גירסה 1.00 – 8.7.2004

Denial of Service via Algorithmic Complexity Attacks

Nir Adar

Email: underwar@hotmail.com

Home Page: http://underwar.livedns.co.il

אנא שלחו תיקונים והערות אל המחבר.

1. תוכן עניינים

1.	תוכן עניינים	2
2.	פתיחה	3
3.	רקע	3
4.	התקפת HASH-TABLES	4
4.1.	ביצוע התקפה	6
4.2.	הגבלות תוכנה המשפיעות על ההתקפה	10
5.	הגנה מהתקפות מבוססות סיבוכיות	11

2. פתיחה

מסמך זה מציג בקצרה סוג חדש יחסית של התקפות המבוססות על ניצול בעיות באלגוריתמים של מבני נתונים, הנמצאים בשימוש באפליקציות גדולות.

ההתקפה מבוססת על העובדה שעבור מבני נתונים רבים, מתקיים כי הזמן הממוצע של פעולתם קטן בהרבה מהזמן הגרוע ביותר שלהם. קלט הנבחר בקפידה עלול לגרום לתוכנות המשתמשות במבני נתונים אלו לעבוד בזמנים הגרועים ביותר.

התקפות מסוג זה שנוסו על מספר שרתים, מראות כי על ידי מודם dialup פשוט, השולח מעט נתונים יחסית, השרתים עלולים להגיע למצב בו הן ינצלו 100% מזמן המעבד, ויפסיקו למעשה לספק שירותים למשתמשים בהן.

מסמך זה מסכם מספר מאמרים שהופיעו באינטרנט בחודשים האחרונים בנושא. עם זאת, הבסיס המתמטי שהוצג במסמכים שמצאתי לא היה מספק, והנושאים נחקרו באופן חלקי.

השאיפה שלי היתה להמשיך ולפתח את הרעיונות שהוצגו באותם מאמרים. אם זאת, סיומו של מסמך זה נדחה כל הזמן, במשך יותר משנה, עקב עומס עבודה ולימודים. לפיכך, החלטתי לפרסם את מה שכבר הספקתי לכתוב, מתוך תקווה שמידע זה יספיק ויעודד את המתעניינים לחפש באינטרנט מקורות נוספות. ייתכן שבעתיד אמשיך לפתח את מסמך זה, אולם אינני רואה מתי בעתיד הנראה לעין.

3. רקע

כאשר אנו דנים במבני נתונים, אנו מנתחים את הסיבוכיות שלהם עבור מספר מקרים בדרך כלל: עבור המקרה האופטימלי, עבור המקרה הממוצע, ועבור המקרה הגרוע ביותר.

למשל, זמן הכנסת n איברים לתוך עץ חיפוש בינארי לא מאוזן לוקחת $O(n\log n)$ זמן בממוצע, אולם אם נכניס למשל רשימה ממוינת לתוך עץ בינארי, העץ יתנהג למעשה כרשימה מקושרת, וזמן ההכנסה יהיה $O(n^2)$.

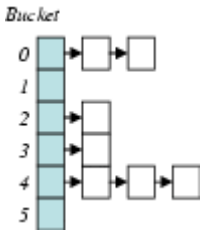
באופן דומה, זמן ההכנסה הממוצע של n איברים לתוך Hash Table הינו $O(n)$, אולם אם יקרה מצב בו כל האיברים ימופו אל אותו התא במערך, הרי שתיווצר רשימה מקושרת וזמן ההכנסה יהיה $O(n^2)$.

קיימים מבנים הפועלים בסיבוכיות טובה גם במקרה הגרוע. עצים מאוזנים, כגון עצי AVL, או עצי 2-3 מאפשרים הכנסת n איברים בזמן $O(n\log n)$ במקרה הגרוע. שימוש בפונקצית ערבול אוניברסאלית יכול ליצור פונקצית ערבול אותה תוקף לא יוכל לנחש.
עם זאת, תוכנות מסחריות רבות משתמשות במבני נתונים פשוטים על מנת לאכסן את הנתונים שלהן. במידה ולתוקף יש אפשרות לקבוע את הקלט של אותם אלגוריתמים, הוא יהיה מסוגל לגרום להם לפעול בסיבוכיות הגרועה ביותר, ועל ידי כך לבצע denial-of-service.

