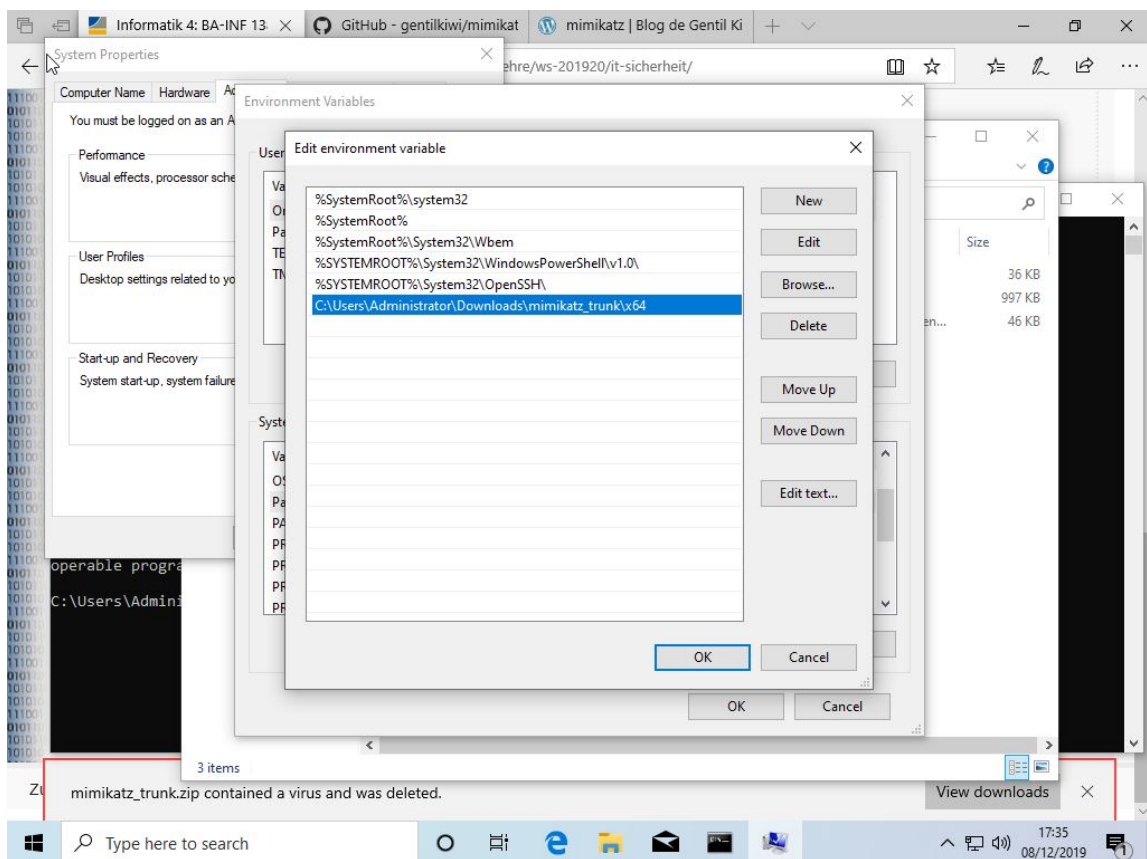
