BadUSB, aktuelle USB Exploits und Schutzmechanismen

Ramon Mörl[1], Andreas Koke[1]

Angriffe über USB-Ports und weitere Systemzugänge auf den PCs und Notebooks gab und gibt es viele [SCHO15]. Die Autoren verfolgen das Thema seit circa 15 Jahren und zeigen in diesem Beitrag am Beispiel des sogenannten BadUSB Angriffes, der auf der BlackHat 2014 in Las Vegas von Karsten Nohl und Jakob Lell vorgestellt wurde [NOH15], sowohl den Werdegang als auch den aktuellen Status solcher Angriffe auf. Neben der Diskussion welche Maßnahmen gegen die konkreten Bedrohungen geeignet sind, wird auch ein Ausblick gegeben, was in diesem Umfeld an Angriffsvektoren und Lösungsmöglichkeiten noch zu erwarten ist.

1. Einleitung

Wie in vielen Gebieten der ITK wurde auch im Umfeld der USB Peripheriegeräte zuerst die Funktionalität implementiert und auf Funktionen der IT-Sicherheit weitgehend verzichtet. Viele der Sicherheitsprobleme lagen (und liegen noch) an der Einbettung in die Betriebssysteme. Der hier behandelte Exploit nutzt aber die zunehmende „Intelligenz“ auch einfacher USB-Geräte, wie etwa Memory Sticks. Diese Intelligenz ist durch den Einsatz von hochleistungsfähigen Standard-Controllern bedingt. Bei Speichern liegt der Zwang zu mehr Leistung in den Controllern daran, dass die Fehlerkorrektur-Algorithmen für Speicher immer aufwändiger werden müssen, um die geringe Ladungsrepräsentanz und Quanteneffekte auszugleichen. Auf der Black Hat Konferenz im August 2014 in Las Vegas haben Karsten Nohl und Jakob Lell [NOH15] eine zu diesem Zeitpunkt neuartige Qualität der IT-Sicherheitsdefizite dargestellt, die für den Insider auch nicht überraschend ist aber zu einigem Aufsehen geführt hat, weil allgemein dargestellt ist, dass der Schutz vor solchen Angriffen schwierig bis unmöglich ist.

Hier geht es also darum, die einzelnen Komponenten des Angriffs wirklich zu verstehen und die möglichen Schutzmechanismen zu diskutieren, welche die Schäden minimieren oder sogar verhindern können. Gleichzeitig wird auf andere ähnliche Angriffsmodelle eingegangen. Nachdem die Angriffsszenarien auf den Windows-Betriebssystemen am komplexesten sind und diese die größte Verbreitung haben wird dieses Betriebssystem hier exemplarisch diskutiert – in anderen verhält sich die Problematik identisch oder ähnlich.

2. Grundlagen

Um das Verständnis dieses Artikels und des Angriffes, der unter dem Namen BadUSB auf der Black Hat Konferenz 2014 in Las Vegas vorgestellt wurde (BadUSB-on accessories that turn evil [NOH15] zu vereinfachen, werden hier zuerst einige technische Grundlagen kurz zusammengefasst. Dabei geht es im Wesentlichen darum, wie sich USB Geräte im allgemeinen und speziell USB-Sticks und Tastaturen über Plug&Play Schnittstellen an das Betriebssystem anbinden und wie Tastatureingaben dann im weiteren in das System übergeben werden.

