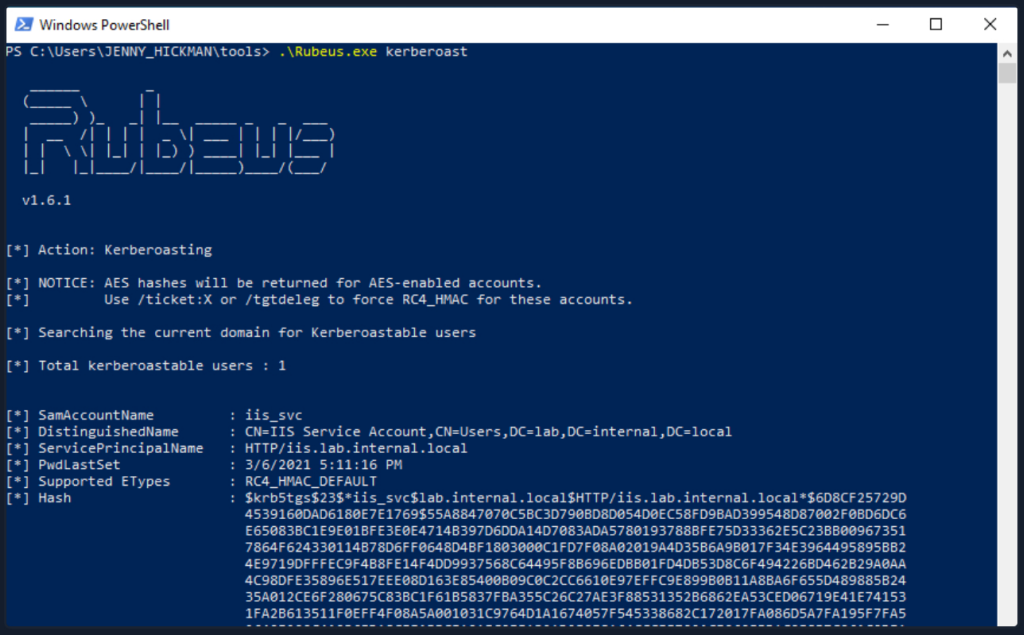
Detecting Kerberoasting/AS-REProasting

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.



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.ToFileTime() + ")(pwdlastset<=" + timeUntilConverted.ToFileTime() + ")";
        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})", 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)
```

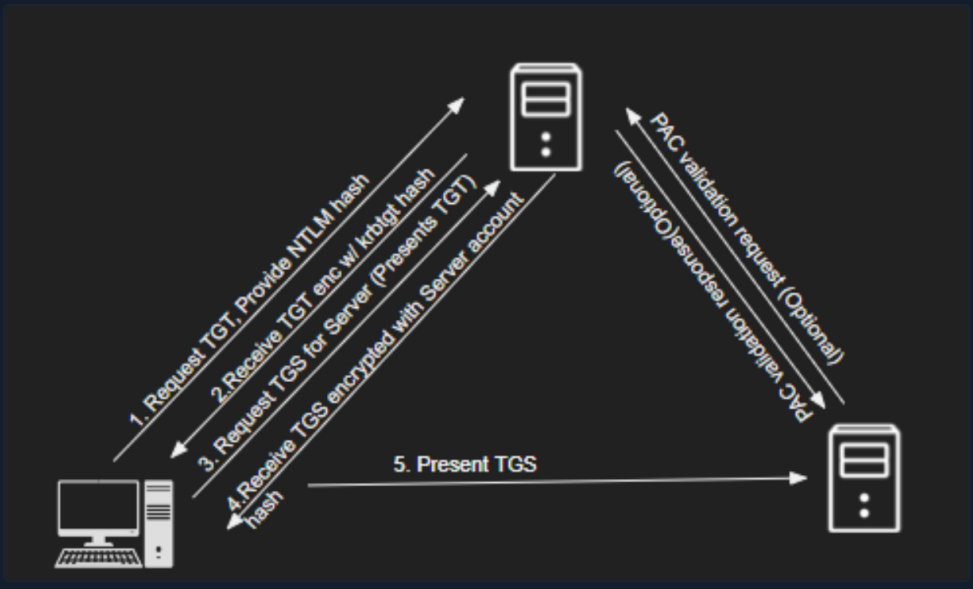
```
{
    // if a TGT .kirbi is supplied, use that for the request
    //      this could be a passed TGT or if TGT delegation is specified

    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.K
        if (!result && autoenterprise)
        {
            Console.WriteLine("\r\n[-] Retrieving service ticket with SPN failed and '/autoenterprise' passed, retrying with the enterprise princi
            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.

