The Windows Security Journey

Version 3.0
March-2024

By Dr. Shlomi Boutnaru

Created using [Craiyon, AI Image Generator]

Introduction

When starting to learn OS security I believe that there is a need for understanding multiple technologies and concepts. Because of that I have decided to write a series of short writeups aimed at providing the security vocabulary.

Overall, I wanted to create something that will improve the overall knowledge of Windows' different security mechanisms included with Windows in writeups that can be read in 1-3 mins. I hope you are going to enjoy the ride.

Lastly, you can follow me on twitter - @boutnaru (https://twitter.com/boutnaru). Also, you can read my other writeups on medium - https://medium.com/@boutnaru. Lastly, You can find my free eBooks at https://TheLearningJourneyEbooks.com.

Lets GO!!!!!!

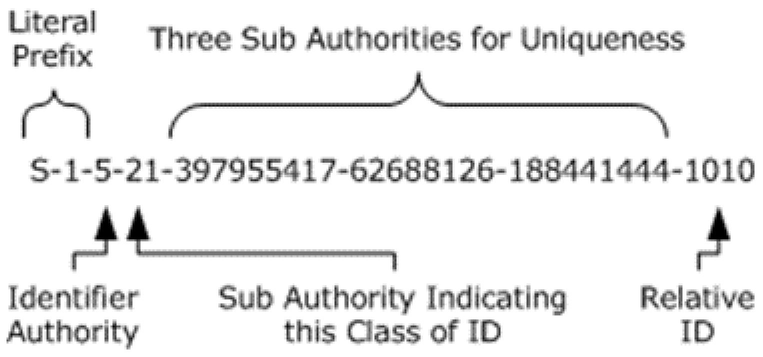
SID (Security Identifier)

The goal of an SID is to uniquely identify a security principal/group. When talking about a security principal we mean any entity that can authenticate to the operating system like: user/computer account or a thread/process that runs with the security context of one of those. Every time a user is logged on the system creates an access token for that user (more on that in a future writeup). This access token holds the user's SID, privileges and the SIDs for any groups that the user is part of[1].

Moreover, there is a specific format for an SID. We can split it into three main parts: revision, identifier authority and sub authorities. Revision, which specifies the version of the SID structure. Identifier authority, which specifies the highest level of authority that can issue an SID for a security principal. Sub authorities, which hold the most important information (can identify a local computer/domain) and its last part is an RID (relative identifier) that identifies a specific user/group in a local computer/domain - as shown in the diagram below[2].

An example of some well-known SIDs are: "S-1-1-0" (group that includes all users), "S-1-0-0" (aka NULL SID, a group with no members). They are called "Universal well-known SIDs"[3]. Also, there are well-known RIDs such as: 500 (Administrator), 501 (Guest). Since Windows 2008/Vista most of the system files are owned by the "TrustedInstaller" SID, in order to prevent a process running with Administrator/Local System permissions to overwrite the OS files[4].

Lastly, there are also "Capability SIDs" which grant access to specific resources (like cameras, documents, location and more). Those type of SIDs that the system is aware of are stored in the registry value "AllCachedCapabilities" under "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SecurityManager\CapabilityClasses"[5].


Literal Prefix
Three Sub Authorities for Uniqueness
S-1-5-21-397955417-62688126-188441444-1010
Identifier Authority
Sub Authority Indicating this Class of ID
Relative ID

[1] https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-identifiers
[2] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-azod/ecc7dfba-77e1-4e03-ab99-114b349c7164
[3] https://learn.microsoft.com/en-us/windows/win32/secauthz/well-known-sids
[4] https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-security-identifiers
[5] https://renenyffenegger.ch/notes/Windows/security/SID/index

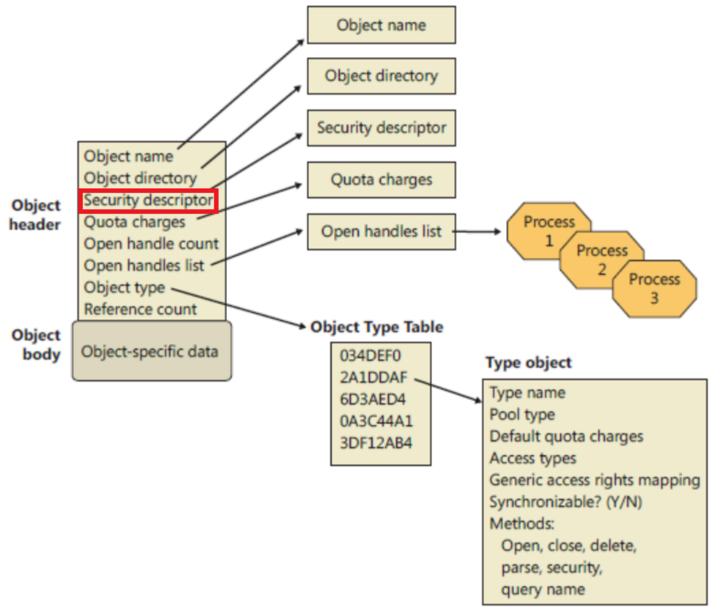
SD (Security Descriptor)

The goal of a security descriptor (SD) is to hold the security information that is related with a specific securable object. Examples for securable objects are: file, folder, network share, printer, registry key, synchronization object, active directory objects and more. The structure which describes a SD is defined in "winnt.h" and is named "SECURITY_DESCRIPTOR"[6].

Overall, every object created by the "Object Manager" in Windows has a SD. Each objects has an header with different fields (like object name, reference count, object type and more) one of them is the security descriptor[7]. You can see an illustration of that in the diagram below[8].

Moreover, we can think about an SD as containing four main fields: an owner, group, DACL (Discretionary Access Control List) and SACL (System Access Control List) . DACL is used for allowing/denying permissions which SACL is used for auditing[9]. The description of each entity in the structure is stored in the form of an SID (Security Identifier). More on those in future writeups.

Lastly, the "SECURITY_DESCRIPTOR" is a compact binary representation of the security associated with a specific object. Because it is not convenient to use it, there is a text-based form for representing it. This format is called SSDL (Security Descriptor Description Language). It has specific text tokens in order to describe: access rights, user accounts, user-mode drivers and more[10].



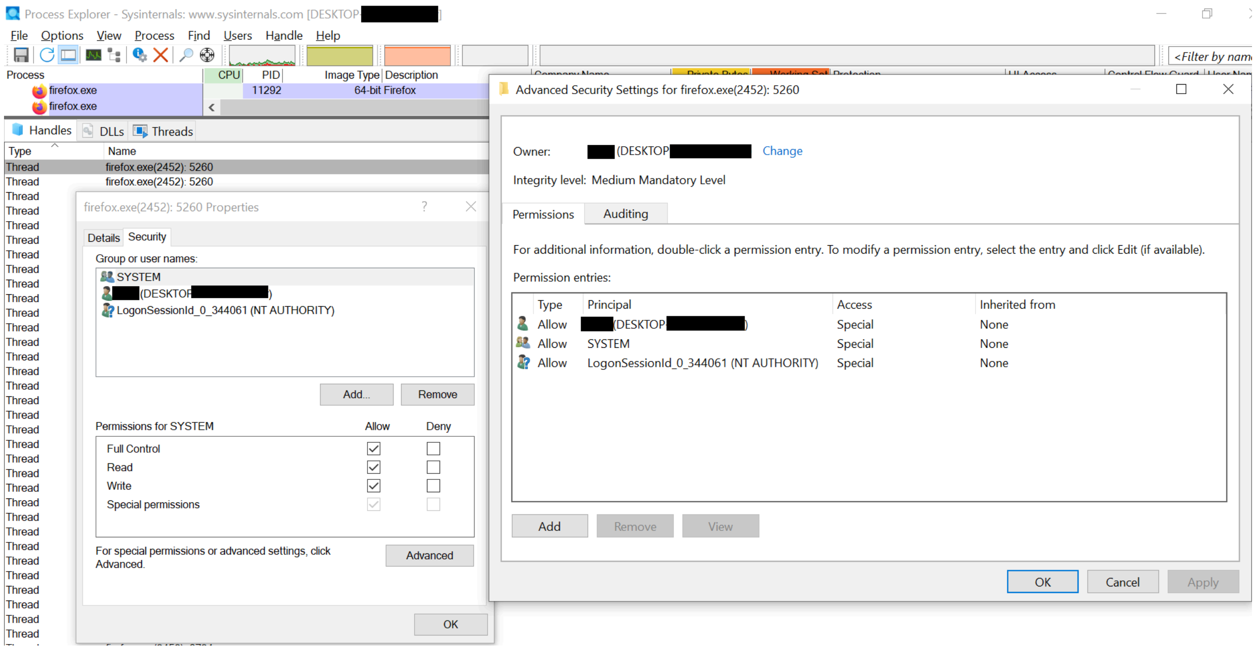
6 https://learn.microsoft.com/en-us/windows/win32/api/winnt/ns-winnt-security_descriptor
7 https://www.geoffchappell.com/studies/windows/km/ntoskrnl/inc/ntos/ob/object_header/index.htm
8 https://www.tophertimzen.com/resources/cs407/slides/week02_01-KernelObjects.html#slide16
9 https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-azod/ec52bde3-9c86-4484-9080-e72148a2d53b
10 https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/sddl-for-device-objects?redirectedfrom=MSDN

Securable Objects

Overall, “securable objects” are Windows objects[11] that can have a “security descriptor”[12]. All named Windows objects are securable. There are also unnamed objects which are securable like processes and threads[13] - as shown in the screenshot below.

