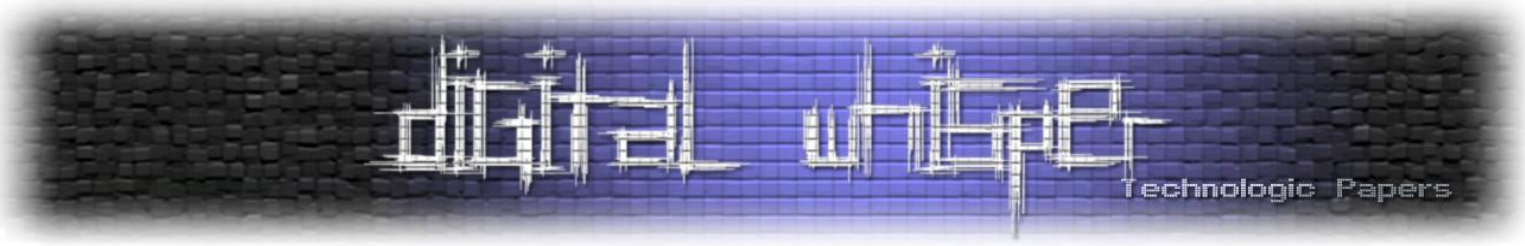


Digital Whisper

גליון 24, ספטמבר 2011

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	ניר אדר, אפיק קסטיאל
כתבים:	אפיק קסטיאל (cp77fk4r), ד"ר אריק פרידמן, עמיחי פרץ קלופשטוק, סשה גולדשטיין, שלמה יונה, לירן בנודיס ואלעד גבאי.

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

דבר העורכים

ברוכים הבאים לגליון ה-24 של Digital Whisper! אני אשתדל להיות כמה שיותר מאופק בדקות הקרובות, אבל... הגליון העשרים וארבע של Digital Whisper בחוץ! (OMG OMG OMG!!!) וכל זה כמובן בזכותכם- הקהילה (כמובן שהייתם זקוקים לדחיפה קלה, אבל מה זה משנה...), במעמד הזה אני מעוניין להגיד תודה לכל מי שעזר לנו, כתב לנו מאמרים, ערך מאמרים של אחרים, הציע הצעות ופעל לטובת המגזין בשנתיים האחרונות:

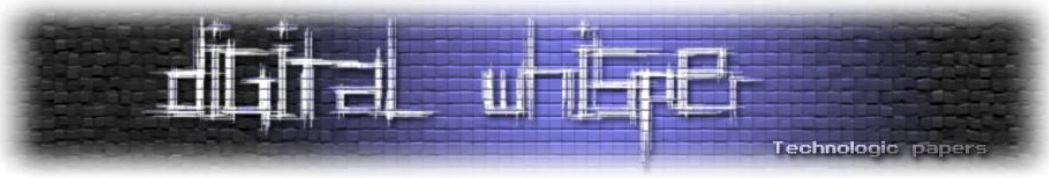
הלל חיימוביץ' (HLL), **Ender**, **הרצל לוי**, **Zerith**, **סולימני יגאל**, **Hyp3rInj3cT10n**, **HMS**, **גדי אלכסנדרוביץ'**, **LaBBa**, **Crossbow**, **עידו קנר**, **צבי קופר**, **עו"ד יונתן קלינגר**, **אלכס רויכמן**, **בנימין כהן**, **ליאור ברש**, **ד"ר אריק פרידמן**, **רועי חורב (AGNil)**, **יוסף רייסין**, **אורי עידן**, **נתנאל שיין**, **אביעד (greenblast)**, **אביב ברזילי (sNiGhT)**, **ניר ולטמן**, **אייל גל**, **שלומי נרקולייב**, **עומר כהן**, **שי רוד (NightRanger)**, **יצחק (Zuk) אברהם**, **TheLeader**, **אוראל ארד**, **ליאור קפלן**, **עמנואל ברונשטיין (emanuel1234)**, **ארז מטולה**, **שלמה יונה**, **דנור כהן (An7i)**, **אורי להב (vbCrLf)**, **אוריאל מלין (Ratinho)**, **קיריל לשצ'יבר**, **אמיתי דן**, **יהודה גרסטל (Do5)**, **ד"ר אור דונקלמן**, **אדיר אברהם**, **עידו נאור (Ce@ser)**, **סשה גולדשטיין**, **בר**, **לירן בנודיס**, **אלעד גבאי**, **סילאן דלאל**, **ליזה גלוך** ו-**SBD**.

בשנתיים האחרונות, פורסמו במסגרת Digital Whisper **129 מאמרים** שנכתבו על ידי **53 כותבים שונים**, אותם מאמרים נפרסו על **1728 (!) עמודים**.

וכמובן, איך לא- תודה רבה לכל מי שנרתם וכתב מאמרים לגליון החודש: תודה רבה ל**ד"ר אריק פרידמן**, תודה רבה ל**שלמה יונה**, תודה רבה ל**עמיחי פרץ קלופשטוק**, תודה רבה ל**סשה גולדשטיין** ותודה רבה ל**לירן בנודיס** ותודה רבה ל**אלעד גבאי**.

קריאה נעימה!

אפיק קסטיאל וניר אדר.



תוכן עניינים

למה מומלץ לבדוק לסוס את השיניים

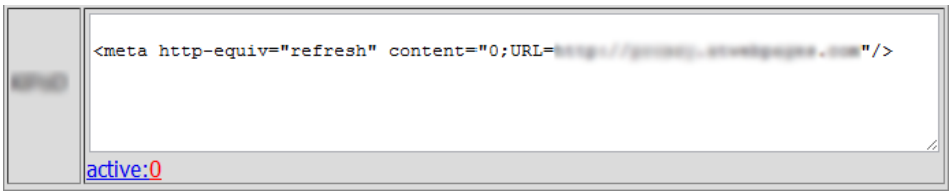
מאת: אפיק קסטיאל (cp77fk4r)

הקדמה

במאמר הזה אני הולך להציג מקרה שקרה בחודש האחרון. מסיבות שאציג בהמשך, אשתדל (עד כמה שאפשר) לתת כמה שפחות נתונים מזהים על המקרה, למרות שידוע לי שבעזרת גוגל תוכלו למצוא בדיוק את המקרה שאני מדבר עליו. ובכל זאת, אני מקווה שהדבר לא יפגע בחווית הקריאה.

הכל התחיל מכניסה שגרתית למערכת אתגרי ההאקינג שאני מנהל: TryThis0ne.com באחד מסופי השבוע. אחד הפיצ'רים במערכת הוא שכל משתמש יכול לפתוח לעצמו עמוד פרופיל, לכתוב בו מידע על עצמו ולעצב אותו כמו שהוא מעוניין. בסופו של דבר על העמוד לעבור אישור של צוות האתר (גם בכדי שנוכל לשלוט בחומר שמתפרסם וגם בכדי שנוכל לוודא שלא מנסים להכניס שום קוד זדוני שיפגע בשאר המשתמשים במערכת).

כאשר נכנסתי למערכת קפצה לי התראה כי יש משתמש שערך את הפרופיל שלו והוא מעוניין שאאשר את השינוי. מכניסה לעמוד העריכה, התברר לי כי השינוי הוא הוספת הקוד הבא:



בתחילה נראה כי מדובר בנסיון של המשתמש לעשות "Deface" טפשי בעזרת ניצול של חולשת Cross Site Scripting (שכמובן לא קיימת) ולאחר בירור עם המשתמש עלתה האפשרות שהחשבון שלו נפרץ.

הלינק מפנה לעמוד ש(בעבר היה)נראה כך:

כן, בטח, בהחלט נשמע אמין.

אישית, בלי להעליב כמובן, אני לא תופס יותר מדי מחברי קבוצת Anonymous, ואף פעם לא אמרתי שניתן להאשים אותם ביתר בגרות, אבל אני חושב שאם שמענו עליהם כל כך הרבה, סביר להניח שהם לא טפשים עד עדי כך, כך שברור לי שאין כל קשר בין העמוד הנ"ל לבינם.

באותו עמוד, לחיצה על התמונה של הלוגו של Anonymous, הובילה לעמוד באתר שיתוף הקבצים "Mediafire.com", לקובץ בינארי השוקל קצת פחות מ-1MB המכיל בשמו, בין היתר, את המחרוזת "Web Hack Pack".

פה חשדתי. ☺

כאן נשאלת השאלה: למה שבן אדם שפרץ לשרת / מערכת מסויימת יפרסם את הכלים שבעזרתם הוא עשה זאת? והתשובה לכך- לא באמת מדובר בכלים לפריצה, אלא מדובר בשיטת הפצה מעניינת של סוס-טרויאני.

הרעיון הוא שבכדי להפיץ את אותו הסוס הבחור היה פורץ לאתרים, מפנה את הגולשים לאותו עמוד, וכך לנסות לגרום להם לחשוב שהאתר נפרץ על ידי אותה קבוצה ולגרום לגולשים הסקרנים לנסות להריץ את אותו הכלי על ידי זה שהם יחשבו שמדובר בכלי האקינג מתקדמים.

מבדיקה ברשת, התברר שלאותו עמוד הפנו מספר רב ביותר של סרטוני Youtube אשר מתיימרים להסביר לגולש כיצד פורצים לשרתים ולמערכות Web אפליקטיביות. המשותף לכלל הסרטים (חוץ מהיוצר שלהם) הוא שבסופם היוצר היה מראה איך הוא מפנה את כלל הגולשים לאותו עמוד וכך בעצם, ע"י פרסום הלינק בהקשר "תמים" היה מפיץ את הכלים הזדוניים שלו.

אז הרצתי את אותה ערכה לפריצת אתרים תחת VM. התוצאות משום מה לא היו מפתיעות במיוחד... מבחינה טכנית אותו קובץ התקין Keylogger על המכונה בשם "Ardamax Keylogger". מדובר ב-Keylogger בשם "Ardamax Keylogger" שמסוגל לתקשר עם היוזם שלו במספר דרכים כגון העלאת המידע שלו לשרת FTP שנקבע מראש או דרך התחברות לתיבת אימייל ושליחת אימיילים לעצמו עם המידע מצורף כ-Attachments.

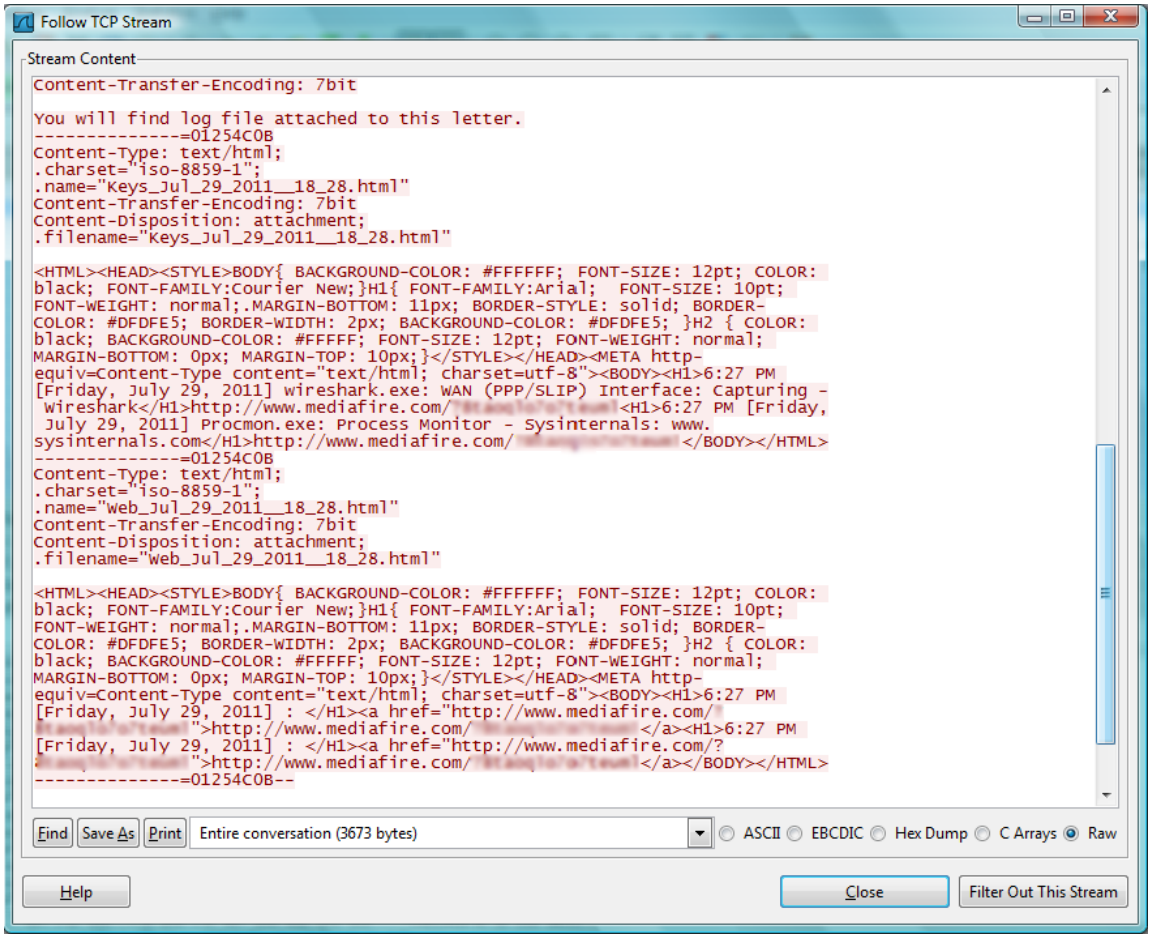
שימו לב ששתי הדרכים שהזכרתי פה מאוד בעייתיות, בשני המקרים פרטי ההזדהות לשרת הביניים נמצאים Hardcoded בקובץ שרץ בשטח. מה שאומר שאם מישהו ימצא את הכלי ויחקור אותו- בקלות רבה תהיה לו גישה לשטח האחסון שעליו מעלה הסוס הטרויאני את המידע אותו הוא אוגר.

דרך נכונה לעשות היא יצירת שני ערוצים שונים שבהם ניתן להתחבר לשרת הביניים: דרך אחת היא רק כתיבה **מבלי הצורך להזדהות** ודרך שניה היא גם כתיבה וגם קריאה, **דרך אשר דורשת הזדהות**. הסוס הטרויאני מתקשר עם שרת הביניים בעזרת הדרך הראשונה. ככה אין צורך שהוא יחזיק בפרטי ההזדהות לאותו מאגר. אגב, למי שעוקב אחר סדרת המאמרים הזאת, יוכל להזכר במאמר על Koobface, שפורסם בגליון ה-14 של Digital Whisper, שבו בדיוק נתקלנו במקרה שתולעת היו רק הרשאות כתיבה ולא הרשאות קריאה.

אז.. למה מומלץ לבדוק לסוס את השיניים?

על פני השטח הרצה של הקובץ לא עושה משהו מועיל, אבל בעזרת Process Monitor ו-Wireshark אפשר לראות בדיוק מה אותו בחור בישל לנו.

בלוגים של Wireshark ניתן לראות את ה-TCP Stream הבא:



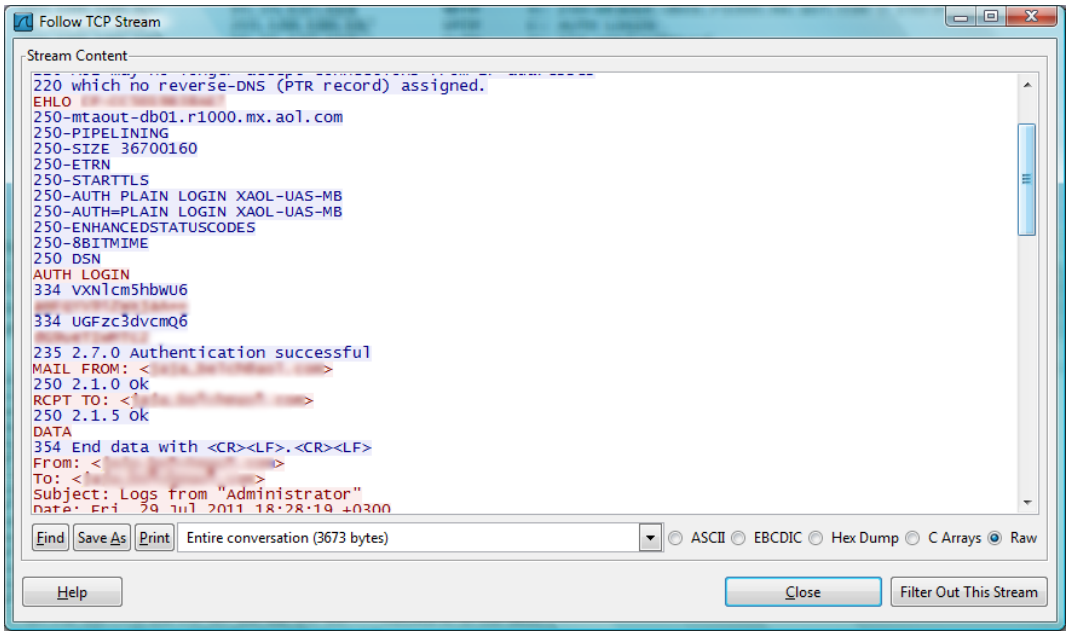
נראה שהקובץ שולח לשרת מסויים מספר קבצי HTML, אחד מהם מכיל את רשימת החלונות שהיו פתוחים בעת ההרצה:

```
<HTML><HEAD><STYLE>BODY{ BACKGROUND-COLOR: #FFFFFF; FONT-SIZE: 12pt; COLOR:
black; FONT-FAMILY:Courier New;}H1{ FONT-FAMILY:Arial;  FONT-SIZE: 10pt;
FONT-WEIGHT: normal;.MARGIN-BOTTOM: 11px; BORDER-STYLE: solid; BORDER-
COLOR: #DFDFE5; BORDER-WIDTH: 2px; BACKGROUND-COLOR: #DFDFE5; }H2 { COLOR:
black; BACKGROUND-COLOR: #FFFFF; FONT-SIZE: 12pt; FONT-WEIGHT: normal;
MARGIN-BOTTOM: 0px; MARGIN-TOP: 10px;}</STYLE></HEAD><META http-
equiv=Content-Type content="text/html; charset=utf-8"><BODY><H1>6:27 PM
[Friday, July 29, 2011] wireshark.exe: WAN (PPP/SLIP) Interface: Capturing -
 Wireshark</H1>http://www.mediafire.com/ <H1>6:27 PM [Friday,
 July 29, 2011] Procmon.exe: Process Monitor - Sysinternals: www.
sysinternals.com</H1>http://www.mediafire.com/ </BODY></HTML>
```

והקובץ השני מכיל את רשימת אתרי האינטרנט שהיו פתוחים בעת ההרצה של הכלי:

```
<HTML><HEAD><STYLE>BODY{ BACKGROUND-COLOR: #FFFFFF; FONT-SIZE: 12pt; COLOR:
black; FONT-FAMILY:Courier New;}H1{ FONT-FAMILY:Arial;  FONT-SIZE: 10pt;
FONT-WEIGHT: normal;.MARGIN-BOTTOM: 11px; BORDER-STYLE: solid; BORDER-
COLOR: #DFDFE5; BORDER-WIDTH: 2px; BACKGROUND-COLOR: #DFDFE5; }H2 { COLOR:
black; BACKGROUND-COLOR: #FFFFF; FONT-SIZE: 12pt; FONT-WEIGHT: normal;
MARGIN-BOTTOM: 0px; MARGIN-TOP: 10px;}</STYLE></HEAD><META http-
equiv=Content-Type content="text/html; charset=utf-8"><BODY><H1>6:27 PM
[Friday, July 29, 2011] : </H1><a href="http://www.mediafire.com/?
">http://www.mediafire.com/</a><H1>6:27 PM
[Friday, July 29, 2011] : </H1><a href="http://www.mediafire.com/?
">http://www.mediafire.com/</a></BODY></HTML>
```

אז לאיפה המידע הזה נשלח? אם נסתכל קצת לפני בלוגים של Wireshark, נוכל לראות את ה-Stream הבא:



אז הבינארי שלנו מבצע הזדהות אל מול שירות SMTP על השרת mtaout-db01-r1000.mx.aol.com, מדובר באחד משרתי המיילים של AOL. אישית אף פעם לא הייתה לי תיבת מייל ב-AOL, אבל שירותי SMTP זה דבר שאני מכיר.

בשניה הראשונה ניתן לחשוב שפרטי ההזדהות מוצפנים או משהו- אבל ברגע השני ניתן ישר להבין שפשוט מדובר ב-"Base64" סטנדרטי.

ניתן לראות בלוגים של Process Monitor בדיוק מה אותו בינארי עושה, אבל עצרתי כאן. הנחתי (מה שהסתבר לאחר מכן כנכון) שאותו קובץ בינארי מעלה את המידע שלו לאותה תיבת אימייל, כך שנבדוק מה עושה אותו הקובץ- פשוט נגש לתיבה ונשם נוכל לראות הכל באופן מסודר...

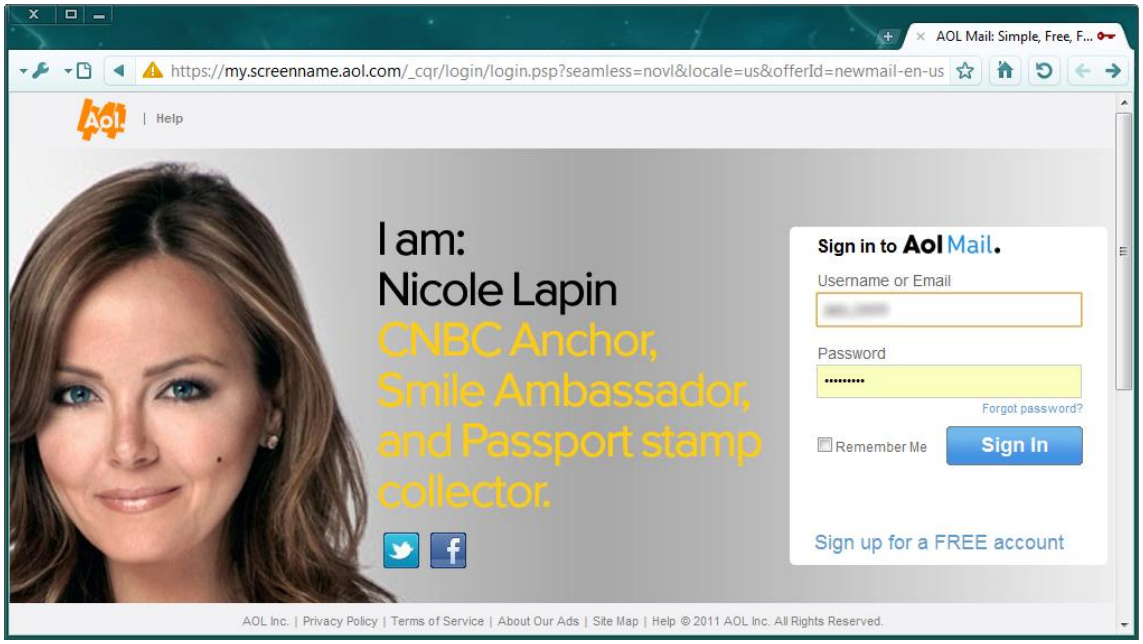
נסתכל שוב על מהלך ההזדהות לתיבה:

```
AUTH LOGIN
334 VXNlcm5hbWU6
[redacted]
334 UGFzc3dvcmQ6
[redacted]
235 2.7.0 Authentication successful
```

המרה פשוטה ל-Base64 ואפשר בקלות לאחזר את פרטי ההזדהות:

```
AUTH LOGIN
334 Username:
[redacted]
334 Password:
[redacted]
235 2.7.0 Authentication successful
```

יש לנו את שם המשתמש ואת הסיסמה. כל מה שאנחנו צריכים בכדי להכנס לתיבה:

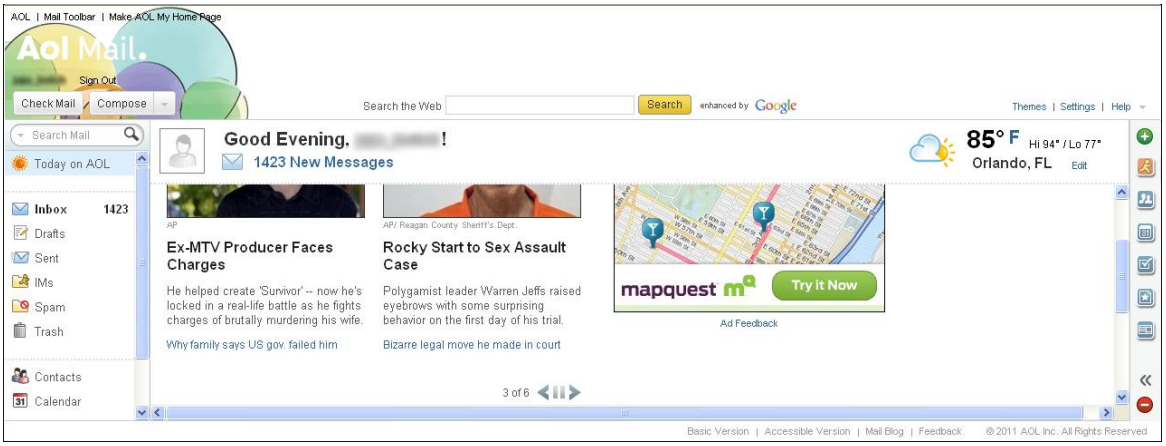


לאחר הזדהות מוצלחת ניתן היה להבין בדיוק מה שולח אותו סוס-טרויאני, חוץ ממה שכבר ראינו, ניתן היה למצוא בתיבה 1423 אימיילים שונים (והמספר עלה כל כמה דקות), שכל אחד מהם הכיל נתונים כגון:

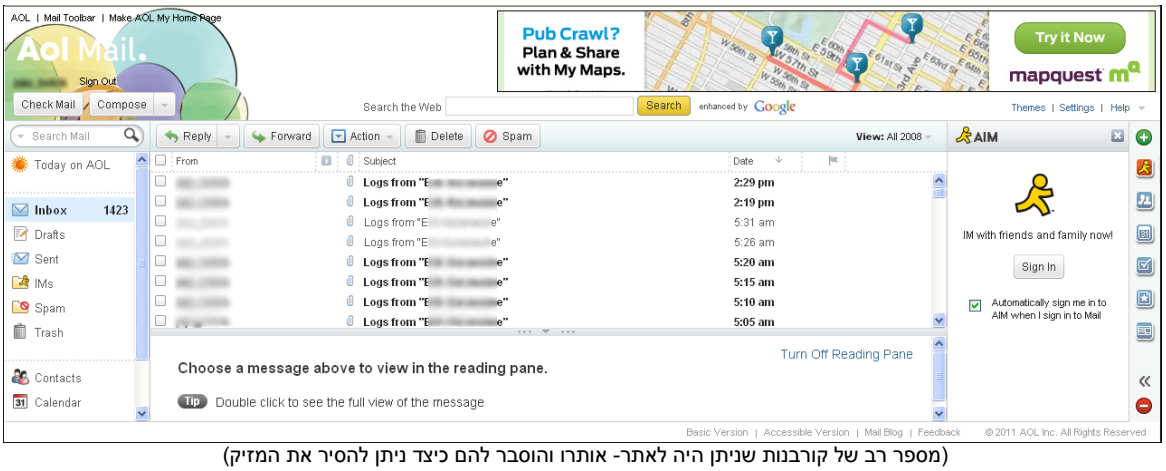
- תצלומי מסך (Print Screens)
- לוג הקשות מקלדת (Key Strokes),
- תהליכים וחלונות פעילים.
- אתרים שבהם ביקר המשתמש וכו'

מספר תמונות להמחשה (פרטים רבים יוצגו באופן מטושטש בכדי לשמור על פרטיותם של הקורבנות).