1 Geschäftsführung, itWatch GmbH

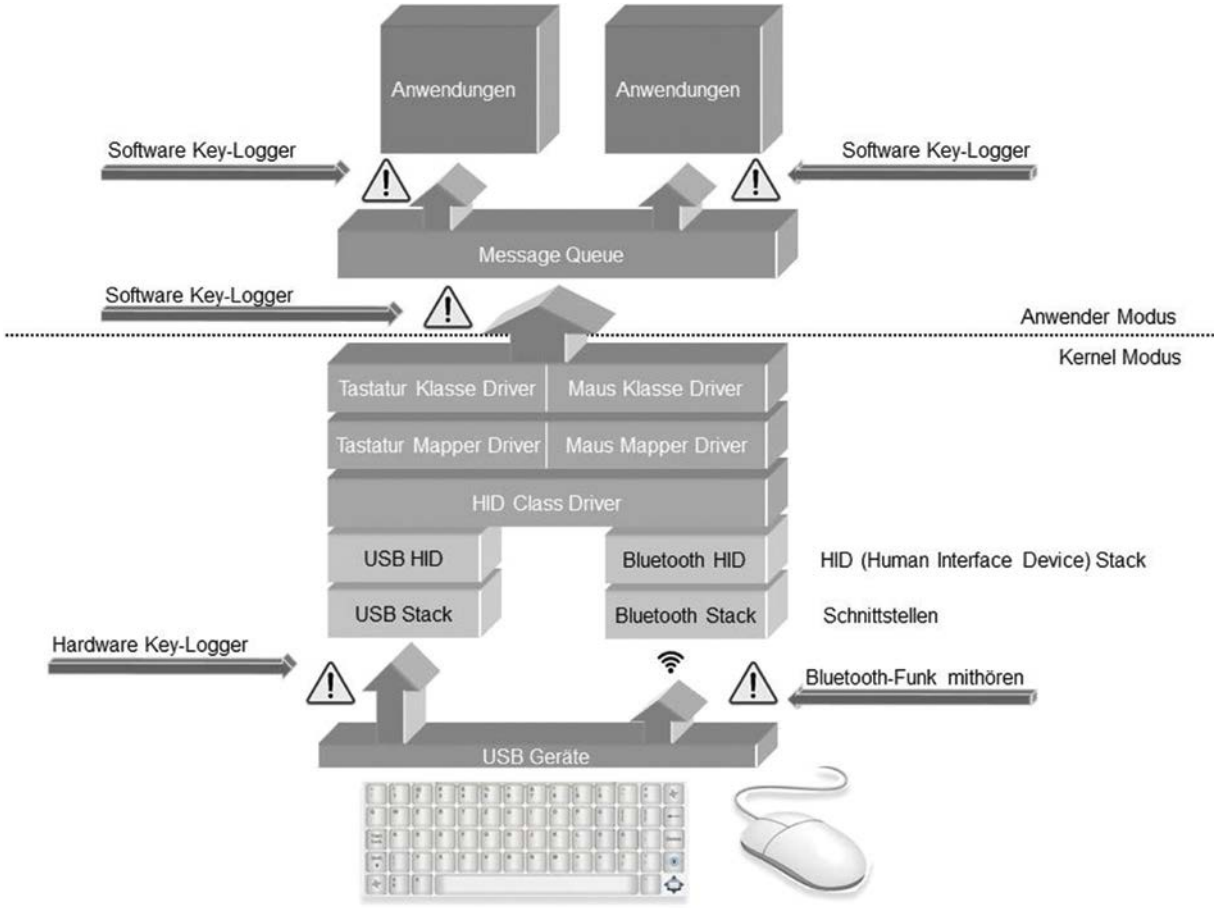

Abbildung 1: Verarbeitung von Eingaben unter Windows (vereinfachte Darstellung)

2.1 Tastatur und generische HID

Um die Eingabe von Anwendern möglichst einfach zu gestalten, ist eine Abstraktionsschicht mit dem Namen HID, kurz für Human Interface Device, in das Betriebssystem eingezogen. Die für die Unterstützung der HID Abstraktionsschicht benötigten Treiber sind bereits im Betriebssystem verankert. Standard-Geräte wie Maus und Tastatur können deshalb unabhängig von dem Kanal, über welchen sie kommen, z.B. USB, Bluetooth,…, direkt an die generischen HID Treiber angebunden werden und benötigen keine eigenen Treiber.

Wenn sich ein Stück Hardware als HID-Gerät zu erkennen gibt, wird dieses folglich vom Betriebssystem direkt bedient, ohne dass weitere Rechte oder Aktionen nötig sind. Alle Eingaben über dieses Gerät landen dann über die generischen HID-Treiber in den Message-Queues (eine Folge von Nachrichten) des Windows-Betriebssystems als „Keystroke-Messages“ oder „Mouse-Messages“. Spezialeingabegeräte benötigen evtl. zusätzliche Treiber, die aber auch durch eine interne Logik in den Geräten emuliert werden können.

2.2 Treiber-Stack

Die Anbindung von Hardware findet über Treiber oder eine Folge von Treibern (Treiber-Stack) statt. Die „unterste Schicht“ bilden dabei sehr hardware-nahe Treiber. Die „oberen Schichten“ werden immer generischer und sind von den eigentlichen Schnittstellen der Hardware immer weiter abstrahiert. Die oberste

Schicht stellt dann die Funktionen des Gerätes für die Nutzer des Betriebssystems – also die Anwender oder die Anwendungen – standardisiert zur Verfügung.

Im Fall einer Tastatur heißt das, dass in der „Keystroke-Message-Queue“ – also der Folge von Tastatureingaben – alle Eingaben, die über die verfügbaren Tastaturen in die Queue eingetragen wurden, für alle Anwendungen im Klartext zur Verfügung stehen. Es ist also nicht so, dass nur die Anwendung deren Fenster (Window) den Eingabefokus hat auf die Tastatureingaben zugreifen kann, sondern jeder Prozess der auf dem System läuft kann auf alle Tastatureingaben zugreifen. Dieser Umstand macht es leicht, sogenannte Software-Keylogger zu programmieren, also Software, welche alle Tastatureingaben mitschneidet / abhört.

Für die unterschiedlichen Geräteklassen gibt es verschiedene Treiber-Stacks. Datenträger benötigen Treiber-Stacks, welche ein Datei-System übergeben. Bestimmte Treiber-Stacks benötigen einen besonders hohen Datendurchsatz – das sind z.B. Treiber-Stacks für FireWire, über welche Video-Daten in das System kommen sollen, die dann in Echtzeit dargestellt werden. Solche Treiber-Stacks haben dann eine aus Sicht der IT-Sicherheit besonders kritische Zugriffsmöglichkeit, den sogenannten Direct-Memory Access, also direkten Speicherzugriff. Dieser direkte Speicherzugriff geht an der CPU – also den darüber liegenden Rechtesystemen – vorbei und ist ungeprüft. Die Treiber-Stacks sind vom Betriebssystem nicht gegeneinander geschützt.

Das neue Einbringen von Treibern oder Teilen von Treibern und das Modifizieren von Treibern erfordern höhere Privilegien. Ein einfacher Benutzeraccount kann normalerweise keine Treiber installieren. Ist ein Treiber auf einem Microsoft Betriebssystem erst einmal richtig installiert, dann wird er ab diesem Zeitpunkt von jedem Gerät für jeden Anwender automatisch genutzt. Device Control Systeme können das personengenau und gruppengenau unterbinden. In UNIX basierten Betriebssystemen können die Treiber meist explizit an Nutzer und Gruppen zugewiesen werden.

Je neuer das Windows-Betriebssystem ist, umso mehr Treiber sind bereits vorab geladen, um es den wenig versierten Anwendern zu erleichtern, mit neuer Hardware umzugehen. Als Schutz gegen Angriffe ist dieses Verfahren also ungeeignet, weil Angreifer zum einen nur auf Treiber Stacks setzen, die möglichst weit verbreitet sind und zum anderen eine einmalige evtl. sogar versehentliche Inbetriebnahme eines Gerätes den Treiber Stack verändert – insbesondere, wenn die Inbetriebnahme mit einem höher privilegierten Account vorgenommen wurde.

