הפרוטוקול Kerberos v5

מאת אפיק קסטיאל (cp77fk4r)

הקדמה

Kerberos הוא פרוטוקול שרת-לקוח מבוסס הצפנה אשר תפקידו למנוע דליפת מידע מהרשת, או בשפה המקצועית - "פרוטוקול אימות" (Authentication Protocol). כחלק מתפקידו הוא מוגדר גם כפרוטוקול לניהול מפתחות (Key Infrastructure). צורת מימוש תפקידיו שונה משאר הפרוטוקולים הדומים לו (כגון CHAP, PAP, RADIUS , TACACS ועוד) באופן שבו הוא פותר את בעיות האבטחה הידועות במנגנונים כאלה - בהמשך הטקסט ניגע גם בהם.

קצת רקע כללי: Kerberos פותח כחלק מפרוייקט חופשי בשם Athena. את הפרוייקט התחילה לפתח קבוצה קטנה יחסית ב-MIT בשנת 1979. הפרוייקט נועד לנהל את מידור ההרשאות במערכת הקבצים של רשת האוניברסיטה. לאחר תיקונים רבים של מספר כשלי-אבטחה בפרוטוקולי החבילה, שוחררה החבילה כ"מוצר מוגמר" רק בגירסתה החמישית. פיתוח כלל החבילה (שכללה בין השאר את Zephyr, Thin client, OLHואת-Discuss) הסתיים קצת יותר מעשר שנים לאחר מכן ב-1993 ונועדה לשימוש הן למערכות PC והן למערכות MAC. זמן קצר לאחר מכן גם שוחררה חבילת ה-"Generic Security Services" שכללה ממשקי API רבים עבור Kerberos בכדי לעזור למפתחים ליצור תוכניות שיכלו "לדבר" עם הפרוטוקול וכך להשתמש בחוסנו.

בכדי לממש מספר הצפנות במהלך ה-Hand-Shake של הפרוטוקול, החבר'ה ב-MIT השתמשו בהצפנות כגון RC ו-DES (שלאחר מכן שונו ל-AES ו-SHA-1 במערכות החלונאיות), ועקב כך ארצות הברית הגדירה אותו כ-"כלי נשק" ולכן היה אסור לייצא אותו אל מעבר ליבשה המערבית.

בנוסף, הפרוייקט מהווה בסיס גדול מאוד לפרוטוקולי ניהול הרשאות אחרים, אשר משמשים אותנו כיום כגון LDAP ו-Active Directory.

הצורך בפרוטוקול

למה אנחנו צריכים אבטחה ברשת הפנימית של האירגון?

נניח ויש לנו באירגון מספר מחלקות/מדורים. לכל מדור יש את הנתונים המיוחסים לאותו מדור. למשל, למחלקת ניהול הכספים יהיה מידע על ההכנסות וההוצאות של הארגון. בנוסף יתכן ויהיה לה את מספרי חשבונות הבנק של האירגון וכדו'. אם מדובר במחלקת משאבי אנוש - סביר להניח שהם מאכסנים במחשביהם את הנתונים האישיים על כל עובד, מצב משפחתי, תפקיד, אולי גם תיק מעקבים וכד'. באירגון מסודר, אסור שלעובד ממחלקה אחת תהיה גישה לקבצים של מחלקה אחרת - המידע אמור להיות ממודר. נכון שכל עובדי הארגון שייכים לאותו הגוף, אבל כל אחד צריך לדעת רק את מה שהוא עוסק בו ואת דרכי הממשק שלנו עם הגופים השונים.

אנחנו יודעים שאנחנו לא רוצים שלכל משתמש תהיה גישה לכלל הרשת, אבל איך עושים את זה? על ידי מידור חשבונות בעזרת ניהול ההרשאות ע"י הגבלת המשתמשים לפי קבוצות או לפי תפקידים.

נניח ויש לנו שלושה מחלקות: מחלקת כספים, משאבי אנוש ומחלקה של עובדי הניקיון, בכל מחלקה יש לנו ראש מחלקה ועשרה עובדים.

לכל עובד יש מחשב. כל עובד הוא משתמש מוגבל ברמת המחשב שלו. בנוסף, כל ראש מחלקה הוא מנהל ברמת הרשת הפנימית של המחלקה שלו, ואנחנו - ראשי האירגון הזה - מנהלים ברמת הרשת כולה. זאת אומרת שאנחנו יכולים ליצור שלושה קבוצות:

- **מנהלים ראשיים** - מנהלים ראשיים מוגדרים כמנהלים ראשיים בכלל האירגון, להם יש גישה לכל מחשב ולכל משאבי הרשת.
- **מנהלי מחלקות** - מנהלי מחלקות מוגדרים כמנהלים מוגבלים בכלל הרשת אך כמנהלים ראשיים ברשת הפנימית של המחלקה שלהם.
- **משתמש מקומי** - משתמש מקומי מוגדר כמשתמש מוגבל או כמנהל ראשי במחשב הפרטי שלו אך מוגבל הן לרשת המחלקתית והן לכלל הרשת האירגונית.

