1. שאלה 1 - Buffer overflow א'
   1. קבלת קלט מהמשתמש וכתיבתו ללוג בצירוף התאריך
   2. מחסנית:

|  |
| --- |
| logMessage 128B |
| B 64B |
| dateLength 4B |
| Date 4B |
| Canary 4B |
| Old EBP 4B |
| Ret address 4B |

* 1. על מנת לדרוס את הreturn address ללא שינוי של ה-canary נרצה שהכתיבה לזכרון בשורה 8 תצביע על ה-return address או במילים אחרות נרצה ש:
  2. ולכן, נרצה שהערך של datelength יהיה 208 אז בקליטה של b נכניס את הערך 0xDEADBEEF נוסיף 60B של זבל על מנת להגיע ל- datelength ונכניס את הערך 208. וכך בכתיבה של b לזיכרון בשורה 8   
     יעבור הערך 0xDEADBEEF ל- return address.

1. שאלה 2 - Buffer Overflow ב'
   1. מימוש נאיבי - מעבר על כל האפשרויות לcanary בגודל 4B- אפשרויות, שניה אחת עבור כל אפשרות ולכן זמן ממוצע של שניות.
   2. במקום לנסות ולנחש את כל ה-canary ביחד, ננסה לנחש כל byte בנפרד ונשתמש באתחול המערכת כאינדיקטור. כלומר, נבצע 4 סדרות ניחושים עבור כל byte- אפשרויות כך שבמקרה הגרוע ייקח לנו שניות או 17 דקות לנחש הכל ובמקרה הממוצע יקח לנו 512 שניות.
   3. נתחיל במציאת הבייט הראשון על ידי הזנת איזה שהוא ערך שוב ושוב עד שהבייט הראשון של הcanary יהיה שווה לבייט שהזנו (לכל היותר איטרציות) משם נמשיך כמו בסעיף הקודם רק שבכל איטרציה נגדיל את הבייט שמצאנו ב-1. זמן הריצה לא השתנה.
2. שאלה 3
3. שאלה 4
4. שאלה 5
   1. מעל המחסנית נמצא מרחב ה-kernel space, תפקידו הוא ניהול הזיכרון והתהליכים של מערכת ההפעלה, ניתן לעבור בין המרחבים ע"י system calls.
   2. ניתן לבצע התקפות buffer overflow, ניקח לדוגמא את הקוד משאלה 1:

char\* date = getDateString();

int dateLength = strlen(date);

char b[64];

ESP

|  |
| --- |
|  |
| logMessage 128B |
| B 64B |
| dateLength 4B |
| Date 4B |
| Canary 4B |
| Old EBP 4B |
| Ret address 4B |

gets(b);

char logMessage[128];

strcpy(logMessage, date);

strcpy(logMessage + dateLength, b);

writeLog(logMessage);

כפי שראינו בשאלה 1 ניתן לדרוס את הreturn address ללא פגיעה בcanary ולכן, ניתן להכניס לכתובת החזרה כתובת גדולה שתגיע לתוך ה-kernel space.

1. שאלה 6
   1. ניתן לראות את ההבדלים העיקריים בטבלה הבאה:

|  |  |
| --- | --- |
| TCP | UDP |
| דורש חיבור רציף בין שני הצדדים | אין צורך בחיבור רציף, שולח מידע לכל מי שמאזין. |
| מבטיח את הגעת המידע ליעד | אינו מבטיח. |
| שולח מחדש חבילות שאבדו | אינו שולח מחדש |
| בודק את תקינות המידע במגוון שיטות, מחכה לאישור שהמידע הגיע | בודק שגיאות רק באמצעות checksum |

* 1. מקים השיחה שולח syn לצד השני על מנת לידע אותו שהוא רוצה לפתוח תקשורת,

הצד השני מחזיר syn-ack על מנת להודיע כי הוא מסכים לפתיחת התקשורת,

הצד הראשון מחזיר ack וכך יודעים שהתקשורת תקינה לשני הצדדים.

