Kerberos and Windows Security: Kerberos on Windows

medium.com/@robert.broeckelmann/kerberos-and-windows-security-kerberos-on-windows-3bc021bc9630

Robert Broeckelmann 25 октября 2018 г.

Robert Broeckelmann

/ Hercules and Cerberus LACMA 65.37.151

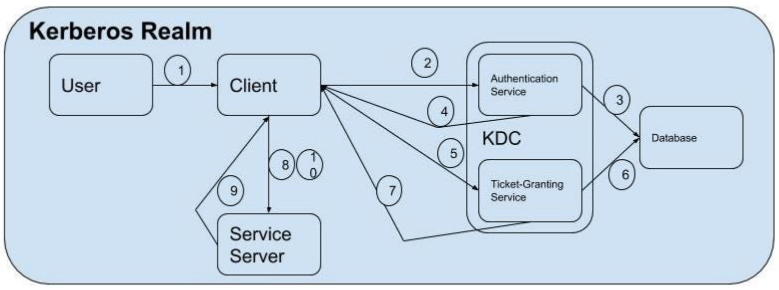
In this next post in my Kerberos and Windows Security Series, we are going to look at the use of Kerberos in Microsoft Windows (Microsoft Kerberos). At the end of the day, Kerberos with Windows is more-or-less the same as Kerberos anywhere else; though, there are several proprietary extensions that Microsoft has implemented. Those proprietary extensions are limited to fields that the Kerberos spec set aside for such extensions; so, this doesn't necessarily break interoperability. In general, I'm not much of a fan of proprietary protocols, especially for security; some solace can be taken in Microsoft's Open Specification Promise. To some extent, the ubiquitous nature of

Windows deployments makes its implementation of Kerberos (plus extensions) a standard in its own right — the good people in the Kerberos (now Kitten) Working Group at the IETF would likely disagree with that assertion. I suppose I would as well if I were them, but what I'm describing below (including the Microsoft's pieces) is very common.

If you're coming to this article directly from a search engine, then it may be helpful to look at my post on the generic Kerberos v5 Authentication Protocol first. If you want to dive right into some examples with network traces, check out my posts on Windows Logins and SPNEGO. Or, you could start at the beginning and read through the whole series. Regardless, of where you start, this particular post is about the use of Kerberos with Microsoft Windows.

Kerberos v5 Summary

The following diagram provides a high-level summary of the messages exchanged between actors during the Kerberos message exchanges.



1. User enters credentials (username + password).
2. Send KRB_AS_REQ.
3. Lookup user (and password) in database.
4. Send KRB_AS_RSP.
5. Send KRB_TGS_REQ.
6. Lookup service (and password) in database.
7. Send KRB_TGS_RSP.
8. Send KRB_AP_REQ.
9. Send KRB_AP_RSP.
10. Send service request to Service Server.

Kerberos Protocol High-Level

For more information, see here. Kerberos basics are covered in that post and won't be revisited here.

Kerberos is complicated. Windows hides a lot of that complexity. This makes it easy to spin up Windows Domains, but can become a nightmare for the Windows Server administrator when things are not working — that job title doesn't usually involve being an identity expert.