Moreover, each securable object has specific elements that are elaborated next. An owner’s (user/group) SID[14]. A DCAL (Discretionary Access Control List) which contains a list of group/user SIDs and the access rights each of them has - as shown in the screenshot below in the permissions tab. A SACL (System Access Control List) which states what logging/auditing should be done when accessing the object. Also, there is a group associated with the object which is for POSIX compatibility only[15].

Lastly, examples of securable objects (but not limited to) are: files, directories, desktops, processes, threads, named pipes, mailslots, network shares, printers, private objects, events, semaphores, WMI namespaces and waitable timers and windows stations[16]..


Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP[redacted]]
File Options View Process Find Users Handle Help
Process CPU PID Image Type Description
firefox.exe 11292 64-bit Firefox
firefox.exe
Handles DLLs Threads
Type Name
Thread

firefox.exe(2452): 5260

firefox.exe(2452): 5260 Properties

Details | Security

Group or user names:
- SYSTEM
- [redacted](DESKTOP[redacted])
- LogonSessionId_0_344061 (NT AUTHORITY)

Add... | Remove

Permissions for SYSTEM	Allow	Deny
Full Control	✓	
Read	✓	
Write	✓	
Special permissions	✓	

For special permissions or advanced settings, click Advanced. — Advanced

OK

Advanced Security Settings for firefox.exe(2452): 5260

Owner: [redacted](DESKTOP[redacted]) Change

Integrity level: Medium Mandatory Level

Permissions | Auditing

For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

Type	Principal	Access	Inherited from
Allow	[redacted](DESKTOP[redacted])	Special	None
Allow	SYSTEM	Special	None
Allow	LogonSessionId_0_344061 (NT AUTHORITY)	Special	None

Add | Remove | View

OK | Cancel | Apply

[11] https://medium.com/@boutnaru/windows-objects-2c289da600bf
[12] https://medium.com/@boutnaru/windows-security-security-descriptor-sd-ba95b8fa048a
[13] https://learn.microsoft.com/en-us/windows/win32/secauthz/securable-objects
[14] https://medium.com/@boutnaru/windows-security-sid-security-identifier-d5a27567d4e5
[15] https://renenyffenegger.ch/notes/Windows/development/objects/securable/index
[16] http://winapi.freetechsecrets.com/win32/WIN32Securable_Objects.htm

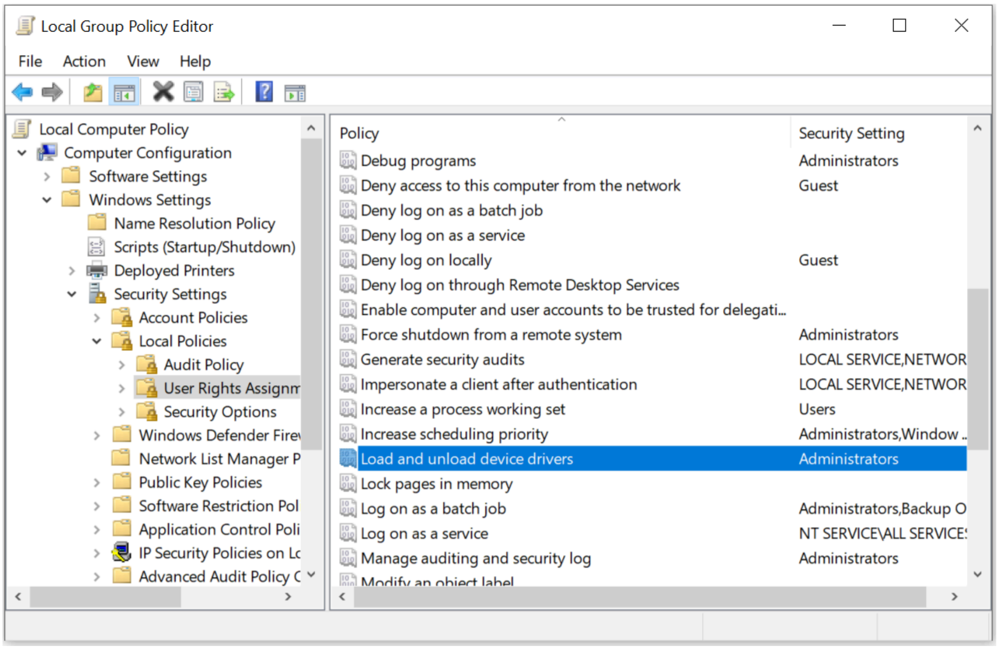
Privileges

Privileges are rights given for a specific account (user/group) which allows performing different system related operations on the local computer. Think about: changing the system time, loading a device driver, shutting down the system and more. There is a difference between access rights to privileges[17].

Thus, we can say that privileges control the access to system resources/system related tasks while access rights control access to securable objects (such as files, directories, registry keys and more). We assign privileges to user/group accounts whereas access rights are granted as part of DACLs.

Moreover, the operating system represents a privilege in a category of "User Rights Assignments". We can modify them using the "Local Group Policy" (or the "Group Policy") MMC snap-in[18] - as shown in the screenshot below.

Lastly, the privileges are defined using constants in the following pattern "SE_[DESCRIPTION]_NAME" and also has a text format which is in the pattern of "Se[DESCRIPTION]Privilege". A couple of examples are: "SE_CREATE_PAGEFILE_NAME"\"SeCreatePagefilePrivilege" which enables creating a new pagefile, "SE_DEBUG_NAME"\"SeDebugPrivilege" which is required for debugging/adjusting the memory of a processes owned by a different user account and "SE_LOAD_DRIVER_NAME"\"SeLoadDriverPrivilege" which is required to load/unload a device driver (it is also the one marked in the screenshot below).



Policy	Security Setting
Debug programs	Administrators
Deny access to this computer from the network	Guest
Deny log on as a batch job	
Deny log on as a service	
Deny log on locally	Guest
Deny log on through Remote Desktop Services	
Enable computer and user accounts to be trusted for delegati...	
Force shutdown from a remote system	Administrators
Generate security audits	LOCAL SERVICE,NETWOR
Impersonate a client after authentication	LOCAL SERVICE,NETWOR
Increase a process working set	Users
Increase scheduling priority	Administrators,Window ...
Load and unload device drivers	Administrators
Lock pages in memory	
Log on as a batch job	Administrators,Backup O
Log on as a service	NT SERVICE\ALL SERVICE:
Manage auditing and security log	Administrators
Modify an object label	

---
[17] https://learn.microsoft.com/en-us/windows/win32/secauthz/privileges
[18] https://learn.microsoft.com/en-us/windows/win32/secauthz/privilege-constants

