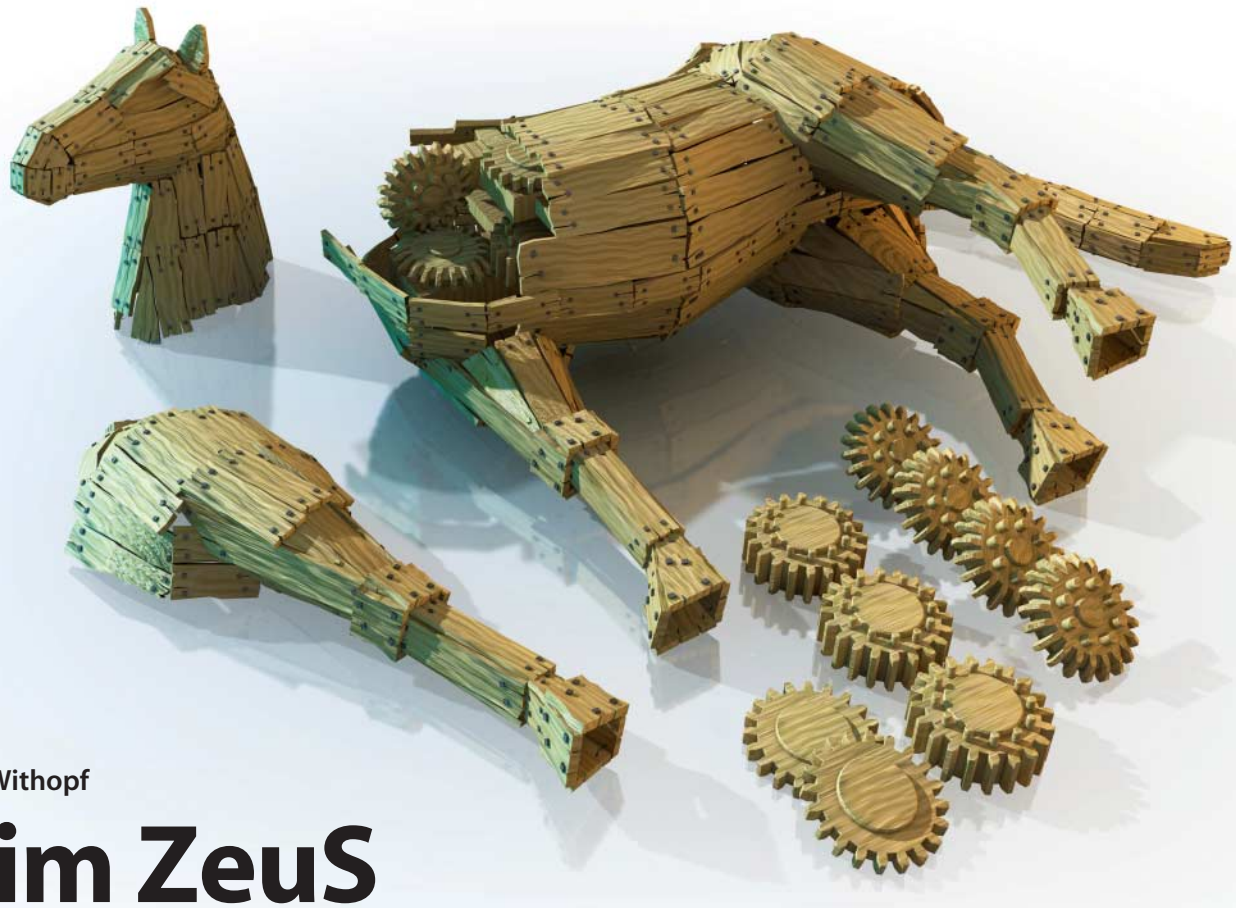


Matthias Withopf

Beim ZeuS

Einblicke in den berühmt-berüchtigten Banking-Trojaner

Er stiehlt Zugangsdaten und TAN-Listen, manipuliert Banking-Webseiten und wartet ständig auf neue Aufträge seines Gebieters. ZeuS ist ein Schädling nach dem Baukastenprinzip und gilt als einer der gefährlichsten Online-Banking-Trojaner. Vor Kurzem ist sein Quellcode im Netz aufgetaucht, was tiefe Einblicke in die Trojaner-DNA erlaubt.

