ERNW Newsletter 17 / Juli 2007

Liebe Partner, liebe Kollegen,

willkommen zur 17. Ausgabe des ERNW-Newsletters mit dem Thema:

Mandatory Integrity Control

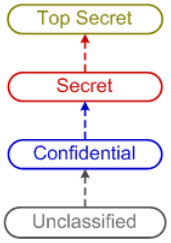
Version 1.0 vom 2. Juli 2007

von: Friedwart Kuhn (fkuhn@ernw.de)

In Windows Vista hat Microsoft mit Mandatory Integrity Control (MIC) erstmals ein Zugriffskontrollmodell implementiert, dass sich unter dem Stichwort „Multilevel Security“ führen lässt und die Integrität des Betriebssystems selbst maßgeblich sichern soll. Der Artikel beleuchtet, wie MIC funktioniert und ob MIC hält, was es verspricht.

1 Die Wurzeln von MIC: Multilevel Security und das Bell-LaPadula-Modell

Da das Windows Vista-Integritätsmodell seine Wurzeln in der sogenannten Multilevel Security hat, soll ein kurzer Rückblick zu den Anfängen der Multilevel Security und dem ersten von ihr realisierten Modell gemacht werden.
Der Begriff *Multilevel Security* (kurz MLS) leitet sich von den verschiedenen Leveln von Vertrauenswürdigkeit und Sensitivität ab, in die – zunächst im militärischen Bereich der US Air Force in den 1970igern – sowohl Informationen als auch Benutzer, die auf Informationen zugreifen, eingeteilt (klassifiziert) wurden. Die auch noch heute gängigen *Classification Level* sind: *unclassified* (nicht klassifziert), *confidential* (vertraulich), *secret* (geheim) und *top secret* (streng geheim). Bevor ein Benutzer auf klassifizierte Informationen zugreifen darf, muss sichergestellt werden, dass dieser Benutzer einen Level an Vertrauenswürdigkeit besitzt, der ihn zum Zugriff auf die klassifizierten Informationen berechtigt. Der Level an Vertrauenswürdigkeit wird in der einschlägigen Literatur als *Clearance Level* bezeichnet und setzt im militärischen wie organisationsweiten Kontext eine formale Überprüfung der Person voraus. Hat ein Benutzer z. B. einen Clearance Level der Stufe *confidential*, dann ist er berechtigt auf Dokumente zuzugreifen, die als *confidential* klassifiziert wurden, nicht jedoch auf Dokumente, die als *secret* oder *top secret* klassifiziert wurden.
Die nachfolgende Abbildung stellt die Regel dar, nach der Datenfluss statt finden darf: Datenfluss ist von unten nach oben, nicht aber in die entgegengestzte Richtung erlaubt.



(Abbildung 1)

Werden *Clearance Level* und *Classification Level* allgemein als *Security Level* bezeichnet, dann lässt sich die Regel ausdrücken als:

Ein Subjekt darf genau dann lesend auf ein Objekt zugreifen, wenn der Security Level des Subjekts höher oder gleich dem Security Level des Objekts ist.

Diese Regel wird in der einschlägigen Literatur auch als *No read up policy* bezeichnet.
Um einen vertrauenswürdigen Informationsfluss innerhalb eines Systems zu gewährleisten, reicht diese Regel (Policy) jedoch nicht aus: man stelle sich beispielsweise vor, dass das Textverabeitungsprogramm eines Benutzers, der über den Security Level *secret* verfügt, von einem Trojaner infiziert wurde, der immer dann, wenn der Benutzer eine Datei öffnet, die als *secret* klassifiziert wurde, eine Kopie dieser Datei in einem *unclassified*-Ordner erstellt. Damit wäre die geheime Datei für alle lesbar! Um einer solchen Manipulation zuvorzukommen, ist offensichtlich noch eine weitere Regel notwendig: der schreibende Zugriff ‚von oben nach unten' soll verhindert werden:

Ein Subjekt kann genau dann schreibend auf ein Objekt zugreifen, wenn der Security Level des Objekts gleich oder höher als der des Subjekts ist.

Diese Regel wird in der Fachliteratur auch als *No write down policy* bezeichnet.
Das hierarchische Klassifizierungsschema zusammen mit den beiden Regeln wird als Bell-LaPadula-Modell bezeichnet und begründete den Beginn der Multilevel Security. Es wurde 1973 von David Bell und Len LaPadula im Auftrag der US Air Force entwickelt und soll die Vertraulichkeit von Informationsflüssen in einem System garantieren.