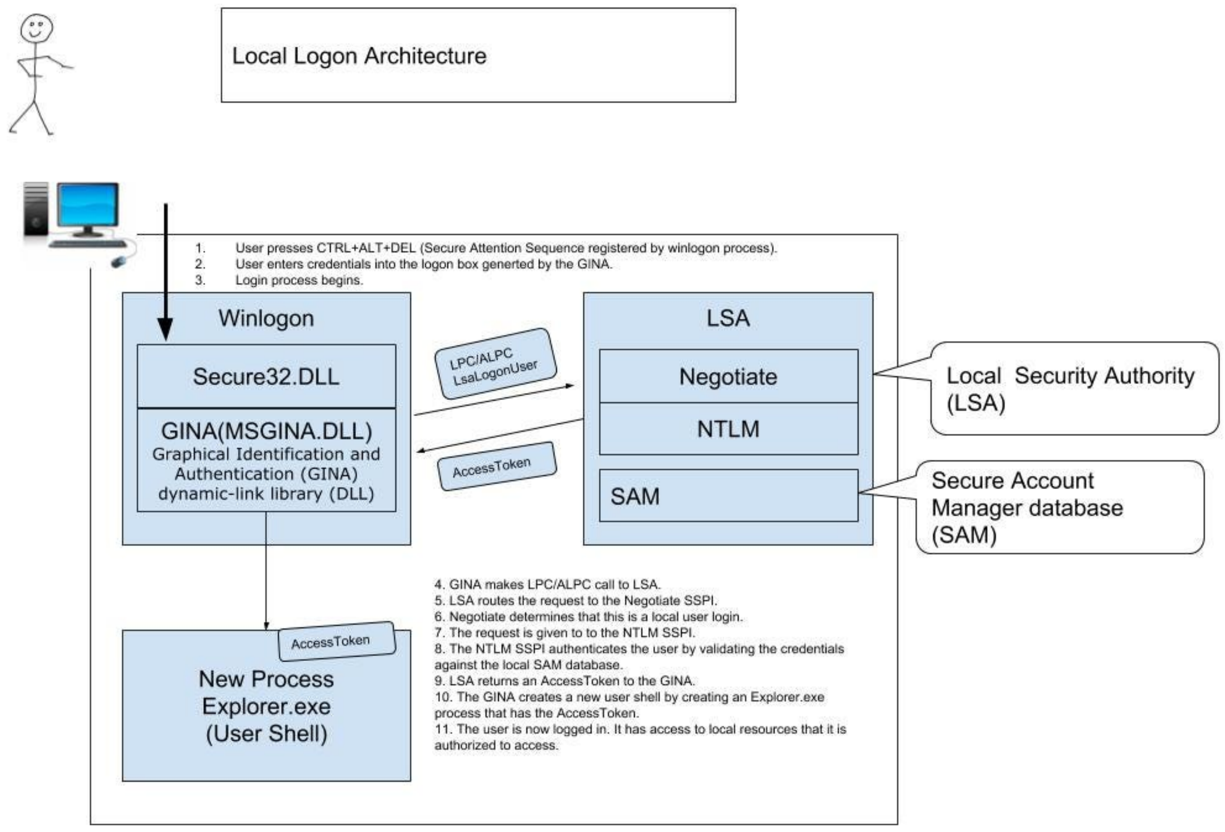
The quintessential writeup on Kerberos on Windows is available here. When I was first researching this series and asked a peer of mine in Microsoft Consulting about it, he pointed me towards that article. Before we dive too deep into this, it's important to remember that Windows Kerberos is Kerberos at the end of the day (with a couple of MS-specific pieces).

Windows Security

If you are logging into a Windows instance (server or workstation, hardware or virtual machine) that is not connected to a domain, then you are performing a local logon (using NTLM and authenticating against a local user database). If it is connected to a domain, then a domain logon is being performed (it will use Kerberos, by default). For domain logons, the user database is on the domain controller. Even if the Windows instance is attached to a domain, there is still an option to log into the local OS — we'll assume typical use cases as described in this paragraph.

When a user logs into Windows, several sub-systems are in place that interact with one another to securely authenticate the user. The details are slightly different depending upon whether a local user or domain user is logging into the system. For this discussion, let's assume username and password are being provided. There are many other ways to authenticate a user to Windows and there are Kerberos specs that support many of those methods, but one thing at a time.

We'll look at how the local user login process works first. The diagram below describes the Local Logon process and security architecture of Windows. Here we assume that the Negotiate SPI chooses NTLM over other options.



Local Logon Architecture
Winlogon
Secure32.DLL
GINA(MSGINA.DLL)
Graphical Identification and Authentication (GINA) dynamic-link library (DLL)
LPC/ALPC LsaLogonUser
AccessToken
LSA
Negotiate
NTLM
SAM
Local Security Authority (LSA)
Secure Account Manager database (SAM)
AccessToken
New Process Explorer.exe (User Shell)

