

# Detecting Kerberoasting/AS-REProasting

## Kerberoasting

**Kerberoasting** is a technique targeting service accounts in Active Directory environments to extract and crack their password hashes. The attack exploits the way Kerberos service tickets are encrypted and the use of weak or easily crackable passwords for service accounts. Once an attacker successfully cracks the password hashes, they can gain unauthorized access to the targeted service accounts and potentially move laterally within the network.

An example of a Kerberoasting attack is using the [Rubeus](#) `kerberoast` module.

```

Windows PowerShell
PS C:\Users\JENNY_HICKMAN\tools> .\Rubeus.exe kerberoast
[{"SPN": "HTTP/iis.lab.internal.local", "AccountName": "iis_svc", "DistinguishedName": "CN=IIS Service Account,CN=Users,DC=lab,DC=internal,DC=local", "PwdLastSet": "3/6/2021 5:11:16 PM", "ServicePrincipalName": "HTTP/iis.lab.internal.local", "SupportedETypes": "RC4_HMAC_DEFAULT", "Hash": "$krb5tgs$23$+iis_svc$lab.internal.local$HTTP|iis.lab.internal.local*$6D8CF25729D4539160DA06180E7E1769$55A8847870C5BC3D790B8D054D8CE58FD9BAD399548D87002F0BD6DC6E65083BC1E9E01BF3E0E4714B397D6DDA14D7083DA5780193788FE75D3362E5C23B8009673517864F624330114B78D6FF0648D4BF1803000C1FD7F08A02019A4D35B6A9B017F34E3964495895BB24E9719DFFEC9F4B8FE14F4D0937568C64495F8B696EDBB01FD4DB53D8C6F494226BD462B29A0AA4C98DFE35896E517EE080163E85400809C0C2CC6610E97EFFF9E899B0B11A8BA6F655D048988582435A012CE6F280675C83B1C61B5837FBA355C26C27AE3F88531352B6862EA53CED06719E41E741531FA2B613511F0EFF4F08A5A001031C9764D1A1674057F545338682C172017FA086D5A7FA195F7FA5"}, {"SPN": "HTTP/autodiscover-lab.internal.local", "AccountName": "autodiscover", "DistinguishedName": "CN=autodiscover,CN=Autodiscover,OU=Autodiscover,DC=lab,DC=internal,DC=local", "PwdLastSet": "3/6/2021 5:11:16 PM", "ServicePrincipalName": "HTTP/autodiscover-lab.internal.local", "SupportedETypes": "RC4_HMAC_DEFAULT", "Hash": "$krb5tgs$23$+autodiscover$lab.internal.local$HTTP|autodiscover-lab.internal.local*$6D8CF25729D4539160DA06180E7E1769$55A8847870C5BC3D790B8D054D8CE58FD9BAD399548D87002F0BD6DC6E65083BC1E9E01BF3E0E4714B397D6DDA14D7083DA5780193788FE75D3362E5C23B8009673517864F624330114B78D6FF0648D4BF1803000C1FD7F08A02019A4D35B6A9B017F34E3964495895BB24E9719DFFEC9F4B8FE14F4D0937568C64495F8B696EDBB01FD4DB53D8C6F494226BD462B29A0AA4C98DFE35896E517EE080163E85400809C0C2CC6610E97EFFF9E899B0B11A8BA6F655D048988582435A012CE6F280675C83B1C61B5837FBA355C26C27AE3F88531352B6862EA53CED06719E41E741531FA2B613511F0EFF4F08A5A001031C9764D1A1674057F545338682C172017FA086D5A7FA195F7FA5"}, {"SPN": "HTTP/autodiscover-lab.internal.local", "AccountName": "autodiscover", "DistinguishedName": "CN=autodiscover,CN=Autodiscover,OU=Autodiscover,DC=lab,DC=internal,DC=local", "PwdLastSet": "3/6/2021 5:11:16 PM", "ServicePrincipalName": "HTTP/autodiscover-lab.internal.local", "SupportedETypes": "RC4_HMAC_DEFAULT", "Hash": "$krb5tgs$23$+autodiscover$lab.internal.local$HTTP|autodiscover-lab.internal.local*$6D8CF25729D4539160DA06180E7E1769$55A8847870C5BC3D790B8D054D8CE58FD9BAD399548D87002F0BD6DC6E65083BC1E9E01BF3E0E4714B397D6DDA14D7083DA5780193788FE75D3362E5C23B8009673517864F624330114B78D6FF0648D4BF1803000C1FD7F08A02019A4D35B6A9B017F34E3964495895BB24E9719DFFEC9F4B8FE14F4D0937568C64495F8B696EDBB01FD4DB53D8C6F494226BD462B29A0AA4C98DFE35896E517EE080163E85400809C0C2CC6610E97EFFF9E899B0B11A8BA6F655D048988582435A012CE6F280675C83B1C61B5837FBA355C26C27AE3F88531352B6862EA53CED06719E41E741531FA2B613511F0EFF4F08A5A001031C9764D1A1674057F545338682C172017FA086D5A7FA195F7FA5"}], [{"SPN": "HTTP/autodiscover-lab.internal.local", "AccountName": "autodiscover", "DistinguishedName": "CN=autodiscover,CN=Autodiscover,OU=Autodiscover,DC=lab,DC=internal,DC=local", "PwdLastSet": "3/6/2021 5:11:16 PM", "ServicePrincipalName": "HTTP/autodiscover-lab.internal.local", "SupportedETypes": "RC4_HMAC_DEFAULT", "Hash": "$krb5tgs$23$+autodiscover$lab.internal.local$HTTP|autodiscover-lab.internal.local*$6D8CF25729D4539160DA06180E7E1769$55A8847870C5BC3D790B8D054D8CE58FD9BAD399548D87002F0BD6DC6E65083BC1E9E01BF3E0E4714B397D6DDA14D7083DA5780193788FE75D3362E5C23B8009673517864F624330114B78D6FF0648D4BF1803000C1FD7F08A02019A4D35B6A9B017F34E3964495895BB24E9719DFFEC9F4B8FE14F4D0937568C64495F8B696EDBB01FD4DB53D8C6F494226BD462B29A0AA4C98DFE35896E517EE080163E85400809C0C2CC6610E97EFFF9E899B0B11A8BA6F655D048988582435A012CE6F280675C83B1C61B5837FBA355C26C27AE3F88531352B6862EA53CED06719E41E741531FA2B613511F0EFF4F08A5A001031C9764D1A1674057F545338682C172017FA086D5A7FA195F7FA5"}]

