17. November 2014 Studie · Digitale Identität · Transparenz · 17 Min. Lesezeit

Bedrohung durch manipulierte USB-Geräte

Archiviert. Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

Quelle https://technology2.a-sit.at/veroeffentlichungen/bedrohung-durch-manipulierte-usb-geraete/

In der jüngeren Vergangenheit traten vermehrt präparierte USB-Geräte in Erscheinung, die von gegenwärtigen Schutzmaßnahmen nicht als Bedrohung erkannt werden können. Im Rahmen einer Kurzstudie wurde erörtert, anhand welcher Methoden USB-Geräte manipuliert werden können, um damit unentdeckt Computer anzugreifen. Nach einer kurzen Einleitung zum USB-Standard wurden bekannte Angriffsmöglichkeiten zusammengefasst. Ein besonderes Augenmerk wurde auf die Manipulierbarkeit der Firmware bei USB-Sticks gelegt („BadUSB"-Angriff). Praktische Fallbeispiele dienten als Unterstützung beim Aufzeigen der Problematik und Veranschaulichung der mögliche Tragweite eines Angriffs. Abschließend wurde der Einsatz möglicher Schutzmechanismen diskutiert.

Die wesentlichen Erkenntnisse der Studie können wie folgt zusammengefasst werden:

- Physikalisch präparierte Geräte eignen sich nur für gezielte Angriffe und implementieren zumeist eine spezifische Funktionalität (z.B. „Hardware Keylogger").
- Angriffe über USB-Geräte funktionieren unabhängig von einer Schwachstelle in einem Betriebssystem. Sie nutzen vielmehr Mängel des USB-Standards oder Schwächen in fehlerhaft programmierten Treibern erfolgreich aus.
- Manipulierte USB-Geräte imitieren die Funktionalität anderer Geräteklassen, können je nach Ausstattung ihrer Hardware jedoch nur beschränkt als solche operieren. Zum Beispiel taugt ein zweckentfremdeter USB-Stick nur bedingt als Chipkarten-Lesegerät.
- Der „BadUSB"-Angriff ist nur bei Geräten anwendbar, für die der Hersteller des verwendeten Mikrocontrollers die Möglichkeit vorsieht, die Firmware per USB auszutauschen.
- Die Firmware von USB-Geräten ist grundsätzlich mit keiner digitalen Signatur versehen, über die sich Aussagen über ihre Integrität und Urheberschaft treffen ließen.

- Der „BadUSB“-Angriff wurde bisher nur für USB-Sticks mit Mikrocontroller der Firma Phison vorgezeigt. Die gewählte Methodik lässt sich systematisch auch auf andere USB-Geräte anwenden, führt allerdings nicht zwangsläufig zu einem erfolgreichen Angriff.
- Die Umsetzbarkeit technischer Schutzmechanismen ist beschränkt; Angriffsmöglichkeiten sollten daher auch anhand von organisatorischen Maßnahmen berücksichtigt werden.

1. Einleitung

Die in fast allen Computern vorhandenen USB-Schnittstellen ermöglichen die Kommunikation mit einer Vielfalt von Geräten. Seit jeher ist auch bekannt, dass Antivirenprogramme prinzipiell verhindern können, dass über USB-Sticks transportierte Schadsoftware ausgeführt wird. Um die Schutzmaßnahmen von Virenscannern zu umgehen, traten daher in der jüngeren Vergangenheit vermehrt modifizierte USB-Geräte auf, deren Verhalten sich gegenwärtig nur sehr schwer analysieren lässt.

Eine wesentliche Eigenschaft von USB ist die Flexibilität, verschiedene Gerätetypen über die gleiche Schnittstelle mit einem Computer verbinden zu können. Da ein Computer nicht im Vorhinein wissen kann, welche Funktionalitäten ein angeschlossenes Gerät bereitstellt, zieht er dafür die Informationen über das Gerät heran, die es über sich mitteilt. Ein Benutzer wiederum verwendet USB-Geräte im Vertrauen darauf, dass sie sich so verhalten, wie man es von einem korrekt implementierten Gerät erwarten würde.

