ניהול סמאות וזהויות ברשתות מיקרוסופט

מאת יהודה גרסטל

הקדמה

כפי שאני מכיר את קהל הקוראים שלי (כלל לא) יש סיכוי לא רע שלא תצליחו לשרוד עד סוף המאמר ולכן החלטתי לחלק חלק מההקדשות והתודות כבר פה בהתחלה. אז תודה מיוחדת להוריי ואשתי שהביאוני עד הלום, להם ולילדיי היקרים שבזכותם יש טעם לחיי.

וכעת למאמר. המאמר מחולק לשלושה חלקים:

1. תיאור המערכת עצמה
2. חולשות וסקירת פרצות
3. התמודדות עם החולשות

בכל אחד משלושת החלקים האלו יופיעו שלושה נושאים אופקיים, כלומר:

- שמירת סיסמאות במערכת לטווח ארוך (נדון בכך בשלושה אופנים: גם כיצד זה עובד בתכלס, גם אלו חולשות ופרצות קיימות במנגנון הזה וגם כיצד מגינים מפני כך).
- שמירה ושימוש מקומי בסיסמאות בזיכרון (ושוב לכל אורך שלושת הנושאים)
- כיצד הזהות וההזדהות עוברים ברשת.

נתחיל מדברים על קצה הרלוונטי ונגיע עד ימינו אנו.

אז איך כל הסיפור הזה עובד? מה קורה בעצם כשאתם מתחברים למערכת חלונות? אתם מזינים את שם המשתמש והסיסמא... ומה אז?

האשים - גיבוב.

בסעיף זה נדון באופן שבו נוהגים לשמור ולאמת סיסמאות בעולם המחשבים - באמצעות גיבוב. אם העיקרון הזה מוכר וברור לכם, הרגישו בנוח לדלג הלאה לכותרת הבאה.

אחת הטעויות השכיחות שיכולים לבצע מפתחים היא שמירת מידע רגיש בצורה גלויה. למשל, לפתח אפליקציה רפואית עם משתמשים רשומים ולשמור את פרטי המשתמשים כמו המידע הרפואי הפרטי שלהם בצורה לא מוצפנת. כאשר גורם לא מורשה מצליח להגיע איכשהו אל מסד הנתונים, הוא מסוגל

לשלוף כמויות אדירות של מידע רגיש ומסווג ללא קושי. **הפתרון הפשוט כדי להימנע מגניבה מעין זו הוא להצפין את המידע הרגיש**.

באופן דומה, כדי שנוכל לאמת התחברות, לכאורה על הסמאות להיות שמורות במערכת היכן שהוא - כדי שאפשר יהיה לבצע השוואה בין הפרטים שמסר המשתמש המזדהה לאלו המקוריים שהוכנסו בפעם הראשונה. כאן, בסיסמאות, יש למתמטיקאים טריק נוסף, שונה מהצפנה. מאחר והמידע המבוקש הוא ברור, ידוע ונקודתי והלקוח/המשתמש גם מספק אותו בעצמו, אין צורך לשמור אותו בצורה כזו שנוכל ממש לקרוא אותו - עלינו רק לבדוק האם הנתונים שסיפק הלקוח הם אותם הנתונים שקבע הוא בעצמו בפעם הראשונה. מה כן עושים?

את המחרוזת של הסיסמא מעבירים תהליך מתמטי חד כיווני. בניגוד להצפנה אין כאן מפתח והתהליך לא ניתן לשחזור ופענוח. מאפיין נוסף של התהליך הוא שלא משנה מה אורך הקלט - תהא זו סיסמא באורך שמונה תווים או מחרוזת קובץ באורך של חמישים מגה-בייט - הפלט של הפונקציה המתמטית יהיה באורך זהה (כדי להשיג תוצאה זו משתמשים ב-"ריפוד" אבל לא נכנס לפרטים כרגע). התהליך הזה נקרא גיבוב, או בלע"ז 'האש' - HASH.

כך בערך זה נראה: בפתיחת חשבון חדש המשתמש מזין את שמו והסיסמא שלו. מאחורי הקלעים המערכת לוקחת את הסיסמא ומעבירה אותה בפונקציית גיבוב מסוימת. רק תוצאת הגיבוב נשמרת במסד הנתונים המקומי ולא הסיסמא עצמה. בזמן אימות גישה למערכת - הלקוח מזין שוב את הסיסמא שקבע בפתיחת החשבון, מתבצע שוב תהליך הגיבוב והמערכת משווה את המחרוזת שנוצרה זה עתה מול זו שקיימת כבר במסד הנתונים שלה. בצורה כזו הסיסמא עצמה אינה נשמרת במערכת לעולם בשום צורה ולמעשה גם נמצאת בשימוש בצורה גלויה בזמן הקצר ביותר האפשרי. אם פורץ הצליח להגיע למסד הנתונים (או כל איזור אחסון שמכיל את גיבובי הסיסמאות) - הוא לא יכול לעשות עם זה דבר.

כמעט...

אין לתוקף דרך להוציא את הסיסמא מתוך הגיבוב, אבל הוא כן יכול לנסות ליצור גיבובים מכל סיסמא אפשרית ולהשוות את התוצאות שלו עד שימצא את הסיסמא המתאימה - נרחיב על כל זה בחלק שדן בתקיפה (בחלק השני).

אלגוריתמי גיבוב במייקרוסופט

עכשיו, אחרי שדנו קצת בגיבובים בואו נדבר על איך מייקרוסופט עושים את זה. בסעיף זה נדון בשלושת הגיבובים הקיימים במערכות חלונות:

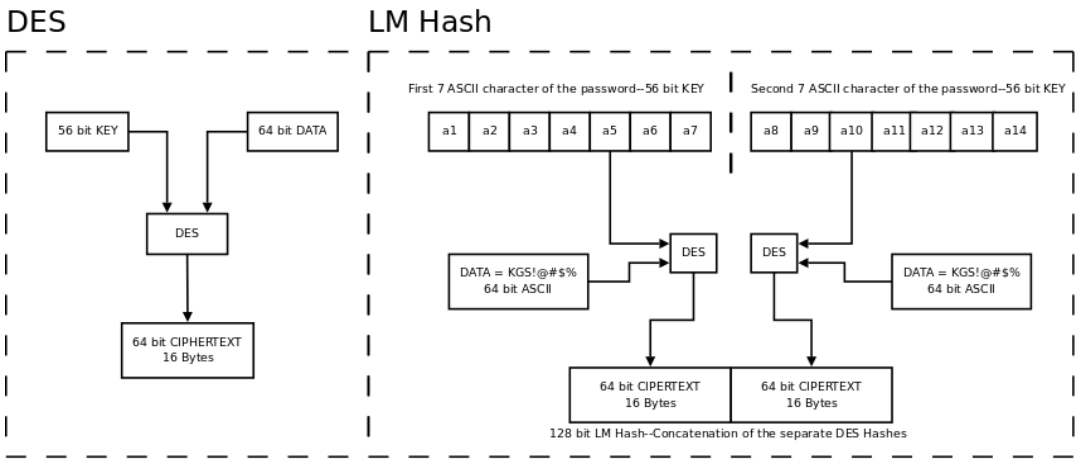
- גיבוב/הצפנה LM
- גיבוב NTLM
- גיבוב MS-CACHE

LM:

נתחיל מההתחלה הכי רחוקה שעדיין רלוונטית: מערכות חלונות 98 ו-NT. במערכות אלו פיתחה מיקרוסופט מנגנון אימות שנקרא LAN Manager או בקיצור LM. מנגנון זה אינו מבצע ממש גיבוב במובן המלא משום שהוא משתמש בפונקציה שנעזרת במפתח הצפנה - זו הצפנה חד-כיוונית. ההצפנה היא מסוג DES והמפתח המשמש להצפנה הוא הסיסמא שלכם עצמה. ההצפנה מתבצעת על מחרוזת קבועה, מפורסמת וידועה: **KGS!@#$%**.

לפני ביצוע ההצפנה המערכת מחלקת את המחרוזת של הסיסמא לשני חלקים של שבעה תווים. (אם אתם יודעים חשבון בסיסי של כיתה ג' הצלחתם להסיק נכון שסיסמאות במערכות שקדמו לחלונות 2000 אינן תומכות בסיסמאות שארוכות מ-14 תווים). במידה ואחד החלקים אינו באורך שבעה תווים מלאים - המערכת משלימה את האורך החסר ו"מרפדת" אותו בתווי NULL.