```

## Attack Steps:

- Identify Target Service Accounts:** The attacker enumerates Active Directory to identify service accounts with **Service Principal Names (SPNs)** set. Service accounts are often associated with services running on the network, such as SQL Server, Exchange, or other applications. The following is a code snippet from [Rubeus](#) that is related to this step.

```

try
{
    DateTime timeFromConverted = DateTime.ParseExact(pwdSetAfter, "MM-dd-yyyy", null);
    DateTime timeUntilConverted = DateTime.ParseExact(pwdSetBefore, "MM-dd-yyyy", null);
    string timePeriod = "(" + pwdLastSet + ">" + timeFromConverted.ToString("MM-dd-yyyy") + ")(<" + pwdLastSet + "<" + timeUntilConverted.ToString("MM-dd-yyyy") + ")";
    userSearchFilter = String.Format("(&(samAccountType=805306368)(servicePrincipalName={0}{1})){2}", userFilter, encFilter, timePeriod);
}
catch
{
    Console.WriteLine("\r\n[X] Error parsing /pwdsetbefore or /pwdsetafter, please use the format 'MM-dd-yyyy'");
    return;
}
else
{
    userSearchFilter = String.Format("(&(samAccountType=805306368)(servicePrincipalName={0}{1})){2}", userFilter, encFilter);
}

```

- Request TGS Tickets:** The attacker uses the identified service accounts to request **Ticket Granting Service (TGS)** tickets from the **Key Distribution Center (KDC)**. These TGS tickets contain encrypted service account password hashes. The following is a code snippet from [Rubeus](#) that is related to this step.

```
if (tgt != null)
```

[Resources](#)
[Go to Questions](#)

## Table of Contents

### Leveraging Windows Event Logs

- [Detecting Common User/Domain Recon](#)
- [Detecting Password Spraying](#)
- [Detecting Responder-like Attacks](#)
- [Detecting Kerberoasting/AS-REProasting](#)
- [Detecting Pass-the-Hash](#)
- [Detecting Pass-the-Ticket](#)
- [Detecting Overpass-the-Hash](#)
- [Detecting Golden Tickets/Silver Tickets](#)
- [Detecting Unconstrained Delegation/Constrained Delegation Attacks](#)
- [Detecting DCSync/DCShadow](#)

### Leveraging Splunk's Application Capabilities

- [Creating Custom Splunk Applications](#)

### Leveraging Zeek Logs

- [Detecting RDP Brute Force Attacks](#)
- [Detecting Beaconing Malware](#)
- [Detecting Nmap Port Scanning](#)
- [Detecting Kerberos Brute Force Attacks](#)
- [Detecting Kerberoasting](#)
- [Detecting Golden Tickets](#)
- [Detecting Cobalt Strike's PSEnc](#)
- [Detecting Zerologon](#)
- [Detecting Exfiltration \(HTTP\)](#)
- [Detecting Exfiltration \(DNS\)](#)
- [Detecting Ransomware](#)

### Skills Assessment

```

    // if a TGT .kirbi is supplied, use that for the request
    // this could be a passed TGT or if TGT delegation is specified

    if (String.Equals(supportedEType, "rc4") &&
        (
            ((supportedETypes & Interop.SUPPORTED_ETYPE.AES128_CTS_HMAC_SHA1_96) == Interop.SUPPORTED_ETYPE.AES128_CTS_HMAC_SHA1_96) ||
            ((supportedETypes & Interop.SUPPORTED_ETYPE.AES256_CTS_HMAC_SHA1_96) == Interop.SUPPORTED_ETYPE.AES256_CTS_HMAC_SHA1_96)
        )
    {
        // if we're roasting RC4, but AES is supported AND we have a TGT, specify RC4
        bool result = GetTGSRepHash(TGT, servicePrincipalName, samAccountName, distinguishedName, outFile, simpleOutput, enterprise, dc, Interop.KERB_ETYPE.rc4_hm);
        if (!result && autoenterprise)
        {
            Console.WriteLine("\r\n[-] Retrieving service ticket with SPN failed and 'autoenterprise' passed, retrying with the enterprise principal");
            servicePrincipalName = String.Format("{0}@{1}", samAccountName, domain);
            GetTGSRepHash(TGT, servicePrincipalName, samAccountName, distinguishedName, outFile, simpleOutput, true, dc, Interop.KERB_ETYPE.rc4_hm);
        }
    }
}

```

- **Offline Brute-Force Attack:** The attacker employs offline brute-force techniques, utilizing password cracking tools like **Hashcat** or **John the Ripper**, to attempt to crack the encrypted password hashes.

Skills Assessment

My Workstation

OFFLINE

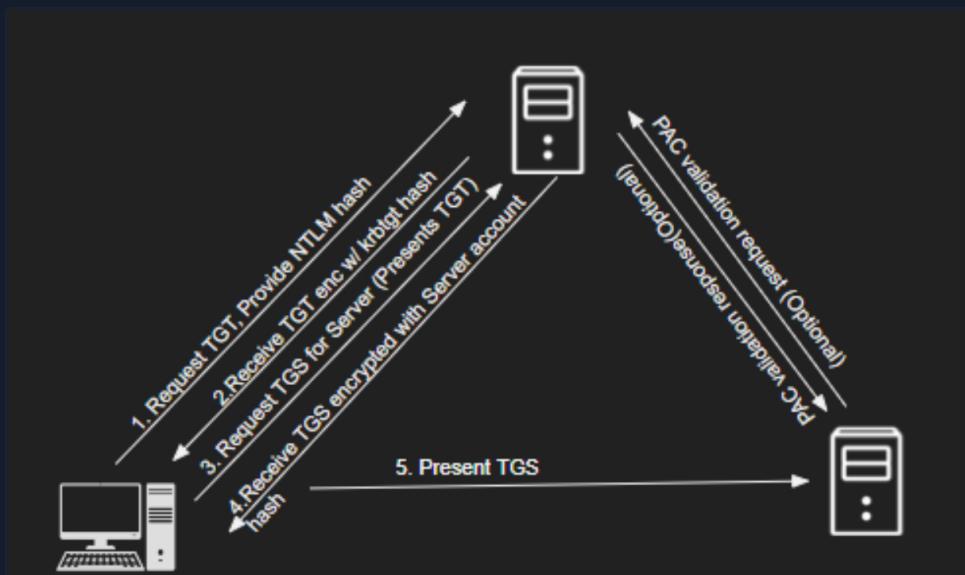
Start Instance

∞ / 1 spawns left

## Benign Service Access Process & Related Events

When a user connects to an **MSSQL (Microsoft SQL Server)** database using a service account with an **SPN**, the following steps occur in the Kerberos authentication process:

- **TGT Request:** The user (client) initiates the authentication process by requesting a Ticket Granting Ticket (TGT) from the Key Distribution Center (KDC), typically part of the Active Directory domain controller.
- **TGT Issue:** The KDC verifies the user's identity (usually through a password hash) and issues a TGT encrypted with the user's secret key. The TGT is valid for a specific period and allows the user to request service tickets without needing to re-authenticate.
- **Service Ticket Request:** The client sends a service ticket request (TGS-REQ) to the KDC for the MSSQL server's SPN using the TGT obtained in the previous step.
- **Service Ticket Issue:** The KDC validates the client's TGT and, if successful, issues a service ticket (TGS) encrypted with the service account's secret key, containing the client's identity and a session key. The client then receives the TGS.
- **Client Connection:** The client connects to the MSSQL server and sends the TGS to the server as part of the authentication process.
- **MSSQL Server Validates the TGS:** The MSSQL server decrypts the TGS using its own secret key to obtain the session key and client identity. If the TGS is valid and the session key is correct, the MSSQL server accepts the client's connection and grants access to the requested resources.



Note that the steps mentioned above can also be observed during network traffic analysis:

18.0.10.100	18.0.10.20	88 TCP	60 50154 -> 88 [ACK] Seq=1 Ack=1 Win=262656 Len=0
18.0.10.100	18.0.10.20	88 KRB5	549 AS-REQ

TGT	10.0.10.20	10.0.10.100	50154 TCP	1514 88 → 50154 [ACK] Seq=1 Ack=296 Win=525568 Len=1460	[TCP segment of a reassembled PDU]
	10.0.10.20	10.0.10.100	50154 KRB5	148 45 → RFP	
	10.0.10.20	10.0.10.20	88 TCP	60 50154 → 88 [ACK] Seq=296 Ack=1555 Win=262656 Len=0	
	10.0.10.20	10.0.10.20	88 TCP	60 50154 → 88 [FIN, ACK] Seq=296 Ack=1555 Win=262656 Len=0	
	10.0.10.20	10.0.10.20	50154 TCP	60 88 → 50154 [ACK] Seq=1555 Ack=296 Win=25568 Len=0	
	10.0.10.20	10.0.10.20	88 TCP	66 50155 → 88 [SYN, ACK] Seq=1578 Ack=1524 Win=262656 Len=0	
	10.0.10.20	10.0.10.20	50155 TCP	66 88 → 50155 [SYN, ACK] Seq=0 Ack=1 Wln=192 Len=0 Ws=1468 Ws=256 SACK_PERM=1	
	10.0.10.20	10.0.10.20	88 TCP	60 50155 → 88 [ACK] Seq=1 Ack=1 Wln=262656 Len=0	
	10.0.10.20	10.0.10.20	50154 TCP	1514 88 → 88 [ACK] Seq=1 Ack=1 Wln=262656 Len=1460	[TCP segment of a reassembled PDU]
TGS	10.0.10.100	10.0.10.20	88 KRB5	171 TGS-REQ	
	10.0.10.20	10.0.10.100	50155 TCP	60 88 → 50155 [ACK] Seq=1 Ack=1578 Win=525568 Len=0	
	10.0.10.20	10.0.10.100	50155 TCP	1514 88 → 50155 [ACK] Seq=1 Ack=1578 Win=525568 Len=1460	[TCP segment of a reassembled PDU]
	10.0.10.20	10.0.10.100	50155 KRB5	117 TGS-REP	
	10.0.10.100	10.0.10.20	88 TCP	60 50155 → 88 [ACK] Seq=1578 Ack=1524 Win=262656 Len=0	
	10.0.10.100	10.0.10.20	88 TCP	60 50155 → 88 [FIN, ACK] Seq=1578 Ack=1524 Win=262656 Len=0	
	10.0.10.100	10.0.10.20	50155 TCP	60 88 → 50155 [ACK] Seq=1524 Ack=1578 Win=525568 Len=0	
	10.0.10.100	10.0.10.20	50155 TCP	60 88 → 50155 [ACK] Seq=1524 Ack=1578 Win=525568 Len=0	
	10.0.10.100	10.0.10.20	88 TCP	1514 88 → 88 [ACK] Seq=384 Ack=1488 Win=262144 Len=1460	[TCP segment of a reassembled PDU]
	10.0.10.100	10.0.10.20	88 TCP	1180 GET / HTTP/1.1	
	10.0.10.20	10.0.10.100	50117 TCP	60 88 → 50117 [ACK] Seq=1488 Ack=2970 Win=525568 Len=0	
Auth	10.0.10.20	10.0.10.100	53306 DR	335 Standard query response 0x009 A download.microsoftupdate.com CMWV.v0.1g-shim.trafficmanager.net CMWV 2.0 3-37	
	10.0.10.20	10.0.10.20	135 TCP	66 49811 → 135 [SYN, ECN, CWR] Seq=0 Wln=8192 Len=0 Ws=1468 Ws=256 SACK_PERM=1	
	10.0.10.20	10.0.10.20	49811 TCP	66 135 → 49811 [SYN, ACK, CWR] Seq=0 Ack=1 Wln=8192 Len=0 Ws=1468 Ws=256 SACK_PERM=1	
	10.0.10.20	10.0.10.20	135 TCP	60 49811 → 135 [ACK] Seq=1 Ack=1 Win=525568 Len=0	
	10.0.10.20	10.0.10.20	135 DCERPC	214 Bind: call_id: 2, Fragment: Single, 3 context items: EPnv4 V3.0 (32bit NDR), EPnv4 V3.0 (64bit NDR), EPnv4 V3.0 (64bit NDR)	
	10.0.10.20	10.0.10.20	49811 DCERPC	162 Bind_ack: call_id: 2, Fragment: Single, max_xmtl: 5840 max_recv: 5840, 3 results: Provider rejection, Acceptan	
	10.0.10.20	10.0.10.20	135 EPM	222 Map request, RPC_NETLOGON, 32bit NDR	
	10.0.10.20	10.0.10.20	49811 EPM	322 Map response, RPC_NETLOGON, 32bit NDR	
	10.0.10.20	10.0.10.20	49669 TCP	66 49669 → 49669 [SYN, ECN, CWR] Seq=0 Wln=8192 Len=0 Ws=1468 Ws=256 SACK_PERM=1	
	10.0.10.20	10.0.10.20	49812 TCP	66 49669 → 49812 [SYN, ACK, CWR] Seq=1 Ack=1 Wln=8192 Len=0 Ws=1468 Ws=256 SACK_PERM=1	
	10.0.10.20	10.0.10.20	49669 TCP	60 49669 → 49669 [ACK] Seq=1 Ack=1 Win=525568 Len=0	
	10.0.10.20	10.0.10.20	49669 DCERPC	204 Bind: call_id: 1, Fragment: Single, 3 context items: RPC_NETLOGON V1.0 (32bit NDR), RPC_NETLOGON V1.0 (64bit NDR), RPC_NETLOGON V1.0 (64bit NDR)	
	10.0.10.20	10.0.10.20	49669 RPC_N...	182 Bind_ack: call_id: 1, Fragment: Single, max_xmtl: 5840 max_recv: 5840, 3 results: Provider rejection, Acceptan	
	10.0.10.20	10.0.10.20	49669 RPC_NE...	686 NetLogonSamLogonEx request	
	10.0.10.20	10.0.10.20	135 TCP	60 49811 → 135 [ACK] Seq=229 Ack=377 Wln=525056 Len=0	
	10.0.10.20	10.0.10.20	49669 TCP	60 49812 → 49669 [ACK] Seq=763 Ack=265 Win=525312 Len=0	
	10.0.10.20	10.0.10.20	50117 TCP	1514 88 → 50117 [ACK] Seq=1488 Ack=2970 Win=525568 Len=1460	[TCP segment of a reassembled PDU]

