**ניתוח איומים לפרויקט MessageU**

**30.10.2021**

**נכתב על ידי פאקינג בוזקוב**



רקע

הפרוייקט MessageU הינו רשת חברתית אשר מאפשרת למשתמשיה תקשורת מוצפנת מקצה לקצה, קרי E2EE.

התקשורת מבוססת ארכיטקטורת שרת-לקוח, בה המסרים בין הלקוחות עוברים תחילה בצד השרת לאחסון, ומאפשרת למשתמשים למשוך הודעות מיועדות להן בעת הצורך.

לטובת קיצור המסמך, לא נפרט את כלל התכולות הנדרשות שכן ניתן למצוא אותן במסמך פירוט המטלה.

מטרת מסמך זה היא לסקור את החולשות והאיומים האפשריים המוצעות בפרוטוקול, בארכיטקטורת ובתפעול הפרוייקט.

המסמך יציג קודם לכן הנחות מוצא עליהן הוא יתבסס בסקירת האיומים, משאבים עליהם ישנה חשיבות להגן, והצעות לתיקון החולשות והסרת האיומים.

המסמך יבנה לפי סיווג התוקפים לפי סבירות בהתאם להנחות המוצא אשר יוצגו בהמשך.

משאבים להגנה

המידע הבא יוגדר כחשוב לצורך המשך המסמך, ונקבע שיש להגן על משאבים אלה בעדיפות ראשונה.

* **הקובץ me.info (נתייחס כ'אשרה' מעתה והלאה)**

קובץ זה מכיל את הפרטים האישיים של כל משתמש:

* שם המשתמש
* המזהה היחודי שלו
* המפתח הפרטי שלו

איבוד קובץ זה מאפשרת לכל אחד להתחזות למשתמש שלו הקובץ הזה שייך.

נוסיף עוד כי מצד התוכנה אין אפשרות לזהות אם מדובר באשרה המקורית של המשתמש או באשרה גנובה.

יתר על כן, שינוי תוכן האשרה ימנע ממשתמש להמשיך להשתמש במערכת בצורה תקינה עד למחיקת האשרה מחדש. פירוט נוסף בהמשך ניתוח האיומים.

* **המפתח הסימטרי של המשתמש**

המפתח הסימטרי משמש את הלקוח לתקשורת מוצפנת מקצה לקצה עם לקוח אחר.

דליפת המפתח תאפשר הצצה לתוכן התקשורת המוצפנת.

שינוי ערך המפתח הסימטרי תדרוש תיאום מפתח מחדש.

* **פרטיות המשתמש**

כאמור מדובר בפרוייקט התומך בהצפנת מידע מקצה לקצה ושימוש במזהים למשתמשים.

נרצה להגן על פרטיות המשתמשים.

* **השרת**

פעולת השרת השותפת הכרחית להמשך ריצת הפרוייקט.

הנחות מוצא

* טווח בטוח בין המשתמש עצמו לבין המכונה המריצה את תוכנת הלקוח.

כלומר, אין צד מתווך בין המשתמש למכונה, והמגע ישיר ולא על גבי טווח תקשורת לא מאובטח.

* המכונה מריצה את תוכנת הלקוח על גבי מערכת הפעלה מודרנית עם שיקולים אבטחתיים, בפרט חלוקת זיכרון בין תהליכים ותהליכונים במערכת והגנה על גישה למערכת הקבצים.

השימוש במערכת הפעלה עם דגש אבטחתי (למשל Windows, Ubuntu) אינה מונעת איומים, אך מורידה את סבירותם לצורך המשך הניתוח.

* כתובת השרת הנמצאת בקובץ server.info מתואמת מול השרת האמיתי.
* למשתמש גישה ישירה למערכת הקבצים על המכונה שלו לצורך שמירת וקריאת האשרה.
* לתוקף יש גישה למבנה ההודעות והפרוטוקול (למשל על ידי השגת תוכנת הלקוח וניתוחה).
* לתוקף יש גישה והיכרות מלאה עם תוכנת צד הלקוח, אך לא עם תוכנת צד השרת.
* התוקף ישתדל לשמור על חסויותו, משמע לא יבצע פעולות רועשות כמו לחסום את התקשורת מצד הלקוח.

