# ERNW Newsletter 12 / Juli 2006

Liebe Partner, liebe Kollegen,

willkommen zur zwölften Ausgabe des ERNW-Newsletters mit dem Thema:

**Vista Security**

Version 1.0 vom 12. Oktober 2006

von:

Enno Rey (erey@ernw.de)

Mit Windows Vista führt Microsoft eine komplette neue Sicherheits-Architektur ein. Zentrale Bestandteile sind hier die Technologien "User Access Control" (UAC) und "Mandatory Integrity Control" (MIC). Wahrend UAC mittlerweile gut dokumentiert ist, ist MIC bis auf einen Blog des Microsoft-Mitarbeiters Steve Riley [1] weitgehend undokumentiert. Unser Mitarbeiter Enno Rey hat daher einige Tests durchgeführt, die zu (für ihn) überraschenden Ergebnissen führten und deren Ergebnis er als Kommentar in Steve Rileys Blog gepostet hat. Wegen des offensichtlich grossen Interesses an Vista und seinem Sicherheits-Modell wird der Text hier nochmals veröffentlicht.

[1] http://blogs.technet.com/steriley/archive/2006/07/21/442870.aspx

==================================================

Steve,

at first thanks for your blog and the valuable insight it provides. After reading your post on MIC I decided to have a look on it, given I've done some research on multi level security systems lately. I stumbled across some obscurities though which I'd like to clarify here. Please note that this was my first installation of Win Vista ever so maybe there are some misunderstandings of concepts on my side...

This is what I did in detail:
- default installation of RC1 (build 5600)
- during installation creation of first and single user (here 'erey')
- logged in as erey, with restricted token and the following privs

```
PRIVILEGES INFORMATION
----------------------

Privilege Name              Description                    State
==============================
==================================== ========
SeShutdownPrivilege         Shut down the system              Enabled
SeChangeNotifyPrivilege      Bypass traverse checking           Enabled
SeUndockPrivilege           Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set     Disabled
SeTimeZonePrivilege         Change the time zone              Disabled
```

- creation of c:\tools directory and download of tools to it (Sysinternals: accesschk, ProcessExplorer and Mark Minasi's chml)

=== Integrity level of files

I then checked the integrity levels of the files just downloaded and here comes the first surprise (or just personal misunderstanding):

```
AccessChk v2.0 - Check account access of files, registry keys or services
Copyright (C) 2006 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\tools\accesschk.zip
  Medium Mandatory Level (Default)
  RW BUILTIN\Administrators
  RW NT AUTHORITY\SYSTEM
  R  BUILTIN\Users
  RW NT AUTHORITY\Authenticated Users
```

```
C:\tools\chml.exe
  Medium Mandatory Level (Default)
  RW BUILTIN\Administrators
  RW NT AUTHORITY\SYSTEM
  R  BUILTIN\Users
  RW NT AUTHORITY\Authenticated Users
```

I had naively expected that these files had an integrity level of 'low' given their origin (internet) and the process that wrote them to disk (IE, supposedly running with 'low' integrity).
Question: why do these files have an integrity level of 'medium'? Lack of intlevel assigning capability in IE in current state?
Or the other way round: what the hell will ever get intlevel 'low' if not such files (executables downloaded from the internet, restricted user, by means of IE, from untrusted sources)?
[one could argue that the Sysinternal stuff is signed, but at least the Minasi stuff isn't].

=== Integrity level of processes

Looking at the integrity level of processes I noticed the following:

1) Restricted token user 'erey' invokes procexp.exe (medium) => process integrity: medium
2) (Run as) Administrator 'erey' invokes procexp.exe (medium) => process integrity: high

This is consistent to your description (or at least my understanding of it ;-):
"when a user invokes a file whose integrity level is higher than low the resulting process will run with the integrity level of the user"
[which contrasts to the Biba model where the lower level of subject (user) and object (file) is chosen, but as you correctly say: a model is just a basis...].

This again rises the question: if a user runs an internet-downloaded executable with admin privileges (and this will happen rather often, think of all the little helper tools available from the internet, requiring admin privs for whatever reason), the resulting process will run with intlevel 'high'. Where's the protection benefit then?
I understand the behaviour is consistent (however I'm not sure if or why the variation from Biba is a good idea here) and I understand that maybe the process needs a 'high' intlevel (to perform it's functions correctly) - and yes UAC came into place and asked me when invoking procexp as an admin - but I'm a bit sceptic about the use then. My previous understanding of "in Vista we're running IE with low privs to protect you from all that bad stuff coming from the internet" was a bit different...

