# Kerberos 101 Labs

This document contains all lab instructions for the Kerberos 101 workshop

Contents

[Kerberos 101 Labs 1](#_Toc523914940)

[1. Preparing the lab environment 3](#_Toc523914941)

[2. Trace a domain logon 3](#_Toc523914942)

[3. Trace accessing a file server 5](#_Toc523914943)

[4. Troubleshooting SPNs Part 1 7](#_Toc523914944)

[5. Troubleshooting SPNs Part 2 8](#_Toc523914945)

[6. Keytab Files 10](#_Toc523914946)

[7. Kerberos Unconstrained Delegation Lab 11](#_Toc523914947)

[8. Simple Web Scenarios 14](#_Toc523914948)

[8.1. Web Site Binding and simple access 14](#_Toc523914949)

[8.2. Registering the HTTP SPN 15](#_Toc523914950)

[8.3. Impersonation 16](#_Toc523914951)

[8.4. Classical delegation network traffic 17](#_Toc523914952)

[9. Kerberos Constrained Delegation Lab 19](#_Toc523914953)

[9.1. Enabling constraint delegation (S4U2Proxy) 19](#_Toc523914954)

[9.2. Protocol Transition (S4U2Self) 20](#_Toc523914955)

## Preparing the lab environment

First, we need to connect to the lab environment

* Download the content of the workshop share (download as zip makes it easier)
* Get the password from the RTF document
* Open the RDP connection and use the previously extracted password

Now, we need to get the PowerShell function library to the KerbClient2 VM. This library provides the functions to break your Kerberos configuration.

* Copy the Kerberos 101.ps1 to the RDP Session (or download it in the RDP session from the dropbox)
* Start the VMs
  + Open the Hyper-V Manager
  + Start the KerbDC0, 1 and 2
  + When the domain controllers are online, start the rest of the machines (Kerb\*)
* Now we need to copy the file “Kerberos 101.ps1” (not the Kerberos 101 Lab.ps1) to [\\KerbClient2\c$](file:///\\KerbClient2\c$). If you get prompted for user credentials use the a\Install with the password “Somepass2”.
* Logon to the KerbClient2 (simply double click on the machine in the Hyper-V Manager) with the a\Install account and “Somepass2”.
* Start a PowerShell and call the previously copied script “Kerberos 101.ps1”
  + You will pretty likely get an error telling that the execution of scripts is disabled. Please run the following command to enable script execution: Set-ExecutionPolicy Unrestricted

## Trace a domain logon

In this lab, we take a first look at Kerberos network traffic. All relevant information is available in clear test. This does not compromise security but help understanding how Kerberos works and troubleshooting issues.

Logon to machine KerbDC2 using the Hyper-V tools. The password for the user “a\Install” is “Somepass2”.

Start the Wireshark and start also a new network trace.

This will cause some network traffic that the network sniffer running on the domain controller will capture.

As there is a lot of communication that is not interesting we can filter the traffic for Kerberos traffic.

Type “Kerberos” into the filter textbox and press enter. If the box is not green something is wrong with the filter.

Now everything is setup and we can logon to the client machine.

Logon to the client machine using the Install user account of domain ‘a’

We see Kerberos packets showing up in Wireshark almost instantly. There are pretty likely also some SMB and LDAP packets showing up.

There are many tools available to make the Kerberos tickets visible. The only one also shipped with the operating system is klist.exe. Another one available on the client is kerbtray.exe which is kind of a graphical version of klist.exe

Start klist.exe on the client machine and take a look at the tickets in the ticket cache.

How many tickets is the client requesting in total?

Why do we see two Authentication Requests (AS-REQ)? Take a look at the “padata” field

On which accounts are the SPNs registered the client requests tickets for?

Why does Wireshark include LDAP and SMB packets even if the filter should only allow Kerberos?

1. Kerberos communication can be easily read in a network trace
2. Kerberos Tickets can be made visible using the tools klist.exe

## Trace accessing a file server

This lab is pretty similar to the previous one. Everything is done from the client.

We are accessing a File Server two times. There should be a big difference visible in the trace.

Logoff the current session from the machine KerbClient2 and logon again using the Hyper-V tools. The password for the user “a\Install” is “Somepass2”.

Start the Wireshark and start the trace.

You might want to set the filter in Wireshark to Kerberos again.

Now we are requesting data from a file server. There is a prepared PowerShell function for that and just needs to be called:

* Get-SmbData1

The function returns the number of files read from the file server.