בנוסף, חשוב לדעת כי המנגנון אינו תומך בתווים מיוחדים, כלומר הוא מוגבל ל-128 סוגי תווים .את שני החלקים השווים של שבעת התווים מעביר ה-LM לאותיות גדולות ורק אז מפעיל את פונקציית ההצפנה על המחרוזת הקבועה שהזכרנו קודם (מכל חלק נוצרת מחרוזת בת 16 תווים). לבסוף האלגוריתם פשוט מחבר את שני החלקים שנוצרו לכדי מחרוזת אחת בת 32 בתים, להלן תרשים אודות מנגנון זה:



[במקור: https://courses.cit.cornell.edu/ece576/FinalProjects/f2008/tt236/tt236/high_level_design.html]

לדוגמא, אם הסיסמא שלי היא "thisISmypass". המערכת תפרק את הסיסמא שלי לשני חלקים:

- החלק הראשון - thisISm
- החלק השני - ypass
- החלק השני **אינו** באורך של שבעה תווים ולכן אנחנו מרפדים אותו בערך כך: ypassXX

כל אחד משני החלקים משמש כמפתח והמחרוזת עוברת תהליך של הצפנה חד כיוונית -

- המילה thisISm הופכת ל-**D478C5B5AB58795A**
- המילה ypassXX הופכת ל-**B7624FF226D45722**
- והמחרוזת המלאה היא: **D478C5B5AB58795AB7624FF226D45722**

אתם יכולים לשחק עם הנושא באתר הזה

הערת צד: כאשר הסיסמא קצרה משמונה תווים, למעשה כל החלק השני ריק ומרופד בתווי NULL כך שהתוצאה תמיד תהיה: 0xAAD3B435B51404EE. באופן הזה ניתן לדעת במבט קצר על מחרוזת הגיבוב האם הסיסמא ארוכה משבעה תווים או אפילו ריקה לגמרי.

עד כאן לגבי שיטת הגיבוב של LM.

NTLM

מאז חלונות 2000 קיים סוג נוסף של גיבוב בשימוש על ידי ה-LAN Manager שלנו. שיטה זו נקראת NT או באריכות - NTLM (ה-NT מגיע מכך שמחלונות 2000 מיקרוסופט החלו לכנות את הטכנולוגיה שלה כ- New Technology).שיטה זו פשוטה למדי והיא משתמשת בפונקצית גיבוב ידועה ומוכרת בשם - MD4. הסיסמא שלנו מוזנת כקלט לפונקציה הנ"ל ושבה אלינו כמחרוזת שונה לחלוטין באורך קבוע של 32 תווים ב-UNICODE. בשיטה זו ניתן להשתמש בסיסמא באורך של עד 127 תווים (מגבלה שקשורה לאורך הקלט בתיבת הזנת הסיסמא, מבחינה תכנותית אפשר ליצור סיסמאות ארוכות יותר).שימו לב שבשיטה זו ניתן גם להשתמש בכל התווים הנתמכים ב-UNICODE (כולל סיסמאות בעברית לדוגמא).

MSCACHE וקרברוס

השיטה השלישית והאחרונה שנדון בה אינה שונה במהותה משיטת ה-NTLM. שיטה זו פועלת בסביבות דומיין של מיקרוסופט. לאלו מכם שלא מכירים מדובר בסביבה "עסקית" של רשת מחשבי חלונות. הסביבה מאפשרת ניהול מסודר ומרוכז של המשתמשים ומשאבי הרשת כמו גם מדיניות ונהלים שיחולו ברמת המחשבים והמשתמשים. בסביבה "מתחם" שכזו קיים תמיד שרת מרכזי המשמש בין שאר תפקידיו גם כשרת לאימות הזדהות. ברשתות מיקרוסופט מבוססות דומיין החל מוינדוס 2000 והלאה מתבצע אימות בעזרת פרוטוקול שנקרא קרברוס. הפרוטקול משמש לזיהוי, אימות ולקבלת הרשאות בין

מערכות ולכן נדון בו מעט יותר בנושא האופקי השלישי של "שימוש בסיסמאות בפועל", כרגע אנחנו רוצים להתרכז באיך הנתונים נשמרים במערכת לאורך זמן.

מטמון

כפי שציינו ממש עכשיו האימות ברשתות ארגוניות מתבצע מול שרת מרכזי (שירות הקרברוס מופעל בדרך כלל על השרת המשמש כ-"Domain Controller" וברשתות גדולות קיימים אף מספר שרתים המשמשים לתפקיד זה), אבל מה קורה כאשר השרתים אינם זמינים? כדי שניתן יהיה להשתמש במחשב ובשירותים מרוחקים בכל זמן, מערכת חלונות מאפשר אימות מול "מטמון" / זמני - CACHE. השם של אימות זה נקרא בפשטות MS-CACHE והוא המשך של מנגנון ה-NTLM אותו כבר הזכרנו. מחרוזת הגיבוב של הסיסמא אותה יצרנו ב-MD4 בשיטת ה-NTLM עוברת תהליך נוסף. אל המחרוזת המגובבת של הסיסמא מצורף שם המשתמש באותיות קטנות ללא שם המתחם (דומיין) והמחרוזת המשולבת עוברת דרך הפונקציה של MD4 פעם נוספת.

אז איפה כל זה נשמר?

כמובן וכאמור בהקדמה, כדי להצליח לאמת את בקשת ההזדהות, כל ההאשים האלו נשמרים בקבצי מערכת במקום כלשהו.

"הרישום" של מיקרוסופט הידוע בכינויו Registry נשמר במספר קבצי מערכת. קובץ SAM שנמצא בנתיב הסטנדרטי הבא: C:\Windows\System32\Config מכיל את כל ההאשים משלושת הסוגים:

- LM
- NTLM
- MS-CACHE

במקרה של MS-CACHE הגיבובים השמורים מוצפנים באמצעות מפתח LSA.

כמו כן מדובר במטמון ולכן המערכת שומרת למעשה רק את הגיבובים של עשרת המשתמשים האחרונים שהזדהו.

גרסאות - תמיכה לאחור

שיטת NTLM משמשת במערכות מסוג חלונות 2000 ו-XP ובגרסאות השרתים המקבילות: 2000 ו-2003. חשוב לדעת אמנם, כי לצרכי תמיכה בשיטות ישנות, מערכות אלו תומכות כברירת מחדל בשיטת LM הישנה - כך שלמעשה הסיסמא נשמרת בשתי התצורות גם יחד.

החל מגרסת Windows Vista והלאה, מערכת חלונות אינה תומכת ב-LM כברירת מחדל, אולם ניתן לשנות זאת ולאפשר תמיכה גם באימות מסוג LM. ניתן לעשות זאת (עם כי זה לא מומלץ) ע"י שימוש ב-Group Policy.

סביבת מתחם (Microsoft Domain)

בסביבות מתחם של מיקרוסופט יש למעשה שני סוגי חשבונות משתמשים:

- משתמשי וקבוצות מתחם / משתמשי רשת או Active Directory בלע"ז.
- משתמשים וקבוצות מקומיים (הרגילים שקיימים בכל מערכת הפעלה של מיקרוסופט)

הסמאות של כל המשתמשים הרשתיים (משתמשי Active Directory) נשמרות בתצורת ה-NTLM שלהן בקובץ מרכזי בשם NTDS.DIT. הקובץ מצוי בכל שרת מסוג Domain-Controler שמשמש לאימות, אך פרטי המשתמשים המקומיים המוגדרים על עמדות הקצה עצמן עדיין שמורים מקומית בכל תחנה ותחנה.

איך התהליך מתרחש בפועל כאשר המחשב דלוק?

עכשיו אחרי שסיימנו את הנושא האופקי הראשון – איפה זה נשמר – בואו נדבר על איך הסיסמאות נשמרות בטווח הקצר ונעשה בהן שימוש מקומי (השלב הבא יהיה שימוש במרחב הרשת).

כאשר המחשב נדלק, מופעל רכיב בשם LSA, ראשי תיבות: Local Security Authority. הרכיב אחראי בין היתר גם על טעינת הגיבובים מאמצעי האחסון ושמירתם במיקום זמין בזיכרון. רכיב ה-LSA מריץ תהליך במערכת שאולי נתקלתם בו מספר פעמים ועכשיו גם תדעו מהו - תהליך בשם LSASS, כלומר Local Security Authority Subsystem Service. LSASS אחראי על כל תהליכי האימות, בין אם מדובר בהזדהות מקומית בכניסה למחשב, בגישה למשאבים שונים ברשת או בניסיונות גישה מרוחקים למשאבים במחשב הנוכחי (כמו קבצים ומדפסות).

