Security analysis of the new Microsoft MAC solution

Martin Ondráček

SODATSW spol. s r.o., product director
martin.ondracek@sodatsw.cz

Ondřej Ševeček

GOPAS, a.s., product manager
ondra@sevecek.com

Abstract

Because the general principle behind the Mandatory Access Control (MAC) is well established and has also been implemented into various security technologies, Microsoft decided to add this security technology to the Microsoft Windows family of operating system products - available from Windows Vista. Microsoft Windows are the most widely deployed enterprise operating systems, so the technology will have an impact on the vast majority of the world's confidential data and expertise.

Until Windows Vista was released the operating system enforced and guaranteed only user identity based Discretionary Access Control (DAC) which had, as it later proved, certain limitations. These include, but are not definitely limited to, the inability to separate operating system and user code and data or to enforce strict administrative access separation upon various user roles. Integrity Levels, as Microsoft calls their implementation of MAC, brings a powerful new level of access control. Application data can now be protected even amongst code segments running under a common user or system identity. This provides for better security for services susceptible to remote attacks. It also enables application developers to prevent network authorities from accessing user data.

In this paper we do not only describe the basic principles of general MAC and details of its current Microsoft implementation. The most important part is a discussion of new possibilities within the data security field and its administrative process model. The limitation of the technology is that at is focused on external threats and only few internal ones. Because employees are the biggest threat to a company's data, we would like do explain how it, in its current state of functionality, still cannot prevent the most severe attacks originating from within organizations themselves.

Keywords: Mandatory Access Control, Windows Vista, Integrity Levels.

1 Mandatory access control - introduction

Mandatory access control in computer security means a type of access control by which the operating system restricts the ability of a subject to access or perform some operation on an object. In fact a subject is typically a process; object means file, directory, shared memory etc. Whenever a subject attempts to access an object, the operating system kernel runs an authorization rule which decides whether the access is allowed. That is the reason why each subject and object has a set of security attributes. MAC assigns a security level to all information and assigns a security clearance to each user. The authorization rules are well-known as a policy. Any operation is tested against the policy to determine if the operation is allowed. The goal is to ensure that users have access only to that data for which they have clearance. The security

policy is controlled by a security policy administrator. Users do not have permission to change the policy - for example to grant access to files that would otherwise be restricted.

By contrast Discretionary Access Control (DAC) defines basic access control policies for objects. These are set at the discretion of the owner of the objects. For example, user and group ownership or file and directory permissions. However, DAC allows users (owners of an object) to make policy decisions and assign security attributes - to grant/deny access permission to other users. MAC-enabled systems allow policy administrators to implement organization-wide security policies. This allows only security administrators to define a central policy that is guaranteed to be enforced for all users.

Mandatory Access Controls are considerably safer than discretionary controls, but they are harder to implement and many applications function incorrectly. That's why users are used to DAC, but MAC has only usually been used in extremely secure systems including secure military applications or mission critical data applications.

Mandatory access control models exhibit the following attributes:

- Only administrators, not data owners, make changes to a resource's security label.
- All data is assigned a security level that reflects its relative sensitivity, confidentiality, and protection value.
- All users can read from a lower classification than the one they are granted (A "secret" user can read an unclassified document).
- All users are given read/write access to objects of the same classification (a "secret" user can read/write to a secret document).
- Access to objects is authorized or restricted based on the time of day and dependant upon the labelling on the resource and the user's credentials (driven by policy).

1.1 Some implementations in operating systems

There are only a few robust operating systems with MAC implemented. Actually none of these is certified by TCSEC (Trusted Computer System Evaluation Criteria) robust enough to separate Top Secret from Unclassified. However, some less robust products exist.