Seit längerer Zeit ist bekannt [1], dass physikalisch präparierte USB-Geräte fähig sind, einen Computer weitgehend unbemerkt zu beeinflussen. Anhand von manipulierten USB-Sticks wurde vor kurzem auf eine neue Form der Bedrohung hingewiesen, gegen die derzeitige Schutzmechanismen weitestgehend wirkungslos sind. Ging man bislang davon aus, dass USB-Sticks eine Gefahr darstellen, weil sie für den Transport von Schadsoftware dienen können, zeigt der „BadUSB“genannte Angriff auf, wie sich USB-Sticks vollständig zweckentfremden lassen.

In dieser Studie wird erörtert, wie manipulierte USB-Geräte möglichst unentdeckt einen Computer angreifen können. Nach einer Zusammenfassung möglicher Angriffsvektoren wird ein besonderes Augenmerk auf die Manipulation von Firmware gelegt.

Im Folgenden wird eine kurze Übersicht über den weiteren Inhalt des Dokuments gegeben. In Abschnitt 2 werden zunächst grundlegende Begriffe und Abläufe im Zusammenhang mit USB erläutert. Im Anschluss werden in Abschnitt 3 gegenwärtige Angriffsmöglichkeiten aufgezeigt. In Abschnitt 4 wird auf die Manipulierbarkeit von Firmware bei USB-Sticks eingegangen und daraus resultierende Angriffsmöglichkeiten durch kurze Fallbeispiele unterlegt. Nach einer kurzen Diskussion möglicher Schutzmechanismen schließt dieses Dokument mit einem Fazit in Abschnitt 5 ab.

2. Grundlagen

Seit der Einführung des Universal Serial Bus im Jahr 1996 wird das serielle Bussystem dazu verwendet, unterschiedliche Gerätetypen (Drucker, Maus, Tastatur, Lautsprecher, Festplatten, etc.) über eine gemeinsame Schnittstelle mit einem Computer zu verbinden.

Die Übertragung von Datenpaketen zur Kommunikation mit USB-Geräten wird zentral über einen (oder mehrere) USB Host-Controller gesteuert. Weil jedem angeschlossenen USB-Gerät eine eindeutige 7-Bit lange Adresse zugewiesen wird, kann ein Host Controller maximal 127 Geräte verwalten [1] . Dieses Limit erhöht sich auch durch die Verwendung von USB-Hubs nicht, welche lediglich das Signal auf weiteren Ports zur Verfügung stellen. Wird ein USB-Gerät an einen Computer angeschlossen, identifiziert es der Host-Controller zuerst durch Abfrage einer fix hinterlegten Hersteller- und Produktbeschreibung („Device Descriptor“). Darüber hinaus teilt ein Gerät mit, welche Funktionalitäten bzw. Geräteklassen es zur Verfügung stellt und unter welchen Endpunkten jene angesprochen werden können.

Eine grundlegende Eigenschaft von USB ist, dass bekannte Geräteklassen [2] oft auch mit generischen Treibern verwaltet werden können. Für USB-Geräte wie Tastaturen, Massenspeicher oder Webcams wird somit nicht zwangsläufig ein eigener Treiber benötigt. Identifiziert sich also ein Gerät beispielsweise mit der Klasse 8 als Massenspeicher, so kann ein Computer unter Zuhilfenahme des für diesen Gerätetyp bekannten Treibers damit interagieren. Wie in Abbildung 1 1 http://www.usb.org/developers/docs/usb_31_102214.zip 2 http://www.usb.org/developers/defined_class Geräteklassen Endpunkte USB-Hub

Col1	Col2
USB-Geräte USB-Stick 8 – Massenspeicher 0 – Steuerung 1 – Datentransfer Webcam 1 – Audio 14 – Video 0 – Steuerung 1 – Videotransfer 6 – Audiotransfer 7 – Video-Interrupts	USB-Geräte

Abbildung 2. USB-Kommunikation

Computer	Col2	Col3
USB Host-Controller	USB Host-Controller	USB Host-Controller
USB Host-Controller		
USB Host-Controller		

exemplarisch illustriert, können Geräte ebenso mehrere Funktionalitäten gleichzeitig bereitstellen. Eine Webcam mit eingebautem Mikrofon würde beispielsweise die Geräteklassen 1 und 14 für Audio und Video implementieren. Um einem System zu signalisieren, dass für ein USB-Gerät ein proprietärer Treiber vonnöten ist, ist die Verwendung der Klasse 255 vorgesehen.