2 Das Ken Biba-Modell und Mandatory Integrity Control

Einer der großen Kritikpunkte am Bell-LaPadula-Modell war der fehlende Schutz der Integrität der Daten, auf die zugegriffen wurde. Durch die fehlende Überprüfung der Integrität ist etwa das folgende Szenario möglich: Ein Virus kann den gemäß Bell-LaPadula möglichen Informationsfluss von unten nach oben ausnutzt, um als nicht klassifizierte Datenstruktur über einen Benutzer mit einem *top secret-Clearance Level* (der die nicht klassifizierte virale Datenstruktur lesen darf – es gibt keine Policy, die dies verbietet) streng geheime Dokumente zu manipulieren. Dies verhindert das von Ken Biba 1975 ebenfalls im Auftrag der US Air Force nach seinem Namen entwickelte Modell. Es benutzt dasselbe hierarchische Klassifizierungsschema mit denselben Security Leveln, aber es definiert andere, dem Bell-LaPadula-Modell gewissermaßen entgegengesetzte Regeln:

Subjekte dürfen nur dann lesend auf Objekte zugreifen, wenn der Security Level des Objekts höher oder gleich dem Securiy Level des Subjekts ist. D. h. es darf nicht ‚nach unten' gelesen werden.

Subjekte dürfen nur dann schreibend auf Objekte zugreifen, wenn der Security Level des Subjekts höher oder gleich dem Securiy Level des Objekts ist. D. h. es darf nicht ‚nach oben' geschrieben werden.

Die erste Regel wird auch als die *No read down-* und die zweite als die *No write up-Policy* bezeichnet. Damit soll die Integrität des Informationsflusses in einem System sichergestellt werden, insbesondere ist ein solches System geeignet, Trojaner zu verhindern. Die Vertraulichkeit wird von dem Biba-Modell allerdings nicht berücksichtigt und muss über andere Mechanismen implementiert werden.
Windows Vista implementiert nun mit MIC ein reduziertes Biba-Modell: Bei Windows Vista ist per Default nur die *No write up-Policy* implementiert. Es soll damit verhindert werden, dass Prozesse eines niedrigen Security Levels – z. B. über den Internet Explorer heruntergeladener Schadcode – Systemdateien verändern kann.

3 Mandatory Integrity Control – Terminologie und Implementierung

Damit sind die notwendigen Voraussetzungen geschaffen, um sich der Implementierung von Mandatory Integrity Control in Windows Vista zu nähern. MIC beruht – wie erläutert – auf dem Biba-Modell, Microsoft verwendet jedoch gegenüber der in der Multilevel Security verwendeten Terminologie eine abweichende, eigene Terminologie: Objekte im Sinne des Objektbegriffs von MLS sind bei Windows *Securable Objects*, dies umfasst alle Elemente (Objekte im allgemeinen Sinn), auf die zugegriffen werden kann. Subjekte im Sinne des Subjektbegriffs von MLS sind bei Windows Vista Benutzer, Dienste und Prozesse. Alles, was mit einer DACL (*Discretionary Access Control List*) versehen werden kann (siehe Abbildung 2), wird nun von Microsoft als *Securable Object*, zu deutsch als *sicherheitsfähiges Objekt* bezeichnet.

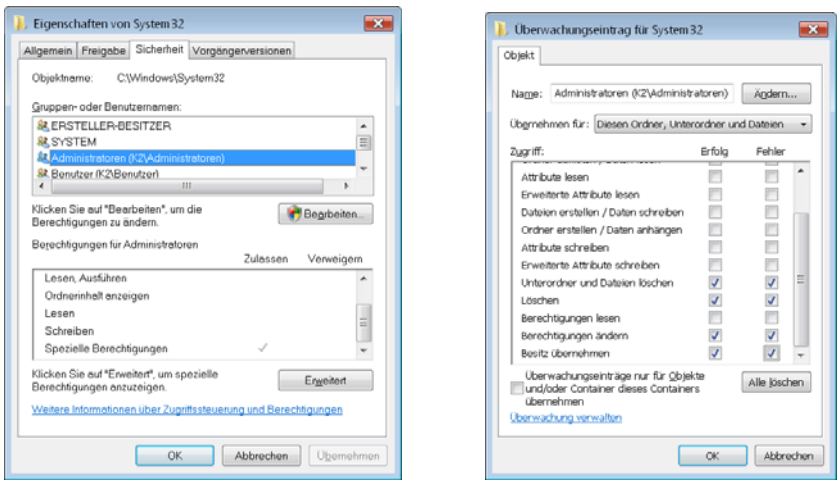


Abbildung 2: DACL (links) und SACL (rechts) für den Ordner %Systemroot%\System32\

Windows kennt viele *Securable Objects*, darunter fallen z. B. Dateien und Ordner, Registry-Keys, Windows-Dienste, Drucker, Prozesse, Benutzer-, Gruppen- und Computerkonten, um nur die wichtigsten zu nennen. *Securable Object* ist demnach die Obermenge von Subjekten und Objekten und sagt nicht mehr und nicht weniger aus, als dass es eine DACL zu dem *Securable Object* gibt. Die Subjekt- oder Objekteigenschaft ist dagegen davon abhängig, wer das zugreifende Subjekt ist und wer das Objekt, auf das zugegriffen wird: So kann ein Dienst Subjekt sein, wenn dieser Dienst z. B. auf einen Drucker zugreift, er kann aber auch Objekt sein, wenn ein Prozess seinerseits auf den Dienst zugreift.
Subjekte und Objekte, kurz jedes *Securable Object* wird nun vom Betriebssystem, so wie es MLS vorsieht, klassifiziert: Bei Windows Vista heißt das ‚mit einem verbindlichen Label versehen' und diesem Label einen Wert zuordnen – Subjekte und Objekte werden gelabelt. Da dieser Label verbindlich und systemweit gültig ist, wird er im englisch-sprachigen Vista mit *mandatory label* bezeichnet. Für das deutsch-sprachige Vista hat sich Microsoft für eine recht umständliche Übersetzung entschieden: *verbindliche Beschriftung*.
Der Wert des Labels ist ein Maß für die Vertrauenswürdigkeit des Subjekts oder des Objekts. Werte für die verbindlichen Labels werden bei Vista als *Integrity Level* bezeichnet und Vista kennt davon fünf, bzw. sechs. Die deutsche Übersetzung für *Integrity Level* von Microsoft ist *Verbindlichkeitsstufe*.
Objekte speichern ihren Integrity Level in der SACL, und Subjekte erhalten einen zusätzlichen SID-Wert, der ihren Integrity Level angibt, in ihrem Token (Mehr zu den Speicherorten von Integrity Leveln weiter unten). Damit kann der in Windows verwendete *Reference Monitor* auch weiterhin den Zugriff kontrollieren: Will ein Subjekt (z. B. ein Benutzer) schreibend auf ein Objekt (z. B. eine Datei) zugreifen, dann prüft der *Reference Monitor* im Token des Benutzers dessen Integritätslevel und vergleicht diesen Wert mit dem Integritätslevel des Objekts, der in dessen SACL gespeichert ist. Ist der Integritätslevel des Subjekts höher oder gleich dem Integritätslevel des Objekts, gewährt der *Reference Monitor* gemäß der *No write up-Policy* den Schreibzugriff. MIC wurde also so implementiert, dass das seit Windows NT bekannte Zugriffskontrollmodell, bei dem eine Kernelkomponente – der *Reference Monitor* – Werte im Token mit Werten in einer ACL vergleicht, beibehalten werden konnte.

4 DIE INTEGRITY LEVEL IN WINDOWS VISTA

Anders als die klassischen MLS-Systeme, die vier Klassifizierungsstufen (*Security Level*) kennen, kennt Vista fünf, bzw. sechs stufen. Es sind die Folgenden (mit den deutschen Übersetzungen von Microsoft in Klammern):

Untrusted (Nicht vertrauenswürdig): Gemäß Microsoftscher Dokumentation erhält jeder Prozess, der auf einer anonymen Anmeldung beruht, diesen niedrigsten Integrity Level. Per Default läuft auf Vista kein Prozess mit diesem Level. Wird der mit Vista ausgelieferte IIS 7.0 installiert und für anonyme Zugriffe auf Webseiten konfiguriert, dann sollen die Prozesse der anonym zugreifenden Benutzer mit diesem Integrity Level laufen.

Low (niedrig): Bei der Installation von Vista werden einige Ordner erstellt, deren Integrity Level den Wert *low* besitzt. Dazu gehören der Ordner für temporäre Internetdateien, der Ordner %Systemroot%\Users\AppData\LocalLow sowie einige Registry-Keys unterhalb von HKCU. Soweit zu den Objekten, mit niedrigem Integritätslevel. Was die Subjekte betrifft, so gibt es per Default nur einen Prozess, der mit dieser Integritätsstufe läuft, und das ist der Internet Explorer im sogenannten *protected mode* (geschützter Modus). Andere Prozesse laufen nur dann mit dieser Integritätsstufe, wenn sie an Orten gespeichert wurden, die einen niedrigen Integrity Level besitzen (und wenn die Standard Policy, nach der Integritätslevel von Containern auf Container-Inhalte vererbt werden, nicht verändert wurde).

Medium (mittel): Standard User und die von ihnen gestarteten Anwendungen und Prozesse laufen mit dieser Integritätsstufe. Dies ist praktisch alles, was auf dem Desktop eines Standard-Benutzers läuft: Office- und Mailprogramme, Winzip, Acrobat Reader u. v. m. Wenn man berücksichtigt, das Administratoren per Default durch UAC geschützt sind und im Standard User-Rechtekontext laufen, dann bedeutet dies, dass alles, was von beliebigen Benutzern gestartet wird und keine Privilegienanhebung über die User Account Control (UAC) erfahren hat, mit mittlerem Integrity Level läuft. Dies ist eine deutliche Verbesserung des Privilegienmanagements im Vergleich zu Windows XP und zu Windows Server 2003.