ZeuS ist kein simpler Trojaner, sondern ein vielseitiges Framework. Kriminelle kaufen den Rahmen und statten ihn mit auf den Einsatzzweck abgestimmten Modulen aus, die in einschlägigen Foren ebenfalls zum Kauf angeboten werden. Das macht den Schädling sehr flexibel und senkt die technische Einstiegshürde für Kriminelle. Die Abzocker verstehen es zwar, gestohlene Kontodaten zu Geld zu machen, sind aber nicht automatisch auch versierte Programmierer.

Die Webinjects genannten Zusatzmodule sind die größte Stärke von ZeuS. Sie sorgen dafür, dass ein Opfer im Browser manipulierte Webseiten untergeschoben bekommt. Das funktioniert auch bei verschlüsselt übertragenen HTTPS-Seiten, da ZeuS vor und nach der Übertragung in den entschlüsselten HTML-Code eingreift. So werden beim Online-Banking etwa zusätzliche TAN-Felder angezeigt, deren Inhalt ZeuS einsammelt und an die Kriminellen übermittelt. Selbstredend schickt der Schädling auch alle nötigen Login-Daten mit, sodass die Kriminellen die unverbrauchten TANs für Überweisungen missbrauchen können. Die Webinjects werden in einschlägigen Foren für bekannte Bankenseiten angeboten. Ein Modul für die Sparkasse kostet 60 US-Dollar, Anpassungen schlagen mit 20 US-Dollar zu Buche. Diese sind auch gelegentlich nötig, denn die Banken versuchen die Webinjects durch minimale Änderungen im Quelltext ihrer Seiten ins Leere laufen zu lassen.

Im Mai dieses Jahres ist der Quellcode des ZeuS-Frameworks ins Netz gelangt. Das uns vorliegende 9,2 MByte große Archiv zeus.rar enthält Quelltexte und Binaries sämtlicher ZeuS-Komponenten: Angefangen beim eigentlichen Bot über das Konfigurationsprogramm ZeuS Builder bis hin zum Back-Connect-Server, der sich um den direkten Verbindungsaufbau mit dem infizierten Rechner kümmert. Auch der Command-and-Control-Server (C&C-Server) von dem der Bot seine Befehle abholt, befindet sich in dem Archiv.

Der ZeuS Builder ist mit einer überschaubaren Anzahl Funktionen ausgestattet. Er erzeugt das eigentliche Schadprogramm, das auch eine verschlüsselte Version der Konfigurationsdatei enthält. In dieser nimmt der Kriminelle alle wichtigen Einstellungen vor. Unter anderem legt man hier die URL zur jeweils aktuellen Version des selbst erstellten Bots und der verschlüsselten Konfigurationsdatei fest – so landen stets die neuesten Ausgaben der beiden Dateien auf den infizierten Rechnern, vergleichbar mit dem automatischen Update-Mechanismus von Windows. Auch die Textdatei, in der sich die Webinjects befinden, muss man angeben. Der Builder enthält eine Deinstallationsfunktion für den Bot, vermutlich um Tests zu vereinfachen.

Ein umfangreiches englischsprachiges Handbuch ist separat im Internet zu finden. Es beschreibt ausführlich die Grundfunktionen, wie das Sammeln von Account-Daten für FTP- und Mail-Zugänge, die Überwachung von Tastatureingaben und die Übertragung von Screenshots. Auch das Auslesen von Zertifikats- und Cookie-Speicher beherrscht ZeuS. Das Handbuch gibt erste Hinweise zur Funktionsweise des Bots. Es erklärt, wie der Bot die API-Aufrufe aus diversen Windows-DLLs, Sockets, wininet.dll und nspr4.dll abfängt.

Bots manipulieren mit Hilfe von Webinjects die Webseiten von Banken, um den Nutzer des infizierten Rechners gleich die ganze TAN-Liste abzuluchsen.

Letztere sind die Bibliotheken, mit denen die verschiedenen Browser ihren Netzwerkverkehr abwickeln: einerseits Internet Explorer, andererseits die Netscape Portable Runtime für alle Netscape-abgeleiteten Browser wie beispielsweise Firefox.