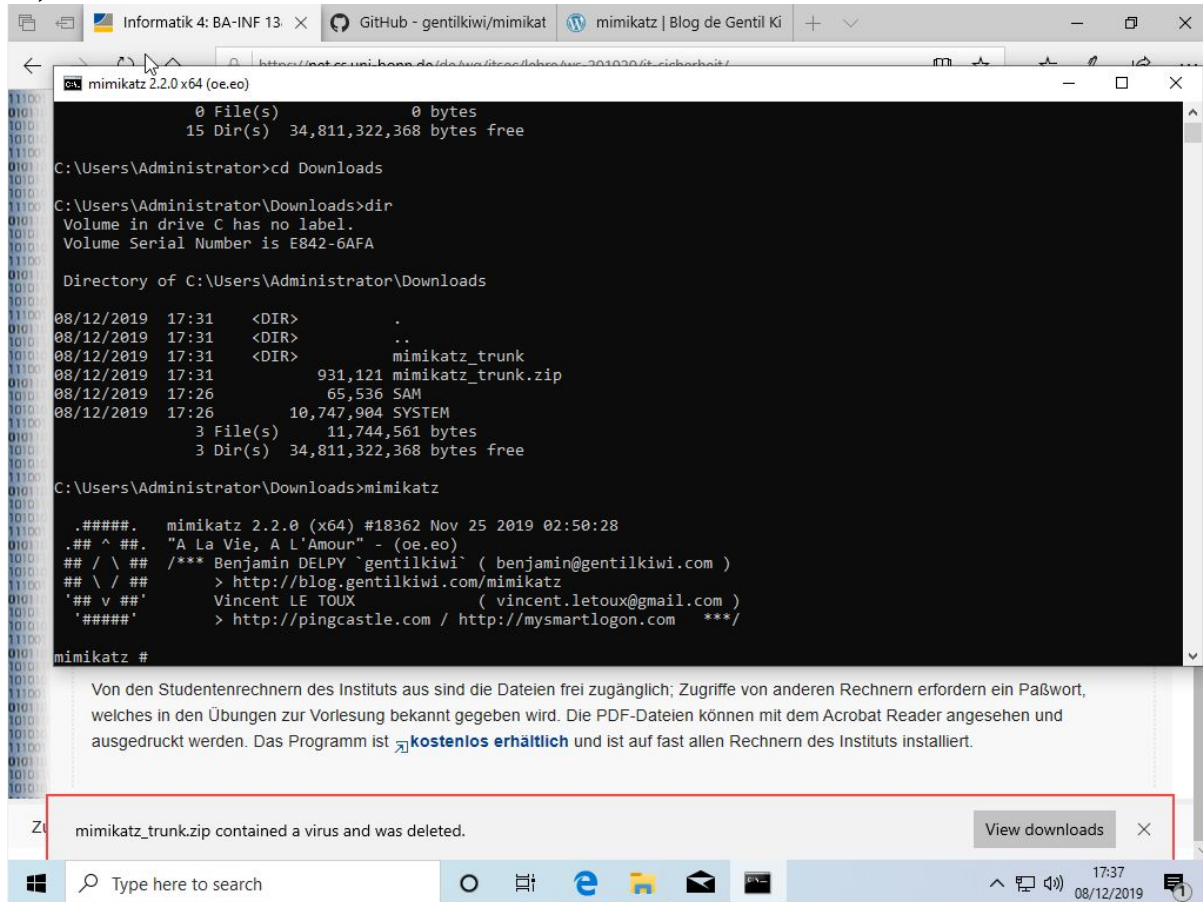
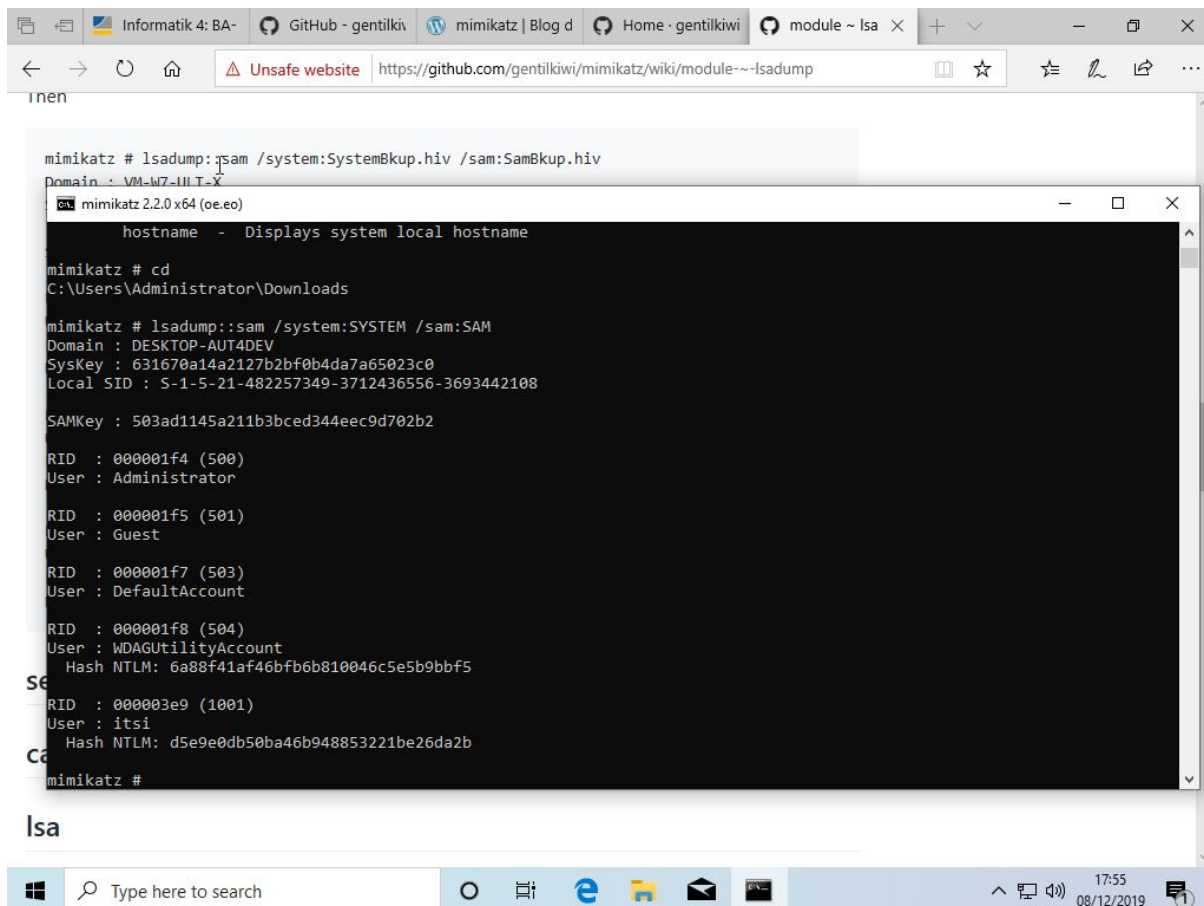