2.3 Identifikation von Geräten / Devices

Ein Gerät welches an einen Kanal / Port angeschlossen wird, muss, bevor es einem Treiber-Stack zugeordnet werden kann, zuerst einer Geräteklasse zugeordnet werden und dann identifiziert werden, um herauszufinden, ob es eventuell besondere Treiberelemente (Filter-Device-Treiber) benötigt. Das Betriebssystem erfährt dazu von der Hard- und Firm-Ware des Devices die Geräteklasse, den Enumerator, den Namen, eine eventuell vorhandene

Seriennummer und eventuell vorhandene weitere Eigenschaften. Das Betriebssystem geht davon aus, dass das Gerät „ehrlich“ meldet was es ist und wie es heißt.

Ein bösartiges Gerät kann also viele verschiedene Angriffsmöglichkeiten nutzen indem es z.B. neben seinen vom Anwender erwarteten Funktionen noch weitere Funktionen erbringt. Die Funktionen, die ein Gerät selbst erbringt werden im Wesentlichen durch die auf dem Gerät vorhandenen Firmware und Hardware-Komponenten, z.B. dem Controller erbracht.

2.4 Controller – die Intelligenz auf den Geräten / Devices

Intelligente Geräte haben ihre Intelligenz im Controller untergebracht. Der Controller kann im Normalfall über einfache Mechanismen aktualisiert werden. Die Aktualisierung erfordert häufig – wie in dem BadUSB Fall beschrieben - keine weitere Authentisierung, sondern nutzt definierte Befehle. Diese Befehle sind meist proprietär werden aber nicht geheim gehalten. Deshalb kann jeder, der diese Befehle kennt, den Controller beliebig modifizieren – insbesondere also zusätzliche Funktionen / Leistungen einbringen.

3. Der Exploit

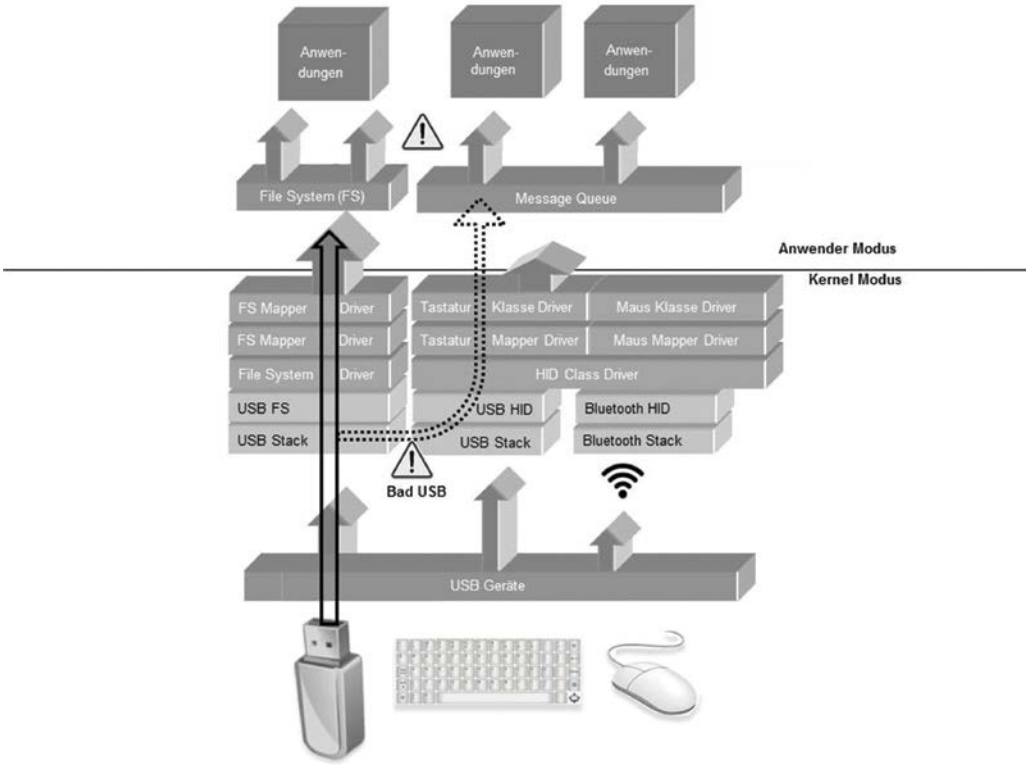

Abbildung 2: Angriffspunkte

Bestimmte Controller, im vorliegenden Fall wurde auf einen im Markt sehr weit verbreiteten Controller der Firma Phison© [PHISO15] aufgesetzt, akzeptieren eine Re-

Programmierung von außen ohne weitere Sicherheitsfeatures. Es findet also keine Authentisierung der Herkunft oder ähnliches statt[2].