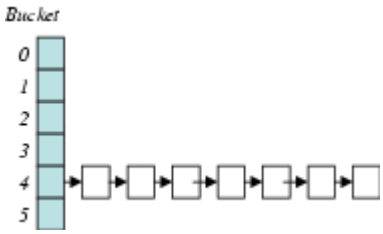
התקפה זו דומה במידת מה להתקפות כגון Buffer Overflows, בכך שעל ידי כמות מידע קטנה יחסית היא עלולה לגרום לשרת להפסיק לתפקד. בניגוד ל-Buffer Overflows התקפה זו עובדת גם על שפות הנחשבות בטוחות, כגון Java, עקב אופייה.

4. התקפת Hash-Tables

טבלת ערבול, Hash Table, הינה אחד ממבני הנתונים הנמצאים בשימוש הנפוץ ביותר - מהדרים משתמשים בהן לשמירת טבלאות הסמלים שלהם, מערכות הפעלה משתמשות ב-Hash Tables למגוון רחב של משימות, כמעט כל שפת תכנות מודרנית מציעה גרסה של Hash Table כחלק מהספרייה הסטנדרטית של השפה.
מתכנתים רבים בוחרים להשתמש בטבלאות ערבול מכיוון שהן מבטיחות זמן גישה ממוצע של $O(1)$, ומתעלמים מהעובדה שזמן הגישה במקרה הגרוע הינו $O(n)$. אחרי הכל - מה הם הסיכויים שטבלת הערבול תיקלע אל המצב הגרוע ביותר?
הבעיה שנציג תודגם ראשית על טבלאות ערבול הממומשות **בשיטת השרשראות**, אולם כפי שנראה היא נכונה גם לגבי טבלאות ערבול הממומשות בשיטות אחרות.



טבלת ערבול במקרה ממוצע



טבלת ערבול במקרה הגרוע

במימוש הנפוץ של טבלאות ערבול, כאשר אנו באים להכניס איבר לטבלה, ראשית מיוצר **מפתח** בן 32 ביטים. לרוב האלגוריתם שמייצר מפתח זה הוא פשוט למדי, למרות שישנם אלגוריתמים מתוחכמים לייצור מפתחות מפוזרים היטב. לאחר שהמפתח מיוצר, אנו מסתכלים בתא ה- $key \bmod N$ של טבלת

הערבול (כאשר N מייצג את גודל טבלת הערבול, ו-key הוא המפתח), ובתא זה אנו שמים את הערך הרצוי.

מתעוררת לעתים בעיה בה שני איברים ממופים אל אותו תא, למרות שהמפתח שלהם שונה (=התנגשות). כדי לפתור בעיה זו, כל תא במערך מחזיק רשימה מקושרת של כל האיברים הממופים אליו. רשימות אלו מכונות לרוב שרשראות.

כאשר מספר האיברים בטבלת הערבול עובר ערך מסוים, מוגדל מספר התאים בטבלת הערבול (בד"כ מוכפל), וכל האיברים מוכנסים אליה מחדש, כאשר הם ממופים אל התא $key \bmod newN$.

זמן הפעולה הגרוע של טבלאות הערבול קורה בשני מצבים:

- כאשר המפתחות המיוצרים עבור האובייקטים השונים המוכנסים אל טבלת הערבול זהים.
- כאשר ערך המפתחות המיוצרים מודולו גודל הטבלה יוצא מספר זהה.

כאשר המפתחות המוכנסים אל טבלת הערבול הם אקראיים, ההסתברות שכל המפתחות יתמפו אל אותו תא הינה נמוכה ביותר - $\left(\frac{1}{b}\right)^{n-1}$, כאשר b הוא מספר התאים בטבלה ו- n הוא מספר הערכים המוכנסים.

אולם, כאשר הקלט נבחר בצורה המתאימה, ההסתברות להתנגשות הופכת לגבוהה ביותר.

כאשר אנו מכניסים ערך לטבלת ערבול, עלינו לבדוק את כל הערכים המצויים בתא אליו הוא ממופה, על מנת להבטיח שאיבר עם מפתח זהה אינו קיים כבר בטבלת הערבול. במקרה כזה, כל פעולת Insert עלולה לקחת $O(n)$.

