Über den Aufwand, Malware auf einem privaten PC zu installieren – Wie einfach lassen sich Virenscanner und *Personal Firewalls* umgehen?

Ulrich Greveler[1] und Christian Puls[2]

Kurzfassung:
Aktuelle Virenscanner und Personal Firewalls stellen nur einen geringen – leicht zu umgehenden – Schutz dar, wenn der unbedarfte Anwender eine Programmdatei ausführt, die zur Kompromittierung des PCs erzeugt wurde. Dies wird anhand eines empirisch gewonnen Modells eines „typischen" Privatanwender-PCs in einer Laborumgebung verifiziert. Der Angreifer benötigt zur Erstellung der Malware weder besondere Ressourcen noch überdurchschnittliche technische Fähigkeiten. Im Lichte dieser Erkenntnis ist die Effektivität der üblichen Sicherheitshinweise für Privatanwender fraglich, da die Installation der weit verbreitenden Sicherheitstools ein möglicherweise trügerisches Sicherheitsgefühl vermittelt.

Stichworte: Malware, Virenscanner, Personal Firewall, Sicherheitshinweise

1. Einführung

Der Beitrag untersucht, inwieweit ein „typischer" Privatanwender-PC mit geringem Aufwand mithilfe einer *Malware* kompromittiert werden kann. Dazu wird zunächst ein empirisches Modell gewonnen, das als Grundlage eines Laboraufbaus dient. Die Kompromittierung des Systems wird im Labor praktisch verifiziert, wobei aktuelle Sicherheitstools (Virenscanner und Personal Firewalls) zu überwinden sind. Unter Kompromittierung wird im Kontext dieses Beitrages die Installation einer komfortablen *Backdoor* bzw. eine Installation eines *Remote Administration Tools* verstanden, das einem Angreifer umfangreiche Möglichkeiten bietet, Daten zu lesen bzw. zu verändern und das System für diverse Zwecke zu manipulieren. Voraussetzung für die hier verwendete Angriffsmethode ist eine entscheidende Fehlhandlung des Opfers, das eine (nicht vertrauenswürdige) Datei ausführt und sich auf die Schutzmechanismen der genannten Sicherheitstools verlässt.

Privatanwender verwenden – nicht zuletzt wegen der Sensibilisierung für Themen der IT-Sicherheit – i. A. Software-Werkzeuge mit Sicherheitsfunktionalität (Virenscanner mit regelmäßig aktualisierter Signaturdatenbank, zusätzlich die Installation einer sog. *Personal Firewall* zum Schutz vor netzwerkbasierten Angriffen bzw. Datenflüssen), die zur Durchsetzung eines mittleren Sicherheitsniveaus beitragen sollen. Tatsächlich können wir aber mit diesem Beitrag Ergebnisse vorlegen, die zeigen, dass bereits ein Angreifer mit durchschnittlichen Fähigkeiten in der Lage ist, die Schutzwirkung dieser Sicherheitstools auszuhebeln und den PC zu kompromittieren.

Zunächst werden die Ergebnisse einer empirischen Untersuchung präsentiert, die Daten über den zu erwartenden „Normalfall" eines Privatanwender-PCs liefert. Zwar existieren allgemeine Erwartungen, wie ein PC üblicherweise konfiguriert und in sicherheitstechnischer Hinsicht softwareseitig ausgestattet ist, diese sollen aber zunächst in einer Weise fixiert werden, dass ein einfaches Referenzmodell eines privaten PCs entsteht, das für die weiteren Untersuchungen herangezogen werden kann.

