20-Access Control List (ACL) Abuse Primer

For security reasons, not **all users and computers in an AD environment can access all objects and files**. These types of permissions are controlled through A**ccess Control Lists (ACLs).** Posing a serious threat to the security posture of the domain, a slight misconfiguration to an ACL can leak permissions to other objects that do not need it.

Access Control List (ACL) Overview

In their simplest form, ACLs are lists that define **a) who has access to which asset/resource and b) the level of access they are provisioned.** The **settings themselves in an ACL are called `Access Control Entries` (`ACEs`)**. Each **ACE maps back to a user, group, or process (also known as security principals) and defines the rights granted to that principal**. **Every object has an ACL, but can have multiple ACEs because multiple security principals can access objects in AD. ACLs can also be used for auditing access within AD.**

1. **`Discretionary Access Control List` (`DACL)`**

   - defines which security principals are granted or denied access to an object. DACLs are made up of ACEs that either allow or deny access. When someone attempts to access an object, the system will check the DACL for the level of access that is permitted. If a DACL does not exist for an object, all who attempt to access the object are **granted full rights**. **If a DACL exists, but does not have any ACE entries specifying specific security settings, the system will deny access to all users, groups, or processes attempting to access it.**

ex:

Let’s say we have a file named `secret.txt` on a Windows machine, and we want to define who can access this file. We will use **DACL** and **ACE** to specify the permissions.

Steps:

1. **Create a DACL:** We create a **DACL** for the `secret.txt` file. Assume we have three users:
   - **User 1**: `UserA` (should only have read permission)
   - **User 2**: `UserB` (should only have write permission)
   - **User 3**: `UserC` (should be denied access completely)
2. **Add ACEs to the DACL:**
   - **ACE 1**: Add an ACE that allows `UserA` to only read the file.

    - **Allow**: `UserA` - Read
  - **ACE 2**: Add an ACE that allows `UserB` to only write to the file.
    - **Allow**: `UserB` - Write
  - **ACE 3**: Add an ACE that denies `UserC` access to the file.
    - **Deny**: `UserC` - Full Control

3. **How the DACL and ACE work:**
   - When **UserA** tries to access `secret.txt`, they will be allowed to read the file because the ACE grants them **Read** permission.
   - When **UserB** tries to access `secret.txt`, they will be allowed to write to the file because the ACE grants them **Write** permission.
   - When **UserC** tries to access `secret.txt`, they will be denied access entirely because there is an ACE that denies them access.

How this works on Windows:

You can use the **icacls** command to modify the **DACL** and **ACE** settings on the file.

Example Commands:

- To grant read permission for `UserA` on `secret.txt`:
  `icacls secret.txt /grant UserA:(R)`
- To grant write permission for `UserB` on `secret.txt`:
  `icacls secret.txt /grant UserB:(W)`
- To deny all access for `UserC` on `secret.txt`:
  `icacls secret.txt /deny UserC:(F)`

Final Outcome:

- **UserA** will be able to read the file only.
- **UserB** will be able to write to the file only.
- **UserC** will be completely denied access to the file.

Summary:

In this example, the **DACL** and **ACE** define who can access the `secret.txt` file and what actions they can perform. If the DACL contains ACEs allowing or denying access, the system applies these settings when a user attempts to access the object.

1. **`System Access Control Lists`** **(`SACL)`**

   - allow administrators to log access attempts made to secured objects. the same DACL but for logs ex:

Example Scenario:

Let's say you have a file named `securefile.txt`, and you want to log any access attempts to this file, whether the access is successful or denied. By configuring a **SACL** on `securefile.txt`, you can track who tries to access the file and whether the access was allowed or blocked.

Steps to Set Up SACL:

1. **Define the SACL:**
   - You create a **SACL** for `securefile.txt` to log access attempts.
   - You can configure **ACE** entries in the **SACL** to specify the type of events to be logged, such as successful or failed access.
2. **Set ACE (Access Control Entry):**
   - **ACE 1:** Track **failed access attempts** by users without proper permissions (denied access).
   - **ACE 2:** Track **successful access attempts** for users with the right permissions.
3. **How the SACL Works:**
   - If **UserA** has permission to read `securefile.txt`, and they try to access it, the system will log this **successful access attempt** in the Event Log.
   - If **UserB** tries to access the file but doesn't have permission, the system will log this as a **failed access attempt**.

Using icacls to Configure SACL:

In Windows, you can configure **SACL** for a file using the **icacls** command.