ניתוח איומים בסבירות גבוהה

1. **פרוטוקול**

איומים הדורשים האזנה לתקשורת ללא כלים מפותחים נוספים.

* 1. **מידע גלוי**

כאמור התקשורת מול השרת אינה מוצפנת, כלומר המזהים חשופים לכל אחד.

חשיפת המזהה מאפשרת לבצע התאמה בין מזהים למשתמשים פוטנציאליים (שכן השם באשרה אינו בהכרח מצביע על האדם ישירות), למשל על פי שעות הפעילות, תדירותן, ומעקב אחרי הנמענים של ההודעות המוצפנות.

מעבר לעולם התקשורת, דליפת מידע זו מאפשרת הצמדה חד חד ערכית בין אדם משתמש לבין השימוש שלו בפרוייקט ובכך פוגעת בפרטיותו.

פתרון:

הצפנת התקשורת בין הלקוח לשרת בצמד מפתחות שונה על מנת לאפשר טווח תקשורת בין השרת ללקוח, ולהסתיר גם את המזהים.

* 1. **התקפת Anti-Replay**

לכל הודעה בודדת אין מזהה ייחודי.

על כן, תוקף אשר מאזין ברשת יכול להסניף הודעה ועל אף שאינו בהכרח מסוגל להצפינה הוא מסוגל לשמור אותה, ולשלוח אותה שוב ושוב, והצד המקבל יאלץ לאשר את ההודעות שכן הן תקינות ברמת הפרוטוקול ודורשות טיפול.

מכיוון שאין הלקוח מאזין לבקשות באופן קבוע, אינו מהווה מטרה עיקרית לתקיפה זו.

אך השרת מעצם היותו ה slave בתקשורת מאזין באופן קבוע ומצפה לטפל בהודעות.

הנספת הודעות תקינות מאפשרת להעמיס על השרת מעבר למצופה בפרוייקט עד לרמה שהעומס יורגש ברמת המשתמשים שכן לשרת יקח זמן להתפנות לטיפול בבקשות אמיתיות ולא העתקות.

תיאור תקיפה לצד השרת:

* + הסנפת בקשת שליפת לקוחות בשלב מתקדם במצב בו טבלת המשתמשים עמוסה בשרת.
  + שליחת הודעה זו כמות גדולה של פעמים.
  + השרת יאלץ לטפל בבקשה מעמיסה פעמים רבות ולא יתפנה לטיפול בהודעות של לקוחות.

תקיפה לצד הלקוח:

* + שליחת בקשת "הודעה מוצפנת למשתמש" פעמים רבות לשרת, ובכך למלא את תיבת הדואר של המשתמש ולהקשות בפענוח כל ההודעות.

פתרון:

הוספת מספור וחתימה להודעות הפרוטוקול ושימוש בחלון זז.

באופן זה, הודעות ישנות והודעות מטופלות לא יטופלו שוב על ידי השרת, ושינוי תוכן הודעה יזוהה בעת וידוא החתימה של ההודעה בשרת.

* 1. **DoS**

במסגרת הרישום, נשלחת בקשת הרשמה עם שם המשתמש ומפתח פומבי.

לפי הגדרת הפרוטוקול, על השרת להחזיר הודעת שגיאה במקרה בו מתקבל שם משתמש אשר כבר קיים בבסיס הנתונים.

תוקף ברשת יכול לאתר בקשת הרשמה, להבין את המבנה שלה, ולשלוח לשרת בקשת הרשמה יחידה לכל שם משתמש אפשרי בעזרת מאגרי שמות משתמשים נפוצים אשר נגישים ברשת (לפחות שמות אנושיים עד 100 תווים).

מהלך זה ימלא את העמודות בצד השרת עבור כל השמות האלה, ותמנע ממשתמשים חדשים להירשם בעזרת שם פשוט (או כל שם, אם לתוקף היה מספיק זמן לשלוח בקשות אך זה פחות סביר). בכך השרת לא יוכל לאפשר הרשמה לאף משתמש חדש.