Über nummerierte Endpunkte stellen Geräte einzelne Funktionen bereit, auf die parallel zugegriffen werden kann. Im Fall einer Webcam findet so z.B. die zeitgleiche Übertragung von Audio- und Videoinformationen statt. Eine USB-Tastatur wiederum nutzt üblicherweise einen Endpunkt zur Übertragung von Tasteneingaben und einen weiteren für die LED-Statusanzeige [3] .

Schließt man ein USB-Gerät an einen Computer an, wird ein definierter Prozess durchlaufen um es beim System gemäß seiner Funktionalität zu registrieren. Das jeweilige Gerät führt zuerst seinen Bootloader aus, der daraufhin die hinterlegte Firmware startet. Nachdem Informationen über unterstützte Geräteklassen und Endpunkte an das System übermittelt wurden, lädt jenes schließlich den entsprechenden Treiber.

Der beschriebene Registrierungsprozess kann von USB-Geräten jederzeit neu angestoßen werden. So ist es beispielsweise möglich, dass sich ein USB-Modem bei der ersten Erkennung als CD-ROM-Laufwerk ausgibt, von welchem ein Anwender einen Treiber installieren soll. Sobald die Installation abgeschlossen ist, kann sich das Gerät neu registrieren und fortan als Modem zu erkennen geben.

2.1. USB-Massenspeicher

Mit einer großen Speicherkapazität auf einer verhältnismäßig kleinen Fläche zählen USB-Sticks zu den gängigsten Datenspeichern, die sich mit der Geräteklasse „Massenspeicher“ ausweisen. Darüber hinaus wird sie auch bei Digitalkameras, Mobiltelefonen und externen Festplatten eingesetzt.

Wie bei jedem USB-Gerät verwenden USB-Speicher einen Mikrocontroller, der zuerst die Firmware des Geräts lädt und schließlich als Baustein zwischen der USB-Schnittstelle und dem verbauten Flash-Speicher fungiert. Sobald sich ein Gerät als Massenspeicher registriert hat, wird es vom Computer über SCSI-Befehle angesprochen, welche in USB-Datenpaketen gekapselt und auf dem USB-Gerät vom eingesetzten Mikrocontroller verarbeitet werden [456] . Die Spezifikation erlaubt es Herstellern, zusätzlich eigene SCSI-Befehle zu definieren um mit dem Mikrocontroller über den Datentransfer hinaus interagieren zu können. Auf diese Weise können sowohl Produktinformationen über das Speichergerät ausgelesen, als auch beispielsweise Firmware-Updates eingespielt werden.

3 http://www.usb.org/developers/hidpage/HID1_11.pdf 4 http://www.usb.org/developers/docs/devclass_docs/Mass_Storage_Specification_Overview_v1.4_2-19-2010.pdf 5 http://www.atmel.com/Images/doc7516.pdf 6 http://www.usb.org/developers/docs/devclass_docs/usb_msc_cbi_1.1.pdf

3. Angriffsmöglichkeiten

In den letzten Jahren wurden verschiedenste Angriffsszenarien entwickelt, die Unzulänglichkeiten des USB-Protokolls und dessen Implementierungen in den Betriebssystemen ausnutzen. Die Angriffe unterschieden sich dabei sowohl in der Art der Durchführung als auch in der Zielsetzung wesentlich voneinander. Eine prinzipielle Einteilung in Kategorien ist dementsprechend schwierig und von vorneherein unvollständig. Zur besseren Übersicht werden daher im Folgenden bedeutende Beispiele für Angriffe durch präparierte USB-Geräte zu Gruppen zusammengefasst, sofern sie systematisch miteinander verwandt sind.

3.1. Physikalisch präparierte Geräte

In diese Gruppe fallen jene Geräte, die grundsätzlich dafür entwickelt werden, über die USB-Schnittstelle in ein System einzugreifen und es möglichst unbemerkt zu beeinflussen. Im Gegensatz zur Software-Manipulation bestehender Geräte, können Angreifer die verwendete Hardware frei nach Bedarf anpassen und ein Gerät somit sehr zielgerichtet konfigurieren. Nachfolgend werden zwei verbreitete Typen von präparierten Geräten exemplarisch vorgestellt.