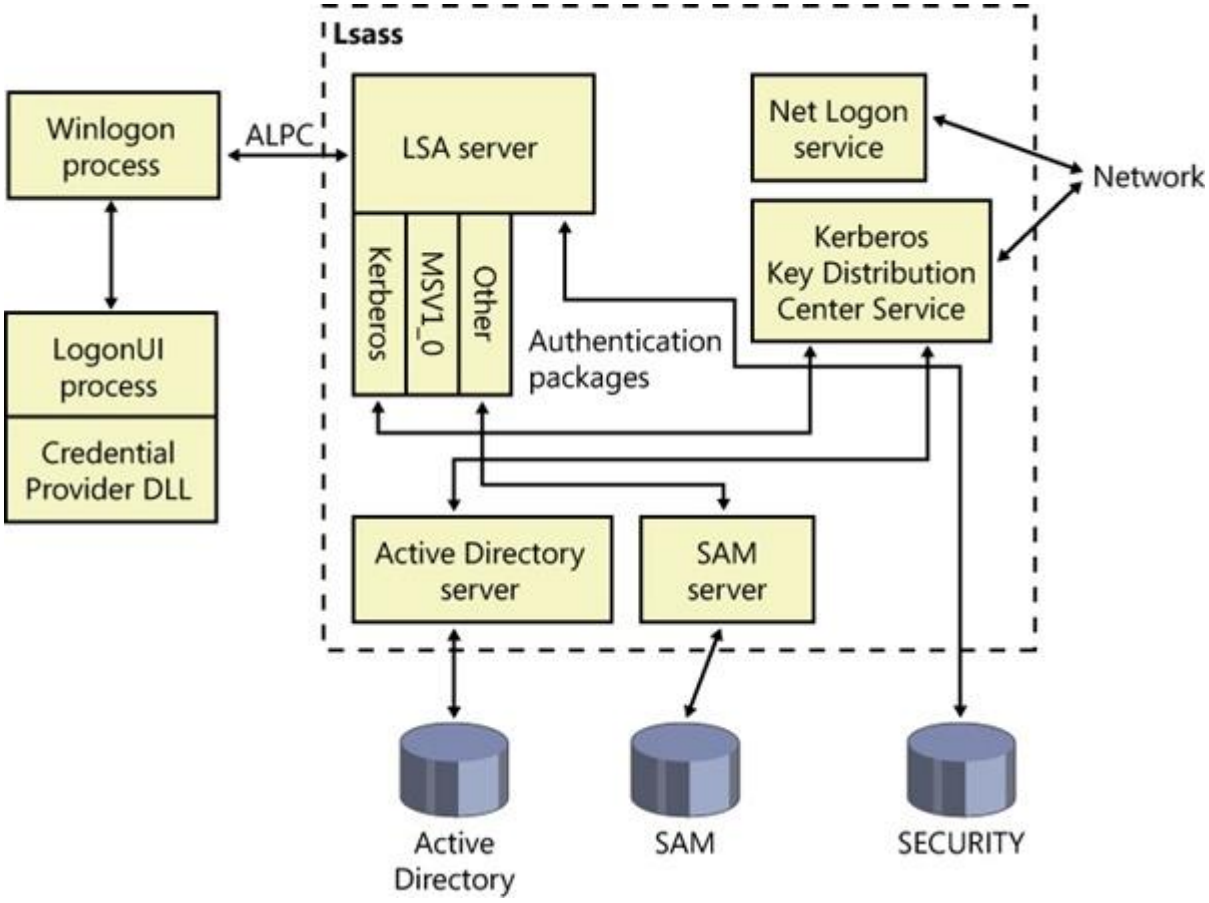
כאשר מופיע מסך כניסה בעליית מערכת חלונות, מאחורי הקלעים תהליך בשם WINLOGON מציג את הבקשה להזנת משתמש וסיסמא. התהליך מעביר את הקלט של המשתמש אל תהליך ה-LSASS יחד עם הגדרה לחבילת האימות שבה יש להשתמש (קרברוס, LM, או NTLM). תהליך ה-LSASS מעביר את הנתונים שקיבל לקבצי DLL רלוונטיים לפי החבילה שמעבדים את הקלט ובמקרה שלנו משווים את התוצאה לתוכן שמופיע בקובץ ה-SAM ומחזירים תשובה אל תהליך ה-LSASS.

שיטות האימות בהן תומך תהליך ה-LSASS בהתקנה סטנדרטית הן:

1. LM, NTLM (MSV_1.0)
2. Kerberos ticket
3. WDigest
4. Terminal Services (TsPkg)
5. PKU2U
6. SCHANNEL

בכל תהליך הזדהות, LSASS פונה לחבילות האבטחה הרלוונטיות (קבצי DLL במערכת) ושומר אצלו את תוצאות האימות של השיטות הללו.

להלן תרשים של כלל הרכיבים המשתתפים בעת תהליך האימות האינטרקטיבי (כאשר המשתמש יושב מול המחשב פיזית ומקליד סיסמה ב-Logon Screen):



[מקור: https://www.microsoftpressstore.com/articles/article.aspx?p=2228450&seqNum=8]

בסביבת דומיין כל גיבובי הסיסמאות נשמרים בשרת המרכזי של סביבת מיקרוסופט, Domain Controller או בקיצור DC. מי שאחראי על תהליך האימות בשרת גם הוא תהליך/שירות LSASS

קישורים להרחבה בנושא:

תהליך הזדהות אינטראקטיבי במערכת:

https://msdn.microsoft.com/en-us/library/windows/desktop/aa376107(v=vs.85).aspx

המשך תהליכי הזדהות מבוססים על ההזדהות האינטראקטיבית הראשונה:

https://msdn.microsoft.com/en-us/library/windows/desktop/aa378779(v=vs.85).aspx

תהליך האימות מול חבילות האבטחה:

https://msdn.microsoft.com/en-us/library/windows/desktop/aa378338(v=vs.85).aspx

סיימנו לפרט את אופן השימוש הסטטי לטווח ארוך וכן את האחסון של הסמאות ופרטי ההזדהות כאשר המערכת רצה. עכשיו נתחיל לדבר על כיצד מועברות הזהויות בפעולות הדורשות תקשורת וגישה בין רכיבי רשת שונים.

שימוש ברשת

יש לא מעט שימושי רשת לפרטי ההזדהות של מיקרוסופט. ברשתות ביתיות אנחנו מכירים בעיקר את שיתוף הקבצים והמדפסות, אבל ברשתות ארגוניות לפרטי ההזדהות יש לא מעט תפקידים נוספים:

- הרצת פקודות מרוחקת - RPC
- גישה גרפית מרוחקת - RDP/Terminal Services
- כניסה למסדי נתונים מסוג MSSQL Server
- גישה מרוחקת למערכת הרישום של מיקרוסופט (Registry)
- אימות לשירותי WEB בשרתי מיקרוסופט IIS (דוגמה לשימוש נפוץ: Sharepoint)
- מערכות צד שלישי רבות שמתממשקות בפרוטוקול LDAP אל ה- Active Directory

התפקוד של כל הרשת מנוהל ממקום מרכזי אחד - זה הכוח (וגם החולשה) של רשתות דומיין.

איך מתבצע תהליך האימות?

האימות הבסיסי נכון לשיטות LM ו-NTLM הראשונות היה פשוט למדי. כדי לא להעביר את הסיסמא ברשת וגם לא את הגיבוב משתמשים בפרוטוקול challenge-response הבא:

1. הלקוח (מבקש השירות) שולח בקשת נתונים לשרת ("השרת" בהקשר זה הוא הגדרה לוגית - יכול להיות שמדובר במחשב, או שירות שפועל על אותו מחשב עצמו)
2. השרת מגיב באתגר - הוא שולח מחרוזת נתונים רנדומלית (NONSE) ומבקש מהלקוח להצפין אותה.
3. הלקוח מקבל את המחרוזת ומצפין אותה באמצעות הגיבוב של המשתמש הנוכחי שהזדהה.
4. הלקוח שולח את המחרוזת המוצפנת יחד עם שם המשתמש של מבקש השירות.
5. השרת בודק האם קיים אצלו משתמש בשם זה ושולף את הגיבוב שקיים אצלו לפרוטוקול.
6. השרת משתמש בגיבוב ששלף כדי להצפין את מחרוזת ה-NONSE ששלח בעצמו ומשווה את התוצאה למחרוזת המוצפנת שקיבל מהלקוח.

7. במידה והתוצאות שוות, סימן שהלקוח מחזיק באותו גיבוב שהשרת מחזיק וכנראה מחזיק גם בסיסמא הנכונה - הלקוח מקבל את התוכן שביקש.
8. חבילת האבטחה יוצרת LOGON SESSION ומעבירה אותו לתהליך ה-LSA
9. תהליך ה-LSA יוצר טוקן שמכיל **LUID קיצור של LOGON ID**

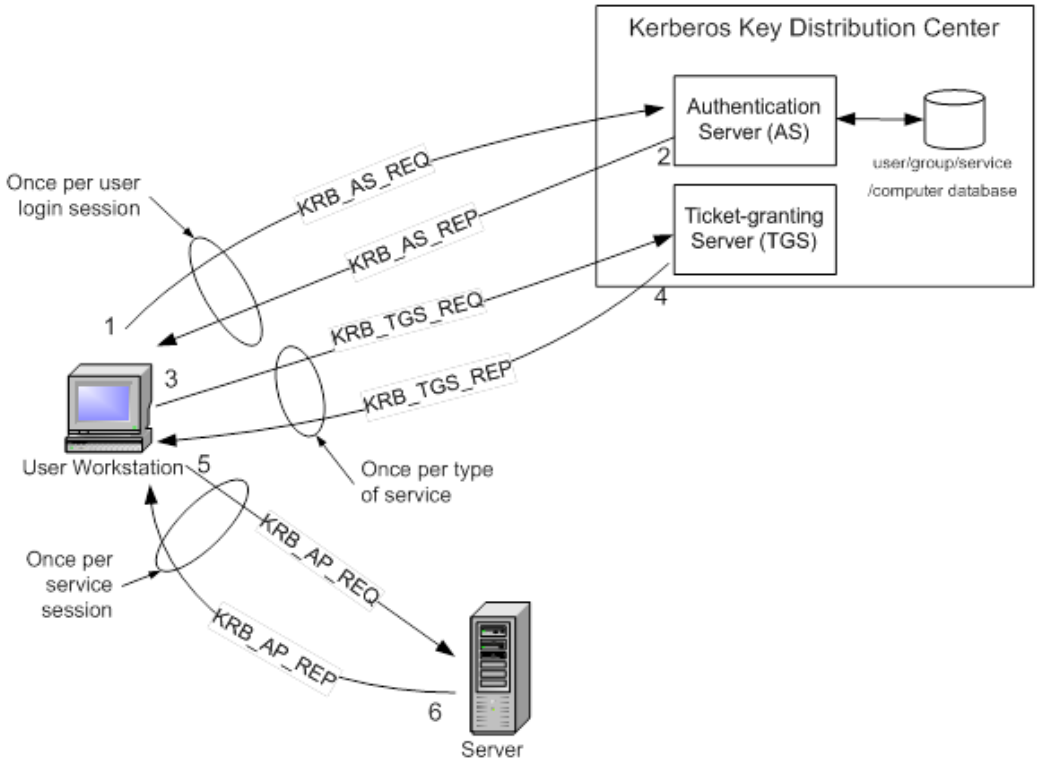
הפרוטוקול החדש יותר מסוג NTLMv2 מוסיף לתהליך הזה שלב אחד קטן בלבד ומאפשר ללקוח להוסיף Nonse משל עצמו, נדון בסיבות לכך בחלק השני שידון בפריצה.

