שם: עומר אלכסנדר , ת"ז: 313703415

**ממ"ן 12**

שאלה 1

1. החולשה היא Type conversion – השוואה בין signed לUnsigned.

על מנת לתקוף את המערכת יעל צריכה להיכנס למינוס קטן כך שהcredit שלה יהיה שלילי, ובשורה של הreturn איפה שיש השוואה ביו credit וbound המשתנה credit שלה יומר לunsigned ובעצם יקבל ערך חיובי גדול יותר מ750 ובכך תזכה במתנה למרות שהcredit שלה היה נמוך מ750.

1. קוד מתוקן: משתנה bound יוגדר כ Int ולא כ unsigned.

bool is\_entitled\_for\_promotional\_gift(int ID)

{

int bound = 750;

int credit = get\_credit(ID);

return (credit >= bound);

}

1. מסמך מחקר:

|  |  |
| --- | --- |
| Type conversion | Threat |
| השוואה לא נכונה בין credit וbound | Affected component |
| (lines 3-5) | Module details |
| לקוח | Vulnerability class |
| ע"י השוואה בין int ל unsigned int יש השוואה שגויה במידה ויש ערך credit שלילי. | Description |
| ניתן לגרום לכך שמישהו שיש לו ערך credit שלילי בעצם יהיה זעאי למתנה | Result |
| none | Prerequisites |
| לייצר השוואה נכונה בין שני ערכי int. | Proposed remediation |

שאלה 2

1. החולשה היא bufferoverflow.

ניתן לראות שבhandle\_excape יוצרים את l שיש לו משתנה buffer בגודל 16 אשר נמצא על המחסנית ולאחר מכן רצים בלולאה (שורה 82) ומעתיקים איבר איבר לתוך המחסנית, הפירצה היא שהלולאה רצה לפי גודל הקלט ולכן במידה והקלט גדול מ16 אז נמשיך לדרום איברים על המחסנית.

על מנת לקרוא לפונקציה unreachable, נדרש לייצר Handler וזה מתבצע רק בתוך handle\_escape שנקראת רק בשורה 182. על מנת להגיע לשם צריכים להתקיים שני תנאים:

if(do\_escape && s[0] == '\\')

על מנת שערך do\_escape יהיה true צריך להגיע לשורה 162, בשביל להגיע לשם צריכים להתקיים:

1. if (allow\_options)
2. while (argc > 0 && \*argv[0] == '-')
3. case 'e':

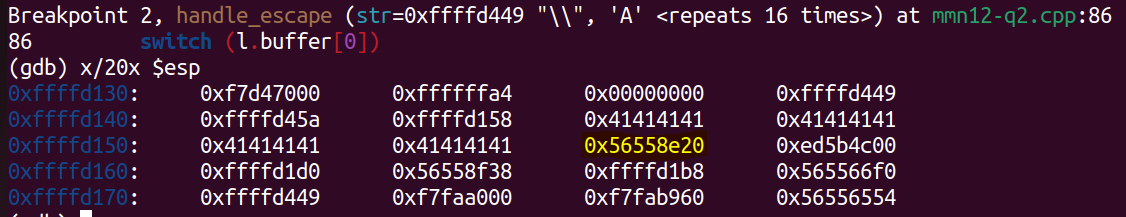
לכן:

1. נבצע export ECHOUTIL\_OPT\_ON=1.
2. הקלט צריך להתחיל ב-e.

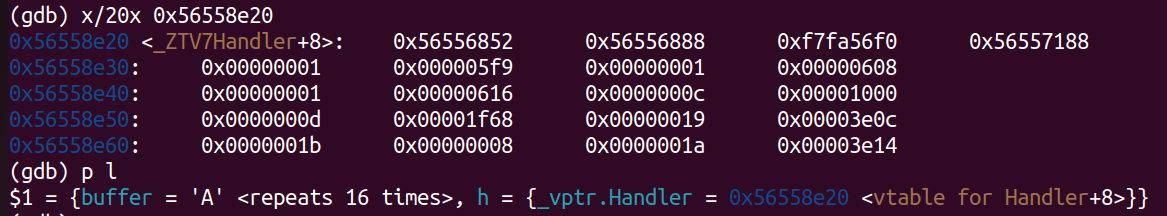
בנוסף בשביל התנאי השני בשורה 182 המשך הקלט צריך להיות \\ ולכן תחילת הקלט צריך להיות:

-e \\

עכשיו נחקור מה בדיוק אנחנו יכולים לדרוס במחסנית: [נמלא את buffer בA\*16]

ניתן לראות שהכתובת המסומנת היא מה שידרס.