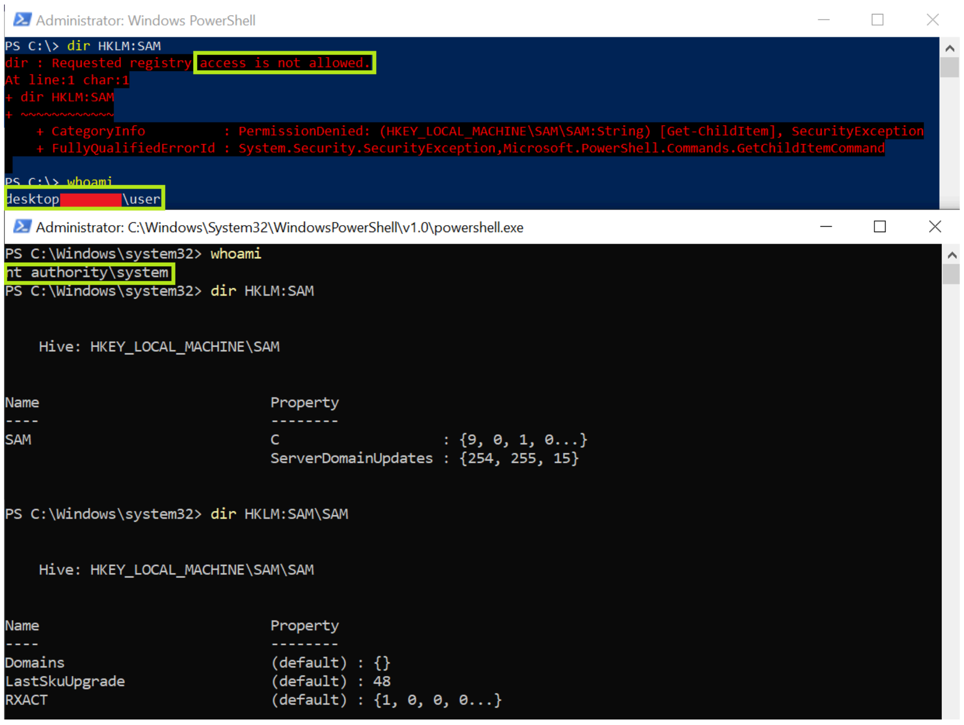
SAM (Security Account Manager)

SRM (Security Account Manager) is the DB in Windows that stores the user names/passwords of the local user defined on the system. By configuring SAM we allow users to authenticate to the local system[19].

Moreover, the SAM file is located at “%windir%\System32\config\SAM” which is mounted in the registry in the following “HKEY_LOCAL_MACHINE\SAM”[20]. In order to view its content we need to run as SYSTEM and Local Administrator is not enough - as shown in the screenshot below.

Thus, different hashes can be stored in SAM like LM hash and NTLM hash (more on those and others in future writeups). We can think about SAM as the equivalent of “/etc/passwd”, “/etc/shadow” and “/etc/group” files under Linux.

Because Microsoft wanted to increase the security around the hashes stored in SAM they have created SYSKEY. It uses a system key for encrypting to protect the account password information stored in the SAM. The system key can be saved locally, on a floppy disk or stored locally but protected by an admin password[21]. Lastly, SYSKEY support was removed from Windows 10 version 1709[22].


Administrator: Windows PowerShell
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
PS C:\> dir HKLM:SAM
dir : Requested registry access is not allowed.
At line:1 char:1
+ dir HKLM:SAM
+ ~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (HKEY_LOCAL_MACHINE\SAM\SAM:String) [Get-ChildItem], SecurityException
    + FullyQualifiedErrorId : System.Security.SecurityException,Microsoft.PowerShell.Commands.GetChildItemCommand

PS C:\> whoami
desktop[redacted]\user
```

```
PS C:\Windows\system32> whoami
nt authority\system
PS C:\Windows\system32> dir HKLM:SAM


    Hive: HKEY_LOCAL_MACHINE\SAM


Name                           Property
----                           --------
SAM                            C                   : {9, 0, 1, 0...}
                               ServerDomainUpdates : {254, 255, 15}


