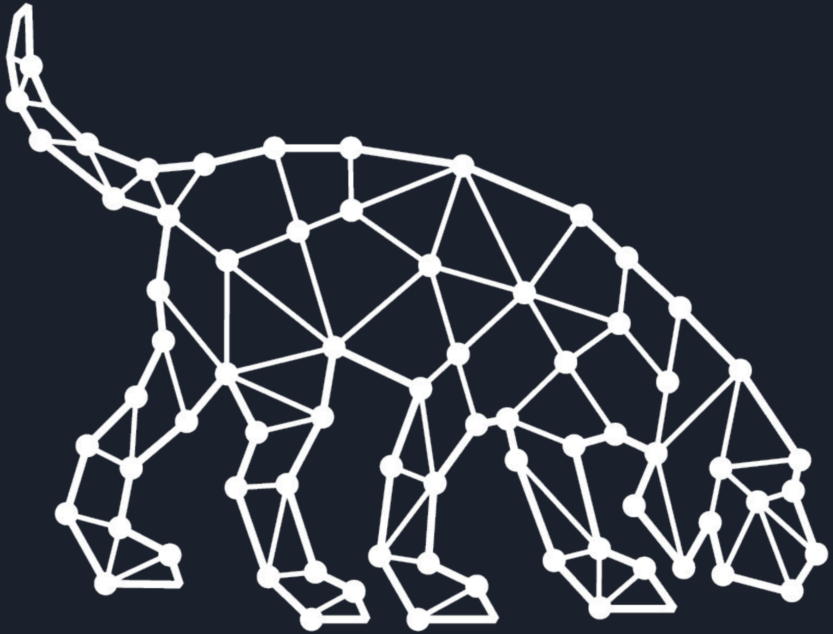
BloodHound Unleashed



Esteban Rodriguez

Frank Scarpella

Esteban Rodriguez

- Senior Security Consultant at TrustedSec
- Blogs at https://www.n00py.io
- Twitter: @n00py1
- Github: https://github.com/n00py

Frank Scarpella

Principal Security Consultant - SwAG

Twitter: @ninjastyle82

GitHub: github.com/ninjastyle82

Secureworks® | Adversary Group

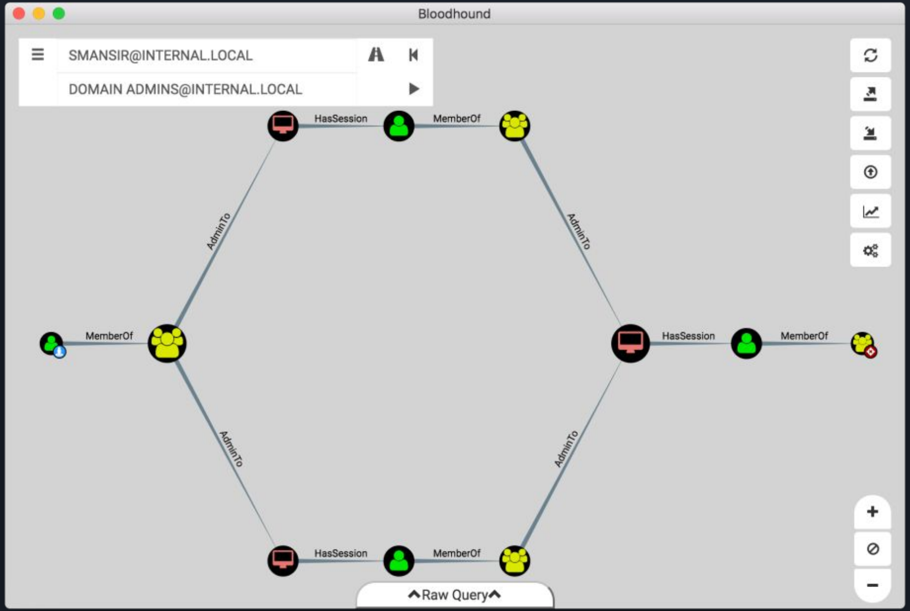
What is BloodHound?

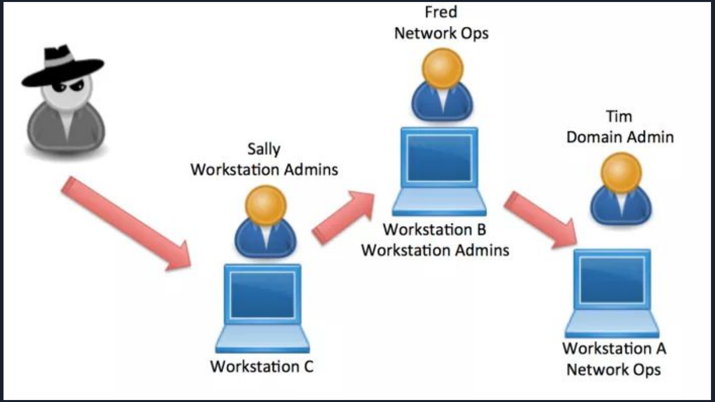
- Released in 2016 at DEF CON 24 by Veris Group’s ATD Team
  - @_wald0 - Andy Robbins
  - @CptJesus - Rohan Vazarkar
  - @harmj0y - Will Schroeder
- Uses Graph Theory
  - Vertices (Nodes) - Objects like Users, Groups, Computers, etc
  - Edges (Relationships) - Relationships between objects
  - Paths - Connecting Objects for Privilege Escalation
- Ingestor
  - Collects data from Active Directory and saves JSON data
- Backend database
  - Neo4j graph database - stores nodes and relationship data
  - Uses Cypher query language
- Frontend application
  - JavaScript/HTML application for drawing graphs, importing data, and performing queries

A Brief History

- All About derivative local admin
  - Who is Admin to What?
  - Who is Logged on Where?

https://i1.wp.com/wald0.com/wp-content/uploads/2016/08/Screen-Shot-2016-08-29-at-6.31.37-PM.png



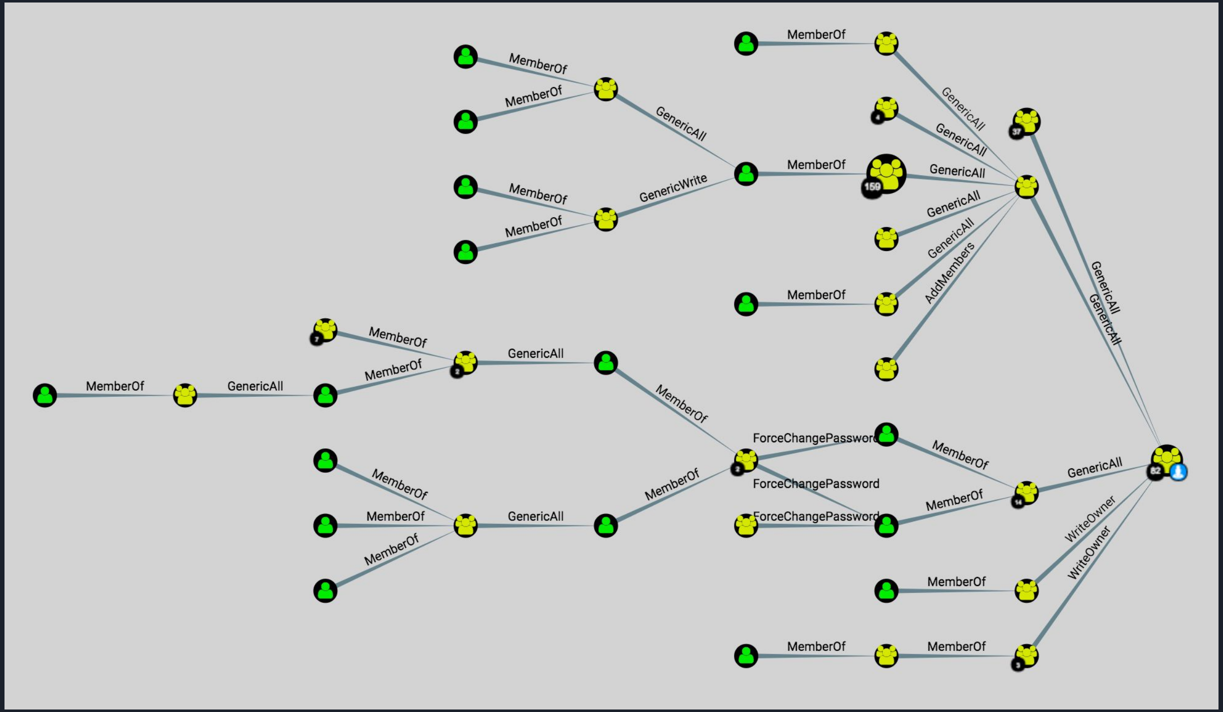


https://wald0.com/?p=68
https://www.slideshare.net/AndyRobbins3/six-degrees-of-domain-admin-bloodhound-at-def-con-24

https://www.sixdub.net/?p=591 (Broken Link)
https://sixdub.medium.com/derivative-local-admin-cdd09445aac8