בצורה כזאת, אנחנו יכולים להבטיח שאף משתמש לא יוכל לגשת או לקרוא מידע ממחשבים שהוא לא אמור לגשת אליהם.

בדיוק בכדי לממש את המודל שהצגתי לכם עכשיו, פותחו הפרוטוקול Kerberos ודומיו. הפרוטוקול מקבל בקשת גישה למשאב מסויים ברשת האירגונית ותפקידו של Kerberos היא להבין מי בעצם נמצא מולו והאם הוא אכן רשאי לגשת לאותו משאב. אם אכן מדובר במשתמש מורשה - ניתנת הגישה. אם מדובר במשתמש לא מורשה Kerberos יזרוק אותו מכל המדרגות. או לפחות ככה הוא אמור לעשות.

נבחן מה קורה כאשר Kerberos מקבל בקשת גישה למשאב מגורם חיצוני.

מימוש

הרעיון הכללי

הרעיון הכללי שעומד מאחורי Kerberos הוא ניצול היתרונות של העבודה מול גורם שלישי כ-KDC (שזה קיצור של Key Distribution Center). במימוש של Kerberos מדובר בעצם בגורם שלישי שמחולק לשני גורמים נפרדים אחד מהשני, הראשון ישמש בעת ביצוע ה-Hand-Shake כ-AS - Authentication Server בכדי שנוכל לדעת במי מדובר, והשני ישתמש כ-TGS - Ticket Granting Server בכדי ליצור את החתימה הרלוונטית לאותו חיבור Ticket.

Kerberos מגיב בגדול כמו שרת proxy ונכנס כמתווך מקדים לפני תקשורת בין Client ל-Server ברשת, אך תפקידו כ-proxy יגמר ברגע שהחיבור אומת.

ניתוח ה-Hand-Shake

בכדי להגדיר Client מסויים כפעיל ברשת, עליו לקבוע מפתח אימות ולתאם אותה עם Kerberos מראש (סומנה כ-CK - Client Key). לאחר מכן הוא יוכל לתפקד עם שאר רכיבי הרשת. נניח וה-Client מעוניין להתחבר ל-Server מסויים ברשת (שקבע עם Kerberos מפתח אימות מראש- סומן כ-SK על אותו משקל), לפני שיווצר החיבור בינהם, על ה-Client להזדהות מול Kerberos והמהלך יתבצע באופן הבא:

1. ה-**Client** מחולל מפתח (N) זמני-חד-פעמי (נקרא גם Nonce) ושולח אותו לשרת ה-Kerberos ביחד עם Credential לחיבור עם ה-**Server**.
2. בזמן זה, KDC מחולל מפתח (K) ומגדיר אותו כ-"פעיל" לזמן קצר אשר נקבע מראש (KT). בעזרת CK (מפתח האימות של **Client**) הוא מצפין את N+K+KT+פרטי ה-Credential של ה-**Server** כיחידה אחת - P1, ולאחר מכן ה-TGS מחולל "Ticket" אשר כולל את K+KT+פרטיו של **Client**, ומצפין אותו בעזרת ה-SK (**מפתח האימות של Server**) כיחידה שניה-P2. את P1+P2 הוא שולח ל-**Client**.
3. ה-**Client** מפענח את-P1 בעזרת CK ומאמת כי:
    1. הפרטים אכן נכונים.
    2. KT - אכן בתוקף.

אם תוצאות הבדיקה יצאו חיוביות, אזי אכן מדובר בשרת Kerberos אותנטי והחבילה אכן בתוקף, ועל כן ה-**Client** מכין Authenticator, אשר מכיל את:

1. Time Stamp ע"פ שעונו המקומי.
2. מאפייני זהותו.
3. מאפייני הבקשה ל-**Server**.
4. נתוני הבקשה.

את ה-Authenticator הוא מצפין בעזרת K ושולח את הבקשה הזאת ל-**Server** ביחד עם P2. **שימו לב ש-Client שולח את P2 למרות שאין לו מושג מה יש בפנים- כי כמו שראינו, P2 הוצפנה בעזרת SK אשר ידוע רק ל-Server ול-KDC! - ההנחה היא שאם P1 אומתה כאותנטית אז אפשר להניח בלב שקט שגם P2 אותנטית.**

4. ה-**Server** מפענח את P2 שנשלח מה-**Client** ע"י ה-SK. לאחר הפענוח ה-**Server** יכול לחלץ את K. בעזרת K הוא מפענח את Authenticator ומאמת כי:
   1. ה-Time Stamp - אכן רלוונטי.
   2. נתוני הזהות.

אם הנתונים אכן תקפים, אזי אין ספק שמדובר ב-**Client** שאכן מאושר אצל ה-KDC ולכן **Server** מאפשר ל-**Client** גישה.