כאן ניתן לראות את כמות המיילים הנכנסים:

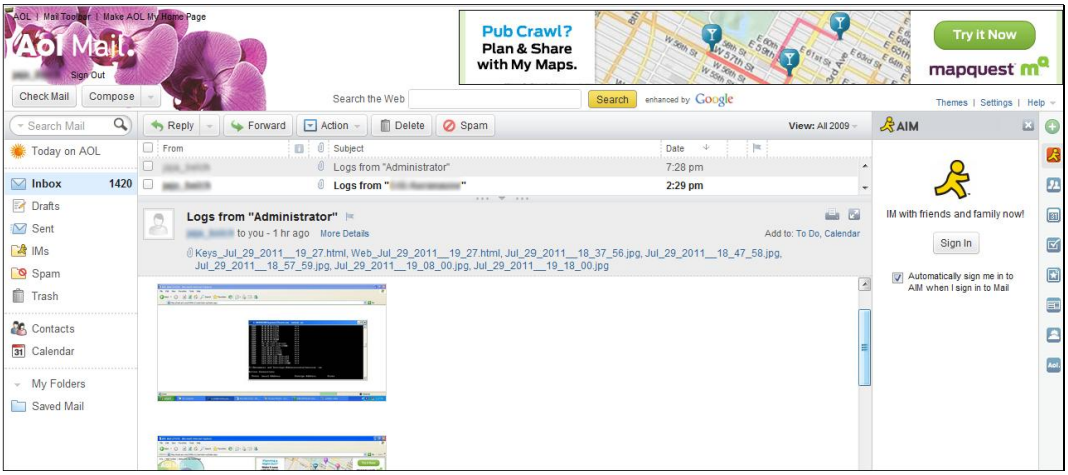


כאן ניתן לראות אימיילים שהגיעו מסוס-טרויאני שנמצא על מחשב בשם "E*** *********e":



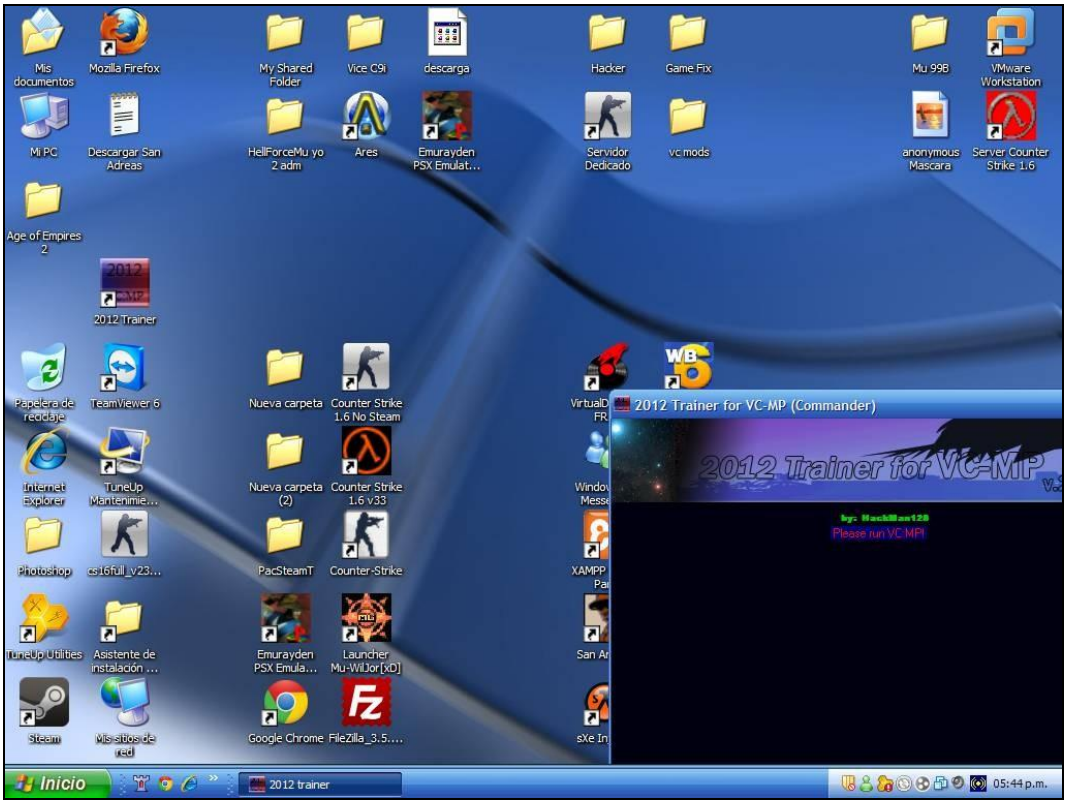
(מספר רב של קורבנות שניתן היה לאתר- אותרו והוסבר להם כיצד ניתן להסיר את המזיק)

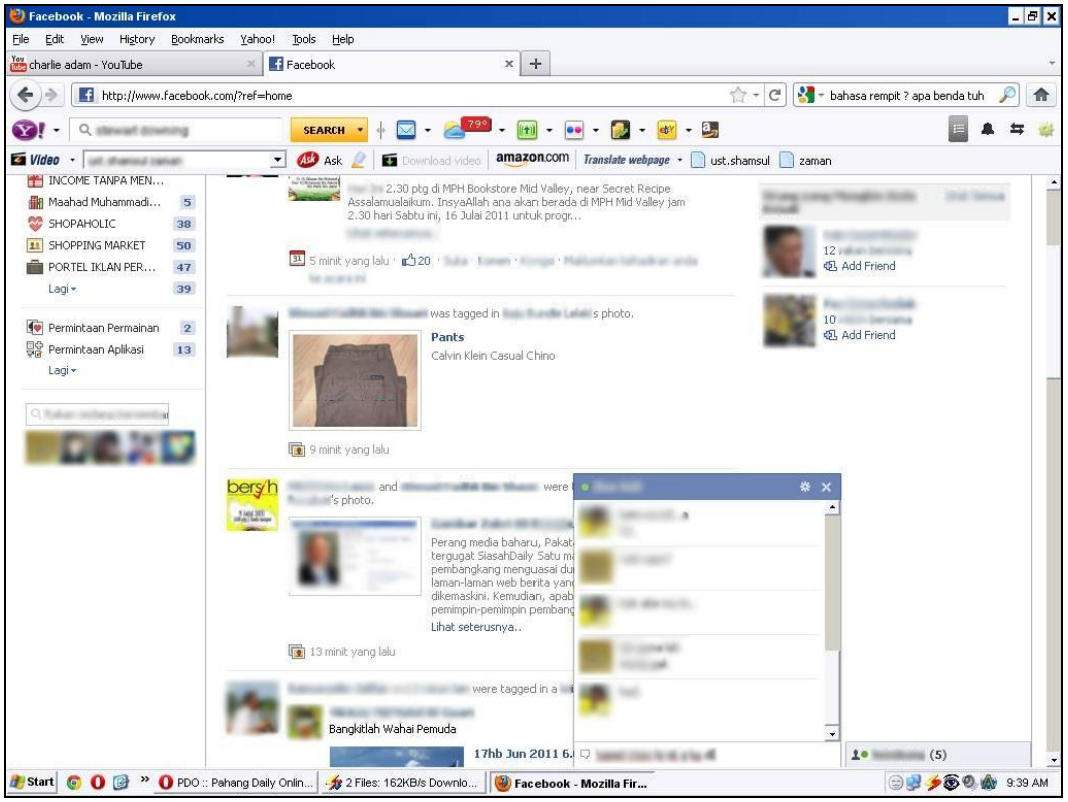
כאן ניתן לראות את תוכן של אימיילים שהגיעו לאחר ההרצה של הכלי על ה-VM שלנו!



(המיילים הספציפים שנשלחו מה-VM שלנו נמחקו מהתיבה)

התמונות הבאות הן מספר תמונות מסך שהגיעו ממחשבים שונים:


Mis documentos
Mozilla Firefox
My Shared Folder
Vice C9i
descarga
Hacker
Game Fix
Mu 99B
VMware Workstation
Mi PC
Descargar San Adreas
HellForceMu yo 2 adm
Ares
Emurayden PSX Emulat...
Servidor Dedicado
vc mods
anonymous Mascara
Server Counter Strike 1.6
Age of Empires 2
2012 Trainer
Papelera de reciclaje
TeamViewer 6
Nueva carpeta
Counter Strike 1.6 No Steam
2012 Trainer for VC-MP (Commander)
Internet Explorer
TuneUp Mantenimie...
Nueva carpeta (2)
Counter Strike 1.6 v33
Photoshop
cs16full_v23...
PacSteamT
Counter-Strike
TuneUp Utilities
Asistente de instalación ...
Emurayden PSX Emula...
Launcher Mu-WilJor[xD]
Steam
Mis sitios de red
Google Chrome
FileZilla_3.5....
Please run VC:MP!
Inicio
2012 trainer
05:44 p.m.


Facebook - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
charlie adam - YouTube
Facebook
http://www.facebook.com/?ref=home
bahasa rempit ? apa benda tuh
SEARCH
Video
Ask
amazon.com
Translate webpage
ust.shamsul
zaman
INCOME TANPA MEN...
Maahad Muhammadi... 5
SHOPAHOLIC 38
SHOPPING MARKET 50
PORTEL IKLAN PER... 47
Lagi 39
Permintaan Permainan 2
Permintaan Aplikasi 13
Lagi
Add Friend
Pants
Calvin Klein Casual Chino
9 minit yang lalu
13 minit yang lalu
Lihat seterusnya..
Bangkitlah Wahai Pemuda
(5)
Start
PDO :: Pahang Daily Onlin...
2 Files: 162KB/s Downlo...
Facebook - Mozilla Fir...
9:39 AM

ועוד ועוד... אותה תיבה הייתה מלאה בדברים בסיגנון הזה. במעבר על המיילים שהגיעו לתיבה ניתן לראות שבעל החשבון סימן בדגל-אדום (אופציה מובנת ב-AOL לסימון מיילים מעניינים/חשובים) כל אימייל שהגיע מהקורבנות שלו שבהם היה מידע עם סיסמאות לחשבונות שונים.

השבתת הטרויאן

אחרי הבנתה של איך הטרויאן המטרה שלי היתה להשבית את הכלי כדי שלא יפגע על משתמשים נוספים.

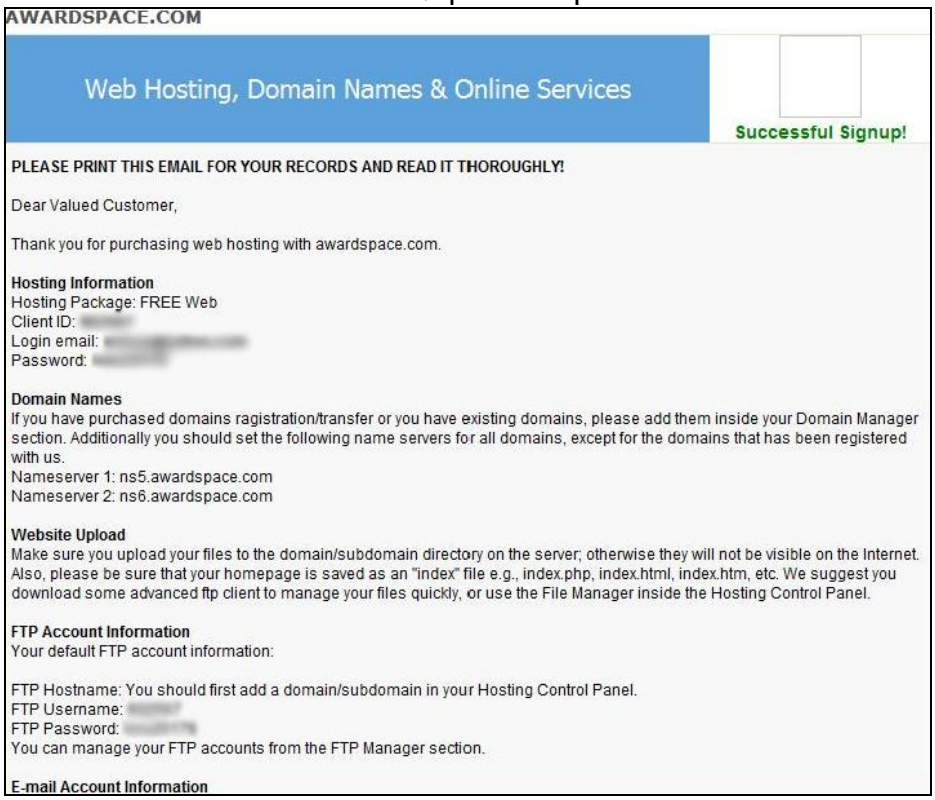
מחיפוש באימיילים הראשונים של התיבה, ניתן היה למצוא מספר נסיונות שאותו בחור ניסה על עצמו. החיפוש עלה יפה ונמצאו מספר ממצאים יפים. **ראשית**- תמונת מסך מכלי שהורץ על המחשב של בעל הכלי:

שנית- אותר לוג הקשות מקלדת ובו נצפה בעל החשבון מתחבר לחשבון המייל המקורי שלו ב-"Yahoo!".

מה עוד אפשר לבקש? ☺ עם גישה לחשבון המקורי של המשתמש הושגה נגישות לחשבון שבו אותו משתמש אחסן את עמוד ה-Deface (שהוצג בתחילת המאמר) והוחלף בעמוד אחר, שונו לו פרטי ההזדהות ואותה ספקית אחסון קיבלה מייל עם המידע על הפעולות הנעשות מאותו חשבון.

בנוסף הושגו פרטי ההזדהות לחשבון ה-Youtube שלו (החשבון שבו היו הסרטונים שהפנו לדף עם הטרויין) וגם שם שונו פרטי ההזדהות. שבסופו של דבר הכלי (לפחות הנוכחי) נוטרל. כמובן שאף אחד לא מבטיח לנו שמחר אותו ילד לא יתעורר בבוקר ויקים הכל מחדש תחת חשבונות חדשים...

האימייל עם פרטי ההזדהות לשטח האחסון שבו אוחסן עמוד ה-Deface:



AWARDSPACE.COM

Web Hosting, Domain Names & Online Services

Successful Signup!

PLEASE PRINT THIS EMAIL FOR YOUR RECORDS AND READ IT THOROUGHLY!

Dear Valued Customer,

Thank you for purchasing web hosting with awardspace.com.

Hosting Information
Hosting Package: FREE Web
Client ID:
Login email:
Password:

Domain Names
If you have purchased domains ragistration/transfer or you have existing domains, please add them inside your Domain Manager section. Additionally you should set the following name servers for all domains, except for the domains that has been registered with us.
Nameserver 1: ns5.awardspace.com
Nameserver 2: ns6.awardspace.com

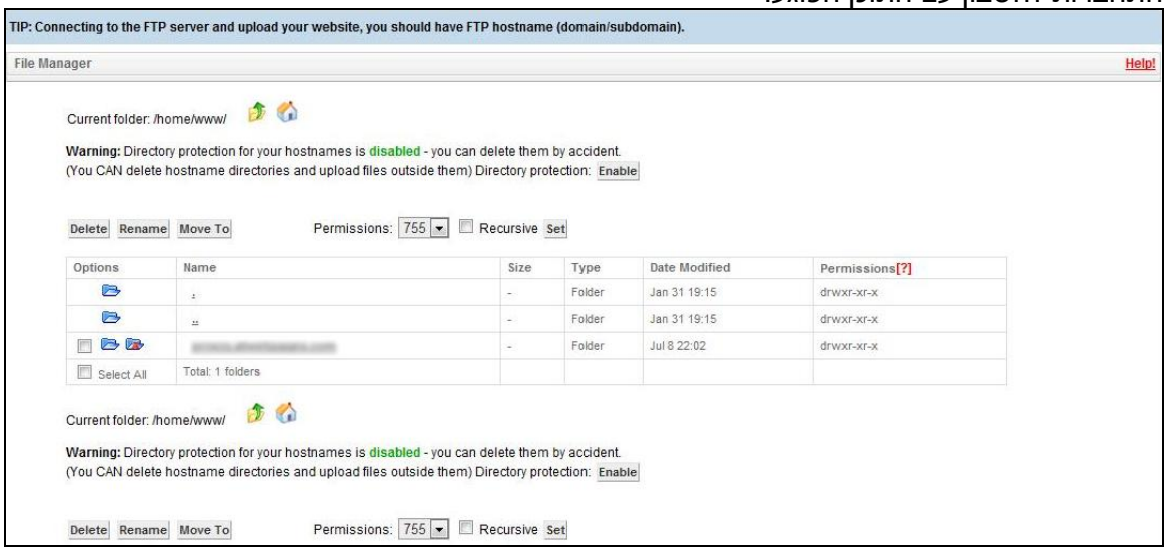
Website Upload
Make sure you upload your files to the domain/subdomain directory on the server; otherwise they will not be visible on the Internet. Also, please be sure that your homepage is saved as an "index" file e.g., index.php, index.html, index.htm, etc. We suggest you download some advanced ftp client to manage your files quickly, or use the File Manager inside the Hosting Control Panel.

FTP Account Information
Your default FTP account information:

FTP Hostname: You should first add a domain/subdomain in your Hosting Control Panel.
FTP Username:
FTP Password:
You can manage your FTP accounts from the FTP Manager section.

E-mail Account Information

התחברות לחשבון עם התוכן הפוגע:



TIP: Connecting to the FTP server and upload your website, you should have FTP hostname (domain/subdomain).

File Manager Help!

Current folder: /home/www/

Warning: Directory protection for your hostnames is disabled - you can delete them by accident.
(You CAN delete hostname directories and upload files outside them) Directory protection: Enable

Delete Rename Move To Permissions: 755 Recursive Set

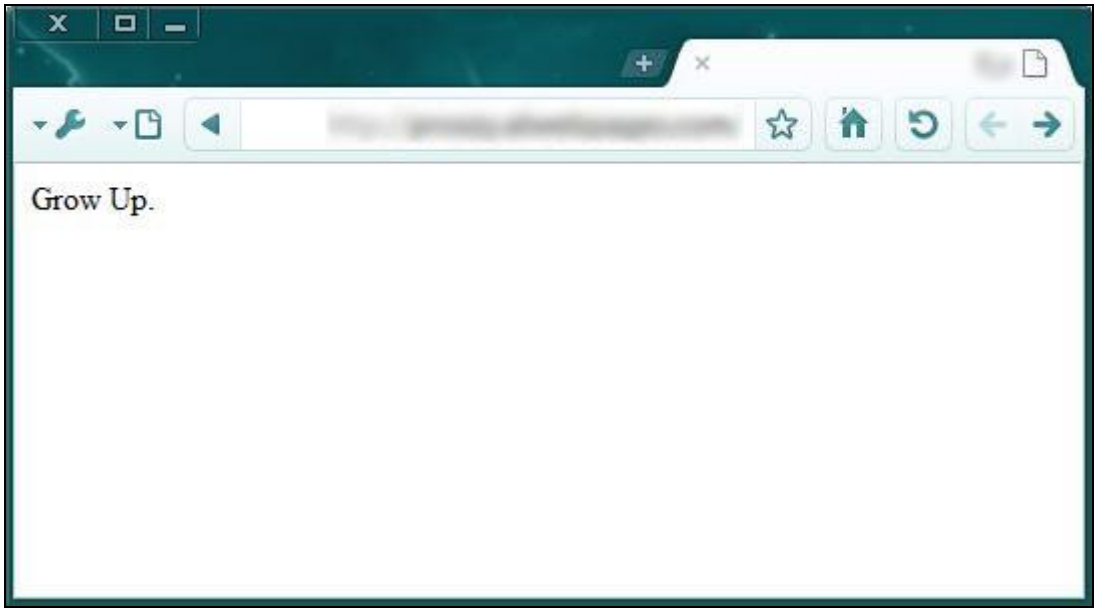
Options	Name	Size	Type	Date Modified	Permissions[?]
	.	-	Folder	Jan 31 19:15	drwxr-xr-x
	..	-	Folder	Jan 31 19:15	drwxr-xr-x
		-	Folder	Jul 8 22:02	drwxr-xr-x
Select All	Total: 1 folders				

Current folder: /home/www/

Warning: Directory protection for your hostnames is disabled - you can delete them by accident.
(You CAN delete hostname directories and upload files outside them) Directory protection: Enable

Delete Rename Move To Permissions: 755 Recursive Set

והתוצאה... עמוד Deface חדש ובלתי פוגע ☺



סיכום

כאן פחות או יותר הסיפור נגמר. את המטרה שהצבתי לעצמי- השגתי. אומנם אני לא בטוח שכעת האינטרנט הוא מקום בטוח לגלוש בו, אבל לפחות אני יודע שלא העלמתי עין, ואם בזכות הפעולה הזאת הצלחנו להציל עוד כמה משתמשים תמימים- את שלנו עשינו.

בקשר לחוקיות הדברים, ברור לי שמדובר בפעולות אפורות, וכששאלתי עורך-דין המתעסק בנושא האם פרסום הדברים כדאי- קיבלתי תשובה שלילית. ובכל זאת פרסמתי את הדברים, והנה הם לפניכם. הדבר נעשה אך ורק בכדי להראות לכם, הקוראים, שניתן להשיב לאותם סקריפט-קידיז שמנסים לפגוע במשתמשים תמימים ברחבי האינטרנט. מאמר זה אינו מהווה המלצה לביצוע פעולות דומות מצדכם - כאמור מדובר בתחום מאוד אפור.

Bitcoin - כסף דיגיטלי ב-P2P

מאת: ד"ר אריק פרידמן

"The chief value of money lies in the fact that one lives in a world in which it is overestimated."

H.L. Mencken

הקדמה

בשנתיים האחרונות הגיח סוג חדש של מטבע. מטבע שמאיים להפוך על-פיהם את סדרי הכלכלה העולמית ולשבור את התלות בממשלות ובנקים בכל הנוגע להעברת כספים בין פרטים. הביטקוין.

הביטקוין הוא כסף דיגיטלי. הרעיון של כסף דיגיטלי כשלעצמו אינו חדש, ונחקר כבר למעלה מ-20 שנה (ייצוג דיגיטלי של כסף אמיתי הוא כמובן ותיק בהרבה). יתר על כן, מטבעות דיגיטליים נמצאים כבר בשימוש נרחב, למשל נקודות מיקרוסופט לרכישת משחקי XBOX, או Facebook Credits לביצוע עסקאות בפייסבוק. החידוש והייחוד של ביטקוין הוא שמדובר במטבע שאין אף גוף מרכזי ששולט או מפקח עליו, אלא הוא מנוהל על-ידי מערכת Peer to Peer. לאי-תלות זו יש מספר השלכות מעניינות. ראשית, הצורך במתווכים (בנקים) לביצוע עסקאות במערכות המסורתיות כרוך בעמלות, שמייקרות את עלות העסקאות. עלות זו הופכת להיות בעייתית במיוחד כשמדובר בעסקאות קטנות (מיקרו-תשלומים). שנית, במערכות המסורתיות אין יכולת לבצע תשלומים לא הפיכים תמורת שירותים שאינם הפיכים, אלא אם נעשה שימוש בכסף פיסי. הונאות הן חלק מהמשחק, ותמיד יש סיכון שהעסקה לא תכובד (צ'ק שחוזר, עסקת אשראי שבוטלה). מערכת הביטקוין מבטיחה לתת מענה לבעיות אלה, תוך התבססות על כלים קריפטוגרפיים וללא צורך באמון בגורם צד ג' כגון בנק או חברת אשראי. אחד האתגרים הגדולים במערכת כזו הוא למנוע שימוש כפול במטבע (כלומר תשלום עם אותו מטבע ביותר מעסקה אחת בו זמנית). במאמר זה נבחן כיצד עובדת מערכת ביטקוין, וכיצד היא מתמודדת עם הונאות.

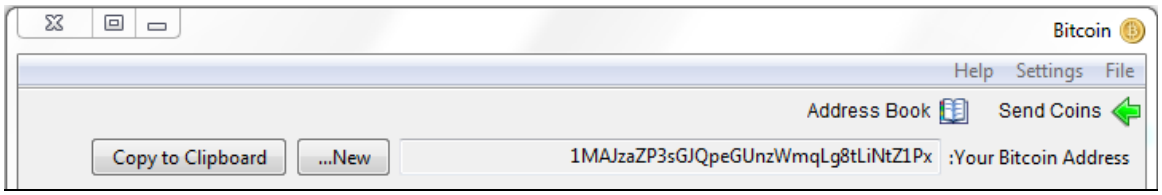
המאמר מניח כי הקורא מכיר את המושגים של מפתחות פרטיים וציבוריים, חתימות דיגיטליות, ופונקציות תמצות קריפטוגרפיות. המעוניינים במידע על מפתחות ציבוריים וחתימות יכולים לקרוא את המאמר של הלל חימוביץ' בנושא בגליון 3, והנושא של פונקציות תמצות קריפטוגרפיות תואר במאמר של ד"ר אור דונלקמן בגליון 21.

מאיפה הגיע BitCoin?

המערכת של ביטקוין החלה לעבוד ב-2009. לא ידוע מי מקים המערכת - הוא השתמש בשם הבדוי סטושי נקמוטו (Satoshi Nakamoto) וטען שהוא מיפן (אם כי נראה שיש לו שליטה מעולה באנגלית, ולא נראה שימוש ביפנית באתר או בקוד של ביטקוין). סטושי פרסם את עקרונות המערכת במסמך טכני, ובמקביל האתר של ביטקוין ספק את התוכנה עצמה ופרטים נוספים. עם זאת, אין צורך להסתמך על מהימנותו של סטושי כדי לבטוח בביטקוין. התוכנה של ביטקוין מנוהלת כתוכנת קוד פתוח, וכל אחד יכול לגשת לאתר של ביטקוין ב-Sourceforge, להוריד את הקוד, לבחון אותו, ולהדר אותו בעצמו.

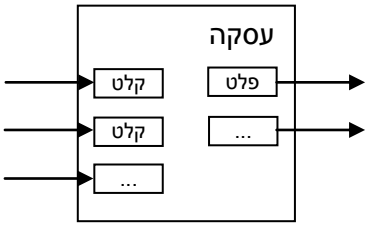
מי יכול להשתמש ב-BitCoin?

כל אחד יכול להשתמש בביטקוין. כל מה שנדרש הוא להוריד את התוכנה של ביטקוין מהאתר של ביטקוין, ואפשר להתחיל לקבל ולבצע תשלומים בביטקוין. מאחר ואין גוף מרכזי שמנהל את המערכת, אין צורך לפתוח חשבון או להזדהות. עם התקנת התוכנה, מוקצה למשתמש זוג מפתחות ציבורי-פרטי חדש, "כתובתו" של המשתמש נגזרת ישירות מהמפתח הציבורי שלו, וניתן להשתמש בה כיעד לתשלום. המשתמש יכול לייצר לעצמו מפתחות נוספים כאוות נפשו. למעשה, התוכנה של ביטקוין מעודדת זאת, וכאשר מתקבל תשלום בכתובת של המשתמש, התוכנה תייצר מייד מפתחות חדשים (וכתובת חדשה) לשימוש לעסקה הבאה. באופן זה, יהיה קשה לקשר עסקאות שונות שביצע אותו המשתמש, אם נעשו באמצעות כתובות ביטקוין שונות.



תוכנת ביטקוין היא תוכנת P2P, והיא מוצאת משתתפים אחרים ברשת באמצעות חיבור לשרת IRC (ערוץ #bitcoin בשרת irc.lfnet.org). במידה ולא הצליחה לייצר קשר כזה, נעשה שימוש ברשימת משתתפים הכלולה בקוד התכנית, ודרכם ניתן ליצור קשר עם משתתפים נוספים.

ביטקוין היא למעשה מערכת לעיבוד עסקאות, והן מהוות את לב המערכת. לכל עסקה יהיו קלטים ופלטים:



הקלטים הם עסקאות קודמות, המגדירות סכומי מטבעות המשמשים בעסקה החדשה, והפלטים הם סכומי המטבעות המוקצים למשתתפים בעסקה (ומהווים בסיס לעסקאות עתידיות). בדרך כלל יהיו שני פלטים: אחד עבור הקצאת סכום למוטב בעסקה, ואחד כדי לייצג את העודף למבצע העסקה. ברוב העסקאות סכום המטבעות בקלט וסכום המטבעות בפלט יהיו שווים, אולם יש גם סוג מיוחד של עסקאות המשמשות ל"הטבעת" מטבעות חדשים, נושא שיפורט בהמשך. למשל, להלן עסקת הטבעת מפתחות:

```
{
  "hash":"f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6",
  "ver":1,
  "vin sz":1,
  "vout sz":1,
  "lock time":0,
  "size":134,
  "in":[
    {
      "prev out":{
        "hash":"0000000000000000000000000000000000000000000000000000000000000000",
        "n":4294967295
      },
      "coinbase":"04e6ed5b1b015c"
    }
  ],
  "out":[
    {
      "value":"50.00000000",
      "scriptPubKey":"04283338ffd784c198147f99aed2cc16709c90b1522e3b3637b312a6f9130
       e0eda7081e373a96d36be319710cd5c134aaffba81ff08650d7de8af332fe4d8cde20 OP_CHECKSIG"
    }
  ]
}
```

כעסקה להטבעת מטבע חדש אין לה קלטים (prev_out מאופס), ויש לה פלט אחד, המקצה 50 ביטקוין לבעל המפתח הפומבי שמתחיל ב-04283... העסקה מיוצגת באמצעות התמצות שלה, המתחיל ב-f5d8ee... ומופיעה בתחילתה.

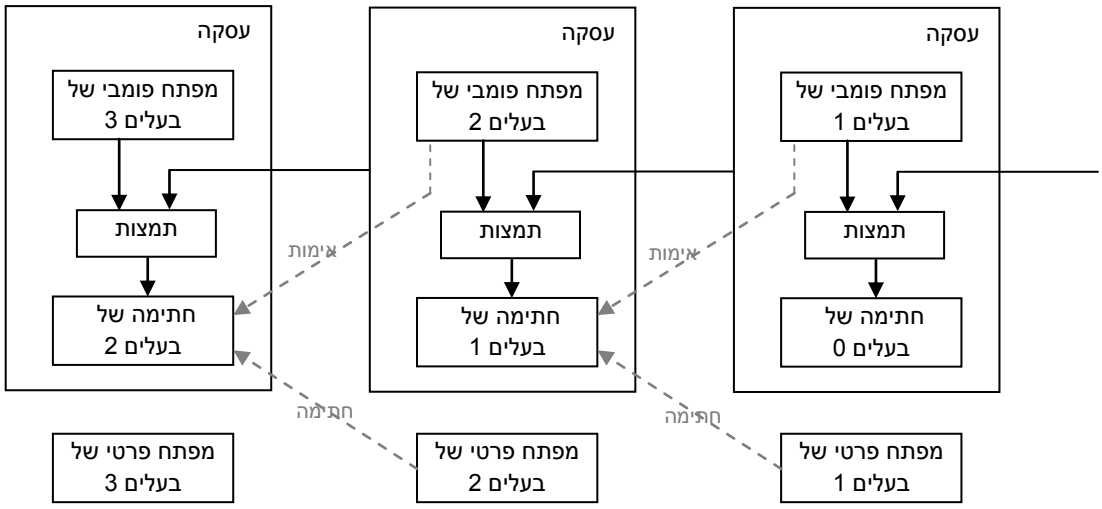
להלן עסקת העברת כספים:


```
{
  "hash":"5a4ebf66822b0b2d56bd9dc64ece0bc38ee7844a23ff1d7320a88c5fdb2ad3e2",
  "ver":1,
  "vin_sz":1,
  "vout_sz":1,
  "lock_time":0,
  "size":158,
  "in":[
    {
      "prev_out":{
        "hash":"f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6",
        "n":0
      },
      "scriptSig":"304502206e21798a42fae0e854281abd38bacd1aeed3ee3738d9e1446618c4571d10
      90db022100e2ac980643b0b82c0e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c6b241501"
    }
  ],
  "out":[
    {
      "value":"50.00000000",
      "scriptPubKey":"OP_DUP OP_HASH160 404371705fa9bd789a2fcd52d2c580b65d35549d
      OP_EQUALVERIFY OP_CHECKSIG"
    }
  ]
}
```


ניתן לראות כי הקלט לעסקה הזו הוא העסקה הקודמת (שערכה 50 ביטקוין), ויש לה פלט יחיד המקצה את כל 50 הביטקוין למשתמש שתמצות המפתח הפומבי שלו מתחילה ב-...40437.