Googles Chrome ist von der von uns untersuchten ZeuS-Version nicht auf diese Weise angreifbar, denn es benutzt nicht die typischen Bibliotheken für den Netzwerkverkehr. Auch für Opera scheint sich der Schädling nicht zu interessieren, was auf die geringen Marktanteile zurückzuführen sein dürfte. Allerdings surft man mit Chrome und Opera auf einem infizierten System keinesfalls sicher: vor der Bildschirm- und Tastaturüberwachung durch ZeuS sind die Browser nicht gefeit. Ein Angreifer könnte ZeuS zudem zur Installation eines Browser-Add-on nutzen, das die aufgerufene Webseite vergleichbar mit einem Webinject manipuliert.

Blick hinter die Kulissen

ZeuS besteht aus mehr als 50 000 Zeilen Code, der zu einem Großteil solide in objektorientiertem C++ geschrieben ist. Teile des Quellcodes sind funktionsbasiert, weil sie die Windows-API zum Abfangen und Manipulieren der Windows-Funktionen nutzen. Weitere Bestandteile sind in Assembler geschrieben. Sämtliche Kommentare im Quelltext sind auf Russisch, immerhin sind die Bezeichner auf Englisch gehalten. Wichtige Strings sind nicht als Literal im Quelltext, was zu auffällig wäre, ZeuS verschleiert sie stattdessen per simpler XOR-Operation.

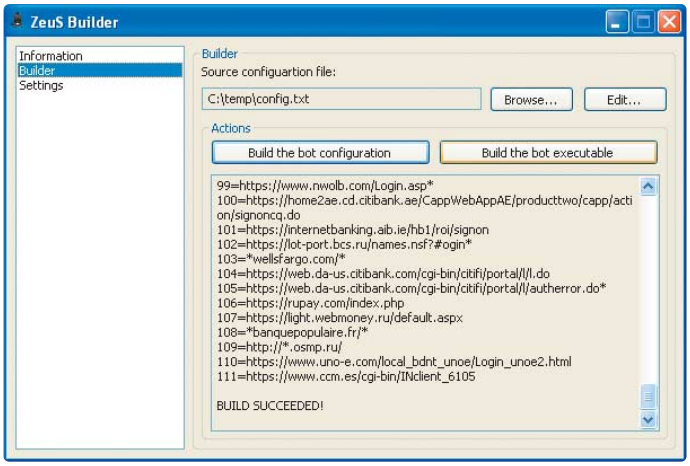
Wer den Code selbst kompilieren will, findet in der ReadMe-Datei nur wenige Informationen zum Build-Vorgang. Als Voraussetzung zum Kompilieren der C++-Quellen ist Visual C++ 9 angegeben, eine Komponente von Visual Studio 2008. In dem Archiv befinden sich auch Projektdateien für Visual Studio, allerdings führen diese nicht zum Erfolg. Wir konnten den Bot und den ZeuS Builder mit Hilfe der Make-Skripte kompilieren. Diese erfordern eine zuvor auszuwählende Konfigurationsdatei. Zur Auswahl steht interessanterweise auch eine Debugging-Konfiguration, die dazu führt, dass der fertige Bot alle wichtigen Aktionen in einer Textdatei protokolliert.

Der Code war offenbar nur für den Einsatz auf dem System des Entwicklers und nicht zur Weitergabe bestimmt: Die Skripte enthalten teilweise hartkodierte Pfade und nutzen 16-Bit-Tools, die unter einem 64-Bit-Windows nicht mehr laufen. Die Grundeigenschaften des Bots wie das Abfangen von Funktionsaufrufen (Hooking) aus Bibliotheken für den Netzwerkverkehr sind einzeln per „Define“ konfigurierbar. Der Build-Prozess erzeugt dynamisch etliche Quell- und Header-Dateien. Im Erfolgsfall beendet sich das Make-Skript schließlich mit einer Meldung in herrlich falschem Englisch: BUILD SUCCESSED!