3.1.1. Hardware Keylogger

Um Tastatureingaben versteckt und ohne zusätzlicher Software zu protokollieren, werden Hardware Keylogger zwischen Computer und Tastatur angeschlossen. Aufgezeichnete Daten werden auf dem Gerät gespeichert und können dort vom Angreifer abgerufen und danach gelöscht werden. Weil ein Hardware-Keylogger nicht in ein Betriebssystem eingreift, funktioniert er unabhängig davon und kann auch von einer Software-Lösung nicht enttarnt werden.

Die praktische Einsetzbarkeit von Hardware-Keylogger hängt für einen Angreifer unter anderem von folgenden Faktoren ab:

- Für die Installation des Geräts wird kurzzeitig physikalischer Zugriff auf den betreffenden Computer benötigt. Sofern das Auslesen der protokollierten Tastenanschläge nicht über Funk geschieht, ist dafür ebenso Zugriff notwendig.
- Je nach Funktionsumfang und Ausstattung des Keylogger fällt ein finanzieller Aufwand an, um die geeignete Hardware zu beschaffen und zu präparieren.
- Für Angreifer besteht das Risiko, dass der Keylogger optisch als solcher erkannt wird.

Als USB-Gerät muss sich auch der Keylogger beim Computer registrieren. Um dabei möglichst nicht aufzufallen, können als Hersteller- und Produktidentifikation z.B. die Daten eines „generischen USB-Hub" missbraucht werden [2]. Weil sich dieser „Device Descriptor" nicht von konventionellen USB-Hubs unterscheidet, ist eine Enttarnung des Keylogger durch Software nahezu aussichtslos.

3.1.2. Programmierbare „Human Interface Devices" (HID)

Über USB angeschlossene Eingabegeräte wie Tastatur und Maus zeichnen sich durch die Verwendung einer gemeinsamen USB-Geräteklasse aus. Sie wird ebenso eingesetzt von programmierbaren HID, die reale Geräte simulieren und Eingaben automatisiert vornehmen. Angreifer wiederum können diese Art von Geräten einsetzen um beliebige Tastatureingaben und Abfolgen von Befehlen auf einem System auszuführen.

Ein bekanntes Beispiel für ein sehr anpassbares HID in der Form eines USB-Sticks ist „Rubber Ducky" [7] . Wird dieses Gerät an einen Computer angeschlossen, simuliert es eine Tastatur und führt zuvor von einem Angreifer konfigurierte Befehle aus. Das präparierte HID stellt somit eine Funktionalität bereit, die in etwa mit einem tatsächlich vor dem Computer sitzenden Angreifer vergleichbar ist. Das Gerät ist darüber hinaus in der Lage, das verwendete Betriebssystem zu ermitteln und je nachdem unterschiedliche Befehle abzuarbeiten.

7 https://github.com/hak5darren/USB-Rubber-Ducky/wiki Programmierbare HID eignen sich beispielsweise für folgende Anwendungsszenarien:

- Download und Ausführung einer schadhaften Datei um dem Angreifer umfassendere Möglichkeiten auf dem kompromittierten System einzuräumen.
- Übertragung persönlicher Informationen wie z.B. Passwörter an fremde Server

- Dateisystem-Operationen wie Kopieren oder Entfernen von Dateien.

Da sich die vom HID gesendeten Tastatureingaben ohne weiteres nicht von denen eines gewöhnlichen Benutzers unterscheiden lassen, sind sie für einen Virenscanner nicht als bedrohlich erkennbar. Ein Schutz kann frühestens aktiv werden, wenn eine Datei heruntergeladen werden soll, die der Virenscanner als schadhaft erkennt.

Um als HID zu agieren, muss sich ein USB-Gerät beim System seiner Funktionalität entsprechend identifizieren. Ein möglicher Ansatz um programmierbare HID durch Software zu erkennen, ist die Überprüfung ob mehrere Tastaturen zugleich bei einem System registriert sind. Solange sich ein USB-Gerät jedoch beim System nicht als HID erkennbar macht, kann es logischerweise auch nicht als solches detektiert werden.