BloodHound 1.3 - The ACL Attack Path Update

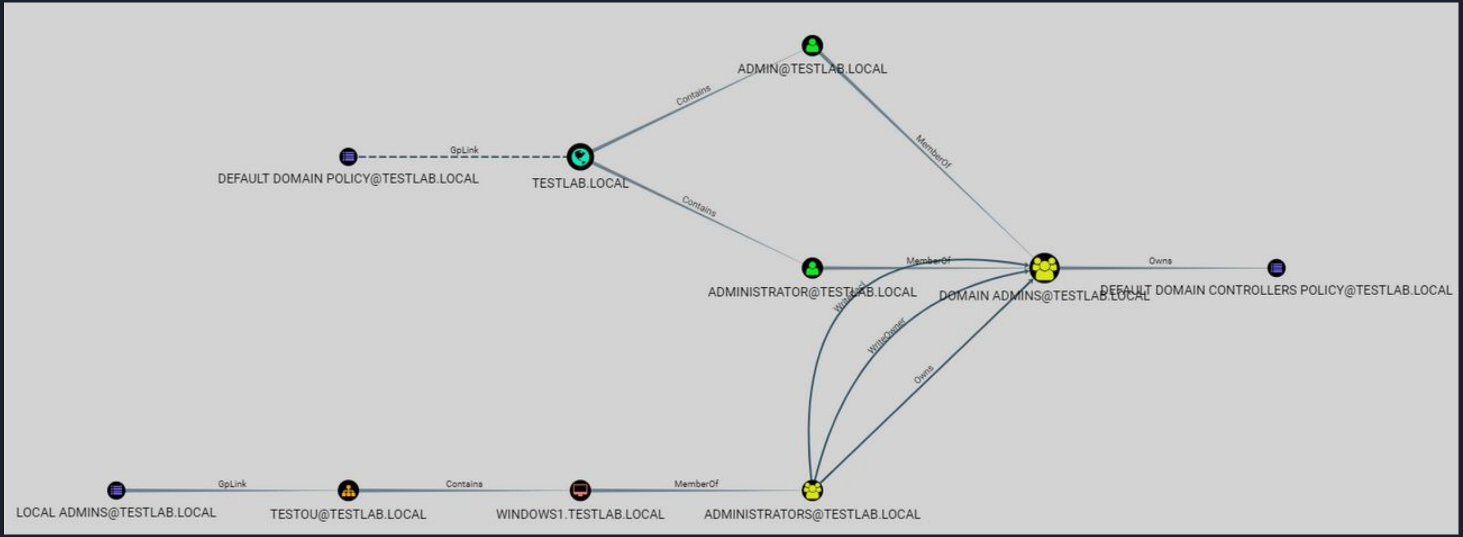
- Completely game changing
- Tons of new attack paths
- https://wald0.com/?p=112



https://i0.wp.com/wald0.com/wp-content/uploads/2017/05/TransitiveControllers.png

BloodHound 1.5 - The Container Update

- Added Objects/Edges for Containers and GPOs
- https://posts.specterops.io/bloodhound-1-5-the-container-update-fdf1ed2ad9da


ADMIN@TESTLAB.LOCAL
Contains
MemberOf
GpLink
DEFAULT DOMAIN POLICY@TESTLAB.LOCAL
TESTLAB.LOCAL
Contains
ADMINISTRATOR@TESTLAB.LOCAL
MemberOf
DOMAIN ADMINS@TESTLAB.LOCAL
Owns
DEFAULT DOMAIN CONTROLLERS POLICY@TESTLAB.LOCAL
WriteDacl
WriteOwner
Owns
GpLink
Contains
MemberOf
LOCAL ADMINS@TESTLAB.LOCAL
TESTOU@TESTLAB.LOCAL
WINDOWS1.TESTLAB.LOCAL
ADMINISTRATORS@TESTLAB.LOCAL

https://miro.medium.com/max/720/0*OcD5QlwNIcp_wAru.png

Ingestors

S+

- SharpHound
  - https://github.com/BloodHoundAD/SharpHound
  - The gold standard, use it if you can
  - Supports session looping
  - Cons: AV = big mad

S

- BloodHound.py
  - https://github.com/fox-it/BloodHound.py
  - Almost just as good
  - Sometimes has memory issues on large orgs
  - Python

A

- RustHound
  - https://github.com/OPENCYBER-FR/RustHound
  - Pro: Single Executable, no dependencies
  - Con: Missing some core functionality, such as session collection

B

- AD Explorer Snapshot
  - https://github.com/c3c/ADExplorerSnapshot.py
  - Pro: AD Explorer is a Microsoft Signed Binary
  - Con: Only collects "DCOnly" information
  - More network intensive

B

- ldif2bloodhound
  - https://github.com/SySS-Research/ldif2bloodhound
  - Convert an LDIF file to JSON files ingestible by BloodHound
  - LDIF file created with ldapsearch
  - Equivalent to DCOnly

C

- SilentHound
  - https://github.com/layer8secure/SilentHound
  - One LDAP query: `(objectClass=*)`

C

- Ldapdomaindump to BloodHound
  - Updated ldapdomaindump converter (BH 4.0)
  - https://github.com/blurbdust/ldd2bh

Collection Methods

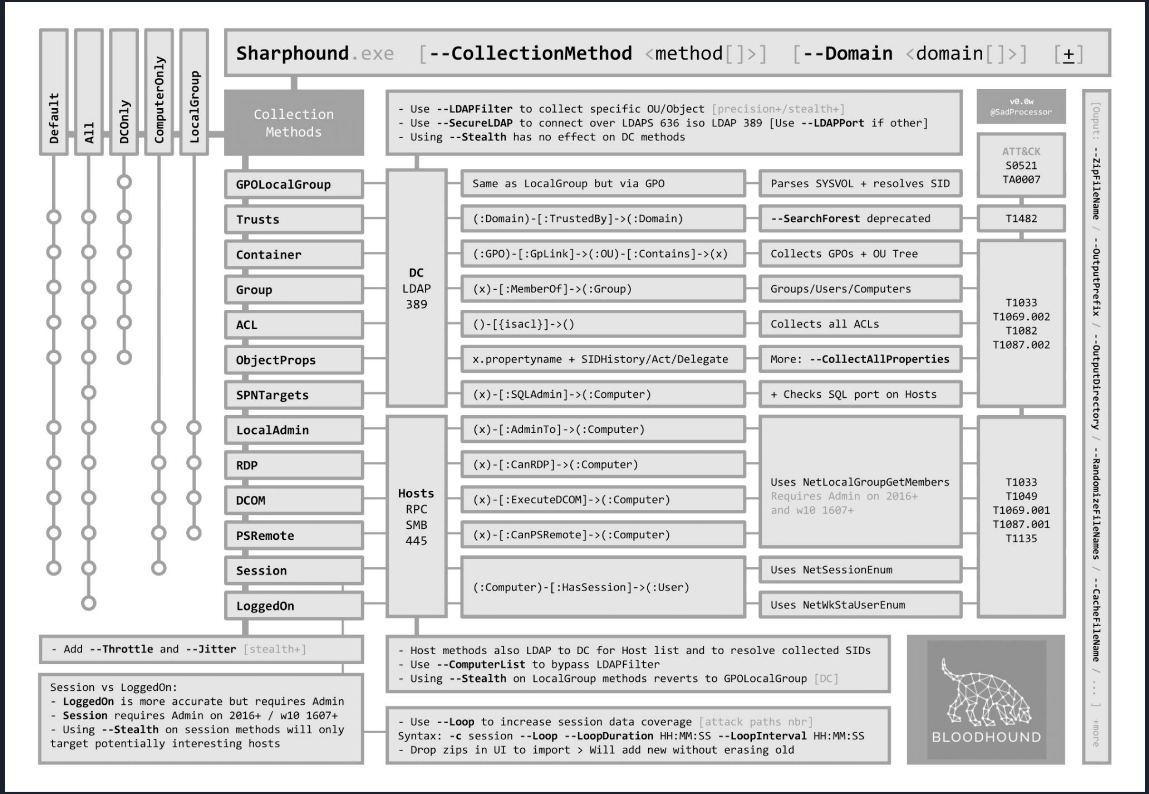
All - Collect all data except GPOLocalGroup

Default - Collects ACL, Container, Group, LocalGroups, ObjectProps, Sessions, Trusts, SPNTargets (from source code, documentation conflicting)

DcOnly - Collects ACL, Container, Group, ObjectProps, Trusts, DCOnly, GPOLocalGroup (from source code, documentation conflicting)

LoggedOn - Collects session information using privileged methods (needs admin!)