למעשה, עסקאות אלה *מגדירות* את המטבעות ואת בעליהם. באופן כללי, כל מטבע מוגדר על-ידי סדרת עסקאות, הראשונה בהן היא עסקת הטבעת המטבע, ואחריה יש שרשרת של עסקאות המתארות את שרשרת הבעלות על המטבע. כדי להעביר מטבע, בעל המטבע מייצר עסקה חדשה. עסקה זו תכלול את התמצות של העסקה הקודמת (כלומר, ממשיכים את שרשרת הבעלות) ואת המפתחות הפומביים של נותן המטבע ומקבל המטבע, והיא תהיה חתומה על-ידי הבעלים של המטבע. קיימת אפשרות גם להציע עמלה עבור עיבוד העסקה (פרטים בהמשך). התהליך מתואר בתרשים בעמוד הבא. כיוון שרק לבעלים החוקיים יש את המפתח הפרטי המתאים למפתח הפומבי בעסקה האחרונה בשרשרת, הוא היחיד שמסוגל לייצר חתימה תקינה ובכך להכריז על העסקה. עם זאת, העסקה החתומה מהווה רק ראייה ראשונית (אם כי עדיין לא מספקת, כפי שנראה) לכך שהועברו המטבעות.

אם כך, עסקאות ביטקוין מאפשרות לייצר מטבעות ולהעביר אותם מיד ליד. אבל מה בעצם מונע מכל אחד לייצר מטבעות כאוות נפשו? או להשתמש באותו מטבע פעמיים באמצעות יצירת שתי עסקאות שונות על-בסיס אותה עסקה קיימת? התשובה טמונה בתהליך עיבוד העסקאות של ביטקוין, המתבסס על יצירת *בלוקים* של עסקאות.



עיבוד עסקאות BitCoin - בלוקים והוכחת עבודה

כאשר מישהו מייצר עסקת ביטקוין וחותם עליה עם המפתח הפרטי שלו, הוא מכריז על נכונותו לבצע את התשלום, אולם העסקה לא תקפה עד שמסה קריטית של משתתפים במערכת ה-P2P של ביטקוין מכירים בעסקה ומקבלים אותה. הדרך שבה עסקאות נבחנות ומתקבלות על-ידי המשתתפים במערכת היא באמצעות תהליך עיבוד *בלוקים* של עסקאות. היסטוריית העסקאות של ביטקוין מתוחזקת על-בסיס שרשרת יחידה המורכבת מבלוקים, כאשר כל בלוק מכיל עסקה אחת או יותר. נכון לזמן כתיבת שורות אלה, השרשרת מכילה 142,620 בלוקים. הבלוק הראשון בשרשרת (בלוק 0) נקרא בלוק בראשית (The Genesis block), והוא נראה כך:

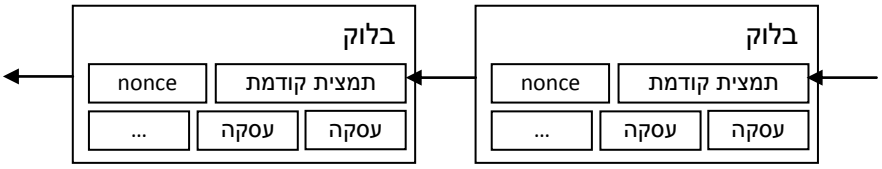

```
{
  "hash":"000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f",
  "ver":1,
  "prev_block":"0000000000000000000000000000000000000000000000000000000000000000",
  "mrkl_root":"4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",
  "time":1231006505,
  "bits":486604799,
  "nonce":2083236893,
  "n_tx":1,
  "size":285,
  "tx":[
    {
      "hash":"4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",
      "ver":1,
      "vin_sz":1,
      "vout_sz":1,
      "lock_time":0,
      "size":204,
      "in":[
        {
          "prev_out":{
            "hash":"0000000000000000000000000000000000000000000000000000000000000000",
```

```
            "n":4294967295
          },

"coinbase":"04ffff001d0104455468652054696d657320030332f4a616e2f32303039204368616e63656c6c6f
72206f6e206272696e6b206f66207365636f6e642061696c6f757420666f722062616e6b73"
        }
      ],
      "out":[
        {
          "value":"50.00000000",

"scriptPubKey":"04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb649f6bc3f
4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6bf11d5f OP_CHECKSIG"
        }
      ]
    }
  ],
  "mrkl_tree":[
    "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"
  ]
}
```


הבלוק הזה כולל עסקה אחת (50 הביטקוין הראשונים שנוצרו). הקוד של ביטקוין כולל את הבלוק הזה, כך שכל לקוח ביטקוין מסוגל לוודא ששרשרת הבלוקים מתחילה בבלוק הבראשית. כל בלוק נוסף בשרשרת כולל את התמצית של הבלוק שלפניו, עסקה אחת או יותר, ערך מספרי בשם nonce (פרטים בהמשך), ותמצית של ערכי הבלוק. כך נוצרת השרשרת הבאה:



שרשרת בלוקים חוקית חייבת להתחיל בבלוק הבראשית, אולם ניתן לייצר יותר משרשרת אחת כזאת. במקרה ומשתתפים ברשת מכריזים על יותר משרשרת חוקית אחת במערכת, מערכת ביטקוין (כלומר האוסף של המשתתפים החברים בה) תמיד תיקח את השרשרת הארוכה ביותר. כפי שנראה בהמשך, יש צורך בהשקעת משאבים חישוביים ניכרים לצורך יצור שרשרת חוקית. במידה וגורם זדוני כלשהו ירצה לשכתב את היסטוריית העסקאות ולספק שרשרת חלופית, הוא יצטרך לספק שרשרת ארוכה יותר מהשרשרת הקיימת. משימה זו תדרוש שיהיה ברשותו כוח חישובי העולה על זה של כל שאר המשתתפים במערכת ביחד. במידה ואין לו כוחות חישוביים כאלה, זיוף שרשראות הופך להיות משימה בלתי אפשרית מכל בחינה מעשית. במידה ונוצר מצב ואכן לגורם כלשהו יש שליטה על רוב הכוחות החישוביים במערכת, הרי שככל הנראה יוכל להרוויח יותר על-ידי עיבוד עסקאות ויצירה של בלוקים "לפי הספר", מאשר באמצעות רמאות שתגרום לאובדן אמון במערכת ובריחת המשתתפים. באופן זה המערכת מתוכננת לתמוך בביצוע מהימן של עסקאות המשתתפים.

"How can you have money," demanded Ford, "if none of you actually produce anything? It doesn't grow on trees you know."
"If you would allow me to continue…"
Ford nodded dejectedly.
"Thank you. Since we decided a few weeks ago to adopt leaves as legal tender, we have, of course, all become immensely rich."
Ford stared in disbelief at the crowd who were murmuring appreciatively at this and greedily fingering the wads of leaves with which their track suits were stuffed.
"But we have also," continued the Management Consultant, "run into a small inflation problem on account of the high level of leaf availability, which means that, I gather, the current going rate has something like three major deciduous forests buying one ship's peanut."
Murmurs of alarm came from the crowd. The Management Consultant waved them down.
"So in order to obviate this problem," he continued, "and effectively revalueate the leaf, we are about to embark on a massive defoliation campaign, and… er, burn down all the forests. I think you'll all agree that's a sensible move under the circumstances."

(Life, the Universe and Everything, Douglas Adams)

כפי שיתואר בהמשך, יצור בלוק כרוך בעבודה חישובית, והמערכת מכוונת לייצר בלוק חדש בערך כל 10 דקות. העסקה הראשונה (ולעיתים היחידה) בכל בלוק תהיה תמיד עסקה המעניקה מטבעות ביטקוין ליוצר הבלוק כשכר על העמל החישובי. חלק משכר זה מגיע ממטבעות חדשים שנוצרים (כלומר העסקה הראשונה היא עסקת הטבעת מטבעות), וחלקו מגיעים מעמלות שמציעים מייצרי העסקאות. הקוד של ביטקוין נכתב כך שב-4 השנים הראשונות, כל בלוק חדש ייצר 50 ביטקוין (כך שיווצרו 10,500,000 מטבעות בתקופה זו). כמות זו תחולק בחצי כל 4 שנים, כך שבשנים 8-4 יתקבלו 25 ביטקוין על כל בלוק (עם 5,250,000 מטבעות חדשים בתקופה), בשנים 12-8 יתקבלו 12.5 ביטקוין וכן הלאה. לאורך זמן, כמות המטבעות הכוללת תתקרב ל-21,00,000, ואף פעם לא תעבור את רף זה - בשנת 2140 התשלום במטבעות חדשים על יצירת בלוק יאופס. תכנון זה של המערכת צופה שבשלבי החיים המאוחרים יותר של המערכת העלות של עיבוד העסקה תכוסה בעיקרה על-ידי העמלות שמציעים מייצרי העסקאות, והצורך בתמריץ של מטבעות חדשים יפחת. כמו-כן, במערכת כזו לעולם לא תהיה אינפלציה - יש חסם עליון וקשיח לכמות הכסף שיווצר.

אחד המרכיבים המרכזיים במערכת ביטקוין הוא העקרון של הוכחת עבודה (Proof of work). עקרון זה הוצע במקור ב-1997 על-ידי אדם בק, בהקשר של מערכת hashcash למניעת ספאם. הרעיון היה שכל משלוח דואר אלקטרוני יצריך את השולח לספק הוכחה לכך שהשקיע כמות מסויימת של משאבי עיבוד בשליחת ההודעה, למשל, חישוב שיצריך זמן עיבוד של שתי שניות. עבור משתמשים רגילים, זה לא היה משפיע באופן מהותי, שכן עיכוב של שתי שניות יהיה כמעט ובלתי מורגש מבחינתם. לעומת זאת, עבור

ספאמרים השולחים מליונים של תכתובות אלקטרוניות תידרש השקעה משמעותית של זמן וכסף, דבר שיהפוך את עסקי הספאם ללא כדאיים. בדיעבד, שיטה זו התבררה כלא מוצלחת למניעת ספאם, בעיקר מפני ששימוש בה היה פוגע גם בגופים השולחים כמויות גדולות של דואר לגיטימי. אולם הרעיון יושם בהצלחה במסגרת המערכת של ביטקוין.

על מנת שבלוק יהיה חוקי, עליו לקיים את התנאי הבא: התמצית של הבלוק צריכה להיות קטנה מערך מטרה מסויים, המוסכם על-ידי המערכת. במילים אחרות, התמצית של הבלוק צריכה להתחיל בשרשרת של אפסים. המפתח ליצירת תמצית כנדרש טמון במחרוזת ה-nonce הנמצאת בבלוק. תהליך יצירת בלוק חדש מתבצע באופן הבא:

1. מצא את שרשרת הבלוקים החוקית הארוכה ביותר במערכת.
2. הגרל מחרוזת nonce. יצר בלוק חדש המורכב מתמצית הבלוק האחרון בשרשרת, מהעסקאות הממתינות לעיבוד (כולל עסקת יצירת מטבעות המעניקה תשלום למייצר הבלוק) ומה-nonce.
3. אם ערך התמצית של הבלוק החדש מתחיל במספר האפסים הדרוש, פרסם את הבלוק.
4. אחרת חזור ל-2 עם nonce אחר.

אם בשלב כלשהו בתהליך תפורסם שרשרת בלוקים חוקית ארוכה יותר (כלומר מישהו הצליח לחשב את הבלוק הבא), אז תהליך יצירת הבלוק יעבור להתבסס על שרשרת זו. במידה ומישהו מנסה לייצר בלוק שאינו עומד בכללים של המערכת (למשל, אי-התאמה בחישובי תמצות, חריגה מסכום הכסף שמגיע למייצר הבלוק), הבלוק יידחה על-ידי שאר המשתתפים ברשת, וכך יימנעו רמאויות.

ערך המטרה של התמצית, או מספר האפסים הדרושים, נקבע על-פי הזמן שלקח לייצר את 2016 הבלוקים הקודמים. המטרה היא להתאים את רמת הקושי של הבעייה החישובית לכמות המשאבים החישוביים הקיימים ברשת, כך שבממוצע יווצר בלוק חדש כל 10 דקות.

באתר http://blockexplorer.com ניתן לעקוב אחר ייצור הבלוקים, ולראות את כלל היסטוריית הבלוקים (והעסקאות) עד כה.

בימים המוקדמים יותר של המערכת, תוכנת ביטקוין כללה את האפשרות להשתתף בתהליך עיבוד הבלוקים, ומכאן גם את היכולת לזכות במטבעות ביטקוין. אולם עם התפשטות המערכת ועליית הערך של ביטקוין, חל תהליך של התמחות ביצור מטבעות. למשל, כרטיסים גרפיים יכולים לכרות מטבעות בקצב מהיר פי 100 ויותר מאשר מעבד מחשב רגיל. התמריץ לייצור מטבעות עלה, והחלו להופיע חוות לכריית ביטקוין. למעשה, צריכת החשמל של חוות כריית ביטקוין מזכירה את זו של חממות לגידול מריחואנה, וכבר דווח על מקרים שמשטרה פרצה לבתים במטרה למצוא מגדלי סמים ומצאה גיקים עם מחשבים.

בנוסף, משתתפים החלו להתאגד לכרייה משותפת של מטבעות (pools), ויצרו קבוצות כגון deepbit, slush ו-btcguild.

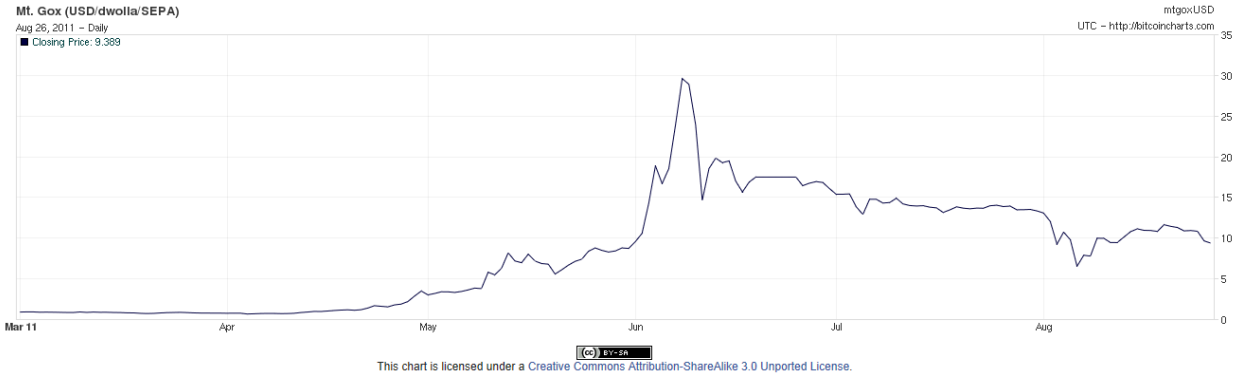
עם גדילת משאבי החישוב ברשת, רמת הקושי של הבעייה החישובית עלתה (נכון לכתיבת שורות אלה דרושה בתמצית תחילית של 52 אפסים). מאחר ובמצב הנוכחי עלות כריית הביטקוין תשתלם רק עבור אלה הבונים מערכת ייעודית שתבצע את הכרייה ביעילות, האפשרות לכרייה הוסרה מהתוכנה, ויש צורך להוריד תוכנה ייעודית לכריית ביטקוין.

(מכרה ביטקוין ברוסיה. התמונה נלקחה מ-http://bitcoinitalia.com/forum/viewtopic.php?f=11&t=26)

שווה משהו, ה-BitCoin הזה?

העסקה המתועדת הראשונה בה נרכש פריט פיסי תמורת ביטקוין התקיימה ב-22 במאי 2010. בעסקה זו המשתמש laszlo שילם 10,000 ביטקוין עבור פיצה. באותו זמן ערכם של 10,000 ביטקוין היה בערך 41$. כיום, כעבור שנה ושלושה חודשים, סכום דומה של ביטקוין שווה בערך 100,000$. שעור קטן על חסכון.

עם התפתחות מערכת ביטקוין, החלו לצוץ בורסות (Bitcoin markets) המאפשרות לסחור בביטקוין תמורת מטבעות אחרים. למשל, אחד מהשווקים הבולטים הוא MTgox, וניתן לראות להלן את התנודות בשערי מטבע הביטקוין ביחס לדולר האמריקאי לאורך ששת החודשים האחרונים:



(שער הביטקוין ביחס לדולר האמריקאי. התרשים נלקח מ- http://bitcoincharts.com/charts)

מערכת ביטקוין הייתה על רכבת הרים בתקופה זו, במיוחד לאור החשיפה הגוברת שקיבלה מאז מאי ויוני השנה. בשלב מסויים מטבע הביטקוין נסחר תמורת כ-30$, אולם מאז רמתו ירדה לאזור ה-10$, עדיין הרבה מעל ערכו רק לפני חצי שנה. קיימים גם שווקי מסחר נוספים לביטקוין, וניתן לעקוב אחריהם באתר http://bitcoincharts.com/markets. אתרים נוספים בהם ניתן לעקוב אחר קורותיו של הביטקוין הם http://www.bitcoinmonitor.com ו-http://bitcoinwatch.com. נכון לזמן כתיבת שורות אלה, קיימים כ-7 מליון מטבעות ביטקוין, ששווים הכולל מוערך בכ-68 מליון דולר אמריקאי.

לאחר דהירה בערכו של הביטקוין במהלך החודשים מאי ויוני, נראה כי המגמה התהפכה. בעקבות צניחה נוספת בערכו של הביטקוין בתחילת אוגוסט, כתב של הפורבס מיהר להספיד את המטבע, אם כי הירידות נבלמו מאז. כך או כך, עתידו של המטבע הוא נושא לספקולציות.

מצד אחד, מערכת ביטקוין מתמודדת בהצלחה מרשימה עם האתגרים שדורשת מערכת פיננסית בהיקף גדול, ובפרט היכולת להבטיח את מהימנות העסקאות שמבוצעות ומניעת שימוש כפול במטבעות. בנוסף, ההבטחה לעמלות עסקה נמוכות הופכת את ביטקוין למערכת מבטיחה מאוד לאור התגברות הצורך בדרך זולה לביצוע עסקאות קטנות ומיקרו-תשלומים באינטרנט. הצדדים הטכנולוגיים והקריפטוגרפיים של המערכת הופכים אותה לחביבת הגיקים; האנונימיות שנותנת היכולת להתחבא מאחורי מפתח פומבי חסר זהות קורצת לארגונים הפועלים מחוץ לחוק ולאקטיביסטים. שוק הביטקוין עדיין צעיר יחסית, ויכול לספק הזדמנויות מעניינות ליזמים. למשל, השוק עדיין לא לגמרי משוכלל, וניתן לנצל פערי שערים בין שווקי מסחר שונים לארביטרז'.

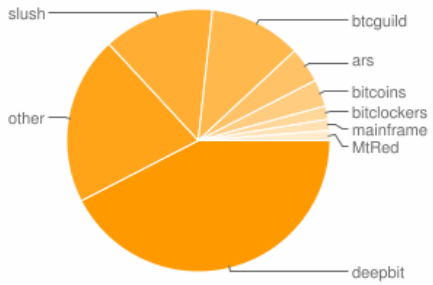
מצד שני, היתרונות של ביטקוין הם גם החסרונות שלה. גופי אכיפת חוק בוחנים את המערכת בחשדנות כמנגנון שעשוי לשמש להלבנת כספים, ואין להוציא מכלל אפשרות שהשימוש במערכת יוצא מחוץ לחוק. למשל, ארגון EFF (Electronic Frontier Foundation), קבוצה העוסקת בשימור זכויות אזרחיות בעולם הדיגיטלי, אימצה בתחילה את ביטקוין ואיפשרה תרומת ביטקוין לארגון. אולם עם התגברות העניין בביטקוין, הארגון נסוג בו מהחלטתו, בין השאר בשל חוסר הבהירות סביב ההשלכות החוקיות הכרוכות בהקמת מערכת מטבע חדשה, ורצונו של הארגון להתרחק ממעורבות כצד ישיר בעימות אפשרי עתידי סביב המערכת.

מכשול נוסף העומד בדרכה של המערכת הוא רמת הידע והכישורים הגבוהה הנדרשת לשימוש בטוח במערכת. בעלות על מטבעות ביטקוין נשענת כולה על אבטחת המפתחות הפרטיים המתאימים. אובדן של מפתחות אלה (למשל בעקבות קריסת כונן קשיח) או גניבתם על-ידי קוד זדוני, משמעותם אובדן בלתי הפיך של הכספים. למשל, ב-13 ביוני משתמש בשם allinvain דיווח על כך שהמחשב שלו נפרץ ונגנבו ממנו 25,000 מטבעות ביטקוין (שווי-ערך ל-375,000$ בזמן הגניבה). על-פי allinvain, הפורץ הצליח להשיג את המפתח הפרטי שלו, והשתמש בו כדי להעביר אליו את הבעלות על המטבעות. העסקה, כמובן, בלתי הפיכה, ואין אמצעי פשוט למציאת הפושע על-פי המפתח הפומבי בו השתמש כיעד לעסקה. מקרה זה ממחיש כי מערכת ביטקוין, לפחות בצורתה הנוכחית, לא מתאימה לתפוצה מעבר לאוכלוסיית הגיקים, שהינם מתוחכמים מספיק כדי להגן כראוי על קובץ ה"ארנק" בו מאוחסנים המפתחות הפרטיים שלהם; האדם מהרחוב יהווה טרף קל לפושעים מקוונים. בנקים וחברות אשראי מספקים בטחון והגנה

מפני הונאות, וערכם של בטחונות אלה עבור רוב האוכלוסיה משמעותי ומהותי יותר מהיתרונות שמציעה ביטקוין.

סיכום

המגמות של החודשים האחרונים במערכת ביטקוין מעלות תהייה נוספת לגבי עתידה של מערכת כזו. התמריץ הכלכלי הביא למיזוג של משאבי מחשוב, ונכון להיום מספר מצומצם יחסית של גופים שולט ברוב המכריע של משאבי החישוב במערכת. אם מערכת ביטקוין תצליח לשרוד לאורך זמן ולהשיג תמיכה מספיק רחבה כדי לשמור על ערכו של המטבע, גופים אלה יהיו בעמדה לקבוע כרצונם תעריפים לביצוע עסקאות. באופן מעשי, הביטקוין עשוי להפוך ממטבע המתוחזק על-ידי קהילת המשתמשים בו, למטבע המנוהל על-ידי מספר מצומצם של גופים פרטיים. באופן אירוני, ייתכן כי דווקא הצלחת המערכת תביא לעלייה הדרגתית בעמלות הנדרשות (ביכולתם של מייצרי הבלוקים להתעלם מעסקאות המציעות עמלות נמוכות), ובסופו של דבר תחתור נגד אחד הצרכים שהביאו ליצירת המערכת מלכתחילה.



(חלוקת משאבי החישוב ברשת ביטקוין נכון ל-26.8.11 - נלקח מתוך http://bitcoinwatch.com)

בין אם מטבע הביטקוין ימשיך ויתפוס תאוצה, ובין אם העניין בו יתמוסס והוא יגווע, מערכת הביטקוין פרצה דרך חדשה, וממחישה את הצורך והעניין בחלופות לשיטות המסחר המסורתיות. קרוב לוודאי שיש עוד דרך מרתקת בפניה של טכנולוגיה זו.

מקורות ומידע נוסף

המאמר מבוסס בעיקרו על המידע באתר של ביטקוין (http://www.bitcoin.org) ובעיקר עמוד השאלות הנפוצות, ועל המאמר של סטושי נקמוטו, Bitcoin: A Peer-to-Peer Electronic Cash System. אין לראות בשום מידע המובא כאן בגדר המלצה להשתתף או להימנע מלהשתתף במערכת ביטקוין. בעלי ביטקוין שנהנו מקריאת המאמר, יכולים להביע את הערכתם באמצעות תרומה בביטקוין לכתובת MAJzaZP3sGJQpeGUnzWmqLg8tLiNtZ1Px1. ☺

למאזיני פודקאסטים, להלן מספר המלצות חמות נוספות:

1. פרק 98 בפודקאסט *עושים היסטוריה* של רן לוי (ברכות לרגל הפרק ה-100!!) עוסק בהיסטוריה של האינפלציה, ודן גם בביטקוין.
2. פרק 287 בפודקאסט Security Now של סטיב גיבסון מבית TwiT כולל תיאור נרחב של ביטקוין וכיצד המערכת פועלת.
3. פרק 423 בפודקאסט This American Life עוסק בהמצאת הכסף, וממחיש את המסקנה שהכסף הוא אשליה.

על המחבר

ד"ר אריק פרידמן הוא חוקר בתחום של פרטיות ואבטחת מידע, ובעיקר שילובם במסגרת אלגוריתמים ללמידה ממוחשבת וכריית נתונים. אריק סיים את לימודי הדוקטורט בפקולטה למדעי המחשב בטכניון בשנת 2011, והוא מחזיק גם בתואר MBA מאוניברסיטת תל-אביב. עד לאחרונה אריק שילב את המחקר עם תפקיד של Program Manager במיקרוסופט, שם עבד על מוצרים בתחום אבטחת המידע והפרטיות.

תזמון חוטים ב-Windows

מאת: סשה גולדשטיין

הקדמה

בפעם הקודמת הסתכלנו על המימוש של מנגנוני סנכרון ב-Windows. הצורך במנגנוני סנכרון עולה משימוש שמערכת ההפעלה עושה במספר מעבדים, ומהרצת מספר חוטים (Threads) על מעבד אחד או כמה מעבדים.

במאמר זה נסקור כיצד Windows מחליטה איזה חוט יש להריץ, על איזה מעבד יש להריץ אותו, כיצד ניתן להשפיע על החלטות אלה, ואילו אופטימיזציות ושיקולים משפיעים על ההחלטות האלה כתוצאה משינויי החומרה של השנים האחרונות.

מדוע בכלל להשתמש בחוטים?

מבלי להיכנס לפרטים שחורגים מגבולות המאמר, ניתן לומר שיש שלוש סיבות עיקריות לשימוש בחוטים הרלוונטיות הן בכתיבת תוכניות משתמש והן בפיתוח מערכת ההפעלה עצמה:

1. **ניצול מספר מעבדים** - חוט אחד יכול לרוץ על מעבד אחד בלבד בכל רגע נתון, ולכן על מנת להשתמש בצורה מיטבית במספר גדול של מעבדים יש ליצור מספר גדול של חוטים ולדאוג לספק להם עבודה בכל רגע. (כמובן, מטרתנו ברכישת מחשב עם מספר רב של מעבדים היא להביא אותם לניצולת של 100%, ולא להניח להם להתבטל).

2. **שילוב של פעולות קלט/פלט ופעולות חישוב** - בזמן שרכיב חומרה מסוים (למשל, דיסק קשיח) מטפל בבקשת קלט/פלט, המעבד פנוי ויכול להמשיך לבצע חישובים. תוכנית שמבצעת הן קלט/פלט והן חישובים יכולה להשתמש במספר חוטים. למשל, בעת שחוט מחכה לתוצאות של פעולת קלט/פלט, חוט אחר יכול לבצע חישובים ולהמשיך לנצל בצורה מיטבית את המשאבים.

3. **תגובתיות למשתמש** - שימוש במספר חוטים (אפילו במערכת בעלת מעבד אחד) מאפשר לתוכנית משתמש להישאר תגובתית, ע"י כך שאת הטיפול בפעולות המשתמש מבצעים בחוט אחד ואת הפעולות הדורשות עיבוד כבד מבצעים "ברקע", בחוט אחר.

מובן שהשימוש בחוטים מרובים כרוך גם בבעיות מרובות, שאת מקצתן ראינו במאמר הקודם - בעיות כגון חֶבֶק (deadlock), גישה לא מסונכרנת לזיכרון משותף, הפרות סדר פעולות הגישה לזיכרון ועוד. בכל זאת, היתרונות עולים על החסרונות, וכמעט כל תוכנית משתמשת היום במספר חוטים - לפעמים באמצעות תשתיות תוכנה מחוכמות המנהלות את יצירת החוטים והשמת עבודה לחוטים בעצמן.

כיצד מחליפים בין חוטים?

Windows מחליפה בין חוטים לעתים קרובות מאוד. כך נשמרת האשליה, אפילו במחשב בעל מעבד אחד, כאילו כמה תוכניות יכולות לרוץ במקביל. בהמשך נסקור את הזמן שניתן לכל חוט כאשר הוא מקבל את ההזדמנות לרוץ, אבל קודם יש לתת את הדעת לסוגיה הבאה: בכל פעם שמערכת ההפעלה בוחרת להחליף בין חוטים, עליה "להסיר" מן המעבד את החוט שמתבצע כעת, ו"להשים" על המעבד את החוט החדש. מה המשמעות של "להסיר" ו"להשים"?