- **SELinux** (NSA project) added a MAC architecture to the Linux 2.6 kernel – it used an LSM Linux kernel feature (Linux Security Modules interface). Red Hat Enterprise Linux version 4 and later comes with an SELinux-enabled kernel.
- **Ubuntu and SUSE Linux** has MAC implementation called AppArmor from version 7.10 onwards – this also uses an LSM Linux kernel feature. AppArmor is incapable of restricting all programs and is not included in the kernel.org kernel source tree. In most Linux distributions MAC is not installed.
- **GrSecurity** – this is a patch for the Linux kernel providing a MAC implementation, it is not implemented in any Linux distribution by default. GrSecurity disables the kernel LSM API, because it provides hooks that could be used by rootkits.
- **TrustedBSD** – from version 5.0 onward this is incorporated into releases of the FreeBSD. MAC on FreeBSD comes with pre-built structures for implementing MAC models such as Biba and Multi-Level Security. It can also be used in Apple's Mac OS X through the TrustedBSD MAC framework.

- **Trusted Solaris** – Sun uses a mandatory access control mechanism, where clearances and labels are used to enforce a security policy. However, the capability to manage labels does not imply the kernel strength to operate in Multi-Level Security mode.
- **Microsoft Windows Vista and Server 2008** – finally, the last version of Microsoft operating system implemented Mandatory Integrity Control along with adding Integrity Levels to processes running in a login session. Because Microsoft Windows are the most widely deployed enterprise operating systems, we will describe the implementation through remainder of this article.

2 MAC in Windows

The article describes MAC (Mandatory Access Control) technology in Microsoft Windows Vista and newer versions of Microsoft operating systems built upon Windows NT platform. Unless mentioned otherwise, the term Windows or Vista refers to the current implementation of Microsoft Windows Vista or Microsoft Windows Server 2008 (which both utilise the same kernel versions). Other terms used are in accordance with Microsoft's own public documentation resources unless explicitly stated otherwise.

2.1 Identities

Windows represent the identities of users, groups and other objects by utilising their respective SIDs (Security IDs). Each user or group is assigned a unique (mostly random but unique) SID. A user can be a member of several groups. This would result in the user having several identities (several SIDs), their own and all the group SIDs.

Starting with Vista, each user is also assigned a special system SID (one per user) defining their Integrity Level. The Integrity Level SID is used to enforce MAC (Mandatory Access Control), which is our topic here.

For example, a user could have the following SIDs:

	SID	Note
user's SID	S-1-5-21-1292003277-1427277803-2757683523-1000	random, but system wide unique
group1 SID	S-1-5-21-1292003277-1427277803-2757683523-1001	random, but system wide unique
group2 SID	S-1-5-21-1292003277-1427277803-2757683523-1002	random, but system wide unique
group3 SID	S-1-5-32-544	built-in group, system wide unique the same on all Windows systems
integrity level	S-1-16-12288	system wide unique the same on all Windows systems

Table 1: Example of user's Access Token contents.

The Integrity Levels defined in Vista are:

Integrity Level (Mandatory Level)	SID	Mandatory access rights level
Untrusted	S-1-16-12285	Lowest
Low	S-1-16-12286	
Medium	S-1-16-12287	
High	S-1-16-12288	
System	S-1-16-12289	
Trusted Installer	S-1-16-12290	Highest

Table 2: Integrity Levels on Vista.

Note: The "Mandatory access rights level" column in table 2 is included only to facilitate understanding the principle, which will be discussed later. For now, we will say only that the higher the Integrity Level, the higher the user rights within the system.

As stated, the user's identity is represented by the list of SIDs. The list is usually stored in a system defined data structure called an Access Token. An Access Token is used for access checks against securable objects.

2.2 Windows Access Control Architecture

The enforcement of access control in Windows is the sole responsibility of each separate component providing services for processes running under a specific user account. Such kernel components include, for example, NTFS file system (kernel mode driver), Registry (kernel mode component of executive), Window manager (kernel mode driver). User mode windows services that enforce access control restrictions are, for example, Service Controller, Server service (file sharing), Active Directory, Terminal Services, Winlogon (logon UI), DNS Server and Cluster Services.

To enable access control checks, the operating system provides developers with two similar API functions available both in kernel mode (SeAccessCheck) and in user mode (AccessCheck). The two functions provide stateless access checking (they are parts of call-based .DLLs). It is the responsibility of an application enforcing access control to provide the function with a user's Access Token (user identity proof) and a Security Descriptor (list of access control rules) of the securable object.

