Zur Funktionsweise von Schadsoftware

Stefan Schumacher
Magdeburger Institut für Sicherheitsforschung
Stefan.Schumacher@Sicherheitsforschung-Magdeburg.de
http://www.sicherheitsforschung-magdeburg.de

Schadsoftware ist nicht auf kommerzielle/Closed Source Betriebssysteme beschränkt, schließlich wurde der Morris-Wurm 1988 auf 4BSD und Sun-3-Systemen im Internet verbreitet. Der Vortrag zeigt daher die prinzipielle Funktionsweise von Schadsoftware am Beispiel des Staatstrojaners. Dazu werden Grundlagen der Computer- und Betriebssystemarchitektur vorgestellt und gezeigt wie Schadsoftware diese zur Infektion und Replikation auch auf BSD oder Linux ausnutzen kann. Desweiteren werden einige Gegenmaßnahmen kurz vorgestellt.

Es ist eine gekürzte Fassung von Stefan Schumacher. »Timeo Danaos et dona ferentes. Zur Funktionsweise von Schadsoftware«. In: *Informationstechnologie und Sicherheitspolitik. Wird der dritte Weltkrieg im Internet ausgetragen?* Herausgegeben von Jörg Sambleben und Stefan Schumacher. Reihe Sicherheitsforschung des Magdeburger Instituts für Sicherheitsforschung. Norderstedt: BoD, 2012, Seiten 45–78. ISBN: 978-3-8017-1817-6

1 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 (2011b, 2012b) vorgenommen. In der Diskussion um Schadsoftware und den Staatstrojaner 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

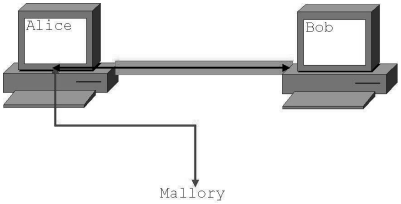
Erstellung manipuliert wurde. *Verfügbarkeit:* Dem Benutzer stehen Dienstleistungen, Funktionen eines IT-Systems oder auch Informationen zum geforderten Zeitpunkt zur Verfügung. *Authentisierung:* Bei der Anmeldung an einem System wird im Rahmen der Authentisierung die Identität der Person, die sich 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, Virus 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.

(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

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

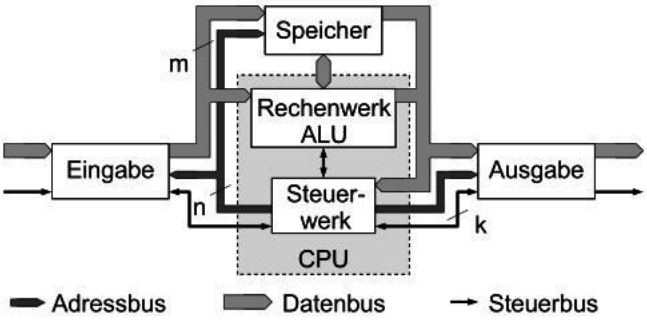


Abbildung 2: Von-Neumann-Architektur

ein einziger Speicher sowohl Befehle als auch Daten enthält. Abbildung 2 zeigt eine schematische Darstellung der Architektur.

Rechenwerk führt logische Verknüpfungen und Rechenoperationen mit den Daten
Steuerwerk regelt die Befehlsfolge und verschaltet Datenquelle und -ziel sowie weitere Komponenten
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

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

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

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 anderen 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.

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

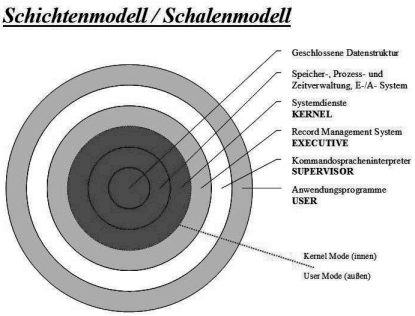


Abbildung 3: Schalenmodell (auch Schichten- oder Ringmodell)
angepasst nach http://de.wikipedia.org/w/index.php?title=Datei:Schichtenmodell.jpg

2 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.
Social Engineering, Phishing und Spear Phishing: Social Engineering ist eine Angriffsform die statt rein technischer Maßnahmen auch die Manipulation des Benutzers 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, 2010, 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 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.