Collection Methods



Sharphound.exe [**--CollectionMethod** <method[]>] [**--Domain** <domain[]>] [±]

v0.0w @SadProcessor

- Use **--LDAPFilter** to collect specific OU/Object [precision+/stealth+]
- Use **--SecureLDAP** to connect over LDAPS 636 iso LDAP 389 [Use **--LDAPPort** if other]
- Using **--Stealth** has no effect on DC methods

Collection Methods	Default	All	DCOnly	ComputerOnly	LocalGroup				ATT&CK
GPOLocalGroup			●			DC LDAP 389	Same as LocalGroup but via GPO	Parses SYSVOL + resolves SID	S0521 TA0007
Trusts	●	●	●				(:Domain)-[:TrustedBy]->(:Domain)	**--SearchForest** deprecated	T1482
Container	●	●	●				(:GPO)-[:GpLink]->(:OU)-[:Contains]->(x)	Collects GPOs + OU Tree	T1033 T1069.002 T1082 T1087.002
Group	●	●	●				(x)-[:MemberOf]->(:Group)	Groups/Users/Computers	
ACL	●	●	●				()-[{isacl}]->()	Collects all ACLs	
ObjectProps	●	●	●				x.propertyname + SIDHistory/Act/Delegate	More: **--CollectAllProperties**	
SPNTargets		●					(x)-[:SQLAdmin]->(:Computer)	+ Checks SQL port on Hosts	
LocalAdmin	●	●		●	●	Hosts RPC SMB 445	(x)-[:AdminTo]->(:Computer)	Uses NetLocalGroupGetMembers Requires Admin on 2016+ and w10 1607+	T1033 T1049 T1069.001 T1087.001 T1135
RDP	●	●		●	●		(x)-[:CanRDP]->(:Computer)		
DCOM	●	●		●	●		(x)-[:ExecuteDCOM]->(:Computer)		
PSRemote	●	●		●	●		(x)-[:CanPSRemote]->(:Computer)		
Session	●	●		●			(:Computer)-[:HasSession]->(:User)	Uses NetSessionEnum	
LoggedOn		●						Uses NetWkStaUserEnum	

- Add **--Throttle** and **--Jitter** [stealth+]

Session vs LoggedOn:
- **LoggedOn** is more accurate but requires Admin
- **Session** requires Admin on 2016+ / w10 1607+
- Using **--Stealth** on session methods will only target potentially interesting hosts

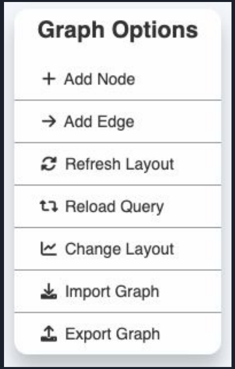
- Host methods also LDAP to DC for Host list and to resolve collected SIDs
- Use **--ComputerList** to bypass LDAPFilter
- Using **--Stealth** on LocalGroup methods reverts to GPOLocalGroup [DC]

- Use **--Loop** to increase session data coverage [attack paths nbr]
Syntax: **-c** session **--Loop** **--LoopDuration** HH:MM:SS **--LoopInterval** HH:MM:SS
- Drop zips in UI to import > Will add new without erasing old

[Ouput: **--ZipFileName** / **--OutputPrefix** / **--OutputDirectory** / **--RandomizeFileNames** / **--CacheFileName** / ...] +more

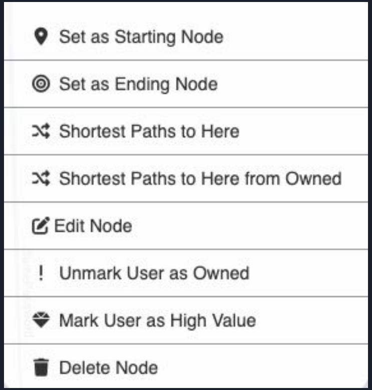
https://github.com/SadProcessor/HandsOnBloodHound/blob/master/BH21/BH4_SharpHound_Cheat.pdf

BloodHound Interface

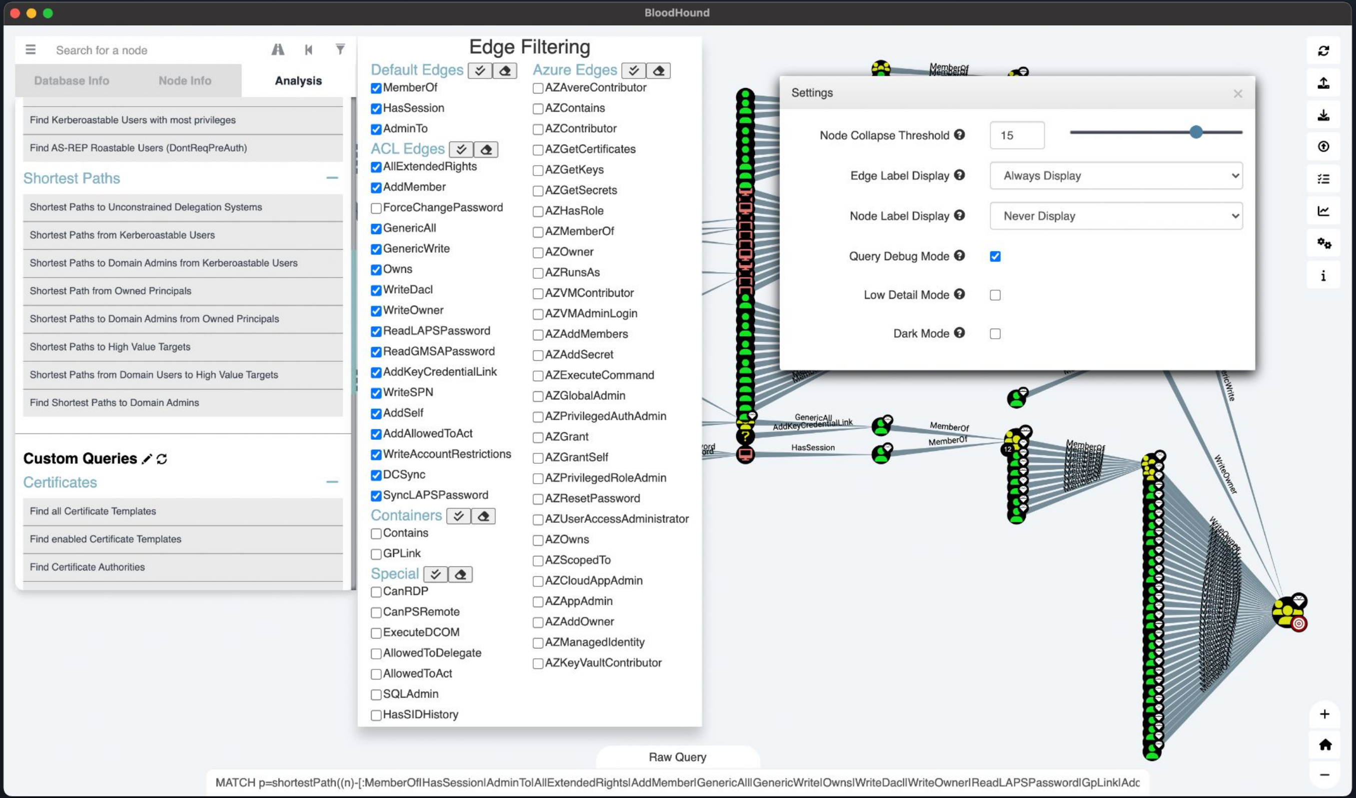


Graph Options

- + Add Node
- → Add Edge
- Refresh Layout
- Reload Query
- Change Layout
- Import Graph
- Export Graph



- Set as Starting Node
- Set as Ending Node
- Shortest Paths to Here
- Shortest Paths to Here from Owned
- Edit Node
- ! Unmark User as Owned
- Mark User as High Value
- Delete Node



BloodHound

Search for a node

Database Info | Node Info | Analysis

- Find Kerberoastable Users with most privileges
- Find AS-REP Roastable Users (DontReqPreAuth)

Shortest Paths

- Shortest Paths to Unconstrained Delegation Systems
- Shortest Paths from Kerberoastable Users
- Shortest Paths to Domain Admins from Kerberoastable Users
- Shortest Path from Owned Principals
- Shortest Paths to Domain Admins from Owned Principals
- Shortest Paths to High Value Targets
- Shortest Paths from Domain Users to High Value Targets
- Find Shortest Paths to Domain Admins

Custom Queries

Certificates

- Find all Certificate Templates
- Find enabled Certificate Templates
- Find Certificate Authorities

Edge Filtering

Default Edges

- [✓] MemberOf
- [✓] HasSession
- [✓] AdminTo

ACL Edges

- [✓] AllExtendedRights
- [✓] AddMember
- [ ] ForceChangePassword
- [✓] GenericAll
- [✓] GenericWrite
- [✓] Owns
- [✓] WriteDacl
- [✓] WriteOwner
- [✓] ReadLAPSPassword
- [✓] ReadGMSAPassword
- [✓] AddKeyCredentialLink
- [✓] WriteSPN
- [✓] AddSelf
- [✓] AddAllowedToAct
- [✓] WriteAccountRestrictions
- [✓] DCSync
- [✓] SyncLAPSPassword

Containers

- [ ] Contains
- [ ] GPLink

Special

- [ ] CanRDP
- [ ] CanPSRemote
- [ ] ExecuteDCOM
- [ ] AllowedToDelegate
- [ ] AllowedToAct
- [ ] SQLAdmin
- [ ] HasSIDHistory

Azure Edges

- [ ] AZAvereContributor
- [ ] AZContains
- [ ] AZContributor
- [ ] AZGetCertificates
- [ ] AZGetKeys
- [ ] AZGetSecrets
- [ ] AZHasRole
- [ ] AZMemberOf
- [ ] AZOwner
- [ ] AZRunsAs
- [ ] AZVMContributor
- [ ] AZVMAdminLogin
- [ ] AZAddMembers
- [ ] AZAddSecret
- [ ] AZExecuteCommand
- [ ] AZGlobalAdmin
- [ ] AZPrivilegedAuthAdmin
- [ ] AZGrant
- [ ] AZGrantSelf
- [ ] AZPrivilegedRoleAdmin
- [ ] AZResetPassword
- [ ] AZUserAccessAdministrator
- [ ] AZOwns
- [ ] AZScopedTo
- [ ] AZCloudAppAdmin
- [ ] AZAppAdmin
- [ ] AZAddOwner
- [ ] AZManagedIdentity
- [ ] AZKeyVaultContributor

Settings

Node Collapse Threshold	15
Edge Label Display	Always Display
Node Label Display	Never Display
Query Debug Mode	✓
Low Detail Mode	☐
Dark Mode	☐

Raw Query

```
MATCH p=shortestPath((n)-[:MemberOf|HasSession|AdminTo|AllExtendedRights|AddMember|GenericAll|GenericWrite|Owns|WriteDacl|WriteOwner|ReadLAPSPassword|GpLink|Adc
```

Cypher Query Breakdown

```
MATCH p=shortestPath((n
{owned:true})-[:MemberOf|AdminTo|AllExtendedRights|AddMember|GenericAll|GenericWrite|Owns|WriteDacl|WriteOwner|ReadLAPSPasswor
d|Contains|ReadGMSAPassword*1..]->(m:Group {name:"DOMAIN ADMINS@DOMAIN.COM"})) WHERE NOT n=m RETURN p
```

MATCH searches for nodes and RETURN defines the data returned from the query. WHERE (NOT) is adding constraints to the query.

p, n, m – Variables p is the result of the shortestPath function, n,m are variables that represent nodes.

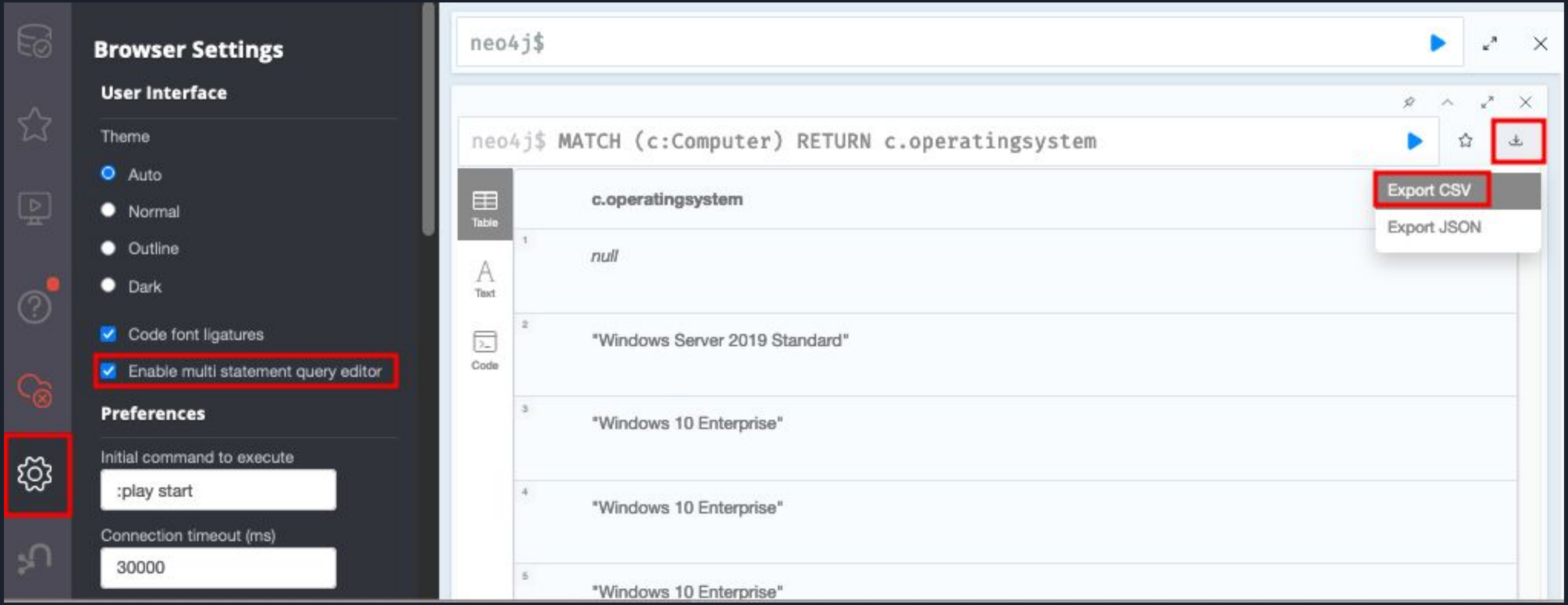
Group – Node label. in BloodHound think of this as the node type: Group, Computer, User, OU, etc.

[:TYPE*minHops..maxHops] – Relationship types can be defined inside of a relationship arrow (-->, <--, --).

{key:value} – Node properties.

Learn Cypher - Dog Whisper Handbook: https://ernw.de/download/BloodHoundWorkshop/ERNW_DogWhispererHandbook.pdf

Using the Cypher Console Example



Theme: [✓ Auto] Normal Outline Dark

✓ Code font ligatures

✓ Enable multi statement query editor

Initial command to execute: :play start

Connection timeout (ms): 30000

```
neo4j$
neo4j$ MATCH (c:Computer) RETURN c.operatingsystem
```

	c.operatingsystem
1	*null*
2	"Windows Server 2019 Standard"
3	"Windows 10 Enterprise"
4	"Windows 10 Enterprise"
5	"Windows 10 Enterprise"

Neo4j Bulk Mark Owned/High Value

Mark Owned:

```
MATCH (n {name:'<NAME@DOMAIN.COM>'}) SET n.owned=true;
```

Mark High Value:

```
MATCH (n {name:'<NAME@DOMAIN.COM>'}) SET n.highvalue=true;
```

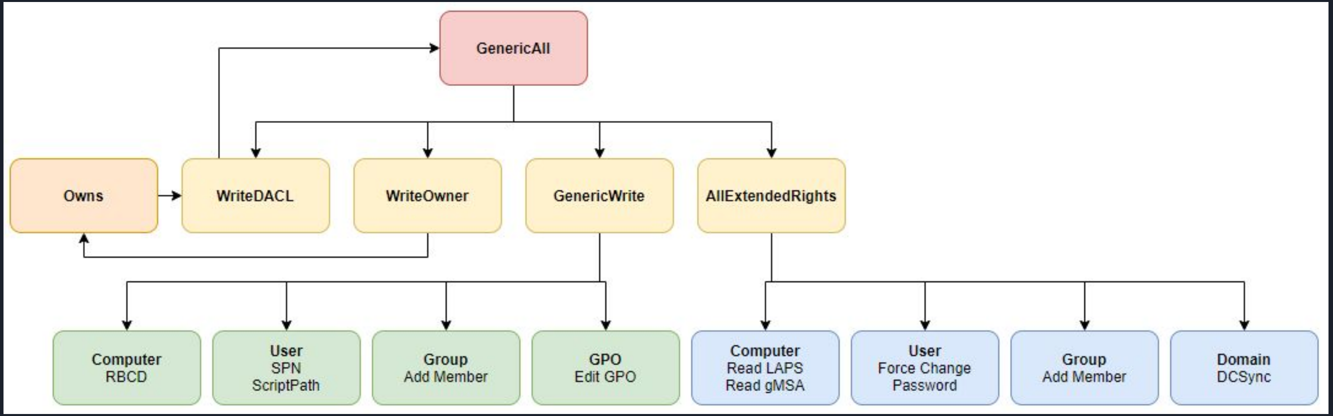
Relationship Types / AD ACL Implications