During the Kerberos authentication process, several security-related events are generated in the Windows Event Log when a user connects to an MSSQL server:

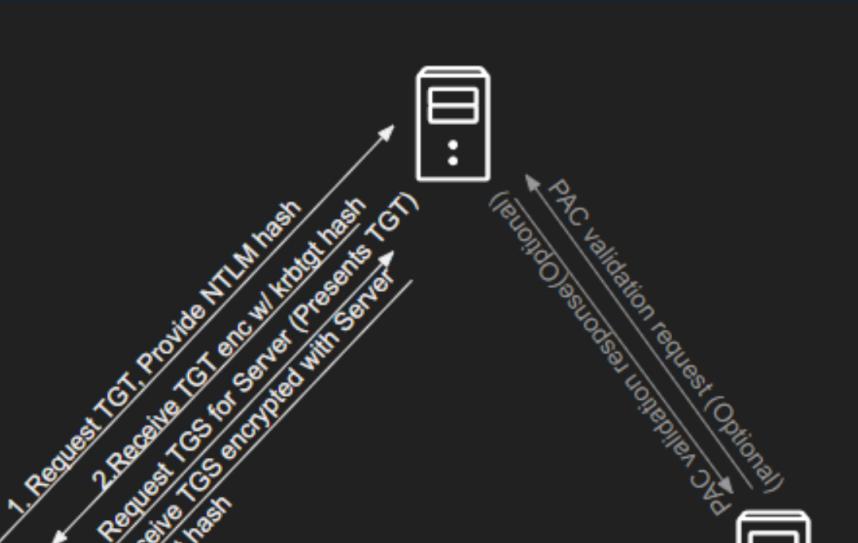
- Event ID 4768 (Kerberos TGT Request):** Occurs when the client workstation requests a TGT from the KDC, generating this event in the Security log on the domain controller.
- Event ID 4769 (Kerberos Service Ticket Request):** Generated after the client receives the TGT and requests a TGS for the MSSQL server's SPN.
- Event ID 4624 (Logon):** Logged in the Security log on the MSSQL server, indicating a successful logon once the client initiates a connection to the MSSQL server and logs in using the service account with the SPN to establish the connection.

'2021-03-11T22:36:38.897284700Z'	iis_svc	LABS	4768	A Kerberos authentication ticket (TGT) was requested	::ffff:10.0.10.100 DC.lab.internal.local krbtgt @x12
'2021-03-11T22:36:38.898666500Z'	iis_svc@LAB_INTERNAL_LOCAL	LAB_INTERNAL_LOCAL	4769	A Kerberos service ticket was requested	::ffff:10.0.10.100 DC.lab.internal.local iis_svc @x17
'2021-03-11T22:36:39.050378400Z'	iis_svc	LAB_INTERNAL_LOCAL	4624	An account was successfully logged on	10.0.10.100 iis.lab.internal.local
'2021-03-11T22:36:39.4137226Z'	iis_svc	LAB_INTERNAL_LOCAL	4648	A logon was attempted using explicit credentials	10.0.10.21 BLUE.lab.internal.local

## Kerberoasting Detection Opportunities

Since the initial phase of Kerberoasting involves identifying target service accounts, monitoring LDAP activity, as explained in the domain reconnaissance section, can help in identifying suspicious LDAP queries.

An alternative approach focuses on the difference between benign service access and a Kerberoasting attack. In both scenarios, TGS tickets for the service will be requested, but only in the case of benign service access will the user connect to the server and present the TGS ticket.



3/  
4.Reg  
account

5. Present TGS



Detection logic entails finding all events for TGS requests and logon events from the same user, then identifying instances where a TGS request is present without a subsequent logon event. In the case of IIS service access using a service account with an SPN, an additional **4648 (A logon was attempted using explicit credentials)** event will be generated as a logon event.

Let's now navigate to the bottom of this section and click on "Click here to spawn the target system!". Then, access the Splunk interface at [http://\[Target IP\]:8000](http://[Target IP]:8000) and launch the Search & Reporting Splunk application. The vast majority of searches covered from this point up to end of this section can be replicated inside the target, offering a more comprehensive grasp of the topics presented.

## Detecting Kerberoasting With Splunk

Now let's explore how we can identify Kerberoasting, using Splunk.

### Benign TGS Requests

First, let's see some benign TGS requests in Splunk.

