Trusted Computing mit Windows Vista und Windows Server „Longhorn“

Thomas Müller
Hochschule der Medien, Stuttgart
Thomas.Mueller@tmxm.de

Kurzfassung:
Am 09.11.2006 veröffentlichte Microsoft im Rahmen seines Customer Preview Program (CPP) die Release to Manufacturing (RTM) Version des neuen Betriebssystems Windows Vista™. Windows Vista ist der Nachfolger von Windows XP™ und wird für 32 Bit und 64 Bit-Systeme zur Verfügung stehen. Auch die Server-Variante Windows Server® „Longhorn“ wurde in der Beta 2 Version bereits veröffentlicht. Vor allem im Bereich Sicherheit will Microsoft mit den beiden Betriebssystemen einen neuen Maßstab setzen. Speziell der Vorgänger Windows XP ist bekannt für seine Anfälligkeit für Manipulationen durch Schadsoftware (Viren, Würmer, Trojanische Pferde und Rootkits). Als Hauptkritikpunkte am Sicherheitsdesign von Windows XP gelten die Rechte- und Benutzerkonten-Verwaltung sowie die einfache Manipulation des Betriebssystemkerns durch Nachladen von schadhaften Systemtreibern. Microsoft hat beide Betriebssysteme mit zahlreichen neuen Sicherheitsfunktionen ausgestattet und viele der bisherigen überarbeitet. Einige der neuen Funktionen bleiben jedoch den beiden 64 Bit-Varianten der Betriebssysteme vorbehalten. Als erste Windows Versionen enthalten Vista und Longhorn Unterstützung für Trusted Platform Module (TPM). Die Kombination der neuen Sicherheits-Funktionen soll laut Microsoft den Aufbau und Betrieb eines Trusted Computing Systems (sichere Rechenumgebung) ermöglichen. Im Rahmen einer Diplomarbeit wurden diese Sicherheitsfunktionen untersucht und mit den Anforderungen an ein Trusted Computing System verglichen.

1. Einleitung

Für die Untersuchung der neuen Sicherheitsfunktionen in Windows Vista muss zunächst der Begriff des *Trusted Computing Systems* für die Arbeit eindeutig definiert werden. Die beiden bedeutendsten Bestandteile eines *Trusted Computing Systems* sind eine sichere Rechnerplattform (*Trusted Computing Platform*) und das darauf aufbauende sichere und vertrauenswürdige Betriebssystem (*Trusted Operating System*).

Der Begriff der *Trusted Computing Platform i*st definiert durch die Spezifikationen der *Trusted Computing Group* (TCG) [Tcg01, Tcg03]. Sie beschreiben die Maßnahmen zur Überprüfung der Integrität von Hard- und Software durch Einsatz eines *Trusted Platform Moduls* (TPM). Das TPM ist eine zusätzliche Hardwarekomponente, die fest mit der Hauptplatine des Computersystems verbunden ist. Die Funktionen des TPM entsprechen in etwa denen einer Smart Card, mit dem Unterschied, dass das TPM an ein Computersystem gebunden ist und dieses dadurch eindeutig identifiziert werden kann. Die Smart Card hingegen ist nicht an ein System gebunden und wird u. a. zur Identifikation einer natürliche Person verwendet. Ein weiteres Alleinstellungsmerkmal des TPM sind die *Platform Configuration Register* (PCR), sie dienen der sicheren Speicherung von Prüfsummen die den aktuellen Zustand des System repäsentieren. Neben dem TPM benötigt eine Trusted Computing Platform eine weitere Komponente die „*Core Root Of Trust For Measurement*“ (CRTM), die in der Regel als Erweiterung des BIOS implementiert ist. Die CRTM enthält die ersten Instruktionen, die beim Start des Systems ausgeführt werden, und erstellt Prüfsummen (SHA-1 Hash) von sich selbst

und dem BIOS und legt diese im ersten PCR (PCR-0) des TPM ab. Erst nach dieser Messung übergibt die CRTM die CPU an das BIOS, welches ebenfalls Prüfsummen über seine Konfiguration, weitere System ROMs und schließlich über den *Master Boot Record* (MBR) der ersten Festplatte erzeugt und in den Registern PCR-1 bis PCR-7 hinterlegt. Dieser Prozess dient dem Aufbau einer Vertrauenskette (*Chain of Trust*) durch alle am Bootvorgang des System beteiligten Betriebsstufen und deren Komponenten.

Eine *Trusted Computing Platform* ist also ein Computersystem, das um ein TPM und eine CRTM erweitert wurde, und bildet dadurch den Vertrauensanker (*Root of Trust*) für darauf aufsetzende Trusted Computing Anwendungen. Durch das Erzeugen der beschriebenen Prüfsummen initiiert die *Trusted Computing Platform* die Vertrauenskette vom BIOS POST bis zum MBR und stellt damit sicher, dass alle bisher ausgeführten Komponenten anhand ihrer Prüfsumme eindeutig identifizierbar sind. Die *Trusted Computing Platform* besteht nur aus Hardware und Firmware und arbeitet dadurch unabhängig von installierten Betriebsystemen. Systeme, die konform zu den Spezifikationen der TCG sind, bieten die Möglichkeit, diese Funktionen im BIOS zu aktivieren bzw. zu deaktivieren.

Die Definition des *Trusted Operating System* (Trusted OS) ist dagegen weitaus komplexer und umfasst zahlreiche Sicherheitsfunktionen und -konzepte. Eine eindeutige Definition des Begriffs oder sogar Spezifikationen, wie für die Trusted Computing Platform, sind zum aktuellen Zeitpunkt nicht verfügbar. Die für diese Arbeit relevante Bedeutung des Begriffs wird deshalb vom Autor im folgenden in Form eines Anforderungskatalogs beschrieben.

Die Aufgaben eines *Trusted OS* lassen sich auf einer abstrakten Ebene in vier Kategorien einteilen:

- Weiterführung der Vertrauenskette (*Chain of Trust*)

  Wie beschrieben, erzeugt die *Trusted Computing Platform* Prüfsummen aller am Systemstart beteiligter Komponenten bis zum MBR. Der ausführbare Teil des MBR ist bereits Teil des Betriebssystems und enthält Instruktionen zum Start der nächsten Komponente. Dies ist in der Regel ein Bootmanager, der eine Auswahl des zu startenden Betriebssystems anbietet. Der Bootmanager ruft entsprechend der Auswahl das Startprogramm des jeweiligen Betriebssystems (*OS-Loader*) auf, welches die nötigen Boot-Treiber und schließlich den Kern des Betriebssystems (*OS-Kernel*) startet. Der OS-Kernel lädt dann die zur Laufzeit verwendeten Treiber und startet weitere Dienste (Daemons oder Services). Um die *Chain of Trust* im Sinne des Trusted Computing fortsetzen zu können, muss ein *Trusted OS* jeden der beschriebenen Schritte um die Erzeugung entsprechender Prüfsummen ergänzen. Konkret bedeutet dies, bereits der MBR benötigt zusätzliche Routinen, die eine Messung des Bootmanagers ermöglichen, bevor dessen Instruktionen ausgeführt werden. Der Bootmanager wiederum muss eine Möglichkeit vorsehen, eine Prüfsumme über den *OS-Loader* zu erzeugen. Dieses Konzept muss bis zum vollständigen Start des Betriebssystems fortgesetzt werden, erst dann kann der Systemstart als „*Trusted Boot*“ bezeichnet werden [Smith01].

- Bewertung der Systemintegrität

  Entgegen der selbst in Fachartikeln zu findenden Aussage, die *Trusted Computing Platform* stelle die Integrität eines Computersystems sicher, bietet sie lediglich die Möglichkeit zur Protokollierung der ersten Instanzen des Startvorgangs und zur Identifikation der daran beteiligten Komponenten. Eine Bewertung der Integrität dieser Komponenten basiert zwar auf den dabei erzeugten Prüfsummen, ist jedoch Aufgabe des *Trusted OS* und erfordert die Implementierung entsprechender Metriken. Hierfür sollte ein *Trusted OS* zumindest einen Teil der durch die *Trusted Computing Platform* im TPM hinterlegten und sämtliche während des restlichen Bootvorgangs (*Chain of Trust*) erzeugten Prüfsummen mit entsprechenden Referenzwerten vergleichen. Zu welchen Zeitpunkten diese Überprüfung stattfindet und welche Komponente des Betriebssystems dafür verantwortlich zeichnet, ist den Entwicklern eines Betriebssystems überlassen. Jedoch sollte darauf geachtet werden, dass die prüfende

Komponente selbst Teil der Vertrauenskette ist und somit über eine Prüfsumme verfügt. Der Mechanismus, mit dem sicher gestellt wird, dass ein Systemstart nur im Falle einer positiven Integritätsprüfung stattfindet, heißt „*Secure Boot*“ [Smith01].