• GenericAll/GenericWrite/Owns -> User	- Change Password, Targeted Kerberoast, Shadow Credentials*
• GenericAll/Owns -> Computer	- Read LAPS/GMSA Password, RBCD*, Shadow Credentials*
• AllExtendedRights -> Computer	- Read LAPS/GMSA Password
• GenericWrite -> Computer	- RBCD*, Shadow Credentials*
• GenericAll/GenericWrite -> Group	- Add/Change Membership
• WriteDacl -> Any	- Grant any of the above permissions

- Shadow Credentials requires Server 2016 Domain Functional Level and ADCS
- Resource Based Constrained Delegation Requires Server 2012 Functional Level

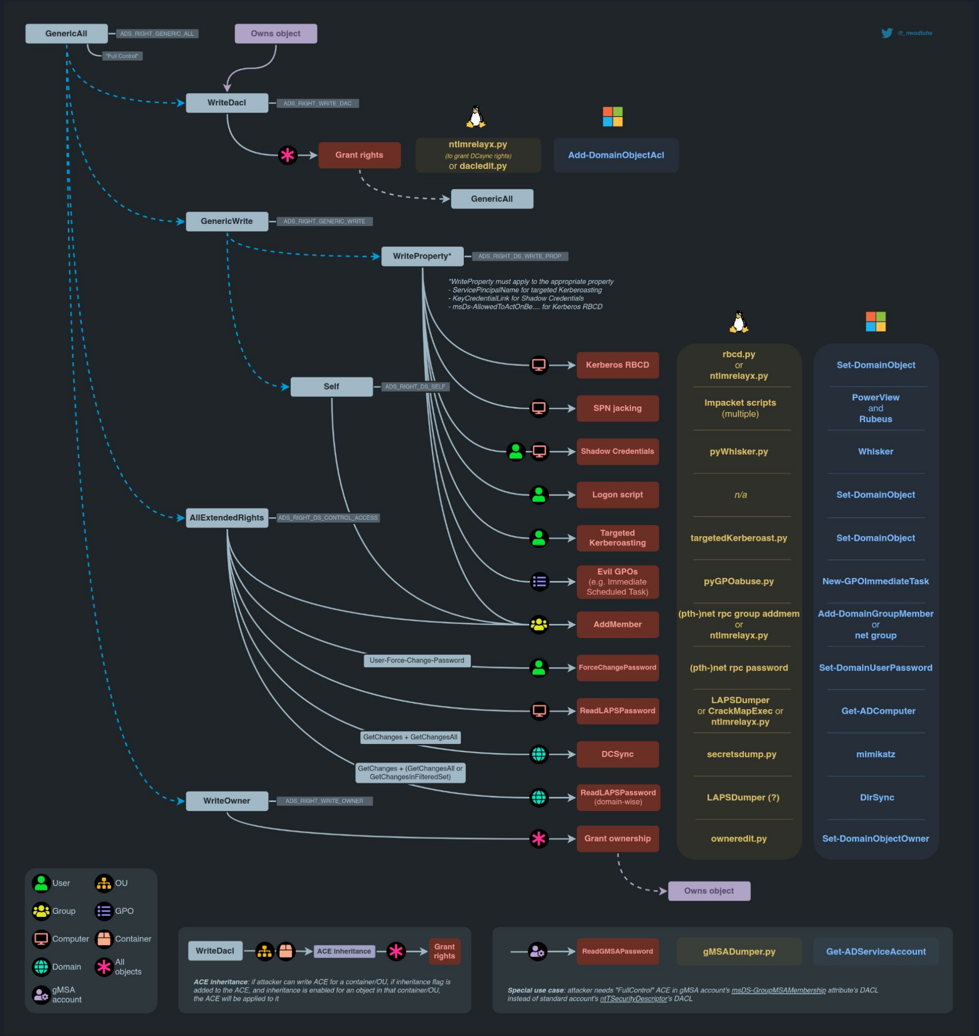
Relationship Types / AD ACL Implications

Attack Methods (Simplified)



https://ppn.snovvcrash.rocks/pentest/infrastructure/ad/acl-abuse

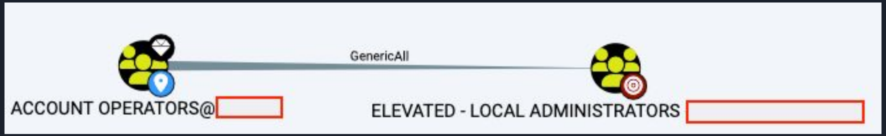
Relationship Types / AD ACL Implications


GenericAll
ADS_RIGHT_GENERIC_ALL
"Full Control"
Owns object
WriteDacl
ADS_RIGHT_WRITE_DAC
Grant rights
ntlmrelayx.py (to grant DCsync rights) or dacledit.py
Add-DomainObjectAcl
GenericAll
GenericWrite
ADS_RIGHT_GENERIC_WRITE
WriteProperty*
ADS_RIGHT_DS_WRITE_PROP
*WriteProperty must apply to the appropriate property
- ServicePincipalName for targeted Kerberoasting
- KeyCredentialLink for Shadow Credentials
- msDs-AllowedToActOnBe.... for Kerberos RBCD
Self
ADS_RIGHT_DS_SELF
AllExtendedRights
ADS_RIGHT_DS_CONTROL_ACCESS
User-Force-Change-Password
GetChanges + GetChangesAll
GetChanges + (GetChangesAll or GetChangesInFilteredSet)
WriteOwner
ADS_RIGHT_WRITE_OWNER
Kerberos RBCD
SPN jacking
Shadow Credentials
Logon script
Targeted Kerberoasting
Evil GPOs (e.g. Immediate Scheduled Task)
AddMember
ForceChangePassword
ReadLAPSPassword
DCSync
ReadLAPSPassword (domain-wise)
Grant ownership
Owns object
rbcd.py or ntlmrelayx.py
Impacket scripts (multiple)
pyWhisker.py
n/a
targetedKerberoast.py
pyGPOabuse.py
(pth-)net rpc group addmem or ntlmrelayx.py
(pth-)net rpc password
LAPSDumper or CrackMapExec or ntlmrelayx.py
secretsdump.py
LAPSDumper (?)
owneredit.py
Set-DomainObject
PowerView and Rubeus
Whisker
Set-DomainObject
Set-DomainObject
New-GPOImmediateTask
Add-DomainGroupMember or net group
Set-DomainUserPassword
Get-ADComputer
mimikatz
DirSync
Set-DomainObjectOwner
User
OU
Group
GPO
Computer
Container
Domain
All objects
gMSA account
WriteDacl
ACE inheritance
Grant rights
ACE inheritance: if attacker can write ACE for a container/OU, if inheritance flag is added to the ACE, and inheritance is enabled for an object in that container/OU, the ACE will be applied to it
ReadGMSAPassword
gMSADumper.py
Get-ADServiceAccount
Special use case: attacker needs "FullControl" ACE in gMSA account's msDS-GroupMSAMembership attribute's DACL instead of standard account's ntTSecurityDescriptor's DACL

https://www.thehacker.recipes/ad/movement/dacl

Attacking Groups

- Add/Change Group Membership
  - Net.exe Commands
  - PowerView's Add-DomainGroupMember
  - https://github.com/FuzzySecurity/StandIn (C#)
  - Net (Samba) and/or pth-toolkit
  - Python Based Tools
    - https://www.n00py.io/2020/01/managing-active-directory-groups-from-linux/
    - https://github.com/PShlyundin/ldap_shell
    - https://github.com/CravateRouge/bloodyAD
    - https://github.com/aniqfakhrul/powerview.py
    - https://github.com/zblurx/acltoolkit


GenericAll
ACCOUNT OPERATORS@[redacted]
ELEVATED - LOCAL ADMINISTRATORS [redacted]