Example Commands:

- To configure **SACL** to log **failed access attempts** for a user:
  `icacls securefile.txt /setaudit UserA:(D,FA)`
  - `(D)` means the access is **denied** for **UserA**.
  - `(FA)` specifies that **failed access attempts** will be audited and logged in the Event Log.
- To configure **SACL** to log **successful access attempts** for a user:
  `icacls securefile.txt /setaudit UserA:(R)`
  - `(R)` specifies **read** permission, and successful read attempts will be logged.

Result of SACL Configuration:

1. **UserA** tries to access `securefile.txt`:
   - If they have the proper permissions, the access will be logged as **successful** in the Event Log.

2. **UserB** tries to access `securefile.txt`:
    - If they do not have permission, the access will be logged as a **failed attempt**.

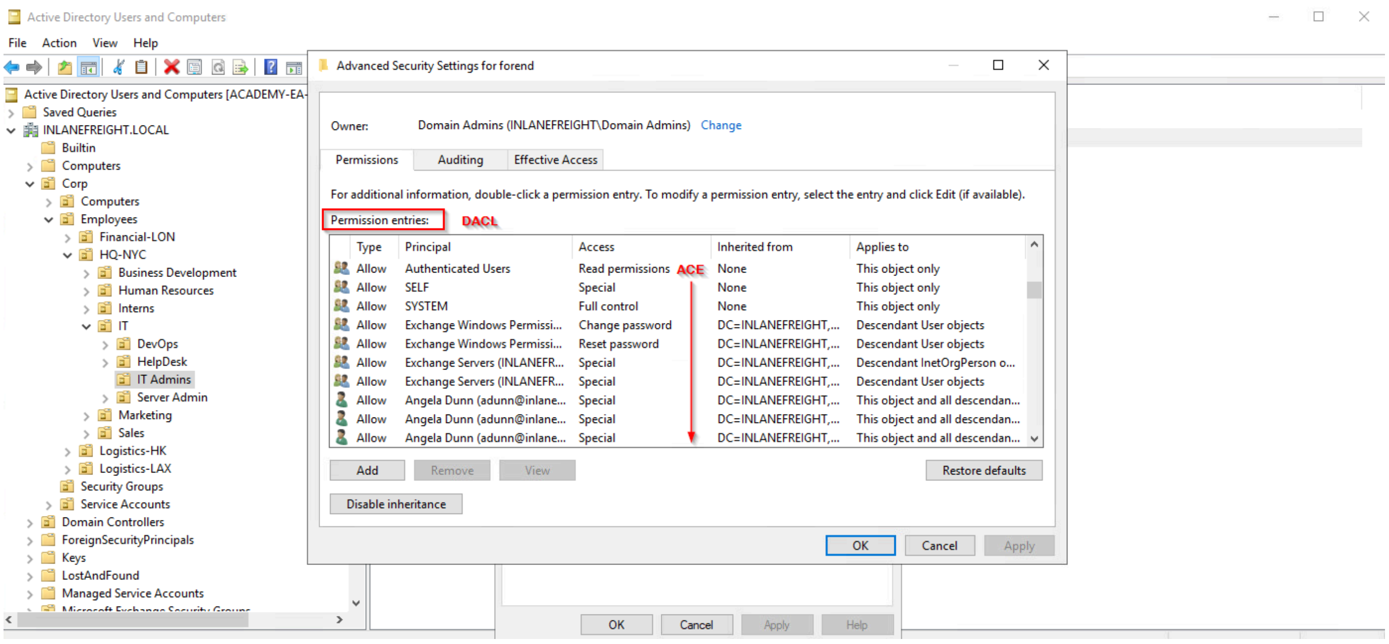
Benefits of SACL:

- **Security Monitoring:** SACL provides administrators with a way to monitor access to sensitive objects.
- **Detect Unauthorized Access:** By tracking failed access attempts, administrators can identify potential security breaches.
- **Auditing and Compliance:** SACL allows for thorough auditing, ensuring that administrators can review access logs to meet compliance standards.

Summary:

SACL is used to **log access attempts** to secured objects. It records both **successful** and **failed** access attempts based on the configuration of **ACE** entries. This helps administrators monitor who is trying to access a file or object and whether the access was allowed or denied, aiding in security monitoring and compliance auditing.