As previously mentioned, an Access Token is a data structure which contains a list of user's identities in the form of SIDs. This list must contain the user's own SID and can also contain a list of SIDs for groups of which the user is a member. Starting with Vista, the list also contains one system SID entry asserting the user's Integrity Level.

An Access Token is built by a user mode process called Local Security Authority (LSA, lsass.exe). LSA is a trusted operating system process which starts automatically at system boot. Its binaries are digitally signed by Microsoft and the signatures are checked by kernel mode process loader. This ensures the validity of the LSA process identity. LSA builds Access Tokens only in response to user requests (usually coming from the Winlogon logon process, but these can even be issued by an arbitrary user mode code) which must contain proof of user identity (such as user login name and password). The resulting Access Token is then returned to the user process which requested it. The token contents are unsigned, which is only an implementation detail and is not necessary to ensure its validity (read further).

An Access Token can be attached to a Process (or Thread), the kernel mode structure which describes each process (thread) running on the operating system. Only trusted operating system processes (such as Winlogon or LSA) can attach an arbitrary token to Process (Thread). This fact offers the same level of security as if the token were signed by LSA as previously mentioned.

When an application is checking access it can either use the Process's (Thread's) attached token, or can build its own token from supplied user credentials. Kernel mode components always use only the system guarantied Process or Thread attached tokens.

Security Descriptor (SD) is a data structure comprised mainly of a list of Access Control Entries. Access Control Entries (ACEs) represent access control rules applied to a securable object. Such objects are, for example, files, folders, registry keys, LDAP containers, LDAP objects, shared folders etc. An ACE is basically an access rule defining "who can do what" based on the SID of the user (identity). For example, Martin's SID can read/write or Sales' group SID can read only. Vista also defines several system SIDs for various Integrity Levels which means that an ACE can also be created for something like "a certain Integrity Level can read and execute".

Security Descriptors are stored by each application in their own defined storage such as $security file in NTFS volumes, in registry files or in NTDS.DIT (Active Directory database file). The application itself guaranties SD validity and it is its own responsibility to prevent tampering with this. Before Vista, there was no built-in mechanism to prevent unauthorized modifications of Security Descriptors when the attacker was able to obtain physical access to the storage. Starting with Vista, the storage can be physically secured by BitLocker Drive Encryption which uses TPM (Trusted Policy Module) for encryption key storage, hardware configuration validation and signature checks of operating system binaries (more on this later).

2.3 Trust

As described previously, an Access Token (the user's identity), is built by LSA, system's trusted security authority. In principle an Access Token can even be built manually by application processes because it is unsigned. However a system (LSA) created Access Token cannot be spoofed. The data structure created by LSA is stored in kernel mode and user applications receive only a handle to it.

If an application accepts only the tokens attached to processes/threads, it can be sure they are not tampered with. This attachment could only have been made by the trusted authority itself. From that point onwards the token lives in kernel mode. Processor architecture (64bit only) provides operating system code with the ability to ensure it cannot be tampered with within the kernel mode memory, even by administrators.

There is also an API which enables applications to ask LSA to securely modify existing tokens issued by itself. A user can ask to flag some of the SIDs with certain values or to duplicate the token with a different Integrity Level.

2.4 Process startup

This duplication is requested, for example, when a new process is created from an existing one. The kernel mode code doing so asks LSA to duplicate the token and attach it to the newly created kernel mode structure. This duplication is made by default. There is no modification applied by the default process, but it is possible on behalf of the user process requesting the process creation operation.

Processes running with specific levels of permission can also create children with completely different tokens. However, the tokens must be those originally created in kernel mode by LSA which ensures trust.

In this manner every process has an Access Token of the respective user attached. It ensures that access control enforcement can be carried out later by whichever service requires it.

2.5 Access Token and Integrity Levels

When LSA builds the user's Access Token it always includes the user's own SID and all the SIDs of the groups of which that user is a member. On Vista, LSA also includes the Integrity Level SID dependent upon some hard coded logic.