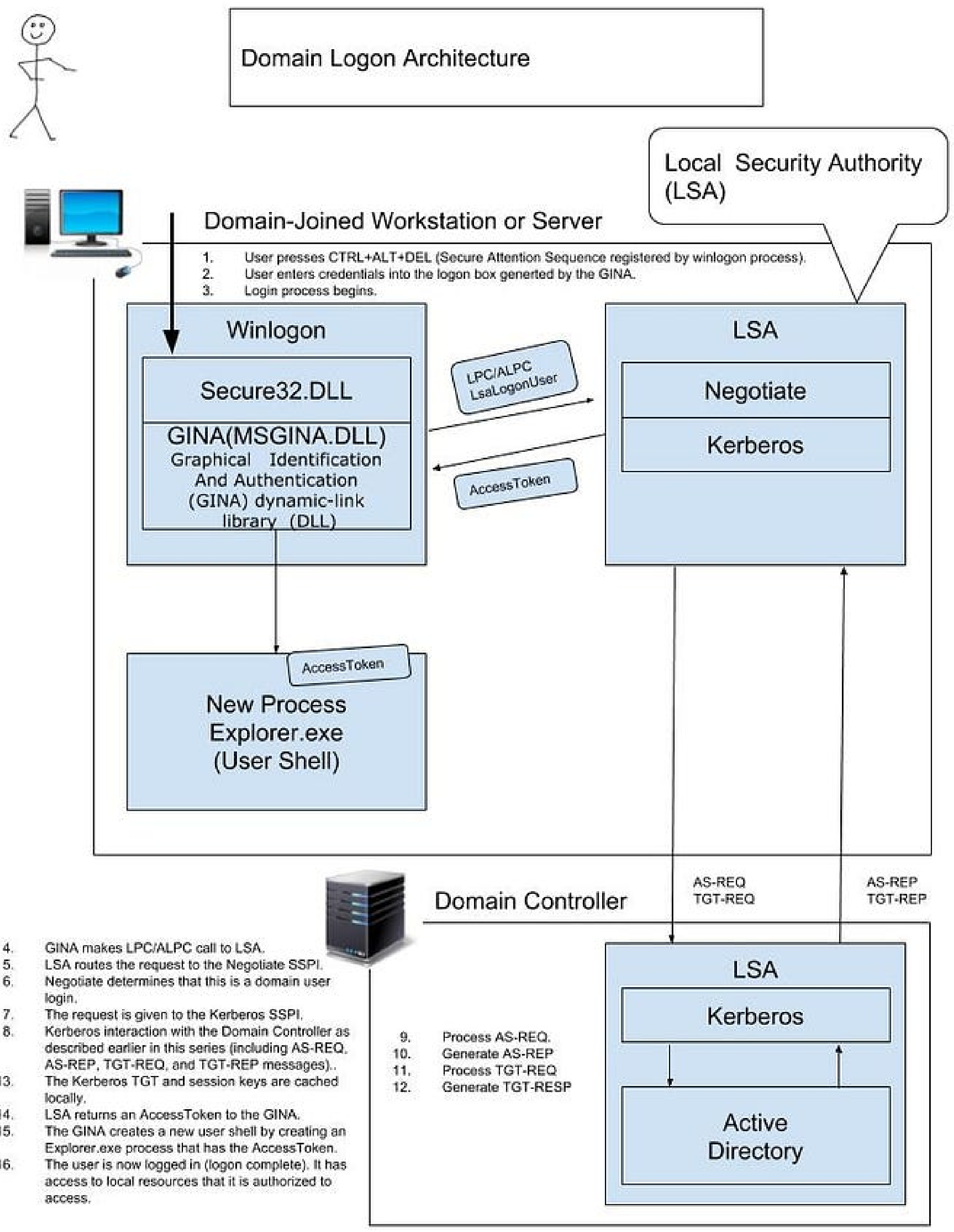
1. User presses CTRL+ALT+DEL (Secure Attention Sequence registered by winlogon process).
2. User enters credentials into the logon box generted by the GINA.
3. Login process begins.

4. GINA makes LPC/ALPC call to LSA.
5. LSA routes the request to the Negotiate SSPI.
6. Negotiate determines that this is a local user login.
7. The request is given to to the NTLM SSPI.
8. The NTLM SSPI authenticates the user by validating the credentials against the local SAM database.
9. LSA returns an AccessToken to the GINA.
10. The GINA creates a new user shell by creating an Explorer.exe process that has the AccessToken.
11. The user is now logged in. It has access to local resources that it is authorized to access.

Local User Logon Process

For more information about how NTLM works, see here and here.

The following diagram describes the domain user logon process and security architecture of Windows (when connected to a domain). Here we assume the Negotiate SPI chooses Kerberos rather than another option.


Domain Logon Architecture
Local Security Authority (LSA)
Domain-Joined Workstation or Server
Winlogon
Secure32.DLL
GINA(MSGINA.DLL)
Graphical Identification And Authentication (GINA) dynamic-link library (DLL)
LPC/ALPC LsaLogonUser
AccessToken
LSA
Negotiate
Kerberos
AccessToken
New Process Explorer.exe (User Shell)
AS-REQ TGT-REQ
AS-REP TGT-REP
Domain Controller
LSA
Kerberos
Active Directory