- Erhaltung der Systemintegrität

  Nach der erfolgreichen Umsetzung des *Secure Boot* Konzeptes befindet sich das System in einem vertrauenswürdigen Zustand, was bedeutet, dass sämtliche am Startvorgang beteiligten und alle im Arbeitsspeicherbereich des Betriebssystems befindlichen Komponenten vor ihrer Ausführung gemessen und auf ihre Integrität überprüft wurden. Eine weitere Aufgabe des *Trusted OS* ist es nun, diesen Zustand über die Laufzeit des Betriebssystems zu erhalten. Hierfür müssen alle Softwarekomponenten (z. B. Hardwaretreiber), die zu einem späteren Zeitpunkt vom Betriebssystemkern nachgeladen werden, die Vertrauenskette erweitern und ebenfalls auf ihre Integrität überprüft werden. Diese Maßnahme ist jedoch wenig effektiv, wenn die Instruktionen nach dem Laden in den Speicherbereich des Betriebssystems verändert werden können. Es muss also sichergestellt sein, dass ein Zugriff auf diese Speicherbereiche nur von vertrauenswürdigen Komponenten durchgeführt werden kann. Zusätzlich sollten die kritischen Datenstrukturen im Speicher regelmäßig mit den vor dem Laden erzeugten Prüfsummen verglichen und somit überwacht werden. Als weiteren Schutz sollten alle zum Betriebssystem gehörigen Dateien bereits auf dem Datenträger vor unberechtigter Manipulation geschützt werden. Die beschriebenen Konzepte können zusätzlich auch auf den Anwendungsbereich (user-mode) des Betriebssystems übertragen werden. Dabei werden auch hier über sämtliche Anwendungen des Benutzers vor ihrer Ausführung durch das Betriebssystem Prüfsummen erzeugt. Dies kann auch dahingehend erweitert werden, dass nur Anwendungen, die auf Grund ihrer Prüfsummen als integer eingestuft werden, zur Ausführung kommen.

- TPM-Schnittstelle für Anwendungen

  Neben den beschriebenen Anforderungen sollte ein *Trusted OS* die durch das TPM bereit gestellten Funktionen auch allen anderen Anwendungen zur Verfügung stellen. In der Regel wird dies durch eine Implementierung der TCG-Spezifikation [Tcg02] für einen *Trusted Software Stack* (TSS) erreicht. Durch den Einsatz eines TSS können die Trusted Computing Konzepte durch Applikationen erweitert werden und es entstehen neue Anwendungsbereiche.

Dieser Anforderungskatalog erhebt keinen Anspruch auf Vollständigkeit - so wurden bewusst einzelne Aspekte des Trusted Computing für diese Untersuchung außen vor gelassen. Die Fähigkeit eines Systems seinen aktuellen Zustand vertrauenswürdig an anderen Computersystemen zu übermitteln (*Remote Attestation*), ist einer dieser Aspekte. Der Einsatz dieser Fähigkeit soll den Aufbau von Vertrauensstellungen in verteilter Computersysteme ermöglichen. Die Langfassung der Diplomarbeit berücksichtigt unter anderem auch dieses Szenario.

2. Windows Vista als Trusted Computing System

Im Folgenden wird der zuvor erstellte Anforderungskatalog mit der Umsetzung im Betriebssystem Windows Vista verglichen. Hierfür werden die für die Untersuchung relevanten Sicherheitsfunktionen kurz beschrieben und falls möglich einer der vier im vorherigen Abschnitt beschriebenen Kategorien zugeordnet. Dem folgt eine Analyse, welche Teile des Anforderungskatalogs abgedeckt werden, und welche Angriffe gegen die Sicherheitsfunktion denkbar bzw. verfügbar sind. Am Ende der Untersuchung werden die nicht adressierten Anforderungen aufgezeigt und die daraus resultierenden Angriffsvektoren beschrieben.

3. Windows Vista TPM Support

Das TPM kann unter Vista mit Hilfe eines Assistenten verwaltet werden. Er bietet die Möglichkeit

das TPM in vom Betriebszustand *deactivated* in den Zustand *activated* zu überführen. Ist es jedoch im Zustand *disabled* kann es nur im BIOS aktiviert werden[1]. Da für den Zustandswechsel von „*deactivated*“ nach „*activated*“ und für das Einrichten des TPM-Eigentümers der Beweis des physikalischen Zugriffs auf das System erbracht werden muss, informiert Vista das BIOS über die gewünschte Befehlsausführung. Das BIOS präsentiert dem Benutzer beim nächsten Start des Systems einen Dialog, dessen Bestätigung den Beweis des physikalischen Zugriffs darstellt und den Zugriff auf den Befehl freigibt. Befindet sich das TPM im Zustand „*activated*“, konfiguriert der Assistent das TPM für die spätere Verwendung durch die BitLocker Festplattenverschlüsselung[2], indem der Assistent den Besitz des TPM übernimmt (TPM_TakeOwnership). Hierbei wird ein RSA Schlüsselpaar im TPM erzeugt, mit dem beliebige Daten durch das TPM verschlüsselt werden können. Diese verschlüsselten Daten werden jedoch nicht innerhalb des TPM gespeichert sondern auf einem der regulären Datenträgern. Um das TPM im Folgenden vor unberechtigtem Zugriff zu schützen, erfordert der Zugriff auf sicherheitskritische Funktionen des TPM die Angabe des Eigentümer-Passworts, welches automatisch durch den Assistenten erzeugt werden kann. Dieses Kennwort wird bei entsprechender Konfiguration der Gruppenrichtlinien im Active Directrory der Windows Domäne abgelegt. Die Möglichkeit, den Zugriff auf das im TPM erzeugte RSA Schlüsselpaar ebenfalls durch ein Passwort zu schützen, hat Microsoft jedoch nicht wahrgenommen, was in erster Linie ein Zugeständnis an die Benutzerfreundlichkeit ist. Wäre dieses Passwort gesetzt, müsste der Benutzer bei jeder Verwendung des Schlüsselpaares das Passwort angeben, im Falle einer aktivierten BitLocker Festplattenverschlüsselung somit bei jedem Systemstart.

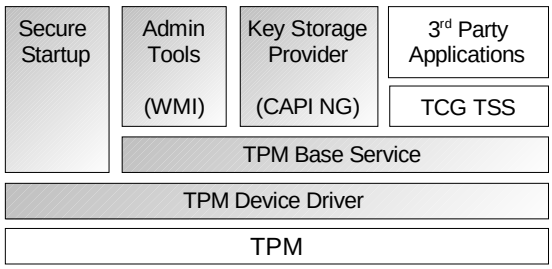

Abbildung 1: TPM Trusted Software Stack

Bei der Implementierung des *Trusted Software Stack* (TSS) für den Zugriff auf das TPM hat sich Microsoft für eine von der TCG-Spezifikation[3] abweichende Umsetzung entschieden. Der Zugriff auf die Funktionen des TPM erfolgt entweder über eine WMI-Schnittstelle[4] oder über die Microsoft Krypto-API. Diese beiden Schnittstellen bilden jedoch nur eine Teilmenge der verfügbaren Funktionen ab. Der TPM Base Service regelt hierbei den simultanen Zugriff auf das TPM und bietet eine Schnittstelle an, auf die vollständige TSS-Implementierungen aufsetzen können, um somit den vollen Funktionsumfang des TPM anzubieten. Der TPM Base Service bietet weiterhin die Möglichkeit die Verwendung bestimmter TPM-Befehle unter Zuhilfenahme einer Sperrliste zu blockieren. Die Sperrliste kann sowohl über die lokale Windows-Sicherheitsrichtlinie als auch zentral über Gruppenrichtlinien erstellt werden. Es können lokal keine Kommandos freigegeben werden, die per Gruppenrichtlinie gesperrt sind. Es ist jedoch möglich, zusätzliche Kommandos über die lokale Sicherheitsrichtlinie zu sperren. Interessanterweise hat Microsoft eine zusätzliche Option in den Gruppenrichtlinien vorgesehen, mit der auch die Erweiterung der Sperrliste verhindert wird. Dadurch können keine Kommandos gesperrt werden, die in den Gruppenrichtlinien als aktiv markiert sind.

