פתרון שאלה 2

כדי לבצע התקפת מילון לא מקוונת קודם כל נצטרך לזהות את המטרה שלנו. בקוד שיצרנו בשאלה אחת החולשה המרכזית שאותה נרצה לתקוף בפרוטוקול היא תהליך רישום הלקוח אשר מתבצע בשרת האימות בפונקציית do_reigster_client_request בקובץ server.py. בתוך פונקציה זאת, מתבצעת הצפנה של הסיסמא של המשתמש בעזרת פונקציית calculate_hash ונשמרת. אם תהליך ההצפנה הוא חלש או שהסיסמאות שהמשתמש בחר הן קלות לניחוש, הן יכולות להיות חשופות למתקפת מילון (אשר מנצלת סיסמאות חלשות על ידי ניסיון התחברות עם מילון רחב של סיסמאות חלשות ונפוצות). כדי לבצע את המתקפה, נצטרך להשיג גישה לסיסמאות המוצפנות, ניתן לעשות זאת במספר דרכים כגון פריצה לאחסון השרת או ניטור תעבורת הרשת ומציאת הפקטות שנשלחות בין הלקוח לבין השרתים. לאחר מכן, נצטרך להבין את לוגיקת ההצפנה והפענוח, הבנה של השימוש בקריפטוגרפיה במערכת, באופן ספציפי איך מפתחות AES נגזרים מתוך סיסמאות של משתמשים, ואיך הצפנת המידע ופענוחו מתבצעים בפונקציות aes_encrypt ו aes_decrypt.

לאחר שהבנו את כל זה, נבצע את ההתקפה באופן הבא:

1. נשיג את רשימת הסיסמאות המוצפנות בשרת – על ידי פריצה לשרת האחסון או על ידי האזנה וניטור תעבורת הרשת בפורטים שעליהם השרת רץ וחילוץ המידע הזה מהפקטות הנשלחות בין הלקוח לשרת האימות ולשרת ההודעות.
2. נייצר מילון של סיסמאות נפוצות אשר בעזרתן נבצע את מתקפת המילון. נצפין את כל הסיסמאות באותה שיטת הצפנה שהשרת עושה בה שימוש (calculate_hash) שזאת הצפנה מסוג sha256. לאחר מכן נבצע השוואה של כל סיסמא שהצפנו ממילון הסיסמאות שלנו, אל הסיסמאות המוצפנות שהצלחנו למצוא בשרת.
3. עבור כל סיסמא שנמצא עבור הצלחה, נגזור את מפתח ההצפנה AES מתוכו.
4. נפענח הודעות – במידה ויש לנו גישה להודעות מוצפנות אשר הוצפנו עם מפתח הAES של המשתמש שפרצנו את סיסמתו, ננסה לפענח את המידע הזה בעזרת מפתח הAES שגזרנו מתוך הסיסמא שנפרצה. במידה ונקבל הודעות הגיוניות נבין שאכן פרצנו את הסיסמא הנכונה והצלחנו לפענח את ההודעות בעזרתן.

כדי למנוע את המתקפה נצטרך לחזק את אבטחת המערכת, ניתן לעשות את זה בדרכים הבאות:

1. שימוש בפונקציית יצירת מפתחות (KDF) חזקה יותר כגון PBKDF2, אשר ייצר ערך רנדומלי ייחודי עבור כל משתמש וישפר את תהליך גזירת מפתחות הAES מהסיסמאות.

2. שימוש ב SALT – הוספת ערך רנדומלי לכל סיסמא לפני שמתחילים בתהליך ההצפנה, ככה כל סיסמא תקבל גיבוב ייחודי אפילו אם יש סיסמאות זהות בין משתמשים, דבר אשר יקשה על תהליך הפענוח ולא יאפשר שימוש במילון סיסמאות נפוצות.
3. נעילת חשבונות לאחר ניסיונות התחברות כושלים מרובים – במידה ויש ניסיון התחברות למשתמש עם סיסמא שגויה יותר מ 10 פעמים נחסום את הגישה למשתמש הזה לזמן מסוים או לאלתר עד לבדיקה של מנהל המערכת. באופן זה נמנע את היכולת המרכזית של המתקפה שהיא בעצם ניסיון של מספר עצום של סיסמאות עד למציאת התאמה.
4. מדיניות סיסמאות חזקות – נכריח משתמש להשתמש בסיסמא ארוכה עם תווים ומספרים ואותיות קטנות וגדולות, ככה הסיכוי שהסיסמא שלו תופיע במילון סיסמאות נפוצות יקטן משמעותית.