Magdeburger Journal zur Sicherheitsforschung

Gegründet 2011 | ISSN: 2192-4260
Herausgegeben von Stefan Schumacher und Jan W. Meine
Meine Verlag – Wissenschafts-, Sach- und Fachbuchverlag, Magdeburg

Dieser Artikel erscheint in der Serie „Informationstechnologie und Sicherheitspolitik. Wird der dritte Weltkrieg im Internet ausgetragen?“ Herausgegeben von Jörg Sambleben und Stefan Schumacher

Timeo Danaos et dona ferentes: Zur Funktionsweise von Schadsoftware

Stefan Schumacher

Schadsoftware verschiedenster Art, von Viren über Würmer bis zu Trojanern, ist eine ernstzunehmende Gefahr für die Sicherheit aller IT-Systeme weltweit. Dieser Artikel stellt die Grundlegende Funktionsweisen vor und analysiert den Staatstrojaner genauer.

Zitationsvorschlag: Schumacher, S. (2012a). Timeo Danaos et dona ferentes: Zur Funktionsweise von Schadsoftware. *Magdeburger Journal zur Sicherheitsforschung*, *2*, 189–221. Zugriff am 20. November 2012, unter http://www.sicherheitsforschung-magdeburg.de/journal_sicherheitsforschung.html

Historischer Abriss

Schadsoftware ist kein neuartiges Problem, bereits im 2. Weltkrieg befassten sich Mathematiker mit den theoretischen Grundlagen selbstreproduzierender Programme.

1966 Neumann (1966) beschreibt die theoretischen Grundlagen selbstreproduzierender Programme. Sein Buch basiert auf Vorlesungen die er bereits in den 1940er Jahren gehalten hat.

1971 Bob Thomas schreibt *Creeper*, welcher DEC PDP-10-Rechner mit TENEX infiziert.

1975 John Brunner verwendet in seinem Roman »The Shockwave Rider« den Begriff *Wurm* für selbstreproduzierende Programme.

1980 Jürgen Kraus schreibt an der Ruhr-Universität Bochum sein Diplomarbeit »Selbstreproduktion bei Programmen«.

1981 Der 14-jährige Richard Skrenta veröffentlicht *Elk Cloner*. Der Virus verbreitet sich auf Apple][-Systemen über die Startdisketten. Elk Cloner gilt als erster großer Malware-Ausbruch in der freien Wildbahn.

1983 Frederick Cohen prägt die Begriffe *Virus* und *infizieren*.

1984 Ken Thompson erhält den Turing-Award. In seiner Rede »Reflections on Trusting Trust« beschreibt er wie man über Manipulationen am Compiler eine Backdoor in Programmen einbaut. (Thompson 1984)

1986 Der Pakistani-Virus wird von Basit Farooq Alvi und Amjad Farooq Alvi veröffentlicht. Er gilt als der erste IBM-PC-kompatible Virus und ist verantwortlich für die erste Epidemie auf PCs.

1987 Mit SCA verbreitet sich der erste größere Amiga-Virus über den Bootsektor.

1988 Robert Tappan Morris entwirft den Morris-Wurm und entlässt ihn am MIT in die Wildbahn. Der Wurm nutzte verschiedene bekannte Sicherheitslücken in Unix-Diensten aus und infizierte DEC VAX-Systeme mit 4BSD oder Sun-3. Man geht davon aus, dass der Wurm ca. 6 000 Rechner infizierte, was 10% aller Rechner im damaligen Internet entsprach. Durch die entstehende Last auf den infizierten Systemen und im Netzwerk wurde das damalige Internet nahezu lahmgelegt.

1989 Friðrik Skúlason entdeckt Ghostball, den ersten multipartiten Virus, d.h. Virus der sich auf verschiedenen Wegen verbreitet.

1992 Der Michelangelo-Virus sollte am 06. März auf Millionen Computern die Daten löschen und eine von den Medien heraufbeschworene digitale Apokalypse verursachen. In der Realität blieb der Schaden aber minimal. Der Vorfall wurde auch im Film Hackers thematisiert.

1994 Mit OneHalf wird der erste polymorphe Virus veröffentlicht. Er verändert bei jeder Infektion seine eigene Signatur, um die Erkennungsrate zu senken.

1995 Der erste Makro-Virus wird veröffentlicht, er greift Microsoft Word-Dateien an.

1999 Der KAK-Wurm nutzt Javascript um sich über eine Sicherheitslücke in Outlook Express fortzupflanzen.

2000 ILOVEYOU verbreitet sich im Internet und nutzt dazu Visual Basic

Script (VBS), eine interpretierte Sprache auf Microsoft Windows. Innerhalb von Stunden infizieren sich Millionen von Rechnern, ILOVEYOU gilt damit als einer der schwersten Wurmausbrüche überhaupt.

2001 Der Anna-Kournikova-Virus verbreitet sich im Internet. Er wird als E-Mail verteilt und verspricht dem Empfänger ein Bild von Anna Kournikova. Öffnet der Empfänger das angebliche Bild, wird stattdessen eine VBS-Datei ausgeführt, die die E-Mail an alle Adressbucheinträge in Outlook verschickt.

2001 Sadmind attackiert Microsoft IIS und Sun Solaris-Systeme.

2001 Nimda wird entdeckt, es verbreitet sich unter anderem über Backdoors die von Code Red II und Sadmind hinterlassen wurden.

2002 Simile, der erste metamorphische Virus, wird entdeckt. Der Virus kann sich selbst umprogrammieren, um seiner Entdeckung zu entgehen.

2004 Santy, der erste Webwurm wird entdeckt. Er infiziert Webseiten die phpBB nutzen, eine Software die ein webbasiertes Diskussionsforum bereitstellt. Der Wurm nutzte Google um Server mit phpBB zu finden, bis Google die Suchanfrage blockierte.

2005 Samy Kamkar entwickelt Samy, eine Cross Site Scripting-Wurm, der sich auf MySpace verbreitete. Innerhalb von 20 Stunden infizierte der Wurm mehr als eine Million MySpace-Profile.

2006 OSX/Leap-A, die erste Malware für Mac OS X wird entdeckt.

2007 Der Storm-Wurm infiziert mehrere Millionen Rechner und stellt sie als Zombies ins Storm-Botnet, wo sie von einem zentralen Steuerprogramm befehligt werden können.

2008 Conficker infiziert MS-Windows-Serversysteme in über 200 Ländern. Betroffen waren unter anderem die französische Kriegsmarine, das britische Verteidigungsministerium und die Bundeswehr.

2009 Im Rahmen der großen Cyberattacken auf die USA und Südkorea werden diverse Botnetze aktiviert.

2010 Stuxnet wird enttarnt.

2011 Duqu wird enttarnt.

2012 Flame wird enttarnt.

Einführung

Schadsoftware ist Software, die entwickelte wurde um auf Computersystemen Schaden anzurichten bzw. um Dinge zu tun, die vom berechtigten Benutzer nicht erwünscht sind oder diesem verborgen bleiben sollen. Schadsoftware wird unter anderem dazu eingesetzt Daten von Rechnern abzufischen, zum Beispiel Zugangsdaten zu Online-Banken oder verschlüsselte E-Mails. Außerdem kann über Schadsoftware die Kontrolle über einen fremden Rechner übernommen und dieser ferngesteuert werden.

Zu Beginn der Analyse von Schadsoftware sind einige Begriffe zu klären bzw. zu definieren.

Sicherheit kann auf verschiedene Arten und Weisen definiert werden, eine ausführliche Analyse des Begriffs »Sicherheit« sowie seine Explikation habe ich in Schumacher (2011c, 2012b) vorgenommen. In der Diskussion um Schadsoftware und den Staats-

trojaner genügt vorerst die rein technische Dimension. Ich expliziere Sicherheit hier anhand der sogenannten VIVA-Kriterien nach Bundesamt für Sicherheit in der Informationstechnik (2006):

Vertraulichkeit Vertrauliche Informationen müssen vor unbefugter Preisgabe geschützt werden.

Integrität Die Daten sind vollständig und unverändert. Der Begriff »Information» wird in der Informationstechnik für »Daten« verwendet, denen je nach Zusammenhang bestimmte Attribute wie z. B. Autor oder Zeitpunkt der Erstellung zugeordnet werden können. Der Verlust der Integrität von Informationen kann daher bedeuten, dass diese unerlaubt verändert wurden oder Angaben zum Autor verfälscht wurden oder der Zeitpunkt der Erstellung manipuliert wurde.

Verfügbarkeit Dem Benutzer stehen Dienstleistungen, Funktionen eines IT-Systems oder auch Informationen zum geforderten Zeitpunkt zur Verfügung.

Authentisierung Bei der Anmeldung an einem System wird im Rahmen der Authentisierung die Identität der Person, die sich anmeldet, geprüft und verifiziert. Der Begriff wird auch verwendet, wenn die Identität von IT-Komponenten oder Anwendungen geprüft wird. Ist die Authentisierung erfolgreich, spricht man auch davon, dass die Person oder ein Datum authentisch ist bzw. die Authentizität gewährleistet ist.

Um die Sicherheit eines technischen Systems zu untersuchen und zu bewerten, genügt es das System auf die genannten VIVA-Kriterien hin zu untersuchen, was ich im folgenden in diesem Artikel tun werde.

Weitere notwendige Begriffe sind die Vulnerability und der Exploit. Die Vulnerability (in etwa *Verwundbarkeit*) ist eine Sicherheitslücke in einem System. Wird diese Sicherheitslücke durch eine Software oder einen anderen Angriff ausgenutzt, spricht man von einem Exploit. Ein Angreifer muss also eine Vulnerability entdecken und sie mit einem Exploit ausnutzen.

Nachdem ein Angreifer Zugriff auf ein System erlangt hat, installiert er in der Regel ein sogenanntes Rootkit. Das Rootkit vereinfacht die Kontrolle über das Opfersystem, in dem es beispielsweise einen neuen Benutzer für den Angreifer anlegt und dessen Dateien und Prozesse vor den berechtigten Benutzern versteckt. Der Name des Rootkits leitet sich von `root` ab, dem Administratorkonto auf Unix-Systemen.

Ein Wurm ist ein Programm, das sich selbständig verbreitet und dazu Exploits auf den Opfersystemen ausnutzt. In der Regel trägt ein Wurm eine Nutzlast mit sich, die z. B. die Kontrolle über das Opfersystem übernimmt. Ein Virus ist hingegen ein Programm mit einer Schadroutine, welches zur Verbreitung eine Wirts-Datei benötigt, ähnlich wie sich das biologische Virus über seine Träger verbreitet. Anfang der 1990er Jahre waren der Master Boot Record von Disketten und Word- und Excel-Dateien verbreitete Träger von Viren. Ein Trojaner ist ein Schadprogramm, dass sich selbständig (als Wurm) oder über einen Träger (als Virus) verbreitet. Ziel des Trojaners ist es, unerkannt in ein System einzudringen und dort beispielsweise Login-Daten abzugreifen und an einen Kontrollrechner zu schicken. In der Praxis oder Tagespresse werden diese klaren Abgrenzungen meist nicht vorgenommen und die Begriffe Trojaner, Vi-

rus und Wurm nicht klar voneinander getrennt. Meist hat dies auch für die Leser bzw. Anwender von Computern keine Auswirkung, in der wissenschaftlichen Diskussion ist dies aber notwendig, da hinter den unterschiedlichen Konzepten auch unterschiedliche Angriffswege und Funktionen stecken.

Eine sogenannte Man-in-the-Middle-Attacke ist ein Angriff, bei dem die Datenübertragung mitgeschnitten und abgehört und/oder manipuliert wird. So lässt sich jede beliebige E-Mail auf dem Übertragungsweg zwischen den Mailservern abfangen und lesen oder verändern und weiterleiten. Dabei wird die Vertraulichkeit, Integrität, Verfügbarkeit und Authentizität der E-Mail gefährdet. Um eine derartige Attacke zu verhindern, können die Kommunikationspartner ihre E-Mails untereinander verschlüsseln. Danach ist es einem Angreifer zwar möglich die E-Mail weiterhin abzufangen, er kann sie aufgrund der Verschlüsselung und Signatur nicht mehr lesen und manipulieren, ebensowenig kann er sich als ein anderer Kommunikationspartner ausgeben.