**Timeframe:** earliest=1690388417 latest=1690388630

```
index=main earliest=1690388417 latest=1690388630 EventCode=4648 OR (EventCode=4769 AND service_name=iis_svc)
| dedup RecordNumber
| rex field=user "(?<username>[^@]+)"
| table _time, ComputerName, EventCode, name, username, Account_Name, Account_Domain, src_ip, service_name, Ticket_Options, Ticket_Encryption_Type, Target_Server_Name, Additional_Information
```

_time	ComputerName	EventCode	username	Account_Name	Account_Domain	src_ip	service_name	Ticket_Options
2023-07-26 16:23:15	BLUE.corp.local	4648	A logon was attempted using explicit credentials	TAYLOR_BENTON	TAYLOR_BENTON		CORP	
2023-07-26 16:23:15	DC01.corp.local	4769	A Kerberos service ticket was requested	TAYLOR_BENTON	TAYLOR_BENTON@CORP.LOCAL	CORP.LOCAL	iis_svc	0x40810000

### Search Breakdown:

- **index=main earliest=1690388417 latest=1690388630**: This filters the search to only include events from the main index that occurred between the specified earliest and latest epoch timestamps.
- **EventCode=4648 OR (EventCode=4769 AND service\_name=iis\_svc)**: This further filters the search to only include events with an **EventCode** of **4648** or an **EventCode** of **4769** with a **service\_name** of **iis\_svc**.
- **| dedup RecordNumber**: This removes duplicate events based on the **RecordNumber** field.
- **| rex field=user "(?<username>[^@]+)"**: This extracts the **username** portion of the **user** field using a regular expression and stores it in a new field called **username**.
- **| table \_time, ComputerName, EventCode, name, username, Account\_Name, Account\_Domain, src\_ip,**

```
service_name, Ticket_Options, Ticket_Encryption_Type, Target_Server_Name,
Additional_Information: This displays the specified fields in tabular format.
```

## Detecting Kerberoasting - SPN Querying

Timeframe: earliest=1690448444 latest=1690454437

```
index=main earliest=1690448444 latest=1690454437 source="WinEventLog:SilkService-Log"
| spath input=Message
| rename XmlEventData.* as *
| table _time, ComputerName, ProcessName, DistinguishedName, SearchFilter
| search SearchFilter="*(&(samAccountType=805306368)(servicePrincipalName=*))"
```

_time	ComputerName	ProcessName	DistinguishedName	SearchFilter
2023-07-27 09:34:29	BLUE.corp.local	rundll32	DC=corp,DC=local	(&(samAccountType=805306368)(servicePrincipalName=iis_svc) (!UserAccountControl:1.2.840.113556.1.4.803:=2))
2023-07-27 09:34:29	BLUE.corp.local	rundll32	DC=corp,DC=local	(&(samAccountType=805306368)(servicePrincipalName=iis_svc) (!UserAccountControl:1.2.840.113556.1.4.803:=2))
2023-07-27 09:33:38	BLUE.corp.local	N/A	DC=corp,DC=local	(&(samAccountType=805306368)(servicePrincipalName=iis_svc) (!UserAccountControl:1.2.840.113556.1.4.803:=2))
2023-07-27 09:33:38	BLUE.corp.local	N/A	DC=corp,DC=local	(&(samAccountType=805306368)(servicePrincipalName=iis_svc) (!UserAccountControl:1.2.840.113556.1.4.803:=2))

## Detecting Kerberoasting - TGS Requests

Timeframe: earliest=1690450374 latest=1690450483

```
index=main earliest=1690450374 latest=1690450483 EventCode=4648 OR (EventCode=4769 AND service_name
| dedup RecordNumber
| rex field=user "(?<username>[^@]+)"
| bin span=2m _time
| search username!=*$
| stats values(EventCode) as Events, values(service_name) as service_name, values(Additional_Information) as Additional_Information
| where !match(Events, "4648")
```

_time	username	Events	service_name	Additional_Information	Target_Server_Name
2023-07-27 09:34:00	JOLENE_MCGEE	4769	iis_svc		

### Search Breakdown:

- index=main earliest=1690450374 latest=1690450483 EventCode=4648 OR (EventCode=4769 AND service\_name=iis\_svc):** Filters the search to only include events from the `main` index that occurred between the specified earliest and latest epoch timestamps. It further filters the search to only include events with an `EventCode` of `4648` or an `EventCode` of `4769` with a `service_name` of `iis_svc`.
- | dedup RecordNumber:** Removes duplicate events based on the `RecordNumber` field.
- | rex field=user "(?<username>[^@]+)":** Extracts the `username` portion of the `user` field using a regular expression and stores it in a new field called `username`.

- `| bin span=2m _time`: Bins the events into 2-minute intervals based on the `_time` field.
- `| search username!=*$`: Filters out events where the `username` field ends with a `$`.
- `| stats values(EventCode) as Events, values(service_name) as service_name, values(Additional_Information) as Additional_Information, values(Target_Server_Name) as Target_Server_Name by _time, username`: Groups the events by the `_time` and `username` fields, and creates new fields that contain the `unique` values of the `EventCode`, `service_name`, `Additional_Information`, and `Target_Server_Name` fields within each group.
- `| where !match(Events,"4648")`: Filters out events that have the value `4648` in the `Events` field.

## Detecting Kerberoasting Using Transactions - TGS Requests

Timeframe: earliest=1690450374 latest=1690450483

```

index=main earliest=1690450374 latest=1690450483 EventCode=4648 OR (EventCode=4769 AND service_name=iis_svc)
| dedup RecordNumber
| rex field=user "(?<username>[^@]+)"
| search username!=$
| transaction username keepevicted=true maxspan=5s endswith=(EventCode=4648) startswith=(EventCode=4769)
| where closed_txn=0 AND EventCode = 4769
| table _time, EventCode, service_name, username

```

Time	EventCode	service_name	username
2023-07-27 09:34:28	4769	iis_svc	JOLENE_MCGEE

### Search Breakdown:

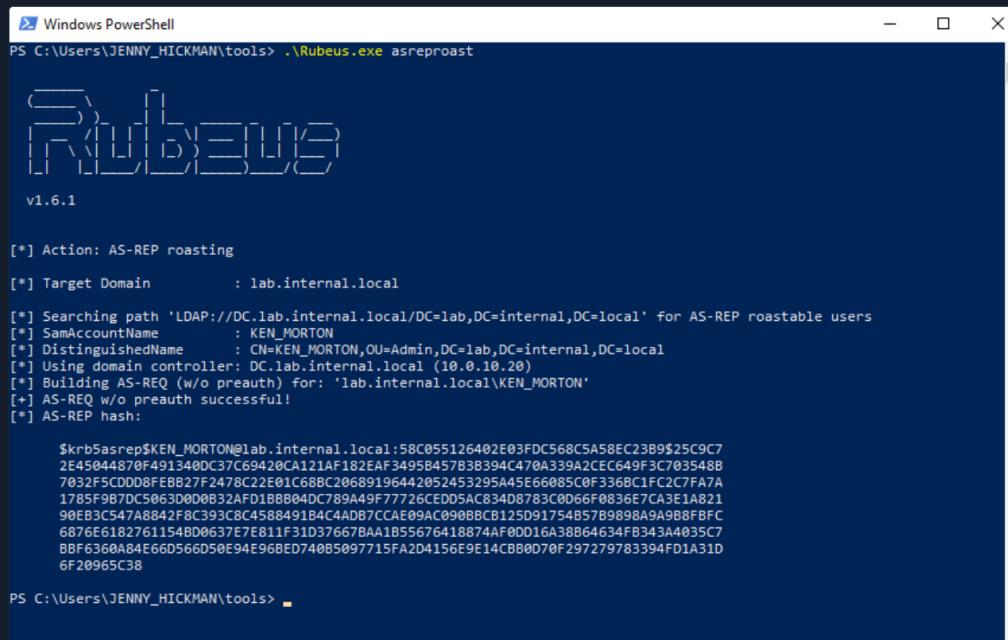
This Splunk search query is different from the previous query primarily due to the use of the `transaction` command, which groups events into transactions based on specified fields and criteria.