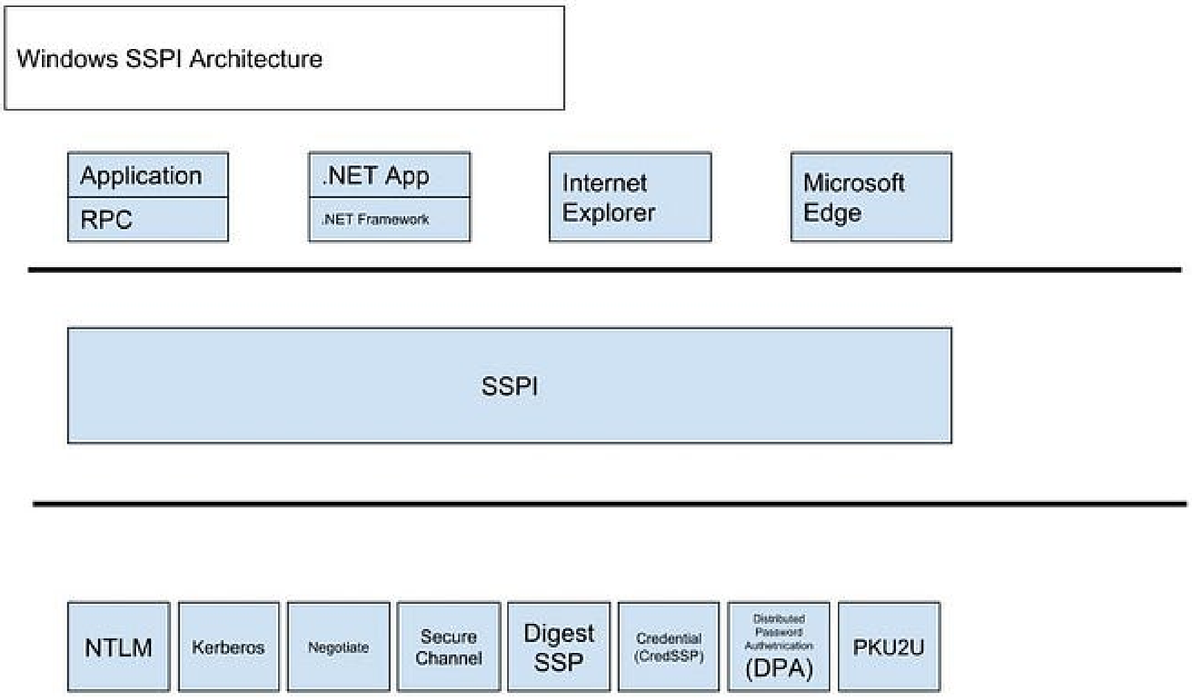
1. User presses CTRL+ALT+DEL (Secure Attention Sequence registered by winlogon process).
2. User enters credentials into the logon box generted by the GINA.
3. Login process begins.

4. GINA makes LPC/ALPC call to LSA.
5. LSA routes the request to the Negotiate SSPI.
6. Negotiate determines that this is a domain user login.
7. The request is given to the Kerberos SSPI.
8. Kerberos interaction with the Domain Controller as described earlier in this series (including AS-REQ, AS-REP, TGT-REQ, and TGT-REP messages)..
13. The Kerberos TGT and session keys are cached locally.
14. LSA returns an AccessToken to the GINA.
15. The GINA creates a new user shell by creating an Explorer.exe process that has the AccessToken.
16. The user is now logged in (logon complete). It has access to local resources that it is authorized to access.

9. Process AS-REQ.
10. Generate AS-REP
11. Process TGT-REQ
12. Generate TGT-RESP

Domain User Logon Process

The Windows SSPI (the Microsoft implementation of GSS-API) architecture looks like the following.



Windows SSPI Architecture

Microsoft decided not to use RFC1964 (GSS-API) in the original Kerberos implementation; their implementation, the Security Support Provider Interface (SSPI), uses a similar set of functions to GSS-API, but with extensions and Windows-specific data types. All Windows protocols use the Windows Authentication API (SSPI). The following SSPs (Security Support Providers) are available in Microsoft Windows:

Obviously, we care about the second one in this story.

At the completion of the Kerberos exchange (as outlined in this post), Privilege Attribute Certificate (PAC) data in theticket's Authorization Data field included in the TGT_REP message is used to create the access token (not to be confused with an OAuth2 access token) depicted in the diagram above. The Authorization Data field also includes:

- Information about the ticket and the client.
- Validate that the ticket belongs to the client who is submitting it.
- Returned by the KDC service to enable PKINIT.
- The Microsoft authorization data contains two digital signatures, one for the domain controller and one for the server offering the service. These signatures are used to prevent clients or untrusted services from generating their own forged Microsoft authorization data.
- Normally, the Microsoft authorization data is included in every pre-authenticated ticket received from an AS request. However, a client can also explicitly request — with the privilege attribute certificate (PAC)-request pre-authentication data field — to include or not include SIDs. If the field is present, the account SIDs will be included. If the field is not present, SIDs will not be included.