Abbildung 1 zeigt die Man-in-the-Middle-Attacke am Beispiel von Alice und Bob, die sicher miteinander kommunizieren wollen. In Abbildung 1(a) kommunizieren Alice und Bob unverschlüsselt, so dass sich Mallory in die Kommunikation einklinken und diese manipulieren kann. Abbildung 1(b) zeigt einen kryptographischen Tunnel zwischen Alice und Bob, den Mallory nicht umgehen kann. Deshalb dringt Mallory in Abbildung 1(c) in den Computer von Alice ein und kann die relevaten Daten vor der Verschlüsselung abschnorcheln. Dazu kann er einen Trojaner einsetzen.

Um einen Trojaner oder eine andere Schadsoftware auf einem Zielsystem ausrollen zu können, muss es angegriffen und kompromittiert werden. Für diese Angriffe existieren verschiedene Angriffsvektoren, so kann ein Angreifer beispielsweise die Passwörter von Benutzern ausspähen, erraten oder cracken, um über dass kompromittierte Konto die Schadsoftware zu installieren. Eine andere Methode ist das Phishing, bei dem einem Anwender ein Trojaner untergeschoben werden soll. So tarnte sich bspw. der Flashback-Trojaner als Flash-Update. Installierte der Benutzer das vermeintliche Flash-Update, wurde im Hintergrund ein Trojaner ausgerollt, der unter anderem den Netzwerkverkehr des Safari-Browsers manipulierte.

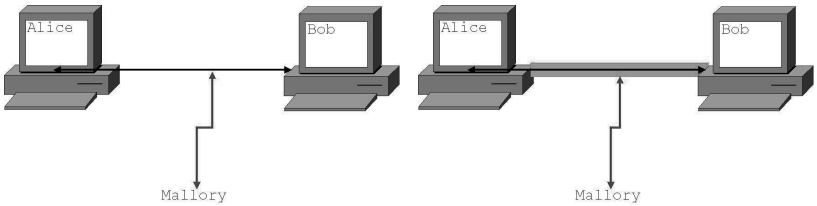
Komplizierter sind in der Regel Angriffe, bei denen technische Sicherheitslücken ausgenutzt werden, wie Buffer Overflows. Dabei werden zu große Datenmengen in einen Speicherbereich geschrieben, der zu klein ist. Dies kann zum Absturz des System führen, oder aber auch dazu dass ein Angreifer erweiterte Rechte auf dem System erlangt.

Dieses Prinzip funktioniert, da die grundlegende Architektur der heute eingesetzten Computer die von-Neumann-Architekur ist. Sie zeichnet sich dadurch aus, dass ein einziger Speicher sowohl Befehle als auch Daten enthält. Abbildung 2 zeigt eine schematische Darstellung der Architektur.

Die Architektur besteht aus folgenden Teilen:

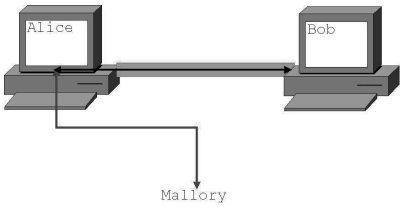
Rechenwerk führt logische Verknüpfungen und Rechenoperationen mit den Daten

Steuerwerk regelt die Befehlsfolge und verschaltet Datenquelle und -ziel sowie weitere Komponenten



(a) Mallory kann die unverschlüsselte Kommunikation abfangen und verändern

(b) Lösung: Verschlüsselung zwischen den Endpunkten ermöglicht Authentifikation und Vertraulichkeit



(c) abschöpfen der Daten auf dem Zielsystem selbst, bevor diese verschlüsselt werden

Abbildung 1: Verschlüsselte und Unverschlüsselte Kommunikation

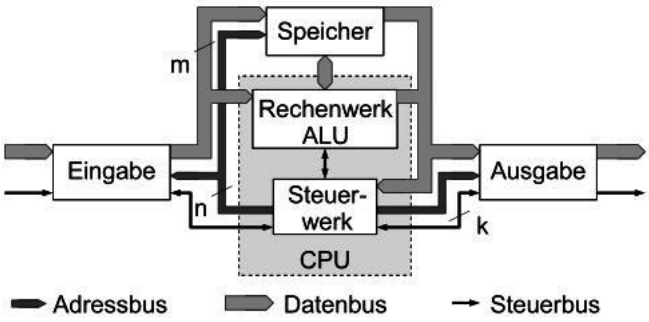


Abbildung 2: Von-Neumann-Architektur

Speicherwerk speichert Programme und Daten

Eingabe-/Ausgabewerk regelt Ein- und Ausgabe von Daten

Durch ihren Aufbau ist die von-Neumann-Architektur kompatibel zur Turing-Maschine, das heißt alle Probleme die mit einer Turing-Maschine berechenbar sind, sind auch mit einem von-Neumann-Rechner berechenbar. Die Turing-Maschine bzw. die Turing-Berechenbarkeit ist eines der zentralen und grundlegenden Konzepte der theoretischen Informatik und wird im allgemeinen dazu verwendet, den Begriff Algorithmus zu definieren. Daher soll es kurz erläutert werden:

Der englische Mathematiker Alan Turing formalisierte während des Zweiten Weltkrieges[1] mathematische Berechnungsvorschriften mit dem Modell der Turingmaschine, welche formal als 7-Tupel definiert ist (vgl. Stiebe 2003):

1. $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, q_f)$:= formales 7-Tupel
2. Q := endliche Zustandsmenge
3. Σ := Eingabealphabet
4. Γ := endliches Bandalphabet mit $\Sigma \subset \Gamma$
5. $\delta : (Q \setminus \{q_f\}) \times \Gamma \mapsto Q \times \Gamma \times \{L, 0, R\}$:= partielle Überführungsfunktion
6. $q_0 \in Q$:= Anfangszustand
7. $\square \in \Gamma \setminus \Sigma$:= leeres Feld
8. $q_f \in Q$:= akzeptierende Zustand

Ein Algorithmus muss daher folgenden Kriterien genügen:

Finitheit der Algorithmus muss in einer endlichen Vorschrift eindeutig beschreibbar sein

Ausführbarkeit jeder Schritt des Algorithmus muss ausführbar sein

Platzkomplexität der Algorithmus darf nur endlich viel Speicherplatz erfordern

Zeitkomplexität der Algorithmus darf nur endlich viele Schritte erfordern

Determiniertheit der Algorithmus muss bei den gleichen Eingaben die selben Ausgaben liefern

Determinismus die nächste anzuwendende Berechnungsvorschrift[2] ist jederzeit eindeutig definiert

Genügt eine Berechnungsvorschrift diesen Kriterien nicht, ist sie kein Algorithmus und kann daher auch nicht mit einem Computer (sinnvoll) ausgeführt werden (vgl. Saake und Sattler 2002, Kap. 2 »Algorithmische Grundkonzepte«).

Ein entscheidender Nachteil der Architektur ist es, dass ein Programm im Prinzip auf jedes beliebige Datum im Speicher zugreifen kann. So kann beispielsweise ein Benutzer das Login-Passwort eines anderen Nutzers auf dem selben Rechner aus dem Arbeitsspeicher auslesen. Damit dies nicht passiert, muss das Betriebssystem dafür sorgen, dass nur berechtigte Nutzer und Prozesse auf die jeweiligen Daten zugreifen können. Allerdings gilt auch hier, dass der Systemadministrator bzw. Root auf alle Daten zugreifen kann. Kann also ein Angreifer Root-Rechte auf einem System erlangen, kann er in der Regel auch alle ande-

1 Turing arbeitete in Bletchley Park daran die deutsche Verschlüsselungsmaschine Enigma zu brechen. Die Formalisierung der Berechenbarkeitstheorie wurde also direkt von einer Sicherheitsanalyse getrieben.

2 Den Begriff Determinismus bzw. Determiniertheit verwende ich in dieser Arbeit ausschließlich im hier gezeigten Sinne der Theoretischen Informatik.

ren Daten auslesen. Dieser Punkt ist zwingend notwendig, wenn man wie in Abbildung 1 gezeigt verschlüsselte Daten im Zielsystem vor der Verschlüsselung abschnorcheln will.

Der hier gezeigte Nachteil in Bezug auf Sicherheit wird aber durch einige entscheidende Vorteile aufgewogen, zum Beispiel kommt es durch das zentrale Steuerwerk nicht zu Race Conditions, bei der konkurrierende Programme auf die gleichen Daten zugreifen wollen und sich gegenseitig blockieren (sogenannter Deadlock).

Verlassen wir nun die Ebene der Hardware und wenden uns der Software, insbesondere dem Betriebssystem zu. Wie bereits oben beschrieben, muss das Betriebssystem die Zugriffe auf die Daten im Speicher koordinieren und unberechtigte Zugriffe ablehnen. Eine Möglichkeit dazu ist das in Abbildung 3 gezeigte Schalenmodell. Die Schalen bauen dabei aufeinander auf und und grenzen die Zugriffsrechte voneinander ab. Je höher bzw. weiter aussen eine Schale ist, desto weniger Zugriffsrechte hat sie. Benötigt die Schale mehr Zugriffsrechte, weil sie auf bestimmte Daten, Prozesse oder Hardware zugreifen will, muss ihr diese das Betriebssystem zuweisen. Die in der Abbildung rot eingefärbten Schalen laufen im sogenannten Kernel-Modus und haben alle Zugriffsrechte auf jeden Prozess. Ziel einer erfolgreichen Schadsoftware muss es daher sein, sich entweder an die auszuspionierenden Prozesse andocken oder Kernel-Rechte erlangen um auf die Zielprozesse zugreifen zu können.

Angriffsvektoren zur Verbreitung von Schadsoftware

Als Angriffsvektor bezeichnet man Wege auf dem sowie die Art und Weise wie ein Angriff erfolgt. Ziel in der Verteidigung von IT-Systemen ist es, mögliche Angriffsvektoren aufzuspüren und abzusichern bzw. deren Ausnutzung zu erschweren.

Passwörter: sind ein beliebter und recht einfach auszunutzender Angriffsvektor. Gelangt ein Angreifer an das Passwort eines Benutzerkontos kann er dieses ausnutzen und hat automatisch und sofort alle Rechte des jeweiligen Benutzers zur Verfügung. Mit diesen kann er versuchen weitere Angriffsvektoren auszunutzen um Root-Rechte zu erlangen. Gelangt der Angreifer an das Root- oder Administrator-Passwort, hat er die volle Kontrolle über einen Rechner erlangt.

Passwörter lassen sich auf verschiedene Arten und Weisen ausnutzen. Eine einfache Variante ist es, ein Passwort zu erraten oder zu recherchieren. So neigen viele Benutzer dazu sich einfach zu merkende Passwörter auszudenken, beispielsweise den Namen der eigenen Kinder oder der Katze. Bei einem gezielten Angriff kann ein Angreifer diese Name recherchieren und als Passwort einfach ausprobieren.

Exkurs: Passwortsicherheit im Stratfor-Hack Weihnachten 2011 wurde der Sicherheitsdienstleister Stratfor von Anonymous gehackt und die gesamte Kundendatenbank gestohlen und auf einem Filehoster publiziert. Die Datenbank enthält unter anderem die Mailadresse des Kunden sowie den MD5-Passwort-Hash.

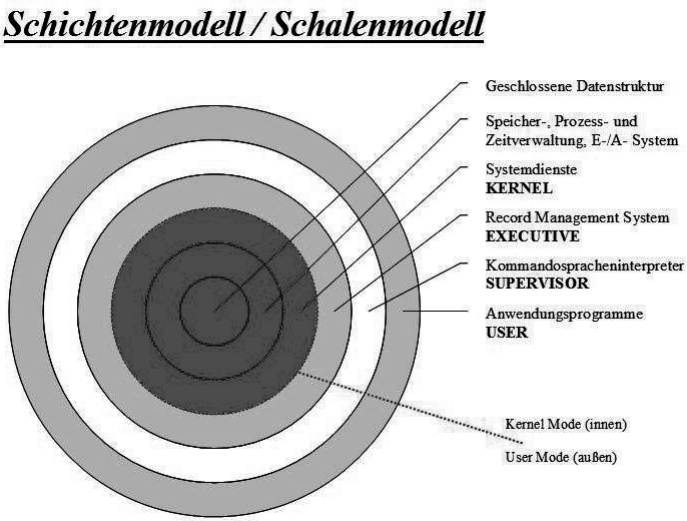


Abbildung 3: Schalenmodell (auch Schichten- oder Ringmodell), verändert nach http://de.wikipedia.org/w/index.php?title=Datei:Schichtenmodell.jpg&oldid=70884859

MD5 ist eine mathematische Funktion, die zu einer beliebigen Eingabe eine 32-stellige Hexadezimalausgabe, den sogenannten Hash erstellt. Dieser Hash wird anstelle des Klartextpassworts in der Datenbank abgelegt und gespeichert. Meldet sich ein Benutzer an, wird seine Eingabe ebenfalls mit MD5 gehasht und mit dem gespeicherten Wert verglichen. Da gleiche Eingaben immer gleiche Hashes erzeugen, stimmt der berechnete Hash mit dem gespeicherten Hash überein, der sich anmeldende Benutzer hat also ein korrektes Passwort eingegeben.