PS C:\Windows\system32> dir HKLM:SAM\SAM


    Hive: HKEY_LOCAL_MACHINE\SAM\SAM


Name                           Property
----                           --------
Domains                        (default) : {}
LastSkuUpgrade                 (default) : 48
RXACT                          (default) : {1, 0, 0, 0...}
```

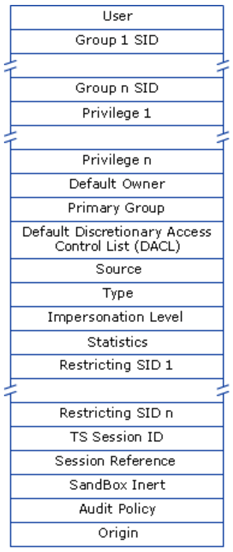
[19] https://www.calcomsoftware.com/what-is-windows-security-accounts-manager/
[20] https://viperone.gitbook.io/pentest-everything/everything/everything-active-directory/credential-access/credential-dumping/security-account-manager-sam
[21] https://learn.microsoft.com/en-us/windows-server/security/kerberos/system-key-utility-technical-overview
[22] https://learn.microsoft.com/en-us/troubleshoot/windows-server/identity/syskey-exe-utility-is-no-longer-supported

Access Token

“Access Token” is an object which represents the access rights/privileges/identity for a specific process/thread. The operating system uses the access token in order to identify the user when a specific thread interacts with a securable object[23] or when it tries to perform a system task (that requires some kind of privilege)[24].

Thus, if a user authenticates to a system, the Local Security Authority (LSA) creates an access token (to be accurate it is the primary access token, as described in more detail later). It contains the SID of the user, the SIDs of all the groups the user belongs to, a list of privileges, the SID of the owner (user/group), the primary group (for POSIX subsystems), default DACL, source (process that caused the token to be created - RPC/LAN Manager/Session Manager/etc), type (primary/impersonation), impersonation level, restricting SIDs, Terminal service session ID (if relevant), session reference, SandBox inert, audit policy and origin - as shown below[25].

Moreover, using different Win32 API functions we can read/manipulate access tokens. As example we can use “OpenProcessToken”[26] or “OpenThreadToken”[27] to get a handle to the access token of the process/thread. Also, we can use “DuplicateTokenEx”[28] for duplicating the access token of the current process and “CreateProcessWithTokenW”[29] which allows creation of a process with a specified token. The access token is stored in kernel mode using “struct _TOKEN”[30].



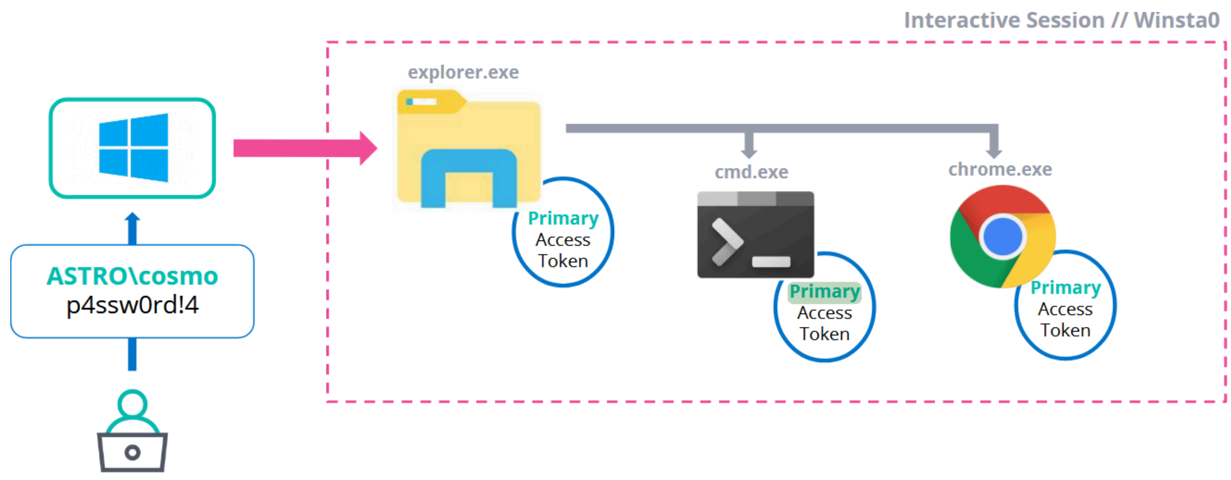
23 https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad
24 https://learn.microsoft.com/en-us/windows/win32/secauthz/access-tokens
25 https://learn.microsoft.com/pt-pt/previous-versions/windows/server/cc783557(v=ws.10)
26 https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocesstoken
27 https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openthreadtoken
28 https://learn.microsoft.com/en-us/windows/win32/api/securitybaseapi/nf-securitybaseapi-duplicatetokenex
29 https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createprocesswithtokenw
30 https://www.ired.team/miscellaneous-reversing-forensics/windows-kernel-internals/how-kernel-exploits-abuse-tokens-for-privilege-escalation

Primary Access Token

Overall, there are two types of access tokens[31] - as stated in the type field of the access token. Those are "Primary Token" and "Impersonation Token". In this writeup I am going to focus on the first one.

A primary token can only be associated with a process. Processes inherit a copy of the parent's process primary token[32]. Due to that, when a thread is attempting to access a securable object[33] by default this token is checked (threads can have an impersonation token but that is for a different writeup). Also, this token belongs to the user account that created the process[34].

Thus, every process has a primary token that it gets from its parent process - as shown in the diagram below[35]. Because primary tokens are associated with processes they are also known as "Process Tokens". We can also state that they are created while authenticating interactively[36].



[31] https://medium.com/@boutnaru/windows-security-access-token-81cd00000c64
[32] https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation/access-tokens
[33] https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad
[34] https://jsecurity101.medium.com/better-know-a-data-source-access-tokens-and-why-theyre-hard-to-get-7bc951eae0b9
[35] https://i.blackhat.com/USA-20/Thursday/us-20-Burgess-Detecting-Access-Token-Manipulation.pdf
[36] https://sensepost.com/blog/2022/abusing-windows-tokens-to-compromise-active-directory-without-touching-lsass/

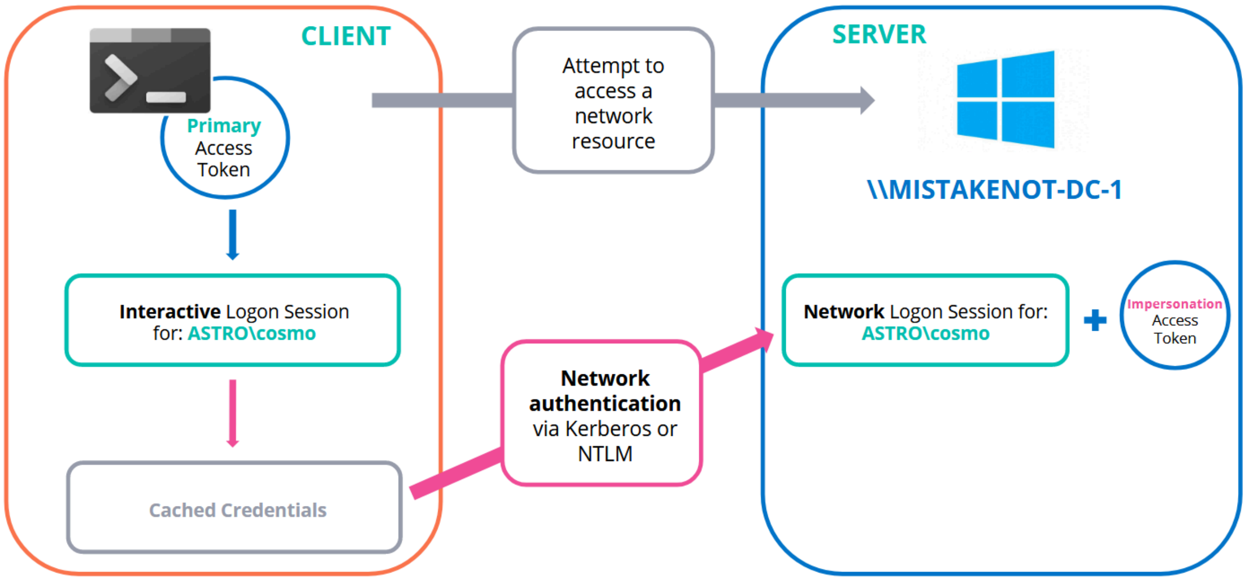
Impersonation Access Token

Overall, there are two types of access tokens[37] - as stated in the type field of the access token. Those are "Primary Token" and "Impersonation Token". In this writeup I am going to focus on the second one.

Basically, impersonation is a mechanism which allows server processes to run by using the credentials of some client. Meaning using creaditails of another user than its primary token[38].

Thus, when can that impersonation allow a thread to switch to a different security context. By default, threads inherit the same security context as the primary token[39] of the process[40].

One of the main use-cases for impersonation is asking the a server to execute code on behalf of the user performing a network authentication - as shown in the diagram below[41]. It can also be used for cases where we want an application/process to have a thread running code with a different security context (than the other threads). However, we need to be careful because all the threads share the same memory space so one thread can hijack the execution flow of another.


CLIENT
Primary Access Token
Interactive Logon Session for: ASTRO\cosmo
Cached Credentials
Attempt to access a network resource
Network authentication via Kerberos or NTLM
SERVER
\\MISTAKENOT-DC-1
Network Logon Session for: ASTRO\cosmo
+
Impersonation Access Token

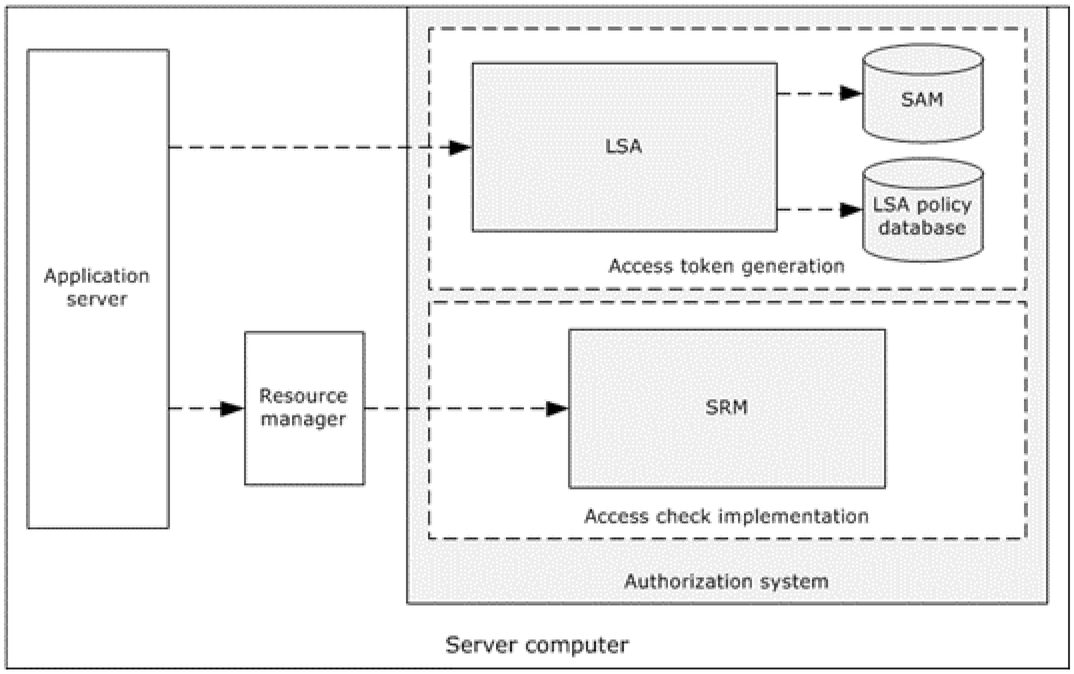
[37] https://medium.com/@boutnaru/windows-security-access-token-81cd00000c64
[38] https://medium.com/@boutnaru/windows-security-primary-access-token-e295a35796a9