[1] TCG-Spezifikation [Tcg03]
[2] Wird später im Dokument beschrieben
[3] TCG-TSS Spezifikation [Tcg02]
[4] Windows Management Instrumentation

4. Secure Startup und Full Volume Encryption (FVE) – BitLocker

Die *BitLocker* Funktion in Windows Vista besteht aus zwei Komponenten, dem *Secure Startup* und der *Full Volume Encryption* (FVE). Notwendige Vorraussetzung für die Nutzung dieser Komponenten ist die Verfügbarkeit eines TPM[5]. Auf einem System mit aktiviertem TPM[6] dient der *Secure Startup* der Fortsetzung der Vertrauenskette. Hierfür erweitert Microsoft den MBR, die letzte von der *Trusted Computing Platform* gemessene Komponente, um die Funktionalität, die nächste Stufe des Startvorgangs, den NTFS Boot Sektor (VBR), zu messen und im PCR-8 zu hinterlegen. Der NTFS Bootsektor wiederum hinterlegt die Prüfsumme über den NTFS Boot Block (BPB) im PCR-9. Zuletzt misst der NTFS Boot Block den Bootmanager und speichert dessen Wert im PCR-10. Dieser Vorgang wird bei jedem Startvorgang wiederholt.

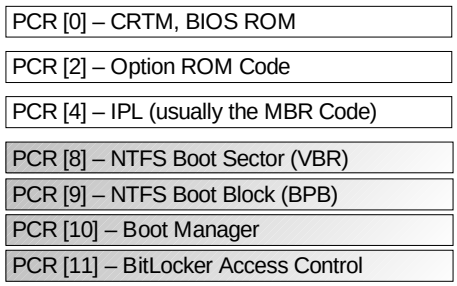

Abbildung 2: Verwendung der PCR

Die *Full Volume Encryption* bietet, zusätzlich zur bereits vorhandenen Verschlüsselung auf Dateiebene (*Encrypted File System* - EFS), die Möglichkeit, die komplette Systempartition zu verschlüsseln. In den kommenden Server Editionen „Longhorn“ soll die FVE zusätzlich auch für weitere Datenpartitionen aktiviert werden können. Diese Verschlüsselung ist im I/O-Stack des Betriebssystems als Filter-Treiber unterhalb des Dateisystems implementiert und findet dadurch für den Benutzer transparent statt. Hierdurch werden neben Daten der Benutzer auch sämtliche System-Dateien, temporäre Dateien, die Auslagerungsdatei und die Dateien des Ruhezustands (Hibernation) verschlüsselt. Um die Partition beim Startvorgang entschlüsseln zu können, muss eine weitere NTFS Partition angelegt werden. Sie enthält neben dem MBR den Bootmanager, der die Entschlüsselung startet und danach den OS-Loader (*%Windows%\system32\WINLOAD.EXE*) aufruft. Als Verschlüsselungsalgorithmus kommt eine abgewandelte[7] Version von AES zum Einsatz, wahlweise mit 128 Bit oder 256 Bit Schlüssellänge. Der Schlüssel wird als Full Volume Encryption Key (FVEK) bezeichnet und ist nochmals mittels AES-256 durch den Volume Master Key (VMK) gesichert. Dieses zweistufige Konzept hat den Vorteil, dass bei einer notwendigen Änderung des Schlüssels nicht die komplette Partition ent- und anschließend wieder verschlüsselt werden muss. Sowohl der FVEK also auch der VMK werden in einem Bereich der Partition hinterlegt auf die der Bootmanager Zugriff hat. Die genaue Lage der Schlüssel wird von Microsoft nicht dokumentiert, ein möglicher Speicherort ist der NTFS-Header der Systempartition.

Für den Schutz des VMK gibt es mehrere Optionen, neben der Möglichkeit des Schutzes durch einen externen Schlüssel auf einem USB-Stick kann auch das TPM verwendet werden. Hierzu kommt die *sealing*-Funktion des TPM zum Einsatz, sie ermöglicht es, beliebige Daten an die aktuelle Konfiguration des Systems zu binden. Dabei fließen die Inhalte der PCR in den Verschlüsselungsvorgang mit ein. Die aktuelle Konfiguration wird durch die von der *Trusted Computing Platform* und der *Secure Startup* Komponente hinterlegten Prüfsummen in den Registern PCR-0 bis PCR-11 repräsentiert. Zusätzlich lässt sich der VMK noch durch ein numerisches

[5] Die FVE kann auch ohne TPM verwendet werden.
[6] Windows Vista unterstützt ausschließlich TPM Chips in der Version 1.2
[7] Microsoft beschreibt dies als AES + Elephant Diffusor [Micro1]

Kennwort (PIN) oder einen zusätzlichen externen Schlüssel schützen. In der Standardkonfiguration verwendet die FVE nur ein Subset dieser Prüfsummen PCR-(0,2,4,8-11) und verzichtet auf den zusätzlichen Schutz durch eine PIN oder weitere Schlüssel auf einem USB-Stick[8].

Beim Systemstart verwendet der Bootmanager die *unseal*-Funktion des TPM, um den VMK zu entschlüsseln. Dies gelingt jedoch nur dann, wenn die verwendeten PCR den selben Zustand wie zum Zeitpunkt der Aktivierung von *BitLocker* aufweisen. Modifikationen an Systemkomponenten wie z. B. dem MBR oder dem BIOS führen somit zu einer Unterbrechung des Startvorgangs, welche nur durch die Eingabe des Wiederherstellungskennworts, das bei der Aktivierung von BitLocker erzeugt wird, fortgesetzt werden kann. Das PCR-11, das ebenfalls beim *sealing* und *unsealing* verwendet wird, hat dahingehend eine Sonderstellung, dass Microsoft dieses Register als „BitLocker Access Control“ beschreibt und im Falle einer aktiven FVE nach der erfolgreichen Entschlüsselung des VMK auf einen anderen Wert abändert. Dadurch soll verhindert werden, dass die Entschlüsselung zur Laufzeit des Betriebssystems wiederholt werden kann, und dass ein parallel installiertes Betriebssystem Zugriff auf den VMK erhält.

Während die *Secure Startup* Komponente eindeutig der ersten Aufgabenkategorie zuzuordnen ist – sie dient der Fortsetzung der Vertrauenskette (*Chain of Trust*) – fällt die Einordnung der Festplattenverschlüsselung etwas schwer. Ist sie aktiviert, werden zwar Modifikationen an den am Systemstart beteiligten Komponenten erkannt, es gibt jedoch keine Möglichkeit die geänderte Komponente zu identifizieren, und somit fällt es schwer, geeignete Maßnahmen zu ergreifen. Das entscheidende Problem besteht jedoch in der Tatsache, dass auch eine erfolgreiche Entschlüsselung des VMK keinen integeren Systemzustand garantiert, da hierbei lediglich sichergestellt wird, dass sich das System im selben Zustand wie bei der Aktivierung von *BitLocker* befindet. War das System jedoch bereits zu diesem Zeitpunkt kompromittiert, gibt es keine Möglichkeit, dies beim Einsatz der *Full Volume Encryption* (FVE) zu erkennen. Da die FVE neben ihrer eigentlichen Funktion, der Geheimhaltung der Daten, auch verhindert, dass Dateien des Betriebsystem durch ein parallel installiertes Betriebssystem modifiziert werden können, und somit ebenfalls zur Erhaltung der Systemintegrität beiträgt, kann auch unter dem Gesichtspunkt des Integritätsschutzes eine Aktivierung der Funktion empfohlen werden.

Die *Secure Startup* Komponente setzt die durch die *Trusted Computing Platform* begonnene Vertrauenskette sinnvoll bis zum Bootmanager von Windows Vista fort. Jedoch lassen sich auch hier zwei Kritikpunkte finden. Zum einen handelt es sich bei *Secure Startup* um keine Implementierung des *Secure Boot* Konzeptes, da *Secure Startup* selbst keine Integritätsprüfungen anhand der PCR vornimmt, und die Integritätsprüfung der FVE auf Grund des beschriebenen Umstands ebenfalls den Anforderungen nicht gerecht wird. Somit handelt es sich um eine Implementierung des *Trusted Boot* Konzeptes. Zum anderen wird die Vertrauenskette nur bis zum Bootmanager erweitert, die beiden nächsten Stufen, der *OS-Loader* (*%SystemRoot%\system32\WINLOAD.EXE*) und der Betriebssystemkern (*%SystemRoot%\system32\NTOSKRNL.EXE*), werden hierfür nicht mehr berücksichtigt. Zwar unterliegen diese beiden Komponenten weiteren Integritätsprüfungen[9], jedoch hätte dadurch eine Schwachstelle, die später im Dokument beschrieben wird, verhindert werden können.