כדי לבצע התקפה על מבנה הנתונים על התוקף לעמוד במספר דרישות:

- פונקצית הערבול חייבת להיות דטרמיניסטית וידועה לתוקף.
- התוקף צריך להיות מסוגל לספק את כל המידע המשמש את טבלת הערבול ליצירת המפתח שלה.
- התוקף צריך להיות מסוגל לשלוח נתונים למערכת המותקפת במהירות/בכמות מספיקה, על מנת שההתקפה תשפיעה, ותגרום לירידה בביצועים של האפליקציה המותקפת.
- התוקף צריך לדעת כיצד הנתונים הנקלטים על ידי התוכנית מעובדים לפני שהם נכנסים אל טבלת הערבול.

כאשר יש באמצעותו ידע זה, התוקף מסוגל להמציא רצף של אובייקטים אשר ייכנס כולו אל אותו התא בטבלת הערבול.

4.1. ביצוע התקפה

זיהוי תוכנית פוטנציאלית להתקפה

השלב הראשון בבדיקה האם תוכנית מסוימת חשופה להתקפה כזו היא לנתח איפה בתוכנית טבלאות ערבול נמצאות בשימוש, וכן לאתר האם נתונים חיצוניים, ממקור לא בטוח, יכולים להיכנס על הטבלה. שלב זה יכול להיות ארוך ביותר כאשר באים לנתח תוכנית מורכבת. בדיקת התוכנית מתבססת על קריאה וניתוח של הקוד שלה. במידה והתוקף אינו מכיר את הקוד, ייתכן ויעבור זמן רב לפני שתיחשף בעיה.

התנגשות מפתחות מול התנגשות תאים

כפי שצוין, ישנם שני מקרים בהם יתרחשו התנגשויות: במקרה בו המפתחות של אובייקטים שונים הם זהים, ובמקרה שהמפתחות מודולו מספר התאים בטבלת הערבול זהים.
תוקף עלול שלא לדעת בדיוק את מספר התאים בטבלת הערבול. ברוב המימושים של טבלאות ערבול מספר התאים משתנה בהתאם לכמות הנתונים האגורה בטבלה.
עם זאת, בהינתן קוד המקור של התוכנית, התוקף יכול לנחש מספר ערכים אפשריים למספר התאים בה.

לפיכך, ישנם שני סוגי התקפות: התקפה בה לתוקף לא אכפת ממספר התאים בטבלה, והתקפה בה התוקף יודע או מנחש מהו מספר התאים.

הגדרה:

התנגשות מפתחות תוגדר להיות מצב בו בהינתן i קלטים, שנסמנם $k_1, k_2, ..., k_i$, מתקיים כי $Hash(k_1) = Hash(k_2) = ... = Hash(k_i)$.

התנגשות תאים תוגדר להיות מצב בו בהינתן i קלטים, שנסמנם $k_1, k_2, ..., k_i$, המפתחות המיוצרים מן הקלטים הינם שונים זה מזה, אולם $f(k_1) = f(k_2) = ... = f(k_i)$, כאשר f היא הפונקציה הממפה את הקלטים אל התאים בטבלה. (לרוב f מוגדרת בצורה הבאה: $f(k) = Hash(k) \pmod n$, כאשר n הוא מספר התאים בטבלה.

לכאורה נראה כי בחירת ערכים כאלו שתיצור התנגשות מפתחות היא השיטה ההגיונית להתקפה, אולם לא תמיד זו הדרך שתביא להתקפה יעילה.
במערכות רבות טווח הערכים המסוגלים להישלח אל אותו התא הוא מוגבל. אפילו אם נניח שמספר הערכים האפשריים עבור אותו תא הוא גדול יחסית, למשל, 2^{16}, ייתכן שמספר כזה של איברים לא

יספיק כדי לגרום להאטה משמעותית של המערכת. (בפועל מספר כזה הוא יותר ממספיק עבור מערכות שהותקפו).

לכן לרוב תוקף יעדיף לחשב את מספר התאים, וליצור התנגשות תאים, ולא התנגשות מפתחות.

מכיוון שמספר התאים לקוח מתחום קטן בהרבה מתחום המפתחות, יהיה קל יותר למצוא ערכים רבים שימופו אל אותו התא.

לדוגמא: נניח שהערכים המוכנסים אל הטבלה מוגבלים לתחום 1 עד 10. עבור פונקצית hash נתונה ייתכן כי מספר הערכים שיתמפו אל התא 7 הוא קטן. לעומת זאת, מספר הערכים שמודולו גודל הטבלה יתמפו אל תא 7 יכול להיות גדול בהרבה.