=== Modifying intlevel of files and the results

Time for a change of the integrity level of an executable now...
Trying that gave the following:

- created a copy of procexp.exe

1) - tried (as 'erey' with restricted token) to modify intlevel by
C:\tools>chml procexp_mod.exe -i:l
=> did not work ("Access is denied").

2) I then gave the user 'erey' the privilege "Modify an object label" by gpedit (+ gpupdate).
After logging out and in again I noticed I did not even (seem to) have it when running cmd.exe as admin:

```
PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
================================
=========================================== ========
...
SeCreateGlobalPrivilege        Create global objects                Enabled
SeRelabelPrivilege            Modify an object label               Disabled
SeIncreaseWorkingSetPrivilege   Increase a process working set         Disabled
```

But now chml worked:

```
C:\tools>chml procexp_mod.exe -i:l

chml v1.010 -- Change Windows Integrity Level
by Mark Minasi (c) 2006 www.minasi.com
"chml -?" for syntax, examples and notes.

Integrity level of procexp_mod.exe successfully set to low.
```

--

```
C:\tools>accesschk.exe -i procexp_mod.exe

AccessChk v2.0 - Check account access of files, registry keys or services
Copyright (C) 2006 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\tools\procexp_mod.exe
```

```
Low Mandatory Level (!!!)
RW BUILTIN\Administrators
RW NT AUTHORITY\SYSTEM
R  BUILTIN\Users
RW NT AUTHORITY\Authenticated Users
```

3) Invoking the procexp_mod.exe with intlevel 'low' gave the following:

invoking as restricted user 'erey' => process integrity: 'low' (as indicated by ProcessExplorer)
invoking as admin => process integrity 'high'

BIG surprise! what's going on here? High integrity user runs low integrity executable and resulting process runs on 'high' integrity!?!

Questions:

a) Looking at the "Explain" text in gpedit:
"Modify an object label

This privilege determines which user accounts can modify the integrity label of objects, such as files, registry keys, or processes owned by other users. Processes running under a user account can modify the label of an object owned by that user to a lower level without this privilege."

... I do not understand why 1) didn't work. I invoked a process cmd.exe (supposedly running with intlevel 'medium' given the user was restricted) and tried to modify a file I owned. The observed behaviour seems to contrast the 'help text' but seems consistent to Mark Minasi's comments in the manpage of chml.

b) why did user 'erey' seemingly not dispose of SeRelabelPrivilege after assigning it to him directly and logging out+in, neither as restricted user nor as admin? or at least: why did 'whoami' indicate that? Problem of privilege enumeration in 'whoami'?

c) please explain scenario 3 mentioned above! ;-))

This seems to contradict fully your explanation (or I got it totally wrong).
There seems to be a different behaviour as for the process intlevel, depending on the level of the invoking user.

And: is this a good idea? Biba wouldn't have liked that a lot. And neither do I ;-)
Concluding my observations mean:

User downloads executable from internet, saves file to hard disk and this file has intlevel 'medium': maybe not a good idea.
[maybe resulting from IE currently not assigning intlevels to files it writes]

User running with admin_privs (most windows users will still sometimes do ;-) invokes this executable resulting in a process running with intlevel 'high': debatable...

Security aware user labels some executables down to 'low' but invokes them as admin, process is running with intlevel 'high': what can I say here? ;-)

These are some observations from a guy with some background on "Mandatory" security models, running Vista for the first time.
Thanks in advance for your answer, I appreciate your efforts.

thanks,

Enno
=====================================================

Gerne stehen wir für weitergehende Fragen zu Vista-Security zu Ihrer Verfügung.

Mit freundlichen Grüßen,

Enno Rey
CISSP, CISA

ERNW Enno Rey Netzwerke GmbH
Breslauer Str. 28
69124 Heidelberg
Tel. +49 6221 480390
Fax +49 6221 419008
Mobil +49 173 6745905
www.ernw.de

Besuchen Sie uns auf der Systems (23. bis 27. Oktober 2006) in München, IT-SecurityAREA Halle A4 Stand 13.