The Integrity Level included is determined by the circumstances of the user account in question. The system will always include only one Integrity Level SID according to table 2 and table 3:

Logon type	Integrity Level	Access rights level
User has been logged on anonymously (without providing credentials)	Low	Lowest
User is normal user logged on with credentials	Medium	
User is member of local Administrators	High	
User is logging on as a Service	System	
User is Trusted Installer system account	Trusted Installer	Highest

Table 3: Integrity Levels according to logon type.

Note 1: The resulting Integrity Level will usually be the highest possible value of those available as long as the logon type meets several conditions. The except is the anonymously logged on user whose Integrity Level is always Low regardless of other conditions.

Note 2: Trusted Installer system account is a special system account which only LSA itself can log onto. LSA creates such a token only when requested by a special kernel mode LPC (Local Procedure Code) call from another trusted system process called Service Controller. The account is used by the operating system itself for self-code modifications such as the application of updates, upgrades, patches or re/installation of system components. The code running under such an account works only with Microsoft signed operating system binaries.

Note 3: Regardless of logon method, a user cannot receive a higher level of access than System.

Note 4: A user can ask for a token that has a lower access level than what would be the default. A User cannot receive an access level higher than that which applies to his account from the logic above. A User can also ask LSA to duplicate an existing token with a lower Integrity Level applied.

Note 5: Medium Integrity Level is usually issued to normal users.

Note 6: The logic mentioned is currently hard coded. There is no way to modify a user account in order to configure it with a certain preset Integrity Level. Although the list of levels is limited to the 7 levels mentioned by table 1, the levels are only a local setting (not affecting other network systems). This means that future versions of the operating system could offer a larger list of levels or different logic for their assignment. It is probable however, considering its essence that the "level" principle will be adhered to.

The newly created token can be attached to a newly created process. This is usually the first user process created for the user who has just logged on. Any future processes started from the actual process will receive a copy of the token by the normal method of its duplication as mentioned above. If requested, the newly created processes can have the tokens modified to a lower Integrity Level.

2.6 Normal user logon process

When the operating system powers up it first starts the two trusted system processes, LSA and Winlogon. Winlogon then displays logon UI to enable users to provide their credentials. Once the credentials have been collected from a user, Winlogon asks LSA to create an Access Token for that user. LSA creates the token in kernel mode memory, obtains a handle to it and returns the user mode handle to Winlogon. Winlogon then creates a new (the first) user mode process for the user and attaches the created token to it.

Next the user's process receives control and can start to obtain user control requests and create further processes on behalf of the user. The original token, or its duplicates, propagate throughout the processes and each service requested can apply access control mechanisms to the user's token.

We can understand this behaviour by imagining the process is running under the user's identity. In our particular case we will refer to it as running under the user's identity with a certain Integrity Level.

2.7 Security Descriptors

Files, registry keys, folders, services, windows etc. can be assigned a single Security Descriptor. The security descriptor can contain several ACEs for a specific SID and an appropriate access type. Some ACEs will be for user or group SIDs, others for Integrity Level SIDs. These are used further down and are called Integrity Level ACEs.

With Regards to Integrity Levels, we are concerned only about Read, Write and Execute access types. It is possible to map each of the more granular access type to these three generic types. This mapping is hard coded into the access check API routines.

A single ACE can be created, for example, so that it specifies Medium Integrity Level for Read access. Another could be created in the same Security Descriptor with High Integrity Level for Write access. The ACEs can also have standard system flags such as object-inherit (OI), container-inherit (CI), inherit-only (IO) and/or no-propagate-inherit (NP) as well. As there are only three access types there may be, at most, three Integrity Level ACEs in a single Security Descriptor.

If there is no such Integrity Level ACE for a certain access type inside a Security Descriptor, the system then assumes the object has been stamped with Medium Integrity Level for that particular access type (Read, Write or Execute respectively).