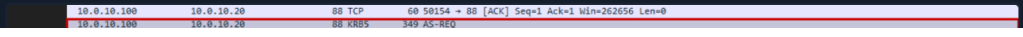
Benign Service Access Process & Related Events

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:



Skills Assessment

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

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 AS-REP
10.0.10.100	10.0.10.20	88	TCP	60 50154 → 88 [ACK] Seq=296 Ack=1555 Win=262656 Len=0
10.0.10.100	10.0.10.20	88	TCP	60 50154 → 88 [FIN, ACK] Seq=296 Ack=1555 Win=262656 Len=0
10.0.10.20	10.0.10.100	50154	TCP	60 88 → 50154 [ACK] Seq=1555 Ack=297 Win=525568 Len=0
10.0.10.20	10.0.10.100	50154	TCP	60 88 → 50154 [RST, ACK] Seq=1555 Ack=297 Win=0 Len=0
10.0.10.100	10.0.10.20	88	TCP	66 50155 → 88 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
10.0.10.20	10.0.10.100	50155	TCP	66 88 → 50155 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
10.0.10.100	10.0.10.20	88	TCP	60 50155 → 88 [ACK] Seq=1 Ack=1 Win=262656 Len=0
10.0.10.100	10.0.10.20	88	TCP	1514 50155 → 88 [ACK] Seq=1 Ack=1 Win=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
10.0.10.20	10.0.10.100	50155	TCP	60 88 → 50155 [ACK] Seq=1524 Ack=1579 Win=525568 Len=0
10.0.10.20	10.0.10.100	50155	TCP	60 88 → 50155 [RST, ACK] Seq=1524 Ack=1579 Win=0 Len=0

Auth

10.0.10.100	10.0.10.21	80	TCP	1514 50117 → 80 [ACK] Seq=384 Ack=1488 Win=262144 Len=1460 [TCP segment of a reassembled PDU]
10.0.10.100	10.0.10.21	80	HTTP	1180 GET / HTTP/1.1
10.0.10.21	10.0.10.100	50117	TCP	60 80 → 50117 [ACK] Seq=1488 Ack=2970 Win=525568 Len=0
10.0.10.20	10.0.10.100	51306	DNS	335 Standard query response 0xe598 A download.windowsupdate.com CNAME wu-fg-shim.trafficmanager.net CNAME 2-01-3cf
10.0.10.21	10.0.10.20	135	TCP	66 49811 → 135 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
10.0.10.20	10.0.10.21	49811	TCP	66 135 → 49811 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
10.0.10.21	10.0.10.20	135	TCP	60 49811 → 135 [ACK] Seq=1 Ack=1 Win=525568 Len=0
10.0.10.21	10.0.10.20	135	DCERPC	214 Bind: call_id: 2, Fragment: Single, 3 context items: EPMv4 V3.0 (32bit NDR), EPMv4 V3.0 (64bit NDR), EPMv4 V3.
10.0.10.20	10.0.10.21	49811	DCERPC	162 Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_recv: 5840, 3 results: Provider rejection, Acceptan
10.0.10.21	10.0.10.20	135	EPM	222 Map request, RPC_NETLOGON, 32bit NDR
10.0.10.20	10.0.10.21	49811	EPM	322 Map response, RPC_NETLOGON, 32bit NDR, RPC_NETLOGON, 32bit NDR
10.0.10.21	10.0.10.20	49669	TCP	66 49812 → 49669 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
10.0.10.20	10.0.10.21	49812	TCP	66 49669 → 49812 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
10.0.10.21	10.0.10.20	49669	TCP	60 49812 → 49669 [ACK] Seq=1 Ack=1 Win=525568 Len=0
10.0.10.21	10.0.10.20	49669	DCERPC	264 Bind: call_id: 2, Fragment: Single, 3 context items: RPC_NETLOGON V1.0 (32bit NDR), RPC_NETLOGON V1.0 (64bit N
10.0.10.20	10.0.10.21	49812	DCERPC	182 Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_recv: 5840, 3 results: Provider rejection, Acceptan
10.0.10.21	10.0.10.20	49669	RPC_NE…	606 NetrLogonSamLogonEx request
10.0.10.20	10.0.10.21	49812	RPC_NE…	190 NetrLogonSamLogonEx response
10.0.10.21	10.0.10.20	135	TCP	60 49811 → 135 [ACK] Seq=329 Ack=377 Win=525056 Len=0
10.0.10.21	10.0.10.20	49669	TCP	60 49812 → 49669 [ACK] Seq=763 Ack=265 Win=525312 Len=0
10.0.10.21	10.0.10.100	50117	TCP	1514 80 → 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:

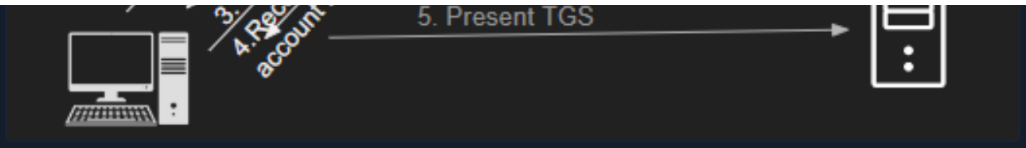
- `Event ID 4768 (Kerberos TGT Request)`: Occurs when the client workstation requests a TGT from the KDC, generating this event in the Security log on the domain controller.
- `Event ID 4769 (Kerberos Service Ticket Request)`: Generated after the client receives the TGT and requests a TGS for the MSSQL 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	0x12
'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	0x17
'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.



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 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.

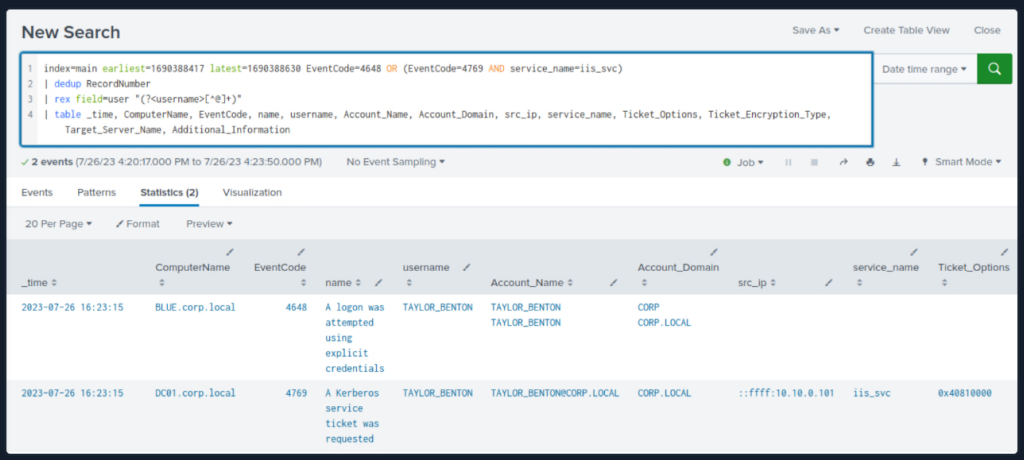
Benign TGS Requests

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

Timeframe: `earliest=1690388417 latest=1690388630`

Detecting Kerberoasting/AS-REProasting

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



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,

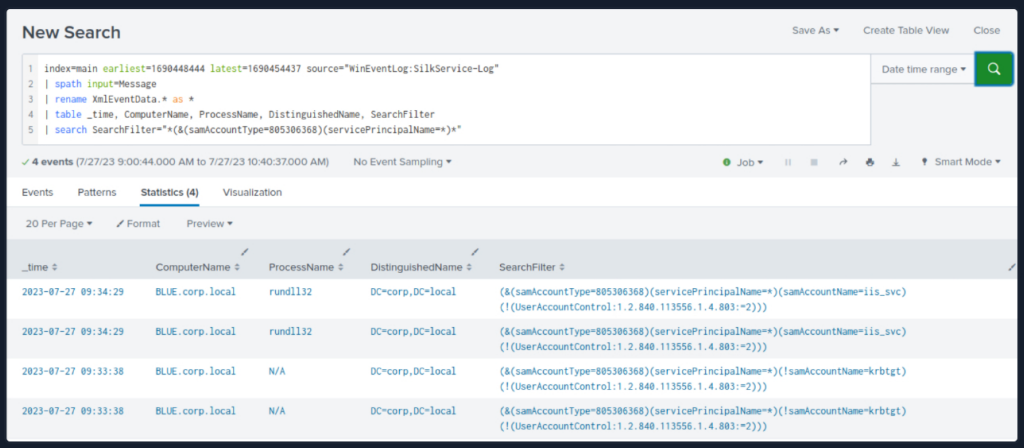