Hashes lassen sich nicht zurückrechnen, da es keine inverse mathematische Funktion gibt. Das heisst man kann sehr einfach zu einer beliebigen Eingabe einen Hash berechnen, aber aus einem beliebigen Hash nicht die Eingabe zurückrechnen. Man kann also das y in der Gleichung $f(x) = y$ einfach berechnen, zu einem beliebigen y aber das x nicht rekonstruieren.

Eine einfache Möglichkeit derart abgespeicherte Passwörter zu cracken, ist eine sogenannte Wörterbuchattacke. Dabei werden Wortlisten, beispielsweise aus Wikipedia oder dem Duden, als Eingabe an MD5 übergeben, so das zu jedem Wort der passende MD5-Wert abgespeichert wird. Nun vergleicht man die MD5-Werte der Wortliste mit den abgespeicherten Passwörtern. Stimmen die MD5-Werte überein, kann man aus der Wortliste das zugehörige Klartext-Wort ablesen.

Aus diesem Verfahren leitet sich daher die Regel ab, keine Passwörter zu verwenden, die in einem Wörterbuch stehen.

Da Passwörter zur Zeit immer noch das

wichtigste Authentifikationsmerkmal an IT-Systemen darstellen und damit die direkte Schnittstelle zwischen psychischer und technischer Dimension darstellen, habe ich die Datensätze statistisch und mit einer Wörterbuchattacke analysiert.

Zuerst habe ich die Hashes auf Dubletten überprüft und festgestellt das es Dubletten gibt. Dies zeigt zum einen, dass die Passwörter während des Hash-Vorganges nicht durch eine Zufallskomponente weiter geschützt wurden (sog. Salt). Der Entwickler bzw. Administrator der Datenbank hat also ein Sicherheitsverfahren verwendet dass nicht mehr dem aktuellen Stand der Technik entspricht, er hat also nicht sicherheitskompetent gehandelt.

Wesentlich interessanter ist aber hierbei die Analyse der vorliegenden Passwort-Hashes. So zeigte sich, dass ein einzelner Passwort-Hash in der 860 160 Einträge umfassenden Datenbank 12 023 mal vorkam. Es gibt insgesamt 9 653 Dubletten die 47 147 Datensätze umfassen. Das heißt, das mehr als 47 000 Nutzer Passwörter ausgewählt haben, die entweder in einem Wörterbuch vorkommen, naheliegende Termini sind oder von einem Systemadministrator vorgegeben wurden. Da die Datenbank Dubletten enthält, habe ich diese mit einer Wörterbuchattacke angegriffen. Dazu habe ich diverse naheliegende Zeichenketten und Passwörter (123456, Stratfor, qwerty etc.) sowie Wortlisten aus Wörterbüchern berechnen lassen und mit der Stratfor-Datenbank verglichen. Tabelle zeigt die 20 häufigsten Passwörter. Bezeichnend ist, das allein das Passwort »stratfor« mehr als 12 000 mal verwendet wird, außerdem wird »usmcportal« 148 mal verwendet. USMC ist die Abkürzung für United States Marine Corps und wird ausschließlich von E-Mailadressen des USMC verwendet. Es zeigt sich hier exemplarisch, das selbst Soldaten, die an sicherheitsrelevanten Systemen arbeiten und bei Stratfor strategische Daten abrufen die eigenen Vorschriften zu sicheren Passwörtern ignorieren oder bewusst missachten um ein naheliegendes aber sehr schwaches Passwort zu verwenden. Ebenfalls auffällig ist das Passwort »changeme«, das vermutlich von einem Systemadministrator von Stratfor oder beim Kunden gesetzt wurde und den eigentlichen Nutzer dazu auffordert, das Passwort zu ändern. Trotzdem wird dieses Passwort noch 265 mal eingesetzt. Insgesamt gelang es mir in nicht einmal 48 Stunden 56 576 Passwörter zu cracken, dies sind 6,7% aller Passwörter. Tabelle 1 zeigt die 15 häufigsten Passwörter in der Stratfor-Kundendatenbank.

Hierbei zeigt sich, das selbst auf einer Webseite die sich mit Sicherheitsthemen befasst und von Mitarbeitern staatlicher Sicherheitsbehörden aufgesucht wird, einfachste Sicherheitsregeln nicht beachtet werden. Belgers (1993) kam bereits 1993 (!) zu einem ähnlichen Ergebnis. Er hat das selbe Verfahren zur Berechnung von Passwort-Hashes auf einer 33-MHz-schnellen SUN Sparcstation laufen lassen und konnte innerhalb 25 Stunden bereits 11% der Passwörter brechen. Diese bestanden aus typischen Mustern (qwerty) und niederländischen Wörtern (vrijheid, kasteel). In den vergangenen 20 Jahren hat sich also im Bereich Passwortsicherheit bzw. Auswahl der Passwörter durch den Benutzer nicht viel geändert.

Social Engineering, Phishing und Spear Phishing: Social Engineering ist eine Angriffsform die statt rein technischer Maßnahmen auch die Manipulation des Benut-

Passwort	Häufigkeit
stratfor	12023
123456	625
0000	519
password	517
stratfor1	426
strat4	265
changeme	265
1qaz2wsx	232
1234	228
wright	179
usmcportal	148
abc123	89
qwerty	85
12345	75
12345678	74

Tabelle 1: Die 20 häufigsten Passwörter in der Stratfor-Datenbank

zers beinhaltet. Es gibt verschiedene Möglichkeiten um Social Engineering zu nutzen, so kann man beispielsweise eine Zielperson per Telefonanruf dazu bewegen, ein Passwort oder andere sicherheitsrelevante Daten auszuplaudern. Wie genau Social Engineering funktioniert habe ich in Schumacher (2009a,b,c,d, 2010b, 2011a) ausführlich beschrieben.

Man kann Social-Engineering-Methoden auch mit technischen Angriffen verschneiden. Ein Beispiels dafür sind Phishing-Attacken. Dabei verschickt ein Angreifer E-Mails die wie echte E-Mails einer Bank oder eines Versandhandels aussehen. In diesen E-Mails wird der Empfänger gebeten aufgrund technischer Probleme oder aus anderen Gründen seine Zugangsdaten auf einer bestimmten Webseite einzugeben. Die verlinkte Webseite ist aber nicht die echte Webseite der vorgetäuschten Organisation, sondern eine gefälschte Variante, bei der die eingegebenen Zugangsdaten gesammelt werden. Die Daten werden dann genutzt um die Konten der Opfer zu plündern oder mit ihren Zugangsdaten einzukaufen.

Eine weitaus elaboriertere Form des Social Engineerings bzw. Phishings ist das Spear Phishing. Die oben vorgestellte Form des Phishings ist im Einzelfall in der Regel nicht sehr erfolgreich. Da es aber recht einfach ist ein paar Milliarden E-Mails zu verschicken, erreichen die Angreifer über die hohe Zahl der Empfänger prozentual gesehen genügend Opfer, die auf den Trick hereinfallen und geplündert werden können. Beim Spear Phishing hingegen wird ein Opfer oder eine kleine Gruppe potenzieller Opfer ausgewählt und gezielt angegriffen. Dazu betreibt ein Angreifer in der Regel einige Nachforschungen über sein Opfer, um basierend auf den Hintergrundinformationen ein psychlogisches Profil zu erstellen. Mithilfe dieses Profils wird der Angriff speziell auf das Opfer zugeschnitten, wodurch sich die Erfolgschance massiv erhöht.

Ein Angriffsszenario aus meiner Beratungspraxis sah so aus: Ein Manager einer höheren Ebene eines Zulieferers aus der Automobilbranche wurde als Opfer auserkoren. Die Angreifer besorgten sich über soziale Netzwerke (Stayfriends, Xing, LinkedIn) Hintergrundinformationen zu dem Opfer. Besonders interessant waren hierbei Informationen über alte Bekannte und Freunde, deren Identität ein Angreifer annehmen konnte. Weitere interessante Informationen waren Vorlieben und Hobbies, wie Lieblingsbands, Urlaubsorte und vor allem der Lieblingsfußballverein des Managers.

Die Angreifer erstellten nun mehrere Identitäten aus dem Bekanntenkreis des Opfers und überprüften mittels dieser Identi-

täten die Korrektheit der Informationen, so wurden bspw. E-Mails unter dem Namen ehemaliger Kollegen geschickt und nachgefragt, ob der Manager immer noch Fan des Vereins X ist, was dieser dann verneinte und antwortete, dass er schon sein Leben lang Fan von Y ist.

Nachdem die gewonnenen Daten so mehrfach überprüft wurden, richteten die Angreifer ein einfaches Webforum für Fans des Vereins Y ein und befüllten es mit einigen Inhalten. Anschließend erzeugte man eine E-Mailadresse für den ehemaligen Klassensprecher der Gymnasialklasse des Opfers. Dieser »Klassensprecher« gab sich als Betreiber des Fan-Forums aus und lud in der E-Mail seinen alten Klassenkameraden ein, dem Forum beizutreten. Gelockt wurde unter anderem mit der Behauptung, im Forum seien Fotos vom »Klassensprecher« mit dem Trainer sowie Kapitän der Mannschaft zu sehen.

Der Manager nahm die Einladung unverzüglich an und meldete sich beim Forum mit seiner persönlichen E-Mailadresse eines Freemailers an. Dummerweise verwendete er für das Foren-Passwort genau das selbe Passwort, so dass die Angreifer sich sofort auf seinem privaten E-Mailzugang einloggen konnten. Sie versuchten sich dann auf dem dienstlichen Konto des Managers einzuloggen. Der Manager verwendete dort zwar ein anderes Passwort, allerdings bestand beim Mailzugang die Möglichkeit das Passwort zurückzusetzen, falls man es vergessen habe. Dazu wurde eine E-Mail an eine andere Mailadresse geschickt, in der sich ein Link zum zurücksetzen des Passworts befand. Diese andere E-Mailadresse war die private E-Mailadresse des Managers, zu der die Angreifer inzwischen Zugriff erlangt hatten. Sie kopierten sofort die Mail zum zurücksetzen des Passworts und nutzen diese Möglichkeit um Zugang zum dienstlichen Account des Managers zu bekommen. Nach dem sie diesen hatten, plünderten sie die Mailbox und installierten über eine gefälschte E-Mail ein Rootkit auf dem Büro-PC des Opfers. Über das Rootkit konnten sie jederzeit auf den PC zugreifen und Passwörter mitschneiden, wodurch unter anderem der Zugriff auf eine Forschungsdatenbank ermöglicht wurde.

Eine andere Möglichkeit ist es einen Trojaner im »Huckepack« auszuliefern. Dazu wird irgendeine interessante Software, etwa ein Softwareupdate, ein kostenloses Spiel oder ein Crack für ein kopiergeschütztes Spiel mit dem Trojaner versehen. Installiert der Benutzer dann die Wirts-Software reist die Schadsoftware, ähnlich wie die griechischen Soldaten im trojanischen Pferd, mit. Gelingt es der Wirts-Software vom Benutzer Root-Rechte zu erlangen, kann auch der Trojaner sofort mit diesen Root-Rechten installiert werden.

Physikalischer Zugriff: Eine recht einfache Methode Schadsoftware auf einem Rechner zu installieren, ist der physikalische Zugriff. So kann man recht einfach die Festplatte ausbauen, an ein anderes System anstecken und mit den Administratorrechten des anderen Systems auf der Opfer-Festplatte Schadsoftware installieren. Ebenso kann man das Opfersystem über einen anderen Datenträger booten (CD, DVD, USB-Stick) und dann von dort aus die Schadsoftware ausrollen. Derartige Szenarien sind beispielsweise in größeren Büros, Wohnungseinbrüchen oder Flughafenkontrollen denkbar.