[39] https://medium.com/@boutnaru/windows-security-primary-access-token-e295a35796a9
[40] https://i.blackhat.com/USA-20/Thursday/us-20-Burgess-Detecting-Access-Token-Manipulation.pdf
[41] https://i.blackhat.com/USA-20/Thursday/us-20-Burgess-Detecting-Access-Token-Manipulation.pdf

SRM (Security Reference Monitor)

SRM (Security Reference Monitor) is a component that is part of the Windows executive (stored in %systemroot%\System32\ntoskrnl.exe). SRM is responsible for implementing the authorization system (together with LSA as shown in the diagram below). Also, SRM implements the access check algorithm[42].. This means it checks the access to different resources by getting the access token[43] of the subject and comparing it to the ACEs (Access Control Lists) in the security descriptor of the securable object[44].

Moreover, the routines that provide a direct interface with the SRM are those prefixed with "Se"[45]. An example of such function is: "SeAccessCheck" which determines if the requested access to an object can be granted[46]. If we want we can go over a reference implementation of "SeAccessCheck" as part of ReacOS[47].

Lastly, we can say that the "Object Manager" uses SRM to check if a specific process/thread has the proper rights to execute a certain action on an object. Also, it is part of the flow when implementing auditing functionality when objects are being accessed[48].



[42] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-azod/d28d536d-3973-4c8d-b2c9-989e3a8ba3c5
[43] https://medium.com/@boutnaru/windows-security-access-token-81cd00000c64
[44] https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad
[45] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-security-reference-monitor
[46] https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-seaccesscheck
[47] https://github.com/reactos/reactos/blob/master/ntoskrnl/se/accesschk.c#L1966
[48] https://cs.gmu.edu/~menasce/osbook/nt/sld034.html

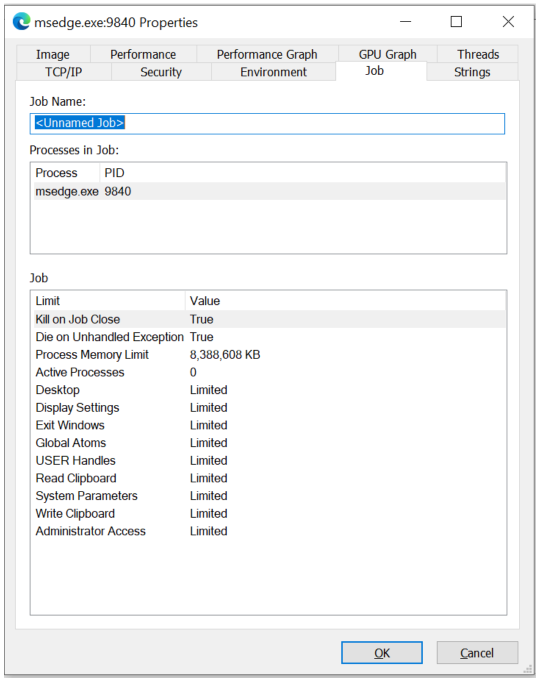
Job Object

By using a job object we can manage a group of processes as one unit. Thus, an operation performed on a job object affects all the processes which are part of that job[49].

Moreover, in order to create a job object we can use the Win32 API call “CreateJobObjectA”[50]. When creating a job it has no processes associated with it, so we need to use the function “AssignProcessToJobObject”[51]. Until Windows 8/Windows Server 2012 a process could be associated with one job only. By the way, for getting an handle for an existing object we can use the “OpenJobObjectA” function[52].

Overall, by using jobs we can limit the usage of system resources by processes like: process priority, time limit, working set, number of child processes, desktop creation, writing data to the clipboard and more. For setting the limits we use the function “SetInformationJobObject”[53].

Lastly, job objects are being used in sandboxes like in web browsers (shown in the screenshot below) and are one of the building blocks of “Windows Containers". There are also named/unanmed, securable objects[54] and shareable objects - as shown in the screenshot below taken from Sysinternals’ “Process Explorer”[55].



Job Name: <Unnamed Job>

Processes in Job:

Process	PID
msedge.exe	9840

Job

Limit	Value
Kill on Job Close	True
Die on Unhandled Exception	True
Process Memory Limit	8,388,608 KB
Active Processes	0
Desktop	Limited
Display Settings	Limited
Exit Windows	Limited
Global Atoms	Limited
USER Handles	Limited
Read Clipboard	Limited
System Parameters	Limited
Write Clipboard	Limited
Administrator Access	Limited

[49] https://learn.microsoft.com/en-us/windows/win32/procthread/job-objects
[50] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createjobobjecta
[51] https://learn.microsoft.com/en-us/windows/win32/api/jobapi2/nf-jobapi2-assignprocesstojobobject
[52] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-openjobobjecta
[53] https://learn.microsoft.com/en-us/windows/win32/api/jobapi2/nf-jobapi2-setinformationjobobject
[54] https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad
[55] https://learn.microsoft.com/en-us/sysinternals/downloads/sysinternals-suite

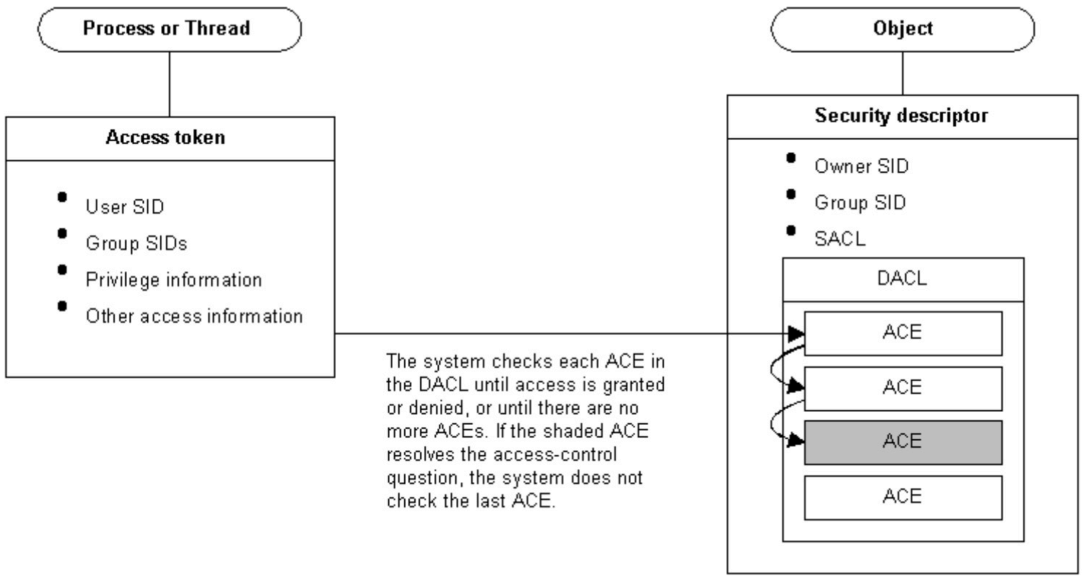
ACL (Access Control List)

ACL (Access Control List) is a list of ACEs (Access Control Entries). Every ACE identifies a trustee (user account/group/logon session) and the relevant allowed/denied/audited access for that trustee[56].

Overall, there are two types of ACLs which are in use in Windows systems: DACL aka as "Discretionary Access Control List" and SACL aka "System Access Control List"[57]. More information about those types in future writeups. Those types of ACLs are part of the security information stored as part of the "Security Descriptor"[58] related to securable objects[59] - as shown in the diagram below[60].

Moreover, every ACE has four main components. The first, the SID[61] to whom the access information in this ACE is relevant for. Second, a flag denoting the type of ACE (deny/allow/audit). Third, flags regarding the inheritance of the specific ACE. Forth, an access mask which is a 32 bit that describes the rights relevant for this ACE[62].

Lastly, due to the fact we have DACL and SACL, usually when saying ACE we talk about the first one and when saying System ACE we mean the second one.


Process or Thread
Access token
User SID
Group SIDs
Privilege information
Other access information
Object
Security descriptor
Owner SID
Group SID
SACL
DACL
ACE
ACE
ACE
ACE
The system checks each ACE in the DACL until access is granted or denied, or until there are no more ACEs. If the shaded ACE resolves the access-control question, the system does not check the last ACE.

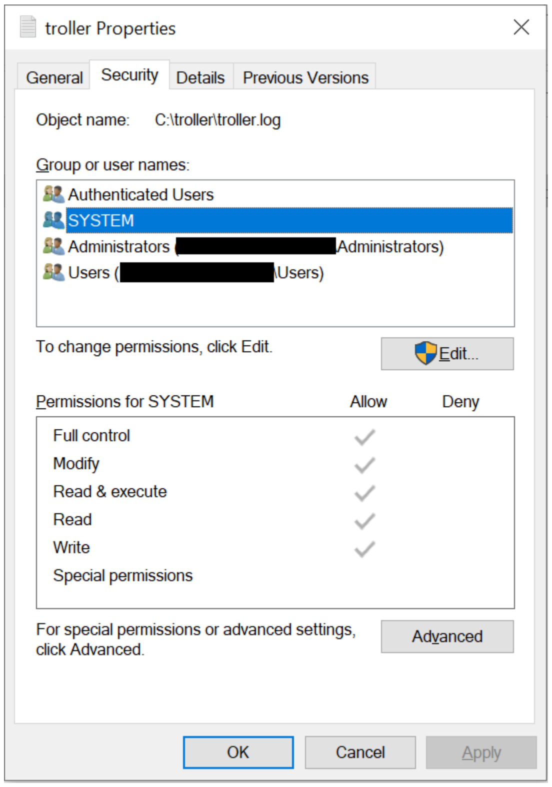
[56] https://learn.microsoft.com/en-us/windows/win32/secauthz/access-control-lists
[57] https://www.securew2.com/blog/windows-access-control-acl-dacl-sacl-ace
[58] https://medium.com/@boutnaru/windows-security-security-descriptor-sd-ba95b8fa048a
[59] https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad
[60] https://developer.aliyun.com/article/747446
