**מסמך עבור לקוח – תיעוד באמצעות LBR**

תיעוד האפליקציה הלינוקסית על בסיס LBR יחולק לשני שלבים מרכזיים ובכל שלב יהיו תתי שלבים שמיועדים לתת מעטפת כוללת לאותו שלב.

1. **הגדרת תצורות - configuration**
2. על המשתמש למסור את ההגדרות שעל פיהן הוא רוצה שהמידע ייקלט בLBR :

* User mode - ניטור הסתעפויות במרחב המשתמש בלבד.
* Kernel mode - ניטור הסתעפויות במרחב הקרנל.
* Call stack mode - שמירת ערימת הקריאות (אם נתמך בחומרה).
* Filter - סינון לפי קריטריונים מוגדרים (למשל טווח כתובות).
* Number of captured branches (depth) – כמה הסתעפויות לשמור (לפי מה שcpu מאפשר).

1. המודול צריך לוודא תקינות של ההגדרות.
2. לבסוף, החלה של ההגדרות דרך פקודות wrmsrl.

עכשיו הLBR מוגדר על ידי הגדרות המשתמש וכעת נותר להריץ את האפליקציה המיועדת.

1. **הרצת האפליקציה**
2. על המודול לבצע בדיקת תקינות של הPATH של האפליקציה שהמשתמש הכניס.
3. ווידוא שבאמת יש LBR .
4. הדלקה של ה LBR.
5. שימוש בfork כדי להריץ את האפליקציה של הלקוח:

FORK ייצור שני תהליכים אב ובן – האב זה המודול שלנו והבן זה האפליקציה החדשה. ברגע שהמודול יבצע waitpid האפליקציה תרוץ, תסיים ותחזיר סטטוס לתהליך האב – המודול שלנו.

1. כיבוי של הLBR.
2. תיעוד והחזרה של ההסתעפויות למשתמש דרך הדפסה לקובץ json.

יש לציין כי התממשקות בין אפליקציית הבקרה למודול תתבצע באמצעות IOCTL.

**חלק ד' – בעיות ופתרונות**

1. **בעיה 1:** במהלך הכתיבה של חלק ג' – נתקעתי בסוג של מחסום כי לא ידעתי איך אני אמור להריץ תוכנה חדש בתוך המודול שלי מבלי לאבד משתנים ואובייקטים שאני צריך על מנת להחזיר למשתמש את מה שהוא צריך.

**פתרון 1:** השתמשתי במנגנון fork() שעליו למדנו בקורס מערכות הפעלה שנועד בידיוק לסנכרון בין שני תהליכים שצריכים לרוץ אחד בתוך השני.

למדתי עליו רק באופן תיאורטי ולא התנסיתי בפועל בשימוש בו ככה שזה היה מאוד מלמד ומאתגר.

1. **בעיה 2:** בתחילת הלמידה שלי על LBR חשבתי שלכל CPU קיים LBR יחיד, ולאחר התעמקות, הבנתי שלכל ליבה במעבד יש LBR שונה. זה בעייתי מכיוון שאם ישנו תהליך שנרצה לנתר, ובאמצע הרצתו הcpu מחליט להחליף ליבה – נאבד מידע יקר.

**פתרון 2:** שימוש ב sched\_setaffinity() שמאפשר הצמדה של תהליך לליבה אחת.

**Tradeoff:** במידה ויש עומס על הcpu או שהאפליקציה עצמה כבדה, יווצר עומס על ליבה אחת וזה יגרום לביצועים הרבה פחות טובים בפעולה עצמה.

1. **בעיה 3:** בין הרצת האפליקציה של הלקוח לבין פעולת הכיבוי של הLBR במודול, ישנם כמה הסתעפויות, זה עלול לגרום לכך שהLBR יתעד גם את ההסתעפויות של המודול ואז המשתמש יקבל פלט לא נכון עבור האפליקציה שלו.

**פתרון 3**: שימוש בדגל user mode – מאפשר ניתור של הסתעפויות אך ורק בצד הUser וכך כל מה שקורה בדרייבר – לא ייקלט בLBR.

**Tradeoff:** אם האפליקציה נמצאת בkernel יהיה קשה יותר להתמודד עם הבעיה.