Aktive Inhalte: Auch rein technische Sicherheitslücken können ausgenutzt werden. Es existieren viele verschiedene Angriffsvektoren in der technischen Dimension: Buffer Overflow, Heap Overflow, Stack Overflow, Session Riding, Format String Vulnerability, Cross Site Scripting, SQL-Injection, Parameter Injection, Direct Memory Access via Firewire etc. pp., der Phantasie des Angreifers sind hier keine Grenzen gesetzt. Ziel dieser Angriffe ist es, den Computer des Anwenders dazu zu bringen ein bestimmtes Programm auszuführen. Dieses Programm soll eine vorhandene Sicherheitslücke ausnutzen und dem Angreifer die Kontrolle über den Rechner ermöglichen.

Da die Erläuterung aller möglichen Varianten von Sicherheitslücken und Schadsoftware den Rahmen dieses Artikels sprengen würde und wohl auch in ihrer Gesamtheit nicht erfolgen kann, werde ich hier nur einen kurzen theoretischen Überlick geben. Bei einer SQL-Injection werden SQL-Befehle in einer URL manipuliert. Aktive Webseiten wie Webshops oder Diskussionsforen speichern ihre Daten häufig in einer Datenbank. Sucht ein Kunde beispielsweise nach einer externen USB Festplatte mit 2GB Kapazität, formuliert eine Webseite die Anfrage als SQL-Statement für die Datenbank und übergibt diese dann. Eine derartige Anfrage könnte so lauten: SELECT * FROM platten WHERE kapazitaet='2GB';. In der URL der Webseite werden der Parameter Anschluss als Variable übergeben[3] und von einer andere Seite ausgelesen und an die Datenbank übergeben. Man kann die URL nun manipulieren und dort beliebige SQL-Befehle einschleusen[4]. Mit dieser manipulierten Anfrage wird zuerst eine Festplatte mit 2GB Kapazität gesucht und anschließend ein Befehl zum löschen der Datenbank festplatte übergeben. Hat der Betreiber des Shops keine Sicherheitsmaßnahmen ergriffen, die derartige Anfragen abfangen, ist die Datenbank nach Ausführen der Anfrage leer. Statt Daten zu löschen können auch Angriffe gefahren werden die dem Angreifer Root-Rechte im Shop-System ermöglichen.

Ein Buffer Overflow ist eine vergleichsweise einfache und alte, aber immer noch sehr verbreitete und wirkungsvolle Sicherheitslücke. Dabei werden zu große Datenmengen in einen zu kleinen Speicherbereich geschrieben, so dass die Buffer-Grenzen überlaufen und die Speichersicherheit verletzt wird. Als mögliche Konsequenz kann die Return-Adresse einer Sub-Routine mit beliebigen Daten überschrieben werden. Abbildung 4 zeigt einen C-Schnipsel, bei dem die Variable `line` mit der Länge 23 reserviert wird. Anschließend werden Eingaben in die Variable geschrieben. Ist deren Länge größer als 23, wird mit den überstehenden Zeichen die Rücksprungadresse überschrieben. Diese kann man so auf einen beliebigen Wert setzen, so dass die Funktion an dieser Speicherstelle ausgeführt wird (vgl. Aleph One 1996).

Schutz vor Schadsoftware

Der Schutz vor Schadsoftware kann beliebig kompliziert werden, je nachdem auf welchem Level man dies tun möchte. Es gibt jedoch einige einfache Abwehrmethoden, die ich hier vorstellen möchte.

3 shop.foo/platten.pl?kap=2gb

4 shop.foo/platten.pl?kap=2gb'; DELETE * FROM latten;

```
1 void input_line()
2 {   char line[23];
3         if (gets(line))
4                 parse_line(line);
5 }
```

Abbildung 4: C-Code der einen Buffer Overflow erzeugt

Dateisystemintegrität: Jede Schadsoftware die eingeschleust wird, verändert das Dateisystem. Es werden entweder neue Dateien angelegt oder bestehende Dateien werden verändert. Daher ist es relativ einfach das Dateisystem auf solche Veränderungen mit bestimmten Programmen zu überwachen.

Man kann dazu zum Beispiel eine Liste mit den Eigenschaften der Dateien erstellen, wie sie `ls -l` erzeugt. Allerdings können die Eigenschaften der Dateien auch von der Schadsoftware manipuliert werden, daher ist es sinnvoll auch kryptographische Prüfsummen wie MD5 oder SHA1 zu erzeugen. Da der manuelle Abgleich der Dateien mit den Eigenschaftslisten aufwändig ist, gibt es spezielle Programme, die diese Aufgabe vereinfachen (vgl. Schumacher 2004, 2005).

Auf Unix-Systemen bietet sich dafür `mtree(8)`, Tripwire oder Aide an. Mtree ist standardmäßig auf allen BSD-Systemen und Abkömmlingen seit 4.3 Reno (NetBSD, FreeBSD, OpenBSD, Dragonfly, Mac OS X) enthalten. Es diente ursprünglich dazu bei Systeminstallation oder Updates die Integrität der kopierten Dateien zu prüfen. Wäre beim kopieren der Betriebssystemdateien beispielsweise der Kernel oder das Login-Programm fehlerhaft kopiert worden, könnte die Maschine nicht mehr adäquat booten und wäre damit nur schwer zu reparieren. Daher erstellt Mtree eine Datenbank mit den Eigenschaften aller Dateien und vergleicht diese mit einer weiteren Datenbank. Die Datenbank mit den Dateieigenschaften nennt man auch Fingerabdruck, er enthält unter anderem die Größer einer Datei, den Besitzer und die Gruppe, das Datum der letzten Änderung, Zugriffsrechte sowie verschiedene kryptographische Prüfsummen.

Nach der Installation des System kann man mit Mtree einen solchen Fingerabdruck des Systems erstellen. Die Datenbank, eine Textdatei, muss man vor Schadsoftware und Manipulationen schützen, beispielsweise durch eine kryptographische Signatur mittels GnuPG, in dem man sie auf eine CD brennt, die nicht wiederbeschreibbar ist, oder in dem man eine Prüfsummer der Datenbank erstellt und diese auswendig lernt. In regelmäßigen Abständen müssen das System und die Datenbank gegeneinander abgeglichen werden. Das jeweilige Programm zeigt dann Abweichungen in den zu überwachenden Dateien an. Allerdings bleibt es dem Anwender selbst überlassen, welche Gründe es für Abweichungen gibt, ob also beispielsweise ein Update für die Änderung verantwortlich ist oder doch eine Schadsoftware.

Ein weiteres Problem kann eine Manipulation des Betriebssystem-Kernels oder anderer Binaries durch die Schadsoftware sein.

So könnte ein Einbrecher ein manipuliertes MD5-Binary installieren, dass immer die korrekten Werte für eine ausgetauschte Datei anzeigt, damit wäre die Manipulation selbst nicht nachweisbar. Dies entspricht dem Grundsatz der Logik *ex contradictione sequitur quodlibet* - aus einem Widerspruch kann beliebiges geschlussfolgert werden. Da das System seine eigene Integrität nicht beweisen kann, muss es als widersprüchlich/falsch angesehen werden. Praktische Abhilfe schafft hier die Überprüfung durch ein 2. System, beispielsweise eine Boot-CD oder DVD, von der aus ein Live-System gebootet wird. Von diesem kann dann das eigentlich System dann überprüft werden.

Proaktive Integritätsprüfung: ist eine Ingegritätsprüfung, die ähnlich wie Mtree, Aide oder Tripwire Dateien auf Veränderungen prüft. Allerdings werden diese Veränderungen nicht im Nachhinein (also reaktiv) durchgeführt, sondern *bevor* eine Binary vom Kernel ausgeführt wird. Dazu muss der `exec()`-Pfad des Kernel entsprechend geändert werden, um eben jenen Dateivergleich durchführen zu können. Derartiges übernimmt beispielsweise das Systrace-Projekt von Niels Provos. Es klinkt sich in den Kernel ein und überprüft vor jedem Aufruf einer Datei deren Signatur mit einer Signatur-Datenbank. Stimmen die aktuelle und die gespeicherte Signatur überein, wird die Datei ausgeführt. Systrace kann auf OpenBSD, NetBSD, Linux und Mac OS X eingesetzt werden, wurde seit 2009 aber nicht mehr weiter entwickelt. Systrace beschreibe ich ausführlich in Schumacher (2007a,b,c, 2011d).

Zurückspielen des Betriebssystems: Eine weitere Möglichkeit um gegen Manipulationen am System vorzugehen ist es, nach potenziell gefährlichen Situationen ein Image des Systems zurückzuspielen. Ein Image ist ein Abbild einer Partition oder Festplatte. Ein solches Abbild kann man erstellen, nachdem man ein System neu eingerichtet hat und alle notwendigen Updates und Anwendungen installiert hat. Spielt man das Image wieder zurück auf die Festplatte, hat man genau den Zustand des Systems wieder, den es bei der Erstellung des Abbildes hatte. Somit kann man beispielsweise nach Kontrollen an Flughäfen oder Aufenthalten im Ausland nach der Rückkehr ein Image zurückspielen, welches einen Zustand vor der Abreise gesichert hat und damit vor möglichen Manipulationen liegt.

Eine einfache und freie Möglichkeit derartige Images zu erstellen und zurückzuspielen ist g4u von Hubert Feyrer. Dabei handelt es sich um eine extrem abgespeckte NetBSD-Version die via Diskette, CD oder USB-Stick bootbar ist und alle notwendigen Skripte mitbringt, um bequem Images zu erstellen und zurückzuspielen. Das Programm ist Open Source und unter `feyrer.de/g4u` zu finden.

Virenscanner: Virenscanner können unter Umständen Schadsoftware entdecken und neutralisieren. Allerdings muss man sich bewusst sein, dass Virenscanner und artverwandte Systeme nur sehr bedingt funktionieren können. Dies hängt mit der Funktionsweise von Virenscannern und artverwandten Systemen wie Intrusion Detection Systemen zusammen. Die einfachste Möglichkeit Schadsoftware zu erkennen und deren Ausbreitung zu verhindern ist die Suche nach Signaturen in Dateien. Bei Signa-

turen handelt es sich um spezifische Merkmale einer Schadsoftware, wie Dateigröße, Dateiname oder bestimmte Zeichenketten. Der Virenscanner verfügt über ein Lexikon in dem die Signaturen der bekannten Schadsoftware gesammelt wird. Bei einem Virenscan vergleicht er die Dateien im Dateisystem mit dem Lexikon und schlägt bei Übereinstimmung einer Datei mit dem Lexikon Alarm.

Diese Vorgehensweise ist relativ einfach zu implementieren, hat aber einen entscheidenden Schwachpunkt: Der Virenscanner kann nur Schadsoftware erkennen, deren Signaturen bereits im Lexikon vorhanden ist. Das heißt die Schadsoftware muss überhaupt erst einmal entdeckt und analysiert, die Signatur muss erstellt und an die entsprechenden Virenscanner auf den Rechnern ausgeliefert werden. Die Schadsoftware hat hier also allein schon aufgrund der zeitlichen Verzögerung einen immensen Vorsprung. Desweiteren gibt es noch die diagnostische Lücke: Schadsoftware kann nur analysiert werden wenn sie überhaupt erst einmal entdeckt wurde. Gelingt es der Schadsoftware unentdeckt zu bleiben, haben Virenscanner bzw. Virenanalysten keine Möglichkeit die Schadsoftware zu analysieren und damit zu bekämpfen. Dies gilt besonders für speziell angepasste Schadsoftware, die nur in einzelnen Fällen bei besonders ausgewählten Opfern eingesetzt wird.

Neben der einfachen Signaturerkennung werden noch andere Methoden zur Schadsoftware-Erkennung eingesetzt, zum Beispiel sogenannte heuristische Methoden. Dabei wird versucht mit begrenztem Wissen über ein System Mutmaßungen anzustellen und diese dann empirisch zu verifizieren. Im Bereich der Schadsoftwareerkennung wird beispielsweise das Verhalten von Software überwacht und auf verdächtige Anhaltspunkte untersucht. So sollte eine Virenscanner-Heuristik bei einer DLL-Datei[5], die keine Funktionen exportiert anschlagen. Trotzdem ist bis zur Analyse des Staatstrojaners kein Virenscanner in der Lage gewesen ihn zu erkennen – obwohl die Implementierung alles anderes als gelungen ist.

Die Probleme mit der Diagnose und Intervention im Bereich Schadsoftware habe ich ausführlich in Schumacher (2005, 2010a, 2011c) diskutiert.