2.8 Access checks against Integrity Level ACEs

As opposed to the previous access checking behaviour, Vista introduced modified logic in regard to the Integrity Level ACEs. When an Access Token is checked against the ACE, there can be, basically, only two results from such a check. The Access Token either meets the ACEs requirements or it doesn't.

We could define two simple terms:

- ACE Level means the Integrity Level in the specific Integrity Level ACE.
- Token Level means the Integrity Level in a user's token.

Note: There can be up to three ACE Levels in a Security Descriptor (one for each access type of Read, Write and Execute) and only a single Token Level in a token.

Then the following logic applies. A token meets the ACE rule if the Token Level is the same or higher (table 2) than than the ACE Level.

For example:

Token Level	Possible ACE levels meeting the requirement
High	High, Medium, Low, Untrusted
System	System, High, Medium, Low, Untrusted

Table 4: Example of Token Levels versus ACE Levels in a Security Descriptor.

Examples from the other point of view:

ACEs on a file	Token Level	Granted access
Low Read Medium Write	Medium	Read, Write, Delete, Change ACE
Low Read Medium Write	Low	Read
High Read High Write	Medium	-
System Read System Write	Medium	-

Table 5: Examples of file access ACEs in relation to user Token Levels.

Note: Delete and Change ACE access types are mapped by the API system as the generic Write access type.

Note: The Integrity Level ACEs do not grant appropriate access by themselves. There still have to be appropriate normal ACEs that allow the user access. Integrity Level ACEs can be viewed as a second layer of security applied only after the regular ACEs have been evaluated. It only restricts further access.

2.9 Usage examples

Normal resources (files, registry, etc.) are not stamped with any Integrity Level ACE. This results in them having Medium ACE Level for all three types of access. A normal user is logged on with a Medium Token Level. This means that normal users can generally access whatever resources they are granted access to by other ACEs.

If the system wants certain resources to be accessible only to administrators, services and itself (Trusted Installer), it can stamp them with a High ACE Level. There can also be resources accessible only to system services (System ACE level). This provides a layer of isolation from administrators. Although administrators cannot access the files directly by themselves, they could ask a system service called Service Controller to install their arbitrary code as a system service. This code would in turn run with System Token Level and they would be able to gain access to some files stamped with System ACE Level.

However, there is currently one ACE Level which could prevent even Administrators from accessing some resources. The Trusted Installer ACE Level requires processes running under the Trusted Installer Token Level. Only trusted operating system processes can run under such a Token Level. Unless the trusted processes enable Administrators to access or modify the data, they would have no access at all. Currently, there are no such resources within Vista that would prevent Administrators from gaining access to them, but this could change in future versions of the operating system.

Another use could be demonstrated upon vulnerable or insecure data/processes. For example Internet Explorer, which is one of the most common malicious software attack venues, runs intentionally at Low Token Level. There is only one disk folder (the temporary storage) which is stamped with Low ACL Write access. This means that the IE process, even if compromised, cannot write/modify/harm anything other than the temporary files stored in this location..

3 Impacts for data security

Our analysis of Windows MAC implementation shows, how the operating system works with identities, files, processes etc. This technical description can uncover possible advantages and disadvantages. The most important thing is that a standard user is logged on with Medium Token Level. Standard files, directories or the registry also have Medium ACE Level for all access (because, they are not stamped with any Integrity Level ACE). By default, processes started by a user obtain Medium Integrity Level and elevated processes have High Integrity Level. A process must be explicitly configured to run with Low Integrity Level = low-integrity process.

The limitation of the technology is that it is only focused only upon external threats. The technology protects the system core, programs and files especially against internet threats. Because Internet Explorer (and in future, hopefully, other internet clients) is the only low-integrity process, it cannot interact with other processes with a higher Integrity Level. This means that it cannot interact with other user or system processes, data or the registry. If Internet Explorer runs an harmful code which tries to perform API functions such as CreateRemoteThread (inject a DLL), WriteProcessMemory (send data to a different process), SetThreadContext and related APIs, it will be blocked. This allows potentially-vulnerable applications to be isolated with limited access to the system (system level objects cannot be read or modified by it).