מערכת ההפעלה מתחזקת מבנה נתונים הנקרא Context, שמתאר חוט שמתבצע כרגע על המעבד. ה-Context מכיל את כל האוגרים (Registers) של המעבד. למשל, במעבד x86, ה-Context מכיל את האוגר EIP המתאר את הפקודה הבאה שהמעבד יבצע, ואת האוגר ESP המתאר את ראש המחסנית. להלן ההגדרה של מבנה הנתונים CONTEXT עבור מעבדים ממשפחת x86[1]:

```
typedef struct _CONTEXT {
    DWORD ContextFlags;
    DWORD   Dr0;
    DWORD   Dr1;
    DWORD   Dr2;
    DWORD   Dr3;
    DWORD   Dr6;
    DWORD   Dr7;
    FLOATING_SAVE_AREA FloatSave;
    DWORD   SegGs;
    DWORD   SegFs;
    DWORD   SegEs;
    DWORD   SegDs;
```

[1] מבנה הנתונים CONTEXT עבור חוטים שאינם פעילים כרגע נשמר על המחסנית של החוט. ניתן גם להשפיע על תוכנו ישירות באמצעות הפונקציות GetThreadContext ו-SetThreadContext.

```
    DWORD   Edi;
    DWORD   Esi;
    DWORD   Ebx;
    DWORD   Edx;
    DWORD   Ecx;
    DWORD   Eax;
    DWORD   Ebp;
    DWORD   Eip;
    DWORD   SegCs;
    DWORD   EFlags;
    DWORD   Esp;
    DWORD   SegSs;
    BYTE    ExtendedRegisters[MAXIMUM_SUPPORTED_EXTENSION];
} CONTEXT;
```

(מבנה הנתונים CONTEXT עבור מעבדי x86, לקוח מתוך WinNT.h)

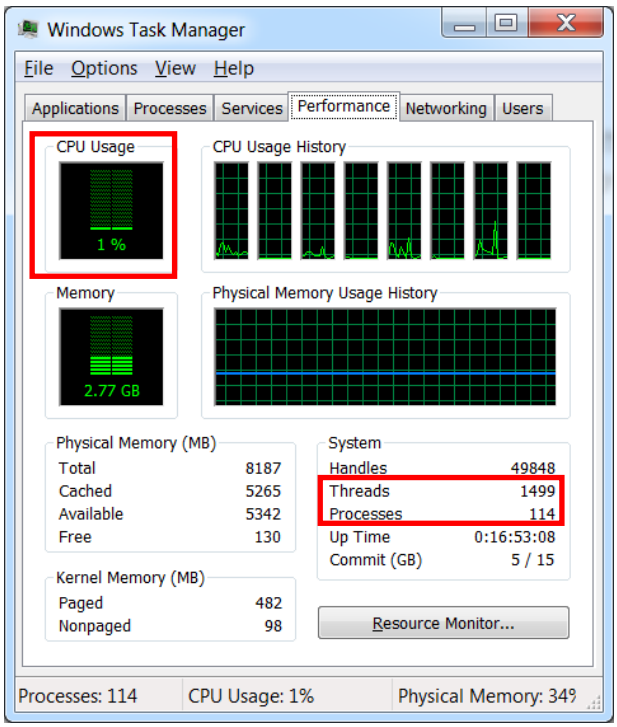
לכן, החלפה בין חוטים משמעותה שמירת האוגרים מהמעבד בתוך ה-Context של החוט המתבצע כרגע, ואחזור האוגרים של המעבד מתוך ה-Context של החוט החדש. זה מספיק בדיוק כדי להחליף בין החוטים - למשל, החלפת האוגר EIP תגרום לכך שהחוט החדש יריץ קוד שונה מהחוט הקודם, והחלפת האוגר ESP תגרום לכך שפעולות של החוט החדש על המחסנית לא ישפיעו על המחסנית של החוט הקודם.

ההבטחה הגדולה: החוט האחד בעל העדיפות הגבוהה ביותר שמוכן לריצה ירוץ תמיד

סכמת התזמון של מערכת ההפעלה מבוססת על רעיון אחד: בכל רגע נתון בוחרים מבין החוטים המוכנים לריצה את החוט בעל העדיפות הגבוהה ביותר, ומריצים אותו. אין זה משנה האם החוט הזה רץ כבר בעבר או שהוא נוצר ברגע זה, והאם העדיפות שלו השתנתה או מאז ומעולם הייתה הגבוהה ביותר. ההחלטה מבוססת אך ורק על המצב הנוכחי, מה שהופך את המימוש של המתזמן (Scheduler) לפשוט יחסית[2].

יש לציין שבמערכת טיפוסית, ישנם מאות חוטים בכל רגע, אפילו אם המשתמש לא הפעיל עדיין אף תוכנית משלו. אם כל החוטים האלה היו מוכנים לריצה, לא ניתן היה לעבוד בצורה סבירה על המערכת. מכאן, רוב החוטים אינם מוכנים לריצה בכל רגע נתון (כי הם ישנים, ממתינים על מנגנון סנכרון, ממתינים לתוצאות של פעולת קלט/פלט, ועוד) - מה שהופך את מלאכתו של המתזמן לפשוטה עוד יותר, שכן הבחירה היא בין מספר מצומצם יחסית של חוטים.

[2] יש לציין שרעיון "פשוט" זה עובד היטב במערכת בעלת מעבד אחד. ישנה הרחבה מתבקשת למספר מעבדים, אבל נראה בהמשך שמסיבות של ביצועים וסקאלאביליות לא משתמשים בה.



(תצלום מסך ממערכת בעלת 8 מעבדים, שעליה כרגע 1499 חוטים. בכל זאת, צריכת המעבד היא 1% - מרבית החוטים אם לא כולם אינם מוכנים לריצה כרגע)

למעשה, בעיקר במערכות קצה, במשך מרבית הזמן אין אפילו חוט אחד המוכן לריצה - ובמקרה כזה המתזמן מריץ קוד המכונה Idle Loop, שבין היתר יכול להוריד את צריכת החשמל של המערכת או אפילו לכבות מעבדים שאינם בשימוש.

מטבע הדברים ישנן גם מערכות שבהן עשרות חוטים המוכנים לריצה בכל רגע נתון. יכול להיות אפילו שיש חוט אחד או שניים שכל הזמן מוכנים לריצה, מאחר שכל עבודתם היא חישובית. במקרה כזה, ייתכן שהרעיון של הרצת החוט החשוב ביותר נראה קצת לא הוגן - ייתכן שקבוצה קטנה של חוטים מסגלת לעצמה את כל משאבי החישוב של המערכת, ולא מאפשרת לחוטים אחרים לרוץ בכלל. מסיבה זו, בין היתר, Windows מממשת מנגנון של שינויי עדיפות דינאמיים, כפי שנראה בהמשך.

עדיפות של חוט ב-Windows מורכבת משני חלקים: **עדיפות הבסיס** (Process Priority Class) של התהליך שלו, **והעדיפות היחסית** (Relative Priority, Thread Priority) של החוט. כל אחד מהחלקים נקבע בנפרד באמצעות פונקציות שונות - SetPriorityClass עבור עדיפות של תהליך, ו-SetThreadPriority עבור עדיפות יחסית של חוט. אולם בסופו של דבר, המערכת לא מתעניינת בעדיפות היחסית של החוט או בעדיפות הבסיס של התהליך, אלא רק בעדיפות האבסולוטית של החוט - מספר בין 0 ל-31. העדיפות 0 היא העדיפות הנמוכה ביותר, והעדיפות 31 היא העדיפות הגבוהה ביותר.

הטבלה הבאה מסכמת כיצד עדיפות הבסיס של התהליך והעדיפות היחסית של החוט משתלבות יחד ליצירת העדיפות האבסולוטית:

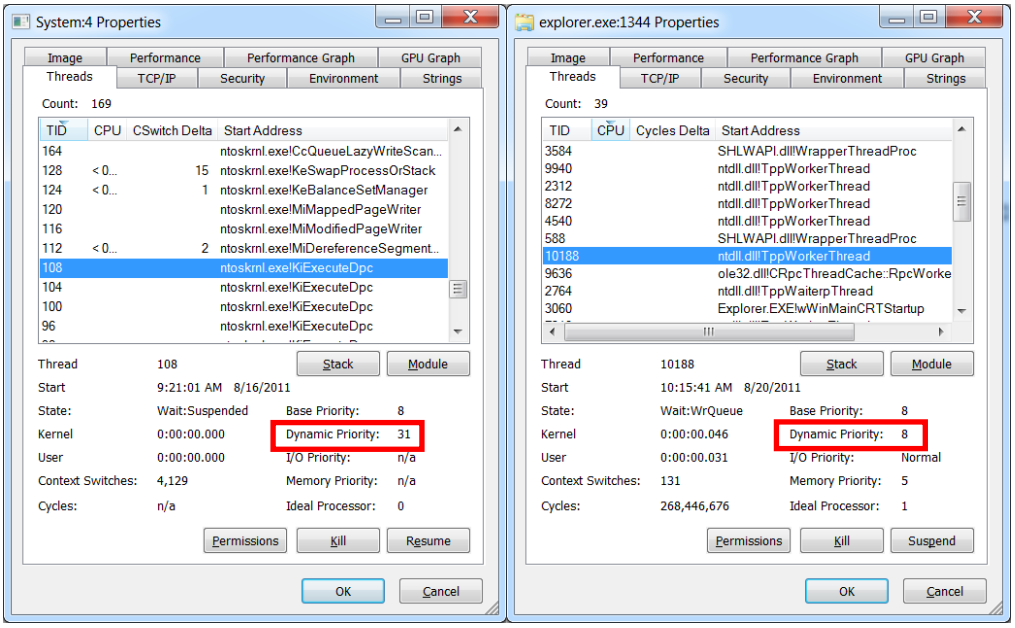
Process Priority Class	Thread Priority Level	Absolute Priority
Idle	Idle	1
	Lowest	2
	Below Normal	3
	Normal	4
	Above Normal	5
	Highest	6
	Time Critical	15
Below Normal	Idle	1
	Lowest	4
	Below Normal	5
	Normal	6
	Above Normal	7
	Highest	8
	Time Critical	15
Normal	Idle	1
	Lowest	6
	Below Normal	7
	Normal	8
	Above Normal	9
	Highest	10
	Time Critical	15

Above Normal	Idle	1
	Lowest	8
	Below Normal	9
	Normal	10
	Above Normal	11
	Highest	12
	Time Critical	15
High	Idle	1
	Lowest	11
	Below Normal	12
	Normal	13
	Above Normal	14
	Highest	15
	Time Critical	15
Realtime	Idle	16
	Lowest	22
	Below Normal	23
	Normal	24
	Above Normal	25
	Highest	26
	Time Critical	31

(רשימת העדיפויות האבסולוטיות האפשרויות כתלות בעדיפות היחסית של החוט ועדיפות הבסיס של התהליך)

יש לציין שקביעת עדיפות הבסיס Realtime דורשת מהמשתמש הרשאות מיוחדות, שלרוב נתונות רק למנהל המערכת (Administrator). שאר העדיפויות זמינות לכל התוכניות, אולם רוב התוכניות אינן טורחות לשנות הן את עדיפות הבסיס של התהליך והן את העדיפויות היחסיות של החוטים. האם זה כדאי? על פניו נראה שכדאי לשנות את רמת העדיפות בהתאם למשימה, או להעלות את העדיפות של התוכנית שלנו על חשבון תוכניות של משתמשים אחרים, או של יצרני תוכנה אחרים. אבל אם כל התוכניות היו משנות את העדיפות שלהן, למשל לעדיפות High, כי אז רמת העדיפות הזאת הייתה הופכת למעשה לרמת ברירת המחדל.

באילו רמות עדיפות משתמשת מערכת ההפעלה עבור החוטים שלה? ובכן, התשובה תלויה בהגדרה של "מערכת ההפעלה". למשל, החוטים של התהליך Explorer.exe רצים בעדיפויות רגילות, הגם שעבור משתמשים רבים הוא מערכת ההפעלה, כיוון שהוא מציג את שולחן העבודה ואת שורת המשימות. לעומת זאת, ישנם חוטים של המערכת שרצים בעדיפויות גבוהות מאוד, למשל חוטים של מנהל הזיכרון הווירטואלי.



(שני חוטים לדוגמה מתוך Explorer.exe והתהליך System. העדיפויות הן 8 ו-31 בהתאמה)

מה ההבדל בין Base Priority ל-Dynamic Priority? Windows מבצעת שינויים דינאמיים של העדיפות לפי פרמטרים שונים. בתצלומי המסך מוצגת העדיפות של החוט כפי שנקבעה לעומת העדיפות של החוט כפי שהמערכת החליטה שהיא צריכה להיות באותו רגע. עוד על שינויי עדיפות דינאמיים בהמשך.

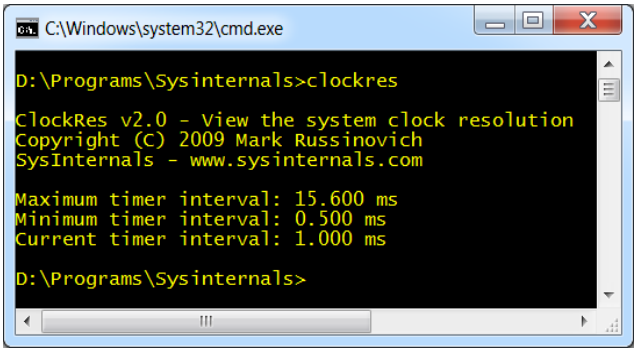
אם אתם מכירים את המושג IRQL (רמת עדיפות של פסיקות), אל תתבלבלו: אין קשר בין עדיפויות של פסיקות לבין עדיפויות של חוטים. למעשה, ניתן לומר שכל פסיקה היא יותר חשובה מאשר כל חוט - ולראיה, אפילו אם כרגע במערכת מתבצע חוט בעדיפות 31, ומתרחשת פסיקה, המערכת תפסיק את ביצוע החוט ותבצע את הפסיקה. מצד שני, במהלך טיפול בפסיקה לא מתרחשת (בדרך כלל) החלפת Context של חוטים, כך ששני המושגים - עדיפות של פסיקות ועדיפות של חוטים - הם אורתוגונאליים לחלוטין.

יחידת הזמן הניתנת לחוט (Quantum)

לפעמים המתזמן של המערכת מופעל כאשר מתרחשים שינויים בחוטים - למשל, העדיפות של חוט מסוים משתנה, חוט נכנס להמתנה, חוט "חוזר" מהמתנה לתוצאות של פעולת קלט/פלט. לעומת זאת, לעתים קרובות לא מתרחשים שינויים בקבוצת החוטים שמוכנה לריצה. במקרה כזה, דרוש מנגנון חיצוני שיגרום למתזמן לרוץ כדי שלא לאפשר לחוט אחד (שאולי תקוע בלולאה אינסופית) להשתלט על כל משאבי החישוב של המערכת.

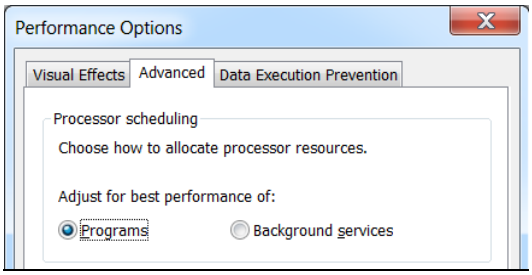
כאשר חוט נבחר לריצה והמערכת מאפשרת לו לרוץ על מעבד מסוים, הוא מוגבל בזמן. זמנו של החוט קצוב אפילו אם החוט מוכן להמשיך לרוץ לנצח ולא מתרחשים אירועים מעניינים אחרים במערכת הגורמים לתזמון. לאחר שזמן זה פג, המתזמן מקבל מחדש את ההחלטה על החוט שירוץ ביחידת הזמן הבאה.

משך הזמן שניתן לחוט (הנקרא Quantum) קשור בשני גורמים. הגורם הראשון הוא קצב השעון המחובר למערכת (אין מדובר על השעון של המעבד, אלא על רכיב חומרה חיצוני היושב על לוח האם). השעון מייצר פסיקות במרווחי זמן קבועים, והמתזמן יכול להשתמש בפסיקות האלה כדי להחליט מתי פגה יחידת הזמן שהוקצבה לחוט מסוים. קצב השעון נמדד בדרך כלל במילישניות - 15 מילישניות הן ערך נפוץ[3].



(בדיקת קצב השעון של המערכת באמצעות הכלי Clockres.exe של Sysinternals)

הגורם השני המשפיע על ה-Quantum הוא הגדרה שנמצאת בידי המשתמש. Quantum קצר גורם לכך שהחלפה בין חוטים מתרחשת לעתים קרובות יותר, מה שיכול לשפר את התגובתיות של המערכת - כל חוט מקבל הזדמנות לרוץ לעתים קרובות יותר, מה שמתאים לעתים קרובות לתוכניות המבצעות אינטראקציה עם המשתמש. לעומת זאת, Quantum ארוך מגדיל את הסיכוי שחוט יסיים את עבודתו (או לפחות חלק ניכר ממנה) לפני שיבקשו ממנו לפנות את המעבד, מה שמתאים לעתים קרובות לתוכניות המבצעות עבודה ברקע. לא תמיד ברור מה האפשרות העדיפה, ולכן ניתן לשלוט עליה באמצעות ה-Registry או הגדרות "המחשב שלי":



(הגדרה המשפיעה על ה-Quantum. האפשרות המסומנת משמעותה Quantum קצר יותר)

[3] בהרבה מקרים, יש אפשרות להשפיע על קצב השעון. למשל, בתצלום המסך להלן, קצב השעון הנוכחי הוא מילישניה אחת, אולם הקצב המקסימלי הוא 15.6 מילישניות והקצב המינימלי הוא 0.5 מילישניות (500 מיקרושניות). שינוי קצב השעון בזמן ריצה גורם לכך שמנגנוני תזמון כמו Timers יכולים להיות מדויקים יותר, אבל הוא לא משפיע על תדירות הריצה של מתזמן החוטים, שמסתמך על נתון ה"מקסימום" בתצלום המסך לעיל.

כיצד ההגדרות הנ"ל מתורגמות ליחידות זמן? הבחירה ב-"Programs" גורמת ל-Quantum להיות באורך כפול מתדירות השעון, למשל 31.2 מילישניות אם תדירות השעון היא 15.6 מילישניות. לעומת זאת, הבחירה ב-"Background services" גורמת ל-Quantum להיות כפולת-12 של תדירות השעון, למשל 187.2 מילישניות אם תדירות השעון היא 15.6 מילישניות.

גם כאן אין מנוס מדקויות: ראשית, לא כל חוט יוכל לרוץ בדיוק את משך הזמן שהוקצב לו מלכתחילה, שכן במהלך ריצתו יכול להיווצר (או להתעורר) לפתע חוט חדש, חשוב יותר, והמתזמן יחליף אותו מיד. שנית, ייתכן שבסוף ה-Quantum, המתזמן יחליט להמשיך להריץ את החוט, מאחר שהוא עדיין החוט החשוב ביותר המוכן לריצה. שלישית, ייתכן שבמהלך ריצת החוט מתרחשות פסיקות רבות הגוזלות מזמן הביצוע של החוט, ובמקרה כזה מערכת ההפעלה תתחשב בכך ותאריך את משך הזמן המוקצב לו[4].

שינויי עדיפויות דינאמיים

מחד, השימוש בעדיפויות מאפשר לתוכניות לתעדף משימות חשובות יותר ופחות, ולמערכת ההפעלה להחליט בקלות איזה חוט להריץ בכל רגע. מאידך, שימוש עיוור בעדיפויות מוביל לבעיות של אי-הוגנות, כאשר הנפוצות ביותר מביניהן הן הרעבה והיפוך עדיפויות.

הרעבה היא מצב שבו חוט לא מקבל אפשרות לרוץ מכיוון שכל זמן שהוא מוכן לריצה, יש חוט חשוב יותר שמוכן גם הוא לריצה. היפוך עדיפויות הוא מצב עדין יותר, שבו משתתפים לפחות שלושה חוטים. נניח ש-A, B, C הם חוטים בעלי עדיפויות 1, 2, 3 בהתאמה. נניח גם שהחוטים A, C משתפים ביניהם משאב, ולכן משתמשים במנגנון סנכרון M למניעה הדדית. כעת ייתכן המצב הבא: החוט A מתחיל לרוץ, ותופס את ה"מנעול" M. מיד לאחר מכן החוט C מתעורר, מה שגורם למתזמן החוטים לסלק את A מהמעבד ולהחליפו ב-C. החוט C זקוק גם הוא ל-M, ולכן הוא נכנס למצב המתנה. כעת מתעורר החוט B ומתחיל לרוץ - הוא חשוב יותר מ-A, ואם כעת הוא מוכן לריצה למשך זמן רב, החוט A מורעב. אבל לא זו בלבד ש-A מורעב ולא מקבל הזדמנות לרוץ, אלא שבגלל M גם C לא מקבל הזדמנות לרוץ (כיוון שהוא זקוק ל-M ש-A מחזיק). נוצר היפוך עדיפויות - חוט בעדיפות 2 רץ בעוד שיש חוט בעדיפות 3 שהיה יכול לרוץ, אבל אינו מוכן לריצה באותו רגע.

על מנת לטפל במצבים מעין אלה Windows מבצעת שינויי עדיפות דינאמיים של חוטים במהלך ריצתם. רק חוטים בעלי עדיפויות 1-15 נהנים משינויי עדיפות דינאמיים אלה, וכל שינויי העדיפות הם זמניים,

[4] התחשבות כזו קיימת רק בגרסת NT 6.0 ומעלה, כלומר החל מ-Windows Vista.

ובדרך כלל דועכים ברמת עדיפות אחת מדי Quantum עד חזרתם לרמה המקורית. כמו כן, כל השינויים הם כללי אצבע - כלומר, המערכת לא יכולה לדעת בוודאות ששינוי העדיפות ישפר את התגובתיות, ההוגנות, או פרמטרים חיוביים אחרים. התקווה היא שבמרבית המקרים, שינויי העדיפות לא מזיקים, ולפעמים יכולים גם להועיל.

קצרה היריעה מכדי למנות את כל שינויי העדיפות הדינאמיים ש-Windows מבצעת, ולכן נתבונן רק בשלושה מהם:

מהות השינוי	גודל השינוי	הסבר
העלאת עדיפות לאחר חזרה מהמתנה על אובייקט סנכרון	1 או 2 רמות עדיפות	התייחסות לחוט שמבצע המתנה בתור חוט ה"מוותר" בחביבות על ה-Quantum שלו
העלאת עדיפות לאחר סיום פעולת קלט/פלט	כתלות במנהל ההתקן (Driver) כאשר הוא קורא לפונקציה `IoCompleteRequest`	לאפשר לחוט שחיכה לסיום פעולת קלט/פלט לעבד את התוצאות של הפעולה
העלאת עדיפות והארכת ה-Quantum לחוט שלא רץ במשך מספר שניות, ומוכן לריצה	העלאה לרמת עדיפות 15 וריצה במשך שני Quantum-ים	התמודדות עם הרעבה והיפוך עדיפויות

(שלוש דוגמאות לשינויי עדיפות דינאמיים ש-Windows מבצעת במקרים שונים)

Yin and Yang: בחירת מעבד פנוי להרצת חוט ובחירת חוט להרצה על מעבד פנוי

כאשר מערכת ההפעלה צריכה לבחור מעבד להריץ עליו חוט מסוים (למשל, חוט שנוצר כרגע או שהתעורר מהמתנה), היא צריכה להתחשב לא רק בעדיפות של החוט, אלא גם באיכויות היחסיות של המעבד ביחס לחוט. למשל, המערכת צריכה להעדיף את המעבד שעליו החוט רץ לאחרונה, שהרי שם יש עוד סיכוי למצוא במטמון את המידע שהחוט יצטרך בריצה העתידית. אולם גם החוט יכול לקבוע את העדפותיו - באמצעות פונקציות כגון `SetThreadAffinity` חוטים יכולים לקבוע מעבד או קבוצת מעבדים שרק עליהם יוכלו לרוץ[5]. באופן דומה, כאשר מתפנה מעבד (למשל, כיוון שחוט מסתיים או נכנס להמתנה), ויש לבחור חוט חדש להריץ עליו במקום החוט שהוסר ממנו, מערכת ההפעלה תעדיף חוטים שרצו על המעבד לאחרונה.

[5] מבלי להיכנס לפרטים שאין מקום עבורם במאמר זה, ישנה בעייתיות גדולה בקביעת העדפת מעבדים בצורה נוקשה. למשל, ייתכנו מצבים שבהם קביעת העדפה כזאת תגרום לחוט מסוים להיות מורעב, כיוון שהמעבדים המועדפים עליו תפוסים כל העת על ידי חוטים חשובים אפילו יותר. הדבר נכון עוד יותר כאשר התוכנית צריכה לרוץ על מחשבים בעלי מספר שונה של מעבדים ובעלי ארכיטקטורת חומרה שונה – כי אז קשה מאוד לנחש מראש מהו תעדוף המעבדים המיטבי עבור התוכנית, ולעתים קרובות עדיף להשאיר את ההחלטות למערכת ההפעלה.

כיצד המערכת מתאימה את עצמה לשינויי חומרה

Windows נאלצת להתאים את עצמה לשינויי חומרה המתרחשים בשנים האחרונות. שינויים במעבדים ובמבנה הזיכרון גורמים למתזמן החוטים לקבל החלטות בצורה מורכבת יותר. נביא שלוש דוגמאות מייצגות למצבים כאלה.

מזה כעשור, חברות שונות ובראשן אינטל מייצרות מעבדים בעלי יכולת HyperThreading, כלומר סימולציה של מספר מעבדים לוגיים (בדרך כלל 2) על מעבד פיזי אחד. הדבר נעשה על ידי שכפול האוגרים במעבד השומרים את מצב הביצוע, כגון EIP, ESP שראינו קודם, אך ללא שכפול משאבי החישוב, כמו היחידה האריתמטית-לוגית (ALU). כאשר המעבד אינו משתמש במשאבי החישוב וממתין - למשל כתוצאה מגישה לזיכרון - משתמשים במשאבי החישוב להרצה של חוט אחר. שיפור הביצועים המתקבל לעומת מעבד לוגי אחד על מעבד פיזי אחד הוא בדרך כלל קטן יחסית, ואפילו בתחזיות האופטימיות ביותר מדובר על שיפור של 30% בלבד.

מערכת ההפעלה צריכה להתחשב ב-HyperThreading במהלך תזמון חוטים. למשל, כיוון שהמטמון משותף במילא לכל המעבדים הלוגיים על אותו מעבד פיזי, אין יתרון להעדפת מעבד לוגי מסוים כיוון שהחוט רץ עליו לאחרונה. כלומר, כאשר בוחרים חוט עבור מעבד לוגי פנוי, יש להסתכל גם על חוטים שרצו לאחרונה על מעבדים לוגיים אחרים על אותו המעבד הפיזי. באופן דומה, המערכת תעדיף לשבץ חוט על מעבד לוגי שהמעבד הפיזי שלו פנוי לגמרי על פני מעבד לוגי שהמעבד הפיזי שלו כבר מריץ חוט אחר (במעבד לוגי אחר).

דוגמה אחרת לשינויי ארכיטקטורה שמכתיבים שינויים במתזמן החוטים היא NUMA, ארכיטקטורת חומרה שבה המעבדים והזיכרון מסודרים בקבוצות הנקראות Nodes. בכל קבוצה נמצאים חלק מהמעבדים וחלק מהזיכרון. למשל, במערכת בעלת 8 מעבדים ו-24 ג'יגה-בתים של זיכרון, יכולות להיות ארבע קבוצות, כאשר בכל קבוצה 2 מעבדים ו-6 ג'יגה-בתים של זיכרון. הסיבה לחלוקה לקבוצות היא הנדסית בעיקרה: ככל שמוסיפים יותר מעבדים ויותר זיכרון למערכת, כך קשה יותר להחזיק את כל המעבדים במרחק זהה (וקטן) מהזיכרון. מכאן שמעבד הניגש לזיכרון בתוך הקבוצה שלו עושה זאת במהירות מירבית, ואילו גישה לזיכרון בקבוצה אחרת היא איטית בהרבה (לעתים פי שניים או פי שלושה).

כמובן, גם במקרה זה מערכת ההפעלה מוכרחה להתחשב ב-NUMA כאשר מתזמן החוטים עושה את עבודתו. למשל, המתזמן ייצא מגדרו על מנת שחוט לא ינדוד בין מעבדים הנמצאים בקבוצות שונות. כמו

כן, המערכת צריכה לדאוג שהקצאות זיכרון המתבצעות על ידי חוט מסופקות מהזיכרון הפיזי הנמצא בתוך הקבוצה של המעבד עליו הוא רץ[6].

לבסוף, בשרתים מודרניים ישנה היכולת לכבות חלק מהמעבדים לחלוטין בזמן שלא נעשה בהם שימוש. יכולת זו נקראת Core Parking והיא מאפשרת חיסכון משמעותי באנרגיה. גם כאן, מערכת ההפעלה נדרשת להתחשב במעבדים כבויים - היא תשבץ חוטים קודם במעבדים פנויים שאינם כבויים, לפני שהיא מדליקה מעבד כבוי.

סיכום

תזמון חוטים הוא משימה מורכבת בהרבה מכפי שנראה אולי בתחילה, במערכות בעלות מעבד אחד ומספר מצומצם של "טריקים" לשינויי עדיפות ומשך זמן עבור ריצת החוטים. בימינו, מערכות ההפעלה השונות מתחרות הן במישור הביצועים והן במישור צריכת החשמל, וכמובן הגידול המעריכי במספר המעבדים אינו מקל את עבודת מתזמן החוטים.

ימים יגידו האם Windows ומערכות הפעלה אחרות הבנויות סביב מודל תזמון החוטים הקלאסי שהוצג כאן יוכלו להתמודד, כפי שהן, עם אלפי או עשרות אלפי מעבדים. בינתיים, Windows מסוגלת לתזמן עשרות אלפי חוטים על 256 מעבדים בהצלחה - ובמורכבות - גדולה.

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 ממחברי הספר Windows 7 for Developers, ובין היתר מלמד במכללת סלע קורסים בנושא Windows Internals. בזמנו הפנוי, סשה כותב בלוג על נושאי פיתוח שונים.

[6] התמיכה המפורשת של Windows ב-NUMA החלה בתקופת Windows Vista והשתפרה עוד ב-Windows 7.

