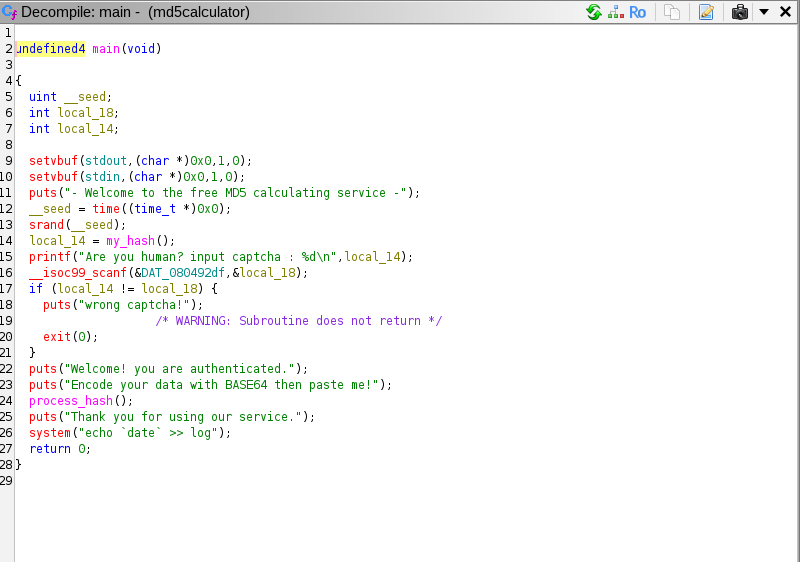
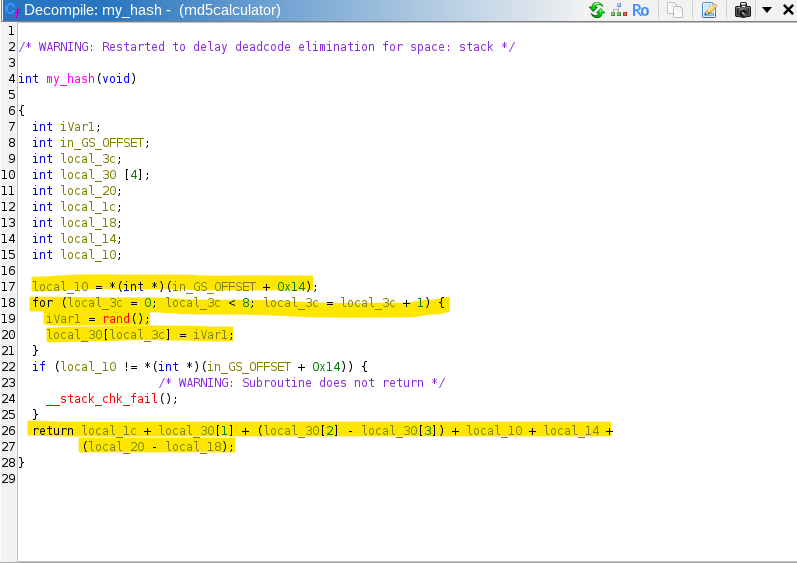
**Md5calculator - Writeup**

**ניכנס לתיקייה של האתגר, ונראה מה יש לנו שם.  
אפשר לראות שיש לנו בינארי בשם *md5calculator*, וגם את קובץ ה־ README שמסביר בקצרה את המשימה. ננסה להריץ את התוכנית כדי לראות מה היא עושה: היא מבקשת מאיתנו קלט, מציגה איזשהו *captcha*, ואז מאפשרת לנו לשלוח נתונים בBase64**

**ניתוח ראשוני עם Ghidra**

**"נפתח עכשיו את הקובץ ב־Ghidra ונבצע דה־קומפילציה.  
ב־main נוכל לראות שהקריאה הראשונה היא ל־my\_hash(), שזו פונקציה שמחזירה את הערך של ה־captcha שאנחנו מקבלים על המסך.**