נבדוק מה היא:



ניתן לראות שזאת הכתובת לvtable של האובייקט Handler שיצרנו.

נבדוק כעת מה הכתובות שהוא מחזיק:

Graphical user interface

Description automatically generated

גילינו שהוא מחזיק קודם את הכתובת לunreachable ולאחר מכן לhelper.

בשביל להגיע לunreachable אנחנו צריכים להשתמש בקריאה כלשהי של אובייקט Handler. ניתן לראות שהפונקציה היחידה שנקראת היא interpret שנקראת בשורה 89:

switch (l.buffer[0])

{

case 'x':

l.h.interpret(l.buffer);

ולכן המשך הקלט חייב להיות x בשביל לקרוא לפונקציה הזו.

לאחר מכן נראה שבתוך פונקציה זו יש קריאה לפונקציה אחרת של Handler שנקראת helper.[שורה 51]

מה שננסה לעשות זה לגרום לכך שבמקום ללכת לhelper נקפוץ לפונקציה unreachable ובשביל זה נחקור מה בדיוק קורה במחסנית:

Text

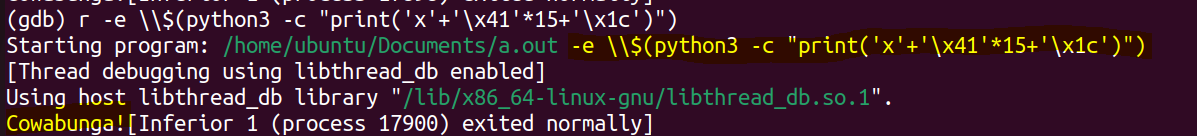
Description automatically generatedאפשר לראות שיש הוספה של 4 לכתובת של vtable [מה שאנחנו יכולים לדרוס] בשביל לקרוא לפונקציה helper.

וכמו שגילינו הפונקציה unreachable נמצאת 4 בתים לפני helper, לכן אנחנו נרצה להקטין את הכתובת שeax מקבל מהמקום שאנחנו יכולים לדרוס ב4 ולשנות את הכתובת מ0x56558e20 ל0x56558e1c

[משום שבהקסה 20-4 = 1c] וכך בעצם לגרום לו לקפוץ לכתובת של unreachable במקום helper.

ולכן הקלט שנרצה להכניס הוא:

-e \\$(python3 -c "print('x'+'\x41'\*15+'\xc1')")



1. התיקון שצריך לעשות בשביל לתקן את הקוד הוא לדאוג שלא תתבצע דריסה של המחסנית ונעשה זאת ע"י הגבלת כמות העתקות לגודל של buffer:

Text

Description automatically generated

1. מסמך מחקר:

|  |  |
| --- | --- |
| bufferoverflow | Threat |
| דריסת מחסנית ע"י הכנסת קלט ארוך יותר מ16 תוים. | Affected component |
| Handle\_escape (lines 82-83) | Module details |
| Handler | Vulnerability class |
| ע"י הכנסת קלט גדול מ16 תווים ניתן לדרוס במחסנית את את ניהול הכתובות של אובייקט Handler ע"י שינוי הכתובת vtable שלו. ובכך בעצם | Description |
| ניתן לגרום לקוד לבצע משהו אחר שלא התכוונו. כמו למשל שהדגמנו להגיע לפונקציה unreachable. | Result |
| none | Prerequisites |
| להגביל את העתקת הקלט לbuffer לגודל שמוקצה לו [כמו שהדגמתי למעלה]. | Proposed remediation |