Infektionserreger

Das Ergebnis des Bot-Build ist kein lauffähiges Binary, sondern nur das Grundprogramm, das erst nach der Behandlung mit dem ZeuS Builder einsatzbereit ist. Der Builder verschlüsselt die Konfigurationsdatei und bettet sie direkt in die Exe-Datei ein. Der erste Start der „bot.exe“ auf dem zu infizierenden System stößt zunächst einen Installationsprozess an, der keine Adminrechte benötigt. ZeuS begnügt sich sogar mit dem Gast-Account von Windows. Um sich dauerhaft im System einzunisten, kopiert der Bot eine Kopie seiner selbst unter einem zufällig generierten Dateinamen in das Verzeichnis „Anwendungsdaten“ im Benutzerordner. Die Kopie ist speziell auf das angemeldete Benutzerkonto zugeschnitten und nur mit diesem Konto lauffähig – so wird verhindert, dass man das Binary etwa zu Analysezwecken in einer virtuellen Maschine ausführt.

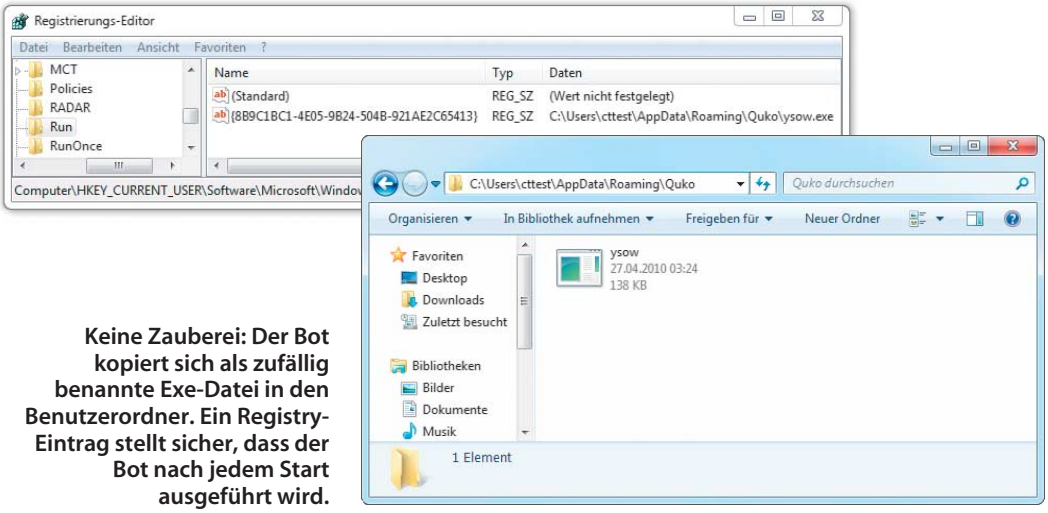
Durch einen Registry-Eintrag unter HKLM\Software\Microsoft\Windows\CurrentVersion\Run stellt der Bot sicher, dass er bei jedem Systemstart aktiviert wird. Auch hierfür nutzt ZeuS zufällige Werte, um seine Identifikation zu erschweren. Löscht man den Registrierungsschlüssel, wird er sofort wieder angelegt. Das liegt nicht etwa daran, dass ZeuS den Registry-Eintrag ständig im Auge behalten würde. Stattdessen legt der Schädling den Eintrag stumpf fünfmal pro Sekunde neu an. Als letzten Installationsschritt startet die Setup-Routine den Bot von seinem neuen Verzeichnis und löscht sich mit Hilfe eines Scripts anschließend indirekt selbst, um die Spuren der Installation zu verwischen.



Der Bot verschafft sich nach der Installation mittels CreateToolhelp32Snapshot() einen Überblick über alle laufenden Prozesse. Anschließend kopiert er sich mit Hilfe der Windows-Funktion WriteProcessMemory() in jeden Prozess und startet den injizierten Code als eigenen Thread mit der Windows-Funktion CreateRemoteThread(). Dieses Vorgehen ist etwas ungewöhnlich, hat aber gewisse Vorteile gegenüber dem üblichen Weg, den zu befallenden Prozess zu zwingen, eine DLL mit Schadcode zu laden. In diesem Fall würde jeder Prozess die gleiche unbekannte DLL enthalten, was sehr auffällig wäre. Bei der Methode von ZeuS gibt es weder einen unbekannten Prozess in der Taskliste, noch verdächtige DLLs. Nur der Aufwand für den Bot ist höher, denn statt des Ladens einer DLL muss er das Kopieren und Einschleusen des eigenen Codes „zu Fuß“ erledigen. Hat er alle Prozesse abgeklappert, beendet der Bot seine Ursprungsinstanz schließlich. Anschließend sieht das System sauber aus, der Schadcode schlummert in jedem Prozess im Hauptspeicher, ohne dass er einer Bibliothek zugeordnet ist.