Attacking Groups

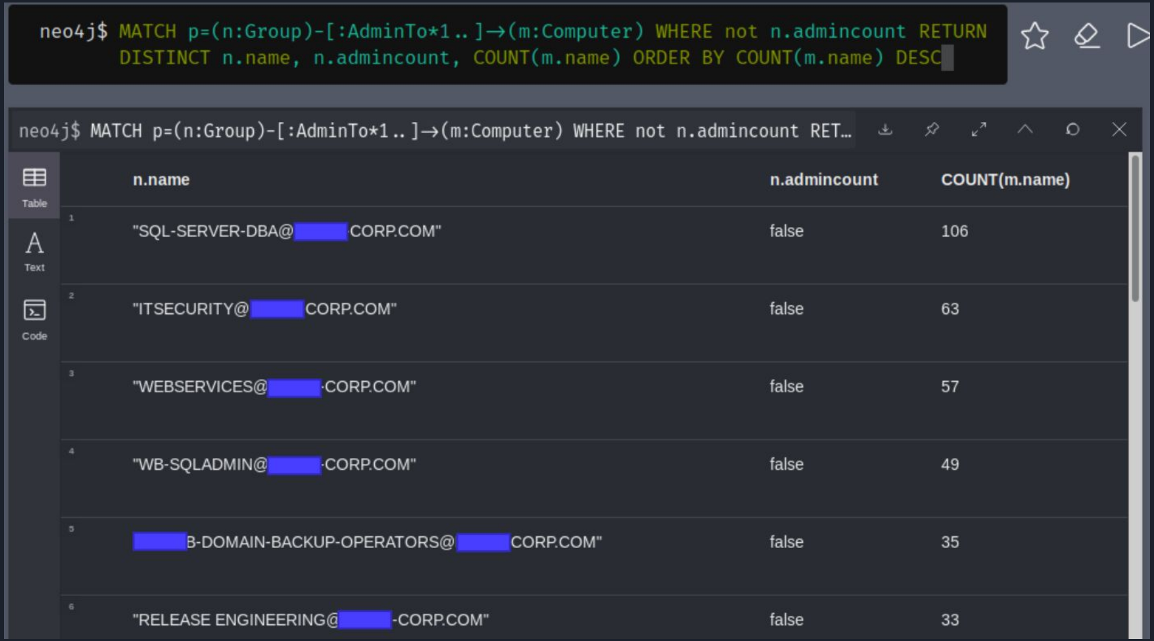
- Leverage BloodHound/Neo4j to find groups with admin to computers

```
MATCH p=(n:Group)-[:AdminTo*1..]->(m:Computer) WHERE NOT
n.admincount RETURN p

MATCH p=(n:Group)-[:AdminTo*1..]->(m:Computer) WHERE not
n.admincount RETURN DISTINCT n.name
```

Attacking Groups

```
MATCH p=(n:Group)-[:AdminTo*1..]->(m:Computer) WHERE not n.admincount RETURN
DISTINCT n.name, n.admincount, COUNT(m.name) ORDER BY COUNT(m.name) DESC
```



```
neo4j$ MATCH p=(n:Group)-[:AdminTo*1..]→(m:Computer) WHERE not n.admincount RETURN
       DISTINCT n.name, n.admincount, COUNT(m.name) ORDER BY COUNT(m.name) DESC
```

```
neo4j$ MATCH p=(n:Group)-[:AdminTo*1..]→(m:Computer) WHERE not n.admincount RET…
```

	n.name	n.admincount	COUNT(m.name)
1	"SQL-SERVER-DBA@[redacted]CORP.COM"	false	106
2	"ITSECURITY@[redacted]CORP.COM"	false	63
3	"WEBSERVICES@[redacted]-CORP.COM"	false	57
4	"WB-SQLADMIN@[redacted]-CORP.COM"	false	49
5	[redacted]B-DOMAIN-BACKUP-OPERATORS@[redacted]CORP.COM"	false	35
6	"RELEASE ENGINEERING@[redacted]-CORP.COM"	false	33

Attacking Computers

- Read LAPS Passwords
  - https://github.com/n00py/LAPSDumper
  - crackmapexec smb <ip> -u user -p pass --laps
  - https://github.com/swisskyrepo/SharpLAPS
- Shadow Credentials
  - Modify ***msDS-KeyCredentialLink*** Attribute
  - https://github.com/eladshamir/Whisker
  - https://github.com/ShutdownRepo/pywhisker

- Resource Based Constrained Delegation (RBCD)
  - Modify ***msDS-AllowedToActOnBehalfOfOtherIdentity*** Attribute
  - PowerView's Set-DomainObject / Rubeus
  - https://github.com/FuzzySecurity/StandIn
  - https://github.com/CravateRouge/bloodyAD
  - https://github.com/PShlyundin/ldap_shell
  - Impacket rbcd.py
  - https://github.com/NinjaStyle82/rbcd_permissions

Attacking Users

- Force a Password Reset
  - Net.exe Commands
  - PowerView's Set-DomainUserPassword
  - Set-ADAccountPassword
  - Rpcclient
  - Net (Samba) and/or pth-toolkit
  - https://github.com/PShlyundin/ldap_shell
  - https://github.com/CravateRouge/bloodyAD
  - Cleanup: https://www.trustedsec.com/blog/manipulating-user-passwords-without-mimikatz/
- Targeted Kerberoast
  - Powerview’s Set-DomainObject and Get-DomainSPNTicket
  - Rubeus.exe kerberoast
  - https://github.com/ShutdownRepo/targetedKerberoast
- Read gMSA Passwords
  - https://github.com/micahvandeusen/gMSADumper
  - crackmapexec smb <ip> -u user -p pass --gmsa
  - https://github.com/rvazarkar/GMSAPasswordReader
- Targeted AS-REP Roast
  - https://github.com/FuzzySecurity/StandIn
  - Rubeus.exe asreproast
  - https://github.com/PShlyundin/ldap_shell
  - https://github.com/CravateRouge/bloodyAD
  - Impacket GetNPUsers.py
- Shadow Credentials
  - Modify ***msDS-KeyCredentialLink*** Attribute
  - https://github.com/eladshamir/Whisker
  - https://github.com/ShutdownRepo/pywhisker
- Modify Logon Script
  - Powerview’s Set-DomainObject
  -

Attacking GPOs

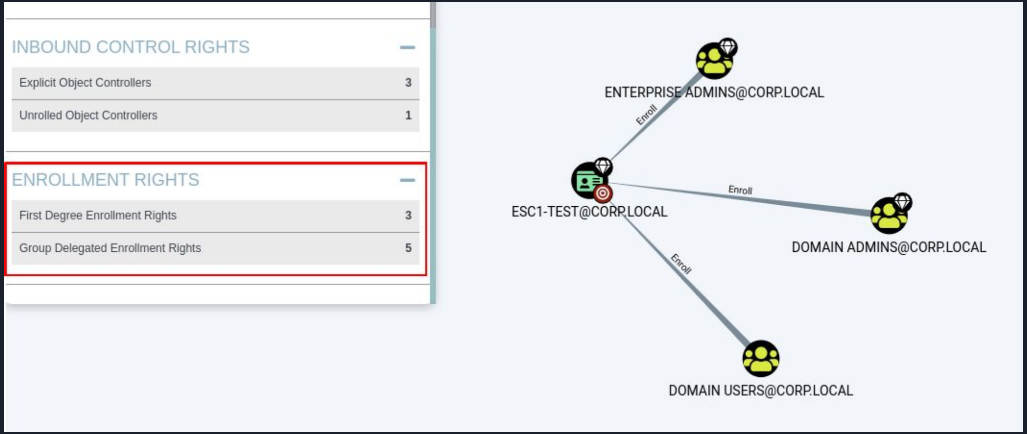
- Adding Scheduled Task
  - https://github.com/X-C3LL/GPOwned
  - https://github.com/Hackndo/pyGPOAbuse
  - PowerView’s New-GPOImmediateTask
  - https://github.com/FSecureLABS/SharpGPOAbuse
- Creating Local Users
  - https://github.com/FuzzySecurity/StandIn
  - Use Remote Server Administration Tools (RSAT)
  - Group Policy Management Editor
- Probably a bunch of other ways, lots of things you can configure via GPO

Attacking Domain

- If you have GenericAll, AllExtendedRights, or DS-Replication-Get-Changes-All + DS-Replication-Get-Changes:
  - DCSync
    - Impacket secretsdump.py
    - Mimikatz lsadump::dcsync
- If you have Owns or WriteDACL:
  - PowerView's Add-DomainObjectAcl
  - https://github.com/n00py/DCSync
  - https://github.com/CravateRouge/bloodyAD
  - https://github.com/PShlyundin/ldap_shell
- If you have WriteOwner:
  - PowerView's Set-DomainObjectOwner
  - https://github.com/CravateRouge/bloodyAD
  - https://github.com/PShlyundin/ldap_shell

ADCS - Certipy

- Certipy is a tool for abusing/exploiting ADCS
- Supported BloodHound ingestible output since version 2.0
  - Uses GPO objects to represent certificates in “Old BloodHound” mode
  - Forked GUI uses new vertices to represent CAs and Certificate templates



INBOUND CONTROL RIGHTS

Explicit Object Controllers	3
Unrolled Object Controllers	1

ENROLLMENT RIGHTS

First Degree Enrollment Rights	3
Group Delegated Enrollment Rights	5

https://miro.medium.com/max/720/1*3RCynhxvuArY-6X2xWEqQg.png

https://research.ifcr.dk/certipy-2-0-bloodhound-new-escalations-shadow-credentials-golden-certificates-and-more-34d1c26f0dc6

https://research.ifcr.dk/certipy-4-0-esc9-esc10-bloodhound-gui-new-authentication-and-request-methods-and-more-7237d88061f7

Using BloodHound - Tips and Tricks

- Mark every compromised computer or user “Owned”.
  - Possible to automatically assign this with CrackMapExec and Cobalt Strike
    - https://github.com/NinjaStyle82/cme2bh (deprecated)
    - cme smb <ip> -u <user> -p <password> -M bh_owned
    - https://github.com/waffl3ss/bloodpath
    - https://github.com/Coalfire-Research/Vampire
- Run queries from Owned to High Value.