---

[1] Fachhochschule Münster, Labor für IT Sicherheit, 48565 Steinfurt, greveler@fh-muenster.de
[2] Advanced Nuclear Fuels GmbH, 49811 Lingen, christian.puls@areva.com

Wir konnten nach dem empirisch ermittelten Modell (erwartungsgemäß) davon ausgehen, dass der typische privat genutzte PC nicht völlig ungeschützt ist; beispielsweise gehört die Verwendung von Virenscannern und *Personal Firewalls* (inkl. der so genannten Malware-Detektoren) heutzutage zum üblichen und behördlicherseits empfohlenen Nutzerverhalten. Gleichzeitig muss jedoch konstatiert werden, dass ein privater Anwender sicherheitskritische Fehler begeht, z. B. eine ihm auf geschickte Weise zugespielte ausführbare Datei öffnet und potentielle Malware damit in sein System einbringt. Dass über diesen Weg Schadsoftware installiert werden kann, ist unstrittig; interessant ist jedoch die Frage, ob die Schutzmechanismen hier nennenswerten Widerstand bieten.

Wir können erwarten, dass mit hohem technischem bzw. technisch-organisatorischem Aufwand (z. B. physikalischer Zugang zum Gerät) eine erfolgreiche Kompromittierung des privaten PCs wahrscheinlich wird (dies war ein zentraler Aspekt der Diskussion um die sog. *Onlinedurchsuchung* [VB07]). Wir wollen jedoch untersuchen, inwieweit eine Einbringung bereits mit geringem Aufwand möglich ist, d. h. insbesondere wie gefährlich Angreifer bereits werden können, die über durchschnittliche technische bzw. finanzielle Ressourcen verfügen und wie stark die Schutzwirkung der üblicherweise verwendeten Sicherheitstools ist. Die betrachteten Tools sind insbesondere für den privaten Anwender konzipiert, der im Einzelfall sicherheitskritische Fehlentscheidungen treffen kann, da er – im Gegensatz zum technisch versierten und im Hinblick auf IT-Sicherheit zum Experten ausgebildeten Anwender – nicht bei jeder Einzelentscheidung die Sicherheitsrelevanz erkennt; beispielsweise überwachen einige dieser Tools die Ausführung von Dateien, indem beim Öffnen eine Überprüfung auf Virensignaturen erfolgt. Wird eine Infektion festgestellt oder lassen heuristische Untersuchungen eine Gefährdung als wahrscheinlich erscheinen, wird der Anwender gewarnt bzw. aktiv an der (mutmaßlichen) Fehlentscheidung der Programmausführung gehindert.

Heuristische Algorithmen untersuchen Funktions- und Prozessaufrufe der verdächtigen Applikation auf spezielle Anomalien, welche in „harmlosen“ Anwendungen nicht zu finden sind. Die Identifizierung der Anomalien anhand des Programmcodes erfolgt in der Regel durch Erkennung verdächtiger Byte-Folgen. Eine Alternative zu dieser statischen Analyse des Programmcodes ist die dynamisch-heuristische Analyse, die das Programm zum Schutz des Systems in einer gesicherten, oft emulierten Umgebung ausführt. [LH06]

2. Ergebnisse der empirischen Untersuchungen, Modelle

Ergebnisse der hier im Rahmen einer Abschlussarbeit[3] durchgeführten empirischen Untersuchung bieten folgenden Rahmen für das Referenzmodell eines (anzugreifenden) privaten PCs: Es liegt eine *Windows*-basierte Systeminstallation (*XP* oder *Vista*)[4] vor. Der Großteil der Umfrageteilnehmer setzt auf seinem System ein oder mehrere Sicherheitswerkzeuge ein. So gaben ca. 77 % an, einen *On-Access*-Virenscanner, also einen Virenscanner, welcher bei jedem Dateizugriff die entsprechende Datei überprüft, zu verwenden. Weitere 10 % verwenden einen *On-Demand*-Virenscanner, einen Virenscanner, welcher nur auf Anforderung aktiv wird. Ebenfalls 10 % geben an, keinen Virenscanner zu benutzen und 3 % machten hierzu keine Angabe. Eine *Personal Firewall* (*PFW*) wird von 62 % der Befragten eingesetzt, 33 % nutzen keine Personal Firewall, 5 % machten hierzu keine Angabe. Häufigste Nennungen im Bereich Virenscanner waren *Avira*, *Kaspersky*, und *Norton*, im Bereich Personal Firewall *Zone*

[3] Masterarbeit von Christian Puls (B.Sc.), Fachhochschule Münster, September 2008

[4] Nur etwa 20% der Befragten gaben an, kein Microsoft-OS zu verwenden (Linux-Derivate führten vor Mac OS X).

