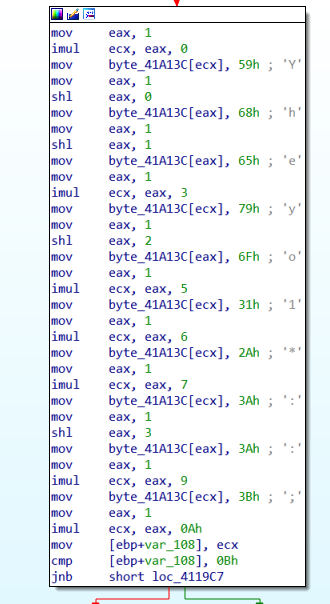
מבחן, מגיש אלעד פישר, 213924624:

1. אם נסתכל בתוך הTLS אפשר לראות שהוא לא מאוד תמים כמו שאני הייתי חושב בהתחלה, אחרי הכל IDA מעיר לנו על דבר די זדוני שהוא עושה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיבתוך הTLS יש קוד זדוני שקורא לפונקציה שבודקת אם הדיבאגר פעיל, לצערי הTLS חשוב מידי כדי שאני אוכל להתעלם ממנו ולכן אני חייב להפעיל אותו כמו שצריך כי הוא גם חשוב להמשך הרצת התוכנית(למעשה כבר עכשיו אני יודע שהוא מאתחל משתנים מסויימים בזיכרון:

כמו שאפשר לראות בקטן הוא מאתחל מחרוזת, אבל היא מחרוזת חשובה משום שעוד נשתמש בה להדפיס את הודעת ההצלחה...(למעשה נחזור לזה אחכ כי זה משומה לא מאותחל, אבל הפירוט יהיה על הסדר ויותר לקראת סוף המבחן).

הנקודה בקיצור שאי אפשר לדרוס את הקריאה לTLS או לבצע חזרה מיד בלי לשנות מלא קוד שקריטי לנו להמשך ההרצה ואני לא אוהב את זה ולכן הפיתרון הכי פשוט שיש זה לראות את הבדיקה עצמה:

תמונה שמכילה טקסט

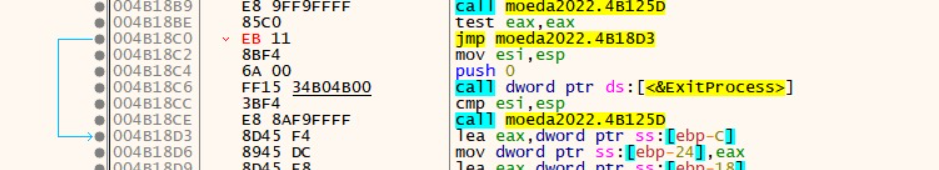
התיאור נוצר באופן אוטומטי

אז בTLS יש לנו קריאה בקטן לבדיקה האם הדיבאגר פעיל, אם הוא כן(הערך בEAX זה 1) אני אמשיך לקוד שיקרא ליציאה מהפרוסס והתוכנית, למעשה זה קצת יותר מבאס המנגנון הזה כי אם זה רק מקריס את התוכנית אפשר בקלות לראות לפי המחסנית קריאות איפה זה קרס ואז לעלות קצת לבדיקה.

בכל מקרה למעשה אני פשוט אשנה בית אחד בזיכרון:

כדי לדלג על הבדיקה אני מתכוון לקפוץ בכל מקרה על הקוד הבעייתי וככה גם אם התוכנית תזהה שיש דיבאגר פעיל היא תדלג על הקוד שאמור לזרוק אותנו מהתוכנית והיא תמשיך כרגיל...

ככה ייראה הקוד אחרי הפצ'פוץ':



ככה זה ייראה מבחינת הקוד עצמו, אפשר לראות את הבית שהשתנה כדי לעשות קפיצה לא מותנה באדום.