בסביבת דומיין, העניינים מתנהלים קצת אחרת. הפרוטוקול הינו קרברוס. לא נכנס כאן לכל ההסברים של הפרוטוקול משום שזה כמעט מאמר בפני עצמו, רק אקצר ואומר שהפרוטוקול נשען על מספר דברים עקרוניים:

- אימות מול שרת מרכזי שמכיל את פרטי ההזדהות של כל המשתמשים ברשת.
- שימוש בטיקטים בכל תהליך האימות וההתקשורת לאחר ההזדהות
- שימוש בחתימות זמן

תהליך ההזדהות בשלבים:

להלן ציור אשר ממחיש את הצורה שבה עוברת בקשת שירות בין לקוח לשרת בשיטת קרברוס



דוגמאות לשימושים נוספים

קיימים עוד שימושים רבים בשיטות אימות אלו מעבר לשימוש ה"קלאסי" ברשתות מקומיות. מפאת קוצר היריעה (כל נושא הינו פוטנציאל למאמר בסגר גודל של המאמר הנ"ל) לא אפרט על נושאים אלו, אך אתן כותרות וקישורים לשם ההבנה והרחבה למתעניינים:

- **אופן פעולה NTLM בגישה לשירותי WEB:**

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

http://www.innovation.ch/personal/ronald/ntlm.html

https://technet.microsoft.com/en-us/library/cc778868(v=ws.10).aspx

- **הזדהות בפרוטוקול RDP, אופן פעולה:**

http://download.microsoft.com/download/9/5/E/95EF66AF-9026-4BB0-A41D-A4F81802D92C/[MS-RDPBCGR].pdf

כאמור, אלו שתי דוגמאות קיימות, אך יש עוד מספר לא קטן של מנגנונים במערכת ההפעלה (ואף במערכות נוספות) אשר עושות שימוש במנגנוני אימות אלו.

כאן סיימנו את הנושא "האנכי" הראשון.

ועכשיו הגענו לחלק המעניין - אז איך שוברים את כל התהליכים האלו?

שלב האחסון לטווח ארוך

אם יש לנו גישה כלשהי למערכת הקבצים כאשר המערכת כבויה, ניתן לשלוף בקלות את הגיבובים מתוך קובץ ה-SAM המרכזי. בהינתן הרשאות ניהול מקומיות על מערכת ניתן לשחרר את הנעילה של קובץ ה-SAM המרכזי גם בזמן שהמערכת עובדת ולשלוף את הגיבובים. כאמור, סוגי הגיבובים הנשמרים תלויים במערכת ובסביבת הרשת בה היא נמצאת. אבל מה הם עוזרים לנו אם אין אפשרות לבצע פעולה חוזרת הפוכה של תהליך הגיבוב או ההצפנה (במקרה של LM) ולהוציא את הסיסמא?

כדי לקבל את הסיסמא שיצרה את הגיבוב הדרך היחידה המוכרת כיום היא ליצור את כל הגיבובים האפשריים ולהשוות אותם לגיבוב ששלפנו מתוך המערכת. שיטה זו עשויה להיות איטית למדי ונדבר קצת על האופנים השונים של ניצולה.

הדרך הראשונה לתקוף סיסמאות ובמקרה הזה גם גיבובים הוא לנסות סיסמאות קלות לניחוש:

1. מבוסס מילון - שימוש בסיסמאות נפוצות כמו "123456" וכדו'
2. מבוסס מידע אישי - תאריך לידה, מספר זהות, מספר טלפון, שמות ילדים וכדו'
3. ניסוי כל הסיסמאות האפשריות על בסיס טווח תווים הגיוני (למשל אותיות קטנות בלבד או שילוב של אותיות קטנות ומספרים בלבד)

הדרך השנייה היא ניסוי כל הסיסמאות האפשריות ללא שום מגבלה או צמצום טווח האפשרויות.

הערה: שימו לב שהתיאוריה אומרת שקשה למצוא שתי מחרוזות שיצרו את אותו הגיבוב, אך בפועל הדבר אפשרי. כמות האפשרויות שמכסה **מחרוזת התוצאה** של גיבוב NTLM לדוגמא היא 16 בחזקת 32 (מחרוזת באורך 32 תווים כשבכל תו יש 16 אפשרויות - תו הקסדצימאלי כלשהי), שזה בעצם 2 בחזקת 128 אפשרויות. בעוד כמות הסיסמאות האפשרית מצד המשתמש היא לפחות 256 בחזקת 128 (אם ניקח רק את תווי ASCII בלי להתחייחס לכל ה-UNICODE), שזה בעצם 2 בחזקת 2,048. כלומר יש לפחות 2 בחזקת 1920 אפשרויות שהן **בוודאות גמורה כפולות של מחרוזות אחרות**. מחקר מעניין שנעשה הציג "התנגשויות" שכאלה בתהליך גיבוב מסוג MD5. ניתן לקרוא על כך במאמר הבא:

https://eprint.iacr.org/2013/170.pdf

פיצוח גיבובי LM

כפי שחלקכם כבר הבין לבד, את הגיבובים שנשמרים בשיטת LM ניתן לפצח די בקלות.

מדובר בסיסמאות מוגבלות למדי בטווח התווים האפשרי ובנוסף, לפני ביצוע הגיבוב מתבצעת העברה לאותיות גדולות. כלומר שטווח כל הסיסמאות האפשריות שלנו בסך הכול מגיע **לסדר גודל של 64**

בחזקת 14 ובוודאי שזה גם מקל על פיצוח מבוסס מילון או מידע אישי. מעבר לכך, הסיסמא אינה באמת באורך של 14 תווים אלא שני חלקים של שבע, מה שאומר שאם נפצח את כל הסיסמאות האפשריות המורכבות מאותיות גדולות מספרים ותווים מיוחדים באורך של שבעה תווים בלבד - נוכל לפתוח כל גיבוב של LM.

עדיין מדובר בלא מעט זמן עבודת עיבוד ומיד נדון בשיטה לקיצור התהליך הזה, אבל קודם בואו נדבר על הגיבובים האחרים.

פיצוח גיבובי NTLM

שיטת NTLM כבר קשה יותר לפיצוח, מדובר בטווח תווים גדול יותר משמעותית ואורך סיסמא כמעט לא מוגבל. למעשה ההגבלה של שיטת הגיבוב עצמה קצרה יותר מכמות הסיסמאות האפשרויות בשיטת NTLM (כפי שהוזכר בהערה). עדיין, גם במקרה זה, השימוש במילונים ובמידע אישי עובד לא מעט פעמים ומחזיר אותנו אל הבעיה האמיתית שהיא בחירת סיסמאות נכונה מצד המשתמשים/המערכת. גם בתקיפה בזמן אמת של גיבובי NTLM החישוב עצמו של האלגורתים אורך מעט יותר זמן ולמעשה מאט משמעותית את קצב התקיפה. שוב, כדי להתמודד עם בעיות אלה מיד נדון בשיטה לפיצוח גיבובים שבה לא נעשה חישוב בזמן אמת.

סוג הגיבוב השלישי שדיברנו עליו הוא... גיבובי סביבת דומיין זמניים שנקראים גם MS-CACHE. כדי לדון בנסיונות הפריצה לגיבוב הזה הגיע הזמן לדבר קצת על SALT ועל פיצוח בשיטה של עיבוד מוקדם.

פיצוח גיבובים לא מקוון ועיבוד מוקדם

רמת הביצוע של תקיפה בסגנון של ניסיון סיסמאות אפשריות משתנה בהתאם לסוג הגיבוב אותו מנסים לתקוף, בכלי התקיפה שבו משתמשים, באיכות הסיסמא ובמחשב (או מחשבים) שבאמצעותם מתבצעת שיטת הפיצוח. (תודו שאתם מתים על העברית שלי).

בכל המקרים אם נוכל לבצע את התקיפה מול מחרוזת הגיבוב עצמה ולא מול מערכת חיה יהיה לנו הרבה יותר קל. חישוב והשוואת מחרוזות היא פעולה קלה בהרבה מאשר התחברות לשרת לוגי כלשהי וציפייה לתגובה. מלבד זאת קיימות הגנות שמונעות ניסיונות התחברות חוזרים ונשנים. לכן, לאחר שהשגנו את מחרוזת הגיבוב מקובץ ה-SAM, נרצה לייבא אותה אל מחשב או רשת מחשבים שתבצע בשבילנו את עבודת החישוב באופן "לא מקוון" – כלומר ללא ניסיון ממשי להתחבר למערכת.