[61] https://medium.com/@boutnaru/windows-security-sid-security-identifier-d5a27567d4e5
[62] https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation/acls-dacls-sacls-aces

DACL (Discretionary Access Control List)

In general DACL (Discretionary Access Control List) is an ACL[63] which identifies the trustees that allowed/denied access to a securable object[64].Thus, if the securable object does not have any DACL (Null) the SRM[65] allows everyone full access to it. If the list of ACL is empty no one has any access to the object[66].

Moreover, when a thread tries to access a securable object, the system goes over the ACEs in the DACL until it finds one that allows/denies the access (think about it like firewall rules). The predefined order of ACEs are as follows: all explicit ACEs are before inherited ACEs and the inherited ones are placed in the order in which they are inherited. By the way, in every level access denied ACEs are placed before the access allowed ACEs ones[67].

Lastly, for configuring a DACL using the UI we just go to the properties of the object and select the "security tab", there we can edit the DACL of that specific object - as shown in the screenshot below. We can also use CLI tools like cacls.exe/icacls.exe (but that is for a different writeup).



Object name: C:\troller\troller.log

Group or user names:
- Authenticated Users
- SYSTEM
- Administrators ([redacted]\Administrators)
- Users ([redacted]\Users)

To change permissions, click Edit.

Permissions for SYSTEM	Allow	Deny
Full control	✓	
Modify	✓	
Read & execute	✓	
Read	✓	
Write	✓	
Special permissions		

For special permissions or advanced settings, click Advanced.

[63] https://medium.com/@boutnaru/the-windows-security-journey-acl-access-control-list-b7d9a6fe4282
[64] https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad
[65] https://medium.com/@boutnaru/windows-security-srm-security-reference-monitor-d715f96d9fd6
[66] https://learn.microsoft.com/en-us/windows/win32/secauthz/dacls-and-aces
[67] https://www.tenouk.com/ModuleH2.html

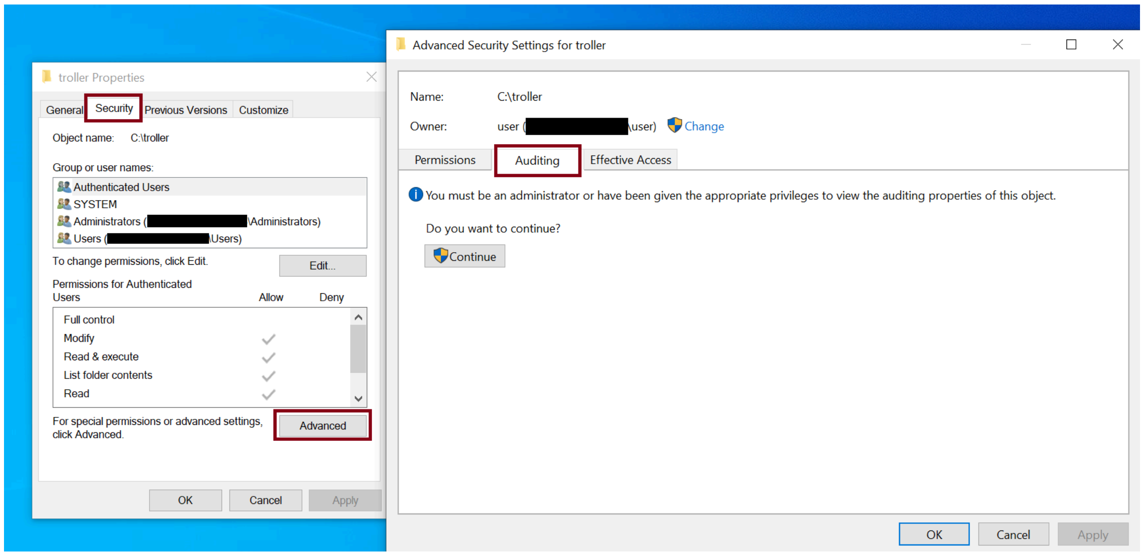
SACL (System Access Control List)

Overall, a SACL (System Access Control List) is an ACL[68] which enables the administrators of a system to audit attempts of accessing securable objects[69]. Every ACE (Access Control Entry) defines the type of access attempt that causes to generate an audit trail while performed by a trustee[70].

Thus, an ACE as part of an SACL can emit an audit record when an access attempt is failed/succeeds/both. The system writes audit messages to the security event log[71]. In order to read/write object's SACL the relevant thread/process should enable as part of its access token[72] the "SE_SECURITY_NAME" privilege[73].

Moreover, the "SE_SECURITY_NAME" privilege is defined as managing auditing and the security log[74]. We can use "SetNamedSecurityInfoA"/"SetNamedSecurityInfoW"[75] or "GetNamedSecurityInfoA"/"GetNamedSecurityInfoW" in order to access the SACL. Those functions enable the "SE_SECURITY_NAME" privilege.

Lastly, in order to configure an SACL on a securable object like a file/directory we go to its properties and then we go to the "security tab". In the "security tab" we need to press the "Advanced" button - as shown in the screenshot below. In the advanced security setting we can go to the "auditing tab" - also shown in the screenshot below.



Object name: C:\troller

Group or user names:
Authenticated Users
SYSTEM
Administrators ([redacted]\Administrators)
Users ([redacted]\Users)

To change permissions, click Edit.

Permissions for Authenticated Users

For special permissions or advanced settings, click Advanced.

Name: C:\troller

Owner: user ([redacted]\user)

You must be an administrator or have been given the appropriate privileges to view the auditing properties of this object.

Do you want to continue?

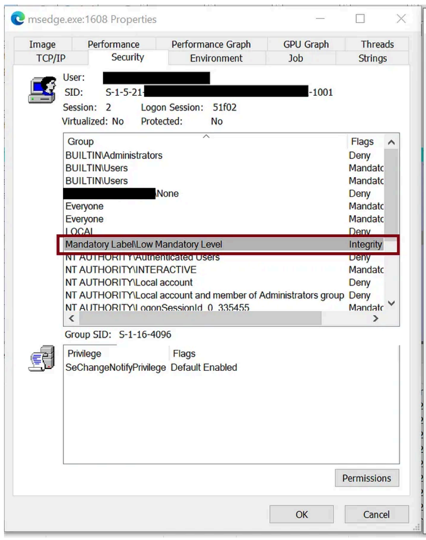
[68] https://medium.com/@boutnaru/the-windows-security-journey-acl-access-control-list-b7d9a6fe4282
[69] https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad
[70] https://learn.microsoft.com/en-us/windows/win32/secauthz/access-control-lists
[71] https://learn.microsoft.com/en-us/windows/win32/secauthz/audit-generation
[72] https://medium.com/@boutnaru/windows-security-access-token-81cd00000c64
[73] https://medium.com/@boutnaru/windows-security-privileges-b8fe18cf3d5a
[74] https://jeffpar.github.io/kbarchive/kb/188/Q188855/
[75] https://learn.microsoft.com/en-us/windows/win32/api/aclapi/nf-aclapi-setnamedsecurityinfow

Mandatory Integrity Control (MIC)

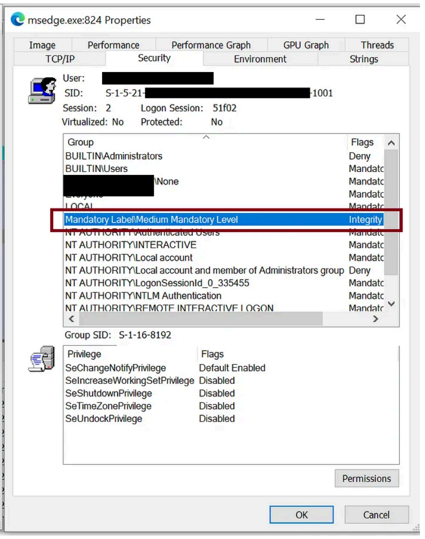
In general, "Mandatory Integrity Control" (MIC) has been added to Windows from Vista for adding support of MAC (Mandatory Access Control) to running processes[76]. This is done using a new attribute called "Integrity Level" (IL). MIC is designed to control access to securable objects[77]. The mechanism works in conjunction with DACL[78]. It is important to know that MIC evaluates access before the access check is made versus the object's DACL, and itself is implemented as ACEs (Access Control Entries) using special SIDs[79].

Moreover, each security principal[80] and any securable object is marked with an integrity level which is aimed at determining their level of access/protection. In Windows we have different integrity levels: "untrusted" (S-1-16-0), "low" (S-1-16-4096), "medium" (S-1-16-8192), "high" (S-1-16-12288) and "system" (S-1-16-16384). By default, standard users are given an integrity level of "medium" while elevated users get "high". Also, objects which lack an integrity level are treated as "medium"[81].

Lastly, the integrity level SIDs (as shown above) are stored in the SACL[82] of the secure object[83]. The Windows security policy states that a process can't interact with another process that has a higher integrity level[84], due to that it is also used by different sandbox implementations (like with Web Browsers). By the way, the integrity level is stored in the access token[85] of a process/thread - as shown in the screenshot below.