Weiter ist es zu empfehlen, im Falle der *Full Volume Encryption* den zusätzlichen Schutz mittels PIN zu aktivieren, da die alleinige Verwendung des TPM einen entscheidenden Nachteil hat. Wird ein PC mit dieser Konfiguration gestohlen, ist es für den Angreifer zwar nicht möglich, auf die Systemdateien zuzugreifen, in denen die Benutzerpasswörter hinterlegt sind, was einen Brute-Force-Angriff auf die Passwörter verhindert. Da die PCR jedoch nach wie vor die richtigen Werte enthalten,

[8] Die Optionen (TPM+PIN,TPM+Key) sind erst nach Aktivierung in den Gruppenrichtlinien verfügbar.

[9] Kernel Integrity Checks und PatchGuard werden später im Dokument beschrieben.

entschlüsselt das TPM auf Anfrage des Bootmanagers den Volume Master Key, welcher diesen in den Arbeitsspeicher lädt. Mit speziellen PCI-Erweiterungskarten kann nun der Inhalt des Arbeitsspeichers und somit auch der VMK, durch einen zweiten PC ausgelesen werden. Solche DMA-Karten werden u. a. in der Computer-Forensik eingesetzt, weitere Informationen hierzu liefern [Petro01] und [Carr01].

5. Kernel Integrity Checks/Driver Signing (nur 64-Bit Versionen)

Eine weitere Maßnahme zur Sicherung der Systemintegrität ist die Überprüfung von Treiber-Signaturen. Hierfür müssen in den 64-Bit-Versionen der Betriebssysteme alle Treiber mit einem Zertifikat der Klasse 3 signiert sein [Micro05]. In den 32 Bit Versionen ist eine Signatur nur für die am Startvorgang beteiligten Treiber zwingend vorgeschrieben. Die Überprüfung der Treiber-Signaturen findet zu mehreren Zeitpunkten im Lebenszyklus des Betriebssystems statt.

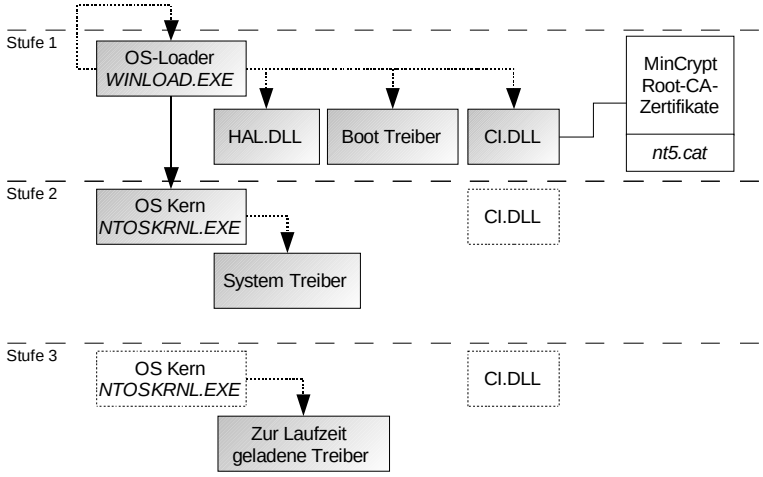

Abbildung 3: Überprüfung der Treiber Signaturen

Für die erste Überprüfung ist der *OS-Loader* zuständig. Dieser überprüft zunächst die korrekte Funktionsweise der für die Überprüfung eingesetzten Crypto-Library (*MinCrypt*) durch den Aufruf eines Selbsttests. Die *MinCrypt*-Bibliothek wird beim Übersetzen des *OS-Loaders* statisch an diesen gebunden und ist somit vor Manipulationen durch die digitale Signatur des *OS-Loaders* geschützt. Danach lädt er eine ebenfalls digital signierte Liste von gültigen Prüfsummen (*%SystemRoot%\System32\catroot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\nt5.cat*) in seinen Speicherbereich. Nun überprüft der *OS-Loader* seine eigene Integrität, indem er eine Prüfsumme über sich selbst erzeugt und mit der Prüfsumme seiner im Header der Datei hinterlegten Signatur vergleicht. Ist diese Überprüfung positiv, stellt er zusätzlich sicher, dass die errechnete Prüfsumme auch in der Liste der gültigen Prüfsummen enthalten ist. Genau das selbe Verfahren wird im Folgenden auf sämtliche Boot-Treiber angewandt, also auf alle Treiber, die zum Laden des eigentlichen Betriebssystemkerns nötig sind. Dies gilt ebenso für die nächste Instanz des Startvorgangs, den Betriebssystemkern. Dadurch werden zwei Anforderungen überprüft. Zum einen kann durch die Verifizierung der Signaturen sichergestellt werden, dass die Treiber aus einer vertrauenswürdigen Quelle stammen und seit der Signierung nicht verändert wurden. Zum anderen stellt der Vergleich mit der Liste der gültigen Prüfsummen zusätzlich sicher, dass nur von Microsoft für dieses Stadium des Systemstarts vorgesehene Treiber geladen werden. Die für die Überprüfung der digitalen Signaturen benötigten Root-CA-Zertifikate sind fest in der *MinCrypt*-Bibliothek verankert, weswegen Angriffe scheitern, die darauf basieren, die Liste der vertrauenswürdigen Root-CA-Zertifikate des Betriebssystem um CA-Zertifikate zu erweitern.

Wurden alle Überprüfungen positiv abgeschlossen, kommen der Betriebssystemkern und die verifizierten Boot-Treiber zur Ausführung. Der Betriebssystemkern ist verantwortlich für die Überprüfung der System-Dateien, hierzu zählen die einzelnen Komponenten des Betriebssystems sowie die Treiber für systemspezifische Hardware. Hierfür verwendet er einen der geladenen Boot-Treiber (*\%SystemRoot%\system32\CI.DLL*), diese Bibliothek greift ebenfalls auf die Funktionen der MinCrypt-Bibliothek zurück. Der Betriebsystemkern überprüft nun nach dem selben Verfahren wie zuvor bereits der *OS-Loader* sämtliche Treiber bevor er sie in den Speicherbereich des Betriebsystems lädt. Zusätzlich wird eine weitere Überprüfung von Treiber-Signaturen zur Laufzeit des Systems durchgeführt. Dieser Vorgang wird unter anderem beim Verbinden von neuer Hardware mit dem System ausgelöst. Hierfür ist ebenfalls der Betriebssystemkern bzw. die *CI.DLL* verantwortlich. Eine weitere interessante Funktion der *CI.DLL* ist die Überprüfung der digitalen Signatur einer Applikation, die in ihrem Applikations-Manifest administrative Rechte anfordert oder explizit vom Administrator mit vollen Rechten gestartet wird. Verfügt eine Applikation nicht über ein gültiges Zertifikat, erscheint ein Windows–Dialog, der den Benutzer über dieses Defizit informiert. Zusätzlich kann mittels Gruppenrichtlinie die Ausführung solcher Applikation grundsätzlich gesperrt werden. Im Unterschied zur Überprüfung der Treiber-Signaturen zur Laufzeit ist diese Funktion auch in den 32-Bit Versionen verfügbar.

Die Driver Signing Komponente erfüllt somit die Anforderung der Bewertung der Systemintegrität durch die Überprüfung der Treiber Signaturen und leistet einen Beitrag zum Erhalt der Integrität, indem zur Laufzeit hinzugefügte Treiber ebenfalls einer Prüfung unterzogen werden. Jedoch hat Microsoft die Chance nicht genutzt, die, im Falle eines vorhandenen TPM durch die *Trusted Computing Platform* und den Secure Startup aufgebaute Vertrauenskette (Chain of Trust) fortzusetzen. So werden zwar sämtliche am Startvorgang beteiligte Komponenten, beginnend mit dem OS-Loader, anhand ihrer Signatur überprüft und nur im Falle eines positiven Ergebnisses in die nächste Betriebsstufe gewechselt, jedoch beginnt die Überprüfung mit einem Selbsttest des *OS-Loaders*. Das Konzept einer Vertrauenskette sieht jedoch vor, dass jede Komponente vor ihrer Ausführung durch ihren Vorgänger geprüft bzw. zumindest gemessen werden muss. [Cono02] beschreibt einen Angriff auf die Driver Signing Funktion, indem die Datei des Betriebssystemkerns (*%SystemRoot%\system32 \NTOSKRNL.EXE*) modifiziert wird, dies führt jedoch dazu dass die Signatur der Datei ungültig wird. Deshalb muss zusätzlich die Datei des *OS-Loaders* (*%SystemRoot%\system32\WINLOAD.EXE*) modifiziert werden, dadurch kann neben der Überprüfung der Treiber Signaturen auch der Selbsttest des *OS-Loaders* deaktiviert werden und somit eine Erkennung des Angriffs verhindert werden. Würde der Bootmanager (*%BitLockerPartition%\bootmgr*) vor der Ausführung des *OS-Loaders* eine Prüfsumme über diesen in einem weiteren PCR hinterlegen und somit die Vertrauenskette erweitern, gäbe es die Möglichkeit, diesen Angriff zu erkennen.