- `index=main earliest=1690450374 latest=1690450483 EventCode=4648 OR (EventCode=4769 AND service_name=iis_svc)`: Filters the search to only include events from the `main` index that occurred between the specified earliest and latest epoch timestamps. It further filters the search to only include events with an `EventCode` of `4648` or an `EventCode` of `4769` with a `service_name` of `iis_svc`.
- `| dedup RecordNumber`: Removes duplicate events based on the `RecordNumber` field.
- `| rex field=user "(?<username>[^@]+)"`: Extracts the `username` portion of the `user` field using a regular expression and stores it in a new field called `username`.
- `| search username!=$`: Filters out events where the `username` field ends with a `$`.
- `| transaction username keepevicted=true maxspan=5s endswith=(EventCode=4648) startswith=(EventCode=4769)`: Groups events into `transactions` based on the `username` field. The `keepevicted=true` option includes events that do not meet the transaction criteria. The `maxspan=5s` option sets the maximum time duration of a transaction to 5 seconds. The `endswith=(EventCode=4648)` and `startswith=(EventCode=4769)` options specify that transactions should start with an event with `EventCode` `4769` and end with an event with `EventCode` `4648`.
- `| where closed_txn=0 AND EventCode = 4769`: Filters the results to only include transactions that are not closed (`closed_txn=0`) and have an `EventCode` of `4769`.
- `| table _time, EventCode, service_name, username`: Displays the remaining events in tabular format with the specified fields.

This query focuses on identifying events with an **EventCode** of **4769** that are part of an incomplete transaction (i.e., they did not end with an event with **EventCode 4648** within the **5-second window**).

## AS-REP Roasting

**AS-REP Roasting** is a technique used in Active Directory environments to target user accounts without pre-authentication enabled. In Kerberos, pre-authentication is a security feature requiring users to prove their identity before the TGT is issued. However, certain user accounts, such as those with unconstrained delegation, do not have pre-authentication enabled, making them susceptible to AS-REP Roasting attacks.



```
Windows PowerShell
PS C:\Users\JENNY_HICKMAN\tools> .\Rubeus.exe asreproast

v1.6.1

[*] Action: AS-REP roasting
[*] Target Domain      : lab.internal.local
[*] Searching path 'LDAP://DC.lab.internal.local/DC=lab,DC=internal,DC=local' for AS-REP roastable users
[*] SamAccountName     : KEN_MORTON
[*] DistinguishedName  : CN=KEN_MORTON,OU=Admin,DC=lab,DC=internal,DC=local
[*] Using domain controller: DC.lab.internal.local (10.0.10.20)
[*] Building AS-REQ (w/o preauth) for: 'lab.internal.local\KEN_MORTON'
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:

$krbtgsasrep$KEN_MORTON@lab.internal.local:58C055126402E03FDC568C5A58EC23B9$25C9C7
2E45044870F491340DC37C69420CA121AF182EAF3495B45783B394C470A339A2CEC649F3C703548B
7032F5C0D08FE8B27F2478C22E01C68BC20689196442052453295A45E66085C0F336BC1FC2C7FA7A
1785F987DC5063D000832AF1BBB04DC789A49F77726CE05AC834D8783C0D66F0836E7CA3E1A821
90EBC3C547A8842F8C393C8C4588491B4C4AD87CCA09AC090BBCB125D91754857B9898A9A98FBFC
6876E61827611548D00637E7E811F31D376678AA1B55676418874AF0DD16A38B64634FB343A4035C7
8BF6360A84E66D566D50E94E96BED740B5097715FA2D4156E9E14CB80D70F297279783394FD1A31D
6F20965C38

PS C:\Users\JENNY_HICKMAN\tools>
```

### Attack Steps:

- **Identify Target User Accounts:** The attacker identifies user accounts without pre-authentication enabled. The following is a code snippet from **Rubeus** that is related to this step.

```
try
{
    string userSearchFilter = "";

    if (String.IsNullOrEmpty(userName))
    {
        userSearchFilter = "(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304))";
    }
    else
    {
        userSearchFilter = String.Format("(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304)(samAccountName={0}))", userName);
    }
    if (!String.IsNullOrEmpty(ldapFilter))
    {
        userSearchFilter = String.Format("(&{0}({1}))", userSearchFilter, ldapFilter);
    }
    userSearcher.Filter = userSearchFilter;
}
catch (Exception ex)
{
    Console.WriteLine("\r\n[X] Error settings the domain searcher filter: {0}", ex.InnerException.Message);
    return;
}
```

- **Request AS-REQ Service Tickets:** The attacker initiates an AS-REQ service ticket request for each identified target user account. The following is a code snippet from **Rubeus** that is related to this step.

```
try
{
    SearchResultCollection users = userSearcher.FindAll();

    if (users.Count == 0)
    {
        Console.WriteLine("[X] No users found to AS-REP roast!");
    }
}
```

```

        foreach (SearchResult user in users)
    {
        string samAccountName = user.Properties["samAccountName"][0].ToString();
        string distinguishedName = user.Properties["distinguishedName"][0].ToString();
        Console.WriteLine("[*] SamAccountName      : {0}", samAccountName);
        Console.WriteLine("[*] DistinguishedName   : {0}", distinguishedName);

        GetASRepHash(samAccountName, domain, domainController, format, outFile);
    }
}

```

- **Offline Brute-Force Attack:** The attacker captures the encrypted TGTs and employs offline brute-force techniques to attempt to crack the password hashes.