High (hoch): Administratoren, die etwa MMC-Konsolen mit administrativen Berechtigungen ausführen (die also einer Privilegienanhebung der jeweiligen MMC-Konsole zugestimmt haben), laufen mit dieser Integritätsstufe.

System (System): Benutzer können diesen Integrity Level nicht erreichen – jedenfalls ist bisher kein Tool bekannt, dass die Anhebung eines von einem Benutzer initiierten Prozesses auf diesen Level ermöglicht. Mit dieser Integritätsstufe laufen vor allem systemnahe Dienste, der größte Teil des Kernels läuft ebenfals in dieser Stufe.

(?) Installer (??): Die ersten Spezifikationen von MIC sahen vor, dass Objekte eines bestimmten Integritätslevels nur durch Subjekte mit einem höheren (ein gleicher Integrity Level sollte nicht ausreichen) Integritätslevel verändert werden dürften. Wenn dem so wäre, dann bräuchte es eine höchste Instanz, die Systemprozesse verändern oder auch wieder deinstallieren kann. Im dem aktuellen Vista-Release können jedoch Prozesse mit mindestens gleichem Integrity Level wie der des Objekts schreibend auf dieses zugreifen. Es ist bisher nicht klar, ob dieser höchste Level nicht implementiert worden ist oder ob er nur nicht sichtbar ist. Auch Mark Minasi, der mit *Administering Windows Vista Security*[1] das bislang ausführlichste Dokument zu MIC veröffentlich hat, muss zugestehen, dass er bislang keine Anzeichen für einen Prozess finden konnte, der mit einer höheren Integritätsstufe als *System* läuft. Die diesbezügliche Dokumentation von Microsoft ist mehr als spärlich.[2]

---

[1] *Mark Minasi und Byron Hynes,* Administering Windows Vista Security. The Big Surprises*. Wiley: Indiana, 2007.*

[2] Das Anliegen, das Microsoft mit dem Integritätslevel des *Installer* erreichen wollte: nämlich vor allem den Schutz von Betriebssystemdateien vor nicht autorisierten Modifikationen, wird derzeit durch sehr restriktive NTFS-

5 SPEICHERUNG UND ANZEIGEN VON INTEGRITÄTSLEVELN

Subjekte (Benutzer und Prozesse) ‚speichern' ihren Integritätslevel als einen zusätzlichen SID-Wert im Token, das sie bei der Anmeldung erhalten. Die folgende Tabelle gibt einen Überblick über die Zuordnung von SID-Werten zu Integritätsleveln. Dabei bezeichnet die erste Spalte den Integritätslevel; die zweite Spalte, die Bezeichnung des Integritätslevels, so wie man ihn in kommandozeilenbasierten Tools sieht; die dritte Spalte den SID-Wert des Integritätslevels; die vierte den entsprechenden Hexadezimalwert; die fünfte typische Anwendungen und Prozesse, die mit der jeweiligen Integritätsstufe laufen.

Integrity Level (engl.)	**Bezeichnung des Levels durch das Betriebssystem (engl.)**	**SID-Wert**	**Hexade-zimaler Wert**	**Welche Anwendungen und Prozesse laufen**
Untrusted	Mandatory Label\Untrusted Mandatory Level	S-1-16-0	0	Anonymous
Low	Mandatory Label\Low Mandatory Level	S-1-16-4096	1000	Protected Mode Internet Explorer und alles, was vom Internet Explorer an einen Ort mit niedrigem Integritätslevel gespeichert wird
Medium	Mandatory Label\Medium Mandatory Level	S-1-16-8192	2000	Standard Benutzer und die von ihnen gestarteten Anwendungen und Prozesse
High	Mandatory Label\High Man-datory Level	S-1-16-12288	3000	Administratoren
System	Mandatory Label\System Mandatory Level	S-1-16-16384	4000	Die meisten Systemdienste

Subjekte erwerben ihren Integritätslevel über die Mitgliedschaft in einer der folgenden drei ‚Gruppen':

- Mandatory Label\Low Mandatory Level
- Mandatory Label\Medium Mandatory Level
- Mandatory Label\High Mandatory Level

Der Verständlichkeit halber liste ich auch noch die deutschen Bezeichnungen, denn diese zeigt das deutsche Vista:

- Verbindliche Beschriftung\Hohe Verbindlichkeitsstufe
- Verbindliche Beschriftung\Mittlere Verbindlichkeitsstufe
- Hohe Beschriftung\Niedrige Verbindlichkeitsstufe

Berechtigungen auf das Windows-Verzeichnis und dessen Unterverzeichnisse umgesetzt. Hier verfügen Administratoren über keine Berechtigungen zum Löschen, und auch die Berechtigungen der Besitzübernahme und des Änderns von Berechtigungen sind ihnen entzogen. Über Vollzugriff verfügt dagegen die Instanz des *TrustedInstaller*, eines neuen Systemdienstes, der für das Update von Betriebssystemkomponenten verantwortlich ist. Er scheint den Integritätslevel *Installer* im aktuellen Release von Vista zu ersetzen.