Ein weitere Möglichkeit, die Verifikation der Treiber Signaturen zu umgehen, wird in [Rutk01] beschrieben. Dieser Angriff manipuliert die Auslagerungsdatei, um schadhaften Code in signierte Treiber einzuschleusen. Um diesen Angriff zu verhindern, ist es seit dem Release Candidate 2 von Windows Vista nicht mehr möglich, direkt auf die Datenstrukturen der Datenträger zuzugreifen. Zusätzlich kann über die Gruppenrichtlinien die Verschlüsselung der Auslagerungsdatei aktiviert, und somit deren Manipulation, verhindert werden.

6. Windows Resource Protection (WRP)

Die *Windows Resource Protection* ersetzt die *System File Protection* (SFP) von vorhergehenden Windows-Versionen. Das Ziel von WRP ist es, die Integrität der Systemdateien zu erhalten und gehört somit zu der dritten Kategorie des Anforderungskatalogs. Hierzu schützt WRP kritische Teile des Betriebssystems vor unerlaubten Veränderungen, so z. B. kritische Teile der Registrierung und Dateien, die Komponenten des Betriebssystems darstellen. Dieser Schutz wird über das Setzen

restriktiver Zugriffsrechte auf den entsprechenden Ressourcen erreicht. Somit handelt es sich nicht direkt um eine zusätzliche Sicherheitsfunktion, sondern eher um den automatischen Einsatz bereits vorhandener Funktionen.

Der einzige Prozess, der die nötigen Rechte besitzt, diese Objekte zu ändern, ist der *TrustedInstaller* Dienst. Muss eines dieser Objekte aufgrund eines Updates geändert werden, wendet sich der *Windows Update* Prozess an diesen Dienst, um schreibenden Zugriff auf das Objekt zu erhalten. Der *TrustedInstaller* installiert ausschließlich Updates, die über eine digitale Signatur von Microsoft verfügen. Dies soll verhindern, dass wichtige Teile des Betriebssystems gegen manipulierte Versionen ausgetauscht werden können.

Auch Mitglieder der Administratorengruppe unterliegen der WRP und erhalten nicht den vollen Umfang der Zugriffsrechte auf diese Ressourcen. Zum Beispiel verfügt ein Mitglied der Benutzergruppe Administratoren nicht über Schreibrechte im Verzeichnis *%SystemRoot%\Windows*. Dadurch soll verhindert werden dass der Administrator versehentlich wichtige Dateien beschädigt, bzw. dass Schadsoftware, die administrative Rechte erlangt hat, diese Dateien modifizieren kann. Es ist jedoch möglich, sich als Administrator die nötigen Rechte über Umwege zu beschaffen. Die WRP hat nämlich dem Administrator nicht das Recht zur Änderung des Besitzers eines Objekts (*TakeOwnership*) entzogen. Somit kann der Administrator sich als neuen Besitzer der Datei eintragen und erhält dadurch sämtliche Zugriffsrechte für das Objekt. Somit kann auch die im vorhergehenden Kapitel beschriebene Modifikation des *OS-Loader* und des Betriebssystemkerns durchgeführt werden.

7. PatchGuard (nur 64-Bit Versionen)

Obwohl es sich bei *PatchGuard* um keine neue Funktion handelt – sie war bereits in Windows Server 2003 SP1 und Windows XP Professional x64 Edition enthalten – lohnt sich dennoch eine nähere Betrachtung.

Neben den beschriebenen Funktionen *Secure Startup* und *Driver Signing* ist *PatchGuard* eine weitere wichtige Komponente zur Erhaltung der Systemintegrität. Im Gegensatz zur den beiden anderen Funktionen, die ihre Integritätsprüfung während des Systemstarts bzw. beim Laden von Treibern und wichtigen Betriebssystemdiensten durchführen, versucht *PatchGuard*, die Integrität des Betriebssystemkerns während der Laufzeit zu überwachen.

PatchGuard schützt das System durch regelmäßige Überprüfung von wichtigen Datenstrukturen (ca. alle 5-10 Minuten) im Speicherbereichen des ausgeführten Betriebssystems. Hierfür wird unter anderem die *System Service Descriptor Table* (SSDT), welche die Adressen der Betriebssystemfunktionen enthält, auf Änderungen überprüft. Durch eine Manipulation dieser Tabelle ist es möglich, Aufrufe von Betriebssystemfunktionen auf andere Implementierungen umzuleiten. Diese Technik ist als „kernel patching“ oder „kernel hooking“ bekannt und wird häufig von Rootkits, in Form von Systemtreibern, verwendet.

Jedoch verwenden auch Hersteller von Sicherheitssoftware (vor allem bei On-Access-Viren-Scannern und Personal-Firewalls) diese Technik. Aufgrund dieser Tatsache und weil bereits Verfahren zur Umgehung [Skape01, Skape02] des Schutzes bekannt sind, fordern einige Hersteller die Entfernung der Funktion. Als Reaktion auf die Kritik hat Microsoft ein weiteres API für Sicherheitssoftware angekündigt. Damit sollen digital signierte Applikation auch ohne Manipulationen der Datenstrukturen des Betriebssystemkerns Zugriff auf die notwendigen Funktionen erhalten.

Gegen die nächste Generation der Rootkits bietet jedoch auch *PatchGuard* keinen effektiven Schutz. Diese Rootkits verwenden Virtualisierungs-Technologie, um sich vor dem Betriebssystem zu verstecken. Durch die Verwendung von Befehlsatzerweiterungen aktueller Prozessoren von AMD oder INTEL ist es einem Rootkit sogar möglich, ein Betriebssystem zur Laufzeit in eine virtuelle Maschine zu verschieben, was seine Erkennung nahe zu unmöglich macht [Rutk01].

8. User Account Control

Eine weitere Neuerung in Windows Vista ist die User Account Control (UAC), eine weitere Maßnahme zu Erhaltung der Integrität des Betriebsystems. Unter dem Begriff UAC sind mehrere Sicherheitsfunktionen zusammengefasst, die eine Erweiterung der Zugriffskontrolle (Access Control) in Windows implementieren. Einleitend wird deshalb zunächst die bisherige Windows Zugriffskontrolle beschrieben [Govi01].

Unter Microsoft Windows (NT, 2000, XP) wird jedem Betriebssystemobjekt (Dateien, Prozesse etc.) ein *Security Descriptor* zugeordnet, der u.a. eine DACL (*Discretionary Access Control List*) und eine SACL (*System Access Control List*) enthalten kann. Ist keine DACL vorhanden, so erhält jeder Benutzer Vollzugriff auf das Objekt. Ist die DACL vorhanden aber leer, so erhält kein Benutzer Zugriff. Eine DACL besteht aus einem Header und maximal 1.820 *Access Control Entries* (ACE). Ein ACE enthält die Information, ob einem Benutzer oder einer Benutzergruppe eine bestimmte Zugriffsart erlaubt oder verweigert werden soll. DACLs werden dynamisch vererbt. Wird die DACL eines übergeordneten Verzeichnisses geändert, so hat dies Auswirkungen auf die darunterliegende Verzeichnisstruktur. Einträge innerhalb der SACL dienen nicht der Zugriffssteuerung, sie regeln, welche Zugriffsversuche auf das Objekt im Windows Ereignisprotokoll aufgezeichnet werden sollen.

Benutzer und Benutzergruppen werden eindeutig durch ihren *Security Identifier* (SID) identifiziert. Windows unterscheidet zwischen Benutzern bzw. Benutzergruppen, die sich interaktiv am System anmelden können (User Accounts), und denen, die für die Ausführung des Betriebssystems und dessen Diensten verwendet werden (System Accounts). Meldet sich ein Benutzer am System an, so wird ein für diese Sitzung gültiges *Access-Token* generiert. Dieses Token enthält die SID des Benutzers, die SIDs seiner Benutzergruppen, eine Liste der Systemrechte des Benutzers und einige Metainformationen. Startet ein Benutzer einen Prozess, so erhält dieser eine Referenz auf das *Access-Token* des Benutzers und erbt somit seine Rechte. Startet ein Prozess weitere Prozesse, so erhalten auch diese die Rechte des Benutzers. Die Zugriffskontrolle in Windows ist formell als eine Kombination aus *Role Based Access Control* (RBAC) und *Descretionary Access Control* (DAC) beschreibbar [Bish01].