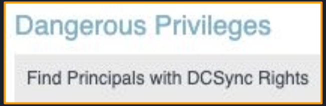
```
MATCH p=shortestPath((g {owned:true})-[*1..]->(n
{highvalue:true})) WHERE g<>n return p
```

- Use built-in filters to narrow to specific relationship types.
  - For example: fill out the filter checkboxes, then run “Shortest Paths to Domain Admins from Owned Principals”
- query, and copy pasta what’s inside the square brackets.
- Use allShortestPaths if you think ShortestPath is showing you a bad relationship/edge. (Shortest path only shows one relationship type)

High Value: Principals With DCSync Rights

- Find Objects with DCSync (built-in queries)

```
MATCH p=()-[:DCSync|AllExtendedRights|GenericAll]->(:Domain
{name: "DOMAIN.LOCAL"}) RETURN p
```

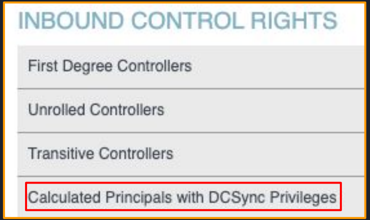

Dangerous Privileges

Find Principals with DCSync Rights

```
MATCH (n1)-[:MemberOf|GetChanges*1..]->(u:Domain {name:
"DOMAIN.LOCAL"}) WITH n1,u MATCH
(n1)-[:MemberOf|GetChangesAll*1..]->(u) WITH n1,u MATCH p =
(n1)-[:MemberOf|GetChanges|GetChangesAll*1..]->(u) RETURN p
```

- Set them all as High Value

```
MATCH p=(n)-[:DCSync|AllExtendedRights|GenericAll]->(:Domain
{name: "DOMAIN.LOCAL"}) SET n.highvalue=True
```


INBOUND CONTROL RIGHTS

First Degree Controllers

Unrolled Controllers

Transitive Controllers

Calculated Principals with DCSync Privileges

```
MATCH (n1)-[:MemberOf|GetChanges*1..]->(u:Domain {name:
"DOMAIN.LOCAL"}) WITH n1,u MATCH
(n1)-[:MemberOf|GetChangesAll*1..]->(u) WITH n1,u MATCH p =
(n1)-[:MemberOf|GetChanges|GetChangesAll*1..]->(u)
SET n1.highvalue=True
```

Tip: Adding Admin Groups to High Value

- Groups not marked as “High Value” already, but have the Admin Count Flag

```
MATCH p = (g:Group {admincount: True}) WHERE NOT EXISTS(g.highvalue)
OR g.highvalue = False RETURN g
```

- Groups that do NOT have the Admin Count flag, but do allow local admin to computers

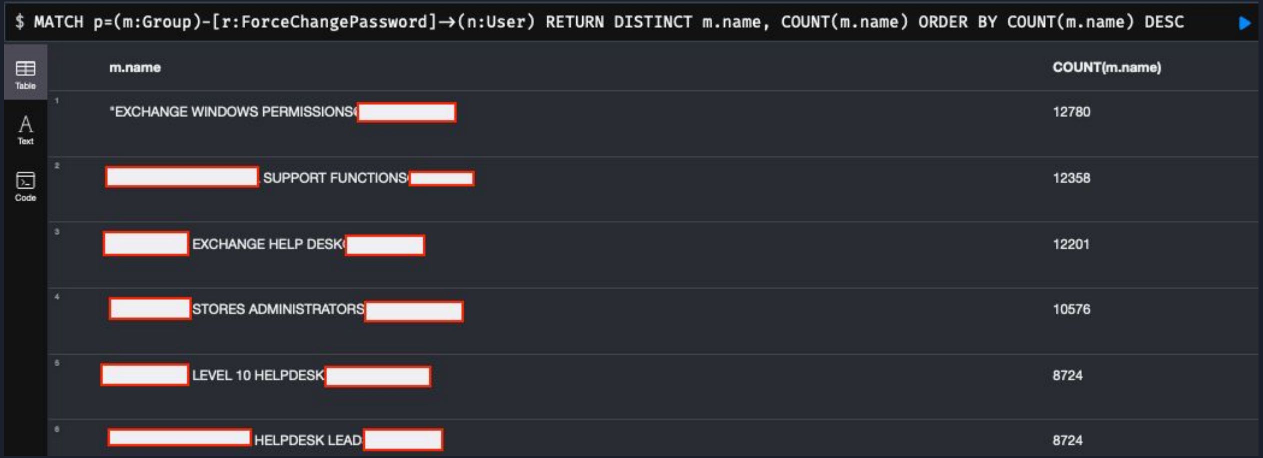
```
MATCH p=(n:Group)-[:AdminTo*1..]->(m:Computer) WHERE NOT n.admincount
RETURN p
```

High Value: Groups That Can Reset Passwords

- Groups that can change user passwords, sorted by the amount of users

```
MATCH p=(m:Group)-[r:ForceChangePassword]->(n:User) RETURN m

MATCH p=(m:Group)-[r:ForceChangePassword]->(n:User) RETURN DISTINCT
m.name, COUNT(m.name) ORDER BY COUNT(m.name) DESC
```



```
$ MATCH p=(m:Group)-[r:ForceChangePassword]→(n:User) RETURN DISTINCT m.name, COUNT(m.name) ORDER BY COUNT(m.name) DESC
```

	m.name	COUNT(m.name)
1	"EXCHANGE WINDOWS PERMISSIONS[redacted]	12780
2	[redacted] SUPPORT FUNCTIONS[redacted]	12358
3	[redacted] EXCHANGE HELP DESK[redacted]	12201
4	[redacted] STORES ADMINISTRATORS[redacted]	10576
5	[redacted] LEVEL 10 HELPDESK[redacted]	8724
6	[redacted] HELPDESK LEAD[redacted]	8724

High Value: Unconstrained Delegation

- Find all computers that can perform unconstrained delegation but are not DCs.

```
MATCH (c1:Computer)-[:MemberOf*1..]->(g:Group) WHERE g.objectid ENDS
WITH '-516' WITH COLLECT(c1.name) AS domainControllers MATCH
(c2:Computer {unconstraineddelegation:true}) WHERE NOT c2.name IN
domainControllers RETURN c2
```

- Exploit with Rubeus.exe monitor (Windows)
- https://github.com/dirkjanm/krbrelayx (Python)

https://hausec.com/2019/09/09/bloodhound-cypher-cheatsheet/

High Value: Azure AD Connect

- Synchronization service that keeps Active Directory and Office 365 in sync
- Under a default set-up, an account is created with DCSync permissions
  - MSOL_[HEX]
- This account plaintext password can be extracted from the AD Connect Server
  - https://blog.xpnsec.com/azuread-connect-for-redteam/
  - https://gist.github.com/xpn/f12b145dba16c2eebdd1c6829267b90c

- If you have access to Azure, you can find it under *Azure Active Directory Connect Health -> Sync Services -> Azure Active Directory Connect Servers*
- MSOnline Powershell Module:

```
(Get-MsolCompanyInformation).DirSyncClientMachineName
```

High Value: Azure AD Connect

- Find Azure AD Connect servers and mark them as high value

```
MATCH (n:User) WHERE n.name STARTS WITH "MSOL" RETURN
split(n.description,' ')[15]
```

```
MATCH (u:User)WHERE u.name STARTS WITH "MSOL" WITH split(u.description, "
")[15] AS word UNWIND word AS w MATCH (c:Computer)WHERE c.name STARTS
WITH w RETURN c
```

- Note: This finds servers created with defaults, but there may be more, look for computers with names like "*azure*", "*sync*", "*AAD*", etc.