איך לעקוב אחרי גולשים בעזרת עוגיות חסינות מחיקה

מאת: שלמה יונה (shlomo.yona@gmail.com | http://shlomoyona.blogspot.com)

הקדמה

אתרי אינטרנט יכולים לספק שירות טוב יותר ולהרוויח יותר באמצעות איסוף מידע על הגלישה בהם. מידע כזה שימושי במיוחד כאשר הוא מעובד באמצעי ניתוח שונים, שמאפשרים תצוגה חזותית נוחה של המידע, בחינת השערות והסקת מסקנות. דוגמה לשירות כזה הוא ה-Google Analytics. מתברר שישנו עוד מידע רב ערך שמאפשר להבין יותר ולהעלות יותר את השירות ואת הרווחים. מידע זה הוא מידע על אודות הגולשים גם מחוץ לאתר עצמו. אם יודעים מה גולשים עושים גם באתר וגם מחוץ לאתר, אפשר להסיק מסקנות נרחבות יותר, ולהשתמש במסקנות אלה כדי להציע הצעות שתהיינה מושכות יותר לגולשים, ושתעלינה את ההסתברות לרכישה, וכך יגדילו את הסכום שבפועל מקבלים בממוצע מרכישות באתר.

KissMetrics היא חברה מסוג כזה שמוכרת מידע על גלישה לאתרים. את המידע היא אוספת באופנים שונים. בנוסף למידע, החברה מוכרת גם כלים לניתוחו ולהסקת מסקנות מהמידע. הדרכים להשיג מידע על גולשים באתר ידועות בדרך כלל, וביניהן, מידע שנאסף מתוך בקשות HTTP (למשל, על ידי רישומי logs של שרת האינטרנט), מידע מתוך מסדי הנתונים וגם עוגיות בצד הלקוח. העוגיות הן מנגנון מובנה ב-HTTP, ודפדפנים תומכים בו. זאת שיטה לאפשר שמירת מידע בצד הלקוח. זה אחד המנגנונים לשמירת מצב הגולש מול אתר בעת גלישה. מנגנון לשמירת מצב הוא הכרחי, מפני שפרוטוקול HTTP הוא פרוטוקול שאינו שומר מצב. אפשר גם לשמור את מצב הגלישה בצד השרת. ישנה נוחות בשמירת מצב בצד הלקוח, משום שהמידע מגיע יחד עם הבקשות מהלקוח, וקל לנהל בקשות אלו, ואין הכרח לאחסן אותן. יחד עם זאת, העוגיות מאפשרות מעקב אחרי הגולש, שמירת ההיסטוריה שלו באתר (מה עשה, מתי עשה וכו'). לכאורה, מנגנון ה-same origin policy אמור למנוע מעוגיות שמנוהלות בשם מתחם מסוים להיות בשימוש בשם מתחם אחר. את זה עוקפות חברות באמצעות חברות צד ג' שיש להן קוד שמובנה בדפים באתר. הקוד מאפשר לשלוח מידע על גולש משם מתחם מסוים לשם מתחם אחר. למשל, חברת פרסום שיש לה קוד באתרים רבים מקבלת מידע על גולשים, אותו היא שומרת בעוגיות תחת שם המתחם שלה, וכך יכולה לאסוף מידע על גולש בין אתרים שונים וגם בין שמות מתחם שונים.

בימינו גולשים רבים מודעים לכך, ולכן מוחקים עוגיות. בנוסף, קיימים כלים נחמדים שמאפשרים לדעת בקלות אילו עוגיות צד ג' מותקנות בכל עמוד שגולשים אליו, ולקבל מידע על החברות ועל השירותים שמאחורי העוגיות, ואפילו לחסום אותן. כלי אחד כזה הוא Ghostery. כלי אחר, שמאפשר גם לראות גרף קשרים בין חברות צד ג' שמרגלות אחרינו, הוא Collusion.

ישנן יוזמות חדשות שעוסקות ב-opt out של משתמשים לפי רצונם, למשל, do no track - זהו HTTP header שמנסים לתמוך בו באופן אחיד בדפדפנים כדי למנוע מחברות ומשרותי צד ג' לעקוב אחר הפעילות.

עקב המודעות אליהן, החברות שמתפרנסות ממעקב אחר משתמשים הבינו שעליהן להשתכלל. אחד השכלולים של חברות צד ג' להמשיך ולעקוב אחרי משתמשים שמוחקים את קבצי העוגיות שלהם היה לאחסן גיבוי של העוגיות בתור עוגיות פלאש, Flash Cookies, אשר עד לא מכבר (ובחלק מהדפדפנים גם היום) יש צורך בפעולה נפרדת כדי למחוק אותן. מה שקורה הוא שמשתמש שמגיע לאתר ואין לו עוגיה מתאימה, זוכה לחיפוש עוגיית הפלאש המתאימה, ואם המידע נמצא בעוגיית הפלאש הוא משוחזר לעוגייה. כאשר שומרים מחדש מידע בעוגייה אז מגבים אותה ליתר בטחון גם בעוגיית הפלאש. המחקר שחשף את השימוש בעוגיות פלאש לצורכי שרידות העקיבה זמין ב:
http://ssrn.com/abstract=1446862

דרכים נוספות להמשיך ולשמור מצב (ולאפשר מעקב...) בצד הלקוח הן באמצעות שיטות אחסון אחרות ומודרניות יותר בצד הלקוח, כמו למשל ה-WebStorage ב-HTML5 שמאפשר באופן מוסדר (עדיין לא כל הדפדפנים מממשים את זה במלואו) לאחסן בדפדפן זוגות של מפתח-ערך, ואפילו מסד נתונים שלם. בגרסאות ישנות יותר של דפדפנים יש גם SessionStorage ו-localStorage (בפיירפוקס) ו-UserData (באקספלורר). לאור עליית המודעות אצל הגולשים, גרסאות חדשות של דפדפנים מאפשרות בקליק יחיד למחוק, אם רוצים, את כל המידע שנאסף על המשתמש מכל ההתקנים הללו לפי דומיין, ולפי קריטריונים נוספים.

המצב הזה נחזה מראש וחברות המעקב שרוצות להתפרנס לא הפסיקו לחפש דרכים נוספות כדי לגבות את המידע ולאפשר המשך המעקב.

בסוף יולי 2011 פורסם מחקר: מחיקת עוגיות, מחיקת עוגיות פלאש, אפילו נטרול הפלאש לחלוטין ונטרול אמצעי אחסון בצד הלקוח לא יועילו - גם לא ביטול השימוש ב-JavaScript ואף לא גלישה במצב אנונימי (incognito) בדפדפן.

אז איך זה עובד?

בפרוטוקול HTTP החל מגרסה 1.1 יש מנגנון שמאפשר חסכון בהעברת הודעות באמצעות מטמון, cache. מנגנון זה מכונה ETag. הדרך המקובלת להשתמש ב-ETag היא לעדכן בשדה הערך של כותרת ה-ETag בבקשת HTTP את גרסת המשאב (תוכן, תמונה וכדומה) כדי לאפשר למכונות ולשירותים בדרך (לרבות אפילו השרת של השירות שאליו גולשים והדפדפן של הגולש, בפרשנות רחבה של משמעות השדה הזה) להחזיר גרסה קיימת של המשאב אם זו קיימת אצלם (ולחסוך את הזמן ואת התקשורת כדי לבקש מהשרת שוב ולקבל ולהחזיר אותו ללקוח).

מתי זה שימושי? זה שימושי כאשר יש תוכן ותמונות שחוזרים על עצמם בין דפים, ואז אפשר לקבל אותם מהמטמון ולא לשוב ולבקש ולשנע אותם באינטרנט בכל פעם מחדש.

עדכון של גרסת המשאב בכותרות מאפשר לכל הצדדים שמעוניינים לתמוך במטמון לשתף פעולה ולחסוך. המשאב יובא מחדש רק כאשר השרת ידווח שהגרסה השתנתה או אם יש גורם בדרך שמוחק או משנה באופן שאינו נתמך את המידע על הגרסה. ה-ETag מאפשר לא רק לשלוט בגרסה אלא גם לזהות באופן ייחודי את המשאב. אם למשל שומרים על ערך ערבול, hash, שמייצג את המשאב, למשל באמצעות CRC או md5, אז אפשר בהסתברות גבוהה לייצג משאב ללא התנגשויות ולנהל גרסאות מטמון עבורו.

חברת KissMetrics השתמשה בשדה הזה לשימוש אחר. היא אחסנה בו העתק של ייצוג המידע שמאפשר לייצר מחדש את המידע בצד הלקוח (למשל עוגיות). בפרט, אחסון של ערך העוגייה בשדה ה-ETag. פרסום המאמר שהזכיר שני אתרים ידועים, hulu ו-foxnews גרר דיון ציבורי, תביעות בנושא פרטיות ואלה גררו את פרסום המאמר הזה.

נפרט על מנגנון ה-ETag. ראשית נראה כיצד משתמשים בו כראוי, ואז נראה מהו השימוש הלא שגרתי לצורך גיבוי העוגיות. מנגנון ה-ETag, כאמור, מאפשר לבדוק האם אפשר לצרוך משאב מסויים מתוך מטמון קיים או שיש צורך להביא אותו מחדש מהמקור. רעיון המטמון פשוט. חשבו לעצמכם שאתם יושבים לכם במשרד ובמטבח יש חטיף שמתחשק לכם לנשנש ממנו. אתם יכולים לגשת למטבח, לקחת יחידה אחת, לנשנש ולשוב למקומכם. אם תרצו עוד, תצטרכו לקום, לגשת למטבח לקחת עוד לנשנש וחוזר חלילה. מה עושים ילדים קטנים במסיבות יום הולדת כדי לפתור לעצמם את הבעיה? הם יוצרים לעצמם מטמון מקומי: לוקחים כוס חד פעמית, ממלאים אותה בחטיפים ואז ניגשים למקומם שם הם זוללים. כך הם יכולים לחסוך לעצמם את הגישה שוב ושוב לשולחן החטיפים, אולי אפילו חוסכים את העמידה בתור ומרוויחים זמן וצריכה נוחה מהמקום בנוחות. למען האמת, משל החטיפים מתאר מנגנון אגירה, aggregation, או חציצה, buffering. הרעיון אומר, שעדיף לאסוף בפעם אחת כמה מהדברים שאנו

צריכים להעביר ממקום למקום. באופן זה, אפשר לחסוך מסע הלוך ושוב כמספר הפריטים. האילוץ הוא הזמן שיש לנו בכל פעם לאסוף פריטים, וכמובן מגבלת כמות המטען שאנו יכולים לשאת בכל מסע. מטמון במערכות מחשב הוא אפילו מתוחכם יותר. הוא משלב אגירה שכזאת ביחד עם יצירה של עותק מקומי: להבדיל מחטיף שנצרך ואוזל, כאשר מדובר במשאב כמו טקסט גדול, תמונה, סרטון או מבנה נתונים גדול, שמירה על העתק זמין במטמון מקומי טוב לשימוש חוזר לנצח. או עד אשר המשאב מתעדכן או משתנה. חשבו למשל על רשומה ענקית של נתונים. כל עוד העותק המקומי שלנו נכון אז אין בעיה להשתמש בו שוב ושוב. אולם משעה שהרשומה מתעדכנת במקור, היינו רוצים להיות מסוגלים למשוך העתק חדש, או רצוי רק עדכון של השינויים.

מנגנון ה-ETag מאפשר לעשות את הדברים האלה בכך שהוא מעביר מידע על העדכניות של המשאב שבמטמון הדפדפן. כך הדפדפן מביא משאבים רק אם אין עותק עדכני שלהם ברשותו. מי מחליט האם העותק עדכני? מה רושמים ב-ETag? השרת רושם מזהה ייחודי לכל משאב שאפשר למצוא באמצעות URL. בכל פעם שישתנה המשאב בכתובת השרת יעדכן את המזהה. דפדפן שצריך תמונה, למשל, כדי להציגה בעמוד, בודק אם יש ETag. אם יש ETag, האם הערך שמתאים ל-URL זהה לערך שנשמר בפעם הקודמת? אם כן, אז אין צורך להביא את המשאב מחדש. אחרת, הדפדפן מביא את המשאב מחדש. הערך שמשתמשים בו כחתימה או כטביעת אצבע לזיהוי הגרסה נשמר כמחרוזת טקסט בתווים המותרים בערכים של כותרות ב-HTTP.

לפניכם דוגמה טיפוסית: בקשת GET ראשונה להצגת התמונת הלוגו של DigitalWhisper תראה כך:

```
GET /logo.png HTTP/1.1
Host: digitalwhisper.co.il
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US)
AppleWebKit/533.4 (KHTML, like Gecko) Chrome/5.0.375.70 Safari/533.4
Cache-Control: max-age=0
Accept:
application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,i
mage/png,*/*;q=0.5
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,vi;q=0.6
Accept-Charset: windows-1255,utf-8;q=0.7,*;q=0.3
```

התגובה שנקבל מהשרת בתמורה תראה כך:

```
HTTP/1.1 200 OK
Date: Thu, 25 Aug 2011 21:32:18 GMT
Server: Apache/2.2.9 (Unix) mod_ssl/2.2.9 OpenSSL/0.9.8e-fips-rhel5
PHP/5.2.6
Last-Modified: Wed, 06 Jul 2011 20:59:07 GMT
ETag: "25cd3c-33ab5-4a76cdf081cc0"
Accept-Ranges: bytes
Content-Length: 211637
```

```
Content-Type: image/png

[PNG DATA]…
```

תוכן התמונה חזר ואיתו ה-ETag:

```
ETag: "25cd3c-33ab5-4a76cdf081cc0"
```

במידה ונבצע "ריענון" לעמוד / או שנגיע לעמוד אשר טוען את אותה התמונה, הדפדפן ישלח את בקשת ה-GET באופן הבא:

```
GET /logo.png HTTP/1.1
Host: digitalwhisper.co.il
Proxy-Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US)
AppleWebKit/533.4 (KHTML, like Gecko) Chrome/5.0.375.70 Safari/533.4
Cache-Control: max-age=0
Accept:
application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,i
mage/png,*/*;q=0.5
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,vi;q=0.6
Accept-Charset: windows-1255,utf-8;q=0.7,*;q=0.3
If-None-Match: "25cd3c-33ab5-4a76cdf081cc0"
If-Modified-Since: Wed, 06 Jul 2011 20:59:07 GMT
```

והתגובה שנקבל בתמורה תראה כך:

```
HTTP/1.1 304 Not Modified
Date: Thu, 25 Aug 2011 21:39:02 GMT
Server: Apache/2.2.9 (Unix) mod_ssl/2.2.9 OpenSSL/0.9.8e-fips-rhel5
PHP/5.2.6
ETag: "25cd3c-33ab5-4a76cdf081cc0"
```

כך הדפדפן יודע שאותו המשאב (תמונת הלוגו) השמורה אצלו ב-Cache עדיין רלוונטית וניתן להציג אותה ללא חשש שהיא שונתה.

איך מתבצע גיבוי העוגיות באמצעי הזה? השרת מניח שהדפדפן יכבד כותרות ETag, וישתמש בערך שלהן בבקשות עתידיות באמצעות If-None-Match. המשמעות היא שהדפדפן שומר על הערך הזה אצלו וחוזר ומשדר אותו לשרת שוב ושוב. זה בדיוק המנגנון הרצוי. עכשיו השרת לא רק מדביק עוגיה עם המידע שנצבר על המשתמש, אלא גם שומר על עותק שלה בשדה ה-ETag. אם השיטה הזאת חוזרת על עצמה בכל עמוד ועמוד באתר, אז גם אם הדפדפן מוחק את העוגיה מטעמי פרטיות, אין לו סיבה למחוק את

ערכי ה-ETag שנשמרו מסיבות של חיסכון בתקשורת. כך אם השרת אינו מוצא עוגיה בבקשות מדפדפן, אבל מוצא ערך שמתאים לפורמט העוגייה שלו בשדה If-None-Match, אז הוא משחזר מתוך כך את העוגייה, מעדכן אותה, מדביק אותה מחדש (או שלא...) בתגובה הבאה, ובכל מקרה מעדכן עותק שלה ב-ETag.

התגוננות

איך אפשר לבדוק אם יש אצלנו דבר כזה? בודקים עוגיות למיניהן ואמצעי אחסון צד לקוח - או שמוחקים - ומעתה גם מנסים להסיר שדות של כותרי HTTP שאיננו מכירים ושאיננו יודעים שהם משרתים מטרה מועילה בתקשורת שלנו מול אתרים, למשל ה-ETag.

זה סיפור לא פשוט למשתמש רגיל אבל משימה לא מסובכת לתוסף לדפדפן (התוסף Ghostery כבר לכאורה מאפשר חסימה של הטכנולוגיה הזאת של KissMetrics - אבל אל תאמינו עד שתראו שזה באמת עובד) או לפתרון אבטחה צד לקוח אחר (כמו מכונות קצה שמבצעות אבטחת מידע אפליקטיבית).

בינתיים, החברה נתבעה, וטענה שהיא הפסיקה את השימוש בטכנולוגיה. אולם חברות נוספות נמצאו משתמשות בתרגיל הזה, חלקן נתבעו ולא ברור מי או מה ימנע מהן או מכל גוף אחר להמשיך להשתמש בה לצורכי מעקב. כנראה שתוצאה ברורה היא שחברות מהוגנות יפסיקו להשתמש בטכנולוגיה כזאת.

כנראה שגופים אחרים לא בהכרח ירגישו מחויבים גם כן לעשות זאת. כנראה שהמירוץ בין נסיונות שונים להשתמש בטכנולוגיה כדי לממש פרטיות לבין שימוש בטכנולוגיה כדי לממש מעקב ימשיך בדיוק כמו שהמירוץ והמאבק בין הפורצים למיניהם ופתרונות האבטחה ממשיך. יש כאלה שחושבים שחקיקה תסייע. אני סבור שהיא לא תסייע כנראה יותר מאשר היא מסייעת כנגד פשעים אחרים, במחשב ושלא במחשב.

למתעניינים, אפשר לקרוא תיאור שווה לכל נפש של אחד מהחושפים של הטכנולוגיה כאן:

http://ashkansoltani.org/docs/respawn_redux.html

תשתית מפתחות ציבוריים

מאת: עמיחי פרץ קלופשטוק

הקדמה

פעמים רבות בעולם המחשבים ואבטחת המידע, נוצר צורך ברשת תקשורת מאובטחת ומהימנה, אשר העברת נתונים דרכה תהיה סודית ובטוחה מפני שינוי - זדוני או מקרי כאחד. ישנם פתרונות רבים לבעיה זו, כשהבולטים שבהם מגיעים מתחום הקריפטוגרפיה. במאמר זה אסקור את שיטת המפתחות הפומביים, כיצד היא עובדת, מהם היתרונות והחולשות שלה, ודרכים שונות לממש אותה כתשתית תקשורת.

הצפנה סימטרית ואסימטרית

בתורת הקריפטוגרפיה, קיימות שתי צורות הצפנה עיקריות: הצפנה סימטרית, בה מפתח ההצפנה זהה למפתח הפענוח, והצפנה אסימטרית, בה קיימים זוגות של מפתחות הצפנה תואמים זה לזה. עקב התכונות המתמטיות של המפתחות האסימטריים, הודעה שתוצפן באמצעות מפתח אחד תוכל להיות מפענחת רק באמצעות המפתח השני, ולהפך.

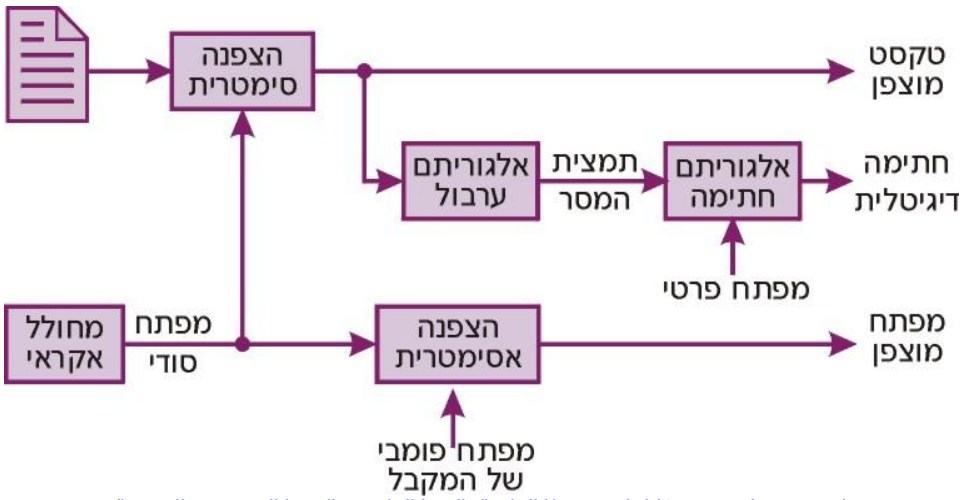
לכל אחת מן השיטות הנ"ל יתרונות וחסרונות, אך הסיבה העיקרית שבזכותה אנו מעדיפים להשתמש בהצפנה אסימטרית לתקשורת מסוג זה, היא שבשונה מהצפנה סימטרית, בה יש צורך להעביר את מפתח ההצפנה בצורה סודית לשני הצדדים שמשתתפים בתקשורת, בהצפנה אסימטרית אין צורך לשמור על סודיות המפתחות - למעשה, ניתן לפרסם אותם באופן פומבי באינטרנט.

כאשר משתמשים בהצפנת מפתח פומבי, לכל גורם ברשת התקשורת ישנם שני מפתחות. באמצעות מפתח אחד, שנהוג לכנותו מפתח ציבורי או פומבי (Public Key), ניתן להצפין הודעות אל האדם, אשר רק הוא יוכל לפענח. מפתח זה מופץ באופן חופשי לכל דורש, על גבי שרתי מפתחות יעודיים או בדרכים אחרות.

המפתח השני, הנו מפתח סודי, אשר רק לבעליו יש גישה אליו, ונקרא מפתח פרטי (Private Key). מפתח זה משמש את בעליו כדי לפענח הודעות שהוצפנו באמצעות המפתח הפומבי שלו, ולחתום על הודעות יוצאות.

לדוגמה, אברהם מעוניין להעביר הודעה סודית אל יצחק. ראשית אברהם יחתום על תמצית (Hash) של ההודעה המקורית, יצרף אותו אל ההודעה. כעת אברהם יצפין את כל המסר בצופן סימטרי, ואת המפתח יצפין באמצעות המפתח הציבורי של יצחק, וישלח את ההודעה המוצפנת+המפתח המוצפן אל יצחק דרך ערוץ תקשורת לא מאובטח.

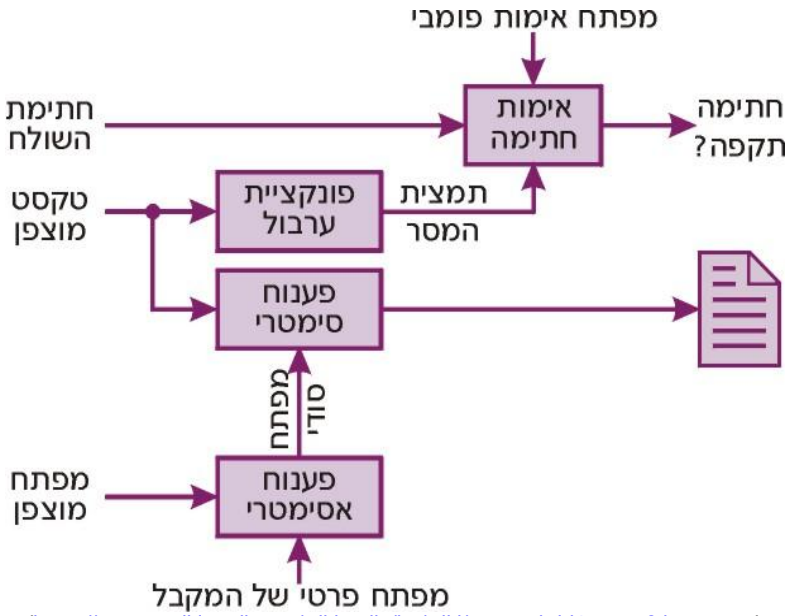
ניתן לראות את התהליך המלא בדוגמה הבאה:



(במקור: https://secure.wikimedia.org/wikipedia/he/wiki/קובץ:HybridSystem1.jpg)

כאשר יצחק יקבל את ההודעה, ראשית הוא יפענח אותה בעזרת המפתח הפרטי שלו. לאחר מכן, הוא יוודא שאכן ההודעה נשלחה על ידי אברהם, בכך שיפענח את התמצית המוצפנת שצורפה להודעה,

וישווה אותה מול תמצית שנוצרה באותו רגע מההודעה עצמה. כל שינוי קטן בהודעה יתגלה בצורה כזו באופן מיידי. תהליך זה מוצג באיור הבא:



(במקור: https://secure.wikimedia.org/wikipedia/he/wiki/קובץ:HybridSystem2.jpg)

כך, אברהם יודע שרק יצחק יהיה מסוגל לקרוא את ההודעה, משום שרק לו יש את המפתח הפרטי הנכון, ויצחק ידע שרק אברהם יכל לשלוח לו את ההודעה, כיוון שהיא חתומה עם המפתח הפרטי שלו.

חולשה פוטנציאלית

נתאר לעצמנו מצב כזה: יצחק חי בישראל, בזמן שאברהם חי בארצות הברית. יש להניח, שהם לא ייפגשו לעתים קרובות, ויתכן שהתקשורת שלהם מבוססת על מפתחות פומביים שנשלחו בדוא"ל או הופצו דרך שרת מפתחות.

מצב כזה עשוי לפתוח דלת למתקפת האדם שבאמצע (Man in the middle). בפשטות, מדובר על מצב בו גורם זדוני כלשהו יתחזה ליצחק כלפי אברהם, ולאברהם כלפי יצחק. אם הפושע גורם ליצחק לחשוב שהמפתח הציבורי שלו הוא המפתח הציבורי של אברהם, ולאברהם לחשוב שהמפתח הציבורי שלו הוא של יצחק, כל התקשורת ביניהם תהיה נתונה לחסדיו - הוא יוכל להאזין, לסנן, לזייף ולשנות הודעות, אשר יראו אמינות לחלוטין. למעשה, חוזקה של מתקפה זו הוא בכך שאין לקרבן כל סיבה לחשוד בפעילות זדונית, ובכך שהתוקף מקבל אמינות גבוהה יחסית.

חתימה על מפתחות

ישנן שיטות רבות להתגבר על בעיה זו, וכולן מבוססות על זיהוי פיזי של האדם, שיקושר למפתח ציבורי מסוים. בצורה כזאת, לגורם זדוני יהיה קשה בהרבה להתחזות לאדם אחר.

הבעיה, כמובן, היא הצורך בזיהוי אמין - זיהוי פנים אל פנים, באופן אישי או על ידי גוף שלישי אמין. כאשר מזהים את הגורם המבוקש, חותמים על המפתח הציבורי שלו (למעשה, על תמצית של המפתח), באמצעות המפתח הפרטי של הגורם המאשר.

קיימות שתי שיטות עיקריות לזיהוי כזה:

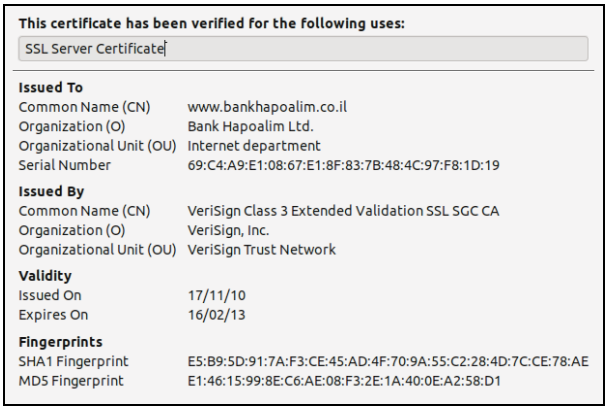
- שיטת הרשות המאשרת (Certificate Authority)
- שיטת רשת האמון(Web of trust)

רשות מאשרת

בשיטה זו, ישנן חברות אשר עוסקות בווידוא זהותם של גורמים, ובחתימה על מפתחותיהם. גורם אשר מבקש לקבל אישור כזה, צריך לאמת את זהותו על פי דרישות ונהלי החברה. רשויות מאשרות נפוצות בעיקר במגזר העסקי, בו יש צורך בזיהוי אמין מעל לכל ספק, ובתחום האינטרנט, לצורכי אימות של אתרים שאין באפשרות הגולשים לאמת בעצמם.

שיטה זו נפוצה לרוב באתרי אינטרנט גדולים, לצורך שימוש בפרוטוקול SSL. במקרים כאלו, האתר מאמת את זהותו באמצעות אישור אבטחה חתום על ידי רשות מאשרת, אשר המפתח הציבורי שלה מוטמע בדפדפן או מתקבל משרת מיוחד. בזכות האמון ברשות המאשרת (שנובע מגורמים רבים), ניתן לסמוך על זהותו של אתר האינטרנט.

האמון של המשתמשים הנו בחברות הללו בלבד, ואישורים הדדיים בין "משתמשים" אינם נפוצים, ואינם נחשבים לבטוחים. חברות מוכרות בתחום זה כוללות את VeriSign, Comodo, RSA, ועוד... ניתן לצפות באישורי האבטחה באמצעות הדפדפן - הנה, לדוגמה, אישור האבטחה של בנק הפועלים:


This certificate has been verified for the following uses:

SSL Server Certificate

Issued To	
Common Name (CN)	www.bankhapoalim.co.il
Organization (O)	Bank Hapoalim Ltd.
Organizational Unit (OU)	Internet department
Serial Number	69:C4:A9:E1:08:67:E1:8F:83:7B:48:4C:97:F8:1D:19
Issued By	
Common Name (CN)	VeriSign Class 3 Extended Validation SSL SGC CA
Organization (O)	VeriSign, Inc.
Organizational Unit (OU)	VeriSign Trust Network
Validity	
Issued On	17/11/10
Expires On	16/02/13
Fingerprints	
SHA1 Fingerprint	E5:B9:5D:91:7A:F3:CE:45:AD:4F:70:9A:55:C2:28:4D:7C:CE:78:AE
MD5 Fingerprint	E1:46:15:99:8E:C6:AE:08:F3:2E:1A:40:0E:A2:58:D1

רשת אמון

השיטה השנייה לבניית תשתית מפתחות ציבוריים מסתמכת על האמון שקיים בין אנשים באופן טבעי. בשיטה זו כל גורם ברשת משמש הן כרשות מאשרת והן כמשתמש. הרשת החברתית שנוצרת בצורה כזאת מבטיחה את אמינותם של המפתחות.