Durch das Abfangen der Funktion zum Starten eines Programms wird jeder neu gestarte-

Per Klick zum Bot: Der ZeuS-Builder erfordert nur rudimentäre PC-Kenntnisse.



Keine Zauberei: Der Bot kopiert sich als zufällig benannte Exe-Datei in den Benutzerordner. Ein Registry-Eintrag stellt sicher, dass der Bot nach jedem Start ausgeführt wird.

te Prozess ebenfalls sofort infiziert. Die Infektionsroutine befindet sich in der Datei „source\client\core.cpp“:

```
static bool copyDataToProcess(HANDLE
process, void *image, void *curVa, void
*data, DWORD dataSize)
{
  DWORD_PTR rva =
(DWORD_PTR)(((LPBYTE)curVa) -
((LPBYTE)coreData.modules.current));
  return (CWA(kernel32,
WriteProcessMemory)(process,
(LPBYTE)image + rva, data, dataSize,
NULL) == FALSE) ? false : true;
}
```

In „source\client\coreinject.cpp“ ist die Basis der Prozess-Infektion mit injectMalwareToProcess() per CreateRemoteThread() enthalten. Interessanterweise bezeichnet der Entwickler sein Projekt an dieser Stelle selbst als Malware – offensichtlich ist er sich sehr wohl darüber im Klaren, was er da erschaffen hat. Mit CoreInject::_injectToAll() listet der Bot alle laufenden Prozesse auf und versucht, sie zu infizieren:

```
bool CoreInject::_injectToAll(void)
static bool
injectMalwareToProcess(DWORD pid,
HANDLE processMutex, DWORD
proccessFlags)
```

Nach der Infektion der Prozesse beginnt der Schädling mit dem Hooking. Das bedeutet, dass sich ZeuS vor bestimmte Funktionen klemmt, sodass zuerst der eigene Code aufgerufen wird und dann erst die originale Funktion. Dazu kopiert der Bot zunächst die erste Instruktion im Speicherbereich der Funktion und überschreibt sie mit einem JMP-Befehl auf die Schadfunktion. Ist die Schadfunktion durchgelaufen, springt sie anschließend den gesicherten Befehl an und führt dann den Rest der Originalfunktion aus.

So kann ZeuS die Daten, die ursprünglich an die Funktion übergeben werden sollten, beliebig manipulieren. Durch das Hooking der Netzwerkfunktionen kann er etwa Datenpakete aus dem Internet verändern, ehe sie den Browser erreichen. Die Adressen der zu überwachenden Funktionen im Speicher werden dabei mit der Windows-Funktion GetProcAddress() bestimmt oder aus der Import Address Table (IAT) gelesen, einer Liste von Adressen für alle aus Bibliotheken importierten Funktionen.

ZeuS überwacht zudem die wichtigen Windows-Basisfunktionen für die Benutzer- und Thread-Verwaltung (NtCreateUserProcess(), NtCreateThread(), NtQueryInformationProcess(), RtlUserThreadStart()), für die Dateiverwaltung (NtQueryDirectoryFile(), NtCreateFile()) und Bibliotheksverwaltung (LdrLoadDll(), LdrGetDllHandle()). Das Modul „source\client\corehook.cpp“ enthält die Implementierung der gehookten Ersatzfunktionen für die Basisfunktionalität aus den Bibliotheken ntdll und kernel32, „source\client\userhook.cpp“ das Überwachen wichtiger Funktionen der grafischen Oberfläche. Beispielsweise gelingt es dem Bot, per Hooking von TranslateMessage() sämtliche Tastatureingaben innerhalb des infizierten Prozesses zu überwachen oder mit GetClipboardData() den Inhalt der Zwischenablage zu stehlen.

Das Auslesen und Löschen des Zertifikatsspeichers von Windows erledigt das Modul „source\client\softwaregrabber.cpp“. Auch auf Cookies (einschließlich Flash-Cookies), das Windows-Adressbuch sowie in Outlook Express und Windows Mail gespeicherte Zugangsdaten hat es das Modul abgesehen. Zudem kann es Account-Daten von folgenden FTP-Clients abgreifen: FlashFXP, CuteFTP, Total Commander, WS_FTP, FileZilla, FAR Manager, WinSCP, FTP Commander, CoreFTP, SmartFTP. Die Kennwörter sind meistens in einer Konfigurationsdatei oder der Registry gespeichert, aber nur simpel verschleiert.