Additional_Information: This displays the specified fields in tabular format.

Detecting Kerberoasting - SPN Querying

Timeframe: earliest=1690448444 latest=1690454437

Detecting Kerberoasting/AS-REProasting

```
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=*)*"
```

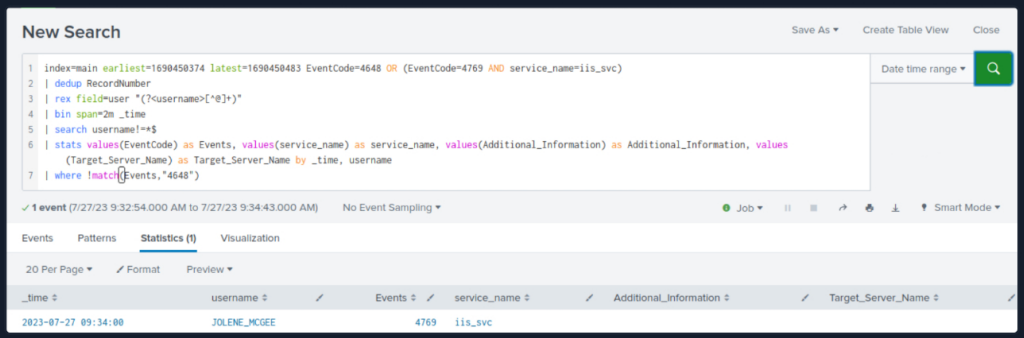


Detecting Kerberoasting - TGS Requests

Timeframe: earliest=1690450374 latest=1690450483

Detecting Kerberoasting/AS-REProasting

```
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_Inform
| where !match(Events,"4648")
```



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

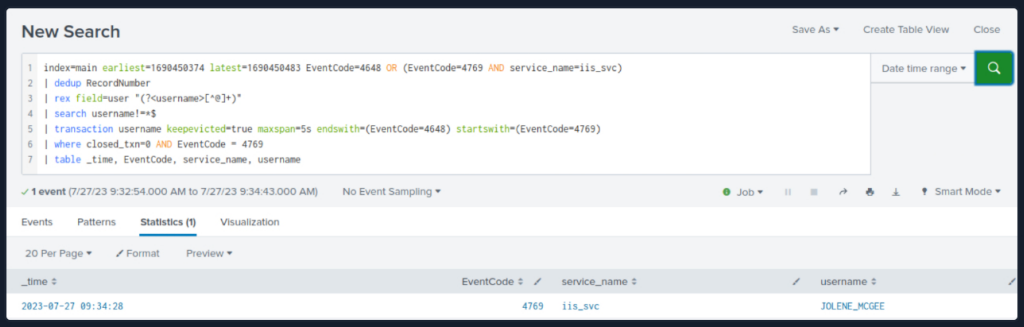
- `| bin span=2m _time`: Bins the events into 2-minute intervals based on the `_time` field.
- `| search username!=*$`: Filters out events where the `username` field ends with a `$`.
- `| stats values(EventCode) as Events, values(service_name) as service_name, values(Additional_Information) as Additional_Information, 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`