Alarm und *Sygate*. Dieses Ergebnis liegt im Erwartungsrahmen und deckt sich mit Untersuchungen zu Marktanteilen [AS08].

Problematischer als die Gewinnung eines Referenzmodells für den anzugreifenden Rechner ist die Modellierung eines typischen Angreifers, der über durchschnittliche technische Fähigkeiten und Hilfsmittel verfügt. Hierzu liegen den Autoren auch keine empirischen Daten vor (eine Umfrage über einschlägige Internetforen wäre zwar grundsätzlich denkbar, jedoch wäre eine Repräsentativität kaum erreichbar, eine Selbsteinschätzung als *durchschnittlich befähigt* in diesem Umfeld zudem fraglich). Wir gehen daher hilfsweise davon aus, dass der Angreifer die kompromittierende Software nicht selbst erstellt, jedoch über Suchmaschinen in der Lage ist, frei verfügbare Software zu finden, in einfacher Weise zu modifizieren und ggf. zu kompilieren (einfache Programmierkenntnisse werden demnach vorausgesetzt); zudem kann er Virenscanner und Firewallsoftware zu eigenen Testzwecken installieren und benutzen.

3. Laborergebnisse

Für die Untersuchung wurde ein Demonstrator auf Basis von *Back Orifice 2000* entwickelt, der (aus Angreifersicht) eine umfangreiche Funktionalität bereitstellt. *Back Orifice 2000* (kurz: *BO2k*) basiert auf dem Remote Administration Tool (RAT) *Back Orifice*, welches im August 1998 auf der Konferenz *DEFCON 2* von der Hackergruppe *Cult of the Dead Cow* vorgestellt wurde und zu den ersten frei verfügbaren RATs gehörte. Der Name *Back Orifice* spielt auf Microsofts seinerzeit verfügbare Produktlinie *Back Office* an. Die Funktionalität umfasst neben dem Übertragen von Dateien die Übermittlung von Tastatureingaben, *Keylogging*, eine Maussteuerung und das Anfertigen von Screenshots.

Der von uns modellierte Angreifer ist (nach Nutzung einer Suchmaschine) in der Lage, *BO2k* herunterzuladen und so zu modifizieren, dass es unbemerkt als Bestandteil einer harmlos erscheinenden Anwendung installiert wird (wir wählten für den Demonstrator als Beispiel eine Maustreiberinstallation). Entsprechende Tools hierfür sind frei verfügbar, so bietet zum Beispiel das *Windows*-eigene Bordmittel *IExpress 2.0* die Möglichkeit, Anwendungen zu bündeln und versteckt ausführen zu lassen. Durch Einsatz frei verfügbarer Software[5] lässt sich die Modifikation weiter verschleiern.

Allerdings sind dem Angreifer zunächst Grenzen gesetzt, da sowohl die Virenscanner als auch die Personal Firewall den Anwender bei Ausführung des Programms warnen. Grund dafür ist, dass das bekannte Tool *BO2k* in den Virensignaturen[6] verzeichnet ist und dass das Tool eine Netzwerkverbindung bzw. einen Serverdienst initiiert, ein Verhalten, das von der PFW bemerkt würde.

Der Angreifer steht nun vor der Herausforderung, die beiden Schutzmechanismen zu umgehen. Die weitere Untersuchung bezieht sich auf die Frage, ob dies ohne erheblichen Aufwand möglich ist. Die erfolgreiche Täuschung des ahnungslosen Nutzers, der eine scheinbar gewöhnliche Treiberinstallation vornimmt, wird zunächst vorausgesetzt.

[5] Ein Beispiel wäre hier die Freeware RessourceHacker, welche durch eine graphische Oberfläche entsprechend leicht einzusetzen ist.

[6] Korrekter wäre die Bezeichnung *Malware-Signatur*, die sich aber nicht durchgesetzt hat.