```
msedge.exe:1608 Properties

Image | Performance | Performance Graph | GPU Graph | Threads
TCP/IP | Security | Environment | Job | Strings

User:      [redacted]
SID:       S-1-5-21-[redacted]-1001
Session:   2        Logon Session:  51f02
Virtualized: No     Protected:      No

Group                                                      Flags
BUILTIN\Administrators                                     Deny
BUILTIN\Users                                              Mandatc
BUILTIN\Users                                              Mandatc
[redacted]\None                                            Deny
Everyone                                                   Mandatc
Everyone                                                   Mandatc
LOCAL                                                      Deny
Mandatory Label\Low Mandatory Level                        Integrity
NT AUTHORITY\Authenticated Users                           Deny
NT AUTHORITY\INTERACTIVE                                   Mandatc
NT AUTHORITY\Local account                                 Deny
NT AUTHORITY\Local account and member of Administrators group Deny
NT AUTHORITY\LogonSessionId_0_335455                       Mandatc

Group SID:  S-1-16-4096

Privilege                  Flags
SeChangeNotifyPrivilege    Default Enabled

Permissions
OK   Cancel
```



```
msedge.exe:824 Properties

Image | Performance | Performance Graph | GPU Graph | Threads
TCP/IP | Security | Environment | Strings

User:      [redacted]
SID:       S-1-5-21-[redacted]-1001
Session:   2        Logon Session:  51f02
Virtualized: No     Protected:      No

Group                                                      Flags
BUILTIN\Administrators                                     Deny
BUILTIN\Users                                              Mandatc
[redacted]\None                                            Mandatc
                                                           Mandatc
LOCAL                                                      Mandatc
Mandatory Label\Medium Mandatory Level                     Integrity
NT AUTHORITY\Authenticated Users                           Mandatc
NT AUTHORITY\INTERACTIVE                                   Mandatc
NT AUTHORITY\Local account                                 Mandatc
NT AUTHORITY\Local account and member of Administrators group Deny
NT AUTHORITY\LogonSessionId_0_335455                       Mandatc
NT AUTHORITY\NTLM Authentication                           Mandatc
NT AUTHORITY\REMOTE INTERACTIVE LOGON                      Mandatc

Group SID:  S-1-16-8192

Privilege                        Flags
SeChangeNotifyPrivilege          Default Enabled
SeIncreaseWorkingSetPrivilege    Disabled
SeShutdownPrivilege              Disabled
SeTimeZonePrivilege              Disabled
SeUndockPrivilege                Disabled