Viewing forend's ACL



Owner: Domain Admins (INLANEFREIGHT\Domain Admins) Change

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

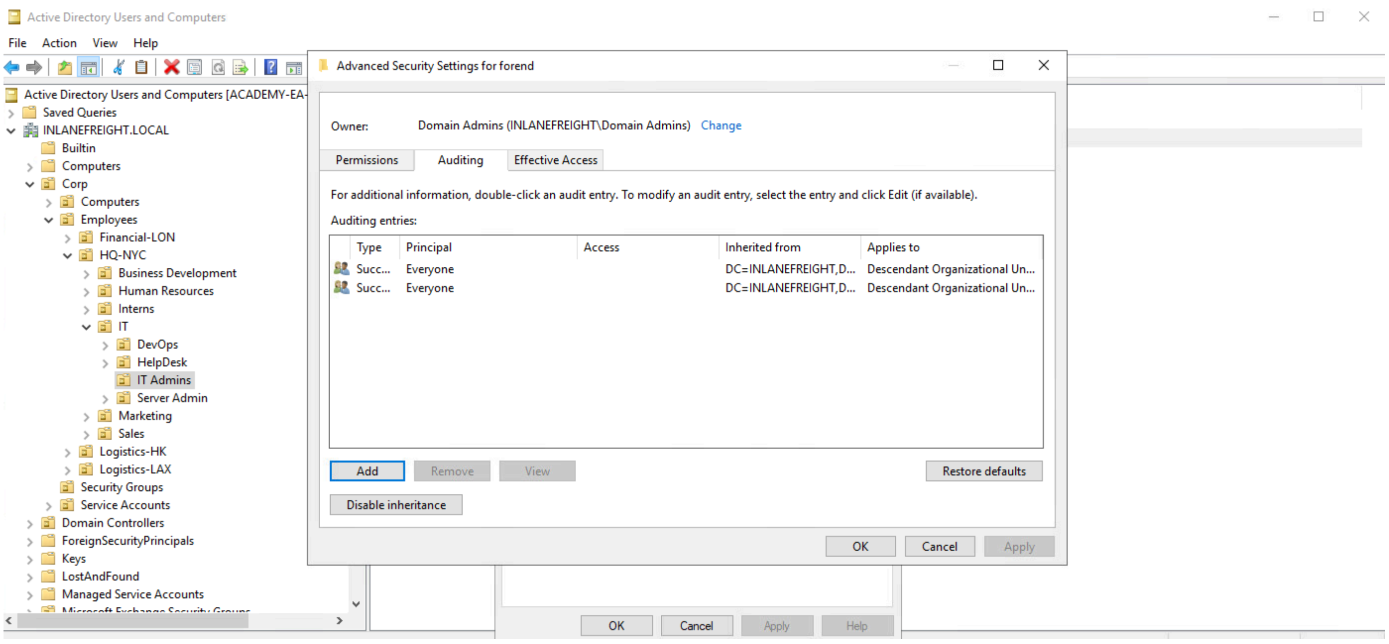
Permission entries:

Type	Principal	Access	Inherited from	Applies to
Allow	Authenticated Users	Read permissions	None	This object only
Allow	SELF	Special	None	This object only
Allow	SYSTEM	Full control	None	This object only
Allow	Exchange Windows Permissi...	Change password	DC=INLANEFREIGHT,...	Descendant User objects
Allow	Exchange Windows Permissi...	Reset password	DC=INLANEFREIGHT,...	Descendant User objects
Allow	Exchange Servers (INLANEFR...	Special	DC=INLANEFREIGHT,...	Descendant InetOrgPerson o...
Allow	Exchange Servers (INLANEFR...	Special	DC=INLANEFREIGHT,...	Descendant User objects
Allow	Angela Dunn (adunn@inlane...	Special	DC=INLANEFREIGHT,...	This object and all descendan...
Allow	Angela Dunn (adunn@inlane...	Special	DC=INLANEFREIGHT,...	This object and all descendan...
Allow	Angela Dunn (adunn@inlane...	Special	DC=INLANEFREIGHT,...	This object and all descendan...

Active Directory Users and Computers [ACADEMY-EA-
- Saved Queries
- INLANEFREIGHT.LOCAL
  - Builtin
  - Computers
  - Corp
    - Computers
    - Employees
      - Financial-LON
      - HQ-NYC
        - Business Development
        - Human Resources
        - Interns
        - IT
          - DevOps
          - HelpDesk
          - IT Admins
          - Server Admin
        - Marketing
        - Sales
      - Logistics-HK
      - Logistics-LAX
    - Security Groups
    - Service Accounts
  - Domain Controllers
  - ForeignSecurityPrincipals
  - Keys
  - LostAndFound
  - Managed Service Accounts

The SACLs can be seen within the `Auditing` tab.

Viewing the SACLs through the Auditing Tab



Owner: Domain Admins (INLANEFREIGHT\Domain Admins) Change

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

Auditing entries:

Type	Principal	Access	Inherited from	Applies to
Succ...	Everyone		DC=INLANEFREIGHT,D...	Descendant Organizational Un...
Succ...	Everyone		DC=INLANEFREIGHT,D...	Descendant Organizational Un...