After (Kerberos) credentials reach the Windows instance (where the login was initiated), the token creation process is largely the same as for other authentication methods. The access token in the diagram above is an object (a Microsoft Windows proprietary construct that is independent of Kerberos) that describes the security context of a thread or process. Each process that is spawned by the user shell (explorer.exe, most likely) or any of its children inherits the access token that is originally issued to the user shell. The information in a token includes identity information and that identity's security attributes, including (from this)

- The (SID) for the user's account
- SIDs for the groups of which the user is a member
- A that identifies the current
- A list of the held by either the user or the user's groups
- An owner SID
- The SID for the primary group
- The default that the system uses when the user creates a securable object without specifying a
- The source of the access token
- Whether the token is a or token
- An optional list of
- Current impersonation levels
- Other statistics

The system uses the access token associated with a thread or process to identify the user when it interacts with securable objects or performs a task that requires priviledges.

The LSA caches the user's hashed password; however, computer and service password hashes are stored in a secure area of the computer's registry. This secure area of the registry is also used to store local account password hashes.

Windows Domains

The concept of a domain has evolved since Windows NT in the late 1990s. I'm not going to try to cover all those details here. These guys have done that already.

A Windows Domain provides administrators with a way to manage many workstations, laptops, and other devices from one place. One (or more) servers (domain controllers) in the domain control the domain and the computers in it.

A domain is the windows equivalent of a Kerberos realm.

Domain Controllers

If you are looking for information regarding how to setup a domain controller, check out the Windows Login example post in the second paragraph for details of how I setup a Windows Server 2016 domain. Note, if you are trying to setup a Windows Domain from scratch and are looking for documentation about how to do it, finding someone who has done it before will likely save you a lot of money in the long run.

A domain controller is an instance of Windows Server that is running Active Directory Domain Services. It processes authentication requests (logins, permission checks, token updates, etc.) in a Windows Domain. A domain will have one or more Domain Controllers. From a practical standpoint, there is typically more than one.

The Kerberos Key Distribution Center (KDC) is a domain service running on one or more Domain Controllers. Microsoft implements the KDC as a single process that provides two services: the Authentication Service and Ticket Granting Service. The KDC runs on every domain controller.

Active Directory

At the heart of Windows Kerberos is Active Directory. Everybody has heard of Active Directory (AD). But, what is it? Active Directory is the software components running on a Windows Domain Controller that implements:

- Kerberos account database that contains people users, computer users, and passwords.
- an LDAP server
- Some other stuff that isn’t important right now.

If you are like me and spent most of your career in a non-Windows environment, then LDAP meant a lot more than the Kerberos components of AD did. But, the LDAP support in AD is just an interface that hides a much richer set of functionality. Though, for most purposes involving username + password authentication and RBAC authorization, LDAP is going to be a much simpler way of integrating with AD than Kerberos. — I’m sure we could find plenty of Microsoft security people who would disagre with that assertion.

Active Directory clients never access the data store directly just like with any other enterprise database system. A client that wants to read or write directory data uses one of the supported Active Directory Service Interfaces (ADSI) to connect to the Directory Services Agent (DSA) and search for that data. The supported ADSIs are:

Active Directory objects have Access Control Lists (ACLs) just like files and folders on an NTFS file system. Attributes within AD objects also have ACLs; so, attributes containing sensitive account information can be further restricted than other attributes.

Domain Members: Windows Servers

A Windows Server instance that is a member of a domain that is not running Domain Services (and Active Directory) is a domain member. Such a server is typically running specific services (may be provided with Windows, may be third-party) that provide a common function to other domain members. All of these could be acting as either Kerberos clients or servers/services. Likely, they are acting as both at the same time in different contexts.

Domain Members: Windows Workstations/Laptops

These are essentially the same thing as what was described above, but are running different flavors of Windows (geared towards desktops, laptops, other devices). Rather than running common services these are running programs/services used to support a single domain user's moment-to-moment work.

Windows Resources

An Active Directory Resource is a shared resource on the network (domain) such as a printer or a shared file system (that can be accessed with proper permissions). By registering the resource in AD, the user does not have to log into individual servers to find and access those resources. Instead, AD can provide information about registered resources in a central place — think about the last time you searched for networked printers at work, they were all listed in one place (we take stuff like that for granted these days).

Logoff

When the user logs off, the credentials cache is flushed and all service tickets — as well as all session keys — are destroyed. The user's shell for the session is terminated and all child processes.

Summary

This blog post describes how Kerberos ties into Windows security to authenticate a user and create a Windows session on a Windows Server or Workstation. This post ties the details I've been describing in my Kerberos Series to Windows security and its Kerberos implementation.

Image: / Hercules and Cerberus LACMA 65.37.151