If PowerShell complains that the execution of scripts is disabled, run the following command first

Set-ExecutionPolicy Unrestricted

Open a PowerShell and call the function “Get-SmbData1”.

After the script is finished, stop the network trace.

How many Kerberos packets do you see in the trace?

What was the authentication protocol?

Now we do the same thing again but using the second PowerShell function to read the data from the file server.

Re-Logon to the client machine.

Start a new trace in the Wireshark.

Call the PowerShell function “Get-SmbData2”

Stop the network trace

How many Kerberos packets do you see in the trace this time?

What is the SPN the client asks a ticket for?

What is the difference between “Get-SmbData1” and “Get-SmbData2”?

1. In order to get a ticket from the KDC you need for a specific SPN. If you use the IP address instead of the machine name the SPN cannot be build, hence the client tries to use a different way of authentication.  
     
   However this depends on the client as the client is in charge of constructing the SPN. The function “Get-SqlData1” connects to a SQL server by using the IP address AND Kerberos.

## Troubleshooting SPNs Part 1

This lab convers another well-known issue: Missing SPNs.

Restart the client machine and the file server. Then logon to the client machine

Open a PowerShell and invoke the function Start-Lab4. Please note the computer name returned by the function

Open Wireshark and start a network trace again

Browse to the c$ share of the machine returned by Start-Lab4

Stop the network trace

Accessing the share works so we do not expect anything wrong with the Kerberos configuration.

Analyze the network trace. A closer look on the communication shows that there is something wrong.

What is the Kerberos error in the trace?

Is there anything interesting about the requested SPN?

What was the authentication protocol?

1. If an SPN cannot be found, authentication may still work as the operating system falls back to NTLM

## Troubleshooting SPNs Part 2

This lab convers another well-known issue: SPN registered on the wrong account.

SPNs are unique identifiers for services running on servers. Every service that will use Kerberos authentication needs to have an SPN set for it so that clients can identify the service on the network. If an SPN is not set for a service, then clients will have no way of locating that service. Without properly set SPNs, Kerberos authentication is not possible.

An SPN is registered in Active Directory under a user account as an attribute called Service-Principal-Name. The SPN is assigned to the account under which the service the SPN identifies is running. Any service can look up the SPN for another service. When a service wants to authenticate to another service, it uses that service's SPN to differentiate it from all of the other services running on that computer.

Do a fresh logon on the client machine and also restart the file server.

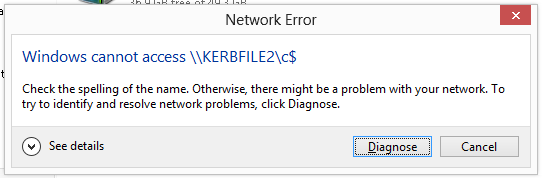
Open a PowerShell and invoke the functions **Repair-Lab4** and **Start-Lab5**. Please note the computer name returned by the function Start-Lab4.

Open Wireshark and start a network trace again

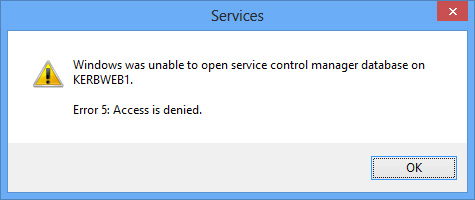
Browse to the c$ share of the machine returned by Start-Lab5

Stop the network trace

You are not be able to connect to the machine’s c$ share and get the following error:



Now we try connecting to the event log of the machine. You might want to start a new network trace.



Interestingly the error messages are totally different.

It is time to analyze the network trace. We see a lot of TGS-REQs and TGS-REPs but also many messages with the error code KRB5KRB\_AP\_ERR\_MODIFIED.

In general cases, when information cannot be decrypted, the system provides the error KRB\_AP\_ERR\_MODIFIED, meaning that the encrypted information is not accessible, but with no further information. For example, the data could have been modified in transit, the data could have been malformed initially, or there could be no available key. Each of these possible causes requires troubleshooting at different locations and using different techniques.

What might be the cause for the problem?

Correct the problem either manually or by invoking the function Repair-Lab5.

1. Application usually do not tell about the underlying Kerberos problem. Two different errors messages have the same root that is actually a decryption problem.

## Keytab Files

In this lab you will remove create a new keytab file to enable a Linux machine to authenticate against a Windows Active Directory domain.