Access Control Entries (ACEs)

As stated previously, Access Control Lists (ACLs) contain ACE entries that name a user or group and the level of access they have over a given securable object. There are `three` main types of ACEs that can be applied to all securable objects in AD:

ACE	**Description**
`Access denied ACE`	Used within a DACL to show that a user or group is explicitly denied access to an object
`Access allowed ACE`	Used within a DACL to show that a user or group is explicitly granted access to an object
`System audit ACE`	Used within a SACL to generate audit logs when a user or group attempts to access an object. It records whether access was granted or not and what type of access occurred

Each ACE is made up of the following `four` components:

1. The security identifier (SID) of the user/group that has access to the object (or principal name graphically)
2. A flag denoting the type of ACE (access denied, allowed, or system audit ACE)
3. A set of flags that specify whether or not child containers/objects can inherit the given ACE entry from the primary or parent object
4. An access mask which is a 32-bit value that defines the rights granted to an object

access mask: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-dtyp/7a53f60e-e730-4dfe-bbe9-b21b62eb790b?redirectedfrom=MSDN

1. **Security Identifier (SID)**:

- A **Security Identifier (SID)** is a unique identifier assigned to a user or group within Windows. It is used to control access to system resources and objects.
- When a new user or group is created, a **SID** is assigned to them, and this SID is used by the system to identify who has access rights to certain objects.
- **Example**: If you have an "Administrators" group on your system, the **SID** for that group might look like `S-1-5-32-544`. This **SID** is unique for the "Administrators" group, and you would use this in Access Control Lists (ACLs) rather than using the group's name.

2. **ACE Type Flag**:
   - This flag defines the type of access control entry (ACE) in the Access Control List (ACL). There are three possible types of ACE:
     - **Access Allowed ACE**: Grants specific rights to a user or group (e.g., read, write, execute).
     - **Access Denied ACE**: Denies specific rights to a user or group (e.g., deny write access).
     - **System Audit ACE**: Logs access attempts (whether successful or unsuccessful) for auditing purposes.

   Example: If you have a file and want to allow the "Users" group to read the file, you would use an **Access Allowed ACE**. If you want to deny the "Guests" group access to the file, you would use an **Access Denied ACE**.
3. **Inheritance Flags** (Determining if Child Objects Can Inherit ACE):
   - These flags specify whether child objects (subfolders, files, etc.) can inherit the ACE from the parent object.
   - **Inheritance flags** control whether child objects can inherit permissions from the parent object.
     - **Allow Inheritance**: The child objects will inherit the same ACE permissions as the parent.
     - **Deny Inheritance**: The child objects will not inherit permissions from the parent.

   Example: If you set read permissions for the "Developers" group on the folder "ProjectFiles" and enable **Allow Inheritance**, then any subfolders or files within "ProjectFiles" will inherit the same read permissions. If you disable inheritance (**Deny Inheritance**), child objects will not inherit these permissions.
4. **Access Mask**:
   - An **Access Mask** is a 32-bit value that specifies the rights granted or denied to a user or group. It defines the allowed actions on an object (e.g., read, write, execute).
   - The **Access Mask** determines the types of operations that can be performed on an object, such as:
     - **Read**: Permission to read the contents of the object.
     - **Write**: Permission to modify the object.
     - **Execute**: Permission to execute the object if it's an executable file.
     - **Delete**: Permission to delete the object.

Example: If you have a folder and you want to grant the "Developers" group permissions to read and write to the folder, the **Access Mask** value might be something like `0x00120089`. This value indicates that "read", "write", and "execute" permissions are granted to the user or group.