User Account Protection (UAP)

Die *User Account Protection* ist auch unter den Namen *Least-Privilege User Accounts* oder *Limited User Accounts* (LUA) bekannt. Das Konzept, das hinter dieser Entwicklung steht, ist unter dem Namen *Principle Of Least Authority* (POLA) bekannt [Salt01]. Jeder Nutzer und jeder Prozess eines Systems soll nur die Rechte erhalten, die für die Ausführung unbedingt benötigt werden.

Die Hauptaufgabe der UAP ist es, die potentielle Gefahr, die durch das Arbeiten unter dem Administrator Benutzerkonto entsteht, abzuschwächen. Jeder Benutzer, der Mitglied der Benutzergruppe der Administratoren ist, erhält bei seiner Anmeldung am System zwei unterschiedliche *Access-Token*, ein *Administrator-Token* und ein *Standard-User-Token*. Das *Administrator-Token* enthält sämtliche Rechte des Benutzers, wohingegen das *Standard-User-Token* nur einen Teil dieser Rechte widerspiegelt. Startet ein Mitglied dieser Benutzergruppe nun ein Programm, wird dieses in einem Kontext ausgeführt, der nur einen kleinen Teil der Rechte des Benutzers enthält, indem diesem Prozess nur eine Referenz auf das *Standard-User-Token* übergeben wird [Micro04]. Benötigt ein Programm zur korrekten Ausführung administrative Rechte, kann der Administrator den Prozess ohne Einschränkungen ausführen. Hierbei erhält der Prozess eine Referenz auf das *Administrator-Token* und somit alle Rechte des Benutzers.

Die UAP ist standardmäßig für alle Administratoren aktiv und kann über die lokalen Sicherheitsrichtlinien oder über die Gruppenrichtlinien konfiguriert werden. Hier kann auch definiert werden, wie im Falle eines notwendigen Zugriffs auf das *Administrator-Token* verfahren werden soll. Es stehen hierbei drei Optionen zu Auswahl:

- Keine Interaktion des Administrators nötig. Es erscheint kein Dialog und der Zugriff wird automatisch genehmigt. Dies kommt einer Deaktivierung der UAP gleich und sollte nicht gewählt werden.
- Der Administrator muss den Vorgang genehmigen. Es erscheint ein Dialog mit den Aufforderung, der Vorgang zu bestätigen. Dies entspricht der Standardeinstellung, sie sollte aber nur in Verbindung mit dem *Secure Desktop*[10] verwendet werden.
- Der Administrator muss den Vorgang genehmigen. Es erscheint ein Passwort-Dialog, den der Administrator mit seinem Benutzernamen und Passwort bestätigen muss. Dies entspricht dem UAP-Dialog, der einem Standard-Benutzer beim Aufruf eines Prozesses, welcher administrative Rechte benötigt, angezeigt wird. Dies ist die sicherste der drei Optionen.

Um die Abwärtskompatibilität zu älteren Windows Versionen gewährleisten zu können, enthält auch Windows Vista das Benutzerkonto des *Built-In-Administrators*. Dieses Konto ist standardmäßig nicht aktiviert und nicht durch UAP geschützt, somit erhält jeder von diesem Benutzer gestarteten Prozess ein Access-Token, das sämtliche Rechte des Administrators enthält. Dieses Konto ist ebenfalls Mitglied in der Benutzergruppe der Administratoren, somit verfügt das Access-Token des *Built-In-Administrators* über die selben Systemrechte wie das Administrator-Token eines durch UAP geschützten Administrators[11]. Dies wird innerhalb der Dokumentation von Microsoft anders dargestellt, möglicherweise war eine zusätzliche Beschränkung der unter der UAP stehenden Konten geplant, wurde aber für die finale Version von Vista wieder verworfen.

Das mit der UAP eingeführte Konzept des eingeschränkten Administrators ist auf jeden Fall ein begrüßenswerter Ansatz, der über das Potential verfügt Schadsoftware die administrative Rechte benötigt einzuschränken Ob diese Schutzfunktionen greifen hängt jedoch ganz davon ab, ob die neuen UAP-Dialoge von den Benutzern verstanden und entsprechend genutzt werden. Ein versehentlich bestätigter UAP-Dialog eröffnet auch unter Vista ausgeführter Schadsoftware Angriffsvektoren auf das System. Einen nicht unwesentlichen Einfluss auf den Erfolg der UAP haben die Software Hersteller, die ihre Applikationen nach Möglichkeit an die Rechte des Standard-User anpassen sollten, um somit unnötige UAP-Dialoge zu vermeiden. Warum jedoch der bei der Installation angelegte Benutzer nach wie vor Mitglied der Administratoren Benutzergruppe ist und somit erneut die Chance vergeben wurde, dem Benutzer zur Verwendung eines Standard-User Kontos zu motiviert, ist nicht ersichtlich.

Zusätzlich sei erwähnt das auch diese Implementierung streng genommen keine Umsetzung des POLA-Konzeptes darstellt. So hat auch eine mit dem *Standard-User-Token* arbeitende Applikation in der Regel mehr Rechte als sie benötigt. Ein Beispiel hierfür ist der Windows Taschenrechner, er verfügt durch das *Standard-User-Token* über das Recht zum Herunterfahren des Betriebsystems. Eine vollständige Umsetzung des Konzeptes, wie es z. B. von SeLinux[12] angestrebt wird, ist jedoch auch nur schwer auf einem Desktop-Betriebssystem umsetzbar.

Mandatory Integrity Control (MIC)

In Windows Vista besitzt jedes Objekt (z. B. eine Datei oder ein Eintrag in der Registrierung) eine Integritätsstufe (*integrity level*). Prozesse besitzen ebenfalls eine Integritätsstufe und jeder Kindprozess erbt diese von seinem Elternprozess. Hierfür wurden neue Einträge im *Security*

[10] Eine weitere UAC-Funktion, beschrieben später im Dokument.
[11] Zugriffsrechte eines Benutzers können mittels *whoami /all* ausgelesen werden.
[12] http://www.nsa.gov/selinux/

Descriptor von Objekten und den *Access-Tokens* der Benutzer bzw. deren Prozesse definiert. Die folgende Tabelle listet die möglichen Integritätsstufen[13]:

Integritätsstufe	Systemrechte (vereinfacht)
High (Administrative)	Schreibzugriff auf das Programme Verzeichnis und auf schützenswerte Bereiche der Registrierung wie z.B. *HKEY_LOCAL_MACHINE*
Medium (User)	Erstellen und Ändern von Dateien im Verzeichnis Dokumente des aktuellen Benutzers und an benutzerspezifischen Bereichen der Registrierung (z.B. *HKEY_CURRENT_USER*)
Low (Untrusted)	Ausschließlich Schreibzugriff auf Bereiche mit Integritätsstufe „low“ (*Temporäre Internet Dateien\Low* und *HKEY_CURRENT_USER\Software\LowRegistry*)

Tabelle 1: Definierte Integritätsstufen

So enthält z. B. das Administator-Token die Integritätsstufe *high* und somit schreibenden Zugriff auf kritische Teile der Registrierung. Da der Administrator bei aktiver UAP standardmäßig nur mit einem Standard-User-Token arbeitet, erhält ein vom ihm ausgeführter Prozess nur die Integritätsstufe *medium*. Der Internet Explorer 7 (*iexplore.exe*) wurde zusätzlich eingeschränkt, wird er gestartet erhält er nur die Stufe *low* und kann somit nur auf einen sehr eingeschränkten Bereich schreibend zugreifen [Micro03]. Lädt ein Benutzer eine Datei aus dem Internet, welche er in seinem Benutzer Verzeichnis speichern möchte, wird ein Hilfsprozess (*ieuser.exe*) gestartet, der über die selbe Integritätsstufe wie das Benutzer Verzeichnis verfügt (*medium*). Durch die Einschränkung des Internet Explorer Prozesses soll verhindert werden, dass durch Schwachstellen in kritischen Teilen des Browser (z. B. Rendering-Engine, Javascript-Interpreter) schadhafte Dateien ohne Benutzerinteraktion (*Drive-By-Downloads*) auf den Rechner kopiert werden können.

Da MIC für schreibende Zugriffe eine zusätzliche Zugriffskontrolle zu der bisherigen Windows Zugriffskontrolle (*Access Control Lists* - ACLs) darstellt, müssen nun beide Überprüfungen ein positives Resultat liefern um, schreibenden Zugriff auf das Objekt zu erhalten. D. h. der Benutzer bzw. der Prozess muss über einen entsprechenden Eintrag in der ACL des Objektes verfügen (Schreibrecht) , und er benötigt mindestens die selbe Integritätsstufe.