3.2. Angriffe auf bzw. durch den USB-Gerätetreiber

Für bekannte Geräteklassen setzen moderne Betriebssysteme üblicherweise generische Treiber ein, die die Systeme bereits mitbringen. Signalisiert ein USB-Gerät durch Setzen der Geräteklasse 255 die Notwendigkeit eines proprietären Treibers, wird zumeist auf Internetquellen zurückgegriffen um von dort einen passenden zu beziehen. Weil Gerätetreiber in den meisten Fällen über uneingeschränkte Systemrechte verfügen, kann die erfolgreiche Kompromittierung eines Treibers dazu führen, dass einem Angreifer umfassende Privilegien verliehen werden.

Ein geeigneter Gerätereiber dient der Steuerung eines USB-Geräts und interagiert dazu mit dem Mikrocontroller des Geräts. Durch Schwachstellen in fehlerhaft programmierten Treibern können Angreifer über präparierte USB-Geräte mitunter die Privilegien des Treibers für ihre Zwecke missbrauchen. Dazu schickt ein Gerät beispielsweise entsprechend manipulierte Identifikations-Informationen an den verwundbaren Treiber. Im Zuge dessen wird dann eine Schwachstelle ausgenutzt, die einem Angreifer eine Möglichkeit zur Ausführung von schadhaftem Code bietet [8910] (Code Execution durch Buffer Overflow).

Unabhängig davon ob der Quellcode eines Treibers einsehbar ist oder nicht, ist die Evaluierung der Fehleranfälligkeit keine triviale Angelegenheit. Als präventive Maßnahme empfiehlt es sich jedoch, gleich wie für das Betriebssystem, auch für Treiber fortlaufend Updates einzuspielen.

3.3. Manipulierte Firmware

Durch eine Änderung der Geräte-Firmware lassen sich das Verhalten und die Funktionalität von USB-Geräten beeinflussen. Um die Firmware eines USB-Geräts auszutauschen, sieht der USB-Standard einen „Device Firmware Upgrade“-Mechanismus vor [11] . Herstellern von Geräten wird so die Möglichkeit gegeben, ihre Produkte mit aktualisierter Firmware auszustatten. Dabei war allerdings von vorneherein nie vorgesehen, dass die Urheberschaft von Firmware überprüfbar sein soll, z.B. durch digitale Signaturen (Code Signing). Demzufolge ist es, abgesehen vom Hersteller, auch Angreifern möglich, die Firmware von USB-Geräten durch eigens modifizierte zu ersetzen.

Durch die Manipulation der Firmware können Angreifer sowohl die Funktionalität des Geräts an die eigenen Vorstellungen anpassen, als auch das Gerät nach Änderung des hinterlegten „Device Descriptor“ als ein komplett anderes ausgeben.

8 https://labs.mwrinfosecurity.com/system/assets/173/original/mwri_linux-usb-buffer-overflow_2009-10-29.pdf 9 https://labs.mwrinfosecurity.com/blog/2011/07/14/usb-fuzzing-for-the-masses/ 10 https://media.blackhat.com/bh-dc-11/Larimer/BlackHat_DC_2011_Larimer_Vulnerabiliters%20w-removeable%20storage-Slides.pdf 11 http://www.usb.org/developers/docs/devclass_docs/DFU_1.1.pdf
Unter Erhaltung der vorgesehenen Funktionalität kann z.B. die Firmware einer Webcam so angepasst werden, dass über die eingebaute LED fortan nicht mehr ersichtlich ist, ob gerade ein Video aufgezeichnet wird [3]. Dasselbe Gerät kann durch eine anderweitige Firmware-Änderung dazu gebracht werden, sich als HID auszugeben und festgelegte Befehle auf dem System auszuführen. Das geschilderte Szenario beschränkt sich nicht auf einen spezifischen Gerätetyp, wie etwa Webcams. Ein vergleichbarer Angriff ist beispielsweise realisierbar, wenn sich per USB an einen Computer angeschlossene Mobiltelefone als HID identifizieren [4].