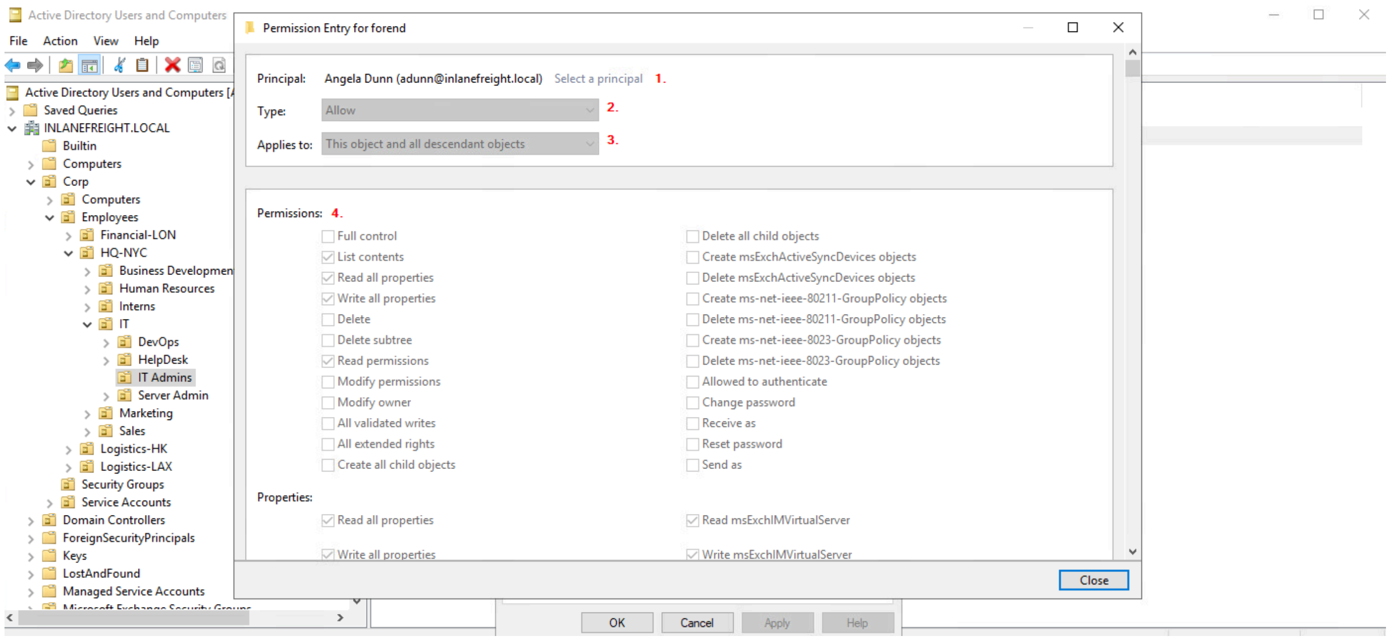
Complete Example in Practice:

- You have a folder called `ProjectFiles`.
- You want to allow the "Developers" group to read and write to the folder.
- You want to deny the "Guests" group access to the folder.
- You would set the ACE in the ACL as follows:
    - **SID**: The SID for the "Developers" group.
    - **ACE Type Flag**: **Access Allowed ACE** (to allow access).
    - **Inheritance Flag**: **Allow Inheritance** (so child objects inherit the same permissions).
    - **Access Mask**: A value like `0x00120089` (to allow read and write permissions).

With this setup, the "Developers" group will have read and write access to the "ProjectFiles" folder, and any subfolders or files inside "ProjectFiles" will inherit the same permissions. The "Guests" group will not have any access to the folder.

We can view this graphically in `Active Directory Users and Computers` (`ADUC`). In the example image below, we can see the following for the ACE entry for the user `forend`:

Viewing Permissions through Active Directory Users & Computers



Permission Entry for forend

Principal: Angela Dunn (adunn@inlanefreight.local) Select a principal 1.

Type: Allow 2.

Applies to: This object and all descendant objects 3.

Permissions: 4.

☐ Full control	☐ Delete all child objects
✓ List contents	☐ Create msExchActiveSyncDevices objects
✓ Read all properties	☐ Delete msExchActiveSyncDevices objects
✓ Write all properties	☐ Create ms-net-ieee-80211-GroupPolicy objects
☐ Delete	☐ Delete ms-net-ieee-80211-GroupPolicy objects
☐ Delete subtree	☐ Create ms-net-ieee-8023-GroupPolicy objects
✓ Read permissions	☐ Delete ms-net-ieee-8023-GroupPolicy objects
☐ Modify permissions	☐ Allowed to authenticate
☐ Modify owner	☐ Change password
☐ All validated writes	☐ Receive as
☐ All extended rights	☐ Reset password
☐ Create all child objects	☐ Send as

Properties:

✓ Read all properties	✓ Read msExchIMVirtualServer
✓ Write all properties	✓ Write msExchIMVirtualServer

1. The security principal is Angela Dunn (adunn@inlanefreight.local)
2. The ACE type is `Allow`

3. Inheritance applies to the "This object and all descendant objects," meaning any child objects of the `forend` object would have the same permissions granted
4. The rights granted to the object, again shown graphically in this example

When access control lists are checked to determine permissions, they are checked from top to bottom until an access denied is found in the list.

Why are ACEs Important?