Detecting Kerberoasting/AS-REProasting

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



Search Breakdown:

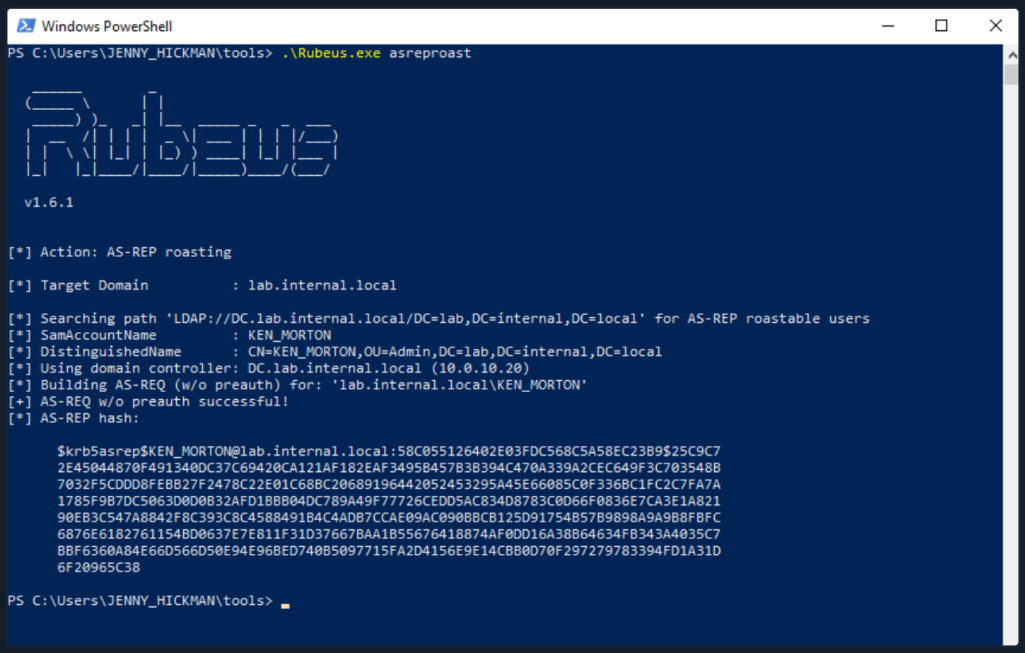
This Splunk search query is different from the previous query primarily due to the use of the `transaction` command, which groups events into transactions based on specified fields and criteria.

- `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-REPRoasting

`ASREPRoasting` 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 ASREPRoasting attacks.



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!");
    }
```

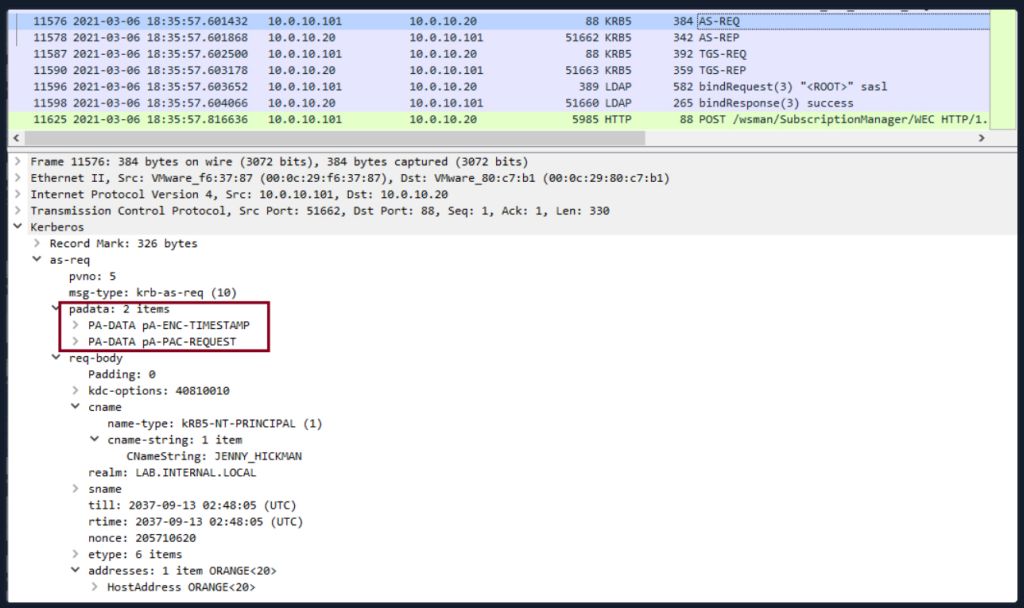
```
    foreach (SearchResult user in users)
    {
        string samAccountName = user.Properties["samAccountName"][0].ToString();
        string distinguishedName = user.Properties["distinguishedName"][0].ToString();
        Console.WriteLine("[*] SamAccountName         : {0}", samAccountName);
  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