Das Auslesen und die Modifikation einer bestehenden Firmware bzw. das Einspielen einer eigenen sind mit nicht unerheblichem Arbeits- und Zeitaufwand verbunden:

- Typischerweise hat der Angreifer keinen Zugriff auf den Quellcode der offiziellen Hersteller-Firmware. Als Konsequenz muss er zuerst die Kommunikation eines Computers mit einem USB-Gerät in geeigneter Form aufzeichnen und verstehen. Unter Microsoft Windows kann dieser Schritt z.B. durch die Programme USBPcap [12] und Wireshark [13] unterstützt stattfinden.
- Ein Angreifer muss herausfinden, wie sich eine bestehende Firmware von einem Gerät auslesen lässt, um sie anschließend modifizieren zu können.
- Firmware, deren Quellcode nicht vorliegt, muss ein Angreifer durch Reverse Engineering analysieren und kann sie frühestens dann anpassen und erweitern wenn das dafür notwendige Verständnis erarbeitet wurde.
- Um eine Firmware einzuspielen, muss vom Hersteller des Mikrocontrollers die Möglichkeit gegeben sein, die Firmware per USB auszutauschen. Ist dies der Fall, muss dem Angreifer eine entsprechende Hersteller-Applikation vorliegen oder bekannt sein, welche Befehle der Mikrocontroller des USB-Geräts erwartet, sodass eine Aktualisierung vorgenommen werden kann. Wenn das nicht zutrifft, kann der Angriff in der beschriebenen Form nicht stattfinden.

Im folgenden Kapitel wird Bezug genommen auf eine aktuelle Form der Firmware-Manipulation bei USB-Sticks.

4. „BadUSB“-Angriff

Im August 2014 haben die Sicherheitsforscher Nohl et al. vorgeführt, wie sich die Firmware von handelsüblichen USB-Sticks manipulieren lässt und dabei mehrere mögliche Angriffsszenarien aufgezeigt [14] . Der „BadUSB“-Angriff illustriert plakativ die Gefahr, die von manipulierten USB-Geräten ausgeht und verdeutlicht, dass gegenwärtig Mechanismen keinen hinreichenden Schutz davor bieten können. Nachfolgend wird die Funktionsweise des Angriffs erläutert und die mögliche Tragweite durch praktische Fallbeispiele veranschaulicht.

4.1. Details der Firmware-Manipulation

Bei der auf dem SCSI-Standard basierenden Kommunikation zwischen Computer und USB-Sticks, haben Hersteller von Mikrocontrollern die Möglichkeit, zusätzliche eigene Befehle einzuführen. Entsprechenden Applikationen wird auf diese Weise ermöglicht, gerätebezogene Informationen über den USB-Stick auszulesen oder neu zu schreiben. Sinnvoll verwendbar ist diese Funktionalität beispielsweise für die Problemdiagnose oder eine reguläre Aktualisierung der Firmware.

Im konkreten Fall wurde der Angriff mit einem USB-Stick durchgeführt, der einen Mikrocontroller der Firma Phison [15] verwendet. Über die proprietären SCSI-Befehle des Mikrocontrollers kann unter anderem die Firmware des Geräts ausgelesen und neu geschrieben werden [16] .

Angelehnt an das in Kapitel 3.3 beschriebene Konzept zur Firmware-Manipulation, lässt sich der „BadUSB"-Angriff wie folgt nachvollziehen:

12 http://desowin.org/usbpcap/ 13 https://www.wireshark.org 14 https://srlabs.de/badusb/ 15 http://www.phison.com 16 https://bitbucket.org/flowswitch/phison/wiki/ScsiCommands 1. Die eingesetzte Firmware kann grundsätzlich durch Ausführung des entsprechenden SCSI-Befehls vom Gerät ausgelesen werden. Weil Nohl et al. die Firmware jedoch im Internet vorfinden [17], wird dieser Schritt übergangen. Entsprechender Code um das das Auslesen vom Gerät zu realisieren, wurde später von Caudill et al. veröffentlicht [18] . 2. Die Firmware wird durch Reverse Engineering hinlänglich analysiert um schließlich geeignete Punkte zu finden, an denen sie durch eigenen Code erweitert werden kann. Durch eine Modifikation bestehender Inhalte können zudem Geräteklassen, Endpunkte und zur Identifikation dienende Verweise auf den Hersteller und das Produkt angepasst werden. 3. Vergleichbar mit einer gewöhnlichen Aktualisierung wird die modifizierte Firmware in den Speicher des Mikrocontrollers übertragen. Etwaige Überprüfungen aus welcher Quelle die Firmware stammt, sind nicht vorgesehen und finden somit auch nicht statt.

Obwohl die Firmware-Manipulation bei USB-Sticks bisher nur mit Mikrocontroller von Phison demonstriert wurde, ließe sich die gewählte Methodik grundsätzlich auch bei anderen Herstellern anwenden, sofern die Firmware per USB aktualisiert werden kann.

