

Hunting in Active Directory: Unconstrained Delegation & Forests Trusts



Roberto Rodriguez · [Follow](#)
Published in Posts By SpecterOps Team Members · 16 min read · Nov 28, 2018

--

2

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.

Page 1 of 26

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 *“An attacker who compromises a domain controller in a forest (or any server with unconstrained delegation in said forest) can coerce domain controllers in foreign forests to authenticate to the attacker-controlled server through “the printer bug.” Due to various delegation settings, the foreign domain controller’s ticket-granting-ticket (TGT) can be extracted on the attacker-controlled server, reapplied, and used to compromise the credential material in the foreign forest.”*

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:

Delegation	Delegation Style	Description
Unconstrained	TGT Forwarding	When a user accesses a server with unconstrained delegation enabled, the user sends their TGT to the server. The server can then impersonate the user by using the user's TGT to authenticate to other services in the network.
Traditional Constrained	Service for User to Self (S4U2self) Service for User to Proxy (S4U2proxy)	Any accounts (user or computer) that have service principal names (SPNs) set in their msDS-AllowedToDelegateTo property can pretend to be any user in the domain (i.e. they can "delegate") to those specific SPNs.
Resource-Based Constrained		Implemented with a security descriptor on the target resource, instead of a list of SPNs on the "frontend" account that the frontend account is allowed to delegate to. This security descriptor is stored as a series of binary bytes in the msDS-AllowedToActOnBehalfOfOtherIdentity property on a target computer object.

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:

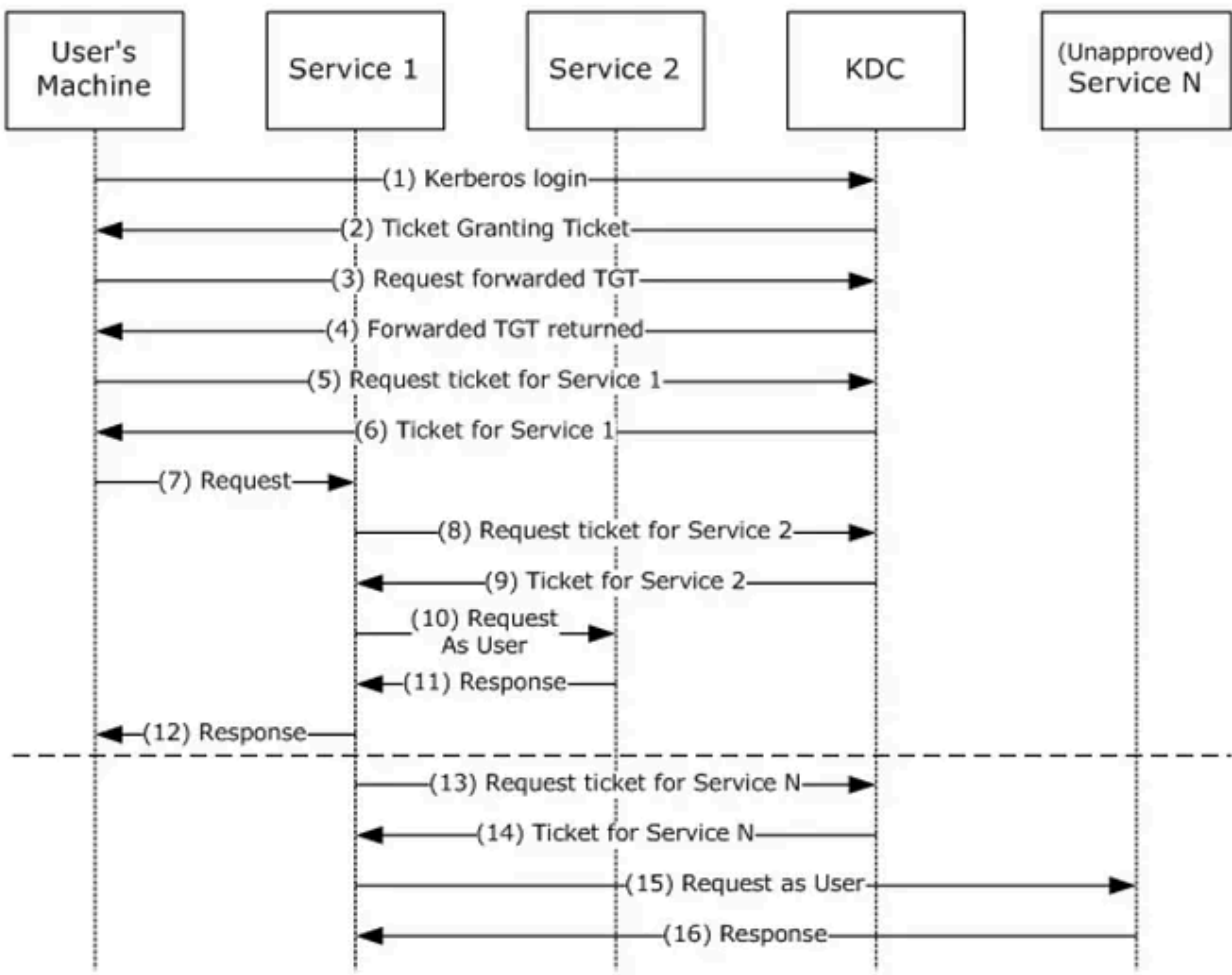


Figure 1: Kerberos Delegation with Forwarded TGT

- 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. **The user makes a request to Service 1 by sending a KRB_AP_REQ message, presenting the service ticket, the forwarded TGT, and the session key for the forwarded TGT.**
8. 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.
9. 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.
10. Service 1 makes a request to Service 2 by a KRB_AP_REQ, acting as the user.
11. Service 2 responds.
12. With that response, Service 1 can now respond to the user's request in step 7.
13. 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.
14. The KDC will return the requested ticket.
15. 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.
16. 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](#).

Other Trusts

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:

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:

- **ERROR_ACCESS_DENIED:** *“Target server attempted authentication and got an access denied. If coercing authentication to an NTLM challenge-response capture tool (e.g. responder/inveigh/MSF SMB capture), this is expected and indicates the coerced authentication worked”*
- **ERROR_INVALID_HANDLE:** *“Attempted printer notification and received an invalid handle. The coerced authentication probably worked!”*

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 (**hydrogen.covertius.local**) 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 (**rikers.cyberpartners.local**) as the victim since we want its TGT to then perform a DCSync attack from the compromised DC with unconstrained delegation configured.

Tools

- Rubeus and SpoolSample 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.

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:

```
Rubeus.exe monitor /interval:5 /filteruser:VICTIM-DC-NAME$
```

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

```
SpoolSample.exe VICTIM-DC-NAME UNCONSTRAINED-SERVER-DC-NAME
```

(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.

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

Data Sources Needed

Security Events Sequence

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

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.

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.

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

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**

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

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

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).

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

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.

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.

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

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

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

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

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](#).

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**)

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.

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**.

Special privileges are assigned to new logon

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)**

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:

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.

As part of the handle to LSA, Rubeus registers the logon application name “**User32LogonProcess**” 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](#).

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: **User32LogonProcesss** (yes with three “S” at the end). That should stick out right away in your environment in **Security event 4611**. Also, even when spelled right, “**User32LogonProcess**” 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 **SeTcbPrivilege** privilege set, it fails when calling the **LsaRegisterLogonProcess** privileged service. Check for **Audit Failure** and privilege services being called by non-system users in **Security Event 4673**.

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 **Security event 4675**.
- Filter the results by **SID S-1-5-9**. You will get communications about **DCs** from other forests (potential victims or regular behavior).
- You can also stack the values obtained by your first aggregation by the **Tdo (trusted domain object) Domain SIDs**. 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 **Security event 5145**. 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://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://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/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-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/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>

- Microsoft
- Threat Hunting
- Cybersecurity
- Data Analytics
- Data Analysis

 --  2



Written by Roberto Rodriguez

1.6K Followers · Writer for Posts By SpecterOps Team Members

Follow