Der Angriff BadUSB, so wie er auf der BlackHat 2014 [NOH15] beschrieben wurde, kann also wie folgt ausgeführt werden: Man nimmt ein marktübliches USB Device (USB-Stick) mit einem Controller von Phison© [PHISO15] und programmiert diesen so um, dass zu den Standardfunktionen des Controllers weitere Funktionen dazu kommen. Das wären zum Beispiel:

1. Der USB-Stick gibt sich zusätzlich zu seiner Funktion als USB-Stick auch als Tastatur aus (es könnte auch Netzwerkkarte oder ähnliches gewählt werden). Es ist absolut üblich, dass USB Geräte mehrere Funktionen übernehmen, wie z.B. ein Smartphone an einem USB Slot angesteckt folgende Funktionen wahrnimmt:
   a. Strom laden
   b. Modem
   c. Massenspeicher
   d. …
2. Für die unter 1. Genannten Funktionen, werden die vom Betriebssystem gelieferten generischen Treiber verwendet.
3. Die „neue“ schädliche Tastatur würde nun eigenständig aus einem Vorrat von Daten, die auf dem re-programmierten Controller hinterlegt sind, Tastatureingaben an den Rechner schicken. Fast alle Betriebssysteme kennen Kommandos deren Inhalt besagt „Führe die nachfolgenden Zeichen als Kommando aus“. Eine solche Zeichensequenz würde über die schädliche Tastatur (eigentlich den Controller des USB-Sticks) gesendet, ohne dass der Anwender oder das System bemerken, dass es sich nicht um die Eingaben eines natürlichen Menschen handelt.
   Verfeinerte Systeme können auch über die Tastatureingabe einen Prozess starten und die Tastatureingabe an diesen Prozess senden, so dass die Eingaben über die schädliche Tastatur für den Anwender nicht als „Störsignale“ sichtbar werden.
   In diesen Kommandos würde sich der Schadteil des Angriffs beliebig einbetten lassen: Datendiebstahl, Ransom Ware, Software Keylogger, Nutzung teurer kommerzieller Services und so weiter. Die schädliche Tastatur würde:
   a. zusätzlich Schadcode hinterlegen und sogar in der Lage sein
   b. das Booten des Systems zu erkennen und so vorgefertigte Bootviren zu hinterlegen
4. Der auf dem Rechner hinterlegte Schadcode greift nun seinerseits alle an diesen Rechner angesteckten USB Geräte an, welche einen gleichartigen Controller der Firma Phison [PHISO15] an Bord haben, und re-programmiert diese Controller, so dass sich der Schadteil weiter ausbreitet.

2 Handelsübliche Controller für Industrierechner oder industrielle Anlagen bieten einfache und effektive Mechanismen zur Verhinderung der Re-Programmierung und des Auslesens von Programmcode, der in dem Umfeld der USB-Sticks aber nicht angewendet wird.

4. Historie und ähnliche Exploits

Bereits vor einiger Zeit wurde ein Exploit einer schädlich veränderten Maus [HEIS11] einem größeren Publikum vorgestellt. In der Fachszene sind Internetseiten bekannt, auf welchen man Hardware-Keylogger in vorhandene Original-Tastaturen einbauen lassen kann, wenn der Besitzer mehr als 4 Werktage nicht an seinem Arbeitsplatz ist. Besonders perfide ist eine neuere Entwicklung, bei welcher der Hardware-Keylogger gleich mit einem Funkchip ausgestattet wird und so die erhaltenen Daten zu einem vorgegebenen Hotspot oder anderem Empfänger sendet. Die Stromversorgung des Funkchips erfolgt dabei ebenso wie die des Hardware-Keyloggers über den Wirt, also über den PC oder das Notebook bzw. die zugehörige Docking-Station. Dadurch vermeidet es der Angreifer, dass er nach dem erfolgreichen Anbringen der Hardware noch einen physischen Kontakt mit dem angegriffenen System benötigt, wobei er leicht identifiziert, überführt oder sogar festgenommen werden könnte.

Sichere, potentiell zertifizierte Devices, werden durch solche mit gleichen optischen Eigenschaften aber anderem Innenleben ersetzt, um die Sicherheitsservices, die sie anbieten zu manipulieren. Ein hardwareseitig verschlüsselnder USB-Stick, der auf starke Kryptografie aufsetzt, kann beispielsweise mit einem nach außen baugleichen Modell vertauscht werden, der nur die PIN Prüfung abfragt und dann eine Fehlermeldung abgibt, die PIN aber gleichzeitig über Luft- oder Netzschnittstelle verschickt, so dass der Original-Datenträger geöffnet werden kann.

Einige Exploits, die ebenfalls auf der Unzulänglichkeit von Controllern basieren, sind in der nationalen Verschluss-Sache (VS) -Community bekannt, sollen aber nicht in einem öffentlichen Forum diskutiert werden, um Nachahmung und eine frühzeitige Verbreitung zu verhindern.

Dass USB-Sticks immer wieder als Einfallstor genutzt wurden, wenn die Hürde über die Firewall und die geschützten Gateways zu gut abgesichert ist, ist bekannt. Angriffe, die zuerst über Autorun [JOHA08], dann U3 [JOHA08],WMF [GDI08] und später über technische Automatismen der Betriebssysteme (z.B. LNK [MIC10]) und immer wieder über modifizierte Dateiinhalte wie z.B. rtf [RTF14], zeigen auf, dass sich die Angriffe immer besser verstecken. Das Verstecken in einem Controller ist deshalb besonders perfide, weil, wie Karsten Nohl bereits anmerkt, der Controller die einzige Schnittstelle eines Devices ist, die man zu den Eigenschaften des Devices befragen kann. Ist der Controller also wirklich gut mit Schadcode ausgestattet, dann wird er auf die Fragen nach den Eigenschaften des Systems einfach die richtigen Lügen als Antwort einprogrammiert haben.

Der verfügbare Speicherplatz auf Flash Datenträgern ist aktuell für das Hinterlegen von Schadcode kein Problem, da bei modernen Flash Datenträgern 16 bis 64 GB zur Verfügung stehen. Besonders problematisch ist, dass bereits Exploits dokumentiert wurden, welche durch Modifikation des Controllers Daten auf einem Flash-Datenträger verstecken können [SCHA11]. Diese Exploits verbergen die versteckten Daten z.B. in sogenannten Bad Tracks, also als nicht mehr nutzbar markierten Speicherbereichen, und nutzen die Daten zu ihren eigenen Zwecken. Nachdem die Liste der Bad Tracks vom Controller erstellt und verwaltet wird, schützt sich der modifizierte Controller aus Sicht der Angreifer selbst – ist also sehr gut versteckt.