Der Befehl *whoami –groups* listet auf, in welchen Gruppen ein Benutzer Mitglied ist, und zeigt darüber die Integritätsstufe an, in der der Benutzer läuft:

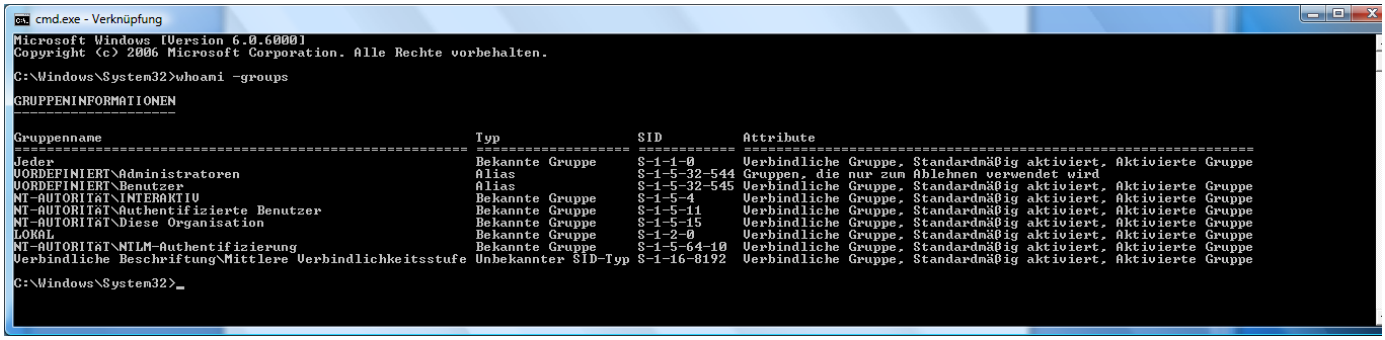


Abbildung 3: Anzeigen des Benutzer-Integritätslevels über den Befehl *whoami*.

Die Gruppenmitgliedschaft in der letzten Zeile gibt den Integrity Level des Benutzers an; man erkennt den SID-Wert 8192, der dem mittleren Integritätslevel entspricht.
Sollen die Integritätslevel sämtlicher auf dem System laufenden Prozesse angezeigt werden, so eignet sich dafür der *Process Explorer* von ehemals Sysinternals (jetzt Microsoft).
Die ‘Gruppe’, über die ein Benutzer seinen Integritätslevel erhält, sind dabei keine Gruppen im eigentlichen Sinne, d. h. Gruppen, bei denen ein Benutzer zum Mitglied gemacht werden kann. Da es sich bei MIC um ein Mandatory Access Control-Modell, also um ein systemweit gültiges und verbindliches Zugriffskontrollmodell handelt, nimmt das System eine Zuordnung vor. Wie erhalten denn aber Benutzer ihren Integritätslevel?
Immer dann, wenn ein Benutzer mindestens über eines von neun besonders gewichtigen Privilegien verfügt[3], erhält er in seinem Token den SID-Wert für die hohe Integritätsstufe. Bricht man dieses auf die beiden wichtigsten Funktionsgruppenkonten, nämlich *Benutzer* und *Administratoren* herunter, dann erhält ein Mitglied der Gruppe *Benutzer* einen mittleren Integrity Level und ein Mitglied der Gruppe *Administratoren* einen hohen.

Objekte speichern ihren Integrity Level in der *System Access Control List* (*SACL*), die bis vor Windows Vista ausschließlich für die Definition von Auditkategorien und der Art ihrer Überwachung (*Erfolg* oder *Fehlschlag*) zuständig war (siehe Abbildung 2). Leider werden die Integrity Level eines Objekts im GUI nicht sichtbar gemacht. Es gibt allerdings kommandozeilenbasierte Tools, mit denen sich die Integrity Level anzeigen lassen. Microsoft liefert mit *icacls.exe* ein Tool direkt mit Vista aus, zwei weitere Tools (*chml.exe* und *accesschk.exe*) stammen von Sysinternals und können über die Technetseiten von Microsoft heruntergeladen werden. Die Ausgabe der beiden Sysinternals-Tools sieht hinsichtlich des Integritätslevels für *notepad.exe* wie folgt aus:

[3] *Diese neun Privilegien sind:* Create a token object, Act as a part of the operation system, Back up files and directories, Restore files and directories, Load and unload device drivers, Impersonate a client after authentication, Modify an object label, Take ownership of files and other objects, Debug programms.