Attackers utilize ACE entries to either further access or establish persistence. These can be great for us as penetration testers as many organizations are unaware of the ACEs applied to each object or the impact that these can have if applied incorrectly. They cannot be detected by vulnerability scanning tools, and often go unchecked for many years, especially in large and complex environments. During an assessment where the client has taken care of all of the "low hanging fruit" AD flaws/misconfigurations, ACL abuse can be a great way for us to move laterally/vertically and even achieve full domain compromise. Some example Active Directory object security permissions are as follows. These can be enumerated (and visualized) using a tool such as BloodHound, and are all abusable with PowerView, among other tools:

- `ForceChangePassword` **abused with** `Set-DomainUserPassword`
- `Add Members` **abused with** `Add-DomainGroupMember`
- `GenericAll` **abused with** `Set-DomainUserPassword` **or** `Add-DomainGroupMember`
- `GenericWrite` **abused with** `Set-DomainObject`
- `WriteOwner` **abused with** `Set-DomainObjectOwner`
- `WriteDACL` **abused with** `Add-DomainObjectACL`
- `AllExtendedRights` **abused with** `Set-DomainUserPassword` **or** `Add-DomainGroupMember`
- `Addself` **abused with** `Add-DomainGroupMember`

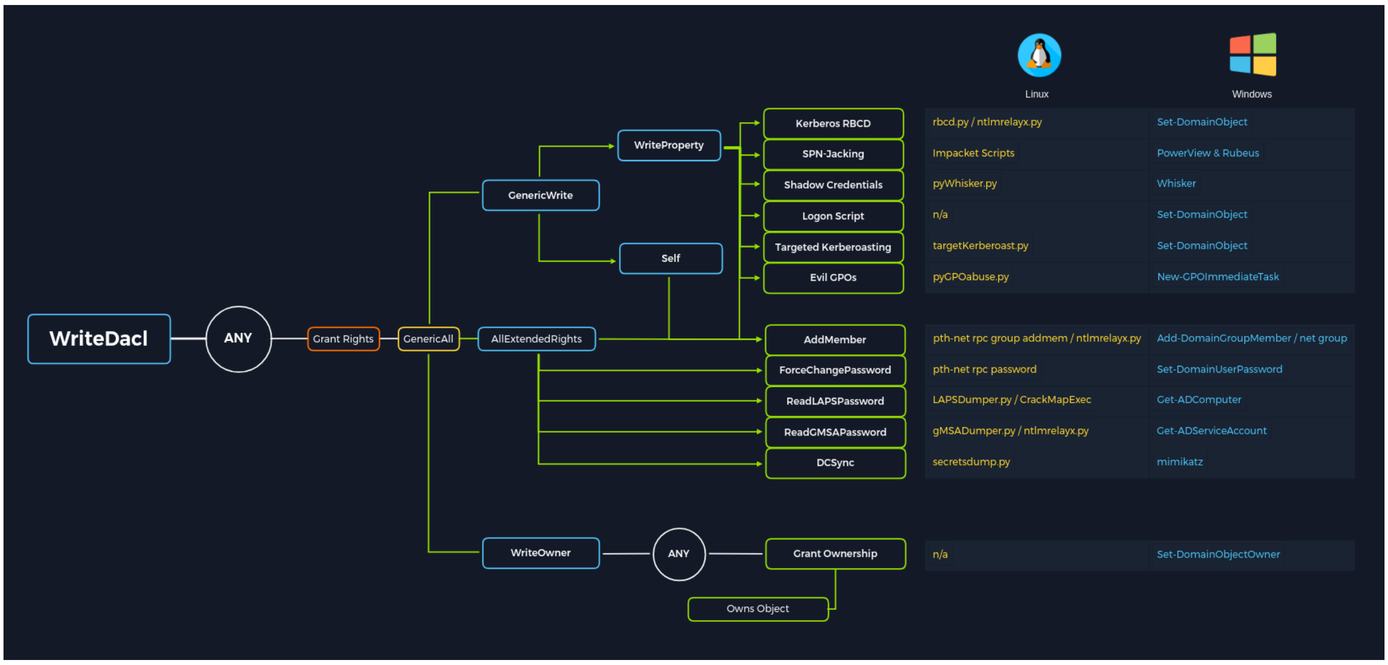
تعتبر الـ ACEs (وهي عناصر تحكم الوصول في **Active Directory**) من الثغرات التي يمكن للمهاجمين استغلالها بسهولة لتحقيق أهدافهم، مثل التحرك داخل الشبكة، الحصول على امتيازات أعلى، أو حتى السيطرة على النظام بالكامل. العديد من هذه الإعدادات يمكن أن تكون غير مرئية لأدوات الفحص التقليدية، وتظل غير مفحوصة لسنوات طويلة في بيئات **Active Directory** المعقدة، مما يجعلها هدفًا مثاليًا في اختبارات الاختراق.