In the lab all Linux machines are already configured for Kerberos. And there is much more involved than just creating the keytab file. This lab just puts the focus on the keytab and expects all the other configurations to be in place.

In the next step you will remove the existing keytab file from the KerbLinux1. It is important to create the new keytab file first and copy it to the Linux machine as after removing the existing keytab file, copying files between the Windows and Linux machines get complicated:

First make sure that the Linux machine can access the Test share on KerbFile2 by changing into the /test folder and do a directory listing. If there is no file visible

Logon to KerbLinux1 with the user [install@vm.net](mailto:install@vm.net).

Get the directory listing of /test and make sure there is a file in there

Cd /test

ls

if you do not see a file, reconnect the share by running the following command on the host computer

Invoke-LabCommand -ActivityName 'Mounting test share' -ComputerName KerbLinux1 -ScriptBlock {

sudo mkdir /test

'Somepass1' | sudo kinit install@VM.NET

sudo mount -t cifs -o sec=krb5 //KerbFile2.a.vm.net/Test /test --verbose

} -PassThru

The next step is to create a new Kerberos Keytab file. For this please logon to the domain controller KerbDC1. It is not required to create keytab files on the domain controller. However, in the lab the machine has the tools already installed to do that.

Create the keytab file with the following command

ktpass /out krb5.keytab /princ KERBLINUX1$@VM.NET /mapuser 'KerbLinux1$' /crypto AES256-SHA1 /ptype KRB5\_NT\_PRINCIPAL /pass Password2 /target KerbDC1.vm.net

After you have created it, copy it to the test share on KerbFile2.a.vm.net.

Logon to KerbLinux1 with the user [install@vm.net](mailto:install@vm.net).

Remove the file /etc/krb5.keytab

sudo rm /etc/krb5.keytab

This breaks the trust relationship between the Linux machine and the domain as there is no pre-shared secret anymore. As tickets stay in the cache, everting should work as normal unless you do not purge the cache.

Restart the Linux machine KerbLinux1

Try to logon with the account [install@vm.net](mailto:install@vm.net) again. This should fail now.

## Kerberos Unconstrained Delegation Lab

This lab demonstrates a typical issue when working remotely on a machine. From the remote machine you want to access a network share or Active Directory and it does not work at all. Kerberos Delegation seems to be the easy solution to this, but it comes with security issues.

First, we reproduce the common remoting problem and then we configure Kerberos Delegation to work around it.

Logon to the client machine and open a PowerShell

Connect to the web server that is in the same forest / domain as the client using PowerShell remoting. This can be done with the following command. The connect does not work you have not invoked Repair-Lab4

Enter-PSSession –ComputerName <web server name>

The prompt should have changed and reflect that you are working on the remote machine.

Get the directory listing of C:\

Get the directory listing of a remote location, for example the c$ share of the domain controllers that is in the clients domain or the file server

You not be able to access the remote share getting the error “Access denied”.

Why is the server we have connected to not be able to authenticate us against a third machine?

What does the machine require to authenticate us to a third party?

We want to enable Kerberos delegation now. This is done on the computer account of the web server.

Open dsa.msc on the client

Get the properties of the web server’s account and navigate to the delegation tab

Enable the setting “Trust this computer for delegation to any service (Kerberos only)” and click OK

**Restart the web server and re-logon to the client**

**Connect to the web server again using PowerShell Enter-PSSession**

**Take a look at your tickets using klist**

This is it. Delegation is enabled. After the machine has rebooted, do the first steps of this lab again to connect to the web server and access a remote share from there. It works now and you can access all resources from the remote machine – there is no restriction.

1. Classical Kerberos Delegation is pretty powerful, maybe too powerful.
2. Unconstrained delegation works by passing a forwarded TGT to the remote machine. In fact, the remote machine can work with the TGT like you can on your local machine.

Is there another way of allowing a second hop from a remote machine?

What are the downsides when enabling Kerberos delegation on a machine?

## Simple Web Scenarios

In the following labs we are mainly working on the web server and the client.

The web server has three web applications installed that are basically configured. All web applications have windows integrated authentication enabled und are running in its own application pool.

During the next labs we want to learn more about how Kerberos is used in web scenarios and which settings are available for configuration.

### Web Site Binding and simple access

Per default all applications running in the “Default Web Site” are also running in the “DefaultAppPool”. The default app pool runs under the "ApplicationPoolIdentity" for your. Windows creates a virtual account with the name of the application pool and runs the Application Pool's worker processes under this account.