Verschlüsselung der Festplatte Zeitungsberichten zufolge wurde der Staatstrojaner während einer Zollkontrolle am Flughafen München installiert. Dort wurde dem Überwachungsopfer der Laptop entzogen und in einen anderen Raum gebracht, nach kurzer Zeit erhielt das Opfer den Laptop zurück.

Hat ein Angreifer physikalischen Zugriff auf einen Rechner, kann er in der Regel allem mit der Festplatte und den enthaltenen Daten sowie der Hardware machen, was er will.

So kann er beispielsweise einen Hardware-Keylogger installieren, der jeden Tastenanschlag aufzeichnet und damit auch alle eingegebenen Passwörter verrät.

Desweiteren kann eine Festplatte in wenigen Minuten kopiert werden, so dass ihr Inhalt hinterher in Ruhe analysiert werden kann. Der entgegengesetzte Weg ist auch möglich: da ein Betriebssystem den eigenen Zugriffschutz für verschiedene Benutzer nur dann durchsetzen kann, wenn es selbst aktiv ist und läuft, kann es die enthaltenen

5 Siehe dazu die Staatstrojaner-Analyse in diesem Artikel

Daten nicht schützen, wenn die Festplatte in einem anderen Betriebssystem eingebunden wird. Ein Angreifer kann daher die Festplatte einfach ausbauen und an seinem eigenen Rechner einbinden - dort verfügt er selbst über Administrator-Rechte und kann daher alle beliebigen Daten abschöpfen und später auswerten. Er kann dann auch beliebige Programme installieren, die ebenfalls automatisch mit Administrator-Rechten laufen. Alternativ kann ein moderner Laptop oder Comuter auch über das CD/DVD-Laufwerk oder einen USB-Stick o.ä. gebootet werden. Dies dient dazu, Betriebssysteme überhaupt erst installieren zu können. Ein Angreifer muss daher eine Festplatte nicht zwangsläufig ausbauen, wenn er am Rechner über den USB-Port oder das CD-Laufwerk ein eigenes Betriebssystem booten kann. Von diesem Betriebssystem kann er dann die volle Kontrolle übernehmen und Daten auslesen oder einfügen.

Eine weitere Möglichkeit besteht bei Apple-Computern: Macs verfügen seit dem Powerbook 100 (vorgestellt 1991) über den sogenannten *Target Disk Mode*. Hält man beim Booten der Maschine die `T`-Taste gedrückt, fährt der Rechner nicht das Betriebssystem hoch, sondern die Firmware fährt in den TDM-Festplattenmodus. In diesem Festplattenmodus kann man die Maschine via SCSI, FireWire oder Thunderbolt an einen anderen Rechner hängen. Der Mac im TDM-Modus verhält sich dann wie eine einfache Festplatte und kann am anderen Rechner wie ein USB-Stick auch eingebunden werden. Da hier ebenfalls wieder das Betriebssystem des anderen Rechners die Rechte verwaltet, kann man alle Daten auf dem Mac im TDM-Modus beliebig manipulieren. Den Target Disk Mode kann man verhindern, wenn man im Extensible Firmware Interface (EFI) ein Passwort vergibt. Dieses wird dann beim Start abgefragt, wenn man nicht normal in das installierte Mac OS X booten möchte. Ohne das Passwort kann man dann auch nicht in den TDM-Modus booten.

Möchte man die Daten auf der Festplatte vor derartigen Zugriffen schützen, muss man auf Verschlüsselung zugreifen. Lediglich ein System, das die komplette Festplatte inklusive Betriebssystem verschlüsselt, kann die Daten zuverlässig schützen.

Erlangt ein Angreifer physikalischen Zugriff auf das System des Opfers, kann es die enthaltenen Daten weder auslesen noch verändern, da die Daten durch das Krypto-System verschlüsselt und damit geschützt sind.

Es existieren verschiedene Lösungen, um eine Festplatte komplett zu verschlüsseln.

Unter NetBSD ermöglicht es das *Cryptographic Device CGD*, eine Slice komplett auf Blockebene zu verschlüsseln und somit zu schützen. Wie CGD funktioniert und eingerichtet werden kann, habe ich ausführlich in (Schumacher 2006, 2011b) beschrieben.

Unter Mac OS X ab 10.7 kann FileVault 2 genutzt werden. Dieses ist, anders als FileVault 1, auch in der Lage die System-Partitionen zu verschlüsseln. Um Mac OS X zu booten, muss der Benutzer das FileVault-Passwort für die entsprechende Systempartition eingeben und ausserdem vom Systemadministrator als berechtigter FileVault-Mutzer eingetragen worden sein.

Für Windowsnutzer steht TrueCrypt als Freeware zur Verfügung. Neben Datenpartitionen und einzelnen Containern kann Truecrypt seit Version 5.0 auch die komplette Festplatte samt Windows-

Systempartition (in der Regel `C`) verschlüsseln. Erst nach Eingabe des Truecrypt-Passworts kann Windows hochfahren und die Daten stehen dem Anwender transparent zur Verfügung.

Hierbei gilt aber zu beachten, dass die Verschlüsselung nur davor schützt, dass ein Trojaner direkt installiert wird bzw. dass die Dateien direkt ausgelesen werden können. Im laufenden Betrieb sind die Daten bzw. das Betriebssystem nicht geschützt, da die Daten transparent entschlüsselt werden. Außerdem kann eine anderweitig eingeschleuste Schadsoftware mit Root-Rechten die Tastatureingaben überwachen und so auch eventuell eingegebene Passwörter auslesen.

Sichere Passwörter

Passwörter sind und bleiben ein attraktiver Angriffsvektor. Gelingt es einem Angreifer, das Administrator- oder Root-Passwort zu erbeuten, hat er die uneingeschränkte Kontrolle über den Rechner. Da immer noch viele Benutzer leicht zu erratende oder zu berechnende Passwörter verwenden, können selbst technisch wenig versierte Angreifer so Rechte auf einem Rechner erlangen. Daher ist es notwendig, sichere Passwörter zu verwenden.

Normalerweise tendieren Menschen dazu sich ein einfaches Passwort auszudenken, da sie es sich ja merken müssen. Also wählen sie den Namen ihres Hundes, Kindes, Autos oder den Wohnort, Geburtsjahr oder ähnliche naheliegenden Dinge. Dies machen sich Einbrecher zu nutze, in dem sie einfach versuchen das Passwort zu erraten. Außerdem nutzen Einbrecher ebenfalls Social-Engineering-Taktiken. *Social Engineering* kommt ursprünglich aus der Soziologie, bezeichnet im Hacker-Umfeld aber Methoden, mit denen man versucht Menschen zu hacken. Dazu kann sich ein Einbrecher bspw. am Telefon als Vorgesetzter oder Administrator ausgeben und versuchen dem Opfer im Gespräch Kontodaten zu entlocken. Oder man sammelt verfügbare Informationen über Mitarbeiter, bspw. indem man deren Webseite oder Blog durchsucht. Wenn sich auf der Webseite Bilder einer Katze befinden, liegt es doch nahe, dass der Name der Katze ihr Passwort ist, oder?

Um derartige Passwortangriffe durch raten zu erschweren, pausieren die meisten Systeme nach 3 oder 5 fehlgeschlagenen Anmeldeversuchen für einige Sekunden.

Hier nun einige Regeln für sichere Passwörter:
Verwenden Sie kein Passwort das erraten werden kann!

Benutzen Sie nicht den Namen ihrer Kinder, des Lieblingsvereines, Hundes oder ähnliches. Auch nicht sehr offensichtliche Wörter sind unbedingt zu vermeiden. Alles was sich in der näheren Umgebung des Rechners befindet ist ebenfalls eine schlechte Inspirationsquelle für Passwörter. The truth[6] is out there.

Ein Angreifer, der es auf Ihr Konto abgesehen hat, wird mit den oben beschriebenen Social-Engineering-Methoden ihre Umgebung aufklären. Also möglichst alle verfügbaren Informationen sammeln, die auf ein Passwort schließen lassen und diese Erkenntnisse in seine Einbruchsversuche ein-

6 Fox Moulder hatte in »Akte X« über seinem Monitor ein Poster hängen. Das Poster zeigte eine Fliegende Untertasse mit dem Untertitel »The Truth is Out There«. Als Dana Scully an Moulders Rechner mußte, tippte sie »Truth« als Passwort ein und war drin.

fließen lassen.

Verwenden Sie kein Passwort das in einem Wörterbuch steht!

Würde man Benutzernamen und Passwort im Klartext in einer Datei ablegen, könnte man das Passwort einfach auslesen. Damit dies nicht geschieht, wird es mit einer sogenannten Einwegfunktionen (engl. *one way hash*) verschlüsselt. Dazu wird das Passwort mit dieser speziellen mathematischen Funktion in einen sog. Hash umgewandelt. Es ist aber nicht möglich aus dem Hash das Passwort zu berechnen. Möchte sich jemand am System anmelden, gibt er sein Passwort an. Die Eingabe wird dann mit der entsprechenden Funktion in den Hash überführt, (Abbildung 5 zeigt ein Beispiel) anschließend werden beide Hashes verglichen. Wenn das Passwort korrekt ist, stimmen beide Hashes überein und der Benutzer wird am System angemeldet.

Gelingt nun ein Einbrecher in den Besitz der Passwort-Datei (`/etc/master.passwd`) kann er die Passwörter nicht einfach auslesen. Da die mathematischen Funktionen normalerweise sicher sind [7] kann aus den Hashes das Passwort nicht berechnet werden.

Was man aber trotzdem tun kann, ist selber derartige Hashes zu erzeugen und diese mit den erbeuteten zu vergleichen. Dazu kann man einen Generator oder ein Wörterbuch verwenden, wobei das Wörterbuch nichts anderes als eine simple Wortliste ist. Wird bei einem Benutzerkonto nun ein Passwort verwendet, das im Wörterbuch steht, wird es der Einbrecher finden. Daher dürfen keine Passwörter verwandt werden, die in einem Wörterbuch stehen. Auch wenn die Generierung von Passwort-Hashes dauert, kann man die Berechnung auf mehrere Rechner verteilen und die Ergebnisse der Falltürfunktion in Datenbanken ablegen.

Im Netz kursieren sogenannte Regenbogenlisten (engl. *Rainbow Tables*), dies sind Dateien, in denen Passworthashes bereits berechnet wurden. So kann man beispielsweise den Inhalt aus /usr/share/dict an einen Passwortgenerator verfüttern und die entstandenen Hashes zum schnellen Cracken von Passwortdateien verwenden. Dies ist möglich, da eine Hashfunktion über einer Zeichenkette immer den selben Hashwert erzeugt. Egal wann man auf welchen Rechner den SHA1-Wert von »08/15« berechnet, er wird immer gleich[8] sein. Da dies die beschriebenen Regenbogenlistenattacken ermöglicht, wurde eine Zufallskomponente in die Hasherzeugung eingefügt. Dieser sogenannte »Salt« (dt. Salz, weil er ja die Würze gibt) ist eine einfache Zeichenkette, die zufallsgeneriert und dem Passwort angehängt wird.

Verwenden Sie ein langes Passwort mit Groß- und Kleinschreibung, Zahlen und Sonderzeichen!

Nun kann man natürlich auf die Idee kommen ein Wort mit ein, zwei Zahlen zu erweitern, da soetwas nicht im Wörterbuch steht. Statt eines Wörterbuches kann man für die oben beschriebene Attacke aber auch einen Generator verwenden, der Wörter erzeugt oder ergänzt. So findet man auch Passwörter wie »Waldi1«, »Anja85« oder »Magdeburg*«.

Wenn ein Einbrecher nun alle möglichen Passwörter durchkalkulieren möchte, muss

7 (Das traditionelle crypt(1) bzw. DES ist nicht mehr sicher, aber nach dreißig Jahren ist das normal.)

8 baa4ef9f76419f3361826c618418ac6a441f7c6

er

$$verfügbare\ Buchstaben^{Passwortlänge}$$

Kombinationen berechnen.

Verwendet man nur die 26 Kleinbuchstaben a-z und das Passwort ist 5 Zeichen lang, sind dies $26^5 = 11\,881\,376$ Kombinationen. Verwendet man stattdessen 52 Groß- und Kleinbuchstaben (a-z, A-Z), 7 Umlaute, 10 Ziffern und 30 Sonderzeichen mit 10 Buchstaben, kommt man auf $99^{10} = 90\,438\,207\,500\,880\,449\,001$ Kombinationen.