View report (Grabbed data [FTP client], 33 bytes)	
Bot ID:	CTTEST7HP_E6B6B0E4EB28ECC8
Botnet:	-- default --
Version:	2.0.8.9
OS Version:	Seven, SP 1
OS Language:	1031
Local time:	15.08.2011 14:30:33
GMT:	+2:00
Session time:	00:08:26
Report time:	15.08.2011 12:30:33
Country:	--
IPv4:	192.168.204.129
Comment for bot:	-
In the list of used:	No
Process name:	C:\Windows\system32\taskhost.exe
User of process:	cttest7hp\cttest
Source:	FileZilla

ftp://rei:geheim@ftp.heise.de:21

Die gestohlenen FTP-Zugangsdaten kann der Botnetz-Betreiber über die Verwaltungsoberfläche des Kommandoservers einsehen.

Auch Anwender einer 64-Bit-Version von Windows sind nicht vor ZeuS gefeit. Zwar ist der Bot ein reiner 32-Bit-Prozess und kann also solcher auch nur 32-Bit-Prozesse infizieren, da gemischte „Bittigkeit“ innerhalb eines Prozesses technisch nicht möglich ist. Tatsächlich sucht der Bot auf einem frisch aufgesetzten 64-Bit-Windows zunächst vergeblich nach 32-Bit-Prozessen, die er infizieren kann. Er benötigt wenigstens einen, um die eigentliche Schadroutine ausführen und Kontakt mit dem C&C-Server aufnehmen zu können. Die Situation ändert sich bereits beim Start des Internet Explorers: Klickt man nicht explizit auf die Programmverknüpfung mit dem Namenszusatz „(64 Bit)“, startet die 32-Bit-Ausgabe des Browsers. Zudem greifen die meisten Nutzer bei der Installation von Anwendungsprogrammen auf die 32-Bit-Fassung zurück – oftmals wird vom Hersteller auch gar keine 64-Bit-Version angeboten.

Obwohl sich ZeuS nicht direkt in 64-Bit-Prozesse einklinken kann, ist man mit diesen nicht vor der Überwachung durch den Schädling geschützt. Wenn der Angreifer den Bot erst mal ins System geschleust hat, kann er sämtliche Aktivitäten des Anwenders über den VNC-Server des Bots überwachen oder etwa einen globalen Keylogger auf dem System platzieren. Bei der 64-Bit-Ausgabe des Internet Explores kann ZeuS die aufgerufenen Webseiten nicht mit seinen Webinjects manipulieren, wodurch Online-Banking-Betrug nicht ohne Weiteres möglich ist. Auf einem 64-Bit-Windows-System beendet sich der Bot nach dem Infektionslauf nicht selbst, weil er möglicherweise der einzige 32-Bit-Prozess ist und sonst komplett inaktiv würde, wenn er seinen Code keinem anderen Prozess weitergeben kann. Man findet im Quellcode einige wenige Stellen, die auf eine geplante 64-Bit-Unterstützung hindeuten. Allerdings scheint es dem Entwickler nicht gelungen zu sein, dieses Vorhaben vollständig umzusetzen – vielleicht auch wegen der in Assembler geschriebenen Funktionen, die schwierig umzustellen sind.

Laut Handbuch wurde der Bot für den Einsatz unter Windows

Vista und 7 entwickelt, er unterstützt aber alle Versionen seit XP – einschließlich der Server-Ausgaben. Findet er eine ältere Version, beendet es die Infektion:

```
coreData.winVersion =
OsEnv::_getVersion();
    if(coreData.winVersion <
OsEnv::VERSION_XP)
    {
      WDEBUG1(WDDT_ERROR, "Bad
windows version %u.",
coreData.winVersion);
      return false;
    }
```

Windows 2000 hält der Autor offenbar nicht mehr für lohnend, denn aus technischen Gründen spricht nichts gegen eine Infektion.