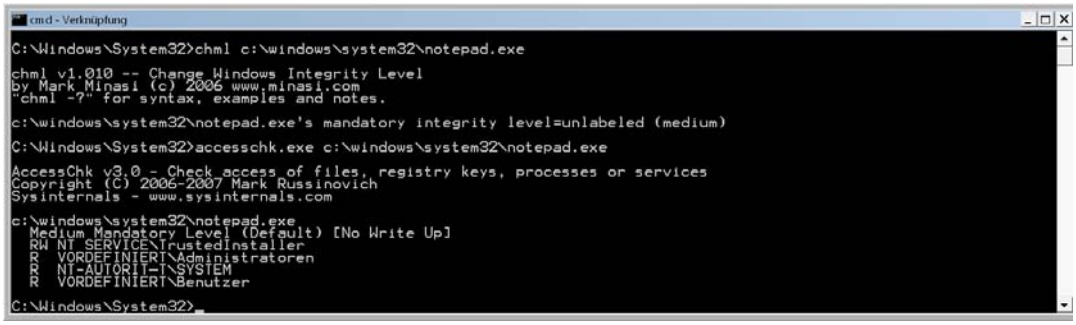


Abbildung 4: Anzeigen von Objekt-Integritätsleveln mit *chml.exe* und *accesschk.exe*

Icacls liefert eine analoge Ausgabe, zeigt den Integrity Level jedoch nur bei explizit gelabelten Objekten an. *Accesschk.exe* zeigt zusätzlich noch die implementierte Policy an, die bei Vista – in Anlehnung an das Biba-Modell – eine reine *No write up*-Policy ist.

6 DER INTERNET EXPLORER UND DER PROTECTED MODE

Eine der größten Herausforderungen der Betriebssystem- und Applikationssicherheit ist es, den Schaden, der durch Einschleppen von Schadcode aus dem Internet angerichtet werden kann, so gering wie möglich zu halten. Ein wichtiger Schritt in diese Richtung ist der *Protected Mode* (dt. *Geschützter Modus*), in dem der Internet Explorer 7 (IE7) standardmäßig läuft. Was verbirgt sich dahinter? Dahinter verbirgt sich ein bestimmter Integrity Level, mit dem der IE7 per Default läuft: nämlich der Integrity Level *Low*. Was bedeutet das in der Theorie? Microsoft vermittelt, dass damit ‚alles, was aus dem Internet' kommt, mit niedrigem Integrity Level läuft. Doch dem ist in der Praxis nicht so: zwar läuft der Prozess *iexplore.exe* immer mit niedrigem Integrity Level, doch laufen vom IE7 downgeloadete und anschließend gestartete Programme nur dann auch mit niedrigem Integrity Level, wenn sie an Orten gespeichert wurden, die selbst über einen niedrigen Integrity Level verfügen. Per default sind jedoch nur die folgenden Orte mit einem niedrigen Integrity Level versehen:

- %Systemdrive%\Users\%username%\AppData\LocalLow
- %Systemdrive%\Users\%username%\AppData\Local\Temp\Low
- %Systemdrive%\Users\%username%\AppData\Local\Microsoft\Temporary Internet Files\Low

Als Standardspeicherort wird dem Benutzer jedoch *\%username%\Downloads* angeboten, und dieser Ordner besitzt den Integritätslevel *medium*. Wie kann ein Benutzer nun dort Dateien speichern, wenn *iexplore.exe* mit niedrigem Integritätslevel läuft? Hier kommt der Prozess *ieuser.exe* ins Spiel, der mit mittlerem Integritätslevel läuft und es ermöglicht, dass ein Benutzer Dateien in Ordnern mit mittlerem Integritätslevel speichern kann, auch wenn der Explorer mit niedrigem Integritätslevel läuft – vorausgesetzt der Benutzer besitzt die nötigen NTFS-Schreibberechtigungen. Über den IE7 heruntergeladene Software läuft in der Standardkonfiguration damit immer mit *mittlerem* Integritätslevel (!) Zwar warnt ein Popup den Benutzer, dass eine Webseite das heruntergeladene Programm ausführen möchte, doch geschieht dies auch nur dann, wenn der Aufruf direkt aus dem Speicherdialog erfolgt. Es ist daher nur eine Frage der Zeit, bis wann es Schadcode gelingt, UAC zu umgehen und damit einen hohen (oder gar System-) Integritätslevel zu erreichen. Dass dies kein hypothetisches

Beispiel ist, hat Matthew Conover für Symantec an dem Vista CTP-Build von Februar 2006 gezeigt.[4]
Es ist zu vermuten, dass Microsoft hier (wieder) einem Kompromiss zwischen Komfort und Sicherheit zugunsten des Komforts nachgegeben hat.