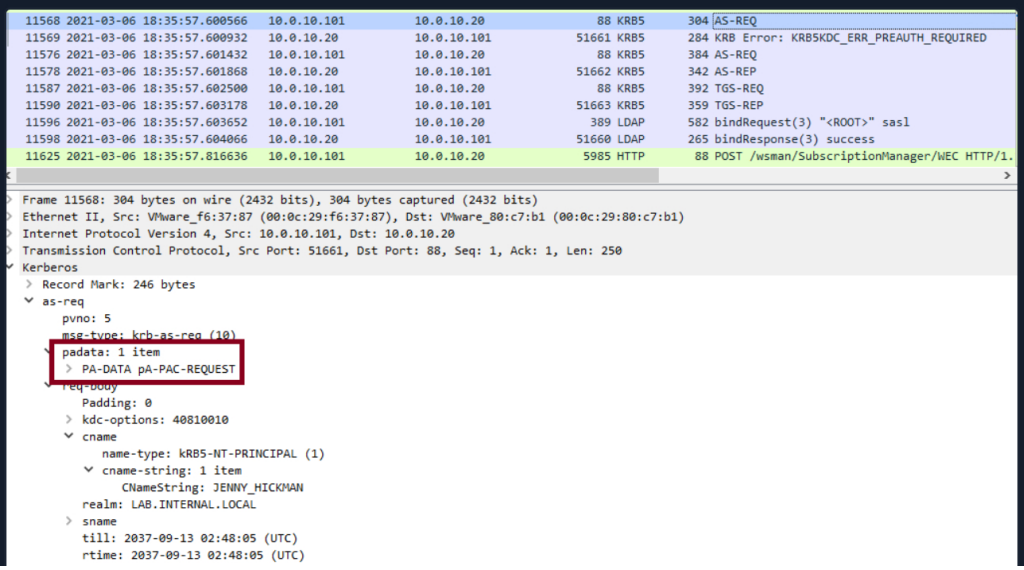
Kerberos Pre-Authentication

`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.



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.



```
    nonce: 205710609
  > etype: 6 items
  v 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

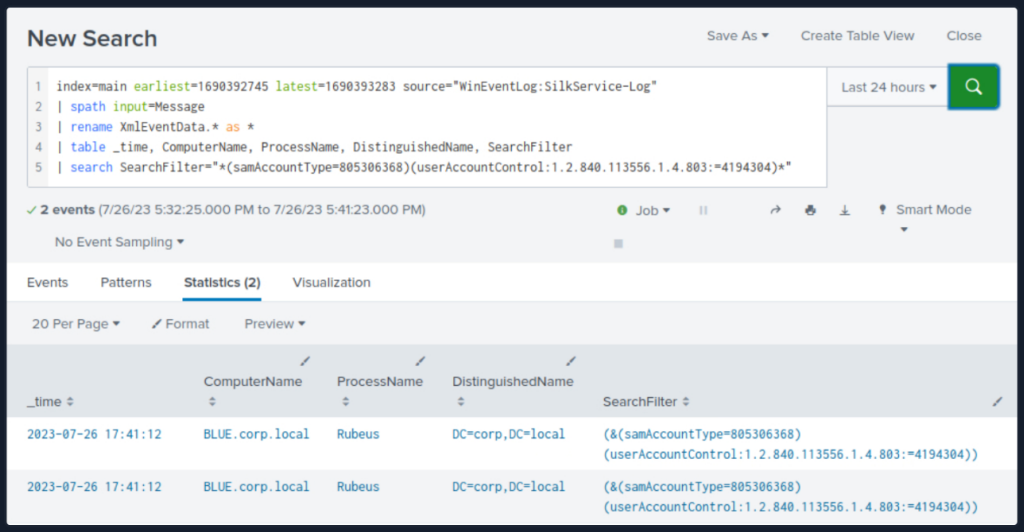
Now let's explore how we can identify AS-REPRoasting, using Splunk.

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:=41943
```