4.2. Angriffsmöglichkeiten durch „BadUSB"

Die umfassende Anpassbarkeit der Firmware ermöglicht weitreichende Änderungen am Verhalten des USB-Sticks. Im Weiteren wird durch mehrere Fallbeispiele exemplarisch dargelegt, welche Angriffsszenarien sich durch manipulierte USB-Sticks umsetzen lassen:

- **Ein USB-Stick registriert sich als Tastatur (HID)** Nach Änderung der Geräteklasse kann sich ein USB-Stick beispielsweise als HID dem System gegenüber identifizieren. Vergleichbar mit einem physikalisch präparierten Eingabegerät (siehe Kapitel 3.1.2), können über den manipulierten USB-Stick beliebige Tastatureingaben ausgeführt werden.

Durch die unterschiedliche Art und Weise wie Betriebssysteme angeschlossene USB-Geräte eruieren („USB Enumeration"), kann ein USB-Stick feststellen, welches System in Verwendung ist [5]. Die über das HID ausgeführten Befehle können dadurch an das gegenwärtig benutzte System angepasst werden. In letzter Konsequenz deutet dieser Umstand darauf hin, dass das grundsätzliche Problem nicht auf die Unzulänglichkeit eines gewissen Betriebssystems zurückführbar ist.

- **Ein USB-Stick fungiert als Netzwerkadapter** Durch Hinzufügen der Geräteklasse eines Netzwerkadapters kann sich ein USB-Stick als passendes Gerät einem Betriebssystem zur Verwendung anbieten.

In einem möglichen Angriffsszenario fordert ein Betriebssystem vom Netzwerkadapter über DHCP eine Netzwerkkonfiguration an. Der Adapter liefert daraufhin eine IP-Adresse und DNS-Server zurück, spart den Gateway jedoch aus. Bei richtiger Einstellung ist somit erreichbar, dass DNS-Anfragen von den Servern des Angreifers aufgelöst werden. Der eigentliche Datenverkehr wird weiterhin über einen gewöhnlichen Netzwerkadapter mit Gateway geleitet; Domains lösen mitunter jedoch auf irreguläre IP-Adressen auf.

Es sei angemerkt, dass ein USB-Stick, der sich als Netzwerkadapter identifiziert, nicht zwangsläufig bei jeder Anfrage des Systems das Verhalten eines authentischen Adapters widerspiegeln muss. Die Abfrage der transferierten Datenmenge kann etwa nur fiktive Werte zurückliefern, da die Hardware des präparierten USB-Sticks technisch gar nicht in der Lage wäre, tatsächlich Daten zu übertragen.

17 http://www.usbdev.ru/files/phison/ 18 https://github.com/adamcaudill/Psychson

- **Ein USB-Stick als sich selbst-replizierender Virus** Sobald ein manipulierter USB-Stick an einen Computer angeschlossen wird, können durch entsprechend schadhafte Aktionen auch andere Geräte durch die Übertragung modifizierter Firmware manipuliert werden.

Denkbar ist beispielsweise, dass ein manipulierter USB-Stick das Betriebssystem eines Computers mit Malware infiziert, wodurch wiederum die Firmware aller angeschlossenen USB-Sticks austauscht wird. Weil USB-Geräte bereits beim Starten eines Computers initialisiert werden, kann ein USB-Stick auch zu dieser Zeit bereits das System infizieren. Dafür genügt es, wenn sich das Gerät beim Laden des BIOS als HID ausgibt, dadurch die Boot-Reihenfolge ändert, und sich in weiterer Folge als Massenspeicher neu registriert, von dem schließlich Malware geladen wird. Diese Schadsoftware kann wiederum das zu ladende Betriebssystem in einer frühen Phase infizieren, in der noch keine Anwendungssoftware einen Angriff abwehren könnte.

Die große Bandbreite an unterschiedlichen Angriffsszenarien untermalt die prinzipielle Problematik, wonach sich ein optisch zumeist unscheinbarer USB-Stick als jedes beliebige USB-Gerät ausgeben und im Rahmen der jeweiligen Möglichkeiten operieren kann.