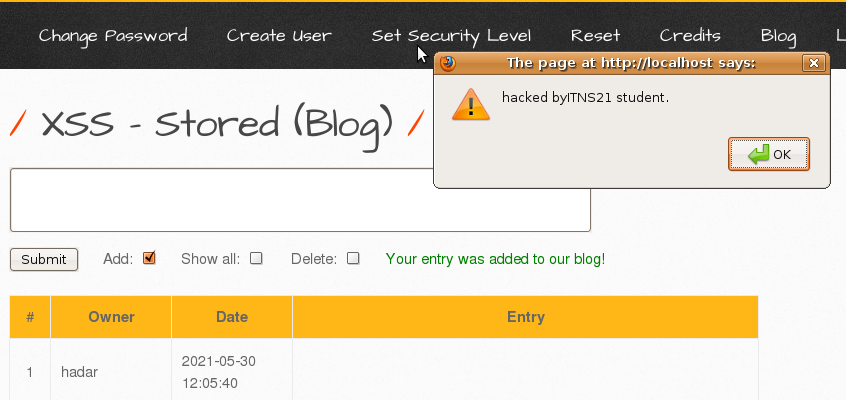
* 1. Syn-flood זוהי התקפת dos שבה התוקף שולח הרבה הודעות syn מכתובות ip מזויפות, השרת צריך לפתוח סוקט עבור כל בקשה כזאת ומחכה ל-ack וכך התוקף מנצל את כל משאבי השרת.
     1. דרך ראשונה: שימוש ב-proxy:  
        ע"י שימוש בפרוקסי בעל קיבולת חיבורים גדולה יותר מהשרת, נעביר לשרת רק את החיבורים אשר אומתו וכך נשאיר את החיבורים החצי פתוחים בפרוקסי ולא ננצל את משאבי השרת.
     2. דרך שניה: הגבלת החיבורים החצי פתוחים

נגדיר איזה שהוא קבוע k שיהווה את מספר החיבורים החצי פתוחים המקסימליים וכך בכל פעם שמספר החיבורים החצי פתוחים יעבור את k נתחיל לסגור חיבורים חצי פתוחים לפי fifo.

* 1. התקפת syn-Ack זאת התקפה בה התוקף מנסה לתקוף לקוח של השרת. התוקף שולח בקשות syn רבות עם הכתובת ip של הקורבן והשרת שולח לקורבן בקשות syn-ack בהתאם ובעצם מעמיס על המשאבים של הקורבן.

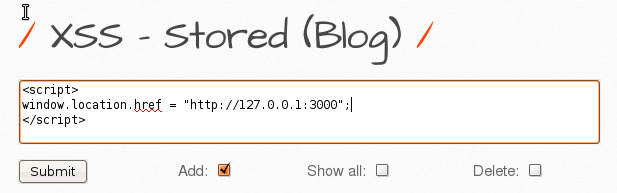
1. שאלה 7
2. שאלה 8
3. שאלה 9
   1. XSS-Stored הינה מתקפה בה התוקף מזריק סקריפט זדוני אל אתר אינטרנט כך שהסקריפט נשאר שם באופן קבוע. בצורה זאת הסקריפט מופעל בכל פעם שגולשי האתר נכנסים אליו.
   2. נכניס את הסקריפט הבא:





ההודעה תחזור גם כשנלחץ OK והדרך היחידה לצאת מהלולאה היא להרוג את התהליך של הדפדפן.

* 1. התקנו bee-box ולכן אנו רוצים להעביר את הנתקף לאתר לבחירתנו ולקבל חיווי על המעבר.  
     לכן על התוקף לבנות שרת שאליו יאזין.

השתמשנו בסקריפט הבא על מנת להעביר את הנתקף לשרת לבחירתינו:

בנוסף הקמנו שרת המדפיס את כתובת האייפי של הנתקף ומעביר אותו לאתר לבחירתינו:



ובכניסה לבלוג התקבל בצד השרת:

