0ZAPFTIS - TEIL 2
ANALYSE EINER REGIERUNGS-MALWARE

DREI JAHRE SIND IN DER IT EINE WIRKLICH LANGE ZEIT

Dem Chaos Computer Club (CCC) wurden weitere Exemplare der als Staatstrojaner bekanntgewordenen Schadsoftware zugespielt. Zwei Wochen nach der Veröffentlichung der Analyse des Trojaners aus dem Jahr 2008 möchten wir die bislang modernste bekannte Variante der wohl populärsten staatlichen Schnüffelsoftware des Landes näher vorstellen.

26. Oktober 2011

AUSGANGSPUNKT DER ANALYSE

Die hier analysierte Version 3.6.44 ist gegenüber der drei Jahre alten 3.4.xx-Version weiterentwickelt worden. Das Kommunikationsprotokoll des neuesten Staatstrojaner ist zur alten Command&Control-Software nicht vollständig rückwärtskompatibel.

Nicht erst IT-Spezialexperte und Staatssekretär Ole Schröder, der den abwesenden Bundesinnenminister in der Aktuellen Stunde des Deutschen Bundestages zum Staatstrojaner vertrat, machte deutlich, daß drei Jahre eine lange Zeitspanne für Verbesserungen sind:

„Der Präsident des BKA hat uns heute im Innenausschuss dezidiert erklärt, welche verfahrensrechtlichen Absicherungen im BKA selbst durchgeführt werden. Anders als bei der vom Chaos Computer Club untersuchten Software – diese ist im Übrigen drei Jahre alt; drei Jahre sind in der IT wirklich eine lange Zeit – findet bei der eingesetzten Software des Bundes eine Verschlüsselung in beide Richtungen statt."[1]

In diesen drei Jahren hatten die Verantwortlichen Entwickler in der Tat eine ganze Menge Zeit, die Software zu verbessern. Auch die Ergebnisse der behördeninternen Prüfungen auf Rechtmäßigkeit und Vertraulichkeit der Anwendung hätten in dieser Zeit wohl Durchschlag in der Weiterentwicklung finden können.

Nach den vollmundigen Versprechungen im Bundestag haben wir uns besonders auf zwei Bereiche konzentriert, bei denen signifikante Verbesserungen in Aussicht gestellt wurden: revisionssichere Protokollierung sowie Vertraulichkeit und Sicherheit. Weiterhin ergab sich bei der Analyse noch ein Schwerpunkt bei der stärkeren Verschleierung der „Program Upload & Execution"-Funktion.

REVISIONSSICHERE PROTOKOLLIERUNG

Ole Schröder berichtete aus dem Innenausschuß vom 19. Oktober 2011:

„Durch eine revisionssichere Protokollierung sämtlicher Schritte ist sie [die Software] auch für den zuständigen Richter kontrollierbar. Durch diese Protokollierung ist es auch nicht möglich, mal eben weitere Schadmodule nachzuladen."[2]

Unter revisionssicherer Protokollierung versteht man, daß ein Tatbestand, so wie er sich zugetragen hat, ohne Zeitverzug manipulationssicher protokolliert wird. Üblicherweise fügt man den Daten einen kryptographisch gesichertem Zeitstempel bei, versieht sie mit einer digitalen Signatur, schreibt sie auf ein Write-Once-Medium wie ein MO-Laufwerk oder einen CD/DVD-Rohling oder druckt sie an einer physisch abgesicherten Stelle auf Papier aus. Selbst dieses Vorgehen ist nur unter der Annahme revisionssicher, daß

---

[1] Plenarprotokoll 17/132 des Deutschen Bundestages, 19. Oktober 2011, S. 15604, http://www.bundestag.de/dokumente/protokolle/plenarprotokolle/17132.pdf

[2] Ebd.

sich die Hardware vollständig unter der eigenen Kontrolle befindet und die Software nicht kompromittiert wurde.

Im Fall eines Trojaners läuft nun nicht nur der Trojaner selbst in einem fremden Rechner außerhalb der Kontrolle des Protokollanten, auch die Leitung zu diesem Rechner ist unter fremder Kontrolle. Wir haben bereits im ersten Bericht zum Staatstrojaner[3] gezeigt, daß dieses Protokoll Manipulationen auf der Leitung nicht verhindern oder erkennen kann.

Im ersten Bericht zum Staatstrojaner haben wir gezeigt, daß wir eine eigene Kontrollsoftware schreiben können, die der Trojaner nicht von der echten unterscheiden kann.[4] Der vorliegende Bericht zeigt, daß wir auch einen eigenen Gegen-Trojaner entwickelt haben, der für die Kontrollsoftware nicht vom echten zu unterscheiden ist und der manipulierte Daten hochladen kann. [5] Wir erklären außerdem, daß es nicht nur praktisch, sondern sogar theoretisch unmöglich ist, Trojaner (oder Protokolle) zu ersinnen, die nicht durch Trojaner-Imitate ersetzt werden können.

Da wir eine solche revisionssichere Protokollierung in der alten Version des Trojaners nicht identifizieren konnten, nahmen wir an, daß wir sie in der aktuellsten Version schlicht übersehen hatten. Jedoch: Es ist unwahrscheinlich, daß es eine Protokollierung in dieser Form gibt. Denn die Protokollierung müßte auf dem Computer vorgenommen werden, der aktiv belauscht wird, da das Protokoll auf der Leitung ja keine Sicherheit vor Manipulation bietet. Das wiederum widerspricht aber der Vorgabe, daß keine weiteren Daten auf dem Rechner modifiziert werden dürfen. Eine Protokollierung auf Seiten der Überwachungsbehörden ist jedoch ungefähr so sinnvoll wie ein Angeklagter, der sein eigener Richter ist, oder etwa (wie ein jüngstes Beispiel zeigt) ein Dr. Hans-Peter Uhl, welcher das Protokoll einer live übertragenen Bundestagssitzung wegen eines sprachlichen Mißgriffes inhaltlich modifizieren läßt.

Eine revisionssichere Protokollierung der Aktivitäten des Staatstrojaners kann auschließlich durch hardware-seitige Unterstützung sichergestellt werden, damit unwiderruflich und unveränderbar die Protokolle dort erfaßt werden, wo sie entstehen: auf Seiten des Opfers.

Wir haben bereits *gezeigt*, daß wir vorbei an jeglicher revisionssicheren Protokollierung beliebige Daten auf dem Zielcomputer modifizieren und ausführen konnten, und wir werden dies gern jederzeit erneut unter Beweis stellen. Da ein hardware-basiertes Protokollierungswerkzeug auf Seiten des Zielcomputers durchaus auffällig und somit unpraktisch wäre, kommen wir zu dem Schluß, daß die „revisionssichere Protokollierung" in diesem Kontext nur eine leere Phrase bleiben wird. Diese Protokollierung dient ausschließlich als Alibifunktion in Diskussionen um die Rechtsmäßigkeit des Trojaners. Ohnehin sind weder genaue Angaben noch Beweise für eine revisionssichere Protokollierung vorgelegt worden.

---

[3] Analyse einer Regierungs-Malware, http://www.ccc.de/system/uploads/76/original/staatstrojaner-report23.pdf

[4] http://haha.kaputte.li/staatstrojaner_720p.mp4

[5] http://haha.kaputte.li/0zapftis-2_922x578-final.mov

Über die Art der „Protokollierung" auf Seiten der Behörden können an dieser Stelle nur Mutmaßungen gemacht werden. Auf Seiten des Trojaners sind jedoch offensichtlich keine Protokollmechanismen vorhanden.

VERTRAULICHKEIT UND SICHERHEIT

Wenn informationstechnisch mental unbewaffnete Politiker über Kryptographie reden, geben sie sich offenbar schon damit zufrieden, daß der Begriff „Kryptographie" an sich schon hinreichend kryptisch klingt. Selbst eiligst hinzugezogene und speziell geschulte Kollegen aus dem „BKA Labor" lassen schon mal das Halteproblem[6] links liegen, um einen überzeugend klingenden Sprechzettel für die Debatte zu liefern: Es existiere nun eine bi-direk tionale Verschlüsselung, ist die neue Sprachregelung, und somit seien die Aussagen des CCC insgesamt vollständig haltlos.

Im Prinzip ist diese Aussage richtig, es gibt im Wintermodell 2010 des Digi-Task-Trojaners tatsächlich eine Verschlüsselung der vom Kontrollsystem zum Trojaner gesendeten Kommandos. Aber: Der nunmehr als allgemein bekannt geltende und in allen vom CCC untersuchten Trojaner-Varianten immer gleiche AES-Schlüssel hat sich mindestens in den vergangenen drei Jahren nicht geändert, und es wird immer noch AES im untauglichen ECB-Verschlüsselungsmodus verwendet. Damit wäre es wohl technisch korrekter von einer „beidseitigen Verschleierung" zu reden.

Die ebenfalls großspurig angekündigte neue Authentisierung von Steuer-Kommandos stellt sich praktisch als Einfügen einer konstante Zahlenkolonne heraus, die sich dank des schlechten Verschlüsselungsverfahrens in der „verschlüsselten" Version ebenfalls auf eine konstante Zahlenkolonne abbildet. Ein Angreifer muß daher noch nicht einmal den Kanal entschlüsseln können, um die „Authentisierung" austricksen zu können. Ein sogenannter Replay-Angriff ist einfach möglich: Der Unbefugte muß nur die relevanten Pakete aufzeichnen und zu passender Zeit zum Trojaner senden.

Ist er aber im Besitz des nicht besonders geheimen und nie veränderten AES-Schlüssels, kann er schnell sehen, daß sich die Command&Control-Seite jetzt beim Trojaner zu erkennen geben muß, indem sie jedes Kommando mit dem folgenden konstanten Token anreichert:

```
688583950000000000000000000000000
d4fa94b800000000000000000000000000
0e840282000000000000000000000000000
81f3b5830000000000000000000000000
```

Die grundsätzliche Idee, eine weitere Autorisierung für Steuerkommandos zu erwarten, ist ja erst einmal nicht verkehrt. Die Umsetzung hat jedoch nichts mit industrieüblichen Verfahren zu tun, oder um ein großes deutsches Boulevardblatt zu zitieren: „weniger Sicherheit als beim Flirtforum".

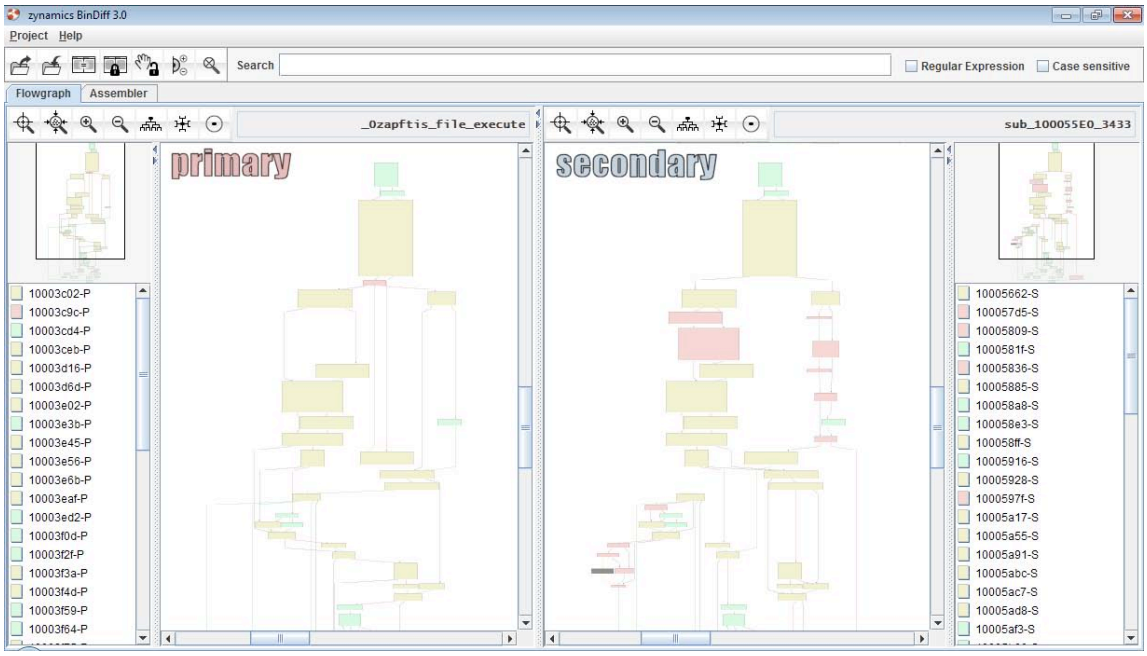
---

[6] http://de.wikipedia.org/wiki/Halteproblem

EINSENDUNG WEITERER ANALYSE-ERGEBNISSE

Das als „Update-Funktion“ schöngeredete Hochladen und Ausführen belie biger Schadsoftware wurde – interessanterweise im Kontrast zu ihrer nachdrücklich behaupteten Rechtmäßigkeit – gegenüber der drei Jahre alten Version noch weiter verschleiert. Es stellt sich schon die Frage, wer hier offenbar doch etwas zu verbergen hat.

Die folgende Abbildung illustriert die Unterschiede der Funktion `_0zapftis_file_execute` der neuen und alten Version der analysierten Schadsoftware.



Die im Bild rechts rot markierten Blöcke sind im Binary vom Dezember 2010 neu hinzugekommen. Die Illustration visualisiert, wie sehr sich die beiden Funktionen voneinander unterscheiden. Faßt man die wesentlichen Änderungen zusammen, ergibt sich folgendes Bild:

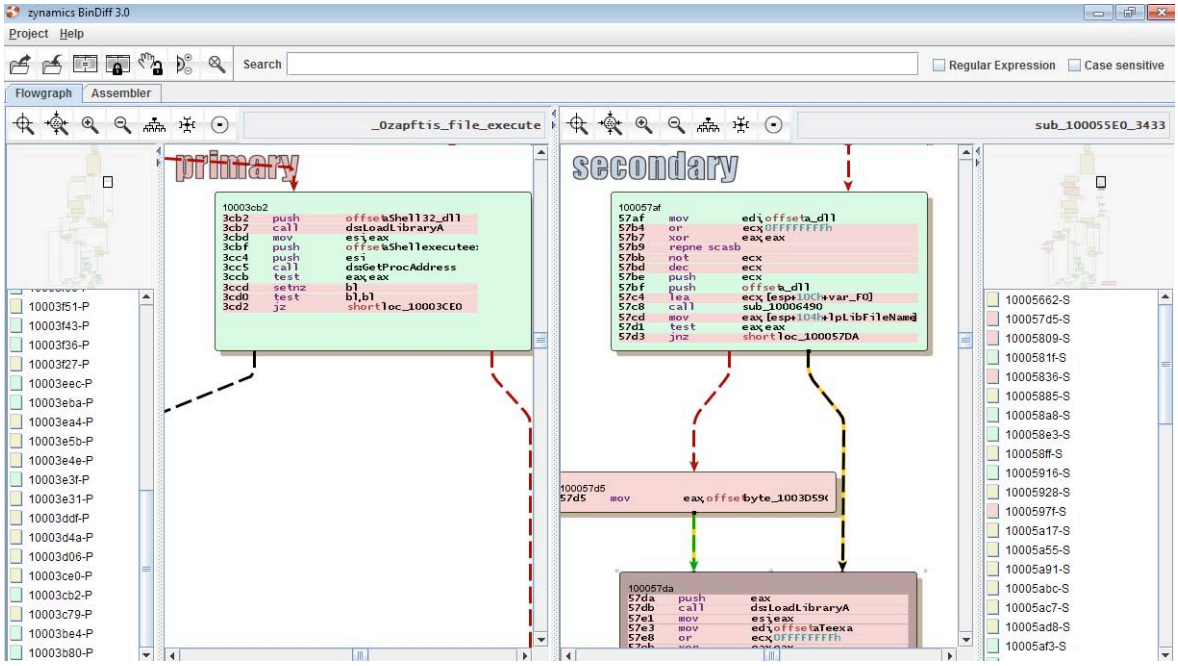
Die neuere Version des Codes zerhäckselt das `ShellExecuteExA` auf die gleiche Art und Weise wie die drei Jahre ältere Version schon das `CreateProcessA`.[7] Auch der Name der DLL, die dort zur Laufzeit geladen werden soll, wird auf die gleiche Weise verschleiert. Sowohl Name der DLL als auch der Name der Funktion werden als ASCII-Zeichenketten an bestimmte Windows-APIs übergeben, um jeweils Handles auf die entsprechenden Ressourcen (DLL oder eben einen Function Pointer) zu erlangen.

[7] ShellExecuteExA() ist in Shell32.dll implementiert,
http://msdn.microsoft.com/en-us/library/bb762154(v=VS.85).aspx
CreateProcessA() ist in Kernel32.dll implementiert,
http://msdn.microsoft.com/en-us/library/ms682425(VS.85).aspx

Wir finden in der besagten Funktion:

```
...
push    offset aShel    ; "Shel"
...
push    offset aShe     ; "she"
...
push    offset aCrea    ; "Crea"
...
push    offset aLl32    ; "ll32"
...
push    offset aLexecu  ; "lExecu"
...
push    offset a_dll    ; ".dll"
...
call    ds:LoadLibraryA     ; <- shell32.dll
mov     esi, eax            ; <- Handle auf shell32.dll
...
push    offset aTeexa   ; "teExA"
...
push    eax             ; lpProcName   <- "ShellExecuteExA"
push    esi             ; hModule      <- shell32.dll
call    ds:GetProcAddress
...
call    eax              ; <- call ShellExecuteExA() !!
...
mov     esi, offset aTeproc ; "teProc"
...
push    offset aKernel32_0 ; "Kernel32"
call    ds:GetModuleHandleA
...
mov     esi, offset aEssa ; "essA"
...
push    eax             ; lpProcName   "CreateProcessA"
push    ebx             ; hModule      <- kernel32.dll handle
call    ds:GetProcAddress
...
call    eax              ; <- call CreateProcessA() !!
...
```

Der Chaos Computer Club hat sein Command&Control-Werkzeug innerhalb von Stunden an die neuen Gegebenheiten (Protokoll, Crypto, Upload) angepaßt, damit eine unbefugte und unprotokollierte Programmausführung dokumentiert werden kann. Hier ist ein Bild des Funktionsflusses zu sehen, das die Unterschiede übersichtlich darstellt (links alt, rechts neu):

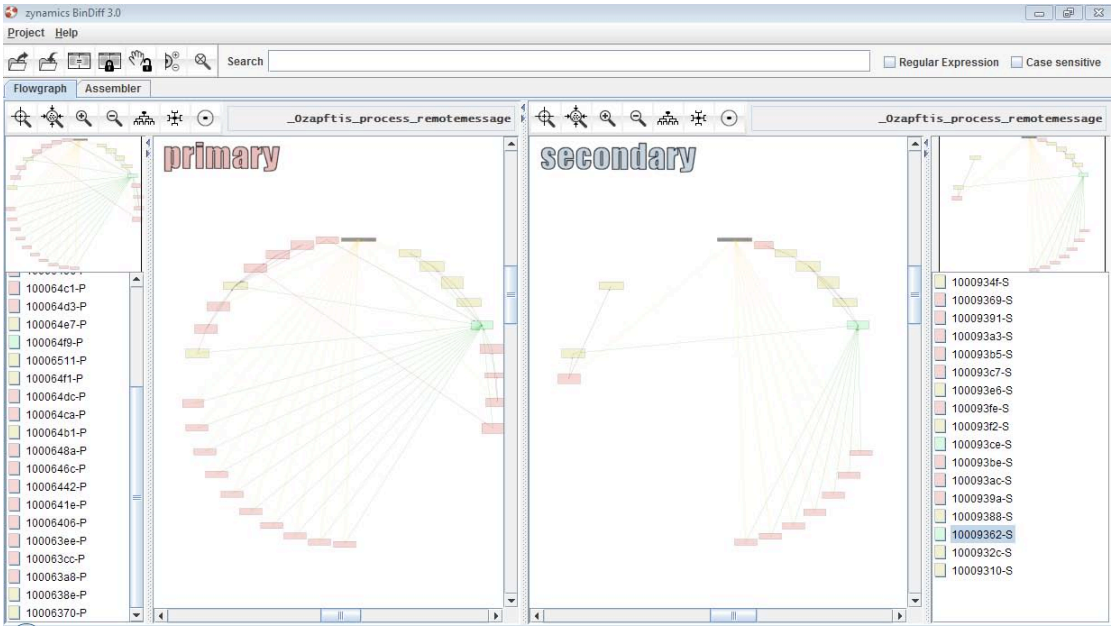


Q.e.d.

WEITERE NENNENSWERTE ÄNDERUNGEN

Die besonders brisanten Themen wurden in der aktuellen Trojanerversion etwas detaillierter untersucht und oben beschrieben. Anders als die Anti-Virus-Industrie interessierte uns in erster Linie der programmierte Verfassungsbruch dieser Anwendung. Weniger brisant, aber dennoch erwähnenswert ist es, daß die Struktur einen geringfügig aufgeräumteren Eindruck macht. Nicht, daß wir behaupten würden, es seien hinsichtlich der vergangenen Zeitspanne und geflossenen Gelder angemessene Änderungen, aber wir wollen ja auch nicht ein Qualitätsgutachten für die vorliegenden Schadsoftware erstellen. Vielleicht haben die DigiTask-Experten auch einfach eine neuere Compilerversion und einen Optimizer benutzt.

Ausgangspunkt für die Untersuchung der Unterschiede ist und war natürlich der Command Dispatcher, dessen Unterschiede man anhand der verschiedenen Graphen darstellen kann:



In der neuen Version stehen nicht mehr 17, sondern weit weniger Kommandos zur Verfügung. Stichprobenhaft ziehen wir die Kommandos mit den Nummern 4 und 14 heran. Die 4 der alten Version führte zu einer Registrierung des Kernel-Moduls auf dem Zielsystem, die 14 führte zu Program Upload & Execution. Die weiteren Kommandos der neuen Trojaner-Version wurden noch nicht im Detail untersucht, daher kann nicht ausgeschlossen werden, daß sich nicht noch weitere brisante Funktionalitäten eingeschlichen haben.

Alt:

command 4 (register kernel module)

```
.text:100063EE                          cmd_disp_case4:
; CODE XREF: _0zapftis_process_remotemessage+31j

.text:100063EE
; DATA XREF: .text:__0zapftis_init_cmd_dispatchero

.text:100063EE 014 8B 16                mov     edx, [esi]
; jumptable 100063A1 case 4

.text:100063F0 014 8B CE                mov     ecx, esi

.text:100063F2 014 FF 52 1C             call    dword ptr [edx+1Ch]

.text:100063F2

.text:100063F5 014 E9 FF 00 00 00       jmp     loc_100064F9
```

command 14 (file upload & exec)

```
.text:100064C1

.text:100064C1                          cmd_disp_case14:
; CODE XREF: _0zapftis_process_remotemessage+31j

.text:100064C1
; DATA XREF: .text:__0zapftis_init_cmd_dispatchero

.text:100064C1 014 8B 16                 mov     edx, [esi]
; jumptable 100063A1 case 14

.text:100064C3 014 8B CE                 mov     ecx, esi

.text:100064C5 014 FF 52 2C              call    dword ptr
[edx+2Ch]

.text:100064C5

.text:100064C8 014 EB 2F                 jmp     short loc_100064F9
```

Neu:

command 4 (register kernel module)

```
.text:10009369 008 68 F2 BC 72 84          push    8472BCF2h
; jumptable 10009362 case 4

.text:1000936E 00C 68 F1 49 AD BA          push    0BAAD49F1h

.text:10009373 010 68 FD BA 74 84          push    8474BAFDh

.text:10009378 014 68 F7 48 AF DE          push    0DEAF48F7h

.text:1000937D 018 8B CE                   mov     ecx, esi

.text:1000937F 018 E8 FC FE FF FF          call    sub_10009280

.text:10009384 008 84 C0                   test    al, al

.text:10009386 008 74 6A             jz      short loc_100093F2

.text:10009388 008 8B 16             mov     edx, [esi]

.text:1000938A 008 8B CE             mov     ecx, esi

.text:1000938C 008 FF 52 0C          call    dword ptr [edx+0Ch]

.text:1000938F 008 EB 3D             jmp     short loc_100093CE
```

command 14 (file upload & exec)

```
.text:100093B5                       loc_100093B5:
; CODE XREF: sub_10009310+52j

.text:100093B5
; DATA XREF: .text:off_1000940Co

.text:100093B5 008 8B 06                                mov
eax, [esi]      ; jumptable 10009362 case 14

.text:100093B7 008 8B CE             mov     ecx, esi

.text:100093B9 008 FF 50 1C          call    dword ptr [eax+1Ch]

.text:100093BC 008 EB 10             jmp     short loc_100093CE
```

Offensichtliche Parallelen:

Command	**Alt**		**Neu**	
cmd 4	call	dword ptr [edx+1Ch]	call	dword ptr [edx+0Ch]
cmd 14	call	dword ptr [edx+2Ch]	call	dword ptr [eax+1Ch]

Die neuen Offsets aller Kommandos in call [REG + offset] sind immer (offset - 0x10), scheinen also eine ähnliche Struktur bzw. Codebasis aufzuweisen.

Weitere Verschleierungsmaßnahmen:

Auffälligstes Merkmal der neuen Command-Dispatcher-Routine sind die folgenden Code Listings. Unmittelbar vor dem Command switch statement findet sich folgender Code:

```
.text:1000932C                          loc_1000932C:
; CODE XREF: _0zapftis_process_remotemessage+D0j
.text:1000932C 008 68 81 F3 B5 83            push    83B5F381h
.text:10009331 00C 68 0E 84 02 82            push    8202840Eh
.text:10009336 010 68 D4 FA 94 B8            push    0B894FAD4h
.text:1000933B 014 68 68 85 83 95            push    95838568h
.text:10009340 018 8B CE                     mov     ecx, esi
.text:10009342 018 E8 39 FF FF FF            call    _new_0zapf-
tis_checkauth
```

An der Stelle, wo Kommando 4 abgearbeitet werden soll, findet sich eine ähnliche Überprüfung:

```
.text:10009362 008 FF 24 85 0C 94 00+       jmp
ds:off_1000940C[eax*4] ; switch jump

.text:10009362 008 10

.text:10009369

.text:10009369                          loc_10009369:
; DATA XREF: .text:off_1000940Co

.text:10009369 008 68 F2 BC 72 84                   push
8472BCF2h       ; jumptable 10009362 case 4

.text:1000936E 00C 68 F1 49 AD BA           push    0BAAD49F1h

.text:10009373 010 68 FD BA 74 84           push    8474BAFDh

.text:10009378 014 68 F7 48 AF DE           push    0DEAF48F7h

.text:1000937D 018 8B CE                    mov     ecx, esi

.text:1000937F 018 E8 FC FE FF FF           call    _new_0zapf-
tis_checkauth

.text:1000937F

.text:10009384 008 84 C0                    test    al, al

.text:10009386 008 74 6A                    jz      short
__cmd_dispatch_error

.text:10009386

.text:10009388 008 8B 16                    mov     edx, [esi]

.text:1000938A 008 8B CE                    mov     ecx, esi

.text:1000938C 008 FF 52 0C                 call    dword ptr
[edx+0Ch] ; -> _new_0zapftis_register_rootkit()

.text:1000938C

.text:1000938F 008 EB 3D                     jmp     short
loc_100093CE
```

Leicht ersichtlich im Pseudocode der Funktion `_new_0zapftis_checkauth()` ist, wie ein C+C-Teilnehmer sich am Trojaner authentifizieren muß, um beliebigen Code auf dem Zielrechner ausführen zu können:

```
bool new_0zapftis_checkauth(int a1, int a2, int a3, int a4)

{
  unsigned char buf[4];

  int v6;

  int v7;

  int v8;

  bool result;

  result = new_0zapftis_recveive_and_decrypt_DWORD(buf) == 4

      && *(_DWORD *)buf == a1

      && new_0zapftis_recveive_and_decrypt_DWORD((unsigned char
      *)&v6) == 4

      && v6 == a2

      && new_0zapftis_recveive_and_decrypt_DWORD((unsigned char
      *)&v7) == 4

      && v7 == a3

      && new_0zapftis_recveive_and_decrypt_DWORD((unsigned char
      *)&v8) == 4

      && v8 == a4;

   return result;

}
```

Die Entschlüsselung der Daten in diesem nun beidseitig „verschlüsselten" Protokoll übernimmt jeweils die entsprechende Funktion, welche ebenfalls die Daten vom Netzwerk liest. Wie bereits beschrieben, kommt hier das gleiche kaputte Verfahren (AES-ECB) mit statischem „geheim"-Schlüssel zum Einsatz. Es ist der gleiche Key, der in den drei Jahre alten Versionen ebenso verwendet wurde.

Durch diese neuen „Schutzmechanismen" hat sich das Protokoll natürlich grundlegend geändert. Die Anwendung erwartet nun die Eingaben/Befehle AES-verschlüsselt, wobei das Kommando als erstes Datenpaket mit 16 Byte Länge geschickt werden muß. Nach dem Befehl wird die oben gelistete Bytesequenz erwartet, mit der nun weiter der Command Dispatcher abgearbeitet werden kann. Dieser Bytesequenz folgen nun die Kommando-spezifischen Parameter, wie wir es aus der alten Version des Trojaners kennen. Also beispielsweise ein Längenfeld für die Anzahl der Bytes in Parametern für einen neuen Prozeß, Parameter als ASCII-Zeichenkette, Größe der ausführbaren Datei, die übertragen werden soll, und die Daten der ausführbaren Datei.

Auffällig in der aktuellsten Version des Trojaners ist zudem, daß hier weitere Kommandos aus der Command-Dispatcher-Funktion herausgenommen wurden: Die Screenshot-Funktionen und die Funktion zum Auslesen der Softwa-

reliste sind nicht mehr über das Standard-Kommando-Interface erreichbar. Dies bedeutet aber nicht, daß der Trojaner hinsichtlich dieser Funktionen entschärft wäre: Ähnlich wie mit den ebenfalls in der älteren Version entdeckten Routinen für den Großen Lauschangriff, befinden sich die hier deaktivierten Screenshot-Routinen weiterhin im Binary – es existiert lediglich kein direkter Call-Pfad.

Letztlich spielt es keine Rolle, welche Funktionalität der Trojaner direkt zur Verfügung stellt, solange jede gewünschte Funktionalität über die Nachlade-Funktion nachträglich installiert werden kann.

DEMONSTRATION

Die Demonstration wurde in einem Screencast-Video festgehalten und räumt mit zwei großen Falschdarstellungen seitens der Trojaner-Fürsprecher auf:

1.) Die neue Version des Trojaners sei sicher und vertrauenswürdig.

2.) Die angefallenen Daten seien authentisch und werden revisionssicher protokolliert.

Wie bereits in der ersten Veröffentlichung gezeigt und dokumentiert, bringen wir erneut ein eigenes Programm an den Ermittlungsbeamten vorbei auf den Computer des Beschuldigten und führen dies aus. Zur Demonstration wird das Programm calc.exe – ein Taschenrechner – verwendet. Hierbei könnte es sich jedoch um beliebigen anderen Schadcode handeln, der beispielsweise gefälschte Beweise auf dem Computer des Opfers hinterlegt.

Der CCC demonstriert nun exakt die gleiche Schwäche durch „Program Upload & Execute" auf der aktuellen Version vom Dezember 2010. Hier mußten lediglich geringe Modifikationen am Kommunikationsprotokoll vorgenommen werden, da nun beide Richtungen verschlüsselt und eine weitere (schwache) Prüfung der Authentität durchgeführt wird.

Diesmal verwendet der CCC erneut die selbstgeschriebene C+C-Software für den 0zapftis-Staatstrojaner und das Programm calc.exe, welches zur Ausführung gebracht wird. Das Zielsystem befindet sich diesmal in einer virtuellen Maschine, um den Sachverhalt übersichtlicher darstellen zu können.

Die zweite Demonstration befaßt sich mit einem Thema, welches in der Praxis bei der ersten Veröffentlichung noch nicht aufgegriffen wurde: das Fälschen von Beweisen über das Netzwerk, das Infiltrieren der sog. revisionssicheren Protokollierung. Da die offizielle C+C-Seite keinerlei Authentisierungs-Checks durchführt, die auch nur annähernd zeitgemäß wären, kann ein beliebiger Computer über das Internet (zum Beispiel auch über das Tor-Netzwerk) einen authentischen Trojaner simulieren bzw. imitieren.

Unser Fake-Trojaner ist in der Lage, sowohl das alte als auch das neue Kommunikationsprotokoll zu sprechen. Er liefert zu Demonstrationszwecken laufend neue Bilder als Screen- oder Applicationshots. Dieses Vorgehen beweist, daß per Trojaner erlangte Screenshots (und andere „Beweise") gene-

rell als gefälscht anzusehen sind und keinerlei Beweiskraft haben, weder in die eine noch in die andere Richtung.

CROWDSOURCING

Der CCC veröffentlicht nun die IDA-Datenbanken der analysierten Binaries. Es wurden überwiegend die Teile analysiert, welche für die Öffentlichkeit aktuell relevant waren. Sehr große Teile der in C++ entwickelten Anwendung sind bislang noch nicht Ziel der Analyse gewesen, teilweise werden sich auch Bezeichner in der IDB finden, die nicht 100% korrekt sind. Unkorrekte Bezeichner kommen vor allem aus der Anfangsphase der Analyse, in der es nur sehr wenige bis gar keine Anhaltspunkte über den Programmfluß gab. Das Gesamtbild wächst und wächst, zuvor erlangte Erkenntnisse müssen immer wieder neu angepaßt und korrigiert werden. Daher empfehlen wir, die durch uns erstellten IDA-Datenbanken mit einer gewissen Skepsis zu betrachten, wir sind auch nicht unfehlbar.

Die Zeit spielte gegen uns. Daher konnten leider zahlreiche andere, interessante Fährten nicht aufgenommen werden, zum Beispiel die Verwendung der anfangs dokumentierten Implementierung der Wave-Device-Capture-Routine. Was hier fehlt, ist der Call-Path zu dieser Funktion. Können wir irgendwie zeigen, daß der hier augenscheinlich tote Code doch auf irgendeine Weise getriggert werden kann, wäre dies der Beweis für ein Werkzeug, welches einen Großen Lauschangriff, also auch eine akustische Raumüberwachung zuließe. Die Skype-API-Nutzung durch den Trojaner haben wir ebenfalls wissentlich vernachlässigt, da dies ja ganz klar zum Leistungsspektrum des Herstellers gehört und die eigentliche Legitimation der „Quellen-TKÜ" darstellt.

Ferner hatten wir bei der ersten Veröffentlichung aus zeitlichen und rechtlichen Erwägungen von einer näheren Inspektion des in einem Rechenzentrum in den USA befindlichen C+C-Servers/Proxies abgesehen. Offensichtlich wurde durch interessierte Lesern aber auch dort eine weitere grobe Fahrlässigkeit festgestellt. Der Server war noch tagelang aktiv und mittels einer Plesk-Administrationsoberfläche mit einem Stand aus dem Jahr 2009 erreichbar. Seit 2009 sind diverse bekannte Sicherheitslücken in Plesk gefunden und behoben worden. Schon die Erreichbarkeit so einer unsicheren Software aus dem Internet macht sämtliche über diesen Rechner geleiteten Daten per se vertrauensunwürdig.

Daher möchten wir gern die interessierte Öffentlichkeit aufrufen, sich der Sache anzunehmen und unsere Arbeit gemeinam mit uns fortzusetzen. Wir sammeln und konsolidieren gern alle Erkenntnisse und werden alle Einreichungen entsprechend würdigen und veröffentlichen, sofern gewünscht.

Fundstücke und weitere Aufklärungsergebnisse nehmen wir gern unter `0zapftis@ccc.de` entgegen. Bevor ihr uns meldet, daß Euer Antivirusprogramm angeschlagen hat, bitte überprüfen, daß es sich nicht einfach um die Detektion des hier heruntergeladenen Samples handelt.

Fingerprint: 62D7 AAB0 7A2F 2ABD 2899 91E3 F06D 0BAB 93C1 30A0

-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: GPGTools - http://gpgtools.org