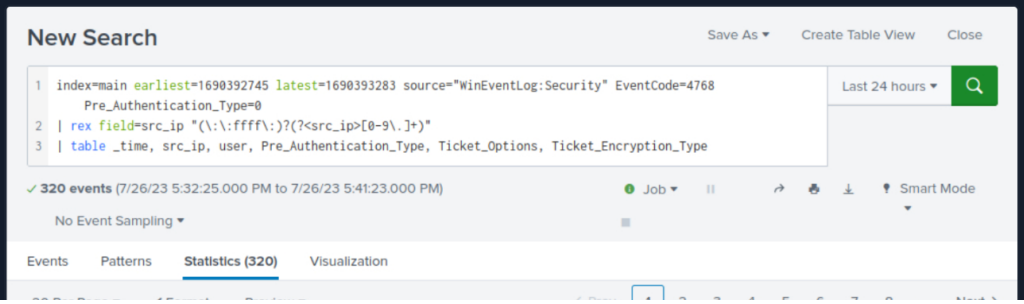


Detecting AS-REPRoasting - TGT Requests For Accounts With Pre-Auth Disabled

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
```

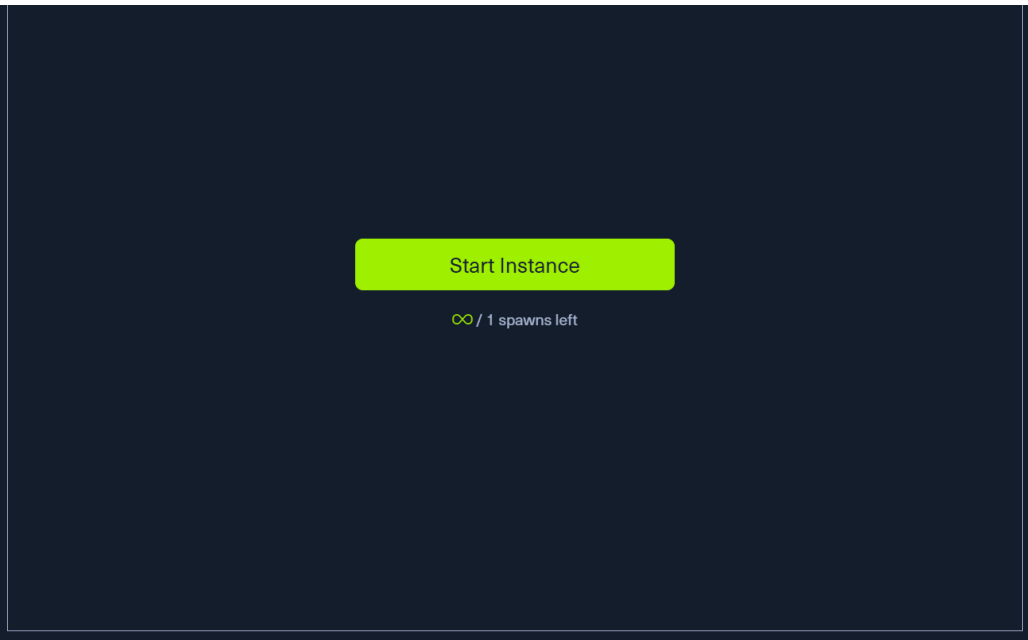


20 Per Page ▾ Format Preview ▾ < Prev 1 2 3 4 5 6 7 8 ... Next >

_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	ROSENDO_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	LORAINE_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.



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 HACKTHEBOX