7 MODIFIKATION VON INTEGRITÄTSLEVELN UND POLICIES

Das eben genannte Beispiel wirft die Frage auf, ob und – falls ja – wie sich Integritätslevel verändern lassen, um damit z. B. dem Download-Verzeichnis den Integritätslevel *Low* zu verpassen. Die gute Antwort ist: Ja, die schlechte: nur unkomfortabel. Um Integritätslevel von Objekten verändern zu können bedarf es zunächst der folgenden Voraussetzungen:

- der Benutzer benötigt für das Objekt die beiden NTFS-Berechtigungen: *Berechtigung ändern* und *Besitz übernehmen*
- der Benutzer muss über das Privileg *Verändern einer Objektbezeichnung* (*Modify an object label Properties – SeRelabelPrivilege*) verfügen

Dieses Privileg ist ein neues und sehr mächtiges Privileg; es gestattet – wie der Name sagt – die Veränderung von Integritätsleveln. Dabei gilt noch die Randbedingung, dass ein Benutzer kein Objekt über seinen eigenen Integritätslevel heben kann und dass in der Standardeinstellung sinnvollerweise kein Benutzer über dieses Privileg verfügt. Die Veränderung von Integritätsleveln kann dabei über jedes der drei oben genannten kommandozeilen basierten Tools durchgeführt werden. Wenn diese Voraussetzungen erfüllt sind, steht einer individuellen Anpassung der Integritätslevel von Objekten nichts mehr im Wege. Integritätslevel besitzen darüber hinaus Vorrang vor NTFS-Berechtigungen; wurde etwa einem Ordner der Integritätslevel *High* zugeordnet, dann kann ein Standardbenutzer diesen Ordner nicht mehr verändern, selbst wenn er *Vollzugriff* auf diesen Ordner besitzt.
Es lässt sich jedoch noch einen Schritt weiter gehen, und zwar durch eine Veränderung der Policy: Die Standardeinstellung von Windows Vista legt, wie schon erwähnt, für alle *Securable Objects* eine *No write up*-Policy fest. Damit kann ein Prozess mit einem niedrigen Integritätslevel zwar nicht schreibend auf ein Objekt mit einem höheren Integritätslevel zugreifen, der Prozess kann dies jedoch lesenderweise. Ein Schadprogramm, das im Kontext des Internet Explorers läuft, kann damit z. B. Passwörter des Benutzers aus Dateien auslesen, die einen höheren Integritätslevel besitzen. Sinnvoll wäre also eine Anpassung der Policy, dergestalt dass etwa alle Verzeichnisse im Benutzerprofil bis auf die Verzeichnisse, die einen *Low*-Integritätslevel besitzen, von einem *Low*-Integritätslevel-Prozess nicht gelesen werden dürften. Eine solche Policy wäre eine *No read up*-Policy, und sie lässt sich über das Tool *chml.exe* implementieren. Über das Kommando:

Chml MyPrivateStuff –nr

lässt sich festlegen, dass Prozesse mit einem niedrigeren Integritätslevel als *MyPrivateStuff* diesen Ordner nicht lesen dürfen. *Chml* lässt des weiteren die Implementierung einer *No*

[4] *Zwar wurden die Lücken bis zum aktuellen RTM-Kandidaten behoben, doch gelang es, Integritätslevel durch den Download von Schadcode mit dem im geschützten Modus laufenden IE7 zu umgehen. Siehe Matthew Conover,* Analysis of the Windows Vista Security Model*, Februar 2006,* http://www.symantec.com/avcenter/reference/Windows_Vista_Security_Model_Analysis.pdf

execute-Policy über den Parameter *–nx* zu. Das mit Vista ausgelieferte *icacls.exe* gestattet ausschließlich das Setzen einer *–nw*-Policy.

8 RAUM FÜR VERBESSERUNGEN

Die genannten Beispiele legen das Potential von MIC offen und lassen Raum sowohl für manuelle Nachbesserungen von Seiten des Administrators als auch für Erweiterungen, die wünschenswerterweise von Microsoft vorgenommen werden sollten. Eine manuelle Nachbesserung soll beispielhaft genannt werden: Um die Integrität der Bootpartition – die Partition auf der das Betriebssystemverzeichnis liegt – wirklich im Sinne von MIC vor Schadcode zu schützen, würde sich die Implementierung einer *nr*-Policy für das gesamte C-Laufwerk (so heißt die Bootpartition bei Vista in aller Regel) anbieten. Explizite Ausnahmen von dieser Policy würden nur für die standardmäßigen (oder manuell hinzugefügten) *Low Integrity*-Ordner gesetzt werden. Damit könnten Prozesse, die im IE7 laufen, nur noch in den Verzeichnissen lesen, in denen sie auch tatsächlich lesen dürfen und lesen können müssen. [Achtung: Ich habe diese Policy bisher noch nicht getestet, werde zu einem späteren Zeitpunkt jedoch berichten.]
Wünschenswerte Erweiterungen /Verbesserungen hinsichtlich der Implementierung von MIC an Microsoft sind:

- Verteilung von Integritätsleveln und Policies per GPO: Gerade in Active Directory-Umgebungen sind GPOs ein mächtiges Instrument, um Einstellungen ggf. OU- oder domänenweit zu verteilen. Die Mächtigkeit und das Potential, das Integritätslevel und Policies bieten, machen eine granular steuerbare und zentralisierbare Verteilung wünschenswert.[5]
- Ein komfortableres Tool, z. B. in Form einer MMC-Konsole zur Verwaltung von Integritätsleveln und Policies: Derzeit scheint MIC ausschließlich auf den Schutz der Integrität von Betriebssystemkomponenten ausgelegt zu sein. Eine komfortablere Verwaltung – eine gute Analogie ist hierzu m. E. die Gruppenrichtlinienverwaltungskonsole – würde es ermöglichen, MIC auch für den Schutz von Dateien und Ordnern, die für Benutzer wichtig sind, sinnvoll einzusetzen.
- Eine Erweiterung des bestehenden Policy-Konzepts um *No down*-Policies: Das bestehende Policy-Konzept kennt ausschließlich *No up*-Policies, d.h. Prozesse mit niedrigerem Integritätslevel können nicht (lesend, ausführend oder schreiben - ja nach Policy-Definition) auf Objekte mit höherem Integritätslevel zugreifen. Unter Umständen kann es jedoch auch sinnvoll sein zu verbieten, dass ein Prozess mit einem höheren Integritätslevel Daten auf einem Objekt mit einem niedrigeren Integritätslevel lesen kann oder ausführen darf. Z. B. könnten dann die *Low-Integrity*-Ordner, in dem der Internet Explorer seine temporären Dateien ablegt, mit einer *No read down*-Policy versehen werden, um zu vermeiden, dass ein Prozess mit einem höheren Integrity Level von eventuell dort vorhandenem Schadcode infiziert wird und eine Privilegien-Eskalation verursacht. In jedem Fall böte sich mit der Implementierung von *No down*-Policies ein deutlich granulareres Integritätslevel-Management. Derzeit sind *No down*-Policies jedoch nicht realisierbar (und es ist eine berechtigte Frage, ob sie es jemals sein werden…)

[5] *Bisher hat Microsoft allerdings nichts in diese Richtung geplant.*

9 FAZIT

Mandatory Integrity Control ist ein mächtiges Werkzeug, das wesentlich zur Integrität von Betriebssystemdateien beiträgt. Dabei sollte jedoch nie vergessen werden, dass MIC nur einen – wenn auch wichtigen – Baustein innerhalb der Gesamtsicherheitsarchitektur von Vista darstellt und seine volle Wirksamkeit vor allem zusammen mit sinnvoll gesetzten NTFS-Berechtigungen und UAC entfaltet. Wenn diese Sicherheitskomponenten durch BitLocker-, EFS-Verschlüsselung und wohlbedachte Gruppenrichtlinien ergänzt werden, formt sich das Bild eines sicheren Betriebssystems. Die unkomfortable Bedienung und das Fehlen einer zentralen Verwaltungsmöglichkeit erwecken jedoch den Eindruck, Microsoft hätte bei der Implementierung von MIC mehr an den Schutz des eigenen Betriebssystems als an die Möglichkeiten gedacht, die MIC zum Schutz von Benutzerdateien vor Manipulation durch Schadcode eröffnet. Die Wahl, den Internet Explorer durch Standardbenutzer Code mit mittlerem Integritätslevel speichern und ausführen zu lassen, ist aus sicherheitstechnischer Perspektive fragwürdig. Wünschenswert wäre darüber hinaus eine Erweiterung des Policy-Konzepts um *No down*-Policies. Insgesamt stellt MIC einen wichtigen Schritt in Richtung sichere Betriebssystemarchitektur dar, es sollten jedoch weitere folgen.

Für weiteren Fragen steht Ihnen das Team von ERNW-Deutschland und ERNW-Portugal gern zur Verfügung.

Mit freundlichen Grüßen,

Friedwart Kuhn.

ERNW GmbH
Friedwart Kuhn
Senior Security Consultant

ERNW Enno Rey Netzwerke GmbH
Breslauer Str. 28
69124 Heidelberg
Tel. +49 6221 480390
Fax +49 6221 419008
Mobil +49 15152411855
www.ernw.de