In this module, we will cover enumerating and leveraging four specific ACEs to highlight the power of ACL attacks:

- ForceChangePassword - gives us the right to reset a user's password without first knowing their password (should be used cautiously and typically best to consult our client before resetting passwords).

- GenericWrite - gives us the right to write to any non-protected attribute on an object. If we have this access over a user, we could assign them an SPN and perform a Kerberoasting attack (which relies on the target account having a weak password set). Over a group means we could add ourselves or another security principal to a given group. Finally, if we have this access over a computer object, we could perform a resource-based constrained delegation attack which is outside the scope of this module.
- `AddSelf` - shows security groups that a user can add themselves to.
- GenericAll - this grants us full control over a target object. Again, depending on if this is granted over a user or group, we could modify group membership, force change a password, or perform a targeted Kerberoasting attack. If we have this access over a computer object and the Local Administrator Password Solution (LAPS) is in use in the environment, we can read the LAPS password and gain local admin access to the machine which may aid us in lateral movement or privilege escalation in the domain if we can obtain privileged controls or gain some sort of privileged access.

This graphic, adapted from a graphic created by Charlie Bromberg (Shutdown), shows an excellent breakdown of the varying possible ACE attacks and the tools to perform these attacks from both Windows and Linux (if applicable). In the following few sections, we will mainly cover enumerating and performing these attacks from a Windows attack host with mentions of how these attacks could be performed from Linux. A later module specifically on ACL Attacks will go much further in-depth on each of the attacks listed in this graphic and how to perform them from Windows and Linux.



	Linux	Windows
Kerberos RBCD	rbcd.py / ntlmrelayx.py	Set-DomainObject
SPN-Jacking	Impacket Scripts	PowerView & Rubeus
Shadow Credentials	pyWhisker.py	Whisker
Logon Script	n/a	Set-DomainObject
Targeted Kerberoasting	targetKerberoast.py	Set-DomainObject
Evil GPOs	pyGPOabuse.py	New-GPOImmediateTask
AddMember	pth-net rpc group addmem / ntlmrelayx.py	Add-DomainGroupMember / net group
ForceChangePassword	pth-net rpc password	Set-DomainUserPassword
ReadLAPSPassword	LAPSDumper.py / CrackMapExec	Get-ADComputer
ReadGMSAPassword	gMSADumper.py / ntlmrelayx.py	Get-ADServiceAccount
DCSync	secretsdump.py	mimikatz
Grant Ownership	n/a	Set-DomainObjectOwner

ACL Attacks in the Wild

We can use ACL attacks for:

- Lateral movement
- Privilege escalation

- Persistence

Some common attack scenarios may include:

Attack	Description
`Abusing forgot password permissions`	Help Desk and other IT users are often granted permissions to perform password resets and other privileged tasks. If we can take over an account with these privileges (or an account in a group that confers these privileges on its users), we may be able to perform a password reset for a more privileged account in the domain.
`Abusing group membership management`	It's also common to see Help Desk and other staff that have the right to add/remove users from a given group. It is always worth enumerating this further, as sometimes we may be able to add an account that we control into a privileged built-in AD group or a group that grants us some sort of interesting privilege.
`Excessive user rights`	We also commonly see user, computer, and group objects with excessive rights that a client is likely unaware of. This could occur after some sort of software install (Exchange, for example, adds many ACL changes into the environment at install time) or some kind of legacy or accidental configuration that gives a user unintended rights. Sometimes we may take over an account that was given certain rights out of convenience or to solve a nagging problem more quickly.

```
1. ForceChangePassword abused with Set-DomainUserPassword
الشرح: إذا كان للمهاجم إذن ForceChangePassword على حساب مستخدم، فهذا يعني
أنه يمكنه إجبار المستخدم على تغيير كلمة المرور في المرة التالية التي يقوم
فيها بتسجيل الدخول. باستخدام هذا الإذن مع الأمر Set-DomainUserPassword، يمكن
للمهاجم تغيير كلمة مرور حساب المستخدم إلى كلمة مرور يعرفها، وبالتالي الحصول
على وصول كامل إلى الحساب.
التهديد: يتمكن المهاجم من فرض كلمة مرور جديدة على حسابات المستخدمين، مما
يسمح له بتغيير الحسابات المهمة مثل حسابات المديرين.
2. Add Members abused with Add-DomainGroupMember
الشرح: إذا كان للمهاجم إذن Add Members في مجموعة، فهذا يعني أنه يمكنه إضافة
مستخدمين إلى مجموعة معينة في Active Directory. باستخدام الأمر Add-
DomainGroupMember، يمكن للمهاجم إضافة نفسه إلى مجموعة مثل Domain Admins أو
أي مجموعة ذات امتيازات إدارية.
التهديد: بمجرد إضافة نفسه إلى مجموعة Domain Admins، يصبح المهاجم لديه
امتيازات عالية، مما يمكنه من التحكم في النظام بالكامل.
3. GenericAll abused with Set-DomainUserPassword or Add-DomainGroupMember
الشرح: إذن GenericAll يعطي المهاجم جميع أنواع الأذونات على الكائنات في Active
Directory. يمكن للمهاجم استخدام هذا الإذن لتغيير كلمة مرور مستخدم باستخدام
الأمر Set-DomainUserPassword أو إضافة نفسه إلى مجموعة ذات امتيازات عالية
باستخدام Add-DomainGroupMember.
التهديد: هذا الإذن يعتبر من أقوى الأذونات، حيث يمنح المهاجم السيطرة الكاملة
```

