Sequoia - CVE-2021-33909

אורי דאבוש ויוראי רוט חזן

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-33909

הקדמה:

בתרגיל זה נתאר חולשה במנגנון ה-seq files של לינוקס.

seq files הם דרך שבה אנחנו מגדירים כיצד לפעול כאשר רוצים לפתוח קבצים מיוחדים כמו אלו הקיימים ב-proc/. היחוד של seq files זה שהם נועדו לעבוד עם טקסטים ארוכים יותר ובכך למנוע מהמתכנת לבצע גישה או שינוי לא נכונים - שיכולים לגרום לנזק רב.

./proc/self/mountinfo בשביל לגשת לקובץ seq files-במקרה שלנו, אנחנו משתמשים ב-

החולשה: (הסבר עם קוד)

ממשק קבצי ה-seq של הקרנל של לינוקס מיצר קבצים וירטואלים המכילים קטעים ורשומות. כל רשומה חייבת להתאים לתוך באפר seq_file שמוארך בהתאמה על ידי הכפלת גודלו (שורה 242).

החולשה עצמה לא מגיעה מהכפלת הגודל בגלל ש m->size החולשה עצמה לא מגיעה מהכפלת הגודל בגלל ש m->size החולשה כאן היא בהעברתו ולמערכת יגמר הזיכרון הרבה לפני שההכפלה תגרום ל-overflow. החולשה כאן היא בהעברתו כפרמטר לאחת הפונקציות שמצפה לקבל int ולא size t (כלומר 32 ביטים במקום 64).

```
...
242 m->buf = seq_buf_alloc(m->size <<= 1);
...
246 }
247 }
```

הערה: סימנתי בכחול איפה שהעבירו ונגעו בגודל. (או מה ששומר את הגודל) ניתן לראות בקוד (מעל) שהגודל מועבר גם בשורה 227 שקוראת ל show_mountinfo שניתן לראות בקוד מתחת שקורא בשורה 150 ל seq_dentry לראות בקוד מתחת שקורא בשורה 530 ל prepend.

```
l35 static int show mountinfo(struct seq file *<mark>m</mark>, struct vfsmount *mnt)
12 {
15
```

בשורה 242 הבאפר של ה2GB מוקצה בזיכרון הוירטואלי בעזרת seq_read_iter בפונקציה vmalloc נקרא.

בפונקציה show_mountinfo בשורה 150 הפונקציה show_mountinfo בשורה 2GB.

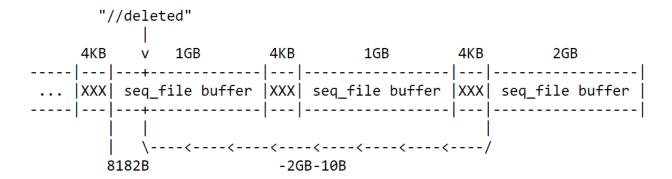
בפונקציה path בשורה בשורה 530 הפונקציה dentry_path בשורה 530 בשורה במשתנה מחג בפונקציה seq_dentry בשורה 530 המקורי הוא 2GB כלומר 2^{32} והוא נשמר במשתנה מסוג path (כלומר אורך הבאפר). אורך ה- 2^{32} אורך ה- 2^{32} הוא נשמר במשתנה מחג int המרתו למשתנה מחג הוא 2^{32} בשיר הקריאה לפונקציה והמרתו למשתנה מחג int הם בטווח 2^{32} 2^{31} בשמירת הערך 2^{32} 2^{31} ולכן שמירת הערך 2^{32} גורמת לחריגה מגבולות ה-int ורק המידע שנמצא בגבולות ה-int נשמר).

בפונקציה dentry_path מוסיפים לבאפר את אורך הבאפר כדי לחשב את הסוף שלו, ובגלל שאורך הבאפר הפך לשלילי נקבל את הכתובת של התא 2GB לפני התחלת הבאפר. לאחר מכן, בשורה מפונקציה prepend נקראת.

בפונקציה prepend בשורה 13 האורך של buflen מוקטן ב 10 והופך ל-10 prepend בפונקציה prepend בשורה 17 האורך של 2GB-10B לבאפר ובשורה 17 הבאפר גם מוקטן ב 10 ומצביע ל 2GB-10B מאחורי ההקצאה של vmalloc לבאפר ובשורה 17 הבאפר גם מוקטן ב 10 ומצביע ל 10 $^{\prime\prime}$ נכתבים למה שאמור להיות 10 תווים לפני סוף הבאפר, אך בפועל נמצאים הרבה לפני כתובת הבאפר.