Geht man nun davon aus das ein Rechner pro Sekunde 5 Kombinationen berechnen kann, benötigt man im ersten Falle $\frac{11881376}{(5*60*60*24)} = 27,5$ Tage, was durchaus ein realistischer Einbruchserfolg ist. Im zweiten Falle sind es aber schon 573555349 Jahrtausende[9] und selbst wenn statt 5 Kombinationen pro Sekunde 5000 berechnet werden, benötigt man immer noch 573555 Jahrtausende.

Diese Berechnung setzt natürlich voraus, das der Angreifer die Länge der Passwörter kennt. Ist dem nicht so, muss er alle möglichen Permutationen aller Längen durchrechnen.

Das Passwort muss geheim bleiben!

Eigentlich eine einfache Regel, die aber trotzdem oft missachtet und daher erfolgreich für Angriffe ausgenutzt wird.

Oft finden sich Passwörter auf einem Klebezettel unterhalb der Tastatur oder am Monitor versteckt, oder man teilt es Kollegen mit, damit diese irgendwas am Rechner tun können. Oder man verwendet überall das selbe Standard-Passwort, bspw. in igendwelchen Webforen und Freemailern.

Dies muss strikt unterbleiben. Es nutzt nichts sich ein starkes Passwort auszudenken, wenn es dann in die Weltgeschichte herausposaunt wird.

Verwenden Sie nicht überall das selbe Passwort! In der heutigen Zeit benötigt man in der Regel mehr als nur ein bis zwei Passwörter. Insbesondere in irgendwelchen Webforen oder für diverse Emailadressen, die als Spamfalle dienen, werden Passwörter benötigt. Sorgen Sie dafür das Sie nicht überall das selbe Passwort oder Passwortmuster verwenden. Wenn Sie sich bspw. an einem Webforum anmelden, müssen Sie meist noch eine Emailadresse angeben – verwenden Sie also nicht für beide das selbe Passwort, denn niemand kann garantieren daß das Passwort im Webforum geheim bleibt.

Benutzen Sie für ihre wichtigen Konten (private Email, PGP, Rechner etc.) starke Passwörter und verwenden Sie diese nirgends sonst. Notieren Sie sich gegebenenfalls Passwörter für »unwichtige«, selten genutzte Konten wie Foren oder ähnliches. Achten Sie aber darauf, daß niemand ohne weiteres an diese Notizen herankommt.

Insbesondere Webforen werden häufig Opfer automatisierter Attacken. So gelang es bspw. in ein privates Polizei-Forum einzubrechen. Die Passwörter für das Forum funktionierten auch bei einigen hinterlegten E-Mailadressen angemeldeter Benutzer.

Verwenden Sie also nicht überall das selbe Passwort!

Emulatoren, Virtualisierung und Sandboxen

Eine weitere Möglichkeit die Auswirkungen von Schadsoftware zu verringern ist der

9 Nicht einmal ein NetBSD-Rechner dürfte eine derartige Uptime erreichen ;-)

Einsatz von Emulatoren, Virtuellen Maschinen oder Sandboxen.

Emulatoren und Virtuelle Maschinen[10] dienen dazu bestimmte Hardware und/oder Software zu simulieren. So kann man über einen Emulator oder eine Virtuelle Maschine Microsoft Windows auf einem Apple Computer betreiben, auf einem Windows-PC einen Gameboy oder eine IBM S/360 oder auf einer NetBSD/Alpha eine VAX mit VMS emulieren.

Neben zahlreichen speziellen Anwendungen für Systemadministratoren, Programmierern oder Forschern gibt es auch einige Einsatzgebiete für normale Anwender. So kann man beispielsweise mit Parallels ein Windows, NetBSD oder weiteres Mac OS X auf einem Apple laufen lassen oder mittels Xen neben einem NetBSD auch ein FreeBSD emulieren. Man kann also auf einem Computer mehrere Betriebssysteme parallel nutzen. Abbildung 6 zeigt ein Mac OS X auf dem VirtualBox läuft. VirtualBox stellt wiederum zwei virtuelle Maschinen für Windows XP und g4u (NetBSD) bereit.

Eine Sandbox hingegen ist allgemein ein abgeriegelter Bereich, in dem bestimmte Prozesse oder ähnliches ablaufen können, ohne die Prozesse ausserhalb zu beeinflussen. So kann man die Gehege in einem Zoo, die Löwen und Giraffen voneinander trennen als Sandbox betrachten.

In der Informatik dienen Sandboxen dazu, Prozesse voneinander zu isolieren, um im Schadenfall den Schaden zu minimieren. So kann man auf einem Unix-System mittels `chroot(8)` eine Sandbox erstellen, in der ein Server oder Programm laufen kann ohne die Programme außerhalb zu gefährden. Googles Webbrowser Chrome setzt jeden Tab in eine Sandbox, um die laufenden Prozesse voneinander zu isolieren und so zu verhindern dass eine schädliche Webseite auf die Daten einer anderen Webseite zugreifen kann.

Egal ob man einen Emulator, eine Virtuelle Maschine oder eine Sandbox einsetzt, der Vorteil aller 3 Systeme ist die Isolation verschiedener Prozesse voneinander. Fängt man sich im VirtualBox-Beispiel eine Schadsoftware auf dem Windows XP ein, kann diese zwar das Windows ausspionieren, ohne weiteres jedoch nicht das NetBSD oder Mac OS X. Man kann daher recht bequem einzelne anfällige Prozesse wie Webbrowser voneinander isolieren. So ist es einfach möglich, mit VirtualBox oder Parallels ein zweites Windows oder Mac OS X einzurichten, dass man zum webbrowsen nutzt. Sensible Seiten (z.B. Internetbanking) kann man dann im Gastsystem aufrufen, wo die Eingaben von einem eventuelle aktiven Trojaner im Zweit-System isoliert sind.

Hierbei gilt allerdings zu beachten, dass auch ein Trojaner aus der Virtualisierung oder Emulation ausbrechen könnte, da die Virtuelle Maschine im Gastbetriebssystem in der Regel Root-Rechte benötigt. Gelingt es einem Trojaner eine Sicherheitslücke in der Virtuellen Maschine zu finden, kann er diese Ausnutzen und darüber auch Root-Rechte im Hostsystem erlangen.

Was muss eine »gute« Schadsoftware können?

Damit eine Schadsoftware erfolgreich sein kann, muss sie verschiedene Anforderungen erfüllen:

- Die Schadsoftware darf nicht ent-

10 Beides sind wesentlich komplexere Themen, und werden in diesem Artikel nur kurz vorgestellt.

deckt werden, dazu muss sie sich vor dem Betriebssystem, Virenscannern, Anti-Spyware, Intrusion Detection Systemen und sonstigen Sicherheitsfunktionen bzw. -programmen verstecken. Dazu dienen verschiedene Stealth-Techniken.

- Ein Trojaner soll in der Regel verschiedene Datenströme mitlesen und auswerten oder manipulieren, um beispielsweise Passwörter oder E-Mails vor der Verschlüsselung abzugreifen.
- Dateien auf der Festplatte nach bestimmten Daten durchsuchen und auswerten, dazu werden in der Regel Reguläre Ausdrücke verwendet.
- Tastatureingaben und Mausklicks sollen aufgezeichnet und Screenshots erstellt werden.
- Der Trojaner soll fernsteuerbar durch einen Command- & Control-Server (C&C) sein.
- Der Trojaner soll über eine Nachladefunktion verfügen um Updates und weitere Programmmodule einspielen zu können. Dies ist zwingend notwendig um Weiterentwicklungen im Anti-Malware-Bereich begegnen zu können.
- Die Kommunikation mit dem C&C soll möglichst unerkannt ablaufen, dazu sind Kryptographie und Steganographie notwendig.

Die hier genannten Fähigkeiten sind notwendig, wenn eine Schadsoftware einen Rechner erfolgreich ausspionieren soll.

Der Staatstrojaner

Dem Chaos Computer Club wurden mehrere Festplatten zugespielt, deren Inhalt forensisch analysiert wurde. Dazu wurde das dort eingesetzte Windows-Betriebssystem analysiert und einige verdächtige Dateien entdeckt. Diese Dateien wurden disassembliert.

Unter Disassemblieren versteht man den Vorgang um aus binär codierter Maschinensprache wieder für den Menschen lesbare Programmiersprache zu kompilieren. Es ist damit der Umkehrvorgang zum kompilieren bzw. dem Compiler. Normalerweise schreibt ein Programmierer ein Computerprogramm in einer Programmiersprache, die bestimmte Operationen des Computers abstrahiert und für den Menschen gut lesbar darstellt.

Abbildung 3 zeigt ein einfaches Programm in der Programmiersprache `C`, welches »Hallo Welt, es ist« und die aktuelle Uhrzeit ausgibt. Dieses Programm wird in einer Datei abgespeichert und dann mit einem Compiler wie `gcc` kompiliert. Der Compiler erzeugt eine eine Binär-Datei, die vom Betriebssystem ausgeführt werden kann. Binärdateien sind in der Regel von Menschen nicht lesbar, da sie direkt in Maschinencode übersetzt werden. Mit einem Disassembler wie `pedisassem` kann man Binärdaten wieder in eine lesbarere Assembler-Form übersetzen und diese interpretieren.

Im Ergebnis des Disassemblats fehlen aber Kommentare und die ursprünglichen Variablennamen, was die Analyse erheblich erschwert. Ein Programmierer vergibt in der Regel für Variablen sogenannte sprechende Namen, die den Zweck der Variable erklären. In einem Matheprogramm könnte man verschiedene Variablen `Pi`, `Umfang` oder `Volumen` nennen, um einfacher zu erkennen welchen Zweck sie erfüllen. Der Compiler ersetzt die Namen, da der Computer diese nicht benötigt. Im Disassemblat

stehen daher nicht mehr `Pi`, `Umfang` oder `Volumen` sondern beispielsweise `a`, `b` und `c`. Abbildung 4 zeigt die ersten Zeilen eines mit `ndisasm` erstellten Disassemblats des Hallo-Welt-C-Beispiels.

Bei den verdächtigen Dateien auf den untersuchten Festplatten handelt es sich um die Programmbibliothek `c:\windows\system32\mfc42ul.dll` sowie das Kernel-Modul `winsys32.sys`. Es handelt sich hierbei um eine unsignierte-32-Bit-Version handelt, die sich unter anderem an `explorer.exe` andockt. Geladen wird es über den Registry-Schlüssel `SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Windows\AppInit_DLLs`.

Die Analyse zeigt, dass es sich um eine Windows-DLL ohne exportierte Routinen handelt. Eine DLL ist eine *Dynamic Link Library*, also eine dynamische Programm-Bibliothek. Eine Programm-Bibliothek dient dazu, Funktionen die häufig verwendet werden, einem Programm zur Verfügung zu stellen. So gibt es beispielsweise eine Mathematik-Bibliothek die grundlegende Funktionen wie Sinus, Cosinus, potenzieren oder ratizieren zur Verfügung stellt oder eine Bibliothek um Drucker anzusteuern. Exportieren diese DLLs die Funktionenen, können sie in Anwendungsprogramme eingebunden werden und stehen dann dort zur Verfügung. Von daher ist eine DLL *ohne* exportierte Routinen sehr suspekt und vergleichbar mit einem Bibliotheksgebäude, in dem sich keinerlei Bücher befinden.

Kommunikation und Verschlüsselung des Trojaners

Der Trojaner kommuniziert über den TCP/IP-Port Nummer 443, also HTTP über SSL mit der IP-Adresse 207.158.22.134. Unter dieser Adresse lief ein Server in den USA, welcher als Command- & Control-Server agierte.

Die Kommunikation des Trojaners mit dem C&C-Server wurde zwar per AES »verschlüsselt«, allerdings im Electronic-Code-Book-Modus ECB und mit einem fest eincodierten Schlüssel. Beim ECB erzeugen gleiche Eingabedaten in der Verschlüsselung gleiche Ausgabedaten. Dies wiederum ermöglicht und vereinfacht eine statistische Kryptanalyse. Abbildung 7 zeigt zwei verschlüsselte Bilder und das Original. In Abbildung 7(b) wurde der ECB-Modus verwendet, man kann die zugrunde liegenden Muster des Bilder noch recht gut erkennen. In Abbildung 7(c) wurde ein verketteter Modus verwendet, das heißt das Verschlüsselungsergebnis eines Blockes wird an den nächsten Block rückgekoppelt, um das Ergebnis weiter zu verwischen. Wie man sieht, erkennt man in diesem Bild keinerlei statistisch signifikante Muster mehr.