**  
נעמיק ל־my\_hash() ונראה:**

****

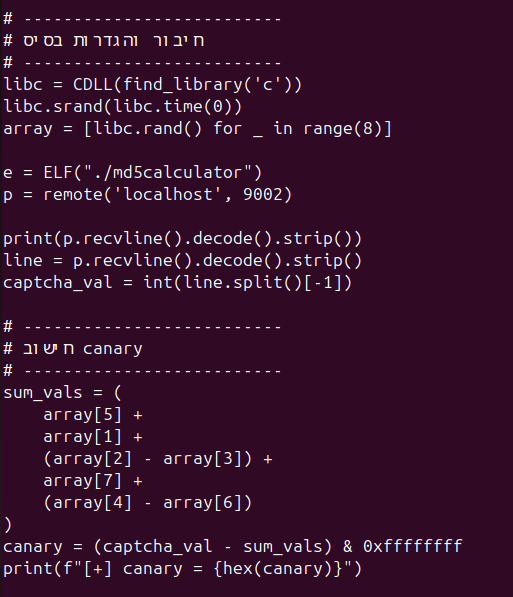
* **היא קוראת את ה־canary מה־GS segment,**
* **מבצעת srand(time(0)) כדי לאתחל את הגנרטור האקראי,**
* **ואז רצה לולאה של 8 קריאות rand(), ושומרת אותן במערך מקומי.  
  בסוף היא מחזירה חיבור של חלק מהמספרים האלה ביחד עם ה־canary.  
  זה בעצם המפתח שלנו – כי אם נדע לשחזר את אותם ערכי rand מקומית, נוכל לחשב אחורה את ה־canary מתוך המספר שהודפס לנו כ־captcha."**

**חישוב ה־canary:**

**הCAPTCHA הוא בעצם חיבור של כמה ערכי הRAND והCANARY, 8 סה"כ.   
בפועל, נריץ אצלנו את rand() שמונה פעמים עם אותו seed שהוא ה־time של השרת, הSEED זה בעצם ערך בסיס כזה להגרלה של המספרים אחרי זה. ונשחזר בדיוק את אותם מספרים. אחרי זה נוכל לחשב את ה־canary**

**A screenshot of a computer program

AI-generated content may be incorrect.**

****

**פרצת ה־Buffer Overflow:**

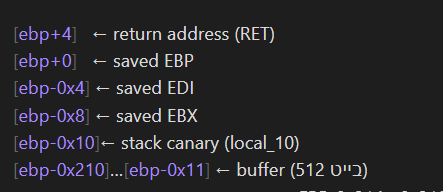
**נעבור עכשיו לprocess\_hash() הפונקציה שמטפלת בקלט שלנו.**

**היא מכניה BUFFER מקומי בגודל 512 בתים, ואז קוראת לFGETS כדי למלא BUFFER גלובלי בשם g\_buf ב 1024 תווים מהקלט שלנו.**

**אחר כך היא קוראת לפונקציית, Base64Decode שמעבירה את הפלט של הBase64 לbuffer המקומי של 512 בתים.**

**ופה בדיוק הבעיה, אם הפלט המפוענח שלנו גדול מ512 בתים, אנחנו נדרוס את המחסנית: קודם את הcanary ואחריו את הEBP ולבסוף את כתובת הRET.**

**מבנה המחסנית**

****

**כלומר, הפיילוד שלנו צריך להיות:**

* **512 בתים של זבל**
* **הCANARY שחישבנו**
* **PADDING**
* **כתובת PLT של SYSTEM, שיחליף את הכתובת RET**
* **fake ret**
* **מצביע ל/bin/sh – כתובת שתבציע למחרוזת /bin/sh**
* **\*\*"/bin/sh\x00"\*\* → המחרוזת עצמה בסוף הפיילוד.**

**הטריק הוא כזה: כשאנחנו שולחים את הפיילוד בBae64 ואחריו מצרפים plain text -/bin/sh**

**הBase64Decoder מתייחס רק לחלק המקודד, ואילו הטקסט הנוסף נשאר בg\_buf בזיכרון כמו שהוא.**

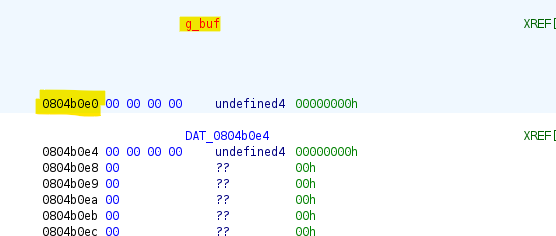
**בגלל שהg\_buf נמצא בכתובת ידועה אנחנו יכולים לחשב את המרחק שבו נדחוף את המחרוזת /bin/sh בסף, לשים אותו ככתובת המחרוזת לפונקציה system ובצע buffer overflow עם הערכים הנכונה.**

**והחלק הכי חשוב הוא כשאנחנו שולחים את הPAYLOAD אנחנו נשלח אותו כBASE64 מוצפן, הוא בעצם יקח את הPAYLOAD שלנו, יהפוך אותו מ540 בייטים ל720 כי זה הנוסחא של BASE64 ENCODE, והפונקציה Base64Decode תיקח רק את הטקסט המוצפן ולכן אנחנו נדע שבמיקום ה720 ימצא המחרוזת /bin/sh**

**ככה זה נראה בסקריפט:**

**הכתובת של המחרוזת היא 0x2d0 מהכתובת של g\_buf**

A black background with white text

AI-generated content may be incorrect.****