את השלבים הבאים לא חובה לממש, והם לא חלק מהפרוטוקול הבסיסי, אך כדאי ואף מומלץ לבצעם, השלבים 5 ו-6 דומים מאוד ל-3 ול-4 רק בכיוון ההפוך והם עוזרים ל-Client לוודא כי הוא אכן מדבר עם Server אותנטי:

5. ה-**Server** שולח ל-**Client** את ה-Time Stamp מוצפן בעזרת K.
6. ה-**Client** מפענח את החבילה שהגיע מה-**Server**, מפענח אותה בעזרת K ואם הTime-Stamp הוא אכן ה-Time Stamp המקורי (שהוא עצמו שלח ל-**Server** בשלב 3) אזי אפשר להניח שמדובר ב-**Server** אותנטי ו-**Client** מאפשר את החיבור.

שימו לב שבכל שלבי השיחה, גם אם נשלחו פרטים חסויים, כגון K, או ה-Time Stamp, הם נשלחו מוצפנים, ולכן, גם אם נניח אליס הצליחה להאזין לתשדורת של בוב, היא לא תקבל שום מידע חיוני שיעזור לה להבין על מה בעצם בוב מדבר. הרעיון מאחורי Kerberos הוא פשוט אימות המשתתפים בשיחה ע"י ערך אשר נקבע מבעוד מועד עם גורם חיצוני (שרת ה-KDC).

רק לאחר שה-Hand-Shake עבר בשלום, מתחיל ה-Triple Hand-Shake בין ה-Client וה-Server, וכך, ה-Server יוכל לדעת שאכן מדובר במשתמש אמיתי, וכך יוכל המשתמש לקבל גישה לאותו משאב רשת בצורה מאובטחת.

עוד נקודה, כשמדובר על ה-KDC ועל ה-TGS, לאו דווקא מדובר על שני שרתים שונים פיזית, וכיום מקובל להריץ את שני השירותים הללו על אותו שרת.

יתרונות וחסרונות

יתרונות:

- **גמישות** - הפרוטוקול מוגדר כפרוטוקול קריפטוגרפי מודולרי, זאת אומרת שאפשר להלביש עליו כל אלגוריתם הצפנה סימטרי (דו-כיווני) מבוסס מפתח אשר ישמש כגורם ההצפנה במהלך ה-Hand-Shake. בתחילה השימוש הנפוץ ביותר היה ב-DES, אך כיום, רוב השימוש שלו הוא ב-RC4 (בדומה לרשתות האלחוט המשתמשות ב-WEP) ובכדי לאמת את שלמות ה-Packets, הוסיפו לו את SHA-1.
- **תוצר / מאמץ**- בעזרת הפרוטוקול אנחנו מקבלים תוצר (אימות המשתמשים) במעט מאוד שלבים ובמעט מאוד מידע שעובר ברשת.

חסרונות:

- **שימוש בחותמת זמן** - שימוש בחותמת זמן נועד בכדי למנוע זיוף של בקשות מקוריות ע"י התוקף בעזרת שליחתן שנית (Replay Attack), אך פתרון זה גם יוצר את אחד החסרונות הגדולים של Kerberos. בגלל שבמהלך ההזדהות, שרת ה-KDC ושרת ה-GTS משתמשים בחותמות-זמן (ה-Time-Stamp), חובה על ה-User וה-Client להיות מסונכרני שעון עם שרת ה-Kerberos. אך בכדי לסנכרן את השעונים אי אפשר להשתמש בפרוטוקול עצמו- כי בכדי להשתמש בו על המשתתפים להיות מסונכרני שעון (וחוזר חלילה...)
- **איכסון ה-CK** - אם לתוקף יש גישה פיזית לאחד המחשבים המדברים עם הפרוטוקול- הוא יוכל לגלות בקלות את ה-CK, כי עליו להיות מאוכסן באותו אופן ששרת ה-KDC מבין אותו, מה שאומר שאי אפשר להצפין את ה-CK בעת איכסונו על המחשב המקומי.
- **שימוש בגורם חיצוני** - כמו שראינו, שימוש בפרוטוקול מחייב את המשתמשים לדבר תחילה דרך שרתי ה-Kerberos, ולכן אם קורה מצב והשרתים קרסו- תעבורת הרשת תחסם ומשאבי הרשת לא יוכלו לדבר אחד עם השני.

סיכום

זהו סופו של המאמר, אני מקווה שלמדתם ממנו. תמיד טוב לדעת מול מה אנחנו עומדים ועובדים, ותמיד טוב להכיר את הטכנולוגיות שנמצאות מאחורי הקלעים, מי שרוצה ללמוד עוד על הנושא, אפשר למצוא המון מידע בקישורים הבאים:

- הפרוייקט Athena באתר של MIT - http://web.mit.edu/acs/athena.html
- Kerberos באתר של MIT - http://web.mit.edu/kerberos
- איך להקים שרת Kerberos בבית - http://www.ornl.gov/~jar/HowToKerb.html