פתרונות:

* + להמנע מיוחודיות השם ולהיעזר במזהה משתמש.
  + להצפין את התקשורת בין הלקוח לשרת.
  + למספר את ההודעות ולחתום עליהן בדומה לסעיף הקודם.

1. **צופן**

איומים הדורשים האזנה לתקשורת, הבנה קריפטולוגית עמוקה ויכולות עיבוד חזקות.

* 1. **הצפנה כפולה**

נבחין כי אף מפתח צופן אינו מוחלף.

אם כן, על ידי הצפנת מידע אשר ניתן לחיזוי או שימוש בהצפנות רבות, תוקף יכול לשבור את הצופן על ידי brute force מושכל.

* + **מפתח סימטרי**

אורך חייו של המפתח הסימטרי אומנם קצר יותר יחסית למפתח הפרטי, אך ניתן לצפות להצפנת תוכן אשר ניתן לחיזוי, למשל תחילת הודעה ב"שלום" או ההדר של מבנה הקובץ.

ההצפנה הכפולה מקלה על שבירת הצופן ומעמידה את המפתח בסכנה.

פתרון:

החלפת המפתחות לאחר פרק זמן קבוע או לאחר מספר שימושים אשר קבוע מראש.

פעולה זו תלווה בעדכון השרת בחלופת המפתח הפומבי, ועדכון הלקוח השני בעדכון המפתח הסימטרי או תיאום מפתח מחדש.

ניתוח איומי המשך – לאחר השמשת חולשות מערכת

1. **תוקף בצד לקוח**

איומים האפשריים עקב גישה של תוקף למכונה של הלקוח.

1. **שינוי האשרה**

במידה ותוקף הצליח לגשת למערכת הקבצים של המשתמש ולשנות את האשרה (למשל על ידי DirtyCow או תקיפות Privilege Escalation אחרות), הלקוח לא יוכל להשתמש בה שנית לצורך התחברות ויהיה עליו למחוק את האשרה התקולה ולהירשם מחדש.

פתרונות:

* + שימוש במערכת הפעלה מרודדת ולצמצם את משטח התקיפה במערכת עצמה.
  + במידה ואפשר, לנעול את הזיכרון בו נשמרת האשרה במנגנון חומרתי לקריאה בלבד.

1. **הדלפת המפתח הסימטרי והפרטי**

על פי ההגדרה, כלל המפתחות נטענים אצל המשתמש בזמן ריצה ועל כן שמורים בערימה.

תוקף מתוחכם יכול למפות את כלל הזכרונות הממופים בגדלים הנדרשים למפתחות ובכך לאפשר לעצמו גישה ישירה למפתחות אותם מקצה המשתמש בזמן הריצה.

בכך יכול התוקף להמשיך ולהסניף את התעבורה אשר תהא גלויה לחלוטין.

פתרונות:

* + שימוש במערכת הפעלה מרודדת ולאפשר הקצאות דינמיות אך ורק לתוכנת הלקוח.
  + שינוי הארכיטקטורה לאפשרת שמירת המפתחות בצורה סטטית.

1. **שינוי server.info להתחזות או MITM**

במידה ולתוקף ישנה אפשרות לשנות קבצי מערכת, אין דבר המונע ממנו לנתב את כלל התעבורה דרך מכונה אישית, בה הוא מסוגל להחזיר כל תשובה שירצה.

דוגמה למימוש התקפה עם התחזות:

* + רקע: קיימת ידיעה שאיש A אמור לשלוח מידע חשוב וסודי לאיש B בעזרת MessageU והם לא מתקשרים במציאות.
  + התוקף משנה את פרטי Server.info במכונה של איש A, ומנתב את הבקשות של A למחשבו האישי של התוקף.
  + התוקף מזייף רשומה של איש B כתשובה לבקשת רשימת המשתמשים, ומוסיף מפתח פומבי שהוא יצר.
  + איש A יתאם מפתח סימטרי מול התוקף במחשבה שהוא מתאם מול B וישלח לו את המידע הסודי.