4.3. Schutzmechanismen

Die schwere Unterscheidbarkeit zwischen legitimen und manipulierten USB-Geräten stellt eine große Herausforderung dar, wenn es darum geht, mögliche Angriffe zu erkennen und abzuwehren.

Ein grundsätzlicher Schutz gegen den „BadUSB“-Angriff ist gegeben, wenn keine USB-Geräte verwendet werden oder die Nutzung der USB-Schnittstelle im BIOS deaktiviert ist. Diese rigide Lösung ist in der Praxis jedoch oftmals keine hinnehmbare Option, da sie einem Verzicht der Nutzung von USB gleichkommt (vgl. [6], S. 171ff.).

Softwarelösungen, die Schutz vor Angriffen bieten sollen, unterliegen der Einschränkung, dass sie erst wirksam werden können, wenn das Betriebssystem bereits gestartet ist. Sollte ein USB-Stick bereits zuvor angesteckt gewesen sein, könnte eine Infektion des Systems mit Malware bereits stattgefunden haben.

Im weiteren Sinne stellt sich die Frage, anhand welcher Kriterien manipulierte Firmware als solche erkannt werden könnte. Im USB-Standard und daraus abgeleiteten Kommunikationsprotokollen ist nicht vorgesehen, dass die Integrität und Authentizität der Firmware auf USB-Geräten prüfbar seien. Analog dazu ist es auch nicht möglich, Geräte durch eine Kennung eindeutig zu identifizieren, ohne dass sich diese Werte in der Firmware ändern ließen. Die beschränkte Wirksamkeit technischer Maßnahmen bedingt, „BadUSB"-Angriffe auch in organisatorische Richtlinien einzubeziehen.

5. Fazit

Dank der Flexibilität von USB ist es möglich, eine Vielzahl unterschiedlicher Geräteklassen über eine gemeinsame Schnittstelle ohne Authentifizierung einzubinden. Weil ein Computer a priori nicht wissen kann, welches USB-Gerät angeschlossen ist, vertraut er auf die Informationen, die das Gerät liefert um sich zu identifizieren.

Sind Angriffe durch physikalisch präparierte Hardware oftmals noch optisch als solche erkennbar, bezeugt der „BadUSB"-Angriff plakativ die Grenzen, an die gegenwärtige, schutzbietende Mechanismen stoßen. Zugleich wird dabei aufgezeigt, dass das grundsätzliche Problem unabhängig vom eingesetzten Betriebssystem besteht und auf Mängel im USB-Standard zurückzuführen ist.

Nach Erarbeitung der wesentlichen Definitionen zu USB wurde in diesem Dokument eine Zusammenfassung über verschiedene Angriffsmöglichkeiten durch manipulierte USB-Geräte gegeben. Anhand des „BadUSB"-Angriffs wurde anschaulich erläutert, welche Möglichkeiten sich durch die Manipulierbarkeit von Firmware erschließen. Abschließend wurde der Einsatz möglicher Schutzmechanismen diskutiert.

6. Literaturverzeichnis

[1] B. Dewey: „'Plug and Root', the USB Key to the Kingdom". http://www.blackhat.com/presentations/bh-usa-05/BH_US_05-Barrall-Dewey.pdf

[2] F. Mihailowitsch: „Detecting Hardware Keyloggers". http://conference.hackinthebox.org/hitbsecconf2010kul/materials/D1T1%20-%20Fabian%20Mihailowitsch%20-%20Detecting%20Hardware%20Keyloggers.pdf.

[3] M. Brocker, S. Checkoway: „iSeeYou: Disabling the MacBook Webcam Indicator LED", *Proceedings of the 23rd USENIX Security Symposium,* 2014.

[4] A. Stravrou, Z. Wang: „Exploiting Smart-Phone USB Connectivity For Fun And Profit". https://media.blackhat.com/bh-dc-11/Stavrou- Wang/BlackHat_DC_2011_Stavrou_Zhaohui_USB_exploits-Slides.pdf

[5] A. Davis: „Revealing Embedded Fingerprints: Deriving intelligence from USB stack interactions". https://media.blackhat.com/us-13/US-13-Davis-Deriving-Intelligence-From- USB-Stack-Interactions-Slides.pdf. Aufgerufen am 7.11.2014

[6] A-SIT: „Österreichisches Informationssicherheitshandbuch". 2014.