## Kerberos Pre-Authentication

**Kerberos pre-authentication** is an additional security mechanism in the Kerberos authentication protocol enhancing user credentials protection during the authentication process. When a user tries to access a network resource or service, the client sends an authentication request AS-REQ to the KDC.

If pre-authentication is enabled, this request also contains an encrypted timestamp (**pa-ENC-TIMESTAMP**). The KDC attempts to decrypt this timestamp using the user password hash and, if successful, issues a TGT to the user.

11576 2021-03-06 18:35:57.601432	10.0.10.101	10.0.10.20	88	KRB5	384	AS-REQ
11578 2021-03-06 18:35:57.601868	10.0.10.20	10.0.10.101	51662	KRB5	342	AS-REP
11587 2021-03-06 18:35:57.602500	10.0.10.101	10.0.10.20	88	KRB5	392	TGS-REQ
11590 2021-03-06 18:35:57.603178	10.0.10.20	10.0.10.101	51663	KRB5	359	TGS-REP
11596 2021-03-06 18:35:57.603652	10.0.10.101	10.0.10.20	389	LDAP	582	bindRequest(3) "<ROOT>" sasl
11598 2021-03-06 18:35:57.604066	10.0.10.20	10.0.10.101	51660	LDAP	265	bindResponse(3) success
11625 2021-03-06 18:35:57.816636	10.0.10.101	10.0.10.20	5985	HTTP	88	POST /wsman/SubscriptionManager/WEC HTTP/1.1

> Frame 11576: 384 bytes on wire (3072 bits), 384 bytes captured (3072 bits)  
> Ethernet II, Src: VMware\_f6:37:87 (00:0c:29:f6:37:87), Dst: VMware\_80:c7:b1 (00:0c:29:80:c7:b1)  
> Internet Protocol Version 4, Src: 10.0.10.101, Dst: 10.0.10.20  
> Transmission Control Protocol, Src Port: 51662, Dst Port: 88, Seq: 1, Ack: 1, Len: 330  
 ✓ Kerberos  
 > Record Mark: 326 bytes  
 ✓ as-req  
 > pvn: 5  
 msg-type: krb-as-req (10)  
 > padata: 2 items  
 > PA-DATA pa-ENC-TIMESTAMP  
 > PA-DATA pa-PAC-REQUEST  
 ✓ req-body  
 > Padding: 0  
 > kdc-options: 40810010  
 ✓ cname  
 > name-type: KRB5-NT-PRINCIPAL (1)  
 > cname-string: 1 item  
 > CNameString: JENNY\_HICKMAN  
 realm: LAB.INTERNAL.LOCAL  
 > sname  
 till: 2037-09-13 02:48:05 (UTC)  
 rtime: 2037-09-13 02:48:05 (UTC)  
 nonce: 205710620  
 > etype: 6 items  
 ✓ addresses: 1 item ORANGE<20>  
 > HostAddress ORANGE<20>

When pre-authentication is disabled, there is no timestamp validation by the KDC, allowing users to request a TGT ticket without knowing the user password.

11568 2021-03-06 18:35:57.600566	10.0.10.101	10.0.10.20	88	KRB5	304	AS-REQ
11569 2021-03-06 18:35:57.600932	10.0.10.20	10.0.10.101	51661	KRB5	284	KRB Error: KRB5KDC_ERR_PREAMUTH_REQUIRED
11576 2021-03-06 18:35:57.601432	10.0.10.101	10.0.10.20	88	KRB5	384	AS-REQ
11578 2021-03-06 18:35:57.601868	10.0.10.20	10.0.10.101	51662	KRB5	342	AS-REP
11587 2021-03-06 18:35:57.602500	10.0.10.101	10.0.10.20	88	KRB5	392	TGS-REQ
11590 2021-03-06 18:35:57.603178	10.0.10.20	10.0.10.101	51663	KRB5	359	TGS-REP
11596 2021-03-06 18:35:57.603652	10.0.10.101	10.0.10.20	389	LDAP	582	bindRequest(3) "<ROOT>" sasl
11598 2021-03-06 18:35:57.604066	10.0.10.20	10.0.10.101	51660	LDAP	265	bindResponse(3) success
11625 2021-03-06 18:35:57.816636	10.0.10.101	10.0.10.20	5985	HTTP	88	POST /wsman/SubscriptionManager/WEC HTTP/1.1

> Frame 11568: 304 bytes on wire (2432 bits), 304 bytes captured (2432 bits)  
> Ethernet II, Src: VMware\_f6:37:87 (00:0c:29:f6:37:87), Dst: VMware\_80:c7:b1 (00:0c:29:80:c7:b1)  
> Internet Protocol Version 4, Src: 10.0.10.101, Dst: 10.0.10.20  
> Transmission Control Protocol, Src Port: 51661, Dst Port: 88, Seq: 1, Ack: 1, Len: 250  
 ✓ Kerberos  
 > Record Mark: 246 bytes  
 ✓ as-req  
 > pvn: 5  
 msg-type: krb-as-req (10)  
 > padata: 1 item  
 > PA-DATA pa-PAC-REQUEST  
 ✓ req-body  
 > Padding: 0  
 > kdc-options: 40810010  
 ✓ cname  
 > name-type: KRB5-NT-PRINCIPAL (1)  
 > cname-string: 1 item  
 > CNameString: JENNY\_HICKMAN  
 realm: LAB.INTERNAL.LOCAL  
 > sname  
 till: 2037-09-13 02:48:05 (UTC)  
 rtime: 2037-09-13 02:48:05 (UTC)

```
nonce: 205710609
> etype: 6 items
> addresses: 1 item ORANGE<20>
> HostAddress ORANGE<20>
```

## AS-REPRoasting Detection Opportunities

Similar to Kerberoasting, the initial phase of AS-REPRoasting involves identifying user accounts with unconstrained delegation enabled or accounts without pre-authentication, which can be detected by LDAP monitoring.

Kerberos authentication Event ID 4768 (TGT Request) contains a PreAuthType attribute in the additional information part of the event indicating whether pre-authentication is enabled for an account.

## Detecting AS-REPRoasting With Splunk

Now let's explore how we can identify AS-REPRoasting, using Splunk.