המהירויות הסטנדרטיות נעות בין נסיונות של כמאה אלף סיסמאות לשנייה ועד כ-600 מיליון סיסמאות לשנייה על מחשב בודד עם הכלים הנכונים:

http://blog.distracted.nl/2009/05/entibr-ntlm-password-brute-forcer.html

שימוש בשיטת פיצוח סיסמאות מבוזרת יכולה לקחת אותנו קדימה לפי כמות המחשבים, כלומר גודל האוניברסיטה שהשתלטתם עליה או גודל הבוטנט שהצלחתם ליצר לעצמכם באמצעים חוקיים כאלו ואחרים - ועשוי להגיע עד קצבים מטורפים של כמה מאות בליוני סיסמאות לשנייה.

אם ניקח את שיטת הגיבוב הישנה ביותר הקיימת היום - LM וננסה לתקוף מחשב שהרגע השגנו גישה לגיבובים שלו מדובר ב**כחצי שעת** עבודה מפרכת בקצב הזוי של **300 מיליון סיסמאות לשנייה** כדי לעבור על כל האופציות ההגיוניות (26 אותיות גדולות + 10 ספרות + כ-14 תווים מיוחדים בשימוש סטנדרטי) לא רע, אבל לא תמיד יש לנו חצי שעה לבזבז וברוב המקרים אין לנו כוח עיבוד מספק בשביל הקצב הנזכר.

ניקח את שיטת NTLM החדשה לדוגמא (תזכורת: LM מופיע רק במערכות XP ומטה) - מדובר בלא מעט זמן, שימוש בסיסמא המורכבת משמונה תווים עם שילוב של אותיות מספרים ותווים מיוחד מביא אותנו לכ-76 בחזקת 8 אפשרויות. גם בקצב המטורף של 300 מיליון סיסמות בשנייה ללא עצירה אנחנו מדברים כאן על כחודש וחצי חישוב.

כדי להתמודד עם הבעיה הזו, במקום לפרוץ את הסיסמאות בזמן התקיפה, אנחנו פורצים אותן לפני.

מה?! איך פורצים סיסמאות לפני שהשגנו את הגיבובים?! פשוט מאד - הפרוטוקול קבוע. אם נבצע עיבוד של כל הסיסמאות האפשריות ונשמור אותן כטקסט קריא, נוכל בקלות בזמן תקיפה להשוות את הגיבוב ששלפנו מהמערכת של הקרבן למסד הנתונים האדיר שלנו עם כל הסיסמאות האפשרויות ונבדוק מה הסיסמא שיוצרת את הגיבוב הרלוונטי. השיטה הזו נקראת "עיבוד מקדים" - Precompiled password attack. השיטה למעשה ממירה זמן בנפח אחסון. במקום לבזבז זמן בניסיון תקיפה, אנחנו משתמשים בכל הזמן שעומד לרשותנו ומאחסנים את התוצאות - הרבה מאד נפח אחסון אבל חוסך זמן בעת הצורך.

כאשר משווים את הפרס של יכולת שליפת סיסמאות בקלות מכל מערכת מיקרוסופטית קיימת - מדובר בהשקעה משתלמת. אפשר להשקיע חודש וחצי ואפילו שנה כדי לכסות יותר ויותר אפשרויות.
למעשה כדי להוכיח את הנקודה הזו והחולשה של הפרוטוקול יש לא מעט שירותים באינטרנט שמחזיקים מאגרים כאלו ומאפשרים שירות בחינם ובתשלום. הנה כמה לדוגמא:

http://www.hashkiller.co.uk/ntlm-decrypter.aspx

http://www.onlinehashcrack.com/list-cracked-hash.php?h=ntlm

https://crackstation.net

בנוסף, קיימות גם קהילות שמשתפות פעולה בהרחבת הטבלאות הקיימות.

אתם יכולים לדמיין באיזה נפחי אחסון אסטרונומיים מדובר וכדי להתגבר על בעיית נפח האחסון משתמשים בשיטה שנקראת "טבלאות קשת בענן" (או בלע"ז: Rainbow Tables). ההסבר על אופן הפעולה הוא מחוץ למסגרת של מאמר זה ואתם מוזמנים לפנות לכאן ולכאן כדי להבין איך זה עובד. אפשרות העיבוד המקדים מעמידה את כל שיטת השימוש בגיבובים בסכנה וכדי להימנע מתקיפה זו עושים שינוי קטן בתהליך ההזדהות שנקרא המלחה - מלשון מלח (Salt) ☺.

הבעיה שהמלחה באה לפתור היא שייצוג הגיבובים זהה בכל המערכות בעולם בכל המצבים ולכן יש לנו אפשרות לבצע חישוב מקדים בידיעה שהמערכת הנתקפת תשתמש באותה שיטה בדיוק אותה אנו מכירים. כדי להימנע מהייצוג הקבוע מוסיפים אלמנט רנדומלי שכתוקפים לא נוכל לצפות אותו מראש.

תאוריה - המלחה

בזמן שמירת הגיבוב הראשונית המערכת מייצרת מחרוזת אקראית קצרה שנקראת SALT ומוסיפה אותה לסיסמא בתהליך הגיבוב. המערכת שומרת גם את המלח וגם את הגיבוב הסופי באותו מקום. כאשר משתמש מבקש להזדהות המערכת שולפת את "המלח" מתוך האחסון ומבצעת את תהליך הגיבוב עם אותו "מלח". כל מערכת מייצרת מלח משלה ולכן כתוקפים אנחנו לא יכולים לצפות מראש את הפרוטוקול ולהשתמש בשיטה של עיבוד מקדים.

זה בדיוק מה שקורה עם שמירת הגיבובים של משתמשי דומיין בשיטת MSCACHE. כמו שהזכרנו בחלק הראשון - MSCACHE הוא למעשה גיבוב בשיטת NTLM שחיברו אליו שם משתמש והריצו את תהליך הגיבוב פעם נוספת. השיטה הזו מכריחה אותנו כתוקפים לבצע את הפיצוח **בזמן אמת** רק **לאחר** שהשגנו את "המלח" שהוא שם המשתמש.

נניח ופרצתי למחשב בסביבת דומיין וקיבלתי את ה-Hash-ים של פרטי חשבונות מסוימים, את LM ו-NTLM אני יכול לחפש בקלות בטבלאות מוכנות מראש. את הגיבובים של MS-CACHE אני צריך לקחת יחד עם שם המשתמש ולבצע תקיפה בזמן אמת. הבחור הנורווגי הנחמד הזה יעשה לכם עבודה מהירה יותר עם הפריצה הזו, אבל עדיין זה עשוי לקחת המון המון זמן... אז מה עושים? עזבו אתכם שטויות, למה לשבור את הראש על גיבובים שנשמרים בדיסק כאשר הסיסמא נשמרת באופן גלוי...