דוגמה למימוש התקפה עם MITM:

* + רקע: קיימת ידיעה שאיש A אמור לשלוח מידע חשוב וסודי לאיש B בעזרת MessageU והם מתקשרים במציאות.
  + התוקף משנה את פרטי Server.info במכונה של איש A, ומנתב את הבקשות של A למחשבו האישי של התוקף.
  + התוקף משנה את פרטי Server.info במכונה של איש B, ומנתב את הבקשות של B למחשבו האישי של התוקף.
  + כאשר A מבקש את רשימת המשתמשים, התוקף מזייף רשומה של איש B כתשובה ומוסיף מפתח פומבי שהוא יצר.
  + כאשר B מבקש את רשימת המשתמשים, התוקף מזייף רשומה של איש A כתשובה ומוסיף מפתח פומבי שהוא יצר.
  + איש A ואיש B יתאמו מפתח סימטרי מול התוקף במחשבה שהם מתאם אחד מול השני.
  + איש A ישלח את המידע הסודי המפתח הסימטרי שלו מול התוקף, והתוקף יפענח את המידע עם המפתח הסימטרי ויקרא את התוכן.
  + התוקף יצפין את המידע עם המפתח הסימטרי שתיאם מול B וישלח לו.

פתרון:

* + טעינת התוכנה עם המפתח הפומבי של השרת.
  + בקשת אשרה מהשרת אשר ניתן לוודא רק בעזרת מפתח וידוא אשר טעון מראש בתוכנה.

1. **תוקף בצד השרת**

איומים מצד תוקף בצד השרת מסוכנים מאוד וקשה להתגבר עליהם, ולכן נניח כי הם פחות סבירים וישנה בקרה מעמיקה על הפעילות על חומרת השרת.

ניתוח איומים בסבירות נמוכה

1. **חולשות specter, meltdown, וגישה ישירה אחרות**

השמשת חולשות מהמשפחה הזו תאפשר לתוקף בעל גישה ישירה למכונה לגשת לאשרת המשתמש ואף למפתח הסימטרי במידה והספיק ליצור אחד.

בצד השרת התקיפה תאפשר לתוקף לגשת לכלל מסד הנתונים המקומי אותו השרת מתפעל.

חולשות אלה דורשות מהתוקף גישה ישירה למכונות ולכן פחות סבירות.

פתרון:

להגן על החומרה, ולהרחיקה מזרים ואנשים שלא אוהבים אותך במיוחד.

1. **חולשות מימוש אשר האפיון אינו מתייחס אליהן**
2. **RCE via pickle deserialization**

בצד השרת נדרשים לבצע Deserialization בפייתון של מידע הנשלח מהלקוח.

לא הוגדר מראש מהו מנגנון הדי-סרלוז הרצוי לפרוייקט.

על כן, ניתן בקלות דעת לבחור ב Pickle, אך ברמת המימוש של המודול נבחין כי ישנו פתח לתקיפה.

מתוך התיעוד של המודול:

Whenever an object is pickled, the \_\_reduce\_\_ method defined by it gets called. This method returns either a string, which may represent the name of a Python global, or a tuple describing how to reconstruct this object when unpickling.

אם כן, דיו לתוקף להמיר לבתים בעזרת Pickle מחלקה זדונית הממשת את \_\_reduce\_\_ להרצת קוד בצד השרת.

פתרון:

לא להשתמש ב Pickle.

1. **הגדרת פרק ה Timeout לשמירה על משאבים**

לא הוגדר זמן Timeout בהמתנה לבקשה מלקוח.

מצד אחד, הפרוייקט לא שואף לספק שירותים ללקוחות מרוחקים ולא לוקח אחראיות על בעיות קליטה או תקשורת אפשריות, אבל על פי הגדרת הפרוייקט מתאפשרת הגדרת Timeout ארוכה אשר פותחת חלון לשלל תקיפות משאבים כגון Slow Loris.

פתרון:

להשתמש בזמן המוקצב הדיפולטי, או להגדיר פרק זמן קצר המאפשר תקשורת מהירה.