3.1 Virenscanner

Eine Umgehung des Virenscanners[7] (die Untersuchung wurde für 36 verschiedene, aktuelle Scanner vorgenommen) erwies sich – trotz geringer Aufwände seitens des Angreifers – als möglich.[8] Bereits das Neukompilieren des Quellcodes senkte die Erkennung via Signatur-Datenbank auf 16 von 36 Scannern. Eine Portierung auf eine projektfremde Entwicklungsumgebung führte zur Erkennung von nur noch 5 Scannern, dies war jedoch mit funktionellen Einbußen des RAT verbunden. Einige Scanner erkannten zwar die Malware nicht mehr aufgrund ihrer Signatur, gaben aber nach Anschlagen einer Malware-Heuristik eine allgemeine Warnung vor der „Gefährlichkeit" der Software aus.

Um sowohl die Virensignaturerkennung als auch die heuristische Erkennung zu vermeiden, wurde ein einfaches Hex-Editor-Verfahren angewandt. Zur Planung einer Intervallschachtelung wurde zunächst die erste, dann die zweite Hälfte des Binärcodes byteweise mit dem Wert *00h* überschrieben, um eine relevante Offset-Position in der Datei zu ermitteln; dieser Vorgang wurde für die Hälften sukzessive wiederholt bis kurze Fragmente (wenige oder einzelne Bytes) ermittelt wurden, die zur Erkennung durch den Scanner relevant waren. Nachdem diese Fragmente identifiziert wurden, konnte durch Modifikation des Binärcodes eine Erkennung durch den Virenscanner unterbunden werden. Anschließend wurde die Funktionalität des RAT getestet, um sicherzustellen, dass wesentliche Backdoor-Funktionen erhalten blieben (um den syntaktischen Aufbau des ausführbaren Codes nicht zu beschädigen, muss u. U. statt eines Nullbytes ein anderer Wert zum Überschreiben verwendet werden). Bereits diese generische Vorgehensweise führte zu nutzbarer, nicht erkannter Malware (auf *BO2k*-Basis). Da diese Vorgehensweise unmittelbar den Binärcode verwendet, muss der Angreifer nicht zwingend im Besitz des Quellcodes sein. Jedoch lässt sich so keine genaue Vorhersage treffen, welche Auswirkungen die Veränderung haben wird. Die Veränderung des Timestamps oder der Basis-Adresse ist im Normalfall problemlos möglich, ohne einen Funktionsverlust in Kauf nehmen zu müssen. Wird hingegen z. B. der Zeiger auf den Code-Abschnitt modifiziert, so wird die Anwendung mit großer Wahrscheinlichkeit nicht mehr funktionieren.

Erweiterte man die Grundmethode der Intervallschachtelung jedoch um Modifikationen im Quellcode und lokalisierte die „Problemstelle" dort als Programmzeile (meist handelte es sich beim identifizierten Binärcodefragment um einen Funktionsbezeichner einer DLL-Funktion) konnte durch Verwendung eines *Alias*-Bezeichners die Erkennungsproblematik ohne funktionelle Einbußen gelöst werden. Dazu sind einfache Programmierkenntnisse auf Seiten des Angreifers erforderlich. Im vorliegenden Fall konnte mit Hilfe eines frei verfügbaren PE-Betrachters[9] der gefundenen Offset-Position ein Eintrag in der *Import Name Table (INT)*[10] zugeordnet werden, welcher durch Verwendung eines schon zuvor genannten Alias-Bezeichners umgangen werden konnte. Im Detail war die Funktion *OpenSCManager* aus der DLL *ADVAPI32* Auslöser der Heuristik-Warnung. Möchte oder muss man die Funktion trotzdem nutzen, so besteht die Möglichkeit, Funktionen aus DLLs dynamisch zu importieren. Hierzu bietet die *Windows-API* die Funktionen

---

[7] Die hier untersuchten Virenscanner umfassen mehrheitlich eine zusätzliche *Anti-Spyware*-Funktionalität.

[8] Zu bemerken ist hier, dass ein unmodifiziertes BO2k bei 34 von 36 Scannern zunächst als Malware bzw. RAT erkannt wurde. Verwendet wurde hier die Serverversion v1.15 aus der v1.16 Distribution.

[9] Ein PE-Betrachter bereitet den Code entsprechend des Portable-Executable-Formats auf.

[10] Die Import Name Table enthält Einträge aller Variablen oder Funktionen einer anderen ausführbaren Datei (so auch DLL) welche in der entsprechenden Anwendung genutzt werden.

HMODULE WINAPI LoadLibrary(__in LPCTSTR lpFileName);
und
FARPROC WINAPI GetProcAddress(
__in HMODULE hModule,
__in LPCSTR lpProcName
);

zum Laden einer DLL und dem Ermitteln der Adresse einer Funktion aus dieser DLL an. Wird die Funktion dynamisch geladen, so kann sie weiterhin genauso verwendet werden wie die direkt aufgerufene Funktion. Nun wird die Funktion jedoch nicht mehr in der *Import Address Table (IAT)* oder der *Import Name Table (INT)* geführt. Ist man nicht im Besitz des Quellcodes, so lässt sich der genannte Funktionsaufruf im Quellcode durch einen gleichlangen ebenfalls im Programm verwendeten Funktionsnamen ersetzen. Somit schlägt die Heuristik auch in diesem Fall nicht an, jedoch werden entsprechende Funktionen, welche die genannte Routine aufrufen, nicht mehr korrekt funktionieren.

3.2 Personal Firewall

Die zweite Hürde, die der Angreifer überwinden muss, wenn er nicht vom PC-Besitzer entdeckt werden möchte, ist die *Personal Firewall* (PFW). PFWs sollen i. A. verhindern, dass unzulässige Netzwerkverbindungen nach außen aufgebaut werden bzw. dass ein unerwünschter Serverdienst („offener Port“) gestartet wird. Das PFW-Konzept wird in der Literatur kritisch betrachtet, da es dem Nutzer eine oft nur scheinbar vorhandene Sicherheit vermittelt, die in der Vergangenheit in vielen zitierten Fällen umgangen werden konnte.[11]

Auch dem von uns modellierten Angreifer (ohne besondere Fähigkeiten) gelingt die Umgehung der PFW mit einem einfachen Trick: Er bestätigt automatisiert das Dialogfenster mit der Firewall-Warnung und lässt das Öffnen der Backdoor zu. Die PFW-Funktionalität wird also nicht im engeren Sinne unterdrückt; der Nutzer bemerkt sie aber nicht mehr, da die Bestätigung ohne sein Zutun (innerhalb von Sekundenbruchteilen) erteilt wird. Ein aufmerksamer Benutzer kann dies unter günstigen Umständen als Bildschirmflackern wahrnehmen; diese Umgehungsmethode ist daher nicht als perfekt zu werten, es ist jedoch nicht anzunehmen, dass ein großer Teil der privaten PC-Benutzer (nach unserem Modell) ein kurzes Flackern als Hinweis für eine Kompromittierung ansieht, sofern nicht entsprechende aktuelle Warnungen in der Öffentlichkeit kursieren. Diese Vorgehensweise wurde anhand aktueller Versionen der PFWs *ZoneAlarm* (Version 7.0.483) und *Sygate* (Version 5.5 *Standard*) erfolgreich demonstriert. Details werden im Folgenden beschrieben.

Die Umgehung der PFW macht sich das *Windows Message-System* zu Nutze. Unter Windows besitzt jedes Fenster einen eigenen individuellen Window-Handle, über den es gezielt angesprochen werden kann. Diese Fenster verfügen über keine expliziten Funktionsaufrufe, um Benutzereingaben zu erhalten, sondern warten darauf, dass das Betriebssystem Eingaben an

[11] „Ein Problem von Desktop-Firewalls [wird] deutlich: Sie wiegen den Anwender in Sicherheit, die sie jedoch letztlich nicht bieten können. Bis jetzt ließ sich mit etwas Phantasie noch jede Personal Firewall umgehen und Daten unerkannt vom Rechner ins Internet senden. Außerdem erhöht jede zusätzliche Software die Komplexität eines Systems und somit auch dessen Fehleranfälligkeit.“ Zitiert aus einer Heise-Meldung vom 18.07.2006 12:55.

sie weiterleitet. Wird nun ein Ereignis ausgelöst, wird eine Nachricht an das jeweilige Fenster versandt. Ereignisse können durch eine Vielzahl von Aktionen sowohl durch Nutzer-Interaktion als auch durch das System ausgelöst werden. Beispiele sind das Drücken einer Taste, die Bewegung der Maus oder z. B. das Sichtbarwerden eines Fenster. Auch Anwendungen können Nachrichten erzeugen und an Fenster senden. Windows unterhält hierfür zwei Arten von Nachrichten-Warteschlangen. Während die *System Message Queue* nur einmal auf dem System existiert, kann es mehrere *Thread-spezifische Message Queues* geben. Beide Warteschlangen werden dabei als *First-In-First-Out* (FIFO) Warteschlangen betrieben. Wenn ein GUI-Thread erstmalig einen *User*- oder *Windows-Graphics-Device-Interface* (GDI)-Aufruf tätigt, wird vom System eine Message Queue für diesen Thread erstellt. Wird beispielsweise die Enter-Taste vom Benutzer gedrückt, so erzeugt das Drücken und das Loslassen der Taste jeweils eine Nachricht, welche in die *System Message Queue* eingereiht wird. Diese Nachrichten werden nacheinander vom *System Dispatcher* aus der Queue gelesen und an eine entsprechende *Thread-spezifische Message Queue* weitergeleitet. An welche Message Queue die Nachricht weitergeleitet wird, hängt beispielsweise bei der Tastatureingabe vom aktuellen Fokus bzw. vom aktuellen Vordergrund-Fenster ab. Der Thread, zu dem das Fenster gehört, liest Nachrichten nacheinander aus der *Thread-spezifischen Message Queue* und leitet sie an die entsprechende Behandlungsroutine weiter. Der hier ausgenutzte Schwachpunkt dieses Systems ergibt sich daraus, dass keine Prüfung vorgenommen wird, aus welcher Quelle die entsprechende Nachricht stammt. Somit kann eine Anwendung vielfältige Aktionen auslösen und nahezu alle Benutzereingaben nachahmen.

Da aktuelle Personal Firewalls auf Nutzereingaben zum Bestätigen oder Ablehnen einer Kommunikationsanfrage warten, wurde diese Eingabe durch das System generiert und an das Fenster gesendet. Hierfür muss zunächst der korrekte Fenstername und Klassename des Dialogfensters ermittelt werden, um diesen in den Vordergrund zu holen und später die Eingaben entsprechend umleiten zu können. Wie sich z. B. über das in *MS Visual Studio* enthaltene Tool *Spy++* ermitteln lässt, hat dieses Dialog-Feld den Klassennamen *#32770*. Sowohl das *ZA*-Fenster als auch das *Sygate*-Fenster sind durch einen eindeutigen Namen identifizierbar. Für *Zone-Alarm* ist dies der String "*ZoneAlarm-Sicherheitswarnung*", für *Sygate* besteht der Fenstername aus dem String "*Sygate Personal Firewall*" und dem aktuellen Datum mit Uhrzeit[12]. Im Laboraufbau wartet ein Thread in einer Endlosschleife auf das jeweilige Dialogfenster, für welches sich anhand des Namens und der Klasse mit Hilfe der Funktion

```
FindWindow(
__in_opt LPCSTR lpClassName,
__in_opt LPCSTR lpWindowName);
```

ein Handle erstellen lässt. Damit die simulierten Tastatureingaben an die Dialog-Fenster gelenkt werden können, müssen diese sich im Vordergrund befinden. Ein Fenster kann ein anderes Fenster durch die Funktion *BOOL SetForegroundWindow(HWND hWnd)* in den Vordergrund bringen. Ein Fenster, welches sich aktuell im Vordergrund befindet, kann sich jedoch dagegen wehren, den Vordergrund an ein anderes Fenster abzutreten. Hierzu kann das Fenster die Funktion *BOOL LockSetForegroundWindow(UINT uLockCode)* verwenden. Dies wird z. B. automatisch bei Drücken der *ALT*-Taste oder dem Navigieren in einem Kontext-Menü aufgerufen. Um das Abfragefenster zuverlässig bestätigen zu können, muss es deshalb in den