Um sicher zu stellen, dass ein Prozess zur Laufzeit seine Rechte nicht ausweiten kann (privilege escalation attacks), sind bestimmte Systemaufrufe von einem Prozess mit niedriger Integritätsstufe auf einen Prozess mit höherer Integritätsstufe deaktiviert.

Die *Mandatory Integrity Control* ist eine Implementierung des *Biba-Sicherheitsmodels* [Bish01] und dient dem Schutz vor unautorisierter Änderung von Informationen (Datenintegrität). Jedoch hat Microsoft darauf verzichtet, zusätzlich Vertraulichkeitsstufen einzuführen. Dies entspräche dem Bell-*LaPadula-Sicherheitsmodels* [Bish01], welches die Umkehrung des *Biba-Models* darstellt und die Geheimhaltung von Informationen (lesende Zugriffe) regelt. Da in dieser Untersuchung der Fokus auf der Erhaltung der Systemintegrität und nicht auf dem Datenschutz liegt, ist dieser Umstand zu vernachlässigen.

Secure Desktop/Trusted Path

Zum Schutz des Windows-Dialogs bei einer Erhöhung der Benutzerrechte und während der Authentifizierung gegenüber anderen Diensten wird ein zweiter Desktop in einer separaten Benutzersitzung ausgeführt (*Session-0*). Dies soll verhindern, dass andere Applikationen auf das Fenster des Dialog zugreifen können, und dadurch die erforderliche Bestätigung durch den Benutzer

[13] Beachte: MIC überwacht ausschließlich Schreibzugriffe

selbst erbringen. Die Verwendung des Secure Desktop für Authentisierungsdialoge kann über Gruppenrichtlinien konfiguriert werden und ist standardmäßig aktiviert. Als zusätzlichen Schutz bietet Windows Vista die Möglichkeit, alle UAC-Dialoge mit einem vorgeschaltenen Strg+Alt+Entf-Dialog zu sichern. Diese Option ist ebenfalls nur über die Gruppenrichtlinien zu erreichen. Erst nach dem Betätigen dieser Tastenkombination wechselt Windows Vista zum Secure Desktop. Microsoft spricht hierbei vom Aufbau eines gesicherten Kommunikationspfades (*Trusted Path*) und verwendet die Tastenkombination als "*Secure Attention Sequence*". Diese Funktion entspricht der mit Windows NT eingeführten Sicherung für den Windows-Anmelde-Dialog. Ist dieser Schutz aktiviert, wird automatisch der Secure Desktop verwendet, unabhängig von der entsprechenden Option in den Gruppenrichtlinien.

Der Mehrwert an Sicherheit resultiert aus der Tatsache, dass die Tastenkombination Strg+Alt+Entf einen Hardwareinterrupt auslöst, welcher nur vom Betriebssystem selbst interpretiert werden kann und somit verhindert, dass Software die außerhalb des Betriebsystemkerns läuft, dieses Signal empfangen bzw. unterbinden kann. Das Simulieren des Signals mittels Software ist jedoch möglich (z. B. Windows Remote Desktop). Der Wechsel zum UAC-Dialog nach Drücken der Tastenkombination entspricht somit einer Authentifizierung des Betriebssystems gegenüber dem Benutzers vor dem Bildschirm und nicht wie oft angenommen der Authentifizierung des Benutzers gegenüber dem Betriebssystem.

Der *Secure Desktop* liefert in der Standardkonfiguration leider nur einen Teil seines Schutzpotentials. So verhindert er zwar, dass der Dialog von Prozessen außerhalb der *Session-0* angesprochen werden kann und somit, dass schadhafte Programme diesen Dialog selbst bestätigen können. Dies ist aber ohnehin nur möglich wenn der aktive Benutzer Mitglied der Gruppe "Administratoren" ist und eine zusätzliche Eingabe des Passwortes nicht mittels Gruppenrichtlinie aktiviert wurde. Ebenso wird eine Brute-Force- bzw. Wörterbuchattacke auf den UAP-Dialog verhindert, welche aber ohnehin durch einen Angriff auf die Passwortdatei effektiver durchgeführt werden könnte. Er verhindert jedoch nicht, dass ein schadhaftes Programm einen gefälschten UAP-Dialog präsentiert und den Benutzer damit zur Eingabe seines Passwortes verleitet. Eine zusätzliche Aktivierung des Strg+Alt+Entf Schutzes führt jedoch zur Verbesserung dieser Situation. Zwar ist es immer noch möglich, den Strg+Alt+Entf-Dialog zu fälschen, jedoch ist es wie erwähnt nicht möglich, das bei dieser Tastenkombination ausgelöste Signal ohne Manipulation des Betriebssystems abzufangen. Wird dem Benutzer bei aktiviertem Strg+Alt+Entf Schutz ein gefälschter Strg+Alt+Entf-Dialog präsentiert und er kommt dieser Aufforderung nach, wird der Windows Vista Anmeldebildschirm aktiv und entlarvt den Angriff, da nicht der erwartete UAP-Dialog erscheint.

Da Microsoft jedoch die aus Sicht des Autors unglückliche Entscheidung getroffen hat, diesem Dialog einen weiteren Windows-Dialog vorzuschalten[14], wird der Benutzer mit zwei zusätzliche Dialogen konfrontiert bevor der den eigentlichen UAP-Dialog erreicht, was der Benutzerakzeptanz nicht unbedingt förderlich ist. Deshalb favorisiert der Autor den in [Dham01, Yee01] beschriebenen Ansatz zur Authentifizierung von Betriebssystem-Dialogen. Die Idee dieses Konzeptes ist es, die Fenster des Betriebssystems von Fenstern normaler Applikation grafisch abzuheben, zum Beispiel durch einen zusätzlichen Farbrand oder durch ein bestimmtes Muster. Dieses Erkennungsmerkmal muss für jede Sitzung unterschiedlich sein und darf nur dem Betriebssystem und dem Benutzer bekannt sein, es bildet somit das gemeinsame geheime Erkennungszeichen. Ein denkbares Szenario wäre die Generierung und Anzeige des für diese Sitzung gültigen "Code" während der Benutzeranmeldung und anschließendes Hinterlegen im Speicherbereich des Betriebssystems. Verfügt nun ein sicherheitskritischer Windows-Dialog über keine oder eine ungültige Markierung, kann er als gefälschter Dialog identifiziert werden. Es sei zusätzlich erwähnt, dass weder die Kombination aus

[14] Dieser Dialog enthält lediglich den Hinweis, dass aus Sicherheitsgründen die Betätigung von Strg+Alt+Entf notwendig ist.

Secure Desktop und Strg+Alt+Entf-Dialog noch die eben beschriebene Maßnahme einen wirkungsvollen Schutz vor Key-Loggern im Betriebssystemkern oder dem Abhören der Kommunikation zwischen Tastatur und Rechner darstellt.

Ein weiterer Kritikpunkt ist, dass Dialoge, die nicht bei der Anmeldung am lokalen System oder für die Erhöhung der Systemrechte verwendet werden, dennoch aber vertrauliche Daten enthalten können, nicht mit der UAP gekoppelt sind und somit nicht zu einer Verwendung des *Secure Desktop* führen. Beispiele hierfür sind der Dialog beim Verbinden von Netzwerklaufwerken und die Anzeige des TPM-Eigentümerpassworts während der Ausführung des Assistenten zur Einrichtung des TPM.

UI Privilege Isolation (UIPI)

In bisherigen Windows-Versionen war es für einen Prozess möglich, Nachrichten an Fenster eines anderen Prozesses zu schicken (SendMessage und PostMessage APIs). Dies ermöglicht unter bestimmten Umständen die Ausführung von beliebigem Code im Kontext eines Prozesses mit höheren Rechten (Shatter Attack [Moore01]).

Deshalb ist es zum einen in Windows Vista nicht mehr möglich, dass ein System-Prozess (Windows Dienst) über eine grafische Oberfläche (GUI) verfügt, zum anderen wird verhindert, dass ein Prozess mit geringerer Integritätsstufe an einen Prozess mit höherer Integritätsstufe derartige Nachrichten senden kann. Hierfür wird zusätzlich zu den bereits beschriebenen Integritätsstufen der UAC eine weitere Integritätsstufe eingeführt (*UIAccess-Integritätsebene*). Benötigt eine Applikation auf Grund ihrer Funktionsweise diese Integritätsstufe, ein Beispiel hierfür ist eine Bildschirmtastatur, kann sie dies in ihrem Applikations-Manifest angeben (*UIAccess=True*) und erhält nach der Bestätigung durch den Administrator die nötigen Rechte. Als zusätzlichen Schutz kann mittels Gruppenrichtlinie die Vergabe dieser Integritätsstufe auf Applikationen beschränkt werden, deren Dateien im Windows Verzeichnis (*%SystemRoot%*; *%SystemRoot%\system32*) abgelegt sind und über eine gültige digitale Signatur verfügen.