If you create a new web site in the IIS and it is recommended to make the new site accessible with a different DNS name (host header name).

Check the binding of the site TestSites by clicking on “advanced settings…”. It should be “http:\*:80:testsites.a.vm.net”. In order to reach the site you need to access it using the name “testsites.a.vm.net”.

Now we want to browse to the site IdentityCheck.

Click on the application “IdentityCheck” (not the web site) and then on “Browse Application”.

This does not work as there is yet no DNS record. So we need to create on the name server for the domain “a”.

Open the DNS management tools on the web server or the domain controller and navigate to the forward zone “a.vm.net”. Create a new A record with the name “testsites.a.vm.net” and the same IP address as the web server.

Browse the application “IdentityCheck” again.

**Note:** that the error "Access denied" or "No network provider accepted the given network path" could appear then accessing the web page using FQDN or its CNAME alias. See [KB926642](https://support.microsoft.com/kb/926642?wa=wsignin1.0) for further information.

Another interesting KB article is [KB281308](http://support.microsoft.com/kb/281308) about DisableStrictNameChecking, talking about accessing certain services using the CNAME.

This time the application could be found but you get a credential prompt. Regardless what you provide you will not get access to the web site.

Try to troubleshoot the problem using a network sniffer.

Do you get any Kerberos errors?

Are there any Kerberos frames at all?

By default the Internet Explorer does not use Kerberos for authentication unless the target site is in the local intranet zone. This is a common issue that makes you think that Kerberos is not working but it is not even considered as an option.

Add the address <http://testsites.a.vm.net> to the local intranet zone.

Start Wireshark and then access the application “IdentityCheck” (<http://testsites.a.vm.net/IdentityCheck/>) again.

Did the authentication work?

What was the used protocol?

### Registering the HTTP SPN

Thanks to the trace we know the SPN the Internet Explorer is requesting a ticket for.

What is the account to register the SPN on?

How can you find out the account the application is running under?

The web site “TestSite” is running under the app pool named “TestSite”. In the IIS Manager go to Application Pools -> TestSites and click on “Advanced settings…”. Here you can see the app pool’s identity.

Now we change the app pool identity to a domain account. The account does already exist in Active Directory and is also named TestSites.

Check the domain for a user names TestSites. If the user does not exist, please create one.

Reset the password on the account TestSites (make sure that “User must change password at next logon” is not checked).

Change the identity of the app pool TestSites to the user TestSites

That is the account you want to register the SPN found out in the previous lab: http/testsites.a.vm.net and http/testsites

Do a fresh logon to the client machine and connect to the web page again. Start Wireshark before to see if Kerberos really works now.

### Impersonation

The application TestSite1 is designed to retrieve information from the pubs database of the specified SQL Server. The web site does not impersonate the user’s identity so it will connect to remote resources with the app pool’s identity.

On the client machine Connect to the web site TestSite1 using the internet explorer.

Enter the name of the SQL Server in the text box and press the button “Read Records”.

The web site should report an error telling that the access is denied to the user A\TestSites. This is expected as the account is not allowed to access the SQL database. As impersonation is not enabled on that application all remote connections are done with the app pool’s identity. That is why even if you have connected to the site as the Install user the connection to the SQL server is done using the TestSites account.

Enable impersonation for the web site Test1. You find the option in the IIS Manager by clicking on the web application and then double-click on the “Authentication” Button. You get a list of authentication options. Enable impersonation here.

After that connect to the web application again and query data from the SQL Server again.

What is the error this time?

Why is the account a different this time?

### Classical delegation network traffic

Let’s active the app pool’s account first for classical delegation.

Open dsa.msc on the client

Get the properties of the account “TestSites” and navigate to the delegation tab

Enable the setting “Trust this computer for delegation to any service (Kerberos only)” and click OK

**Restart the web server and the client.**

After that you should be able to access the SQL Server from the TestSite1.

What happens behind the scenes?

You may want to restart the client and trace the network traffic that happens if you logon to the TestSite1 web app.

Logon to the client machine

Start the Wireshark

Access the web application TestSite1

You are seeing the expected TGS\_REQ for the http SPN but an additional TGS\_REQ requesting a new “krbtgt” for the current domain. Look at the KDCOptions flag. This time the “Forwarded” bit is set.

How does the client know it has to request an addition TGT?