ככה זה ייראה בגרף:

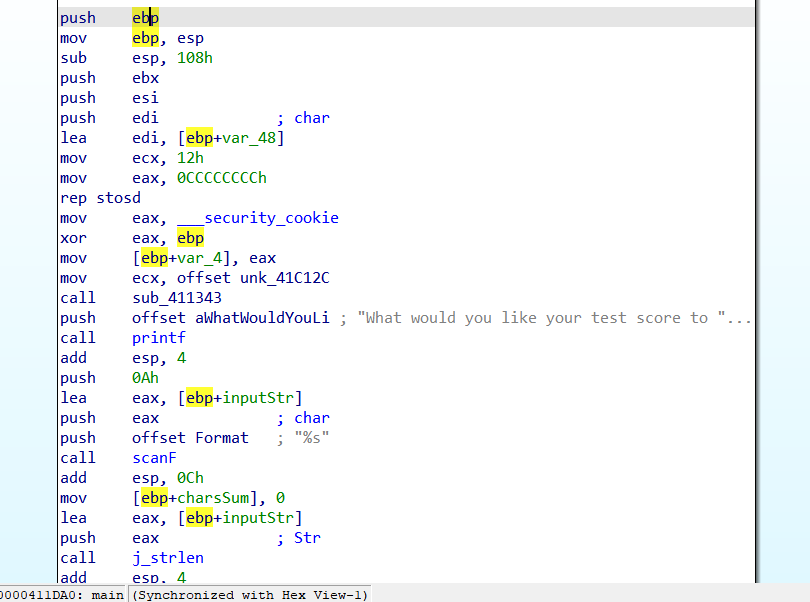
תמונה שמכילה טקסט

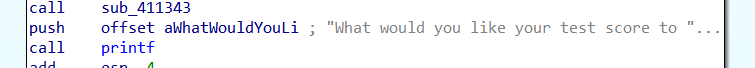
התיאור נוצר באופן אוטומטי כלומר לא משנה מה התוצאה, נקפוץ תמיד להמשך הקוד.

שאלה 2:

תפקידי המשתנים בIDA:

דבר ראשון זה למצוא את הMAIN. כדי לעשות את זה אני פשוט הלכתי תמיד לקריאה האחרונה עד שמצאתי את הבדיקה של הסקיוריטי קוקי ואז הלכתי לקריאה לפני זה שדחפה למחסנית את argV argC וכו' ואז הלכתי לקריאה של המיין, וככה הוא נראה:

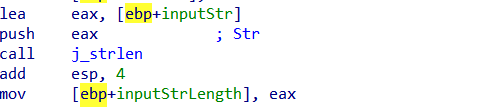
כמובן שאפרות אחרת הייתה לחפש את המחרוזת של What would you… אבל זה דרך אחרת ואם המחרוזת הייתה מוצפנת זה לא היה עובד כזה בקלות, בכל מקרה אני דבר ראשון סימנתי פונקציות קלט\פלט כי ככה קל יותר למצוא את המשתנים ומה התפקיד שלהם:

למשל אחרי הדחיפה יש הדפסה.

ויש בקשת קלט אחכ:

 ולכן ככה אפשר לסמן את המשתנה הזה בתור המחרוזת קלט(מהמשתמש).

לאחר מכן יש קריאה לSTRLEN ולכן יש לנו את המשתנה של אורך הקלט:



ואחר כך יש השוואה עם משתנה שכל פעם עולה באחד עד שהוא מגיע לאורך של הקלט:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי ההשוואה לאורך של הקלט.

אתחול

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיקידום

ובגלל שכל זה קורה בלולאה מבחינת הקפיצות אז ברור שזה האינדקס ואפשר לקרוא לו i אבל לי יותר נוח counter.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטילאחר מכן אפשר לראות שסוכמים את הערכים של הascii לתוך משתנה אחר(שזה ההשמה היחידה אליו חוץ מהאתחול) ולכן נקרא לו charsSum כי הוא עושה סכום לכל הצ'ארים בקלט.

אחכ נשאר לי משתנה אחד שגם עושים לו קידום של 2 בכל פעם בכל סוף בלוק קוד:

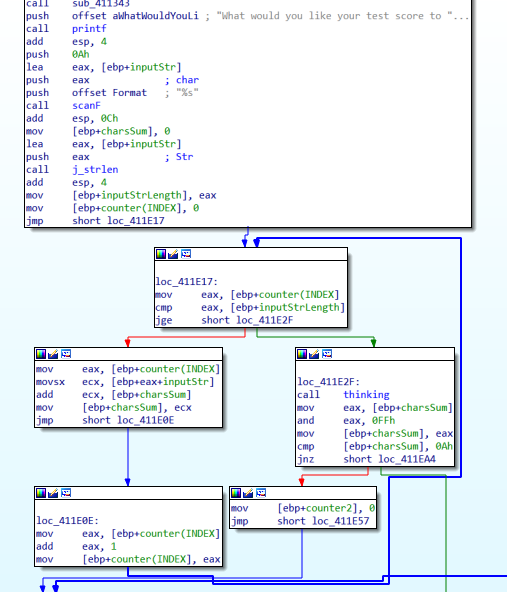
תמונה שמכילה טקסט

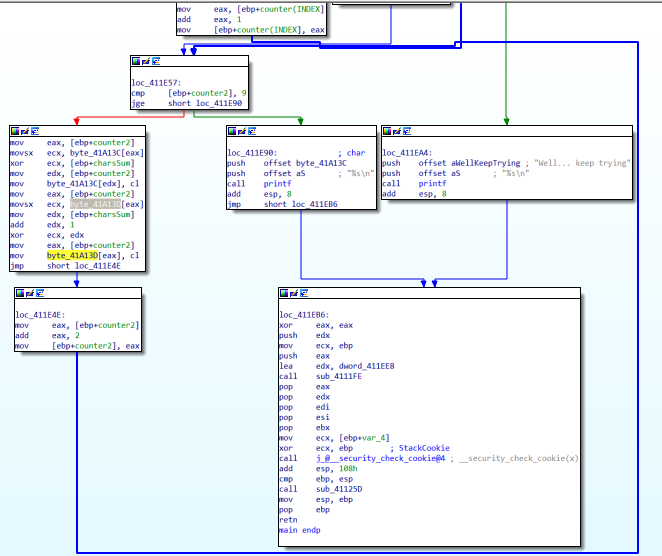
התיאור נוצר באופן אוטומטי

ונראה לפי האתחול ותנאי הסיום שזו לולאה שניה ולכן אני אקרא לזה counter2.תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיהאתחול מימין למעלה והתנאי משמאל למטה.

הקוד של המיין בתמונה אחת:



התמונה רק לקונטקסט, בעיקרון כל הקוד מוסבר למעלה

עכשיו כאשר נתנו שמות להכל אפשר להתחיל לפענח את הקוד:

2ב. בגדול אפשר להבין מהר מאוד לפי הניתוח של המשתנים שיש את המשתנה שסוכם את הצ'ארים ושהוא משמעותי להמשך התוכנית, אז הבדיקה שעושים איתו אחרי הלולאה(בתוך הלולאה אין בדיקה איתו בכלל ולא אחת שהולכת להודעה של הכישלון בפרט) ולכן לפי הבדיקה איתו שהיא:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיהבדיקה עצמה הבלוק הימני למטה. ולכן אפשר לראות שעושים פעולת and לוגית עם FF כלומר לוקחים רק את הבית הכי קטן, ואח"כ עושים השוואה לA0, אחרי ההשוואה אם זה לא שווה הולכים למקום שמדפיס את ההודעת כישלון ואם לא הולכים ללולאה השנייה שנראתה יותר מעניינת וגם איפה שאני רוצה ללכת.

לכן למעשה אני בתור משתמש צריך למצוא מחרוזת שהסכום ערכי הascii של התווים שלה שווים ל ??0aכלומר אני צריך למצוא מחרוזת שהבית הכי קטן שלה יהיה שווה לA ולכן אני אתחיל לחשב מחרוזת שהסכום שלה יהיה 20A אם כי זה שרירותי, גם אפשר 10A או כל דבר דומה.

אני אתחיל בלחסר תווים גדולים יחסית בעזרת מחשבון הקסא כדי לחפש כמה עוד אני צריך לחסר.

אני אקח את התו p כי הוא עם ערך 70 וזה טוב לי. אם אני מחסר 4 פעמים p מהסכום שלי אני רואה שנשאר לי 4A שאני צריך למלא והרי לפי טבלת האסקיי שלי זה בדיוק המספר של התו J ולכן הסיסמה יכולה להיות מחרוזת שמורכבת מ4 פעמים p ופעם אחת J. למעשה היה אפשר לקחת כל מחרוזת שאני רוצה באותה דרך ולחשב את הערך באותה דרך של חיסור של מספרים עד שאני מקבל את התווים שהסכום שלהם שווה למשהו שנגמר עם0A. דרך אחרת אני מניח יהיה להתחיל עם מחרוזת קבועה ולהוסיף תווים כדי למצוא משהו שנגמר עם המספר הנכון אם כי לדעתי זה מסובך יותר(וניסיתי ככה בהתחלה וזה פחות עבד).