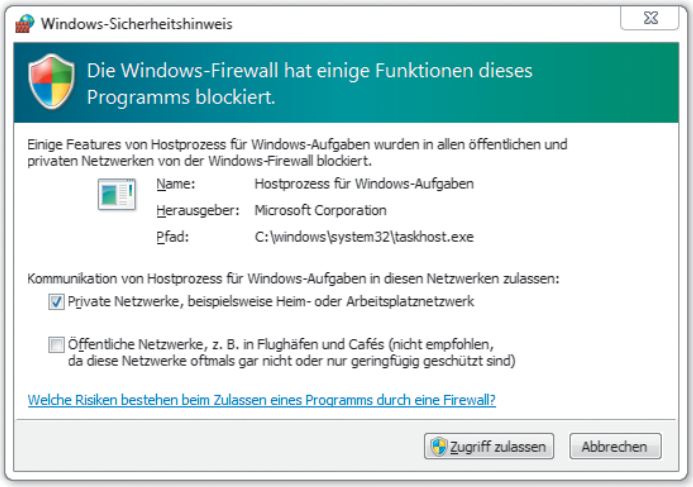
Nach Hause telefonieren

Der Schädling arbeitet in der Standardkonfiguration als reiner Client, alle Verbindungen zur Außenwelt erfolgen ausgehend. Das ist nötig, weil heutzutage typischerweise ein Rechner per DSL und Router mit integriertem NAT (Network Address Translation) ins Internet eingebunden ist und geöffnete Ports dadurch erst mal nicht über das Internet erreichbar sind. Außerdem sind ausgehende Verbindungen viel unauffälliger, da seit Windows XP SP2 sofort eine Firewall-Warnung erscheint, wenn ein Prozess auf Ports lauschen will. Die Kommunikation mit dem in PHP programmierten C&C-Server findet komplett auf Port 80 statt. Vom C&C-Server holt sich der Bot neue Aufträge und auch seine gesammelten Daten liefert er hier ab. Der Kontrollserver liefert dem Botnetz-Betreiber zudem Statistiken über die gesamte Bot-Armee und kann den Betreiber über den Jabber-Messenger informieren, wenn etwas Spannendes passiert; etwa, wenn das Opfer eine Online-Banking-Sitzung startet.

Ist der SOCKS-Server des Bots aktiv, wird die Windows-Firewall hellhörig.

ZeuS bedient sich eines Tricks, um lokale Dienste des infizierten Rechners trotz der im oberen Absatz beschriebenen Einschränkungen über das Internet erreichbar zu machen. Möglich macht dies der Back-Connect-Server, der wie ein Reverse-Proxy funktioniert. Er wird auf dem System des Angreifers gestartet und lauscht dort auf beliebigen Ports. Anschließend verbindet sich der Angreifer mit dem Server und erteilt dem Bot den Befehl, ebenfalls eine Verbindung zum Back-Connect-Server aufzubauen. Der Server vermittelt nun zwischen den beiden Parteien, wodurch der Angreifer auf lokale Dienste auf dem Rechner des Opfers zugreifen kann. Die Verbindungsrichtung vom infizierten Rechner ist nun ausgehend, das im oberen Absatz beschriebene Problem ist gelöst. Das klappt allerdings nur mit TCP-Paketen, mit UDP kann der Server nichts anfangen. Auf diese Weise kann der Angreifer auch auf den in den Bot integrierten VNC-Server zugreifen und dem Opfer in Echtzeit beim Online Banking über die Schulter schauen. Der Back-Connect-Server ist zwar mit Quelltext, aber nur als Windows-Version vorhanden.

Der Bot bringt auch einen SOCKS-Server mit, über den der Angreifer den infizierten Rechner für verschiedenste Netzwerkaktionen missbrauchen kann. Wurde der SOCKS-Server über die Konfigurationsdatei aktiviert, versucht der Bot allerdings, auf einem bestimmten Port zu lauschen. Das führt zu einer Warnmeldung der Windows-Firewall. Diese dürfte aber nur bei den wenigsten Anwendern Skepsis hervorrufen: um Erlaubnis bittet der „Hostprozess für Windows-Aufgaben“.