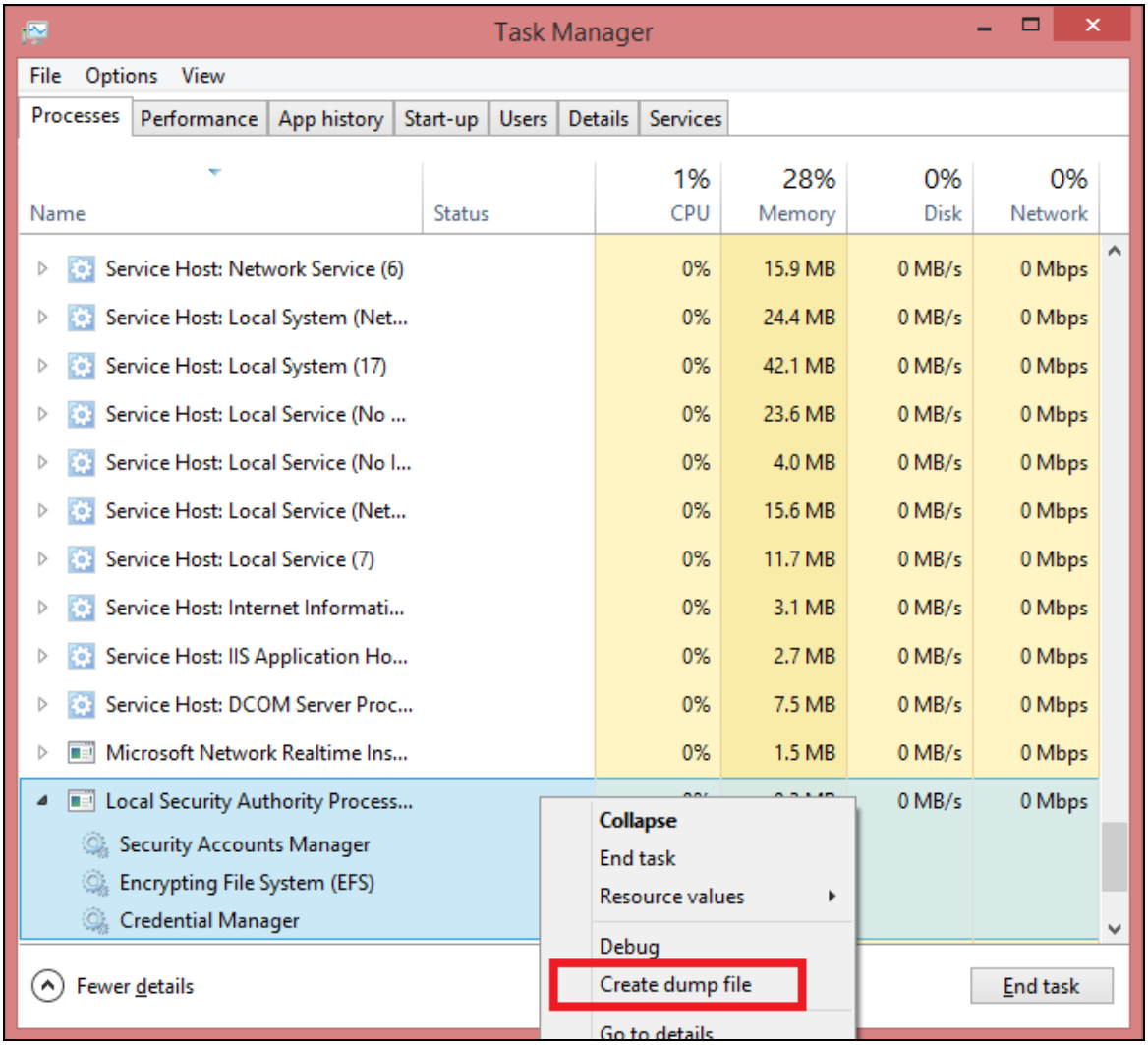
שלב האחסון בזיכרון

לצערנו (או לשמחתנו), בשלב האחסון בזיכרון יש מעט מאד הגנות.

החלק שמעניין אותנו בסיפור הזה הוא שמופיעים גם פרטי אימות זמניים של משתמשים שהתחברו מרחוק מה שמעלה את הסיכויים למצוא פרטי הזדהות של משתמשים לא מקומיים לשרת/תחנת קצה שנפרצה. דוגמא בולטת לכך היא יכולת לשלוף סיסמאות גם של חיבורי RDP מסוג שירותי טרמינל - שירות החיבור המרוחק של מיקרוסופט שמנהלי מערכת עושים בו שימוש רב. בנוסף, נעשה שימוש

בחבילת אבטחה של Web Digest, זהו רכיב אימות שמשמש להזדהות מול שרתי ווב בצורה מתוחכמת יותר מאימות ווב בסיסי. העניין הוא שחבילה זו זקוקה למחרוזת המשתמש והסיסמא עצמם כדי ליצור את תהליך ההזדהות ולכן שומרת בזיכרון את הסיסמא של המשתמש בצורה גלויה לגמרי...
הגישה לזיכרון של תהליך ה-LSASS מצריכה אמנם הרשאות ניהול מקומיות בזמן ריצה (בניגוד לגישה ישירה למערכת הקבצים), מצד שני רוב הפעמים נדרשת גישה בהרשאות ניהול מקומיות - או גישה פיזית כזו שמאפשרת השגת הרשאות ניהול מקומיות במהירות.

כלים מוכרים שמבצעים בשבילכם את התקיפה ומחלצים את המחרוזות הרלוונטיות: WCE ו-Mimikatz.
גרסה עדכנית של Mimikatz גם מאפשרת שליפה מתוך קובץ DUMP של תהליך ה-LSASS.



ויש לא מעט כלים נוספים שמאפשרים לבצע – לדוגמא בפוסט הזה

חלק מעניין ומשמעותי נוסף בשלב השימוש בזיכרון זו היכולת לשלוף נתוני קרברוס שמאפשרים התחזות לכל משתמש ברשת - ניתן לראות טבלה מסכמת של בנג'מין "דלפי" בעמוד הזה

שלב השימוש ברשת

בשלב הזה נתייחס לשני סוגי פרצות:

1. השגת מידע רגיש – גיבובים וסיסמאות
2. גישה ממשית למשאבי רשת

אין לכם עדיין גישה לאף תחנה ברשת ואתם סתם יושבים על הכבל. מה אפשר לעשות? שני הדברים הפשוטים ביותר להבנה וביצוע בשלב זה הם חיקוי של תהליך האימות ונסיונות פריצה של שיחות אימות שעוברות ברשת.

חיקוי תהליך האימות

העברת גיבובים - Pass The Hash

בחלק של "איך זה עובד" הסברנו שלמעשה בפרוטוקול NTLM לא נעשה שימוש מתמיד במשתמש ובסיסמא. בזמן האימות הראשוני המערכת מייצרת גיבוב מסוג NTLM ומשתמשת בו כדי לאמת את זהותה מול מערכות ושירותים שונים. לפי זה, כדי לבצע אימות מול שירותים שונים ברשת אנחנו צריכים רק את הגיבוב ולא את הסיסמא בכלל. בשילוב עם המתקפות משלב האחסון הסטטי והאחסון בזיכרון אנחנו בהחלט יכולים למצוא את גיבוב הסיסמא של המשתמש. בשלב הזה נוכל לבצע אימות מול כל משאב ברשת שהמשתמש שגילינו את הגיבוב שלו מורשה לגשת אליו.

נניח לדוגמא כי בוב מתחבר למחשב באמצעות הסיסמא שלו. הסיסמא מועברת לתהליך ה-LSASS. תהליך ה-LSASS מעביר את המשתמש והסיסמא לכל החבילות שהוא תומך בהן ביניהן MSV1_0 תהליך MSV1_0 מייצר גיבוב מסוג LM ו-NTLM ומעביר אותו בחזרה ל-LSASS, LSASS מחזיק את הנתונים בזיכרון. בניסיון גישה לתיקיית שיתוף במחשב מרוחק המערכת פונה למחשב המרוחק ומקבל אתגר מסוג NTLM. LSASS שולף מהזיכרון את שם המשתמש והגיבוב ומשתמש בהם כדי לענות לאתגר ה-NTLM במידה וקיים במערכת המרוחקת משתמש עם גיבוב זהה, המערכת המרוחקת תאשר את השימוש

> **הערת צד:** שימו לב ששיטה זו תעבוד רק במקרה שהמשתמש קיים במערכת המרוחקת או באחסון לטווח ארוך במערכת הקבצים, בתוך קובץ ה-SAM (משתמש מקומי), או באחסון בזיכרון (משתמש שהתחבר למערכת לאחרונה).

כפי שציינו כבר ברשתות ארגוניות התקשורת בין המחשבים מתנהלת בפרוטקול קרברוס ולא בשיטת NTLM. בשיטת קרברוס נעשה שימוש בכרטיסים ולא בגיבוב של המשתמש ושיטה זו של העברת הגיבוב (PTH) לא תעבוד. מה שחשוב ומעניין לדעת פה הוא שניתן להגיד למחשב המרוחק כי איננו תומכים בקרברוס ולדרוש ממנה בעצם מעבר (ירידה) לשיטת NTLM הבטוחה פחות. אחת הדרכים הפשוטות לעשות זאת היא פנייה למחשב המרוחק באמצעות כתובת ה-IP שלו. בצורה זו נעשה שימוש אוטומטי בשיטת NTLM.

דוגמאות למתקפות שניתן לבצע בשלב זה:

המשותף לחלק מן התקיפות הללו נמצא בתהליך התקשורת שלאחר ההזדהות. תהליך ההזדהות ברשת כולל תהליך של אימות "מבקש השירות", אבל לאחר האימות התקשורת מתבצעת באופן "פשוט" ללא שימוש בהצפנה שנקבעה בשלב ההזדהות (כמו בפרוטוקול SSL לדוגמא). בעקבות זאת, אם צלחנו את תהליך האימות בצורה כלשהי נוכל להתחבר אל השרת ללא קושי. לאלו מכם בעלי רקע במערכות וב, הדבר דומה מאד לגניבת מפתח השיחה או העוגיה של המשתמש. (Cookie or Session ID).

מאפיין נוסף הוא היכולת שלנו כתוקפים לתפוס תקשורת שמכילה גם את האתגר (NONSE) וגם את התשובה המוצפנת של הלקוח (בפרוטוקול NTLM המוכר לנו) ואז אפשר לבצע מתקפת ראש בקיר ולנסות את כל האפשרויות כדי לגלות את מחרוזת הגיבוב ששמשה להצפנת האתגר (NONSE).

SMB Reflection

הבסיס לתקיפה זו הוא שהקרבן שלנו משמש בו זמנית גם כשרת וגם כלקוח במקביל. בפועל פונים לשרת ומבקשים לקבל גישה למשאבים, כאשר השרת מבקש להזדהות אנחנו **משקפים** לו את האתגר שלו עצמו. "השרת" עונה לנו כלקוח על האתגר שלו עצמו ועכשיו יש לנו תשובה בשבילו.
אתם בטח שואלים את עצמכם למה שהשרת יסכים להיות הלקוח שלנו פתאום? אז ככה - שכתוקפים אנחנו מחכים לבקשה ברשת ממחשב כלשהו לגישה בתצורת NTLM. ברגע שאנחנו מזהים בקשה שלא נענית אנחנו מתחזים למחשב שאמור לתת את השירות. עכשיו מתבצע תהליך אימות כפול במקביל, שנינו משמשים בו זמנית גם כשרת וגם כלקוח. המחשב של הקרבן מנסה לקבל מאיתנו שירות (שלמעשה אנחנו רק מתחזים ולא באמת מסוגלים לספק לו) אנחנו כתוקפים מבצעים במקביל תהליך אימות אל המחשב המרוחק שביקש בעצמו לתקשר איתנו.