הנה ההרצה שיש לי:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיוהרי זה לא הפלט הרצוי וגם זה לא נכון ואני כן עשיתי את הפצ'פוץ' כמו שצריך.

אבל פה מגיע מה שראיתי למעלה שיש את האתחול שכנראה לא התבצע טוב ולכן אם אני שם ברייקפוינט ב32X אני באמת רואה שאני מדלג על זה, וזה משומש בזיכרון, ולכן אני אפצ'פץ' את התנאי להיות nop כי מיד אחרי הקוד הזה מופיע האזור שאני רוצה להיכנס אליו, הסיבה שאני לא עושה קפיצה לא מותנה כי אז זה תמיד ילך לאזור של הלא אתחול.תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

ואחרי התיקון ושמירת השינויים באמת הסיסמה עובדת:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי.

3.א.

עכשיו יש לכתוב את הקוד בהנדסה לאחור:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

זה הקוד שמבצע הכל חוץ מהפענוח של ההצפנה, למעשה יש פה את הקלט\פלט בהתחלה, סכימה של הכל ואחרי זה בדיקה של הסכום אם הבית הכי קטן הוא A ובגלל שהכל פה עובד בצורה עשרונית אז FF מתוגם ל255, וA ל10. אבל אפשר לעשות הכל עם הקסא גם בפייתון, רק שאני לא רוצה להסתבך עם זה.

ועכשיו לפענוח מסר ההצלחה שזה סעיף ב:

דבר ראשון צריך להבין איך ההצפנה עובדת כי לא ממש היה חשוב עד עכשיו כי סך הכל ידעתי שזה שם לפי הקוד ואיפה שצריך ללכת, אבל זה לא שינה כדי למצוא את הסיסמה.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיאבל זה איך שזה נראה הבלוק שעושה הצפנה לכל תו.

דבר ראשון יש את המחרוזת שבמקום E8A13C שאנחנו משתמשים בה אז לפי הקוד והזיכרון היא:

 ולכן אני אשתמש בה בהמשך.

אז למעשה אני לוקח כל תו ועושה לו קסור עם A כי זה הערך של המשתנה שנשאר לי מהסכום של הצ'ארים(בקוד שם זה עשה גם השמה כדי לשנות אותו ל0A)

תמונה שמכילה טקסט, שולחן

התיאור נוצר באופן אוטומטיכאן אני מחליף כל תו עם הקסור של A.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיוזה התמונה אחרי שבעקבות זה התו הראשון השתנה לh מה שכמובן שזה לא הכל, אז צריך לראות את החלק השני:

השלב הבא יהיה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

למעשה כאן אנחנו לוקחים כל תו, ואחרי זה מוסיפים 1(זמנית) לA שזה הסכום של הצ'ארים אחרי השינוי, ועושים עוד קסור עם התו שבמקום של קאונטר 2.

הנקודה המעניינת שזה 2 מחרוזות שונות שכל אחת מהן באורך 5 תווים, למעשה אני עושה קסור שונה לכל תו, ולכן כל תו אי זוגי מקבל קסור של 10 וכל זוגי מקבל 11.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

זה הקוד של הפענוח.

הערה: אני יודע שהקוד כאילו עושה את הבית במקום 41A13D כי זה למעשה מביא לו מקום אי זוגי כי זה בעצם מוסיף 1 ידנית לכתובת הבסיס, אני כתבתי את הקוד לפי בדיקה עם תנאי של אם זה זוגי או לא ומה כל פעם, ככה שמבחינה פונקציונלית זה אותו קוד, רק שהתוכנית יותר מהירה כי היא לא באמת בודקת וקופצת 2 תווים כל פעם וככה משנה כל תו עם ערך אחר. אז זה אותו קוד בפועל..

גרסה סופית של הבדיקה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי(ההבדל שראיתי שלא הדפסתי את ההודעה למקרה של כישלון)