mQINBE6OIJYBEADA8V/CA60MHsizwIEk46q3Tw2/DceWdN5jpqr8xD00vhjLMjBx
kFgbZdou6yrYnZbrTC72dQbqj/e0KJaj5gmDjzEb29GKxFRbZkhjMSxYPBb4rawJ
MRQdv/o/Olsf7ucLCEMRjuNxxczpo5dayDZC1yT4P/PcERscOM1RlOkM+laqde4v
ApEZavNMrXBlV/s/cQ6gMnzqyzv9dNRaUN8BbNWufWmvue22DUR2kUpsEWYfXBe6
o70k8nxe91uHBDnfjL12n2E7kI79+umniOdXYPQgfzBLTnAgCjHjt+Xy75LOiYXt
ea7KPaGZoe9RuV+gAcK0G+NElDF7PjeuHbsV3YLXuQ7wjmbsn6qjpxI2E6C+vY30
29+4Si7FgwKLlJ/NVrAg90OGEQ13BvPGFUq5rES/ILs31fa7jclXKaF+ADcMs9o3
ymXQF/wU1ENyUMtLsEz9DZ8yKgLVmLlieVmaiMPaJXSFYyHTccJoJ48QfYQARuMa
OR+bMhW/wWoublKgj1tL35GF9fJ0hYpwtMG+Xfyi/JG8fJHV8J01sKG5w/UaBAY1
T4quFlHcdMjoRXwtExCsDjyqHRJAakL4WZEjulb3ReGVfuk+pVXTG4Hsp3E8oVKT
v62ahMg7X5ugek232DwUTzfU77sNkcTiuXokPMswbElfp5zmm9pUhalcnQARAQAB
tDcwlFphcGZ0IElzISBSZXZlcnNlIEVuZ2luZWVyaW5nIElucHV0IDwwemFwZnRp
c0BjY2MuZGU+iQI+BBMBAgAoBQJOjiCWAhsvBQkHhh+ABgsJCAcDAgYVCAIJCgsE
FgIDAQIeAQIXgAAKCRDwbQurk8EwoEHBD/4vkbPzdBw9Ra7IBJCFe6aTUlw4qskU
WM+2hyC06wOWgZM8KiGABFabInJ+2krc+humAuRJoZPySoHyOi/QY9ND033FgkhX
Vea9EJpZRm0tJmbFMlFLzwT9fZ5r7GL0xLrQKoMEK3vUd0b3xQBqeaFEpB+VfU8Q
vKmgTG4dO8pYKVe3/MnjAkS6fUUFOsl9QvHCW8+u2Qn4fl7mUygBTLfK4y2KDruh
rh3EjeSuSaMdkNlXLDyEI5Hxttn0fTDp8K2Sh15qaeR1uMrwtxPHZRuSUw7jZ7xH
24eUjJ4ipnwLMqeTNiL5JBwzQIRp9pb1cjiNuhxUCT4tGBPTeHgPR2MeEbBfFuYJ
JnSEEO8VRStZXWWAKHj1ku8+YQ90SmFloRAbjlrkZpJn+vrj66wyGyzVbe1bD3Gy
jokwVEhcierUaSpUq9ChBIB+vbMQZlchUfIGZPln/WOuwSgo2L6CNTfcA/6FvHYu
+2Mg5VoeHOxQm28ZXjqCsODx3+j476S9VlHIDsBRmqPbOUoEzY2VIAyDzTlQE5Kn
kBGp8FXk68QYVSS+ZI00cLtoDZlDD22scjn6qDk1y6oHuUnP9UoIF8t2kR1j9xG3
FKrSNufwivgkZ3Fr2n+s9jYMom8YfZi15coNntyYSo7WQOgA29Ssfu8dfG56cdUc
WPrcjLfEcjvPMrkCDQROjiCWARAA4dsiBRvVSRN0YFW8iYJNqH0jzu/CwbjsQAOt
N6xM8OrjEsu3y82q0g/NryJJ4cVq3kl4r+WDoCwD2wR+oT4oMmg5jWtrs8lSikaG
6Gl1W8e81zkyvDol8+BQLFEDxyyOZ313rQznP8RsBzk8u8x1YBPNyeHEJMF3dusm
HUgQW2DY/eUUZQJARb9CHp2DTduTlQbkTPeDnFm6lrvduJyee98QeP+nCw9vTok0
uWc5o9p4VgY+koX10E++iFRlz9rwNzFT2vHPm5MeG1ZlTbWjS17ZQNHsgbOdMDUk
08zQOYl29N4IuXRD1zRhs96oDwuxlo1rUE7A8vtf/6S6RETxkS7ykH1csCWrw6s/
CjaioVVoyWlEvCzn60P8kUCsLJCiXTE4rcdaY9eysM1jeNC2t7BpmuY6gSqBwM5m
0VfL86mClZcE0AaxtrifjjwG8hlnlodyD0Ugi9tO87rPq204wplTMm6iZblKya5a
vzQdKckXOXd4DBSB1fKoZBWAFIuZ2asHa57CsZLXAsM1a1b/eGUlilBN6/bXboum
Gv2q+yF91kEP4tv+5fWLRJOgyUOiljB4g0JN1n9JfnM2iHaX7wcFh+jCnpX+1xZJ
E8urrUEPpt4IOCHB/mJAk2rCrCY69WWJTjuaXlc178TioWDWr+eDuDyENa9pbTCx
5paebg8AEQEAAYkERAQYAQIADwUCTo4glglbLgUJB4YfgAIpCRDwbQurk8EwoMFd
IAQZAQIABgUCTo4glgAKCRBebJ87j17H3iJID/9UVR9HlxmQtQPADWXZxjnNePw1
32O1+Syd9j9JgYczHooudki6mx55ydFHEyu4oDJ7az0UiV92GEl7XV3iwBppf9Ja
WvZ5WvWMX4F/ZmmDWENXqQeniqIUlKa9XmA9jhYAEwy7198pbD/qsGBMioDVai0f
GTYUiBwHt2spu1uopx30spK/RwBKvbH6cbtGmOvfpXmgsFagtg6kPZPbfGZ3Iumh
yWHP4zd/+VcAOkjJv844Nuloh4VMPfwilnakG9bZzg4Ky5kGqB+Vl2WCZSOiVVGo
C4JmdMO7IMkBPMRNXQw23rhWWJVjsnF+nT/TnDlnH7g067IZ5YOZftwSun78Cjb1
sRmwCaj9iNbTwEUnES8Clni/AAirYvXs0Isu67WN1lJWUSwAavs6Thswhvpnrsq7
v0svvtmOU1pZVmYGmOFn8xAC+iK1PHFL4BH8NEhkiCMqBfH0pGljl/hZk60r6Gtf
BNB0Fjt/HOQYVHNaQvbpPhWCLYxeVEUfMk9rRE1FlyzYGhBa/pG5ECoJGNQGriGC
bB4hzSmwjVqaP7N3qzfeP6xQT4y5A7Xe2zN9FnOO8+vjQ6hMMX4Ch4YDaxuHS3C7
4eQTgmJ9CWERuUBz/AdEobY+sakH+2PHN2eBgwbLBX5ti3YKy1L7DE3EZibKwWm5
D4C9KHCwUpT/unjQ9gM9EACHlFOBbxZF/2o4w6VdrsYYBUcihaEtDMc9sywNTwBF
jsxbJM4GHvtwlJlunMp1Iz62f9dL+hAUe76UCq2i7W9eVlT8Pp3xR0+z2Ini1PbG
nfgpsbI9q0ncUlGyo+o/fVNASQqqvfGsfU6SuKapwvMdqK6p7G4y+1XodRHChzli

v9WV2GRXNSp5jTuU7FZzCUaHillU1Xh9P2eo/5/QwTvxkHdEssbCtK6hsWNS/ot9
9IRRB5x3Sr2pnb8JiiZrvwh2YlqaD0cs0gViA5gZXTsOVb6IzcaMgnG4M4xu+fgz
U+G9jHbwWj9UHcEUPEl5rRmrMTpeu5f2xRZddlbDW1QKREATXZkROsP3GLmXMESe
af7BF3+JtUTajYjxQxYwW6hQLkGc4wsIO4nWDFbk/lBh9T25mzTpo54WblDEq9yQ
K83zwI2BC3NFqRoZ9IrC3wJsic5T0/bIKALFXo5quK4pE7xt2+c6VQosymeWBk5q
Exy6jS1C6RjZGF4qXVPznejivZ9jEv4xUh+BXSMKnZCN9SZIzhWU3K6aqacY8LVm
mWE4MA8GJ0dUiw+egWacFBJFRg1I6p1NbuUllU1WdGne2hyz7djbFofLay15x1Lo
wYTTAi2gmp8vxHoZoI30dCJZTtVKb1vIEOE9Tz5Cl/UOVMxtqANGr9/GdVLPY2NB
ZQ==
=jS/I
-----END PGP PUBLIC KEY BLOCK-----