Interessanterweise ist der Trojaner nicht in der Lage selber zu entschlüsseln, das heißt der Trojaner kann nur verschlüsseln. Das heißt aber auch, dass der Steuerverkehr zwischen C&C-Server und Trojaner nicht durch eine adäquate Verschlüsselung oder eine kryptographische Signatur abgesichert sein kann, denn dafür müssten beide Seiten zwingend ver- und entschlüsseln können.

Ein weiterer, noch interessanterer, Punkt ist aber, dass zur Authentifizierung des C&C-Servers lediglich ein Authentifizierungs-

string *fest* eincodiert ist. Der Server schickt das Banner `C3PO-r2d2-POE` an den Trojaner, der daraufhin den Server als Steuerungsrechner akzeptiert. Da der String fest eincodiert ist, muss man erstens davon ausgehen, dass alle ausgerollten Staatstrojaner den selben String zur Authentifizierung verwenden. Zweitens lässt sich der gesamte Netzwerkverkehr auf eben jene Zeichenkette hin überprüfen und ein eventuell vorhandener Staatstrojaner aufklären und enttarnen. Bei der eingesetzten Kryptographie handelt es sich um Bastelfunktionen primitivster Sorte, die jeglichen fundamentalen Grundlagen der IT-Sicherheit widerspricht.

Der Chaos Computer Club (2011b, S. 5) fasst die katastrophale Entwicklung des Trojaners so zusammen: »Wir sind hocherfreut, daß sich für die moralisch fragwürdige Tätigkeit der Programmierung der Computerwanze keine fähiger Experte gewinnen ließ und die Aufgabe am Ende bei studentischen Hilfskräften mit noch nicht entwickeltem festen Moralfundament hängenblieb«.

Noch pikanter ist die Funktion Screenshots zu erstellen. Laut einem Bericht der Tageszeitung[11] (taz), hat der Staatstrojaner auf einem infizierten System 60 000 Screenshots erstellt und an den C&C-Server gesendet. Dabei wurden die Screenshots im JPEG-Format gespeichert und versandt. Weiß ein Angreifer, dass derartig viele Screenshots verschickt wurden, kann er mit dem Wissen um das JPEG-Format (z. B. Magic Header) eine Known-Plaintext-Attacke gegen das kryptographische Verfahren starten und auch erfolgreich zu Ende bringen. Wäre der oben genannte Authentifizierungsstring nicht bekannt, könnte man so trotzdem und höchstwahrscheinlich erfolgreich die Verschlüsselung brechen und daraus den Schlüssel ableiten.

Die oben genannte Funktionalität, die Kommunikation müsste unerkannt und verschleiert ablaufen, wird nicht implementiert. Steganographie zur Verschleierung der Kommunikation wird überhaupt nicht eingesetzt, die verfügbaren kryptographischen Methoden sind so stümperhaft umgesetzt, dass die Bezeichnung »lachhaft« noch äußerst schmeichelhaft wäre. Der hier vorliegende Staatstrojaner hat dabei auf gesamter Linie versagt (vgl. Schumacher 2004, 2011c).

Durch die beschriebene »Authentifizierung« sind verschiedene Angriffe gegen einen Staatstrojaner möglich:

1. Der Angreifer fälscht einen Trojaner und gibt sich als Überwachter aus.
2. Der Angreifer gibt sich als C&C aus und steuert den Trojaner samt Nachladefunktion.
3. Trojaner ermöglicht nach der Übernahme höhere Zugriffsrechte.
4. Ein Staatstrojaner wird als Honeypot ausgerollt.

Eine weitere Analyse der Binärdateien bzw. des Disassemblats ergab unter anderem, dass die Dateinamen und Schlüssel in allen untersuchten Versionen gleich und hart eincodiert waren. Ein derartiges Vorgehen deutet daraufhin, dass es nur jeweils eine Version des Staatstrojaners gibt, die ausgerollt wurde. Allen Überwachungsopfern wurde offenbar die selbe Softwareversion untergeschoben.

Ebenfalls aufgedeckt wurde eine Update-Funktion. Der untersuchte Staatstrojaner

11 http://www.taz.de/Bayerischer-Datenschutzbeauftragter/!98735/, v. 05.08.2012

verfügte über die Möglichkeit sich selbst zu aktualisieren und weitere Module und damit Funktionalitäten nachzuladen. Dies ist aus technischer Sicht durchaus sinnvoll, jeder wirkungsvolle Trojaner benötigt eine derartige Funktionalität, zum einen um der Entwicklung in der Anti-Malware-Software (Virenscanner, IDS) begegnen zu können und zum anderen um den Rechner vollständig kontrollieren zu können.

Äußerst problematisch ist hierbei allerdings der rechtliche Rahmen in dem dies geschieht. Rechtlich wird zwischen Quellen-Telekommunikationsüberwachung (Quellen-TKÜ) und Online-Durchsuchung unterschieden[12]

> Im Folgenden wird daher der Begriff Online-Durchsuchung als Oberbegriff für die Online-Durchsicht und die Online-Überwachung verwendet. Unter der Online-Durchsuchung wird die verdeckte Suche unter Einsatz elektronischer Mittel nach verfahrensrelevanten Inhalten auf informationstechnischen Systemen verstanden, die sich nicht im direkten physikalischen Zugriff der Sicherheitsbehörden befinden, aber über Kommunikationsnetze erreichbar sind. Die zu erlangenden Inhalte sind nicht Gegenstand eines aktuellen Telekommunikationsvorgangs, wie etwa Dateien die nicht für einen elektronischen Versand bestimmt sind. Von der Online-Durchsuchung ist die Quellen-TKÜ zu unterscheiden, bei der Telekommunikationsinhalte und nicht sonstige, etwa auf der Festplatte abgelegte, Daten erhoben werden, die anderen rechtlichen Regelungen unterliegt.

Diese getroffene rechtliche Unterscheidung ist in der Praxis technisch nicht einzuhalten. Es ist für den Staatstrojaner (oder jeder andere Überwachungssoftware) egal ob es sich bei Dateien oder Datenströmen um Daten eines *aktuellen Telekommunikationsvorgangs* oder um *Dateien, die nicht für einen elektronischen Versand bestimmt sind* handelt.

Verwertbarkeit der Beweise und revisionssichere Protokollierung

Um die erbrachten »Beweise« des Staatstrojaners vor Gericht verwerten zu können, dürfen diese nicht manipuliert worden sein, sie müssen also revisionssicher protokolliert werden. Revisionssichere Protokollierung setzt einige Eigenschaften des Systems voraus, die nicht-trivial umzusetzen sind.

Der Staatstrojaner kann zwar alle Daten mitlesen und protokollieren, sofern er Root-Rechte hat. Sobald er aber Root-Rechte hat, kann er alle Daten auch manipulieren - ohne dass dies nachweisbar wäre. Die aufgezeichneten Daten sind somit nicht revisionssicher.

Ole Schröder berichtete aus dem Innenausschuß vom 19. Oktober 2011[13] folgendes:

> Durch eine revisionssichere Protokollierung sämtlicher Schritte ist sie [die Software] auch für den zuständigen Richter kontrollierbar.

12 Antworten des Bundesministerium des Innern auf den Fragenkatalog des Bundesministeriums der Justiz vom 22. August 2007, http://netzpolitik.org/wp-upload/fragen-onlinedurchsuchung-BMJ.pdf

13 Plenarprotokoll 17/132 des Deutschen Bundestages, 19. Oktober 2011, S. 15604

> Durch diese Protokollierung ist es auch nicht möglich, mal eben weitere Schadmodule nachzuladen.

Diese Aussage ist aus technischer Sicht schlicht unsinnig. Eine revisionssichere Protokollierung ist ohne weitere Maßnahmen elektronisch nicht möglich. Jeder elektronische Prozess und jedes elektronisch gespeicherte Datum – und damit jedes Programm und jede Datei – lässt sich ändern ohne dass dies auffällt oder gar revisionssicher zu protokollieren wäre. Erst recht, wenn ein Programm wie der Staatstrojaner eingesetzt wird der mit Root-Rechten läuft.

Es gibt einige Forschungsprojekte, die sich mit der revisionssicheren Protkollierung befassen. So wird es beispielsweise in Zukunft im Automobilbau immer wichtiger, die eingesetzten Mikroprozessoren zu überwachen und deren Funktionen sicher zu protokollieren. Schließlich wird es in absehbarer Zukunft zu Unfällen kommen, die auf Fehlfunktionen in der Elektronik zurückzuführen sind und deren Schuldfrage vor Gericht geklärt werden muss.

Wie bereits mehrfach erläutert läuft ein Trojaner in der Regel mit Root-Rechten, das heißt er kann alle Daten manipulieren und ändern. Man darf aber nicht verkennen, dass das Wirtssystem, also das Windows des Überwachungsopfers, ebenfalls mit Root-Rechten läuft und daher auch problemlos die Daten des Staatstrojaner manipulieren kann. Das heißt in der Praxis, dass ein Überwachungsopfer, wenn es den Trojaner enttarnt hat, diesem Trojaner jedes beliebige Datum vorgaukeln kann. Das Opfer kann also, wenn es möchte, dem Trojaner eine komplette Scheinrealität vorgaukeln.

In der »normalen« Überwachungspraxis, wie in dem oben genannten Beispiel der Automobilbranche, werden häufig Hardware-Lösungen eingesetzt, um Protokolle und Dateien halbwegs sicher zu protokollieren. So kann man beispielsweise recht einfach einen Nadeldrucker mit numerierten Leporello-Papier an einen Computer anschliessen und von diesem sofort die Systemlogs ausdrucken lassen. Steht der Rechner mit dem Papier und den Ausdrucken an einem physikalisch sicheren Ort, also einem Raum zu dem die Systemadministratoren keinen Zugang haben, können diese den Ausdruck des Protokolls nicht ohne weiteres manipulieren.

Eine weitere Möglichkeit ist es, die Protokolle sofort und direkt auf spezielle Datenträger zu schreiben, die nur einmal beschrieben werden können. Liegt eine Datei erst einmal auf einem solchen Medium, kann sie im Nachhinein nicht mehr manipuliert werden.

Ein weiteres Problem sind die Root-Rechte des eigentlich überwachten Rechners bzw. des Benutzers. Nicht nur der Staatstrojaner benötigt Root-Rechte um das System zu überwachen, auch das Betriebssystem selbst hat natürlich diese Root-Rechte. Damit kann es auch dem Staatstrojaner beliebige »Realitäten« vorgaukeln.

Fazit: Es gibt keine sichere Möglichkeit, Daten manipulationssicher durch einen Trojaner zu protokollieren. Revisionssichere Protokollierung ist schon mit speziell auf diesen Zweck ausgerichteter Soft- und Hardware äußerst problematisch. Mit einem Trojaner, der unerkannt in einem fremden System agieren, dabei unerkannt bleiben und die Daten auch noch gerichtsfest abschöpfen soll, ist es nicht möglich. Die mit dem Staatstrojaner erworbenen »Beweise« können vor einem Gericht nicht eingesetzt werden.

Nach der Veröffentlichung des ersten Berichts des CCC äußerte sich »IT-Spezialexperte und Staatssekretär« Ole Schröder dahingehend, dass der Staatstrojaner jetzt eine Verschlüsselung in beide Richtungen einsetzt (Chaos Computer Club 2011a, S. 2). Auch diese Aussage ist blanker Unsinng und zeugt von absolut fehlender Fachkompetenz. Verschlüsselung ist *immer* bijektiv, das heißt sie arbeitet in beide Richtungen, denn ohne Entschlüsselung der verschlüsselten Daten wären diese nicht mehr lesbar und könnten auch nicht ausgewertet werden. Es handelt sich bei dieser Aussage also nur um eine propagandistische Nebelkerze. Desweiteren wird immer noch eine Verschlüsselung im ECB-Modus verwendet und die Steuer-Kommandos wurden um eine fixe Zeichenkette erweitert. Dies führt immer noch zu festen Chiffraten, die relativ einfach analysiert werden können. Es ist sogar möglich, ein mitgeschnittenes Chiffrat mit einer Replay-Attacke wieder einzuspielen und so zu »wiederholen«. Die Update- und Execute-Funktionen sind immer noch vorhanden, ebenso wie die Screenshot-Funktion. Sie sind lediglich etwas besser verschleiert (vgl. Chaos Computer Club 2011a,b; Rieger 2011, 2012).