9. Zusammenfassung und Schlussfolgerung

Microsoft hat mit Windows Vista zahlreiche neue Sicherheitsfunktionen eingeführt, die alle ihren Teil zum Aufbau bzw. zur Erhaltung der Systemintegrität beitragen. Neben vielen kleinen Detailverbesserungen sind hierbei vor allem die beschriebenen Funktionen *BitLocker*, *Driver Signing*, *PatchGuard* und die *User Account Control* hervorzuheben. *BitLocker* bietet die Möglichkeit die gesamte Systempartition zu verschlüsseln und den verwendeten Schlüssel durch den Einsatz eines TPM an einen bestimmten Zustand des Systems zu binden. Die *Driver Signing* Funktion stellt sicher, dass nur digital Signierte Treiber vom Betriebssystem verwendet werden. Und *PatchGuard* überprüft kritische Datenstrukturen des Betriebssystem um eine Manipulation zur Laufzeit zu erkennen. Die UAP versucht durch Erweiterung der bereits in Windows XP vorhandenen Zugriffskontrollemechanismen die Angriffsfläche des Betriebssystem zu verkleinern.

So ist unter dem Aspekt der Betriebssystem-Sicherheit eine deutliche Verbesserung zu erkennen und viele der Schwächen von Windows XP wurden adressiert. Obwohl bereits für einige Funktionen Maßnahmen zur Umgehung bekannt geworden sind, und auch mit der Aufdeckung weiteren Schwächen zu rechnen ist, ist es Microsoft durchaus gelungen die Angriffsfläche insgesamt deutlich zu verkleinert. Somit ist auch der Slogan von Microsoft: „Vista, das sicherste Windows aller Zeiten“ zutreffend. Nach mindestens vierjähriger Entwicklungszeit wäre ein anderes Ergebnis auch sehr enttäuschend gewesen.

Allerdings ist ein Teil der hier vorgestellten Sicherheitsfunktionen der Produktpolitik von Microsoft zum Opfer gefallen. So sind sowohl die Überprüfung von Treibersignaturen wie auch die *PatchGuard* Komponente nur in den 64 Bit Versionen enthalten. Im Falle des *Driver Signing* ist dies laut Microsoft ein Zugeständnis an die Hardwarehersteller. Eine andere Lösung wäre gewesen diese Entscheidung dem Benutzer zu überlassen und eine optionale Aktivierung des Driver Signing auch in

den 32 Bit Versionen anzubieten. Auch von den Vorteilen der *BitLocker* Festplattenverschlüsselung und somit dem Schutz vor Offline-Angriffen, werden wohl die wenigsten der Vista Benutzer profitieren, sie ist nämlich nur in der Enterprise und der Ultimate Version enthalten.

Eine Bewertung von Windows Vista anhand des definierten Anforderungskatalogs für ein *Trusted Computing System* fällt schwer, so sind zwar aus jeder Kategorie des Anforderungskatalogs Funktionen vorhanden, ein durchgängiges Konzept ist jedoch nicht zu erkennen. So ist die *BitLocker* Festplattenverschlüsselung die einzige Komponente die von der *Trusted Computing Platform* gebraucht macht, verwendet diese aber in erster Linie für die Verschlüsselung der Systempartition und verzichtet auf die Fortsetzung der Vertrauenskette bis zum vollständigen Systemstart. Genau dies wäre jedoch einer der Grundlagen für den Aufbau eines *Trusted Computing Systems* gewesen. Microsoft vertraut in diesem Bereich alleine auf die Überprüfung der digitalen Signaturen wichtiger Systemkomponenten. Auch wenn dies ein wirkungsvolles Konzept zum Aufbau und zum Erhalt der Systemintegrität darstellt, ist es unverständlich warum dieses Konzept nicht zusätzlich durch die Funktionen des TPM gesichert wurde.

Mit Palladium, das im Jahre 2003 in *Next Generation Secure Conputing Base* (NGSCB) ungetauft wurde, hatte Microsoft sich das ehrgeizige Ziel gesetzt ein *Trusted Computing System* zu entwickeln. So sollte u. a. das Betriebssystem in einen vertrauenswürdigen Bereich für sicherheitskritische Anwendungen und einen Bereich für vertrauensunwürdige Software aufgeteilt werden. Auch der Einsatz des TPM zur Sicherung der Systemintegrität war geplant[15]. Nach dem Microsoft mehrmals die Anforderung an die NGSCB in Windows Vista nach unten korrigiert hatte, war in Jahre 2006 gar nichts mehr zu diesem Thema zu hören. Einflüsse der NGSCB auf die finale Version von Vista sind nur in der Form der *BitLocker* Festplattenverschlüsselung und der für den Schutz der Systemdienste verwendeten *Session-0*, zu erkennen.

Somit ist das Ergebnis der Untersuchung, dass selbst die 64 Bit Versionen mit aktivierter *BitLocker* Festplattenverschlüsselung nicht den Anforderungen an ein *Trusted Computing System* gerecht werden. Es bleibt abzuwarten ob Microsoft an den Zielen der NGSCB festhält und diese in einer kommenden Windows Version umsetzt.

10. Literaturhinweise

[Bish01] M. Bishop (2005). “Introduction to Computer Security“

[Carr01] B. Carrier, J. Grand (Februar 2004). “A Hardware-Based Memory Acquisition Procedure for Digital Investigation”

[Cono01] M. Conover (März, 2006). “Analysis of the Windows Vista Security Model,”

[Cono02] M. Conover (August, 2006). “Assessment of Windows Vista Kernel-Mode Security,”

[Dham01] R. Dhamija, J.D. Tygar. “The Battle Against Phishing – Dynamic Security Skins”

[Govi01] S. Govindavajhala, A. Appel (Januar, 2006). “Windows Access Control Demystified”

[Petro01] N. Petroni, et al. (August, 2004). “Copilot – a Coprocessor-based Kernel Runtime Integrity Monitor”

[Rutk01] J. Rutkowska (July, 2006). “Subverting Vista Kernel For Fun And Profit”. SyScan 2006, Singapore

[Salt01] J. Salzer, M. Schroeder (April, 1975). “The Protection of Information in Computer Systems“

[Skape01] Skape, Skywing (Dezember, 2005). “Bypassing PatchGuard on Windows x64”

[Skape02] Skape, Skywing (Dezember 2006). ”Subverting PatchGuard Version 2”

[Smith01] S. Smith (2005). “Trusted Computing Platforms – Design and Applications”

[15] Mehr Informationen zur NGSCB liefert http://www.datenreise.de/de/tc/ngscb.php

[Tcg01] Trusted Computing Group (Juli, 2005). “TCG PC Client Specific Implementation for conventional BIOS”
[Tcg02] Trusted Computing Group (Januar, 2006). “TCG Software Stack (TSS) Specification Version 1.2”
[Tcg03] Trusted Computing Group (März 2006). “TPM Main – Part 1 Design Principles”
[Micro01] Microsoft (August 2006). “AES-CBC + Elephant diffussor – A Disk Encryption Algorithm for Windows Vista”
[Micro02] A. Ben-Menahem, A. Tucker. (2005). “Windows Vista and ‘Longhorn’ Server: Understanding, Enhancing and Extending Security End-to-end,”
[Micro03] Microsoft (Januar, 2006). “Understanding and Working in Protected Mode Internet Explorer”
[Micro04] Microsoft. “Restricted Tokens”. MSDN [Online]. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secauthz/security/restricted_tokens.asp
[Micro05] Microsoft. (Januar, 2006). “Digital Signatures for Kernel Modules on x64-based Systems Running Windows Vista”
[Micro06] Microsoft. (April, 2005). “Secure Startup - Full Volume Encryption: Technical Overview,”
[Micro07] Microsoft. (Mai, 2006). “BitLocker Drive Encyption: Technical Overview”
[Micro08] Microsoft. (Mai, 2006). “Windows Vista Beta 2 – Trusted Platform Module Services Step-by-Step Guide”
[Micro09] Microsoft. (April, 2005). “Trusted Platform Module Services in Windows Longhorn”
[Micro10] Microsoft. (July, 2006). “Step-by-Step Guide to Device Driver Signing,”
[Moore01] B. Moore. (Oktober, 2003). “Shattering by Example”
[Yee01] K. Yee. “User Interaction Design for Secure Systems”