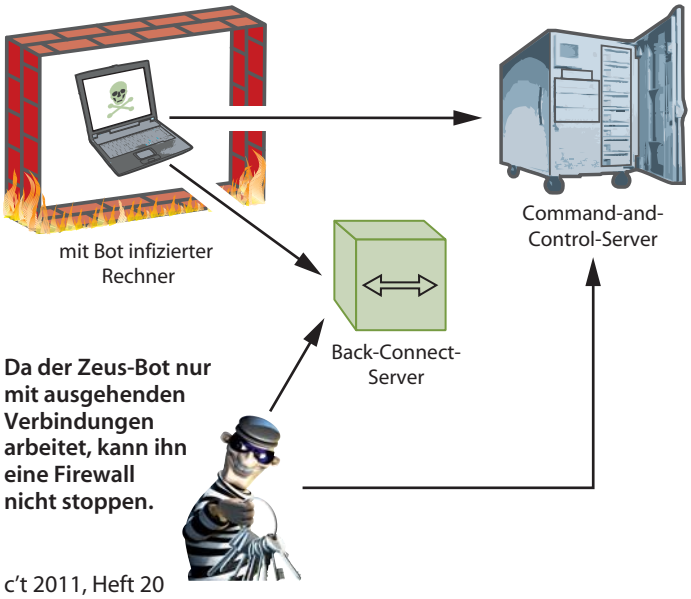
Einfach, aber effektiv

Überraschend ist, wie konventionell ZeuS aufgebaut ist. Es passiert alles im User-Modus, keine Kernel-Treiber sind beteiligt. Der Bot gibt sich keine große Mühe, sich vor Anwender und Virenscanner zu verstecken. Das Schadprogramm lässt sich also sehr leicht finden und entfernen. So genügt es auf einem 32-Bit-System bereits, das zufällig benannte Binary aus dem Benutzerordner zu löschen, da es nach der initialen Infektionsroutine nicht mehr ausgeführt wird. Auf einem 64-Bit-Rechner muss man den Prozess zunächst über den Taskmanager beenden. Nach einem Neustart kann der Bot die Prozesse nicht aufs Neue infizieren und der Spuk ist vorbei.

Das Verhalten des Bots ist eindeutig: Wer zunächst eine Prozessliste aufruft und anschließend in jeden einzelnen Prozess Code einschleusen will, während er mehrfach pro Sekunde einen Autorun-Eintrag in die Registry schreibt, führt sehr wahrscheinlich Böses im Schilde. Spätestens wenn der Prozess dann noch in regelmäßigen Abständen mit dem Internet kommunizieren will, sollten bei einer modernen Verhaltenserkennung eines Virenscanners alle Alarmglocken schrillen. Tatsächlich hat die kostenlose Verhaltensüberwachung ThreatFire (siehe Link) den Schädling auf unserem Testsystem mehrfach ausgebremst. Auf den signaturbasierten Scan allein kann man sich hingegen nicht verlassen: Die Kriminellen setzen in aller Regel sogenannte Crypter ein, die den Bot verändern, sodass er nicht mehr vom Virenscanner erkannt wird. Laut der Webseite Zeus Tracker (siehe Link) werden gerade einmal 39 Prozent der aktiv für Angriffe genutzten ZeuS-Binaries von rein signaturbasierten Virenscannern erkannt. Ein Verhaltensüberwachung ist also Pflicht – selbst wenn es Wege geben mag, auch die wieder auszutricksen.

Das von uns untersuchte Framework ist ein solider Unterbau für Cyber-Angriffe aller Art – und das, obwohl der Entwickler des Schädlingsbaukastens wenig technische Raffinesse eingebracht hat. Den anhaltenden Erfolg kann man sich durch den modularen Aufbau des ZeuS-Kits erklären. Die Kriminellen kaufen einmalig ZeuS und können ihn dann mit geringem Aufwand selbst an den jeweiligen Einsatzzweck anpassen. Und das sogar, wenn das System bereits infiziert ist. Durch den Kommandoserver nimmt der Bot fortlaufend neue Befehle entgegen. Auch die seit geraumer Zeit angebotenen Raubkopien dürften mit zu der hohen Verbreitung geführt haben – denn auch ein Virenschreiber ist vor Piraterie nicht gefeit. (rei)

www.ct.de/1120182



Da der Zeus-Bot nur mit ausgehenden Verbindungen arbeitet, kann ihn eine Firewall nicht stoppen.