[12] Formatbeispiel: Sygate Personal Firewall 2008-08-22 22:34:38

Vordergrund gezwungen werden können. Hierfür wird zunächst ein Handle zum aktuellen Vordergrund-Fenster per Aufruf von *HWND GetForegroundWindow(VOID)* ermittelt. Mit Hilfe von

```
BOOL WINAPI AttachThreadInput(
        __in DWORD idAttach,
        __in DWORD idAttachTo,
        __in BOOL fAttach //TRUE = attach
);
```

kann die Eingabe eines Threads (des aktuellen Vordergrund-Threads) an den Ziel-Thread geknüpft werden, wobei die Eingaben des Ziel-Threads weitergeleitet werden (*idAttachTo*). Nun wird an den aktuellen Vordergrund-Thread per *BOOL SetForegroundWindow(HWND hWnd)* die Aufforderung geschickt, das Dialogfenster der PFW (*hWnd*) in den Vordergrund zu holen. Da das aktuelle Fenster sich im Vordergrund befindet, ist es immer berechtigt, diesen Fokus abzugeben und schließlich kann das Abfragefenster in den Vordergrund gebracht werden. Im Anschluss muss diese Eingabeverknüpfung wieder rückgängig gemacht werden, um Nachrichten an das Dialogfenster zu senden. Mit Hilfe der Funktion

```
UINT SendInput(
        UINT nInputs,
        LPINPUT pInputs,
        int cbSize
);
```

lassen sich nun zunächst einige *TAB-Up-*/*TAB-Down*-Events an das Dialogfenster senden, um so auf die entsprechenden Buttons zu „springen“. Durch das Simulieren eines *Space-Up-*/*Space-Down*-Events wird der entsprechende Button im Dialogfenster anschließend bestätigt. Der hier grob skizzierte Weg kann durch einige Zwischenschritte noch weiter verfeinert werden und lässt sich für entsprechende PFWs effektiver anpassen.

Das automatisierte Bestätigen wurde im Laboraufbau innerhalb einer Zehntelsekunde[13] realisiert, was zur zuvor beschriebenen Wahrnehmung eines grafischen Flackerns führte. Eine Verbesserung ist möglich.

4. Fazit und Ausblick

Das im Rahmen der Untersuchung gewonnene Modell eines privaten PCs sowie der erstellte Demonstrator zum Ausnutzen der Schwachstellen zeigt, dass ein System bereits dann erfolgreich mit einer komfortablen Hintertür versehen werden kann, wenn der Benutzer

- über eine „übliche“ *MS-Windows*-basierte Systemkonfiguration verfügt,
- einen aktuellen Virenscanner und eine *Personal Firewall* nutzt,

[13] Die simulierte Tastatureingabe darf nicht zu früh erfolgen; das Fenster muss erst aus Betriebssystem-Sicht aufgebaut und „im Vordergrund“ sein, damit die Nachrichten empfangen werden. Der Demonstrator wartet daher 100ms, bevor die Bestätigung simuliert wird. Dieser vereinfachte Ansatz ließe sich weiter verfeinern.

- und einmalig (leichtfertigerweise) eine ausführbare Datei (z. B. Mailanhang, Medien-Autostart, Onlineupdate) öffnet.

Der Angriff gelingt auch einem Angreifer, der über durchschnittliche technische Fähigkeiten und überschaubare Hilfsmittel verfügt; besondere Ressourcen (außer einem eigenen PC) oder spezialisiertes Expertenwissen werden nicht benötigt.