אם ישנם מספר ערכים אפשריים לגודל טבלת הערבול, בפני התוקף מספר אפשרויות:

- התוקף יכול לנחש את מספר התאים.
- התוקף יכול לבנות התקפה שתפעל על מספר גדלים שונים אפשריים עבור טבלת הערבול.
- התוקף יכול לשלוח מספר רצפים של נתונים, אשר כל רצף מיועד להתקיף גודל טבלה אפשרי אחר.

תקיפה הפועלת על מספר גדלים אפשריים של טבלאות ערבול לרוב איננה מעשית, מכיוון שערכים הממופים אל אותו התא, עבור מספר גדלים שונים של טבלאות ערבול, הוא לרוב קטן יחסית, ועלול שלא להשפיע על ביצועי מבני הנתונים בצורה משמעותית.

כאשר מספר האפשרויות c לגודל טבלת הערבול קטן יחסית, התוקף יכול להכין מספר רצפי מידע עבור הגדלים השונים האפשריים עבור הטבלה.

נסמן ב- n את מספר הערכים שהתוקף שולח בכל ההתקפה. מכיוון שגודל הטבלה הוא בפועל אחד מבין c ערכים, הרי כי $\left(n\cdot\left(1-\frac{1}{c}\right)\right)$ מהערכים יתפזרו בטבלה בזמן ממוצע של $O(1)$.

$\frac{n}{c}$ האובייקטים הנותרים לעומתם, יכנסו בזמן $O\left(\left(\frac{n}{2}\right)^2\right)$. יותר מכך, אם במהלך הכנסת האיברים גודל הטבלה ישתנה מגודל אחד אל גודל אחר, הטבלה החדשה שתיווצר תסבול עדיין מזמן הכנסה ריבועי.

נציג כעת טענה בה נשתמש בהמשך רעיון ההתקפה.

<u>טענה</u>

תהי $f:\mathbb{N}\to[0,M]$ אפימורפיזם.

יהיו $k_1,k_2\in\mathbb{N}$ שני מספרים המקיימים $f(k_1)=f(k_2)=0$.

נתייחס כעת אל המספרים k_1,k_2 כמספרים המיוצגים על ידי מספר קבוע של ביטים – למשל, 8 ביטים, או 32 ביטים.

נטען: כל קומבינציה אפשרית C של שרשור k_1,k_2 תמופה אף היא אל 0, כלומר: $f(C)=0$.

אם נפרט: $f(k_1k_1)=0, f(k_1k_2)=0, f(k_2k_1)=0, f(k_2k_2)=0$.

<u>דוגמא</u>

ניקח $M=31$, $f(x)=x \bmod 31$, וניקח למשל את המספרים $k_1=31, k_2=62$. מתקיים:

$f(k_1)=f(k_2)=0$. נתייחס למספרים כאל מספרים בני 8 סיביות.

$k_1=31=00011111_2$
$k_2=62=00111110_2$

נביט בביטוי $k_1k_2=0001111100111110_2=7998$. ניתן לראות כי מתקיים ש-$f(7998)=0$.

<u>הוכחה</u>

נסמן ב-k_1,k_2 את צמד המספרים שאנו משרשרים. נניח כי כל מספר הוא בן a סיביות, אזי מתקיים כי:

$$k_1k_2=k_1+\underbrace{2^a}_{c}\cdot k_2=c\cdot k_1+k_2$$

לפי הנתון, k_1,k_2 שייכים לגרעין של האפימורפיזם. כידוע - הגרעין של אפימורפיזם הוא תת חבורה ולכן קומבינציה ליניארית של איברים מהגרעין שייכת גם לגרעין, ולכן הטענה נכונה.

<u>גרימת התנגשות מפתחות בצורה יעילה</u>

הפונקציות המשמשות לערבול ברוב האפליקציות לרוב אינן פונקציות ערבול מסובכות, כגון MD5, אלא פונקציות פשוטות, שנבחרו בעיקר עקב מהירות חישובן.
למרות שהטענה שהצגנו נכונה גם עבור MD5, היא פרקטית עבור פונקציות מהירות.