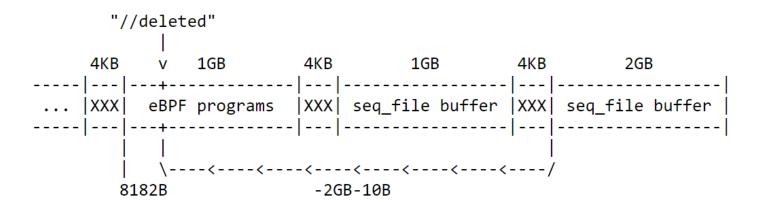
כעת נתאר כיצד ניתן להשמיש את החולשה הזו כדי לקבל הרשאות root:

- 1. יוצרים תיקייה שה-path אליה גדול מג'יגה בייט (זאת על ידי יצירת תיקייה בתוך תיקייה bind-mount מששתמש ללא הרשאות גישה ומוחקים אותה.
 - vmalloc לזיכרון הוירטואלי (זיכרון הקרנל) בעזרת eBPF 2. יוצרים ת'רד שכותב תוכנית נכתבת לזיכרון הקרנל.
- הארוכה, מה path- ומתחילים לקרוא את ה-"/proc/self/mountinfo" הארוכה, מה פותחים את הקובץ ""/deleted" בהיסט של 2GB-10B שיגרום לכתיבת המחרוזת ""/deleted" בהיסט של vmalloc- הקצתה.
- 4. מסדרים את התוכנית כך שהמחרוזת הנ"ל תדרוס פקודה (instruction) של תוכנית ה-PF שלנו (ותאפס את בדיקות האבטחה של ה-PPF שלנו (ותאפס את בדיקות האבטחה של ה-PPF, אנחנו פדי לסדר את התוכנית כך שהמחרוזת תדרוס את הפקודה של תוכנית ה-path, אנחנו דואגים להקצות מקום ע"י שימוש ב-vmalloc לפני שיוצרים את התיקייה עם ה-path הארוך (ב-esplain) המקורי הקצו 2 באפרים של 1GB לפני שהקצו את הבאפר עבור ה-path), כך נוכל לשחרר את הזיכרון המתאים כדי לדאוג שהזיכרון שיוקצה לתוכנית ה-PPF יהיה בדיוק בהיסט המתאים כדי שהמחרוזת "//deleted" תיכתב איפה שאנחנו רוצים שהיא תיכתב. (אמור להיראות כמו בתמונה לאחר שסידרנו את ההקצאות זיכרון)



היה אמור להיכתב. "//deleted" היה אמור להיכתב.

5. לאחר שהקצנו את כל הזיכרון עבור ה-path, אנחנו משחררים את הזיכרון המתאים (בעזרת vfree), כלומר זה שנמצא בהיסט אליו נכתוב בהמשך, ומשחררים את החסימה מהת'רד של תוכנית ה-eBPF כדי שהיא תיכתב לזיכרון בדיוק במקום בו אנחנו רוצים שהיא תיכתב. נחסום את הת'רד שוב אחרי שהתוכנית אושרה ע"י ה-kernel eBPF validator (בנקודה זו יכתב ה"//deleted") ולפני שהיא קומפלה ע"י הקרנל.



- 6. כעת אנחנו רוצים אפשרות לקרוא ולכתוב מידע למקומות שונים בזיכרון הקרנל. כדי לקרוא ולכתוב למקומות שונים בזיכרון הקרנל משתמשים בטכניקות שמנצלות חולשה בתוכניות eBPF (ניתן להרחיב כאן, זוהי חולשה נוספת לכן לא נכנסנו לעומקה).
- 7. בעזרת אפשרות הקריאה אנחנו מאתרים את הבאפר modprobe_path, ובעזרת אפשרות, ובעזרת אפשרות התוכן שלו (שהוא "sbin/modprobe") ל-path ל-path שלנו, שירוץ עם הרשאות root. כעת נקבל שברגע שנחזיר את הת'רד של תוכנית ה-BPF שלנו התוכנית ששמנו את ה-path אליה תרוץ עם הרשאות root.

?כיצד סגרו את החולשה

כדי לפתור את הבעיה ולמנוע את החולשה, הוסיפו בדיקה לפונקציה seq_buf_alloc כדי לפתור את הבעיה ולמנוע את החולשה, שתבדוק שהמקום שאנחנו מקצים לא עובר גבול מסוים. (path שמקצה את המקום עבור ה-hat) שתבדוק שהמקום שאנחנו מקצים לא עובר גבול מסוים. הגבול נתון ע"י הקבוע MAX_RW_COUNT, שלפי הגדרתו לא עולה על wax_rw_country.

```
if (unlikely(size > MAX_RW_COUNT))
return NULL;
```

הגדרתו המקורית היא:

#define MAX RW COUNT (INT MAX & PAGE MASK)

ניתן לראות <mark>כאן</mark> את השינוי הנ"ל בקוד.