Literaturverzeichnis

[Bun06] »Leitfaden IT-Sicherheit IT-Grundschutz kompakt«. De. In: (2006). Herausgegeben von Bundesamt für Sicherheit in der Informationstechnik. URL: http://www.bsi.de/gshb/Leitfaden/GS-Leitfaden.pdf (besucht am 16. 10. 2006).

[Sch09a] Stefan Schumacher. »Admins Albtraum. Die psychologischen Grundlagen des Social Engineering, Teil I«. In: *Informationsdienst IT-Grundschutz* 7 (2009), Seiten 11–13. ISSN: 1862-4375. URL: http://grundschutz.info/fileadmin/kundenbereich/Dokumente/Grundschutz_7-2009_11_13.pdf (besucht am 22. 07. 2009).

[Sch09b] Stefan Schumacher. »Admins Albtraum. Die psychologischen Grundlagen des Social Engineering, Teil II«. In: *Informationsdienst IT-Grundschutz* 8 (2009), Seiten 8–9. ISSN: 1862-4375. URL: http://grundschutz.info/fileadmin/kundenbereich/Dokumente/Grundschutz_8-2009_8_9.pdf (besucht am 24. 08. 2009).

[Sch09c] Stefan Schumacher. »Admins Albtraum. Die psychologischen Grundlagen des Social Engineering, Teil III«. In: *Informationsdienst IT-Grundschutz* 10/11 (2009), Seiten 21–22. ISSN: 1862-4375.

[Sch09d] Stefan Schumacher. »Psychologische Grundlagen des Social-Engineering«. In: *Proceedings des GUUG Frühjahrsfachgespräches 2009*. (Hochschule Karlsruhe). Herausgegeben von German Unix User Group. Berlin: Lehmanns Media, 2009, Seiten 77–98. ISBN: 978-3-86541-322-2.

[Sch10] Stefan Schumacher. »Psychologische Grundlagen des Social-Engineering«. In: *Die Datenschleuder. Das wissenschaftliche Fachblatt für den Datenreisenden* #94 (2010). Herausgegeben von Chaos Computer Club, Seiten 52–59. ISSN: 0930-1054. URL: http://ds.ccc.de/pdfs/ds094.pdf (besucht am 10. 10. 2010).

[Sch11a] Stefan Schumacher. »Die psychologischen Grundlagen des Social Engineerings«. In: *Magdeburger Journal zur Sicherheitsforschung* 1 (2011), Seiten 1–26. ISSN: 2192-4260. URL: http://www.sicherheitsforschung-magdeburg.de/uploads/journal/MJS-001.pdf (besucht am 31. 01. 2011).

[Sch11b] Stefan Schumacher. »Sicherheit messen. Eine Operationalisierung als latentes soziales Konstrukt«. In: *Die sicherheitspolitische Streitkultur in der Bundesrepublik Deutschland. Beiträge zum 1. akademischen Nachwuchsförderpreis Goldene Eule des Bundesverbandes Sicherheitspolitik an Hochschulen (BSH)*. Herausgegeben von Stella Adorf, Jan-Florian Schaffeld und Dietmar Schössler. Magdeburg: Meine Verlag, 2011, Seiten 1–38.

[Sch12a] Stefan Schumacher. »Timeo Danaos et dona ferentes. Zur Funktionsweise von Schadsoftware«. In: *Informationstechnologie und Sicherheitspolitik. Wird der dritte Weltkrieg im Internet ausgetragen?* Herausgegeben von Jörg Sambleben und Stefan Schumacher. Reihe Sicherheitsforschung des Magdeburger Instituts für Sicherheitsforschung. Norderstedt: BoD, 2012, Seiten 45–78. ISBN: 978-3-8017-1817-6.

[Sch12b] Stefan Schumacher. *Zum Verhältnis von psychischen, sozialen und technischen Dimensionen des Einsatzes von IT-Systemen*. Bachelor-Arbeit. Otto-von-Guericke-Universität Magdeburg, 2012.

[Sti03] Ralf Stiebe. *Theoretische Informatik für Ingenieur-Informatiker*. Unveröffentlichtes Skript zur Vorlesung im SS 2003. Otto-von-Guericke-Universität Magdeburg. 2003.