נניח שעבור פונקצית ערבול $Hash$, וקלטים $k_1, k_2, ..., k_i$ אותם נכנה **מייצרים**, כך ש-
$Hash(k_1) = Hash(k_2) = ... = Hash(k_i) = 0$, אזי, לפי הטענה שראינו קודם, גם
$Hash(k_1 k_2) = 0$, גם $Hash(k_1 k_1 k_1 k_2) = 0$ וכו'. כעת, בעזרת מספר קטן יחסית של מייצרים, נוכל באמצעות שרשור להפיק ערכים רבים שיתמפו אל אותו תא – 0.
מספר הערכים מוגבל רק על ידי גודל הקלט שפונקצית ה-hash מוכנה לקבל.

בבדיקות שנעשו על Perl 5.6.1 מעבר על כל המחרוזות המורכבות מהא"ב האנגלי סיפק 46 מחרוזות שמופו לתא 0. על ידי שרשור של, למשל, 3 מחרוזות מתוך האוסף הנ"ל, אנו מגיעים ל-46^3 ערכים שמתמפים אל אותו תא בטבלת הערבול.

עובדה זו נכונה לא רק עבור Perl – עבור אפליקציות רבות קל לחשב לאן יתמפו הערכים, וכך למצוא מייצרים.

4.2. הגבלות תוכנה המשפיעות על ההתקפה

תוכנות רבות בודקות כמה זיכרון הן מנצלות ומגבילות את המידע המקסימלי היכול להיות שמור בהן. הגבלות כאלו יכולות לגרום להתקפה להכשל, אם הערך המוגבל הוא מספיק קטן כך שלהאטה לא תהיה משמעות.
ההגבלה יכולה להיות מפורשת - למשל Apache Server לוקח כותרות HTTP ושם אותן בתוך מערך דינמי (ומשיג בכך זמן הכנסה של $O\left(n^2\right)$). עם זאת, הגבלת הכותרות בשרת הינה 100 כברירת מחדל. אפילו אם כל ה-100 כותרות יהיו קיימות, ההשהיה שהן יגרמו היא עדיין קטנה מכדי להיות מורגשת.

לעתים ההגבלה של התוכניות איננה מפורשת, אלא נגרמת כתוצאה מאילוצים של גודל טיפוסים (ובכך הגבלה על גודל הקלט ומספר הקלטים האפשריים שאפשר לייצר) וכדומה. אפליקציות אלו אינן חשופות להתקפה, אולם בעתיד, עם השתנות המערכת, יתכן שתתאפשר התקפה במקרים כאלה.

5. הגנה מהתקפות מבוססות סיבוכיות

בהתקפות מסוג התקפות סיבוכיות, אנחנו מניחים שלתוקף יש גישה אל קוד המקור של התוכנית (כך הוא יכול למשל לדעת לאן מתמפים ערכים שונים בטבלת הערבול). מכאן - הגנה על ידי הסתרה היא טקטיקה שאינה רלוונטית בהתקפות אלו.

ישנם מספר פתרונות אפשריים לבעיה:

- ניתן להשתמש בפונקציות אשר לא ניתן לצפות מראש את הקלט הגרוע ביותר שלהן.
- ניתן להחליף את האלגוריתמים הבעייתיים.
- ניתן לבנות בתוכנית מנגנון שיזהה מקרים בהם התוכנית מגיעה למצבי worst-case מבחינת הסיבוכיות, וידאג לטיפול הולם בבעיה.

כאשר מדברים על החלפת האלגוריתמים הבעייתיים, ניתן לדוגמא, להחליף עץ בינארי בעץ מאוזן כלשהו. עבור עצים מאוזנים, מובטח זמן פעולה לוגריתמי. ניתן להשתמש גם במבנים דוגמת treap, אשר הינו עץ בינארי, אשר בהסתברות טובה הוא מאוזן. (הדבר נעשה בצורה הבאה: הערכים בעץ מקיים את תכונת העץ, כך שהעץ הוא עץ בינארי. בנוסף, לכל צומת עם יצירתו מוגדר ערך רנדומלי, כך שהערך המוגרל עבור כל צומת מקיים את תכונת הערימה. בצורה כזו מושג בממוצע עץ מאוזן).

עבור טבלאות ערבול, ניתן להשתמש בפונקצית ערבול אוניברסלית על מנת לפתור את ההתקפה. לחילופין – ניתן גם פה לקחת פונקציה פשוטה יחסית, ולהגריל ערך רנדומלי בו נשתמש בחישובי המפתח. הערך הרנדומלי יוגרל עם התחלת התוכנית, ולכן לא יהיה ידוע לתוקף גם אם הקוד ברשותו.

EOF