Unfortunately this is not visible from the network trace. However if you take a look at the tickets you have in the cache on the client using klist.exe you see the http ticket for the web server having set option “ok\_as\_delegate”. This tells the Kerberos client that we are going to connect to a machine that is trusted for delegation. Hence we are requesting the forwarded TGT before actually accessing the web application.

Look at the size of the HTTP GET request. It should be about 4500 bytes even if the Kerberos reply for the http ticket is just about 1600 bytes. This indicates that we are passing two tickets to the web server: the http service ticket and the forwarded TGT.

Kerberos Delegation triggers the client to request a forwarded TGT and pass this along with the service ticket to the target machine (in the AP Request or CS Exchange).

This scenario can cause TokenSize issues as the token and also the http request can get about twice as big.

1. Delegation increases the request size as two tickets have to be sent to the service account.
2. Unconstrained delegation gives the service account full control over your credentials which makes it quite risky.

## Kerberos Constrained Delegation Lab

This lab is about configuring Kerberos constrained delegation and protocol transition for a web application.

All installed web applications are configured to run under the domain account TestSites. This account is not trusted for delegation therefrom we cannot access any resources from the page IdentityCheck or TestSite1.

### Enabling constraint delegation (S4U2Proxy)

We now want to disable the unconstrained delegation and setup constrained delegation. The web server should be able to delegate credentials only to the SQL server and only to the SQL service on the SQL Server.

~~Enable impersonation on the web application “IdentityCheck” as done in 6.3 Impersonation.~~

Enable constraint delegation on the Active Directory Account “TestSites”. Go to the delegation tab of the account and enable the option “Trust this user for delegation to specified services only”. This automatically enables “Use Kerberos only” as well.

Click the “Add…” button now to add the SPNs the service will be allowed to delegate to. Search for the SQL server and pick just the MSSQLSvc SPNs from the list.

Do an “iisreset” on the web server then

Logon to the client and connect to the web site TestSite1. Try to read records form the SQL Server

If that works it might be again interesting how the Kerberos network communication looks like. Constrained delegation does not work like the unconstrained one. Instead of forwarding the TGT to the server the client just passes the normal service ticket. This is sufficient to the web server to request new ticket on the user’s behalf.

Restart the web server

Start Wireshark on the web server

Logon to the client and access the web application TestSite1. Request data from the SQL Server.

What SPN does the “evidence ticket” have?

What additional flag is set in the TGS\_REQ for the SQL Server?

1. Constrained delegation does not forward the passport (TGT). It uses just the service ticket as an evidence to the KDC to request the tickets whose SPN matches the Allowed-to-Delegate-to (A2D2) list.

### Protocol Transition (S4U2Self)

If an account is enabled for Protocol Transition it can request Kerberos tickets for any user but only for the SPNs that are in the Allowed-to-Delegate-to (A2D2) list. The service account just needs the user’s UPN – nothing else.

Go to the delegation tab of the Active Directory account TestSites and set the radio button “Use any authentication protocol”. The list of SPNs stays the same.

There are many test accounts in the Active Directory OU “Lab Accounts”. Pick one account and reset the password (make sure that “User must change password at next logon” is not checked). Then enable this account.

On the web server open the local policies using gpedit.msc. Navigate to “Computer Configuration -> Windows Settings -> Security Settings -> Local Settings -> User Rights Assignment -> Act as part of the operation system” and add the account “TestSites” in here.

Restart the web server

Logon to the client machine and start the Wireshark.

The access TestSite2 from the client and enter the user’s UPN into the text box. Press the Go button.

Stop the network trace and take a look at the Kerberos traffic

The result is just an error. For security reasons the web server does not show the error to the user unless the web page is opened on the web server itself. You might want to do that to get a better idea about what is going on.

1. This time the web server does not have an evidence ticket to show to the KDC. That’s why the frontend requests TGS to its own UPN as Sname, with field PA-FOR-USER (cname/crealm)

Optional: The problem simply is that the user does not have yet a SQL login. You can easily create it by using the following SQL script. Please change the user’s SAM account name to the account you have chosen.

USE [master]

GO

CREATE LOGIN [A\a877777] FROM WINDOWS WITH DEFAULT\_DATABASE=[master], DEFAULT\_LANGUAGE=[us\_english]

GO

USE [pubs]

GO

CREATE USER [A\TestApp1] FOR LOGIN [A\a877777] WITH DEFAULT\_SCHEMA=[dbo]

GO

ALTER ROLE [db\_owner] ADD MEMBER [A\a877777]

GO