Literaturverzeichnis

Aleph One. (1996 November). Smashing The Stack For Fun And Profit. *Phrack*, *7*(49). Zugriff am 3. Dezember 2006, unter http : / / www . phrack . com / issues.html?issue=49&id=14

Belgers, W. (1993). UNIX Password Security.

Bundesamt für Sicherheit in der Informationstechnik (Herausgeber). (2006). Leitfaden IT-Sicherheit IT-Grundschutz kompakt. Zugriff am 16. Oktober 2006, unter http : / / www.bsi.de / gshb / Leitfaden / GS-Leitfaden.pdf

0ZAPFTIS – Teil 2. Analyse einer Regierungs-Malware. (2011a). Drei Jahre sind in der IT eine wirklich lange Zeit. Zugriff am 10. Dezember 2011, unter www . ccc . de / system / uploads / 83 / original / staatstrojaner-report42.pdf

Analyse einer Regierungs-Malware. (2011b). Zugriff am 10. Dezember 2011, unter www.ccc.de / system / uploads / 76 / original / staatstrojaner-report23.pdf

Neumann, J. v. (1966). *Theory of Self-Reproducing Automata* (A. W. Burks, Herausgeber). Champaign, IL, USA: University of Illinois Press.

Rieger, F. (2011). *Ein amtlicher Trojaner: Anatomie eines digitalen Ungeziefers. Wie der Staatstrojaner zerlegt wurde*. Zugriff am 9. Oktober 2011, unter http: / / www. faz . net / aktuell / feuilleton / ein - amtlicher - trojaner - anatomie - eines - digitalen-ungeziefers-11486473.html

Rieger, F. (2012). *Der Staatstrojaner lebt*. Zugriff am 6. August 2012, unter http : / / www.faz.net / aktuell / feuilleton / rechtsbruch - wird - tradition - der - staatstrojaner-lebt-11844056.html

Saake, G. & Sattler, K.-U. (2002). *Algorithmen und Datenstrukturen: Eine Einführung mit Java* (1. Auflage). Heidelberg: dpunkt.Verlag.

Schumacher, S. (2004). Einführung in kryptographische Methoden. Zugriff am 19. April 2004, unter http : / / www. cryptomancer. de / 21c3 / 21c3 - kryptograhie-paper.pdf

```
$ mtree -c
#          user: stefan
#       machine: balmung.net-tex.de
#          tree: /etc/X11/rstart/contexts
#          date: Sat Apr 25 22:24:08 2004

# .
/set type=file uid=0 gid=0 mode=0444 nlink=1 flags=none
.               type=dir mode=0755 nlink=2 time=1140283765.495867000
    @List       size=1306 time=1135575735.0
    default     size=1286 time=1135575735.0
    x           type=link mode=0755 time=1135576379.0 link=x11r6
    x11         type=link mode=0755 time=1135576379.0 link=x11r6
    x11r6       size=1525 time=1135575735.0
stefan@balmung {13}

```

Listing 1: von mtree erzeugte Eigenschaftenliste

```
$ mtree -L -c -K sha1,rmd160,gname,uname,mode -p /bin -p /sbin
        -X /etc/mtree.excl
#          user: stefan
#       machine: balmung.net-tex.de
#          tree: /
#          date: Sat Apr 25 22:29:08 2004

# .
/set type=file uname=root gname=wheel mode=0755 nlink=1 flags=none
.               type=dir nlink=22 time=1143916415.520797000
    .cshrc      mode=0644 size=940 time=1062527967.0 \
                rmd160=7559c5dc7215d8534c68a5e13bb8b6ffcab41d23 \
                sha1=fe2d675b3a4bc6febedde9f75ea8c7a9eac0c5e9
    .profile    mode=0644 size=519 time=1109063537.0 \
                rmd160=189f64161787c5c055bee918d9e8e7ab66c23742 \
                sha1=5a20146835d958b0cd702c401d28a3eb63a634e6
    boot        mode=0444 size=53820 time=1139597480.920000000 \
                rmd160=067a170f143c0b8e842dd3e0280b8af33bbd22dd \
                sha1=4f5af1b54b40c04ac9f277361eff7cd338d5cb86
    netbsd      size=8898275 time=1139697721.90897000 \
                rmd160=64cb2c68bce32d82e9ceca7b07e89a45f820f0ee \
                sha1=29847163d46c719ad6988fcd2f8b648b9d76e712
```

Listing 2: von mtree erzeugte Eigenschaftenliste (2)

```c
#include <stdio.h>
#include <time.h>

main()
{
    time_t now;
    time(&now);
    printf("Hallo Welt, es ist %.24s.\n", ctime(&now));
}
```

Listing 3: Hallo Welt in C

```
00000000  CF              iret
00000001  FA              cli
00000002  ED              in ax,dx
00000003  FE07            inc byte [bx]
00000005  0000            add [bx+si],al
00000007  0103            add [bp+di],ax
00000009  0000            add [bx+si],al
0000000B  800200          add byte [bp+si],0x0
0000000E  0000            add [bx+si],al
00000010  0D0000          or ax,0x0
00000013  0020            add [bx+si],ah
00000015  06              push es
00000016  0000            add [bx+si],al
00000018  8500            test [bx+si],ax
0000001A  2000            and [bx+si],al
0000001C  0000            add [bx+si],al
0000001E  0000            add [bx+si],al
00000020  1900            sbb [bx+si],ax
00000022  0000            add [bx+si],al
00000024  48              dec ax
00000025  0000            add [bx+si],al
00000027  005F5F          add [bx+0x5f],bl
0000002A  50              push ax
0000002B  41              inc cx
0000002C  47              inc di
0000002D  45              inc bp
0000002E  5A              pop dx
0000002F  45              inc bp
00000030  52              push dx
00000031  4F              dec di
...
```

Listing 4: Hallo Welt in C disassembliert mit `ndisasm`

Schumacher, S. (2005). Einbruchserkennung in Netzwerke mit Intrusion Detection Systemen und Honeypots. Zugriff am 21. November 2006, unter http://www.kaishakunin.com/publ/einbruchserkennung.pdf

Schumacher, S. (2006). Verschlüsselte Dateisysteme für NetBSD. *UpTimes*, *4*, 25–31. Zugriff am 7. November 2009, unter http://kaishakunin.com/publ/guug-uptimes-cgd_cfs.pdf

Schumacher, S. (2007a). Systemaufrufe mit Systrace steuern. *UpTimes*, *4*, 12–19. Zugriff am 7. November 2009, unter http://kaishakunin.com/publ/guug-uptimes-systrace.pdf

Schumacher, S. (2007b). Systeme mit Systrace härten. In German Unix User Group (Herausgeber), *Proceedings des GUUG Frühjahrsfachgespräches 2007* (Seiten 35–44). Freie Universität Berlin. Berlin: Lehmanns Media.

Schumacher, S. (2007c). Systeme mit Systrace härten. *Die Datenschleuder: Das wissenschaftliche Fachblatt für den Datenreisenden*, *#91*, 40–48. Zugriff am 3. Januar 2008, unter http://ds.ccc.de/pdfs/ds091.pdf

Schumacher, S. (2009a). Admins Albtraum: Die psychologischen Grundlagen des Social Engineering, Teil I. *Informationsdienst IT-Grundschutz*, *7*, 11–13. Zugriff am 22. Juli 2009, unter http://grundschutz.info/fileadmin/kundenbereich/Dokumente/Grundschutz_7-2009_11_13.pdf

Schumacher, S. (2009b). Admins Albtraum: Die psychologischen Grundlagen des Social Engineering, Teil II. *Informationsdienst IT-Grundschutz*, *8*, 8–9. Zugriff am 24. August 2009, unter http://grundschutz.info/fileadmin/kundenbereich/Dokumente/Grundschutz_8-2009_8_9.pdf

Schumacher, S. (2009c). Admins Albtraum: Die psychologischen Grundlagen des Social Engineering, Teil III. *Informationsdienst IT-Grundschutz*, *10/11*, 21–22.

Schumacher, S. (2009d). Psychologische Grundlagen des Social-Engineering. In German Unix User Group (Herausgeber), *Proceedings des GUUG Frühjahrsfachgespräches 2009* (Seiten 77–98). Hochschule Karlsruhe. Berlin: Lehmanns Media.

Schumacher, S. (2010a). Auf dem Weg zum Intrusion Detection System der nächsten Generation. In Team der Chemnitzer Linux-Tage (Herausgeber), *Chemnitzer Linux-Tage 2010: Tagungsband* (Seiten 19–24). Technische Universität Chemnitz. Chemnitz: Universitätsverlag.

Schumacher, S. (2010b). Psychologische Grundlagen des Social-Engineering. *Die Datenschleuder: Das wissenschaftliche Fachblatt für den Datenreisenden*, *#94*, 52–59. Zugriff am 10. Oktober 2010, unter http://ds.ccc.de/pdfs/ds094.pdf

Schumacher, S. (2011a). Die psychologischen Grundlagen des Social Engineerings. *Magdeburger Journal zur Sicherheitsforschung*, *1*, 1–26. Zugriff am 31. Januar 2011, unter http://www.wissens-werk.de/index.php/mjs

Schumacher, S. (2011b). Kryptographische Dateisysteme im Detail. In Team der Chemnitzer Linux-Tage (Herausgeber), *Chemnitzer Linux-Tage 2011: Tagungsband* (Seiten 39–46). Technische

Universität Chemnitz. Chemnitz: Universitätsverlag.

Schumacher, S. (2011c). Sicherheit messen: Eine Operationalisierung als latentes soziales Konstrukt. In S. Adorf, J.-F. Schaffeld & D. Schössler (Herausgeber), *Die sicherheitspolitische Streitkultur in der Bundesrepublik Deutschland: Beiträge zum 1. akademischen Nachwuchsförderpreis Goldene Eule des Bundesverbandes Sicherheitspolitik an Hochschulen (BSH)* (Seiten 1–38). Magdeburg: Meine Verlag.

Schumacher, S. (2011d). Systemcalls mit Systrace steuern. *Magdeburger Journal zur Sicherheitsforschung*, *2*, 125–138. Zugriff am 28. Dezember 2011, unter http://www.wissens-werk.de/index.php/mjs

Schumacher, S. (2012a). Timeo Danaos et dona ferentes: Zur Funktionsweise von Schadsoftware. *Magdeburger Journal zur Sicherheitsforschung*, *2*, 189–221. Zugriff am 20. November 2012, unter http://www.sicherheitsforschung-magdeburg.de/journal_sicherheitsforschung.html

Schumacher, S. (2012b). Zum Verhältnis von psychischen, sozialen und technischen Dimensionen des Einsatzes von IT-Systemen. Bachelor-Arbeit. Otto-von-Guericke-Universität Magdeburg.

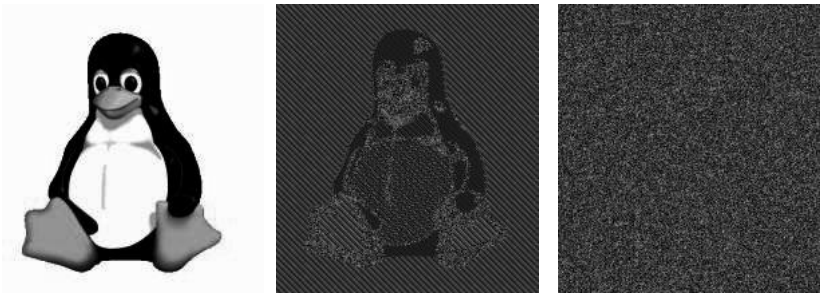
Stiebe, R. (2003). Theoretische Informatik für Ingenieur-Informatiker. Unveröffentlichtes Skript zur Vorlesung im SS 2003. Otto-von-Guericke-Universität Magdeburg.

Thompson, K. (1984 August). Reflections on Trusting Trust. *Communications of the ACM*, *27*(8), 761–763.

```
# grep test /etc/master.passwd
testuser:$sha1$7$7BWO8GFC$BGwFt9i6lnQ6jB5.3nb8YnP75xAF:1009:100::0:91231:\
Cliff Alistair McLaine, Orion-8, Terrestrische Raumaufklärung:/home/test:/bin/ksh
```

Abbildung 5: Beispiel aus /etc/master.passwd

Abbildung 6: VirtualBox emuliert 2 VM für Windows XP und g4u (NetBSD) auf Mac OS X.



(a) Das Original-Bild

(b) Das Bild im sogenannten ECB-Modus verschlüsselt, man erkennt klar die Muster

(c) Das Bild in sicherer Verschlüsselung

Abbildung 7: Verschiedene Verschlüsselungsmodi