Fig. 5.4: Umgebungsvariable für einfachere Benutzung gesetzt

6.)

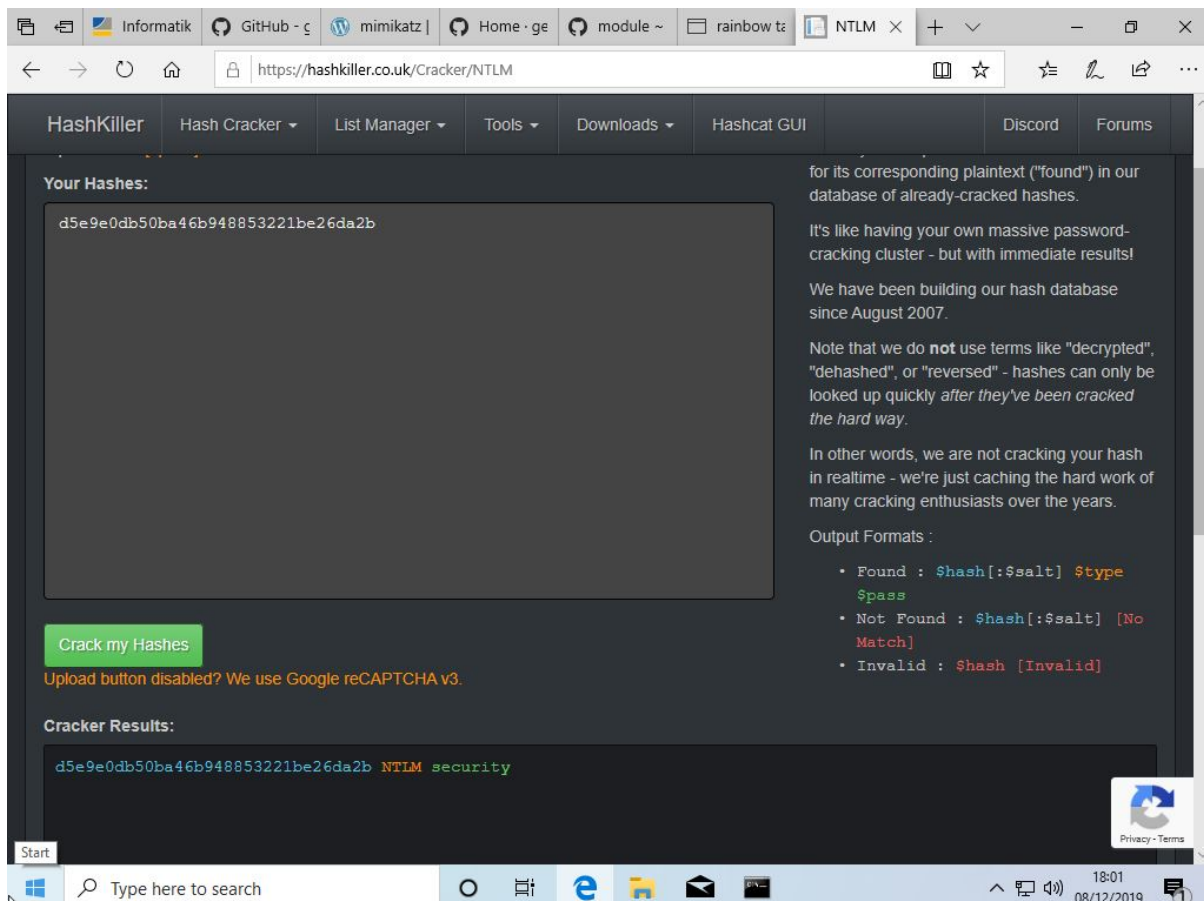


*Fig. 6.1: Mimikatz gestartet. Die Fehlermeldung das mimikatz\_trunk.zip gelöscht wurde war eine ältere Meldung, weil zuerst nicht alle Schutzmechanismen der Windows Virenerkennung ausgeschaltet wurden. (Siehe Fig. 5.2)*



```
mimikatz # lsadump::sam /system:SystemBkup.hiv /sam:SamBkup.hiv
Domain : VM-W7-III-T-X
mimikatz 2.2.0 x64 (oe.oe)
hostname - Displays system local hostname
mimikatz # cd
C:\Users\Administrator\Downloads
mimikatz # lsadump::sam /system:SYSTEM /sam:SAM
Domain : DESKTOP-AUT4DEV
SysKey : 631670a14a2127b2bf0b4da7a65023c0
Local SID : S-1-5-21-482257349-3712436556-3693442108
SAMKey : 503ad1145a211b3bcd344eec9d702b2
RID : 000001f4 (500)
User : Administrator
RID : 000001f5 (501)
User : Guest
RID : 000001f7 (503)
User : DefaultAccount
RID : 000001f8 (504)
User : WDAGUtilityAccount
Hash NTLM: 6a88f41af46bf6b810046c5e5b9bbf5
RID : 000003e9 (1001)
User : itsi
Hash NTLM: d5e9e0db50ba46b948853221be26da2b
mimikatz #
```

*Fig. 6.2: Mittels lsadump::sam lässt sich mithilfe der SYSTEM Datei die SAM Datei auslesen. Der NTLM Hash für den Benutzer „itsi“ lautet: d5e9e0db50ba46b948853221be26da2b*



*Fig. 7.1: Aus dem zuvor gewonnenem NTLM Hash können wir mittels einem über die Google Suche gefundenen Anbieter das Passwort als Klartext einsehen. Das Passwort des Benutzers „itsi“ lautet „security“*

## Aufgabe 1

geheim:

- Der Angreifer muss zuerst die Schwachstelle finden
- Es würde dem Ruf der Institution nicht schaden, da niemand davon weiß

nicht geheim:

- Hersteller kann die Schwachstelle beheben
- Die Öffentlichkeit weiß von der Schwachstelle und handelt vorsichtiger
- Der Hacker kann die Schwachstelle nutzen, muss es jedoch vor den Verbesserungen des Herstellers schaffen

## Aufgabe 2

Eine Rainbow Table ist eine kompakte Repräsentation von zusammenhängenden Passwortsequenzen, welche auch Ketten (chains), genannt werden. Eine Kette startet mit einem initialen Kennwort, das durch eine Hashfunktion geleitet wird. Der Hash der herauskommt, wird durch eine Reduktionsfunktion geleitet, um ein weiteres Klartextkennwort zu haben. Dieser Prozess wird mit einer vorgegebenen Anzahl wiederholt und anschließend wird das erste Kennwort der Kette zusammen mit dem Kennwort aus dem letzten Hashwert gespeichert.

Bei dem erstellen der Tabelle muss darauf geachtet werden, dass kein Kennwort, dass in einer Kette vorkommt, als Startkennwort verwendet wird und dass alle Kennwörter in der Tabelle vorkommen. Die Tabelle wird nur einmal erstellt und dient als Nachschlagetabelle.

Man verwendet einen zweistufigen Prozess um ein Kennwort herauszufinden. Zuerst wird der Hash der herauszufindenden Kennwort durch die Hash-Reduktion-Sequenz geführt, bis das Ergebnis der Reduktion in der Tabelle der letzten Kettenglieder vorkommt. Dadurch bekommt man das Startkennwort der Kette und kann anschließend im zweiten Schritt von diesem ausgehend die Hash-Reduktion-Sequenz anwenden, um das gesuchte Kennwort zu erhalten.

Die Länge der Kette, welche auch die Anzahl der Iterationen zur Erstellung der Tabellen sind, wirken sich auf die Größe der Tabelle aus. Je länger die Iteration gewählt werden, desto kleiner ist die entstehende Tabelle. Im einfachsten Fall ist die Anzahl der Iterationen gleich 1, sodass die Tabelle alle Kennwort-Hash-Paare enthält.

Einer Binärfolge wird mithilfe der Hashfunktion eine Binärfolge fester Länge zugeordnet. Zu einer zufälligen Zeichenkette der Länge  $n$  wird ein Hashwert berechnet. Das Ergebniss der Hashfunktion wird durch eine Reduktionsfunktion in eine neue Zeichenkette

umgewandelt,  
die wieder der Länge  $n$  entspricht. Da die Hintereinanderausführung von Hashfunktion und Reduktionsfunktion die Länge der Zeichenkette nicht ändert, können diese beiden Schritte beliebig oft abwechselnd wiederholt werden. Die Folge bildet am Ende eine Kette und es werden Anfangs- und Endwert gespeichert. Diese Schrittfolge wird  $x$  mal wiederholt und bildet so eine universelle Rainbow Table.

Eine Reduktionsfunktion verkürzt einen Hashwert auf  $n$  Zeichen. Jede Reduktion liefert z. B. durch MD5 eine neue „eindeutige“ 128 bit Zeichenkette, oder eine Kollision. Als eine Kollision bezeichnet man dabei einen Hashwert, der durch verschiedene Ausgangszeichenfolgen erzeugt werden kann. Um Kollisionen zu vermeiden, verwendet man verschiedene Reduktionsfunktionen, die periodisch angewendet eine eindeutige Zuordnung der Eingangs-Zeichenkette und des Ausgangshashes ermöglichen.