Permissions
OK   Cancel
```

[76] https://learn.microsoft.com/en-us/windows/win32/secauthz/mandatory-integrity-control
[77] https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad
[78] https://medium.com/@boutnaru/the-windows-security-journey-dacl-discretionary-access-control-list-c74545e472ec
[79] https://medium.com/@boutnaru/windows-security-sid-security-identifier-d5a27567d4e5
[80] https://medium.com/@boutnaru/windows-security-sid-security-identifier-d5a27567d4e5
[81] https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation/integrity-levels
[82] https://medium.com/@boutnaru/the-windows-security-journey-sacl-system-access-control-list-32488dcc80d7
[83] https://learn.microsoft.com/en-us/windows/win32/secauthz/mandatory-integrity-control
[84] https://en.wikipedia.org/wiki/Mandatory_Integrity_Control
[85] https://medium.com/@boutnaru/windows-security-access-token-81cd00000c64

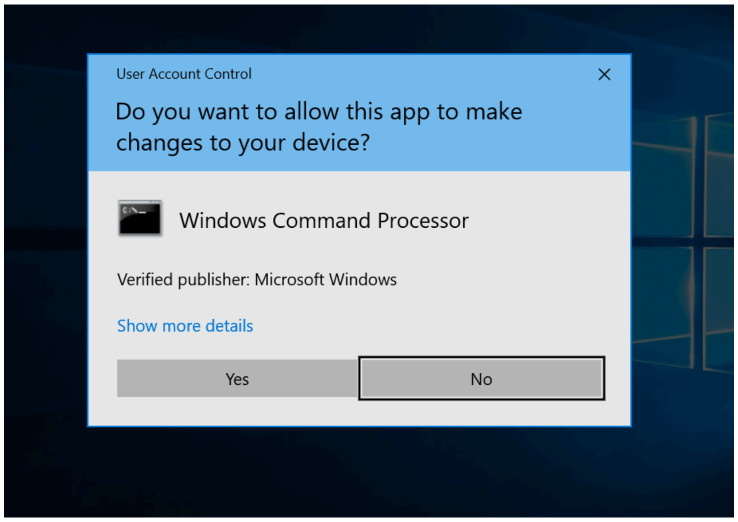
UAC (User Account Control)

The goal of UAC (User Account Control) is to reduce the risk of malware by limiting the ability of malicious code from running with administrator permissions. When UAC is used an application the requests an access token[86] with administrator permissions must prompt the user for consent[87] - as shown in the screenshot below.

UAC (User Account Control) provides MAC (Mandatory Access Control) which was introduced as part of Windows Vista/Server 2008. Together with UIPI[88] UAC is used to isolate between applications with the same user on the same session. When a user tries to perform an operation that requires admin access it will trigger UAC (if it's enabled). Examples of such operations (but not limited to) are: executing an application as an administrator, changing system-wide settings, installing a device driver, changing UAC settings, configuring windows update, opening the registry editor, changing power setting and turning on/of Windows features[89].

Moreover, when UAC is enabled when an administrator logs on to a system two separate access tokens are created (standard access token and administrator access token). The difference between them is that the administrative privileges and SIDs are removed from the standard one[90].

Lastly, UAC is composed of several technologies in order to provide its capabilities, among them are: file and registry virtualization, same desktop elevation, filtered token, UIPI, protected internet explorer and installer detection[91].


User Account Control

Do you want to allow this app to make changes to your device?

Windows Command Processor

Verified publisher: Microsoft Windows

[86] https://medium.com/@boutnaru/windows-security-access-token-81cd00000c64
[87] https://medium.com/@boutnaru/the-windows-process-journey-consent-exe-consent-ui-for-administrative-applications-d8e6976e8e40
[88] https://medium.com/@boutnaru/windows-security-user-interface-privilege-isolation-uipi-db790ad173eb
[89] https://en.wikipedia.org/wiki/User_Account_Control
[90] https://learn.microsoft.com/en-us/windows/security/application-security/application-control/user-account-control/how-it-works
[91] https://learn.microsoft.com/en-us/troubleshoot/windows-server/windows-security/disable-user-account-control

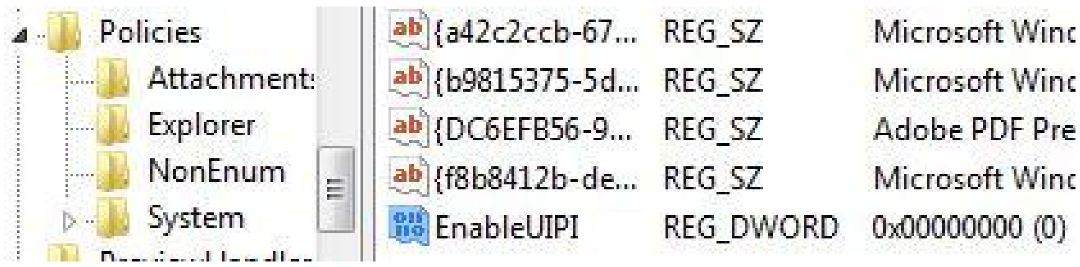
User Interface Privilege Isolation (UIPI)

User Interface Privilege Isolation (UIPI) was introduced in Windows 2008/Vista with the goal of mitigating “Shatter Attacks”. Those types of attacks leverage the Windows’s message passing system which can be used to inject arbitrary commands/code to any application/service running in the same session, those we are using a “message loop”[92].

UIPI allows isolating processes running as a full administrator from processes running as an account with lower permissions than an administrator on the same interactive desktop. UIPI is specific to the windowing/graphic subsystem (aka Windows USER). Thus, a process with lower privileges can’t perform operations on a process with higher privileges like: DLL injection, thread hooks for attaching, journal hooks for attaching, use window messages API (SendMessage/PostMessage) and more[93].

However, there are still resources that are shared between processes at different privilege levels like: clipboard, global atom table, desktop window and the desktop heap read-only shared memory. Also, painting on a screen is not controlled using UIPI, so a lower privilege application can paint over the surface region of a higher privilege application window - the GDI model does not allow control over painting surfaces[94].

Lastly, we can control the configuration of UIPI using the “EnableUIPI” value under the “HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System\” registry path - as shown in the screenshot below[95]. A value of “0” disables UIPI, and if the value is not present by default it means UIPI is enabled[96].



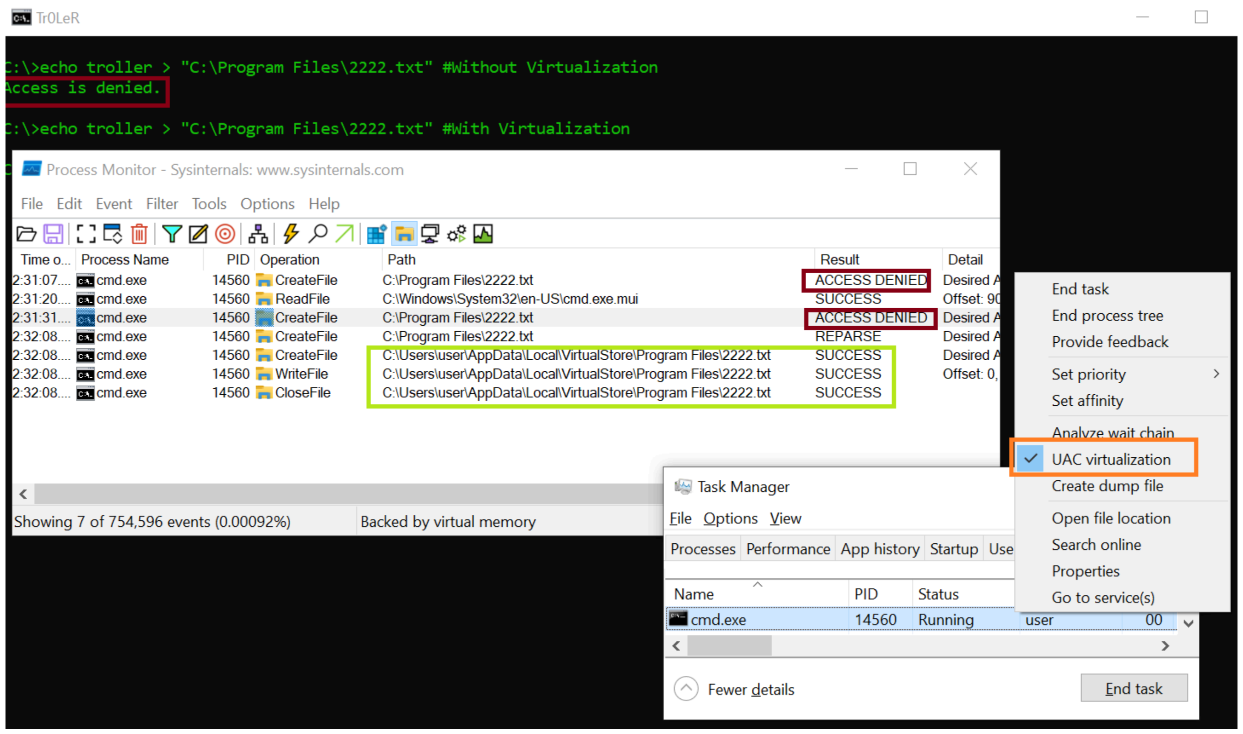
```
Policies            {a42c2ccb-67...   REG_SZ      Microsoft Winc
  Attachment:       {b9815375-5d...   REG_SZ      Microsoft Winc
  Explorer          {DC6EFB56-9...    REG_SZ      Adobe PDF Pre
  NonEnum           {f8b8412b-de...   REG_SZ      Microsoft Winc
  System            EnableUIPI        REG_DWORD   0x00000000 (0)
```

[92] https://www.slideserve.com/milek/shoot-the-messenger-win32-shatter-attacks-by-brett-moore
[93] https://learn.microsoft.com/en-us/previous-versions/aa905330(v=msdn.10)
[94] https://learn.microsoft.com/en-us/windows/win32/gdi/painting-and-drawing
[95] https://www.tipandtrick.net/fix-third-party-input-language-method-editor-ime-issues-in-ie-and-windows-vista-by-disabling-uipi/
[96] http://pferrie.epizy.com/papers/antidebug.pdf

File Virtualization

Due to security considerations (UAC enabled, it is also known as “UAC File Virtualization”) as of Windows Vista it does allow standard (non-administrator) users to access/manipulate folders (like “Program Files” and the “Windows” directory) or specific registry areas - as was allowed in previous Windows versions. However, because there are legacy applications which expect doing those operations Windows include “File and Registry Virtualization”[97].

Thus, if we have an application running with-out administrative permissions and it tries to write to “Program Files” it will be redirected to “C:\Users\%username%\AppData\Local\VirtualStore\Program Files\” and the operation will succeed[98]. Without the file virtualization the operation is going to fail - as shown in the screenshot below. By the way, I have enabled the virtualization on “cmd.exe” using “Task Manager”[99].



```
C:\>echo troller > "C:\Program Files\2222.txt" #Without Virtualization
Access is denied.

C:\>echo troller > "C:\Program Files\2222.txt" #With Virtualization
```

Time o...	Process Name	PID	Operation	Path	Result	Detail
2:31:07....	cmd.exe	14560	CreateFile	C:\Program Files\2222.txt	ACCESS DENIED	Desired A
2:31:20....	cmd.exe	14560	ReadFile	C:\Windows\System32\en-US\cmd.exe.mui	SUCCESS	Offset: 90
2:31:31....	cmd.exe	14560	CreateFile	C:\Program Files\2222.txt	ACCESS DENIED	Desired A
2:32:08....	cmd.exe	14560	CreateFile	C:\Program Files\2222.txt	REPARSE	Desired A
2:32:08....	cmd.exe	14560	CreateFile	C:\Users\user\AppData\Local\VirtualStore\Program Files\2222.txt	SUCCESS	Desired A
2:32:08....	cmd.exe	14560	WriteFile	C:\Users\user\AppData\Local\VirtualStore\Program Files\2222.txt	SUCCESS	Offset: 0,
2:32:08....	cmd.exe	14560	CloseFile	C:\Users\user\AppData\Local\VirtualStore\Program Files\2222.txt	SUCCESS	

Showing 7 of 754,596 events (0.00092%) Backed by virtual memory

Name	PID	Status		
cmd.exe	14560	Running	user	00

✓ UAC virtualization

[97] https://www.c-sharpcorner.com/uploadfile/GemingLeader/windows-file-and-registry-virtualization/
[98] https://flylib.com/books/en/2.955.1.34/1/
[99] https://www.experts-exchange.com/questions/28943516/What-is-UAC-Virtualization-in-the-Process-TASK-Manager.html