Bemerkenswert ist das Ergebnis aus Sicht der Autoren in der Hinsicht, dass die viel zitierten Sicherheitshinweise an den privaten PC-Anwender („Wichtig: Virenscanner und PFW installieren“[14]) nur einen geringen Schutz bieten. Zwar ist grundsätzlich zunächst dem Anwender selbst der Vorwurf zu machen, Dateien zu öffnen, deren Herkunft er nicht zweifelsfrei verifizieren kann; die Schutzmechanismen der Sicherheitstools sind jedoch für dieses Fehlverhalten vorgesehen. Aus Sicht des einzelnen Nutzers stellt sich damit die Frage nach dem tatsächlichen Mehrwert, der mit der Installation von Virenscanner und PFW gegeben ist. Denn berücksichtigt man hier die beträchtlichen Systemressourcen, die diese Tools für sich reservieren, sowie die Kosten und den Zeitaufwand der Installation und Konfiguration, ist ein nur geringfügiger Sicherheitsgewinn nicht mehr in jeder privaten Einsatzumgebung gerechtfertigt. Darüber hinaus ist der private Anwender kaum in der Lage, vertrauenswürdige Sicherheitstools, die wir in dieser Untersuchung berücksichtigt haben, von solchen zu unterscheiden, die mit zweifelhaften Sicherheitshinweisen in erster Linie den Vertrieb des Tools selbst beflügeln sollen oder gar mithilfe der vorgeblichen Sicherheitsfunktionalität erst die Hintertür für einen Angreifer öffnen [DB08]. Im Jahre 2008 wurden mehrere Softwarehersteller von Microsoft gerichtlich belangt, nachdem bekannt wurde, dass vorgebliche Sicherheitswarnungen der Softwareprodukte Falschmeldungen waren, die arglose Anwender zum Kauf nutzloser Sicherheitstools animieren sollten. [MP08]

Es wäre im Weiteren zu untersuchen, ob die genannten Sicherheitshinweise an den privaten PC-Anwender nicht kontraproduktiv sind, denn bei einem Anwender, der über keine Sicherheitstools verfügt, ist ggf. von einer höheren Sensibilisierung auszugehen, die ihrerseits einen höheren Sicherheitsgewinn darstellt. Es ist zunächst nicht offenkundig, ob der psychologisch verursachte Sicherheitsgewinn aufgrund einer Sensibilisierung vergleichbar ist mit dem Sicherheitsgewinn mithilfe technischer Hilfsmittel. Dies ist eine Fragestellung, die wir in weiteren Arbeiten untersuchen wollen.

5. Quellen

[AS08] Anja Schütz: *Vista und Mac OS gewinnen Marktanteile*. silicon.de-Meldung vom 8. Juli 2008. URL: http://www.silicon.de/mittelstand/0,39038986,39193148,00 (Stand: 09.10.2008).

[DB08] Daniel Bachfeld: *Scharlatane und Hochstapler. Zweifelhafte Antiviren-Produkte*. Artikel in Heise-Security vom 25.10.2008

[14] So enthalten die zehn Hinweise des BSI für einen Basisschutz eines Internetnutzers an erster und zweiter Stelle die Aussagen: „**1. Installieren Sie ein Virenschutzprogramm und ein Anti-Spyware-Programm und halten Sie diese immer auf dem aktuellen Stand.**
2. Setzen Sie eine Personal Firewall ein und aktualisieren Sie diese regelmäßig. Sie schützt bei richtiger Konfiguration vor Angriffen aus dem Internet und verhindert zudem bei einer Infektion des PCs mit einem Computerschädling, dass ausspionierte Daten an einen Angreifer übersendet werden können.“
URL: http://www.bsi-fuer-buerger.de/schuetzen/07_01.htm (Stand: 09.10.2008)

[LH06] Marcel Lehner, Eckehard Hermann: *Auffinden von verschleierter Malware*. Datenschutz und Datensicherheit 30 (2006)

[MP08] map: Microsoft verklagt Anbieter von falscher Anti-Spyware. Heise-Meldung vom 30.09.2008

[VB07] Volker Birk: *Der Staat als Einbrecher: Heimliche Online-Durchsuchungen sind möglich*. Telepolis. März 2007. URL: http://www.heise.de/tp/r4/artikel/24/24766/1.html (Stand: 09.10.2008)