على الكائنات في AD، بما في ذلك تغيير كلمات المرور أو إضافة نفسه إلى مجموعات
ذات امتيازات عالية.
4. GenericWrite abused with Set-DomainObject
الشرح: إذن GenericWrite يسمح للمهاجم بتعديل خصائص الكائنات في Active
Directory. باستخدام هذا الإذن مع الأمر Set-DomainObject، يمكن للمهاجم تغيير
خصائص كائنات AD مثل تعديل بيانات المستخدمين أو تغيير قيم الأذونات.
التهديد: يمكن للمهاجم استخدام هذا الإذن لتغيير معلومات المستخدمين في Active
Directory، مثل الاسم أو البريد الإلكتروني أو غيرها من المعلومات الحساسة.
5. WriteOwner abused with Set-DomainObjectOwner
الشرح: إذن WriteOwner يسمح للمهاجم بتغيير مالك الكائنات في Active Directory.
باستخدام هذا الإذن مع الأمر Set-DomainObjectOwner، يمكن للمهاجم تغيير مالك
الكائن إلى حسابه الخاص، مما يمنحه التحكم الكامل على الكائن.
التهديد: هذا الإذن يمكن أن يمنح المهاجم السيطرة الكاملة على الكائنات في AD،
مما يجعله قادرًا على تغيير الأذونات أو حتى إزالة أي حسابات غير مرغوب فيها.
6. WriteDACL abused with Add-DomainObjectACL
الشرح: إذن WriteDACL يسمح للمهاجم بتعديل DACL (قائمة التحكم في الوصول)
للكائن التي تحدد من يمكنه الوصول إلى الكائنات في Active Directory. باستخدام
هذا الإذن مع الأمر Add-DomainObjectACL، يمكن للمهاجم إضافة أذونات جديدة للوصول
إلى الكائنات.
التهديد: إذا تمكن المهاجم من تعديل الـ DACL، فيمكنه إضافة أو إزالة أذونات
الوصول لأي شخص آخر، بما في ذلك نفسه، مما يتيح له الوصول الكامل إلى الكائنات
في AD.
7. AllExtendedRights abused with Set-DomainUserPassword or Add-
DomainGroupMember
الشرح: AllExtendedRights هو مجموعة من الأذونات التي تسمح للمهاجم بالتحكم في
خصائص خاصة تتعلق بحسابات المستخدمين أو المجموعات. باستخدام هذه الأذونات مع
Set-DomainUserPassword أو Add-DomainGroupMember، يمكن للمهاجم تغيير كلمات
مرور المستخدمين أو إضافة نفسه إلى مجموعة ذات امتيازات عالية.
التهديد: المهاجم يمكنه تغيير كلمات المرور أو إضافة نفسه إلى مجموعات تحتوي
على امتيازات عالية، مثل Domain Admins، مما يعطيه وصولاً غير محدود إلى النظام.
8. Addself abused with Add-DomainGroupMember
الشرح: إذن Addself يسمح للمهاجم بإضافة نفسه إلى أي مجموعة في Active
Directory. باستخدام الأمر Add-DomainGroupMember، يمكن للمهاجم إضافة نفسه إلى
مجموعة ذات امتيازات عالية مثل Domain Admins.
التهديد: بعد إضافة نفسه إلى مجموعة Domain Admins، يصبح للمهاجم وصول كامل إلى
جميع الموارد في Active Directory، مما يتيح له إجراء تغييرات كبيرة في النظام.