High Value: Cert Publishers

- Find all computers in the Cert Publishers group.

```
MATCH p=(n:Group)<-[:MemberOf*1..]-(m)  WHERE n.name =~ "CERT
PUBLISHERS.*" RETURN p
```

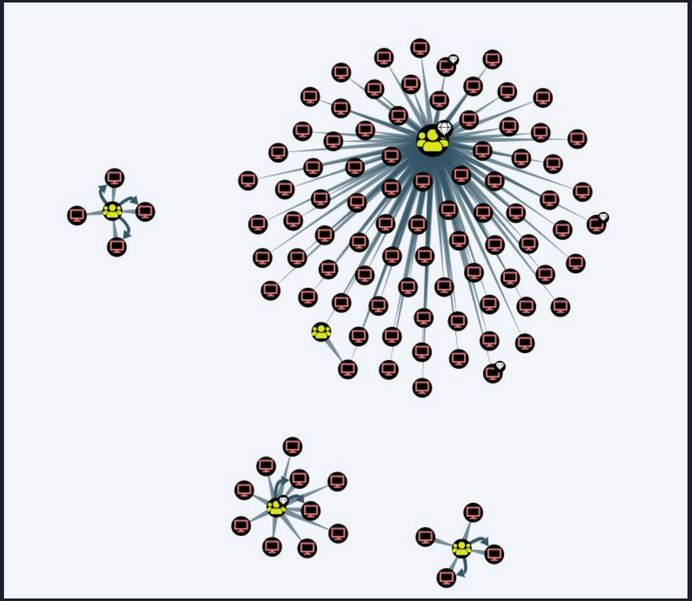
- Pwn a CA -> Golden Certificate!
  - https://github.com/ly4k/Certipy#golden-certificates
  - https://pentestlab.blog/2021/11/15/golden-certificate

- SCCM Servers?
  - Provide “Updates” to High Value Targets
- https://github.com/Mayyhem/SharpSCCM

```
MATCH (n) WHERE n.name CONTAINS "SCCM" RETURN n UNION MATCH (n)
WHERE n.description CONTAINS "SCCM" RETURN n
```

Tip: Computers Admin to Other Computers

```
MATCH p =
(c1:Computer)-[r1:AdminTo]->(c2:Computer)
RETURN p UNION ALL MATCH p =
(c3:Computer)-[r2:MemberOf*1..]->(g:Group)-[
r3:AdminTo]->(c4:Computer) RETURN p
```



- Coerced authentication from one computer another
- SMB Signing must NOT be enforced

  1. Printerbug/Coercer
  2. Impacket Ntlmrleayx.py
  3. Dump SAM/LSA

- Common to see on Exchange, SCCM, and SQL Servers

Tip: Outbound Object Control

- If all your owned users seem truly useless, try these queries to see if they can do ANYTHING at all:

```
MATCH p = (g:User {owned: True})-[r]->(n) WHERE r.isacl=true RETURN p

MATCH p = (g1:User {owned:
True})-[r1:MemberOf*1..]->(g2:Group)-[r2]->(n) WHERE r2.isacl=true
RETURN p
```

Tip: LAPS non-enabled Computers for Lateral Movement

If a computer has LAPS non-enabled, does it potentially share a password with a high-value computer?

Do any high-value computers have LAPS non-enabled?

```
MATCH (c:Computer {haslaps:False}) WHERE c.highvalue=True RETURN c
```

Do any of our owned accounts have paths to computer nodes with LAPS non-enabled?

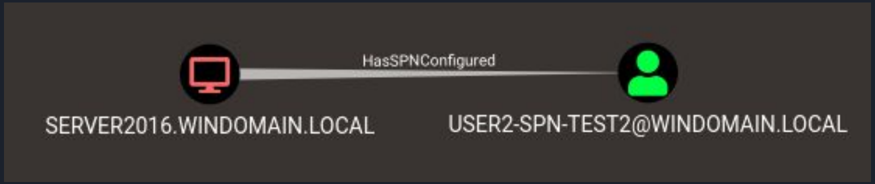
```
MATCH p=shortestpath((u {owned:true})-[:MemberOf|AdminTo|Owns|AllExtendedRights|GenericAll|GenericWrite|ReadLAPSPassword|AddKeyCredentialLink*1..]->(c:Computer {haslaps:false})) RETURN p
```

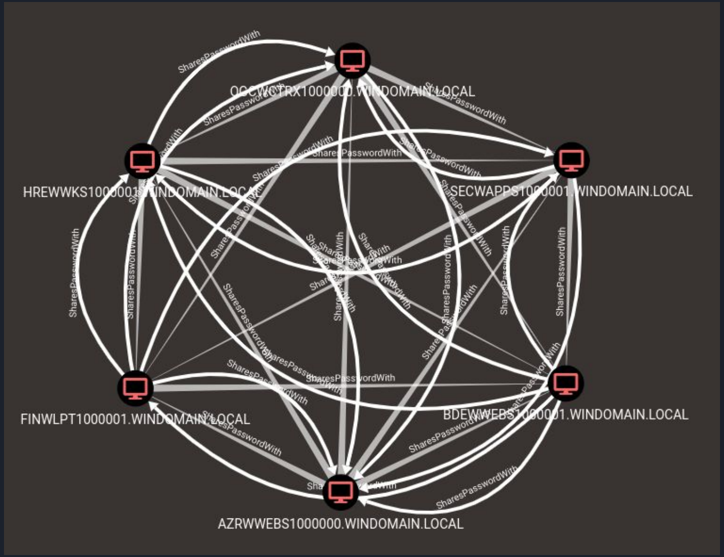
Tip: Extending BloodHound

- There are some open source tools which can expand the data in available in BloodHound
- Max - https://github.com/knavesec/Max
  - Can set a list users/computers as owned or high value

- Add-spns
  - Adds the HasSPNConfigured relationship to objects in the database
- Add-spw
  - Create SharesPasswordWith relationships
  - Visualizes local admin re-use

Note: Custom Edges will not show up on built-in queries



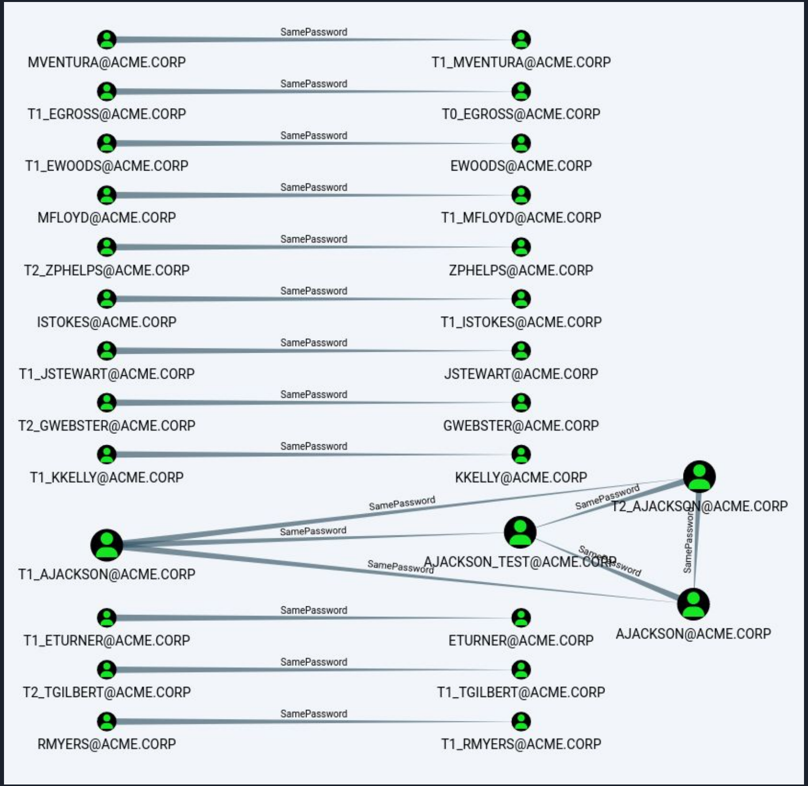

OCCWCTRX1000000.WINDOMAIN.LOCAL
HREWWKS1000001.WINDOMAIN.LOCAL
SECWAPPS1000001.WINDOMAIN.LOCAL
FINWLPT1000001.WINDOMAIN.LOCAL
BDEWWEBS1000001.WINDOMAIN.LOCAL
AZRWWEBS1000000.WINDOMAIN.LOCAL
SharesPasswordWith

https://whynotsecurity.com/blog/max2/

Post-Ex: Shared Password Analysis

- Takes NTDS output and generates shared password clusters
- Can be imported to BloodHound, creates new edges (and thus new paths)
- Excellent at visualizing password sharing issues

https://github.com/SySS-Research/hashcathelper



https://github.com/SySS-Research/hashcathelper/blob/main/doc/bloodhound_clusters.png

Post-Ex: CrackHound

- Allows you to add plaintext passwords to BloodHound, post-compromise
- Search out additional paths via weak passwords
    - What users with a cracked password are members of high value groups?
    - What users with weak passwords have VPN access?
    - What Kerberoastable users were cracked?
    - What users with a weak password have a path to Domain Admin?

https://www.trustedsec.com/blog/expanding-the-hound-introducing-plaintext-field-to-compromised-accounts/
https://github.com/trustedsec/CrackHound

A Slide For the Blue Team

Turning BloodHound Data into useful lists:

- Max: Domain Password Audit Tool
  - https://whynotsecurity.com/blog/max3/
  - Password audit enriched with BloodHound data
- PlumHound - BloodHound Report Engine
  - https://github.com/PlumHound/PlumHound
- Cypheroth
  - Spreadsheets!
  - https://github.com/seajaysec/cypheroth
- WatchDog
  - https://github.com/SadProcessor/WatchDog
  - https://insinuator.net/2019/10/blue-hands-on-bloodhound/
- PingCastle
  - https://www.pingcastle.com

Questions?