1. הקרבן (כלקוח) שולח בקשת שירות ואינו מקבל תשובה.
2. התוקף (כשרת) עונה סבבה, אבל רק רגע...
3. התוקף (כלקוח) שולח בקשת שירות לקרבן
4. הקרבן (כשרת) שולח אתגר הזדהות (NTLM).
5. התוקף (כשרת) שולח את אותו אתגר לקרבן (כלקוח).
6. הקרבן (כלקוח) עונה לאתגר ששלח התוקף.
7. התוקף (כלקוח) שולח את התשובה שקיבל מהקרבן (כלקוח) אל הקרבן (כשרת) בערוץ השני.

בלי שאנחנו יודעים את הסיסמא ובלי שאנחנו יודעים את הגיבוב. פשוט משקפים בחזרה לקרבן את אותו אתגר אימות ששלח לנו ונותנים לו לענות על דרישת ההזדהות של עצמו בשבילנו.

SMB Relay

בשנת 2001, אחד מחברי קבוצת ההאקינג Cult of The Dead Cow בשם Sir Dystic פרסם כלי ומסמך המפרט אודות שיטת תקיפה בשם SMB Relay. המתקפה נועדה לגשת למשאבי רשת על שרת מרוחק באמצעות תקיפת MITM. התוקף דואג לתווך מצב של התחברות בין לקוח לשרת ואז מתחזה ללקוח ו"ממשיך" השיחה בעצמו.

התוקף גורם למשתמש לגשת אל המחשב שלו בבקשת SMB (שיתוף קבצים) באמצעות מייל או דף WEB כלשהו. ברגע שהמשתמש הקרבן מנסה להתחבר אל המחשב של התוקף באמצעות SMB, התוקף מעביר (Relay) את הבקשה אל השרת הרצוי ומציב עצמו בתווך התקשורת בין SMB Client ל-SMB Server. הכלי יודע להעביר את התקשורת בין הצדדים – כמו בכל מתקפת MITM קלאסית – עד לקבלת הרשאה ואז להישאר מחובר לשרת תוך התחזות למשתמש המורשה (אותו עיקרון של בעיית "גניבת קוקי" שהזכרנו במקתפת ה-Reflection SMB.

כמו כן, הכלי מחלץ מנתוני התקשורת את המחרוזת המוצפנת מה-NONSE ומגיבוב ה-NTLM שנשלח על-ידי הלקוח. את המחרוזת המוצפנת (אל תבלבלו בינה לבין גיבוב רגיל) ניתן לנסות לפצח באופן לא מקוון באמצעות מספר כלים. למידע נוסף אודות מתקפה זו ניתן לקרוא:

http://www.xfocus.net/articles/200305/smbrelay.html

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

Responder

אגב חילוץ נתוני תקשורת ופיצוח אתגרי NTLM לקבלת סיסמא מהסעיף הקודם, אי אפשר להתעלם מכלי התקיפה החמוד הזה. Responder יושב על הרשת בשקט יחסי ומנסה למשוך אליו תקשורת מסוג LLMNR, NS-NBT או MDNS שאינה מקבלת מענה, הוא מסוגל להתחזות לשירותים שונים כמו SMTP, FTP, SQL ועוד ולחכות שהקרבן פשוט ישלח אליו את הסיסמאות. ברשתות מתחם כפי שכבר הסברנו האימות ברשת מתבצע באמצעות קרברוס שלא מאפשר פיצוח של תשובת האתגר בפשטות כמו בפרוטוקול ה-NTLM.

ריספונדר מאזין לרשת ולבקשות שונות, הוא מכריח את הלקוח/קרבן להתחבר אליו ישירות דרך כתובת ה-IP. מסיבה לא ברורה, מערכות מיקרוסופט משנמכות לאימות מסוג NTLM במקום קרברוס כאשר הן פונות לשירותי רשת באמצעות כתובת IP (כנראה מתוך הנחה שמחשבים במתחם מיקרוסופט יחזיקו שם מחשב נורמאלי בפרוטקול NBNS או DNS).
ברגע שהלקוח מוכן לעבוד ב-NTLM, שוב אנחנו שולחים לו אתגר ואת התשובה מנסים לפצח תוך שימוש בכל שיטות ניחוש הסיסמאות המוכרות לנו.

SMB Replay

בשנת 2010 שני חוקרי אבטחה בשם Hernan Ochoa ו-Agustin Azubel מחברת AmpliaSecurity פרסמו במסגרת הכנס BlackHat הרצאה על מחקר שעשו אודות מתקפות שונות בפרוטוקול SMB של חברת מיקרוסופט וספציפית על כשלים במימוש ה-NTLM. במסגרת המחקר שלהם הציגו כשל ספציפי במנגנון הרנדומיזציה שנכתב לטובת ייצור ה-Nonce. הבעיה היא שתוך מספר בקשות עשוי לחזור שימוש באותו ה-NONSE.

במתקפה זו על התוקף לצותת לתקשורת ולאסוף ממנה כמה שיותר זוגות של ה-Challenge וה-Response שנשלחו בין הלקוח לבין השרת. לאחר מכן יוזם התוקף מספר תהליכי הזדהות בעצמו אל השרת עד אשר מתקבל אתגר זהה לזה שחילצנו בזמן ההסנפה. על Challenge כזה התוקף כבר יודע לענות, הוא מחזיק תשובה שלו שמורה משלב ההאזנה.

לקריאה מלאה של המאמר הנ"ל, המפרט מעבר למתקפה זו מתקפות רבות על הפרוטוקול, ניתן לקרוא בקישור הבא:

http://www.ampliasecurity.com/research/NTLMWeakNonce-bh2010-usa-ampliasecurity.pdf

שילוב של תקיפות אלו עם מתקפה כגון NBNS Spoofing יכול להיות קטלני. מאמר המפרט על אופן התקיפה הנ"ל פורסם בגיליון ה-32 של המגזין על-ידי אפיק קסטיאל וניתן לקרוא עליה בקישור הבא:

http://www.digitalwhisper.co.il/files/Zines/0x20/DW32-1-NBNSSpoofing.pdf

קיימת מתקפה נוספת בשם Pass The Ticket עליה לא ארחיב במאמר זה, אך היא מעניינת ביותר ומומלץ לקרוא עליה בקישור זה:

http://blog.gentilkiwi.com/securite/mimikatz/pass-the-ticket-kerberos

מעין סיכום לנושא ה-SMB

רעיון האימות של NTLM מתבסס על ידיעת האתגר שנשלח מהמחשב המרוחק וידיעת הסיסמא שמשמשת ליצירת הגיבוב. בפועל, הגיבוב מספיק לנו ובהאזנה לרשת נוכל לראות אתגרי NTLM עוברים גלויים וחוזרים מוצפנים. ניתן לקחת את האתגר ולבצע ניסיונות פיצוח בשיטות ניחוש הסיסמאות השונות (ידע מוקדם, מילון ומתקפת ראש בקיר). חישוב הגיבוב של סיסמא אפשרות + הצפנה באמצעות האתגר ואז השוואת התוצאה לאתגר המוצפן שזיהינו ברשת.

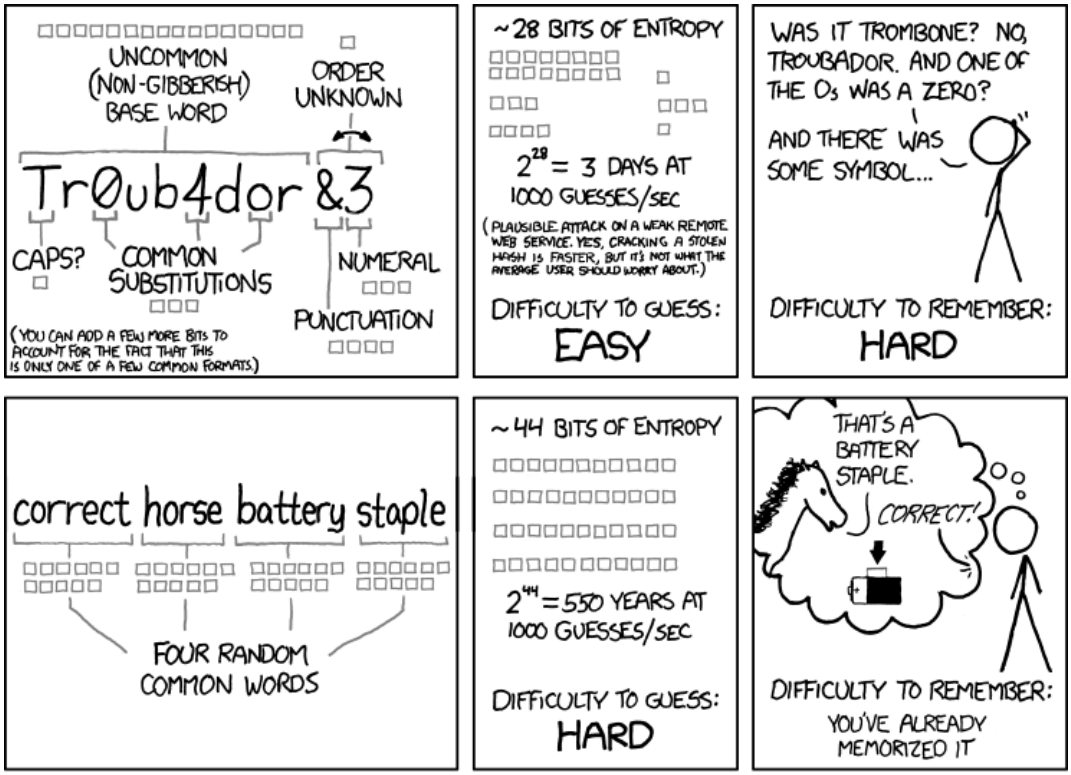
אך גם בלי שנצטרך לבצע MITM ו/או האזנה מלאה לרשת ניתן לבצע ניסיונות לפיצוח סיסמאות. כמו שתיארנו בחלק הקודם של SMB Reflection - ניתן לחכות ל-"בקשה יתומה" לשירות NTLM ואז להתחזות לתחנה שמעניקה את השירות ולהגיש אתגר חוזר. המחשב שמבקש את השירות יחזיר לנו את האתגר המוצפן. עכשיו יש לנו את המחרוזת הרנדומאלית של האתגר וגם את האתגר שמוצפן בעזרת הגיבוב. שוב, ניתן לבצע ניסיונות פיצוח אופליין בלא "להרעיש" ברשת.