Vorteile:

- effizientere Methode für  $n$ -stellige Zeichenketten da beim Brute-Force-Angriff mit

Schlüsselsuche viele Zeichenketten in Hashes umgewandelt werden, die mit hoher Wahrscheinlichkeit niemals fallen bzw. gewählt werden

- Wörterbuch angriff findet nicht immer ein Passwort, da der Angriff sehr stark von der Passwort Liste abhängt und nicht jede Liste alle Passwörter enthält

### Aufgabe 3

(1)  
128 Ascii zeichen

$$f(1)=128^1$$

$$f(1)=128 = 1.28 \cdot 10^2$$

$$f(2)=16384 = 163.84 \cdot 10^2$$

$$f(3)=2097152 = 20971.52 \cdot 10^2$$

$$f(4)=268435456 = 2684354.56 \cdot 10^2$$

$$f(5)=34359738368 = 343597383.68 \cdot 10^2$$

$$f(6)=4.40 \cdot 10^{12}$$

$$f(7)=5.63 \cdot 10^{14}$$

$$f(8)=7.21 \cdot 10^{16}$$

$$f(9)=9.22 \cdot 10^{18}$$

$$f(10)=1.18 \cdot 10^{21}$$

(2)

$$128 / 10^9 = 1.28 \cdot 10^{-7} \text{ s}$$

$$16384 / 10^9 = 0.000016384 \text{ s}$$

$$2097152 / 10^9 = 0.002097152 \text{ s}$$

$$268435456 / 10^9 = 0.268435456 \text{ s}$$

$$34359738368 / 10^9 = 34.359738368 \text{ s}$$

$$4.40 \cdot 10^{12} / 10^9 = 4400 \text{ s} = 73.33333333333333 \text{ min}$$

$$5.63 \cdot 10^{14} / 10^9 = 563000 \text{ s} = 9383.333333333333333 \text{ min} = 156.388888888888888883$$



h

$7.21 \cdot 10^{16} / 10^9 = 72100000 \text{ s} = 1201666.6666666666666667 \text{ min} =$

$20027.77777777777777777783 \text{ h} = \text{ungefähr } 834 \text{ tage}$

$9.22 \cdot 10^{18} / 10^9 = 9220000000 \text{ s} = 153666666.6666666666666667 \text{ min} =$

$2561111.1111111111111111117 \text{ h} = \text{ungefähr } 106712 \text{ tage}$

$1.18 \cdot 10^{21} / 10^9 = 1180000000000 \text{ s} = 1966666666.666666666666667 \text{ min} =$

$327777777.777777777777777783 \text{ h} = \text{ungefähr } 13657407 \text{ tage}$

(3)

Man benötigt 64 byte um einen SHA-512 Hash zu speichern

=>Base64-Kodierung liefert 22 ASCII-Zeichen => 22byte Speicherplatz

1:  $128 * 22 = 2816 \text{ byte}$

2:  $16384 * 22 = 360448 \text{ byte}$

3:  $2097152 * 22 = 46137344 \text{ byte}$

4:  $268435456 * 22 = 5905580032 \text{ byte}$

5:  $34359738368 * 22 = 7,56 \cdot 10^{11}$

6:  $4.40 \cdot 10^{12} * 22 = 9,68 * 10^{13}$

7:  $5.63 \cdot 10^{14} * 22 = 1,24 \cdot 10^{16}$

8:  $7.21 \cdot 10^{16} * 22 = 1,59 \cdot 10^{18}$

9:  $9.22 \cdot 10^{18} * 22 = 2,03 \cdot 10^{20}$

10:  $1.18 \cdot 10^{21} * 22 = 2,60 \cdot 10^{22}$

(4)

Das Passwort muss lang sein, damit es viele mögliche Passwörter gibt, wodurch es sehr lange dauern würde das Passwort zu hacken.

(5)

Ein Jahr hat 31.536.000 sekunden.

$128^l / 4 \cdot 10^{11} = 31.536.000 \text{ sekunden} \quad | * 4 \cdot 10^{11}$

$128^l = 12614400 * 10^{12}$

$\log_{128}(12614400 * 10^{12}) = l \Rightarrow l = 9,065 \quad (9 \text{ wäre etwas weniger als ein Jahr})$

Ein Windows Passwort sollte mindestens 10 Zeichen lang sein, um sicher zu sein.