5. Potentielle Erweiterungen und Limitationen des Angriffes

In dem Beitrag von Karsten Nohl wird noch darauf hingewiesen, dass es bei manchen Angriffen noch eine „OK“ Bestätigung des Anwenders benötigt und der Exploit „nur“ die Rechte des Anwenders nutzen kann. Der Angriff kann erweitert werden, um auch diese Probleme zu lösen. Das automatische Bestätigen eines dann nur kurz „aufblitzenden“ „OK“-Knopfes kann durch geeignete gestartete Hintergrund-programme und geeignete Mausbewegungen, die über die HID Treiber eingesteuert werden, ebenso durch den dann komplexeren Schadcode ersetzt werden. Zusätzlich zu den Rechten des Anwenders erhält der Schadcode die Rechte aller Anwendungen und über oben erwähnte DMA Angriffe evtl. weiteren Zugang zu Systemprivilegien.

6. Problemquellen und ausgenutzte Schwachstellen

USB Geräte werden nicht authentisiert, sondern im besten Fall identifiziert – obwohl die Möglichkeiten einer starken Authentisierung im Prinzip gegeben wären. Das heißt ein Angreifer kann unter Vorspiegelung eines vordefinierten Namens und evtl. einer HardwareID, die sich beliebig in die Soft- oder Firmware integrieren lässt, beliebige Geräteidentitäten annehmen.

Einmal herausgegebene Devices sollen möglichst einfach und für jedermann aktualisiert werden können. Diese Art der Updates und Patches findet sich auf SmartPhones, TVs, Videorecordern, MP3-Playern aber eben auch auf beliebigen USB Geräten wieder. Anders als in den heute marktgängigen PC Betriebssystemen benötigt man aber für das Patchen, Updaten oder Reprogrammieren von vielen der gängigen USB-Geräten keine Authentisierung oder Integritätsprüfung des Updates.

Hauptgründe für diesen Sachverhalt sind Kosten und „Commodity“. Wenn der Hersteller weitere Hürden für einen Update einbauen würde, dann würde der Update zum einen technisch kompliziert werden und zum anderen müssten neue Sicherheitsfunktionen implementiert werden. Beides sind Kostentreiber. Außerdem kostet natürlich jede neue Funktion auch Geld und „time to market“, der Marktzugang verspätet sich also. In den Life-Style und Commodity Produkten ist die Geschwindigkeit mit der man die Produkte auf den Markt bringt entscheidend.

Hinzu kommt die steigende Verfügbarkeit von „middleware Produkten“ auf USB Basis, d.h. es ist heute jedem Schüler (auch finanziell!) möglich, derartige gefälschte Geräte herzustellen.

7. Mögliche Lösungen und Verteidigunsstrategien

In einem Interview [SCHLE14] der Computerwoche von Frank-Michael Schlede und Thomas Bär sagte Udo Schneider, Security Evangelist, zu der Gefährdung, die von einer BadUSB-Attacke ausgehen kann: "Aus den vorliegenden Informationen ergibt sich leider auch, dass es keine Möglichkeit gibt, die Daten, die der Kontroller dem Hostsystem übermittelt, zu verifizieren beziehungsweise die Integrität oder Validität dieser zu verifizieren. Kurzum: Es kann alles gefälscht werden - damit ist USB-Devices de facto nicht mehr zu trauen!"

Diese Aussage geht natürlich etwas zu weit, es gilt also zu verifizieren, welchen Teilen eines USB-Devices man wirklich trauen muss, um die durch das USB-Device zur Verfügung gestellten Daten und Services nutzen zu können. Zweifelsohne ist es eine

gute Abhilfe, wenn – wie das heute bereits der Fall ist – Sicherheitsdevices samt ihrer Controller gegenseitig stark authentisiert werden. Das geht aktuell aber nur mit einigen extra dafür ausgelegten Devices unter Nutzung von guten Device Control Produkten. Für den Rest der Welt gilt heute, dass es wie immer schwierig ist Produkte, die sich langjährig im Markt befinden, im Nachgang über einen neuen Standard sicherer zu gestalten. D.h. die Welt wird noch lange mit einigen der obigen Unzulänglichkeiten der USB Schnittstelle und anderer Kanäle / Ports leben müssen und es bedarf aktuell proprietärer Lösungen, um die Exploits sicher zu verhindern, was aber möglich ist.

Kryptographie kann hier nicht allein schützen sondern nur einen Teil der Lösung als Grundlage bereit stellen. In bestehenden Systemen ist das Thema „Kontrolle der Schnittstellen / Ports und der daran operierenden Geräte / Devices, sowie der Funktionen der Devices“ ein zentrales aber nicht das einzige Bauelement zum Schutz vor solchen Angriffen. Bei den Angriffen gilt es jetzt alle Seiten zu analysieren.

- Zum einen kann es passieren, dass ein „guter“ USB-Stick von einem schädlichen PC infiltriert wird
- Zum anderen kann ein integrer PC von einem schädlichen USB-Stick angegriffen werden.

Wesentlich erscheint es den Autoren[3][MOE14] zu untersuchen, wie vollständige Vertrauensketten aufgebaut werden können, so dass die Angriffe nach den genannten Mustern ohne Schaden an geschützten Systemen vorbei gehen, also sicher abgewehrt werden

8. Schutz der USB-Geräte vor Re-Programmierung

Wahrscheinlich kann man noch nicht einmal erkennen ob ein konkreter Stick befallen ist (solange er nichts Bösartiges tut), denn wie Karsten Nohl auf der Black Hat darstellte: wenn Sie den schädlichen Controller fragen ob er denn Schadcode beinhaltet, wird er einfach lügen. Genauso kann der Controller natürlich viele seiner Eigenschaften verbergen oder anders darstellen, so dass ein Erkennen eines modifizierten Systems von außen unwahrscheinlich ist.

Am wichtigsten wäre es eine Infrastruktur zur Authentisierung vor einer Re-Programmierung aufzubauen, die in der Hardware der Devices verankert ist. Standardisierung, Aufbau und Verbreitung der Infrastruktur benötigen aber viel Zeit, so dass das keine zeitnahe Hilfe bietet.

Prinzipiell kann man einen Reprogramming-stream als eine Serie von standardisierten oder proprietären scsi commands genauso analysieren und scannen wie webstreams, IP-Kommunikation und Dateien. Das ist die eine Richtung in der man verfolgt, dass aus den eigenen Systemen kein Schaden für angesteckte USB-Devices entstehen kann. Interessant ist aber auch, an was man erkennen kann, ob ein eingestecktes USB-Device Funktionen ausführen möchte, die nicht zu dem erwarteten Leistungsspektrum des USB Devices gehören – also potentiell Schadcode enthalten. Dazu mehr in dem nächsten Kapitel.

---

3 „Dieser Beitrag verfolgt die These, dass die IT-Sicherheit deshalb als so kompliziert wahrgenommen wird, weil sie gerade nicht als nachweisbare Vertrauenskette unter Berücksichtigung aller Facetten betrachtet wird.“
<kes> Ausgabe 5/2014 Ramon Mörl: Die großen Lügen der IT-Sicherheit

Klar ist, dass ein potentieller Schadcode in dem Controller durch ein sicheres Formatieren nicht verschwinden wird. D.h. es wäre eine, zweifelsohne aufwändige, Möglichkeit bei jedem Kontakt mit einem sicheren System den Controller jeweils auf einen integren Stand zu re-programmieren, bevor die Kommunikation mit dem Device eröffnet wird. Die Voraussetzung ist, dass alle Firmwarekomponenten bekannt sind, was in einer offenen Umgebung unwahrscheinlich ist. Für geschlossene Systeme des mittleren oder hohen Schutzbedarfs ist aber eine Einschränkung auf Hardware, die eine bestimmte Voraussetzung erfüllt, durchsetzbar.

9. Schutz von integren PCs / Computern vor schädlichen USB-Geräten

Die in 2.3 beschriebene Identifikation, statt einer sicheren Authentisierung, der Devices dient bisher i.d.R. ausschließlich dem Zweck der richtigen Treiber/Softwarezuordnung des Betriebssystems, unterstützt also wie so oft nur das Funktionieren des Plug&Play übernimmt aber keine Sicherheitsprüfung. Zwingende Voraussetzung für eine gute Device Control Lösung ist es deshalb zuallererst, dass alle Treiber-Stacks, die das Device nutzen möchte, einer echten Zugriffskontrolle unterliegen, also ein Gerät nur genau zu dem Treiber-Stack Zugriff bekommt, zu dem es auch berechtigt ist. Unter UNIX® Betriebssystemen ist das deutlich einfacher als unter Microsoft® Betriebssystemen oder IOS®.

Aus der Welt der Device Control Lösungen gibt es weitere Standardverfahren, wie z.B. das automatische Verhindern oder Erkennen einer zweiten Tastatur und die konkrete Freigabe dieser durch vertrauenswürdiges Bedienpersonal – das kann entweder der authentisierte Nutzer sein, der ja in dem Systemzustand weiß, ob er eine zweite Tastatur angesteckt hat oder auch ein Prozess, der berechtigte Administratoren mit einbezieht. Wichtig dabei ist, dass dieser Prozess auch außerhalb einer Benutzeranmeldung z.B. während der Bootphase korrekt arbeitet. Diese Aspekte sind in [SCHO05] dargestellt.

Diese bisher genannten Schutzverfahren decken einige der Angriffsvektoren ab. Nicht abgedeckt ist ein Angriff, bei welchem der Angreifer physikalisch Zugriff auf den PC hat und die „Originaltastatur“ absteckt bevor er die schädliche ansteckt und die Schadsoftware vorher auch aus dem System ausliest, wie die HardwareID der normalerweise verwendeten Tastatur lautet. Dieses Verfahren kann auch der Schadcode direkt anwenden. Der Anwender würde nur für einen kurzen Moment Plug&Play Aktivitäten wahrnehmen, die durch das Anstecken des USB-Sticks ohnehin ausgelöst werden, und die Tastatureingaben des Anwenders würden für einen kurzen Moment verloren gehen. Für die Angriffsszenarien außerhalb einer Benutzeranmeldung, also z.B. während des Bootvorganges, benötigt man z.B. die Möglichkeit Tatstaturen während dieser Phase generell zu verhindern und erst für das Login wieder automatisiert zu aktivieren.

Eine Investition, wie sie die meisten Unternehmen scheuen, würde auch sicher helfen. Es gibt Tatstaturen die sich über kryptografische Verfahren stark authentisieren lassen. Mit professionellen Device-Control Werkzeugen lässt sich dann sicherstellen, dass nur solche Tatstaturen in einem Rechnersystem eingesetzt werden können. Der beschriebene Angriff läuft hier ins Leere, da die schädliche Tastatur nie positiv authentisiert wird.

Im Zuge der Erkennung von Hardware-KeyLoggern wurden Verfahren getestet, die den zusätzlichen Strombedarf der modifizierten Devices erkennen. Unter Laborbedingungen lässt sich so eine gute Erkennung darstellen, die allerdings nicht praxistauglich ist, da die Strommesswerte, die in gängigen PC-Systemen ausgelesen werden können, zu ungenau sind.

Vor der Auswirkung von Hardware-Keyloggern schützt man sich am besten mittels Tastaturen, welche die Tasten-Anschläge verschlüsselt an das System kommunizieren. Diese bringen einen Filter-Treiber in den Treiber-Stack für Tastaturen ein und entschlüsseln erst auf oberer Ebene der Treiber wieder. Hardware-Keylogger können dann nur verschlüsselte Daten abhören und keine echten Nutzdaten ausforschen.

Wie unter 2.2 beschrieben werden aber auch die verschlüsselten Tastendrücke an der generischen Schicht notwendig entschlüsselt an die Anwendungen übergeben. Dort greifen Software-Keylogger an. Einen sicheren Schutz vor Software-Keyloggern kann man (ohne gute Applikations- und Prozess-Kontrolle) nur erreichen, wenn man Tastatureingaben direkt in die sensiblen Eingabefelder bzw. noch besser in die sensible Anwendung lenkt, z.B. für Passwort oder PIN Dialoge. Das ist im Betriebssystem nicht vorgesehen, man benötigt dazu also dedizierte Produkte, welche die direkte Übergabe von einer Tastatur in ein Dialogfeld oder eine Anwendung ermöglichen. Solche Lösungen gibt es am Markt.

Bildschirmtastaturen sind nicht per se für diese Aufgabe geeignet, da sie prinzipiell auch über die standardisierte, entschlüsselte Schnittstelle gehen, welche alle Tastendrücke an alle Anwendungen weiterleiten. Bei Bildschirmtastaturen kommt noch dazu, dass ein Abhören oder ein Einsteuern der Mausbewegungen und Klicks genauso einfach ist wie das Abhören der Tastatur. Insofern sind sichere Maustastaturen auch vor dem Abhören der Mausbewegungen und dem willkürlichen Einsteuern von Eingaben geeignet zu schützen. Auch dazu gibt es bereits Lösungen.

Ein guter weiterer Weg ist die automatische Content-Filterung von Tasteneingaben und das Verhindern von kritischen Tastenkombinationen wie „Execute as Command“ (Content Filter für Key Strokes).

Nahe an diesem Verfahren sind traditionelle Mechanismen, die früher für die Authentisierung von Anwendern entwickelt wurden und auf deren Tastenanschlagsrhythmik abzielen, die für jeden Menschen eigen ist. Durch solche Verfahren lässt sich heuristisch relativ sicher erkennen, ob hinter der „Eingabe“ ein menschliches Wesen sitzt, oder die Tastatureingaben maschinell vorgegeben sind.

Interessant ist auch, dass wesentliche Teile des Exploits durch eine professionelle Applikationskontrolle verhindert werden. Eine professionelle Applikationskontrolle prüft jeden gestarteten Prozess (und nicht nur jede gestartete Anwendung) und verfolgt zurück, wo der Ausgangspunkt für den Prozessstart ist. Wie von Karsten Nohl nahegelegt wird das Verhindern von „Autoplay“ nicht ausreichen. Werden aber keine Prozesse akzeptiert, deren Ursprung auf einem Peripheriegerät liegt oder deren Ursprung nicht systemtechnisch sicher zu einer positiv freigegebenen Quelle nachvollzogen werden kann, so können die Auswirkungen des Schadcodes verhindert werden.

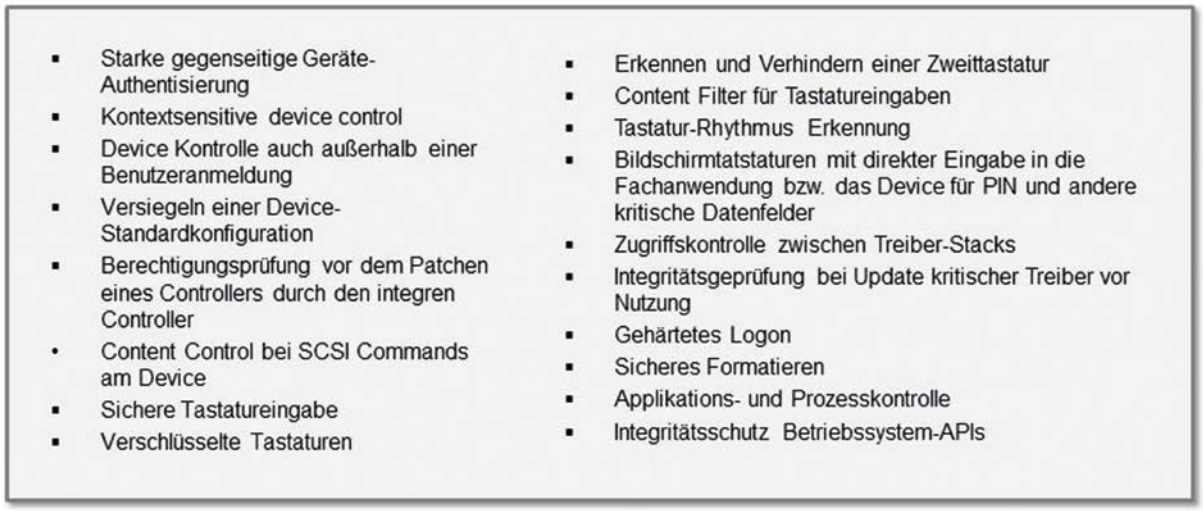

Abbildung 3: Elemente der technischen Verteidigung

10. Fazit und Ausblick

Auf einer Konferenz zwischen verschiedenen Ministerien aus den Ressorts Innen, Wirtschaft und Finanzen und ausgewählten Industrieteilnehmern zu dem Thema IT-Sicherheit wurde der IT-Sicherheit vorgehalten, generell zu kompliziert zu sein. Das Beispiel aus der Politik: „ich kann jederzeit an jeder Ecke eine Bratwurst kaufen und bekomme immer das was ich will – IT-Sicherheit ist immer schwierig“. Die Diskussion daraufhin zeigte auf, dass der Kauf der Bratwurst in ein ganzes Szenario von gesetzlichen Rahmenbedingungen, deren Überprüfung und verschiedenen Nachweisketten eingebettet ist, die so weit gehen, dass der lückenlose Nachweis des Transportes der lebenden Tiere, Herkunft und Verarbeitung rück-verfolgbar sein müssen, damit der Schutz der Bevölkerung vor gesundheitsschädlichen Waren gegeben ist und im Fall von Verstößen (z.B. Gammelfleischskandal) die Schuldigen auch identifiziert und bestraft werden können.

In diesem Beitrag wollten wir nachweisen, dass der Technologie „USB“, wie auch vielen anderen Technologien, eine Unsicherheit an vielen Stellen inhärent ist. Statt nun den Einsatz von USB vollständig als unsicher zu brandmarken und damit realitätsfremd zu reagieren, geht es darum, die Angriffe und deren Wirkung aufzuzeigen, um dann geeignete technische, organisatorische und bezüglich der Haftung rechtssichere Lösungen zum Schutz zu erarbeiten. Nach unserer Erfahrung ist es aber noch nicht ausreichend den Blick auf einen konkreten Exploit oder eine zu erwartende neue Kategorie von Angriffen zu lenken, von dem noch viele vergleichbare kommen werden, vielmehr wollen wir anregen, noch weiter zu gehen und den Blick darauf lenken, vollständige stabile und nachweisbare Vertrauensketten zu etablieren, denn erst durch die vollständigen Vertrauensketten, die bei einer „sicheren Tastatur“ beginnen und über zertifizierte, vertrauenswürdige Services zu adäquat geschützten Daten reichen, wird eine nachhaltige Sicherheit möglich.

Insofern ist das Arbeiten an den sicheren Standards ein wichtiger Weg, gleichzeitig muss aber auf Architekturen gesetzt werden, welche schon heute möglich sind und aus Sicht der IT-Sicherheit fehlertolerant gegen bestimmte unsichere Bestandteile sind. Bei der Erstellung der Vertrauensketten ist die Einsicht, dass Kryptographie nur eins von vielen wesentlichen Mitteln ist, unabdingbar. Systeme deren gesamtes Vertrauen

nur auf Kryptographischen Verfahren, die im besten Fall in sichere Hardware eingebettet sind, beruhen können sich inhärent unsicheren Standardtechnologien wie USB-Geräten nicht öffnen. Erst wenn zu den kryptographischen Verfahren andere, wie oben beschriebene Kontrollmechanismen in geeigneter Qualität dazu genommen werden, entstehen diese Vertrauensketten, die dann auch USB-Standardtechnologien problemlos mit einbeziehen können.

Literatur

[GDI08] Sicherheitsanfälligkeiten in GDI können Remotecodeausführung ermöglichen (956802).9.12.2008. URL: https://technet.microsoft.com/de-de/library/security/ms08-071.aspx (abgerufen am 30.01.2015)]

[HEIS11] Heise Security Online 29.06.2011: Angriff der Computer-Maus URL: http://www.heise.de/security/meldung/Angriff-der- Computer-Maus-1269684.html (abgerufen am 28.01.2015)

[JOHA08] Johansson, Jesper M.: Sicherheit auf dem Prüfstand.TechNet Magazin. © 2008 Microsoft Corporation und CMP Media, LLC. URL: https://technet.microsoft.com/de-de/magazine/2008.01.securitywatch.aspx (abgerufen am 30.01.2015)

[MIC10] Microsoft bestätigt USB-Trojaner-Lücke. 17.07.2010. URL: http://www.heise.de/security/meldung/Microsoft-bestaetigt-USB-Trojaner-Luecke-1039915.html (abgerufen am 15.01.2015)

[MOE14] <kes> Ausgabe 5/2014 Ramon Mörl: Die großen Lügen der IT-Sicherheit

[NOH15] Nohl, Karsten/ Lell, Jakob:Black Hat USA 2014.URL: www.blackhat.com/us-14/briefings.html#badusb-on-accessories-that-turn-evil (abgerufen am 28.01.2015)

[PHISO15] Phison Electronics Corporation. URL: http://www.phison.com/English/Main.asp (abgerufen am 28.01.2015)

[RTF14] RTF-Bug in Microsoft Word: Schutzmaßnahme auch von deutschem Anbieter. Link: 03/28/14-09:38.28.03.2014. URL: http://technet239.rssing.com/chan-4753999/all_p452.html (abgerufen am 28.01.2015)

[SCHA11] Schartner, Peter/ Taeger, Jürgen:D-A-C-H Security 2011. Oldenburg 2011, S.2-9.

[SCHLE14] Schlede, Frank-Michaela/ Bär Thomas. Computer Woche Online. So nutzen Sie USB „sicher“ weiter 25.12.2014: URL: http://www.computerwoche.de/a/so-nutzen-sie-usb-weiter-sicher,3067048 (abgerufen am 28.01.2015)

[SCHO05] Scholz, Peter: Unbekannte Schwachstellen in Hardware und Betriebssystemen, in: Handbuch der Telekommunikation, Wolters Kluwer Verlag, März 2005, ISBN 3-87156-096-0