סיכום נושא תקיפה

עד כאן דיברנו על שיטות התקיפה השונות, נגענו בלא מעט שיטות, אך קיימות עוד הרבה. עם זאת, נעצור כאן על מנת להתחיל לדבר על החלק השלישי - אם וכיצד ניתן בכל זאת לאבטח את הרשת שלנו מפני מתקפות אלו?

שיטות הגנה

עד כה דיברנו על הנושא בכלליות והצגנו את הנושא מנקודת מבטו של התוקף, כעת נשנה את זווית הבחינה שלנו ונראה כיצד מנהל רשת וצוות ה-IT יכולים למנוע או להקשות על מתקפות כגון אלו להתממש. מלבד חילוץ הסיסמאות ClearText בעזרת כלים כגון Mimikatz אשר דורשים גישה ישירת לזיכרון של התהליך, או שימוש ישיר בשיטות-PassTheHash / PassTheTicket, מלבד אלו, על מנת להשיג את הסיסמאות המקוריות של המשתמש עלינו לנסות לשבור אותם בעזרת ניחוש. גם אם יש וגם אם אין Salt, הסיכויים שלנו להצליח במשימה זו תלויים ישירות בחוזק של הסיסמה, ועל כן שימוש בסיסמאות חזקות (הגדרה לא פשוטה כל כך) בהחלט משפר את הסיכויים שלנו בתור משתמש. ישנן לא מעט המלצות על שימוש נכון בסיסמאות, לא ארחיב עליהן כאן, אולם יש מספר קונספטים שחשוב לזכור אותם. השתמשו תמיד בסיסמאות שקל לזכור אך קשה לנחש, דוגמא טובה לכך יש בקומיקס המוכר הבא:



[במקור: https://xkcd.com/936]

זו נקודה משמעותית מאד בעיניי שפותרת חלק נכבד מהבעיות עם סיסמאות בכלל. מעבר לכך, כדאי לחשוב גם על דרך לניהול הסיסמאות המרובות, אם זה בהטמעה של תוכנה לניהול סיסמאות ואם זה בהדרכה של משתמשים איך לנהל את הסיסמאות שלהם בצורה יעילה. נרחיב כעת על אמצעים טכנולוגיים לפתרון בעיות אלו.

רגע קצר לפני כן, עוד דבר שחשוב לזכור הוא שבסופו של דבר, הבעיה האמיתית בפן הטכנולוגי (ולא האנושי) היא בעיה מהותית. זו דוגמא קלאסית בעיניי לכשל ברמת התכנון והעיצוב של המערכת. פרוטוקולי השיחה וההזדהות שמשתמשת בהם מיקרוסופט (לצרכי תמיכה לאחור) הם ישנים על גבול העתיקים שלא הושקעה בהם אותה מחשבה ותכנון מונחה הגנת מידע שיש היום בשוק. כל הפתרונות המיושמים הם טלאים על דלי מלא חורים, בסופו של דבר מיקרוסופט תצטרך לצאת עם חבילת אבטחה חדשה שתסיר כל תמיכה לאחור בפרוטוקולים בעייתיים כמו LM ו-NTLM.

הגנה על המידע השמור בכונן:

את הגיבובים יש לשמור לטווח הארוך, קשה להמנע מזה, כדי להפוך את האפשרות הזו למאובטחת יותר כדאי להתרכז בסיסמאות קשות לניחוש - (ארוכות, שימוש בתווים לא סטנדרטיים כמו עברית). בנוסף, ניתן לבטל את השימוש באלגוריתם LM, אם אנו לא צפויים להתקל במערכות הפעלה הישנות מ-2000, ממש אין צורך באלגוריתם הנ"ל. על מנת לבטל אותו פשוט נוסיף בעורך הרישום את המפתח:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa

ובו את:

NoLMHash

למידע נוסף בעניין:

https://support.microsoft.com/en-us/kb/299656

ביטול משתמשים מקומיים - גיבובים של MSCACHE עוברים "המלחה" באמצעות שם המשתמש. תקיפות על גיבובים מסוג MSCACHE מסתמכות לרוב על שמות משתמשים צפויים כמו USER או ADMINISTRATOR. החלפת שמות משתמשים מובנים כאלה ממזערת את החשיפה לשימוש בעיבוד מקדים כדי לחשוף את הסיסמאות.

כמו כן ברשתות דומיין של מיקרוסופט אין סיבה אמיתית להחזיק משתמשים מקומיים פעילים ומומלץ לאחר החלפת השם גם להכניס אותם למצב Disabled. ניתן גם לבטל שימוש ב-MSCACHE לטווח ארוך:

http://support.microsoft.com/kb/172931

http://www.ampliasecurity.com/research/wcefaq.html#thisisnotcachedump

לוודא התקנה של עדכון MS 2871997 כנגד PTH אשר מבטל הרשאות רבות למשתמשים מקומיים בסביבת דומיין.

הגנה על המידע אשר נמצא בזיכרון:

ביטול חבילות אבטחה לא נדרשות כמו NTLM בסביבת דומיין, WDIGEST בכלל וכו'. שדרוג מערכות הפעלה לגרסת 8.1.

בזמן שימוש במרחב הרשת:

במידה והרשת מאפשרת זאת, נוכל לאפשר הזדהות לתחנה אך ורק באמצעות NTLMv2, נעשה זאת ע"י שינוי הערך הבא בעורך הרישום ל-5:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\lmcompatibilitylevel

ניתן לעשות זאת גם באמצעות GPO:

Computer Configuration -> Windows Settings -> Security Settings -> Local Polices -> Security Options -> Network Security -> LAN Manager Authentication Level

סיכום

במאמר זה הצגתי את רוב שיטות האחסון וההזדהות הנהוגות היום במייקרוסופט. דיברנו על שלושה מחזורי חיים של הסיסמא, על הכונן, בזיכרון ובמרחב הרשת והצגתי דרכים שונות לתקוף את המבנים והפרוטוקולים השונים הללו כמו דרכים להגן מפני פרצות מסוג זה.

תודות

מעבר לתודות שנתנו בתחילת המאמר, ברצוני להעניק הקדשה מיוחדת אחרונה וחביבה לתלמידי שלב כלשהו אדר התשע"ה, שהם וגם אני רצינו מאד לעסוק יחד בנושאים אלו אך הזמן הניף את חרבו הקצרה כדי להכריע סופית את הקרב האלמותי בין סופרמן לבאטמן, בשבילכם: אילן, רועי, גל, אמיתי, עמרי, אסף, מיכל, הילה, אור, אלירן, טל, מור, אורן, אפי, יוסי ותמר. הייתם אחלה תלמידים.

עוד תודה מיוחדת שמורה לעורכי המגזין שבזכותם יש לכולנו תוכן איכותי באמת בצורה נגישה ונוחה כמו שלא ראיתי בשום מקום אחר. שאפו. והם גם עורכים מעולים ומקצועיים.

ביבילוגרפיה ומקורות להרחבה

Wikipedia - suit yourself.

בנושא LSA:

https://msdn.microsoft.com/en-us/library/windows/desktop/aa378326(v=vs.85).aspx

בנושא NTLM:

https://www.youtube.com/watch?v=fyk-0rub6Kw

https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-grutzmacher.pdf

בנושא PTH:

https://media.blackhat.com/us-13/US-13-Duckwall-Pass-the-Hash-WP.pdf

http://www.harmj0y.net/blog/penetesting/pass-the-hash-is-dead-long-live-pass-the-hash/

בנושא סיסמאות בכלל:

https://technet.microsoft.com/library/hh994558(v=ws.10).aspx

https://technet.microsoft.com/en-us/library/hh994565.aspx

בנושא מטמון סיסמאות מתחם:

http://webstersprodigy.net/2014/02/03/mscash-hash-primer-for-pentesters