### Detecting AS-REPRoasting - Querying Accounts With Pre-Auth Disabled

Timeframe: earliest=1690392745 latest=1690393283

```
● ● ● Detecting Kerberoasting/AS-REProasting

index=main earliest=1690392745 latest=1690393283 source="WinEventLog:SilkService-Log"
| spath input=Message
| rename XmlEventData.* as *
| table _time, ComputerName, ProcessName, DistinguishedName, SearchFilter
| search SearchFilter="*(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304)"
```

New Search

Save As ▾ Create Table View Close

Last 24 hours

```
1 index=main earliest=1690392745 latest=1690393283 source="WinEventLog:SilkService-Log"
2 | spath input=Message
3 | rename XmlEventData.* as *
4 | table _time, ComputerName, ProcessName, DistinguishedName, SearchFilter
5 | search SearchFilter="*(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304)"
```

✓ 2 events (7/26/23 5:32:25.000 PM to 7/26/23 5:41:23.000 PM)

No Event Sampling

Events Patterns Statistics (2) Visualization

20 Per Page ▾ Format Preview ▾

_time	ComputerName	ProcessName	DistinguishedName	SearchFilter
2023-07-26 17:41:12	BLUE.corp.local	Rubeus	DC=corp,DC=local	(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304))
2023-07-26 17:41:12	BLUE.corp.local	Rubeus	DC=corp,DC=local	(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=4194304))

### Detecting AS-REPRoasting - TGT Requests For Accounts With Pre-Auth Disabled

Timeframe: earliest=1690392745 latest=1690393283

```
● ● ● Detecting Kerberoasting/AS-REProasting

index=main earliest=1690392745 latest=1690393283 source="WinEventLog:Security" EventCode=4768 Pre_A
| rex field=src_ip "(?:\.:ffff\:)?(?:<src_ip>[0-9\.]+)"
| table _time, src_ip, user, Pre_Authentication_Type, Ticket_Options, Ticket_Encryption_Type
```

New Search

Save As ▾ Create Table View Close

Last 24 hours

```
1 index=main earliest=1690392745 latest=1690393283 source="WinEventLog:Security" EventCode=4768
  Pre_Authentication_Type=0
2 | rex field=src_ip "(?:\.:ffff\:)?(?:<src_ip>[0-9\.]+)"
3 | table _time, src_ip, user, Pre_Authentication_Type, Ticket_Options, Ticket_Encryption_Type
```

✓ 320 events (7/26/23 5:32:25.000 PM to 7/26/23 5:41:23.000 PM)

No Event Sampling

Events Patterns Statistics (320) Visualization

_time	src_ip	user	Pre.Authentication_Type	Ticket_Options	Ticket_Encryption_Type
2023-07-26 17:41:20	10.10.0.101	TAMI_DANIEL	0	0x40800010	0x17
2023-07-26 17:41:20	10.10.0.101	CELIA_RAMIREZ	0	0x40800010	0x17
2023-07-26 17:41:20	10.10.0.101	ELISEO_MOODY	0	0x40800010	0x17
2023-07-26 17:41:20	10.10.0.101	LORA_CHANG	0	0x40800010	0x17
2023-07-26 17:41:20	10.10.0.101	CLAYTON_ROY	0	0x40800010	0x17
2023-07-26 17:41:20	10.10.0.101	JAYSON_BUSH	0	0x40800010	0x17
2023-07-26 17:41:20	10.10.0.101	CHARLOTTE_FORBES	0	0x40800010	0x17
2023-07-26 17:41:20	10.10.0.101	ROSENDY_HARRELL	0	0x40800010	0x17
2023-07-26 17:41:19	10.10.0.101	ANTHONY_GUERRA	0	0x40800010	0x17
2023-07-26 17:41:19	10.10.0.101	LIZA_HAWKINS	0	0x40800010	0x17
2023-07-26 17:41:19	10.10.0.101	LORAINA_BYRD	0	0x40800010	0x17
2023-07-26 17:41:19	10.10.0.101	JOHNATHAN_BROWN	0	0x40800010	0x17
2023-07-26 17:41:19	10.10.0.101	NORMAN_SCHWARTZ	0	0x40800010	0x17
2023-07-26 17:41:19	10.10.0.101	BENNIE_FORD	0	0x40800010	0x17
2023-07-26 17:41:19	10.10.0.101	DEANA_KELLY	0	0x40800010	0x17
2023-07-26 17:41:19	10.10.0.101	ZACHERY_ALEXANDER	0	0x40800010	0x17
2023-07-26 17:41:19	10.10.0.101	KORY_MONToya	0	0x40800010	0x17

#### Search Breakdown:

- `index=main earliest=1690392745 latest=1690393283 source="WinEventLog:Security" EventCode=4768`  
`Pre.Authentication_Type=0`: Filters the search to only include events from the `main` index that occurred between the specified earliest and latest epoch timestamps. It further filters the search to only include events with a source of `WinEventLog:Security`, an `EventCode` of `4768`, and a `Pre.Authentication_Type` of `0`.
- `| rex field=src_ip "(\\:ffff\\:)?(<src_ip>[0-9\\.]+)":` Uses a regular expression to extract the `src_ip` (source IP address) field. The expression matches an optional `::ffff:` prefix followed by an IP address in dotted decimal notation. This step handles IPv4-mapped IPv6 addresses by extracting the IPv4 portion.
- `| table _time, src_ip, user, Pre.Authentication_Type, Ticket_Options, Ticket_Encryption_Type:`  
 Displays the remaining events in tabular format with the specified fields.

#### VPN Servers

⚠ Warning: Each time you "Switch", your connection keys are regenerated and you must re-download your VPN connection file.

All VM instances associated with the old VPN Server will be terminated when switching to a new VPN server.

Existing PwnBox instances will automatically switch to the new VPN server.

US Academy 3

Medium Load

#### PROTOCOL

UDP 1337    TCP 443

DOWNLOAD VPN CONNECTION FILE



#### Connect to Pwnbox

Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK

139ms

ⓘ Terminate Pwnbox to switch location

[Start Instance](#)

∞ / 1 spawns left



Waiting to start...

Enable step-by-step solutions for all questions  



## Questions

Answer the question(s) below to complete this Section and earn cubes!

 Download VPN Connection File



Target(s): [Click here to spawn the target system!](#)

+ 1  Modify and employ the Splunk search provided at the "Detecting Kerberoasting - SPN Querying" part of this section on all ingested data (All time). Enter the name of the user who initiated the process that executed an LDAP query containing the "\*(&(samAccountType=805306368)(servicePrincipalName=\*))" string at 2023-07-26 16:42:44 as your answer. Answer format: CORP\\_\_

CORP\LANDON\_HINES

 Submit



 Previous

Next 

 Mark Complete & Next



Powered by 