However, our daily experience exposes a different problem. The biggest threats to a company's data are its employees – standard users or administrators. They are able to access data (files), make modifications, make copies, use USB discs etc. They also have many notebooks and remote access. The Windows MAC without third party software doesn't change anything in data security. There is still place for off-line data attack (e.g. notebook/harddisk/USBdisk lost...) or on-line attack (e.g. unlocked computer, emails, administrator attack...). Note that unless there is a physical security on meta/data storage, the ACEs can be modified by an offline attack regardless of their ACE Level.

If we view our users as potential attackers, MAC doesn't help us at all. The user has no new limitation when accessing data, copying data or using unauthorized software (of course, they can't install this, but there are so many portable apps ...). We still have to use sophisticated software for monitoring users, file restrictions, encryption or alerting. Of course, security is a combination of various measures and we must view it as such. MAC can help us to beat unwanted code from the internet more easily but other routes to our data are still available.

In addition to that, there are practically only 2 applicable access rights levels – Low and Medium. Higher levels are used by local administrator, system, or Microsoft (trusted installer). Low integrity level is currently used only by Internet Explorer and is prepared for separation an internet code from the rest of computer. So it is unimaginable to use it for standard applications which are working with user files. In general practice it would mean that almost all standard applications will use the same level – Medium Integrity Level. From this point of view all applications will have the same access to files containing company know-how. Every application running under user token is potential security leak.

And there is one new unknown. The Trusted Installer Integrity Level, which is higher then system and is able to do everything and none is capable to stop it. Trusted installer is prepared for Microsoft (patching

Windows), but it can be potentially used for hiding files, processes or anything else. We have to trust Microsoft, that his binaries are safe and without mistakes.

Because there is no possibility to add own access level or modify the logic of Integrity Levels, the result of the whole technology is only operating system protection. Microsoft prepared it to increase system stability and lower quantity of administrator interventions. Data classification and its separation are miles out.

Windows Vista is still not the enterprise standard. This is the reason why many software producers do not prepare software to utilise the future potential of Integrity Levels. There will be quite a large gap between the technology push-off and usage. We predict that vast usage is probable in the next 3-5 years. On the other hand once most of the common threats have been blocked, the virus/trojans/spyware producers will have to change their target. Because social engineering is becoming more and more popular, this could be a suitable field for such groups. Preventing or controlling this will be a much harder goal for software producers. This is going to be challenge in the future.

4 Conclusion

Starting with Windows Vista, Microsoft has added another layer of access control security exhibiting similarities to Mandatory Access Control. Currently this implementation is restricted to a small set of useful levels which are applied only to prevent malicious software attacks and increase system stability from the point of preventing system code/configuration modifications.

Integrity Levels technology is not a silver bullet. It currently has only limited use for applications developers, but future development could bring greater application possibilities due to the fact that the technology does not require network wide compatibility and is only a matter of local system access checks.

There has also been a significant step into the realm of "self-secure" operating systems which prevents even local administrators from doing anything untoward. The presumed goal is probably to enforce the DRM (Digital Rights Management) or RMS (Rights Management Service) features of the operating system and to prevent local administrators from tampering with such technologies.

There is no overall approach in data security against offline attacks and theft by authorised users. There is still a need to encrypt company's data and control user activities.

On the other hand, this not only secures the operating system code itself, but also provides Microsoft (the owner of the code and the higher Integrity Levels) with the future ability to hide code/data or enforce policies that would not be under the local administrator's control.

References

[1] http://msdn.microsoft.com/en-us/library/bb625964.aspx

[2] http://msdn.microsoft.com/en-us/library/bb250462.aspx

[3] http://www.minasi.com/apps/

[4] http://www.securityfocus.com/infocus/1887/2

[5] http://www.symantec.com/avcenter/reference/Windows_Vista_Security_Model_Analysis.pdf

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

[7] http://blogs.technet.com/markrussinovich/archive/2007/02/12/638372.aspx

[8] http://en.wikipedia.org/wiki/Mandatory_access_control