לדוגמה, אברהם רוצה להעביר ליצחק הודעה מוצפנת. אברהם, שמכיר את מתקפת האדם שבאמצע, מעוניין לוודא את אמינותו של המפתח שיצחק שלח לו. במקרה, חבר משותף של השניים, יעקב, נפגש עם יצחק, ומחזיק בעותק של המפתח המקורי. כאשר הוא נפגש עם אברהם, הוא נותן לו את המפתח האמין, וכעת לאברהם יש ערוץ תקשורת בטוח אל יצחק.

שיטה זו נקראת רשת אמון. למעשה, אברהם נתן אמון ביעקב, עקב ידידותם, וסמך עליו כגורם מאשר למפתח של יצחק. שיטת רשת האמון מתבססת על עיקרון זה, אך בקנה מידה גדול בהרבה. כאן, במקום להעביר את המפתח עצמו, המפתחות נמצאים על שרת מפתחות מרכזי, וכל אדם שווידא את זהותו והמפתח של אדם אחר, חותם על המפתח ומעלה את החתימה לשרת. כעת, אם אברהם ירצה לשלוח ליצחק מכתב, הוא יוכל להוריד את המפתח הציבורי שלו, ולראות את כל החתימות שבוצעו עליו. במקרה זה, כאשר יראה שאנשים אמינים (כמו יעקב, שרה, רחל ולאה) חתמו עליו, הוא יוכל לתת בו אמון מסוים.

בתורו, אברהם יחתום על מפתחות של אנשים אותם הוא מכיר, ואנשים שמכירים אותו יחתמו על המפתח שלו. ברשת אמון, אנו נותנים אמון במפתח גם עקב שרשרת של קשרים אישיים: אם אברהם חתם על המפתח של יצחק, ויצחק חתם על המפתח של יעקב, אברהם יוכל לסמוך על המפתח של יעקב.

צורה זו פחות בטוחה, כיוון שאין לנו אפשרות לדעת האם יצחק אכן בדק כראוי את זהותו של יעקב. לעומת זאת, אם יצחק, שרה ורחל חתמו על המפתח של יעקב, ואברהם סומך על כולם, ניתן יהיה לסמוך על המפתח של יצחק.

ובנושא זה רצוי להבחין בין שני סוגי אמון - אמון באדם כמשתמש ברשת, כלומר אמון שהוא אכן מי שהוא טוען שהוא, ושהמפתח אכן שלו, לבין אמון באדם כרשות מאשרת, שבשבילו צריך לבדוק האם האדם ראוי לאמון, ויבדוק היטב כל משתמש ומפתח ברשת לפני שיחתום עליו.

מסיבת חתימת מפתחות

כיוון שבחיי היום-יום רובנו לא מסתובבים עם עותקים של טביעת האצבע של המפתח הפומבי שלנו, ולא מכירים הרבה אנשים שמשתתפים ברשת אמון כזו או אחרת, אין לנו הרבה הזדמנויות לחתום על מפתחות ולשפר ולהרחיב את הרשת.

לשם כך נוצרו מסיבות חתימת מפתחות. מסיבת חתימת מפתחות הנה בעצם מפגש חברתי, בו מתרכזים אנשים (ובעיקר גיקים) לצורך חתימה על מפתחות ציבוריים. כל משתתף מקבל עוד לפני (או בתחילת) המסיבה רשימה של כל המשתתפים, וטביעות האצבע (Key fingerprint) של המפתחות שלהם.

במהלך המסיבה, כל משתתף אמור לזהות את עצמו בפני כל האחרים בעזרת תעודת זהות, רישיון נהיגה או דרכון (ולעתים אף באמצעות תעודת חוגר), ולהקריא את טביעת האצבע של המפתח שלו. שאר המשתתפים מוודאים את זהות האדם, ואת זהות המפתח שלו, ומסמנים זאת בטופס.

לאחר המסיבה, כל משתתף מגיע למחשב בטוח (בעיקר נטול רוגלות), ומוודא לפי הרשימה את טביעות האצבע של המפתחות שברשותו. אם טביעות האצבע תואמות, הוא חותם על המפתח ובכך מאשר את אמינותו. לאחר מכן, ניתן להעלות את המפתחות החתומים לשרת ציבורי.

סיכום

במאמר זה הצגתי את שיטת רשת האמון, ואת רעיון מסיבת חתימת המפתחות. למעשה, שימוש נכון ברשת אמון (ובתנאי שהיא בהיקף גדול מספיק) עשוי להוות צורת תקשורת מהימנה למרבית השימושים.

יתכן שלאחר קריאת המאמר התעורר בכם החשק להקים או להצטרף לרשת אמון. נהדר! בישראל ישנה רשת אמון ותיקה ומוערכת של קהילת הלינוקס והקוד הפתוח, אשר עורכת מסיבת חתימת מפתחות מדי שנה בכנס הקוד הפתוח "אוגוסט פינגווין", וחבריה ישמחו לחתום הדדית על מפתחות כמעט בכל עת, וכמובן שישנן (וניתן להקים) רשתות נוספות לאנשים עמם אתם בקשר.

שימו לב, שככל שיותר אנשים יהיו קשורים לרשת אמון כזו או אחרת, כך תגדל יעילותה של הרשת הגלובלית. רצוי מאוד ליצור קשרים בין רשתות קטנות, ובייחוד מול גורמים בחו"ל. כל קשר כזה מספק תקשורת מהימנה לאנשים רבים.

על המחבר

עמיחי פרץ קלופשטוק, בן 18, עוסק באבטחת מידע מזה כשנה וחצי, חובב לינוקס ותוכנה חופשית, ומתנדב בקבוצת dc9723.

טכניקות התרבות בקרב תולעים חברותיות

מאת: אפיק קסטיאל (cp77fk4r)

הקדמה

במסגרת מאמר זה אני מעוניין להציג סקירה על מספר מנגנוני התפשטות של תולעי-אינטרנט שונות שפקדו ופוקדות אותנו בעבר ובזמן האחרון. מטרת המאמר הינה להציג לכם, הקוראים, איך עולם תולעי-האינטרנט בכלל, ומנגנוני ההתפשטות שלהן בפרט הולך וצובר תאוצה בשנים האחרונות.

קיימים מספר רב של וקטורי הפצה בהם תולעי-אינטרנט מבצעות שימוש בכדי להגיע לתפוצה נרחבת, וקטורי הפצה כגון:

- ניצול חולשות במערכת ההפעלה.
- הדבקת התקני אחסון ניידים (USB).
- שליחת אימיילים (Mass-Mailing).
- הדבקת קבצי ברי-הרצה
- הנדסה חברתית.
- רשתות Peer 2 Peer.
- ועוד...

אך במאמר זה אני מעוניין לגעת רק בוקטורי הפצה בהם התולעים מבצעות שימוש בהנדסה חברתית. מצד אחד, הגיוני מאוד להניח כי מנגנוני הפצה אשר אינן דורשים אינטרקציה של המשתמש ומנצלים חולשות במערכת ההפעלה / דפדפן הם מנגנוני ההפצה הטובים ביותר. אך מצד שני- ברגע שהתולעת נתפסה, נחקרה והובן מהו כשל האבטחה בו היא עושה שימוש בכדי להתפשט, חיי התולעת יהיו קצרים ביותר. אותה החברה שאחראית על המנגנון בו נמצא כשל האבטחה תשחרר טלאי שיסגור אותו, וכאן בערך תמו חייה של התולעת.

מהסתכלות אחורה בזמן, נתן לראות כי דווקא התולעים שלא נצלו כשלי אבטחה אלא עשו שימוש בוקטורי התפשטות אשר כן דורשים אינטרקציה עם המשתמש ומבצעים שימוש בהנדסה חברתית- הן התולעים עם אורך החיים הגדול ביותר.

מדובר בשאלה לא פשוטה, והשאלה האם וקטורים אשר דורשים אינטרקציה עם המשתמש (" user interaction based") הם וקטורי ההפצה הטובים ביותר להפצת תולעים היא שאלה מעניינת בפני עצמה, אך היא אינה תעלה על הפרק במאמר הזה.

ולעניינו, כותב התולעת יכול להשתמש בהנדסה חברתית במספר פלטפורמות שונות, כגון:

- תוכנות מסרים מידיים (ICQ, MSN, Yahoo! Messenger, Skype, ערוצי IRC ועוד).
- משלוח אימיילים (דרך Outlook, שרתי SMTP שונים ו-Webmails)
- רשתות חברתיות (Facebook, Myspace וכו').
- ועוד...

במסגרת המאמר אציג מספר תולעים אשר מבצעות שימוש בפלטפורמות השונות בכדי שנוכל לבצע השוואה.

אני אוהב אותך אנה קורניקובה!

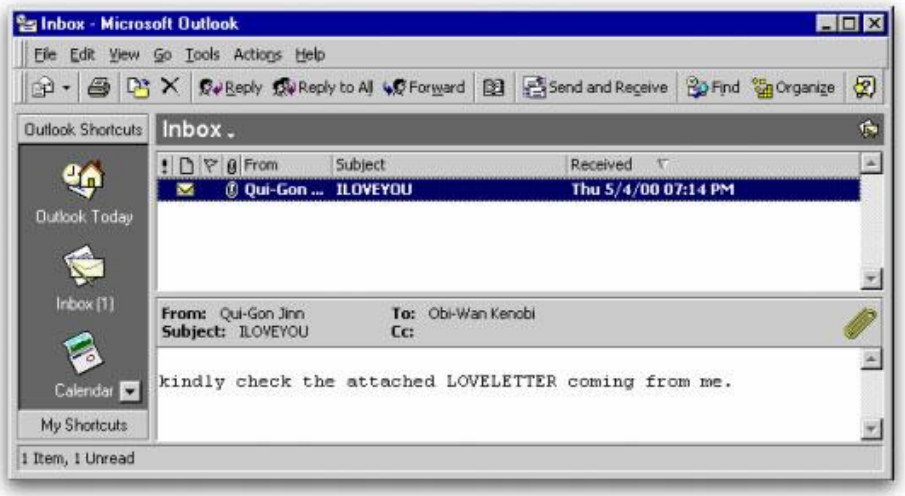
מדובר בשני תולעי-אינטרנט אגדיות (משנת 2000 ו-2001), הראשונה היא ה-"Love Bug" (מוכרת גם כ-"ILOVEYOU" ו-"Love Letter") והשניה היא כמובן התולעת: "VBS.SST@mm", או כמו שכל המדיה באותה התקופה כינתה אותה: "The Anna Kournikova Worm".

שתי התולעים אללו התפשטו בעזרת משלוח הודעות דוא"ל. שתי התולעים הפיצו את עצמן בעזרת התממשקות לתוכנת ה-Outlook שעל המחשב בו הן נפתחו ושלחו את עצמן כקובץ מצורף לאימייל אשר נשלח לכלל רשימת אנשי הקשר של הקורבן.

בשתי התולעים לא נוצלה אף חולשה בקוד של אף אחד מהמנגנונים שליוו את השליחה (כגון חולשות לוגיות או חולשות מבוססות זכרון כמו שנצפה בתולעים אחרות), אלא פשוט התממשקות לתוכנת ה-Outlook בעזרת שורות סקריפט לפקדי מאקרו של התוכנה (אובייקטים כגון Outlook.Application ב-WSH ו-VBS עושים את העבודה יופי).

התולעת "ILOVEYOU" הפיצה אימייל עם הכותרת "ILOVEYOU" ובתוכנו היה רשום המשפט:

```
Kindly check the attached LOVELETTER coming from me"
```



(במקור: http://iamjenessa.wordpress.com/2011/06/28/memories-of-the-love-bug-worm-by-graham-cluley-on-may-4-2009/)

לאותו האימייל היה מצורף קובץ- "מכתב אהבה" בשם: "LOVELETTER.TXT.VBS" או "LOVE-LETTER-FOR-YOU.TXT.VBS". הרעיון הוא שאז, במערכת ההפעלה Windows, ברירת המחדל הייתה לא להציג את סיומות הקבצים, כך שהקובץ הוצג כ-"LOVELETTER.TXT". כמובן שכאשר הקורבן היה לוחץ על הקובץ במחשבה שהוא הולך לקרוא מכתב אהבה- הוא בעצם הריץ את אותו קובץ VBS שהכיל את התולעת שהייתה מדביקה לו את המחשב ומפיצה עצמה לכלל רשימת אנשי הקשר שלו.

התולעת השניה, על שם שחקנית הטניס הרוסיה אנה קורניקובה, הרעיון הוא אותו רעיון. המימוש זהה כמעט לחלוטין, אך במקום לקבל מכתב אהבה, התולעת הפיצה את עצמה באימיילים שהתיימרו להכיל תמונה של השחקנית. נושא ההודעה היה: "Here you have, ;o)" והתוכן היה פשוט: "Check This!". שמו של הקובץ המצורף היה "AnnaKournikova.jpg.vbs", ושוב- עקב הגדרות ברירת המחדל של מערכת ההפעלה Windows, הסיומת ".vbs" הייתה נשמטת.

בכדי שהתולעת לא תרוץ על אותו משתמש מספר פעמים, לאחר הרצה ראשונה, היא הייתה כותבת ערך לעורך הרישום של מערכת ההפעלה, במיקום:

```
HKEY_CURRENT_USER\Software\OnTheFly\mailed
```

וכך, בכדי לא לשלוח לאותה רשימת אנשי קשר את עצמה שנית- הייתה מתבצעת בדיקה לפני כל שליחה- האם אותו מפתח קיים. במידה והיא הייתה מוצאת אותו- לא הייתה מתבצעת השליחה.

ושוב, כאשר הקורבן היה לוחץ על ה"תמונה", התולעת הייתה מורצת ושולחת את עצמה לכלל רשימת אנשי הקשר שלו.

(במקור: http://www.f-secure.com/v-descs/onthefly.shtml)

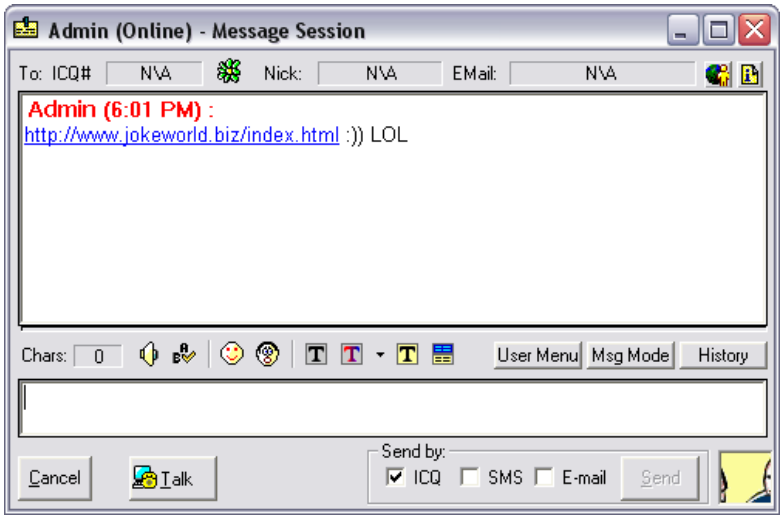
כשחושבים על שתי התולעים כיום, מפתיע לחשוב שהן הצליחו להתפשט לכל כך הרבה מחשבים. הרי לא נעשה כאן כמעט שימוש בהנדסה חברתית, ואפשר להצביע כאן על מספר נקודות די בעייתיות, כגון זה שהסיומת של הקובץ כאשר הוא מצורף למייל מופיע כ-"TXT.VBS" או "jpg.vbs", או זה שהאייקון של הקובץ נשאר כאייקון של קובץ סקריפט ולא מופיע האייקון של הקובץ שהוא מתיימר להיות.

בכדי להבין למה אותן התולעים הצליחו כל כך, אנו צריכים לזכור כי מדובר בתחילת שנות ה-2000, וכמו שכולנו יודעים, קשה להאשים את אותה התקופה במודעות יתר לסיכונים הכרוכים בגלישה באינטרנט. כיום, המודעות לאבטחת מידע ולסיכונים כגון אלו גבוהה יותר. ולכן לתולעים כגון אלו, כיום, כמעט ואין סיכוי ליצור את ההד אותו הן הצליחו ליצור בתחילת העשור הקודם.

סוג התולעים השני עליו נדבר היום הוא ה-"Instant Messenge Worms". מדובר בתולעים שיודעות להתממשק לפרוטוקול של תוכנות המסרים המידיים השונות (כגון ICQ או MSN Messenger) באופן כזה או אחר, ובעזרת גניבת פרטי ההזדהות של בעלי החשבונות השונים שלחו בשמם הודעות לחבריהם עם קישורים לקבצים / אתרים זדוניים אשר הכילו את התולעים עצמן.

לדוגמא, כאשר תולעים כגון "Bizex" או "Stration" היו מורצות במחשב אשר עליו מותקנת תוכנת המסרים המידיים "ICQ" הן היו שולפות את פרטי ההזדהות של המשתמש מאחד מקבצי ה-DAT של התוכנה (הקובץ בו נשמרים פרטי ההזדהות כאשר המשתמש מסמן "Remember Me" בעת ההזדהות לצורכי הזדהויות עתידיות), מפענחות אותם, ומשתמשות בהם בכדי להזדהות בשמו של המשתמש לשרת ה-ICQ ולהוריד את רשימת אנשי הקשר שלו. זאת כמובן בכדי לשלוח בשמו את עצמן לשאר לחבריו. בדרך כלל הן היו שולחות לחברי הקורבן (בשמו כמובן) קישור לאתר אשר דרש מהקורבן להתקין קובץ EXE במסווה של משחק קטן או שומר-מסך נחמד.

דוגמא להודעה שנשלחה על ידי התולעת "Bizex":



(במקור: http://www.kaspersky.co.in/news?id=4277566)

בגרסאות מתקדמות של התולעת, אף נעשה שימוש בחולשה שנמצאה במנוע לפענוח DHTML בדפדפן Internet Explorer (MS03-040) כך שמספיק שהמשתמש היה לוחץ על הלינק, ומבלי להוריד את קובץ Exe הוא היה נדבק.

ICQ אינה תוכנת המסרים המידיים היחידה שכותבי וירוסים ותולעים השתמשו ומשתמשים בה גם היום בתור פלטפורמה להפצת המוצרים שלהם. תולעים אחרות, כגון "Bropia" ו-"Yimfoca" (שמוטציות שונות שלה עדיין רצות בשטח) בחרו להפיץ עצמן דרך תוכנות כגון Msn Messenger ו-Yahoo! Messenger (בהתאמה). הפלטפורמה אולי שונה- אך הקונספט זהה לחלוטין. המשתמש מקבל הודעה ממכר / אדם רנדומלי ומתבקש ללחוץ על קישור באינטרנט. קישור זה בדרך כלל מפנה לקובץ Exe או במקרים אחרים - עמוד המנצל חולשת 0day בדפדפן אשר מריץ קוד ומדביק את המשתמש התמים בתולעת. כאן אציין כי במקרים מסויימים, במידה ולא נשמרו פרטי ההזדהות בקבצים המיועדים לכך, נעשה שימוש ב- Keyboard Sniffing בכדי לדלות את סיסמת החשבון.

עד היום פורסמו לא מעט חולשות בתוכנות מסרים מידיים או בגורמים שונים שניתן היה לנצל באופן מרוחק בכדי לגרום להרצת קוד דרך תוכנות מסוג זה, כגון המקרה הבא, אך בסופו של דבר נצפו מספר קטן מאוד של תולעים שנצלו חולשות אלו. מספר התולעים שניצלו חולשות בפרוטוקול התקשורת, או במנגנון שאחראי על פרסור ההודעות של תוכנת המסרים המידיים בכדי להריץ את עצמן- מאוד נדירות. במקרים שונים, כמו במקרה של התולעת Witty, נעשה שימוש בחולשת Buffer Overflow שנמצאה במנגנון פענוח פרוטוקול ה-ICQ בתוכנות אבטחה שונות, כגון: BlackICE ו-RealSecure (שתיהן של ISS) בכדי להריץ קוד על הקורבן.

תולעי IRC

IRC הינו פרוטוקול המשמש לביצוע שיחות צ'אט רב משתתפים בזמן אמת. שרתי וערוצי IRC קיימים כבר מסוף שנות התשעים, וכותבי הוירוסים לא פסחו עליהם. חשוב להבין את ההבדל בין Botnets שבמספר רב מהמקרים **ארכיטקטורת השליטה עליהם מתבצעת על-גבי שרתי IRC** לבין תולעי IRC שמנצלות את פרוטוקול ה-IRC **לטובת הפצתן**. למידע בנוגע ל-Botnets ניתן לקרוא במאמר: "Botnet - מה זאת החיה הזאת?" שפורסם בגליון השלישי של Digital Whisper.

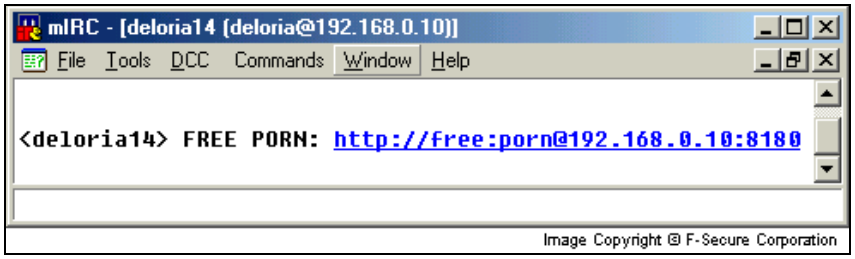
תולעי IRC מתחזות למשתמשי IRC אמיתיים ובעזרת הנדסה חברתית / ניצול חולשות בתוכנות לקוח לפרוטוקול ה-IRC (IRC Clients) מפיצות עצמן. גם בפלטפורמה זאת, המקרים בהם תולעי IRC משתמשות בהנדסה חברתית רבים יותר מתולעים אשר ניצלו חולשות שונות ב-IRC Clients לצרכי הפצה.

אפשר לחלק תולעים מסוג זה לשתי קבוצות:

- תולעים בעלות מנוע IRC מובנה שמאפשר להן להתממשק לערוצי IRC מבלי תלות בתוכנת לקוח ה-IRC שמותקנת על המחשב בו הן רצות.

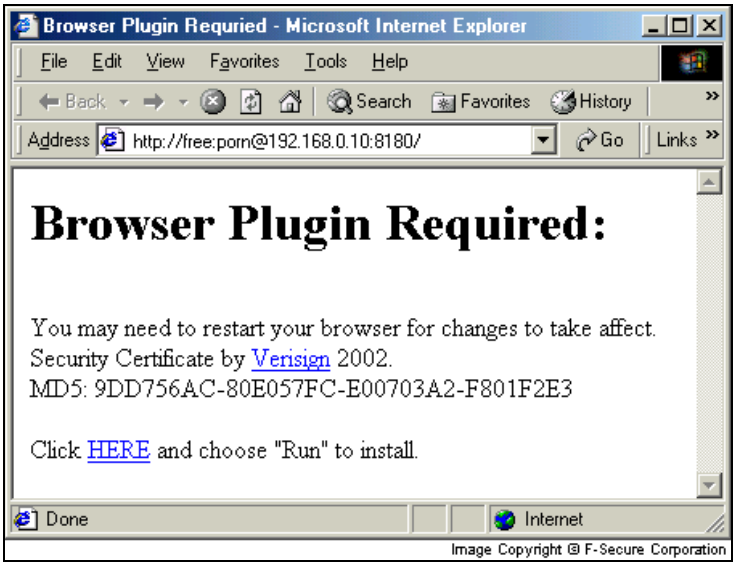
- תולעים חסרות מנוע IRC, אשר מתממשקות לתוכנת הלקוח שמותקנת על המחשב עליו הן רצות ובעזרתן מתקשרות עם שאר משתתפי הערוץ.

נקח לדוגמה את התולעת "Aplore" (מוכרת גם כ-"Aphex") שנצפתה לראשונה בסביבות אמצע שנת 2004. התולעת הפיצה עצמה ע"י מנוע IRC מובנה, שבעזרתו היא הייתה מתחברת לערוצי IRC שונים, ומפיצה הודעה פרטית לכלל המשתמשים הנוכחים בערוץ. תוכן ההודעה הכיל בדרך כלל הודעות בסיגנון "Free Porn" וקישור לכתובת אינטרנט של אתר פורנוגראפי כביכול:



(במקור: http://www.f-secure.com/v-descs/aplore.shtml)

משתמש שהיה נכנס לאתר, היה מתבקש להתקין תוסף לדפדפן הנדרש לצורך צפיה בתוכן האתר:



(במקור: http://www.f-secure.com/v-descs/aplore.shtml)

וכמובן- כל מי שהיה מתקין את התוסף לדפדפן, היה נדבק גם הוא.

לעומת "Aplore", שהכילה מנוע IRC פנימי משלה, תולעים כגון "Momma" ו-"Hellfire" היו תולעים שגם הפיצו עצמן דרך ערוצי IRC אך בשונה מ-"Aplore", הן עשו זאת בעזרת שימוש במנוע הסקריפטינג של התוכנה mIRC (תוכנת לקוח IRC נפוצה). במקרים כאלה יוצר התולעת יכול לבחור בשתי דרכים נוספות להדבקת הקורבן, חוץ מהפניית הקורבן לעמוד עם תוכן זדוני, ניתן לנסות לשכנע את הקורבן להריץ סקריפט (בטענה שמדובר בפקודות הזדהות לערוץ וכו') שיתפקד כ-Backdoor ויאפשר לתוקף שליטה מלאה על תוכנת הצ'אט או שליחת קובץ מדביק על גבי DCC (פרוטוקול נפוץ להעברת מידע Peer to Peer הנתמך במספר רב של לקוחות IRC) שיאפשר גישה למערכת המותקפת. כמובן שלאחר הדבקת המשתמש התולעת אינה נשארת במסגרת ה-mIRC, היא מורידה ומריצה Payloads שונים על המחשב ומקנה לתוקף שליטה מלאה גם על המערכת.

מנוע הסקריפטינג של mIRC (ושאר תוכנות הצ'אט אשר תומכות באוטומציה) מאפשר למשתמש, בין היתר, להגדיר מאקרו- רצף פעולות שיבוצע בכל פעם שאירוע מסויים מתרחש בערוץ IRC בו המשתמש פעיל.

אירועים לדוגמה:

- הרצה ישירה של המאקרו
- הצטרפות לערוץ / שרת
- התנתקות מערוץ / שרת
- הצטרפות משתמש חדש לערוץ
- הופעה של מילה או משפט מסויים בערוץ
- ועוד אירועים רבים...

הפעולות שניתן לבצע הן אינסופיות: מפעולות שניתן לבצע במסגרת ה-IRC (כתיבת הודעה בערוץ, שליחת הודעה פרטית למשתמש, שליחת קובץ למשתמש על גבי DCC, נתינת או לקיחת הרשאות למשתמשים וכו') ועד עבודה על המחשב המקומי (עבודה עם קבצים, הרצת פקודות מערכת, הורדת קבצים מהאינטרנט והרצתם וכו').

שכתוב של קובץ יחיד אשר אחראי על מימוש המאקרו המקושר לאירועים וסיימנו, המשתמש מודבק, אין צורך לבצע Hook לשום פונקציה, ואין צורך לפתוח סוקטים ולפרסר את הצ'אט. תוכנות צ'אט המאפשרות אוטומציה הן מכרה זהב, גם מפני שהן נפוצות וגם מפני שבקלות מאוד ניתן להתממשק אליהן. מסיבות אלו ודומות כמות התולעים מסוג זה היא/הייתה רבה מאוד- כל משתמש ממוצע עם כוונות זדוניות יכול לממש תולעת כזאת גם מבלי להכיר יותר מדי את עולם התכנות.

דוגמאות מעולות לפשטות שבדבר ניתן לראות במאמרים הבאים:

http://users.telenet.be/ahmadi/mircworm.htm

http://vxheavens.com/lib/vsp21.html

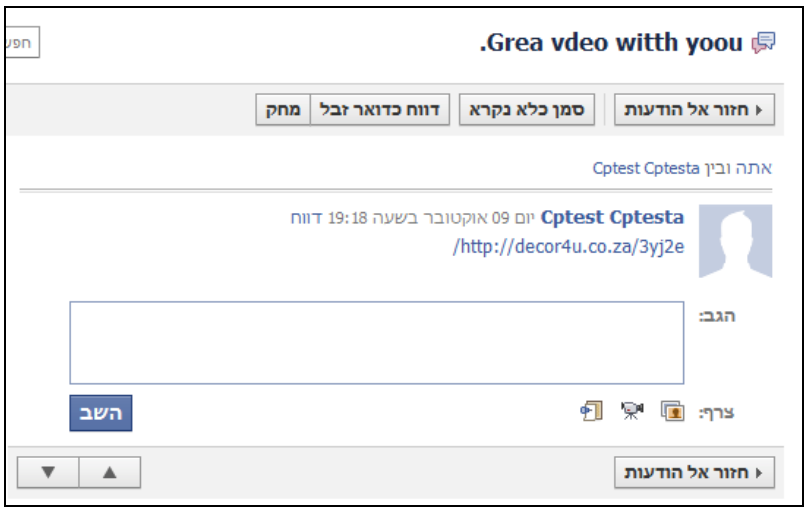
תולעים חברותיות

רשתות חברתיות הן אחת הפלטפורמות הפופולריות כיום בקרב כותבי התולעים, דוגמה טובה לכך אפשר למצוא בחלק הראשון של סדרת המאמרים "Chasing Worms" שפורסם בגליון ה-14 של המגזין, בו הוצג ניתוח של התולעת "Koobface".

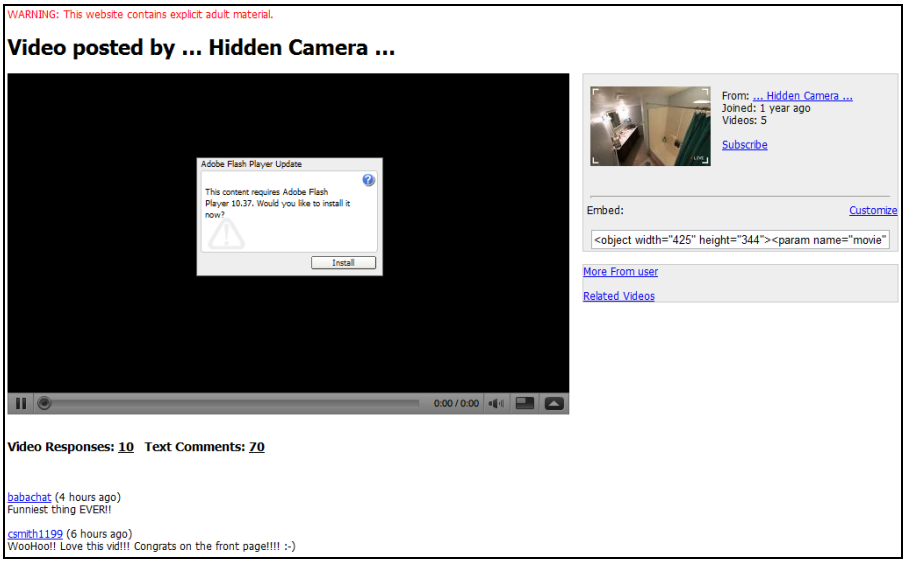
כיום יש מספר רב של רשתות חברתיות, ישנן רשתות בעלות נושאים ספציפיים (LinkedIn, Digg וכו') וישנן רשתות כלליות יותר (Facebook, Google+ וכו'), בשני המקרים, עובר מידע רב בין המשתמשים, וכותבי תולעים הבינו את הפוטנציאל שבדבר.

במקרים הקלאסים, תולעת שמצליחה לגנוב את פרטי ההזדהות של המשתמש פשוט שולחת מהחשבון שנפרץ הודעה פרטית לחבריו ובו קישור לעמוד בעל תוכן זדוני, אך במקרים קצת יותר נדירים נעשה שימוש יפהפה בהנדסה חברתית.

דוגמה למקרה קלאסי הינה Koobface- באופן פשוט יחסית, לאחר השגת גישה לחשבון המשתמש מתבצעת שליחת הודעה פרטית + כתיבה על עמוד הפרופיל של כלל חברי המשתמש עם קישור:



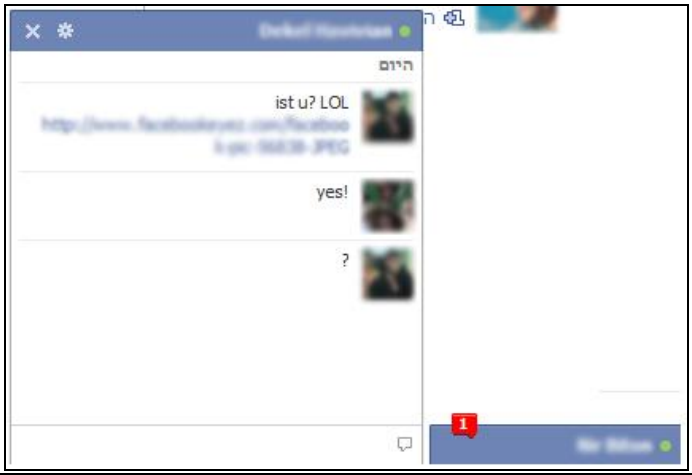
הקישור מפנה לעמוד Youtube פיקטיבי אשר דורש מהמשתמש להתקין עדכון לתוכנת הפלאש שלהם בכדי לצפות בסרטון:



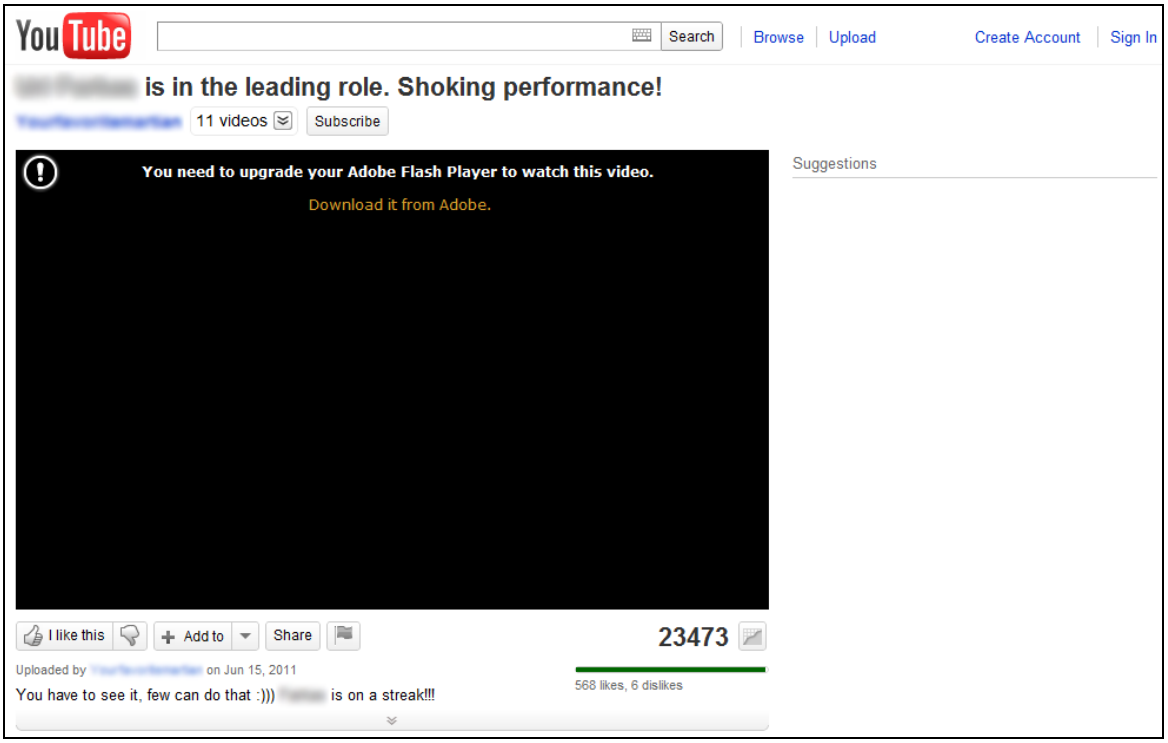
עמוד ה-Youtube סטטי, ומתחתיו יש תגובות של "משתמשי Youtube" פיקטיביים. ולמרות זאת, הדבר הספיק בכדי להפיל אנשים רבים בפח.

דוגמה הרבה פחות קלאסית, והרבה יותר מרשימה, אפשר לזקוף לתולעת חדשה יחסית, עדיין חסרת שם, אך לפי מחקר קצר שעשיתי (ותודה ל-Zerith) שעזר כאן, נראה שהיא עוזרת להפיץ כלי לריקון חשבונות BitCoin.

ברגע שהתולעת מורצת על מחשב ועליו יש חשבון Facebook, היא מתממשקת לחשבון ושולחת הודעת צ'אט לחבר שנמצא כרגע On-Line (גם עם בעל החשבון הפרוץ כרגע מחובר לפייסבוק הוא אינו יראה את שיחת הצ'אט אלא עם החשבון המותקף יכתוב לתולעת בחזרה) עם הודעה וקישור:



הקישור מפנה גם הוא לעמוד Youtube שכמובן דורש מהקורבן להתקין עדכון "תמים" לדפדפן:

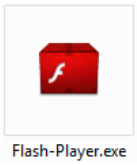


מה שכל כך יפה בעמוד הזה הוא שהוא דינאמי. מה זאת אומרת? זאת אומרת שהוא נוצר On The Fly, על ידי התולעת, שניות מספר לפני שליחת הודעת הצ'אט.

מה דינאמי בעמוד?

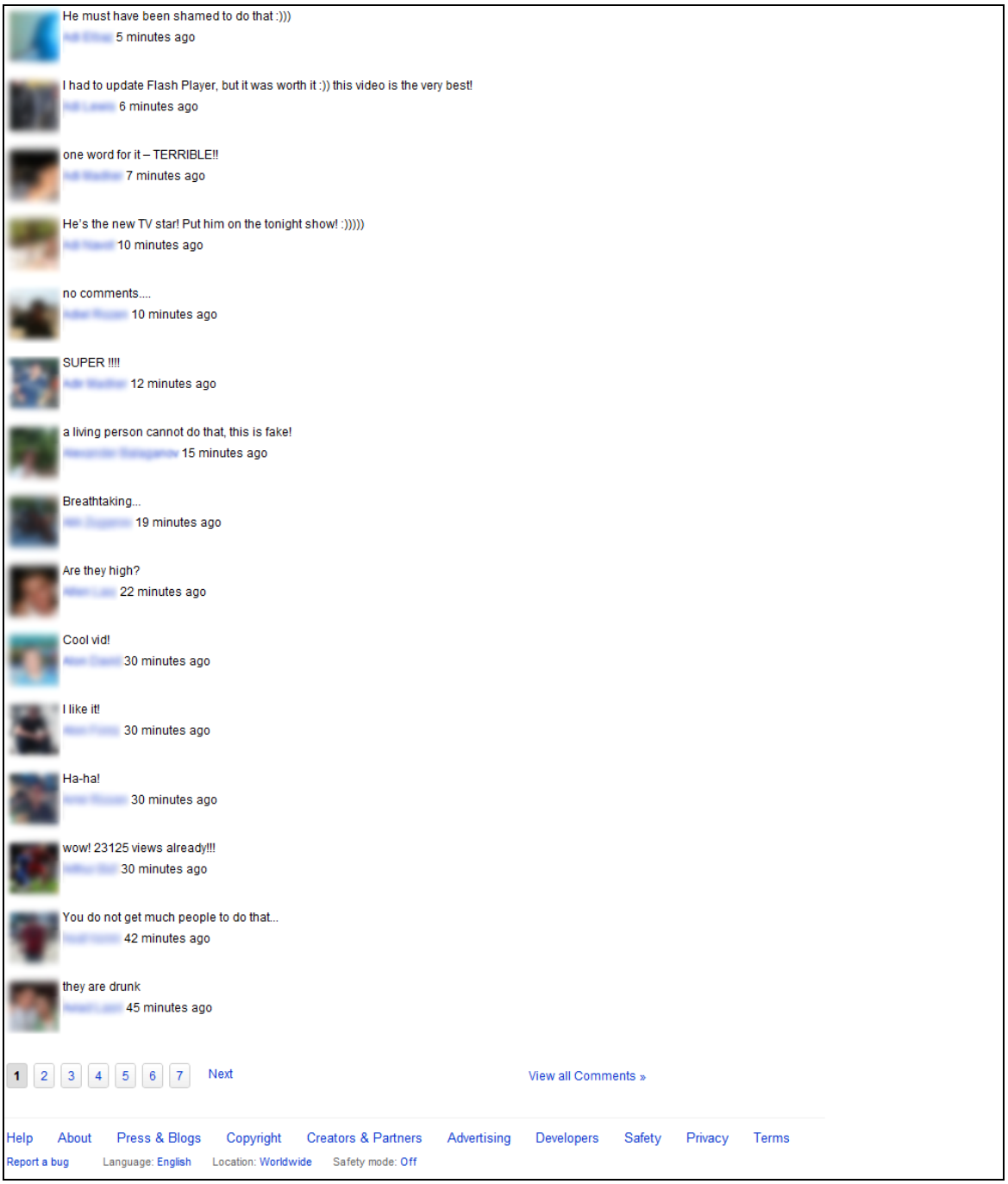
- **הכותרת של הסרטון מרמזת על כך שמדובר בסרטון על הקורבן**
- **התגובות על הסרטון מדברות על הקורבן עצמו.**
- **שמות המגיבי התגובות נלקחו מרשימת החברים של הקורבן.**
- **תמונות המגיבים לקוחות מתמונות פרופיל הפייסבוק האמיתיים של חברי הקורבן.**

שכלול של כלל הנתונים הנ"ל לא רק גורמות לקורבן להאמין כי מדובר בעמוד אמיתי, אלא גם גורמות לו להוריד את העדכון, שאגב, נראה כך:



בהחלט עושה רושם שבוצעה כאן עבודה מקצועית ביותר.

מספר תגובות על הסרטון הפקטיבי:



לאחר הרצת ההעדכון, המשתמש מקבל הודעת שגיאה שאין לו הרשאה להתקין את הקובץ, אך בשלב זה התולעת כבר הדביקה את המחשב...

סיכום

לא נגעתי בכלל הפלטפורמות בהן כותבי התולעים משתמשים, אבל עדיין, מהמידע שהוצג במאמר זה בהחלט ניתן לראות את התפתחות הנושא בשנים החולפות. אם בעבר כותבי התולעים הרשו לעצמם "לעגל פינות", היום, כאשר המודעות לאבטחת מידע גבוהה יותר- נראה כי הם עובדים קשה יותר ויותר בכדי לקנות את אמון הקורבנות. בדרך כלל, משתמש חשדן יוכל לזהות את התרמית, אך במקרים רבים (כגון הדוגמה האחרונה במאמר) נראה כי הסיכויים שגם משתמשים אלו יפלו לפח גבוהים.

נראה כי תולעים אשר מבצעות שימוש בחולשות 0day שונות בעלות עקומת תפוצה נרחבת יותר, אך כאשר מתגלה התולעת ונסגרת אותה החולשה, חיי התולעת קצרים ביותר. לעומת זאת, וקטור הפצה המשתמש בהנדסה חברתית אומנם מאפשר לתולעת להפיץ עצמה בקצב איטי יותר, אך כאן, אין באג שצריך לסגור, והדרך היחידה לעצור את התולעת מלהתפשט היא לעדכן את תוכנת האנטי-וירוס ולהגביר את המודעות הסביבתית לאבטחת מידע.

מקורות

- http://www.wormblog.com/im_worms/
- http://ftp.erm.tu-cottbus.de/security/witty-analysis.html
- http://digitalwhisper.co.il/files/Zines/0x0E/DW14-1-KoobfacePwning.pdf
- http://vx.netlux.org/29a
- http://www.symantec.com/security_response/writeup.jsp?docid=2001-021219-1830-99
- http://en.wikipedia.org/wiki/Anna_Kournikova_(computer_virus)
- http://en.wikipedia.org/wiki/ILOVEYOU
- http://iamjenessa.wordpress.com/2011/06/28/memories-of-the-love-bug-worm-by-graham-cluley-on-may-4-2009
- http://www.f-secure.com/v-descs/onthefly.shtml
- http://www.spywareguide.com/product_show.php?id=3108
- http://www.kaspersky.co.in/news?id=4277566

HTML5 - מנקודת מבט אחרת

מאת: לירן בנודיס ואלעד גבאי

הקדמה

בזמן האחרון אנו שומעים עוד ועוד על HTML5 - יש הגורסים כי זוהי הטכנולוגיה שתשנה את פני הרשת (ואף תהווה תחליף לטכנולוגיות פיתוח רבות), אחרים מתעסקים בגישות העיצוב החדשות שטכנולוגיה זו מאפשרת ואחרים מתעסקים בשאלה האם HTML5 תחליף את Flash של Adobe או Silverlight של מיקרוסופט.

על אף השפע העיצובי והטכנולוגי ש-HTML5 מביאה איתה, ישנם לא מעט אנשים שלא מדברים על אפשרויות העיצוב החדשות שטומנת בתוכה HTML5, לא מתעסקים בדיונים על אפשרויות לשדרג אתרים קיימים בעזרתה ואפילו לא על אפשרויות השיווק החדשות. אנשים אלו, שהתעמקו ב-HTML5 והטכנולוגיות הנלוות, רואים סיבות לדאגה - HTML5 טומנת בתוכה, בין שאר הירקות, הרבה בעיות פרטיות ואבטחת מידע.

בסדרת מאמרים הבאים, נסקור את הנקודות הבעייתיות ב HTML5 ובטכנולוגית הנלוות. מאמר זה יסקור את התגים והתכונות החדשות ב-HTML5, את מנגנון ה-XDR/CORS וכמובן ידון בהשלכות של אלו על מידת האבטחה של התקן החדש.

יש לזכור כי כרגע HTML5 הינו תקן בפיתוח ולכן חלק מדוגמאות הקוד המובאות במאמר זה לא יעבדו בכל הדפדפנים.

HTML (HyperText Markup Language)

HTML היא שפת תגיות המיועדת להצגת תוכן בעמודי אינטרנט. זו שפה קלה יחסית ללמידה וניתן להציג להציג בעזרתה אינסוף תכנים באינסוף צורות שונות. את תקן השפה קובע ארגון ה-W3C (ארגון תקינת הרשת העולמי).

גרסת ה-HTML המוכרת והיציבה, אשר נקבעה ב-1999 ולא שונתה מאז, הינה HTML4.01, אליה מתלווים CSS, JavaScript ו-HTML DOM, ויוצרים את השפה הדינמית DHTML. מאז, עולם המחשוב כמו גם האינטרנט השתנו רבות, ויש צורך לענות על צרכי מפתחי האתרים שתמיד דורשים יותר ויותר, וכמו כן גם על צרכי הקהל אשר צורך את המידע ותמיד דורש אפשרויות נוספות, ובכל פעם צורך את המידע בצורה חדשה ובפלטפורמות שונות (פלאפונים, מכשירי Tab, טלוויזיות, windows8, טוסטר משולשים, ועוד...). לפיכך, גבר הצורך בעדכון שפת HTML.

הדור הבא של התקן הינו HTML5, שגרסת התקן הראשונה שלו יצאה ב-2008 ועדיין נמצא בפיתוח.

כיום, כאשר דרישת הקהל, ובעקבותיו דרישת המפתחים, לפיצ'רים חדשים ופונקציות נוספות היא מיידית, לא תמיד ניתן לספק דרישה זו בזמן. דוגמה חייה היא HTML5. לאחרונה ישנם הרבה אתרי אינטרנט המשתמשים בתקן ה-HTML5. תקן זה עוד נמצא בפיתוח ועתיד לצאת רשמית רק ב-2014!

העניין שלנו ב-HTML נובע בדיוק מסיבה זו. כיום נוצר מצב שהתקן של HTML5 לא קבוע ומשתנה לעתים תכופות.

רוב הדפדפנים הפופולריים כבר תומכים בחלק מן האלמנטים והפיצ'רים החדשים שמציע התקן, כל דפדפן על פי בחירתו ובצורת מימוש שונה. בנוסף על כל זה, יש את עניין הפיצ'רים הנוספים של HTML5, שעצם הוספתם של חלק גדול מהם יוצר בעיות אבטחה.

התגיות החדשות

ב-HTML5 ישנן-28 תגיות חדשות, ניתן לסווגן לסוגים:

<u>תגיות חלוקת הדף וטקסט:</u>

התגיות הללו לא משנות או מבצעות שום דבר בנוגע לעיצוב, אלא רק נותנות סמנטיקה עבור התוכן אותו הן מייצגות. סמנטיקה זו עוזרת מאוד למנועי חיפוש ורובוטים.

חלוקה לדוגמה של הפוסט האחרון ב-DigitalWhisper בעזרת התגים החדשים:

כמו שרואים בתמונה, תג ה-header מסמן את ראש העמוד, בעוד שתג ה-footer מסמן את סופו (לא עוד תג div עם id המתאר את משמעות התוכן). בנוסף, ניתן לראות כי קיים תג article, המכיל את תוכן המאמר, תג nav המכיל את תפריט הניווט של האתר, ותג aside אשר מכיל את התפריט הצדדי, כאשר בתוכו תגי section המגדירים את החלקים השונים בתפריט.

לא נפרט יותר מידי על תגים אלו במאמר זה, אך אלו מכם המתעניינים יכולים למצוא יותר מידע בקישורים בסוף הפרק.

<u>תגיות מולטימדיה:</u>

HTML5 מאפשרת מספר תגי מולטימדיה המשמשים להטמעת קול, סרטונים ופלאגינים בעמוד.

תג הווידאו והאודיו מאפשרים לנו להציג תוכן ממקור (origin) אחר, וזה יאפשר לנו להציג סרטונים אהובים מאתרים אחרים ללא כל צורך לאחסן את אותם הסרטים בשרתים שלנו, או להשמיע רשימות מוזיקה שלמות ומתעדכנות מאתרי מוזיקה מובילים. אפשרות זאת יוצרת בעיית אבטחה משני כיוונים:

כאשר אתר תמים מציג תוכן זדוני, וכאשר אתר זדוני מציג תוכן תמים.

מצד אחד, כאשר עמוד תמים מציג תוכן זדוני, החשש העיקרי הוא שמקור המדיה הזדוני עלול להכיל סקריפט כלשהו אשר ינסה לבצע אינטראקציה עם תוכן העמוד ולשנות אותו. הדפדפנים יכולים לבצע הפרדה בין ה-context של התוכן המוצג לבין זה של העמוד המציג אותו, לדוגמה, אם בעל העמוד חפץ להציג אנימציית svg בתוך תג video, על הדפדפן להפריד בין השתיים כך שהתג יראה את עצמו כעמוד נפרד ללא אב ולא יוכל לשלוט על ה-DOM בעמוד שהטמיע את הוידיאו.

מצד שני, כאשר עמוד זדוני מציג תוכן תמים, החשש הוא שהעמוד יוכל לשלוף מידע על אותו התוכן שבצורה אחרת לא יכל להשיג. דוגמאות למידע שהעמוד הזדוני יכול להציג הן: משך הסרטון\אודיו, משקל הקובץ, עצם קיומו של הקובץ, מידע נוסף על השרת המאחסן את הקובץ ועוד.

חשיפה של מידע מסוג זה לאתר זדוני היא כבר בעייתית, אך קיימת בעיה חמורה עוד יותר: אם הדפדפן חושף מידע נוסף על התוכן (כמו שהרבה גופים אוהבים לעשות), כמו כתוביות, רשימת כותרים או פרקים או כל מידע אחר. מידע שכזה אולי לא נראה חשוב כל כך, אבל תחשבו על התרחיש הבא:

נניח כי קיימת חברת סטארט-אפ אשר עובדת על פיתוח של רעיון אשר עתיד להניב לחברה המון כסף. במחשבי החברה קיימים סרטונים אשר מציגים את שלבי הפיתוח עד כה. בכדי לעדכן משקיעים ולהלהיב משקיעים פוטנציאלים, לסרטונים יש כתוביות על מנת שגם משקיעים אשר לא דוברים את שפת הקודש

יוכלו להבין שהחברה בדרך לכסף הגדול. עובד תמים מן החברה גולש לעמוד זדוני מתוך אחד ממחשבי החברה. בעת הגלישה לעמוד, מורץ ברקע על הדפדפן סקריפט זדוני אשר מציג (בגלוי או בסתר) לעובד את הסרטונים הרגישים הללו. כמובן שכעת הסקריפט יוכל לראות מה אורך הסרטונים. ובאם דפדפן הגולש החליט לספק קצת יותר מידע, יוכל העמוד הזדוני גם לראות את הכתוביות עבור הסרטים, ואלו כבר יספיקו לו בכדי לדעת מהו מוצר הקסם עליו החברה עובדת. בנוסף לבעיות אלו יש לקחת בחשבון כי מימוש המנגנון הנ"ל הינו מטלת הדפדפן, בניגוד למצב כיום שמטלה זו מבוצעת על ידי פלאגינים חיצוניים, בעיקר על ידי הנפוצים כמו Adobe Flash ו-Microsoft Silverlight אשר ידועים כמקורות טובים לבעיות אבטחה. כעת הדפדפנים יצטרכו לממש זאת בעצמם, וכנראה שנראה שוב חורי אבטחה. הדפדפן יצטרך להתמודד עם מספר גדול יותר של פורמטים מאשר היה צריך בעבר (למשל avi, mp4, mp3, ...) ובין השאר יצטרך להתמודד מול התקפות עתידיות אשר ישתמשו בחולשות בפורמטים של קבצי ווידאו ואודיו, ובנגנים אשר יממש על מנת להציגם.

יש לשים לב כי למרות שרוב מה שכיום תוספים כמו Adobe Flash ו-Microsoft Silverlight עושים בעמודי אינטרנט, כעת יכול להתבצע בעזרת HTML5 ו-JS, עדיין HTML5 מאפשר לנו להטמיע תוספים (plugins) בעמוד בעזרת תג ה-embed, בתוספים אלו יכולים להיות פרצות אבטחה בעצמם ובמקרה שכזה אלו יכולים להוות סיכון עבורנו ועבור המשתמשים.

<u>תג ה-canvas:</u>

תג ה-canvas מאפשר רינדור דינמי של תמונות בעזרת סקריפט, או במילים אחרות, ניתן לצייר על תג ה-canvas בעזרת javascript.

בכדי ליצור canvas נכתוב את קוד ה-HTML הבא:

```
<canvas id="example" width="200" height="200">
This text is displayed if your browser does not support HTML5 Canvas.
</canvas>
```

וכעת נצייר עליו בעזרת JS מלבן אדום:

```
var example = document.getElementById('example');
var context = example.getContext('2d');
context.fillStyle = "rgb(255,0,0)";
context.fillRect(30, 30, 50, 50);
```

עד כאן הכל טוב ויפה. מה הבעיה?

אם קיים מצב בו סקריפט ממקור (origin) אחד יכול לגשת למידע אשר נמצא במקור שונה, הסקריפט יכול לקרוא את הפיקסלים מן ה-canvas ולדעת מהי התמונה המוצגת.

בכדי למגר את הבעיה, החליטו ב-W3C להוסיף דגל (flag) בשם origin-clean. הדגל מקבל ערך בוליאני, true אם כל האובייקטים המצוירים ב-canvas (או המשמשים לציור ה-canvas) נמצאים באותו origin כמו העמוד המכיל את אלמנט ה-canvas ו-false אחרת.

התנאים היותר מדויקים נמצאים כאן:

http://dev.w3.org/html5/spec/Overview.html#security-with-canvas-elements

גם עם הפתרון הזה, הבעיה עדיין קיימת ובכל פעם שה-canvas משתמש באובייקט ממקור אחר, סקריפט כלשהו שרץ ב-context של העמוד יוכל לדעת בדיוק מה מוצג על ה-canvas. להרחבה:

http://html5example.net/tag/WebGL%20Browser%20Exploitation

<u>תגיות טפסים</u>

ב-HTML5 נוספו כמה תגי טפסים נוספים כמו, תג ה-output אשר אמור לתחום פלט מסוים הנוצר על ידי סקריפט, ותג ה-datalist אשר מחזיקה אפשרויות לבחירה והשלמה אוטומטית בתיבת טקסט. אך התג הכי מעניין שנוסף הוא תג ה-keygen אשר מייצר זוג מפתחות רנדומליים.

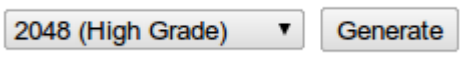
תג ה-keygen

כמו שנאמר, תג זה מייצר זוג מפתחות רנדומליים המתאימים להצפנה א-סימטרית המשמשים לאותנטיקציה. כאשר הטופס (שבו נמצא התג) ישלח, המפתח הפרטי ישמר ב-Local keystore והמפתח הפומבי נארז ונשלח אל השרת.

לדוגמה, נוכל ליצור את הטופס הבא:

```
 <form method=”POST” action=”http://www.server.com”>
<keygen name="pubkey">
<input type="submit" name="createcert" value="Generate">
</form>
```

אשר הדפדפן (כרום) ירנדר בצורה הבאה:



וב-Drop Down List תהיה רשימה של ה-keylength הנתמכים על ידי הדפדפן.

כאשר נלחץ על Generate, הדפדפן ייצר שני מפתחות (פרטי וציבורי) ואת המפתח הפומבי ישלח לשרת http://www.server.com.

לתג זה יש מספר תכונות:

- תכונת ה-challenge אשר מקבלת מחרוזת (אם לא מוגדר, מקבל ערך של מחרוזת ריקה).
- תכונת ה-keytype אשר מגדירה את האלגוריתם ששימש ליצירת את המפתח (כרגע אין רשימה של סוגי הצפנות שעל דפדפנים לממש, אך שדה זה יכול להכיל לדוגמה את הערך "RSA")

המפתח הפומבי ומחרוזת ה-challenge מקודדים בעזרת DER ונקראים PublicKeyAndChallenge. לאחר מכן, ערך זה נחתם בעזרת המפתח הפרטי בכדי ליצור ערך הנקרא SignedPublicKeyAndChallenge. ערך זה נשלח לשרת ביחד עם ערך מפתח ההצפנה הפומבי.

משהו נוסף שיהיה מעניין לראות הוא כמה רנדומלי יהיה רצף הבתים הנוצר. גם בעבר נראו אלגוריתמים אשר כביכול נתנו רצף בתים רנדומלי, אך לבסוף התגלה כי הרנדומליות לא כל כך רנדומלית.

בנוסף לתגים, ישנם כמה ערכים נוספים אשר ניתן לשים בתכונת ה-type של תג ה-input, למשל: search, url, datetime, email, color ועוד. אלו משפיעים בעיקר על תצוגת שדה הקלט ועל אימות תאימות הקלט לסוג השדה על ידי הדפדפן.

כל התוספות הללו לא מהוות בעיית אבטחה מבחינה עקרונית, אך ברגע שכולן ימומשו בדפדפנים יכול מאוד להיות שצורת המימוש תהייה פגומה. בדיקות תאימות קלט הן דבר מסובך, ויעבור זמן עד שנוכל לתת לדפדפן לבצע זאת עבורנו בעיניים עצומות. גם אז נצטרך לחשוב על כל הדפדפנים ולא רק על אחד שהמימוש שלו נכון. כמובן שזה לא חוסך לנו את הבדיקות בצד השרת. משתמש זדוני יכול בקלות לעקוף בדיקות בצד הקליינט, וגם משתמש שאינו זדוני ומשתמש בדפדפן ישן אשר אינו מבצע בדיקות אלו בצורה טובה או בכלל לא, ואף כלים אוטומטים למילוי טפסים יוכלו לשלוח לשרת מידע אשר לא אומת.

אם כן, אסור להסתמך על בדיקות אלו לא בצד הקליינט (אם כי זה פחות קריטי), ובטח שלא בצד שרת! **מנגנוני הבדיקה הללו נועדו אך ורק בכדי להוסיף לחוויות המשתמש ולא כמנגנון אבטחה.**

את רשימת התגים החדשים של HTML5 והסבר עליהם ניתן לראות בלינקים הבאים:

- http://www.w3schools.com/html5/html5_new_elements.asp
- http://www.htm.co.il/2009/10/26/html-5-%D7%9E%D7%94-%D7%A0%D7%9B%D7%A0%D7%A1-%D7%95%D7%9E%D7%94-%D7%99%D7%95%D7%A6%D7%90-%D7%97%D7%9C%D7%A7-%D7%90/

התכונות החדשות:

ב-HTML5 ישנן תכונות גלובליות אשר חלות על כל התגים, כמו תכונת ה-contenteditable אשר קובעת כי המשתמש יכול לערוך את תוכן התג.

כמו כן ישנן גם תכונות חדשות עבור תגים מוכרים כמו תכונת ה-autofocus אשר בעת מילוי טופס תעשה פוקוס אוטומטי על השדה הרצוי, תכונת ה-placeholder אשר בעת מילוי טופס תראה לנו דוגמה למה אמור להיות באותו השדה, תכונת ה-required אשר מגדירה כי שדה מסוים בטופס הינו שדה חובה, תכונת ה-pattern תגרום לדפדפן לבדוק שהערך בשדה ה-input שעבורו יישמנו את התכונה יתאים לביטוי הרגולרי שקיבלה התכונה כערך.

בנוסף, ב-HTML5 לא חייבים לתת לכל תכונה ערך, וכאשר כותבים ערך שהוא ללא רווחים, לא חייבים לשים יותר גרשיים (למרות שזה נראה טוב יותר עם גרשיים). ניתן למשל לכתוב את התג הבא:

```
<input type=email placeholder=”hello@mail.com” autofocus required / >
```

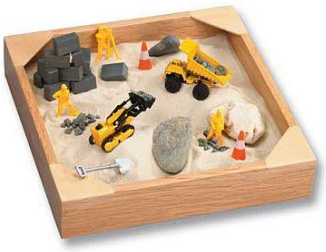
דבר נוסף שחלקכם ימצאו שימושי הוא תכונות מותאמות אישית (Custom Attributes), זאת אומרת שאם יש תכונה שהייתם רוצים להוסיף והיא לא קיימת ב HTML5, אתם יכולים לכתוב אותה ולא תיגרם כל שגיאה. ניתן לאחר מכן להשתמש בה ב-JS וב-CSS. זוהי דרך מצוינת להכניס עוד מידע הרלוונטי לעמוד אך לא נמצא לו מקום מתאים. דוגמה טובה לשימוש בזה היא ליצור תכונת explanation המכילה הסבר על המושג שהתג שהיא נמצאת בו תוחם, ובעזרת JS להציג את אותו הסבר כאשר עוברים על המילה. כך ההסבר יכלל ב-HTML אשר אמור להכיל את התוכן.

כמובן שגם את התכונות המותאמות ניתן לנצל אם קיים מצב כמו בדוגמה. המנגנון אשר מדפיס את ההגדרות של המושגים משתמש ב-document.write, ומעביר לו כפרמטר את ערך ההגדרה. כעת נניח אפילו שיש IDS, IPS או כל מנגנון אחר אשר מסנן כל תכונה קיימת ב-HTML וגם ב-HTML5 כדי שלא נצליח להחדיר שום תכונה או מאורע, אשר ישמשו להרצת סקריפט, לתוך התג הנ"ל. אם המנגנון לא עובר התאמה לתכונות החדשות שבעל האתר הוסיף (וזה משהו שקל יחסית לפספס), נוכל להזריק את אותה התכונה למקומות החשופים ל-XSS (כמובן שהם נבדקים על ידי המנגנון ולכן בעל האתר חושב שהם בטוחים), ולגרום להרצת סקריפט.

תג ה-iframe:

תג ה-Iframe הינו תג המאפשר להטמיע עמוד אינטרנט אחר ישירות בתוך האתר. קיימות הרבה בעיות אבטחה בדבר זה. כחלק מן הניסיון לפתור בעיות אלו, וגם בכדי להוסיף לפונקציונלית של ה-iframe, הוסיפו לתג ה-iframe כמה תכונות מעניינות.

תכונת ה-sandbox



תכונת ה-sandbox היא תכונה שימושית ביותר. כאשר נוסיף תכונה זו לתג Iframe, הדפדפן בעצם יוריד את ההרשאות של הדף הזה וימנע ממנו לבצע פעולות מסוימות שעשויות לסכן את המשתמש ואת האתר עצמו. תכונה זו יכולה לקבל ארבעה ערכים:

- allow-forms - מאפשר שליחת טפסים מן הדף המוטמע.
- allow-same-origin - מתייחס לתוכן הדף כאילו הגיע מן הדומיין של האתר המטמיע (בין השאר מאפשר לסקריפטים בדף המוטמע לגשת לאלמנטי DOM, לעוגיות ו-localstorage הנמצאים בדומיין של הדף המטמיע).
- allow-scripts - מאפשר הרצת סקריפטים בדף המוטמע (למרות שעדיין הסקריפטים אינם יכולים להקפיץ הודעות pop-ups).
- allow-top-navigation - מאפשר ללינקים ולסקריפטים לנווט את העמוד המטמיע (למשל, בזמן טעינת הדף לקשר את העמוד העליון לעמוד פישינג).

בנוסף, במקרה שלא מכניסים אף אחד מן הערכים הללו, אלא נותנים ערך ריק, הדפדפן יחסום את כל הפעולות הנ"ל. זאת אומרת שכאשר הערך הוא ריק, הדפדפן יראה את העמוד המוטמע בתור עמוד מדומיין אחר, שליחת טפסים והרצת סקריפטים מן הדף המוטמע לא תתאפשר, לינקים לא יוכלו לנווט את העמוד המטמיע ותוספים אחרים (plugins) לא יכולו לרוץ.

חדי העין (וחדי הקרן) בוודאי שמו לב שבמקרה ובאתר המוטמע מוגדרים גם allow-same-origin, גם allow-scripts וגם העמוד המוטמע ב-iframe הוא מאותו מקור, האתר המוטמע יכול בקלות לגשת לדף המטמיע ולבטל בעזרת סקריפט את תכונת ה-sandbox ולבטל כל הגנה שזו סיפקה, קוד לדוגמה:

inner.html:

```
<script>
top.document.getElementsByTagName("iframe")[0].attributes.removeNamedIte
m("sandbox");
</script>
```

outer.html:

```
<iframe src="http://127.0.0.1:80/inner.html" sandbox="allow-same-origin
allow-scripts">
</iframe>
```

תכונת ה-srcdoc

תכונת ה-srcdoc מאפשרת למפתח להכניס לתוך ה-Iframe קוד HTML של האתר המוטמע. זאת אומרת, במקום לתת את הלינק לדף שברצוננו להטמיע ולגרום לדפדפן לבצע בקשה עבור התוכן של דף זה, ניתן לבצע את הבקשה בעצמנו ולהכניס ישירות את קוד ה-HTML.

במחשבה ראשונה הרעיון נשמע קצת מוזר, כי הרי באותו אופן יכולנו להעתיק את קוד המקור של העמוד לתוך העמוד שלנו. אך יש לזכור כי חלק מהתפקיד של תג ה-Iframe הוא למנוע גישת תוכן לא מורשה לתוכן העמוד שלנו (בין השאר בעזרת תכונת ה-sandbox).

במידה ולתג Iframe קיימות גם תכונת ה-src וגם תכונת ה-srcdoc תמיד תהיה עדיפות לתכונת ה-srcdoc. רק במצב של דפדפן מיושן אשר לא מכיר את תכונה זו יהיה שימוש בתכונת ה-src.

תכונת ה-seamless

מפתחים רבים משתמשים בתגי HTML עם סמנטיקה מאוד ספציפית עבור תוכן לא מתאים בטענה ש"זה נראה יפה". תכונת ה-seamless היא תכונה בוליאנית שמעלימה את הגבולות של תג ה-Iframe וגורמת לו להיראות כחלק מן העמוד.

ניתן להשתמש בה בצורה הבאה:

```
<iframe seamless>
```

אך זה לא כל מה שתכונה זו עושה. התכונה לא רק גורמת לדף המוטמע להיראות כחלק מהעמוד, היא גם גורמת לדפדפן לרנדר את הדף המוטמע כחלק מהעמוד!

כמה דוגמאות להתנהגות הדפדפן במקרה שהתכונה מופיעה בתג Iframe מסוים:

- לינקים בתוך ה-Iframe ינווטו מחדש את הדף כולו ולא רק את ה-Iframe (אלא אם ב- target צוין _self).
- ה-css של הדף המוטמע ישפיע לא רק על ה-Iframe אלא יצורף גם אל ה-css של העמוד המטמיע וישפיע על העמוד כולו.
- רוחבו של ה-Iframe יהיה זהה לזה של כל אלמנט אחר באותה רמה ובעל רוחב אוטומטי.
- גובהו של ה-Iframe יהיה גובה של מסגרת התוכן המוטמע בתוכו (Bounding Box) בהינתן הרוחב שהוגדר בנקודה הקודמת.
- כאשר תוכנת נגישות כמו קורא קולי יקרא את העמוד, הוא יקרא גם את תוכן ה-Iframe בלי להכריז על כך שמדובר בדף נפרד.

- הדפדפן מתייחס אל הדף המוטמע כאילו הוא חלק מן הדף המטמיע, גם כאשר נריץ סקריפט. לדוגמה, אם בדף המטמיע יהיה סקריפט אשר סופר את מספר הלינקים בעמוד, גם הלינקים אשר נמצאים בתוכן של הדף המוטמע יספרו.

בקשות בין מקורות שונים (Cross-Origin Requests - XDR) ו- Cross Origin Resource Sharing - (CORS

בימים שלפני HTML5, בקשות AJAX הוגבלו על ידי "Same Origin Policy", אשר מאפשרת לבקשות עבור משאב מסוים להתבצע רק בתוך אותו דומיין, זאת אומרת שהעמוד http://www.site.com/index.html יכל לבצע בקשות רק עבור משאבים הנמצאים תחת http://www.site.com. ב-HTML5 ביטלו הגבלה זו, ומאפשרים לבקשות AJAX להתבצע בין מקורות שונים.

למה לבטל את ההגבלה? זהו לא משהו חדש, כבר היום מפתחי אתרים יוצרים מנגנונים משלהם על מנת לאפשר בקשות בין מקורות, על מנת לספק למשל APIs ולנצל יותר מן הפוטנציאל של הרשת (תחשבו על מערכות התגובות של פייסבוק המאפשרת הטמעה באתרים שונים).

מה חדש?

בקשות למשאבים ממקורות אחרים יכלו להתבצע גם עוד לפני HTML5, הרי עוד הרבה לפני ה-AJAX וה-HTML5 דפדפנים היו מבצעים בקשות עבור משאב הנמצא בדומיין אחר, אם על ידי תג img, script או iframe.

אך בבקשות מסוג זה לעמוד לא הייתה גישה לתוכן המשאב. הדפדפן רק הריץ את המשאב או רינדר אותו והציג אותו בעמוד.

ההבדל הוא ש-XDR מאפשר ל-JavaScript לקרוא את תוכן המשאב על ידי תכונת ה-'responseText'.

איך זה פועל?

XDR הוא מעין מפרט על שמגדיר איך לאפשר בקשות Cross-Origin אל עמוד מסוים. מפרט זה מגדיר מספר כותרים (headers) שניתן להוסיף לבקשות ותגובות של HTTP.

הכותרים שמוגדרות ב-XDR

XDR מגדיר מספר כותרים, חלקם יוכלו בבקשות מן השרת וחלקם בתוך תגובות השרת לבקשות אלו. אנו נסביר כאן רק על הכותרים העיקריים:

Access-Control-Allow-Origin

Access-Control-Allow-Origin נמצאת בכותר של תגובה מן השרת, ומגדירה לאילו דומיינים מותר לבקש בבקשת Cross-Origin את המשאב.

התחביר הוא:

```
Access-Control-Allow-Origin = רשימת דומיינים | *
```

(כאשר התו '*' יגרום לכך שכל דומיין יוכל לבצע את הבקשה.)

זה כנראה יהיה ה-header הבעייתי ביותר. הרבה מפתחים ישימו כערך '*' ובכך בעצם יצרו דלת פתוחה לכל מי שרוצה לבצע XDR.

Access-Control-Max-Age

Access-Control-Max-Age נמצאת בכותר של תגובה מן השרת, ומגדירה לכמה זמן התגובה לבקשת preflight (יוסבר בהמשך) תישמר ב-cache.

התחביר הוא:

```
Access-Control-Max-Age = מספר שניות
```

Access-Control-Allow-Methods

Access-Control-Allow-Methods נמצאת בכותר של תגובה מן השרת, ומגדירה באילו שיטות מותר לבקש בבקשת Cross-Origin את המשאב.

התחביר הוא:

```
Access-Control-Allow-Methods: GET/POST/OPTIONS/DELETE/...
```

ניתן להגדיר גם כמה שיטות על ידי רשימת שמן עם רווח ביניהן.

Access-Control-Allow-Headers

Access-Control-Allow-Headers נמצאת בכותר של תגובה מן השרת, ומגדירה עם אילו custom headers מותר לבקש בבקשת- Cross-Origin את המשאב.

התחביר הוא:

```
Access-Control-Allow-Headers: custom headers names
```

Access-Control-Allow-Credentials

Access-Control-Allow-Credentials נמצאת בכותר של תגובה מן השרת, ומגדירה האם התשובה לבקשה יכולה להיעשות כאשר דגל ה-Credentials (יוסבר בהמשך) דלוק.

כאשר כותר זה חוזר עם תשובה לבקשת preflight, הוא מגדיר האם הבקשה האמיתית יכולה להיעשות כאשר דגל ה-Credentials דלוק.

התחביר הוא:

```
Access-Control-Allow-Credentials: true | true: %x74.72.75.65
```

Origin

Origin נמצאת בכותר של בקשה לשרת, ומגדירה מאיזה דומיין נעשית הבקשה.

Access-Control-Request-Method

Access-Control-Request-Method נמצאת בכותר של בקשה לשרת, וכאשר הדפדפן מבצע בקשת preflight, הוא משתמש בכותר זה על מנת להגדיר באיזו שיטה הוא ירצה לבצע את הבקשה האמתית.

Access-Control-Request-Headers

Access-Control-Request-Headers נמצאת בכותר של בקשה לשרת, וכאשר הדפדפן מבצע בקשת preflight הוא משתמש בכותר זה על מנת להגדיר עם אילו custom headers הוא ירצה לבצע את הבקשה האמתית.

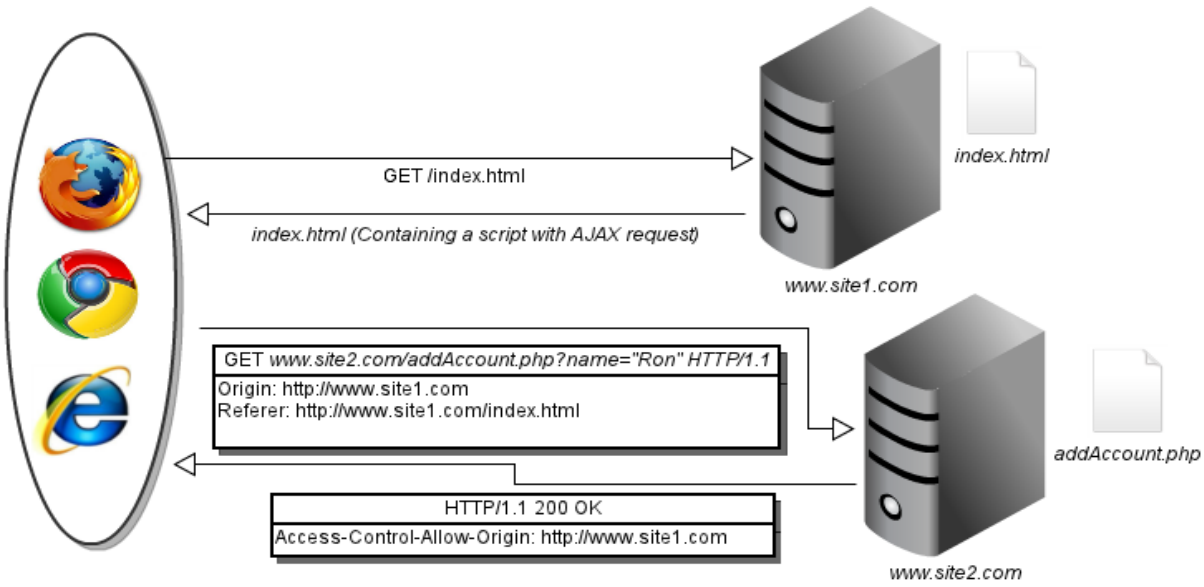
שימוש ב-XDR

ניתן להשתמש במפרט ה-XDR בעמוד http://site2.com/addAccount.php בכדי לאפשר לעמוד http://site1.com/index.html ולעמודים נוספים לבצע בקשת Cross-Origin אליו בצורה הבאה:

בכדי לעשות זאת נוסיף את ה-header:

```
Access-Control-Allow-Origin: http://site1.com
```

לתשובה שהשרת של http://site2.com מחזיר בעת בקשת הדף addAccount.php.



בעת ביצוע בקשה מן העמוד: http://site1.com/index.html לעמוד: http://site2.com/addAccount.php הדפדפן יבצע את הצעדים הבאים:

1. יבצע בקשה לעמוד http://site2.com/addAccount.php
2. יבדוק את ה-headers של התשובה שהוא קיבל חזרה, ויחפש אישור לבקשת Cross-Origin מן הדומיין http://site1.com
   a. אם הדפדפן מצא header אשר מגדיר שלדומיין הזר מותר לבצע את הבקשה הזו, העמוד יוכל לקרוא את תוכן התגובה.
   b. אחרת, העמוד לא יוכל לקרוא את תוכן התגובה.

ניתן לחשוב שסדר זה בעייתי, שהרי הדפדפן קודם מבצע את הבקשה ורק אז לפי תגובת השרת מוודא אם מותר לו לבצע את הבקשה. אבל לפעמים עצם בקשת עמוד מסוים עלולה לגרום לפעולות לא מאושרות בצד השרת.

אך זהו לא באמת סיכון מכוון שגם בצד שרת ניתן לבדוק מיהו הדומיין אשר מבצע את הבקשה על ידי קריאה של ה-origin header שם, בכל בקשה שנעשית, הדפדפן מכניס כערך את הדומיין אשר ממנו התבצעה הבקשה.

כך, במקרה שהשרת זיהה שדומיין לא מורשה ביצע את הבקשה, הוא יכול להחזיר עמוד ריק במקום התוכן המקורי ולא לבצע שום פעולות.

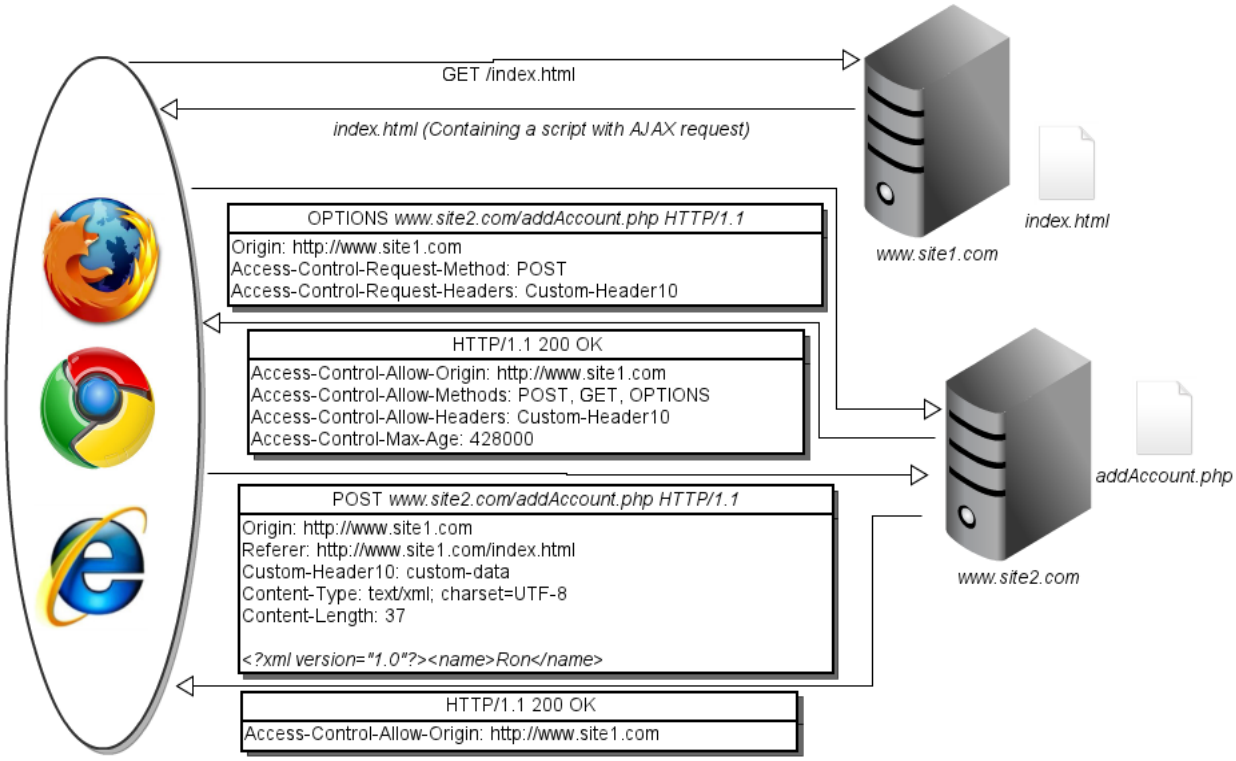
בקשת preflight

בנוסף לבקשות הרגילות, קיימת גם preflight request, אשר קודם שולחת בקשת OPTIONS בהתחלה בכדי לבדוק האם לדומיין מותר לבצע בקשת Cross-Origin, ורק אחר כך תבוצע הבקשה (בתנאי שלדומיין מותר לבצע אותה).

מנגנון זה קיים מכיוון שלפעמים עצם שליחת הבקשה אינה בטוחה.

הדפדפן מבצע את בקשת ה-preflight אוטומטית, ללא צורך בקוד נוסף אם מתקיים:

- הבקשה נעשית בשיטה אשר שונה מ-GET או POST, וגם אם הבקשה נעשית בשיטה אשר שונה מ-POST כאשר ה-Content-Type הנשלח יהיה שונה מ: application/x-www-form-urlencoded, multipart/form-data, או text/plain (או בעברית: אם הבקשה שולחת תוכן XML לשרת עם Content-Type של application/xml או text/xml אז הדפדפן יבצע את הבקשה בעזרת preflight).
- ישנם custom headers בבקשה (לדוגמה, ב-header יש שדה בשם "10custom-header"):



בעת ביצוע בקשה מן העמוד http://site1.com/index.html לעמוד http://site2.com/addAccount.php הדפדפן יבצע את הצעדים הבאים:

1. הדפדפן ישלח בקשת Option לעמוד http://site2.com/addAccount.php כאשר בנוסף הוא מצרף את הכותרות Access-Control-Request-Method ו-Access-Control-Request-Headers.
2. כעת, אם השרת מאשר לדומיין הזר לבצע את הבקשה עם הפרמטרים הללו הוא יענה בתגובה OK.
3. אם השרת אישר לדפדפן לבצע את הפעולה, הוא ישלח כעת את הבקשה המקורית.
4. השרת יקבל אותה וישלח בחזרה את תוכן העמוד.

דגל ה-Credentials

דגל זה מסמן האם ה-Credentials של המשתמש יצורפו לבקשה.

Credentials הם: עוגיות, אותנטיקציות HTTP שנעשו, ו-Certificates של הקליינט עבור SSL.

בברירת המחדל, דגל ה-Credentials כבוי. כלומר, הדפדפן בברירת המחדל אינו מצרף עוגיות לבקשות שנעשות ב-XDR אלא אם הוגדר אחרת, ובמקרה שכזה הוא שולח את הכותרות המתאימות אשר שואלות את השרת האם זה בסדר מבחינתו. במקרה שכן, ה-Credentials נשלחים עם הבקשה. אחרת הבקשה לא מתבצעת.

כותרת ה-Access-Control-Allow-Origin לא יכולה להיות '*' כאשר דגל ה-Credentials דלוק.

בעיות אבטחה של XDR

אפשור כללי

אפשור כללי זוהי הטעות הכי ברורה והכי קלה שמפתחים יכולים לעשות. כמו שכבר אמרנו, הכותרת 'Access-Control-Allow-Origin' אמורה להכיל רשימה של הדומיינים אשר יכולים לבצע בקשת Cross-Origin לעמוד, אך היא יכולה גם לקבל ערך wildcard ('*') אשר יאפשר לכל אתר לבצע בקשות Cross-Origin. ניתן לנצל זאת במספר דרכים:

- אם קיים איזשהו דף השייך לרשת פנימית (intranet) של ארגון כלשהו ואין אליו גישה מבחוץ. עובד בארגון אשר גולש בטעות לאתר זדוני מאחד ממחשבי החברה המחוברים לרשת הפנימית (וגם לחיצונית) עלול לחשוף את המידע ברשת הפנימית לאתר הזדוני, כאשר האתר הזדוני יבצע

בקשת XDR לעמוד ברשת הפנימית, וכך יוכל לקרוא את המידע בו ולשלוח אותו לבעל האתר הזדוני.

- מקרה דומה הוא למשל אם גוגל מאפשרת אופציה נוספת כאשר מחפשים בשירות החיפוש שלה (למשל מחיקת תוצאות מן החיפוש). אך אופציה זו קיימת רק כאשר מחפשים מתוך הרשת הפנימית. נניח בנוסף שגוגל מאפשרת לכל אתר לעשות לה XDR. כאשר עובד גוגל יגלוש לאתר זדוני, האתר הזדוני יוכל לבצע בקשת XDR ולמחוק רשומות מן תוצאות החיפוש של גוגל.

- נניח שמצאנו באתר מסוים SQL Injection בעמוד שניתן לבצע אליו XDR מכל דומיין. אם נשלוף את כל המידע בעמוד בעצמנו, הלוגים של השרת יצביעו עלינו באופן ברור וניתפס. אך נוכל ליצור קוד JS אשר מבצע את השליפה ושולח את התוצאות אלינו בעזרת XDR. את הקוד JS נשים באתר שלנו או שנעביר אותו דרך אתר עם XSS אל משתמש כלשהו. כאשר קורבן יבקר באתר בו שמנו את ה-JS, הדפדפן שלו ישלוף את הנתונים מן האתר וישלח אותם אלינו. בדיקות בלוגים של האתר יראו כי המשתמש הוא זה שביצע את השליפות, מכיוון שכותרות של בקשות בדרך כלל לא נשמרות בלוגים. הקורבן לא יכול לטעון שהמחשב שלו נפרץ, מכיוון שבדיקה על מחשב הקורבן לא תראה עקבות של malware או כל צורת השתלטות אחרת.

Include שגוי

נניח כי קיים מצב שבו לשרת יש קובץ common.php לדוגמה, אליו הוא מאפשר בקשות XDR, והמון קבצים בשרת עושים לקובץ זה Include. עלול להיווצר מצב שעמוד אשר לא התכוונו לתת הרשאות לבצע אליו בקשות XDR עושה Include לקובץ זה, אשר בתורו מוסיף את הכותרים שמאפשרים בקשות XDR, וכעת יהיה ניתן לבצע בקשות XDR אל העמוד וגם אל כל עמוד אחר אשר יבצע Include לקובץ זה.

פתרון אפשרי לבעיה זו הוא לבצע בדף common.php בדיקה מול "מילון" המכיל מידע על דפים אליהם אנו מאפשרים בקשות XDR, ועל פי "מילון" זה לשלוח את הבקשות המתאימות, ולא להסתמך על כך שעשו Include לקובץ זה.

הסתמכות יתר על כותר ה-Origin

יש לזכור שכותר ה-Origin נוסף על ידי הדפדפן, וכל אחד יכול להנדס בקשות-HTTP משלו עם כותר-Origin שיכיל כל ערך שירצה.

לדוגמה, הקוד הבא מכיל פרצה:

```
<?php
      if($_SERVER['HTTP_ORIGIN'] == "http://intranet.andlabs.org”){
      header('Access-Control-Allow-Origin: http://intranet.andlabs.org');
      //perform some important action
```

```
      print >>> sensitive internal information <<< ;
}
else{
    print >>> normal page <<< ;
}
?>
```

תוקף יכול בקלות לזייף את כותר ה-Origin שלו, ובעזרת החולשה לבצע פעולות חשובות ולקבל גישה למידע רגיש.

שמירת preflight cache לתקופות ארוכות

לביצוע בקשת preflight יש overhead, ולכן ביצוע הבקשה לוקח יותר זמן מאשר בקשת XDR רגילה. לכן יש אפשרות לשמור את תוצאת ה-preflight ב-cache, כך שבפעם הבאה שנרצה לבצע את אותה בקשה לא נצטרך לבצע את בקשת ה-preflight שוב ונוכל לפעול על פי הנתונים שיש ב-cache. שמירת נתוני ה-preflight ב-cache נעשית על ידי הוספת הכותרת 'Access-Control-Max-Age' לתשובת השרת לבקשת ה-preflight.

שמירה של cache לזמן ארוך יכולה לגרום לבעיית אבטחה. נניח שהשרת עדכן את מדיניות הגישה אליו. דפדפנים ישתמשו ב-cache שלהם המכיל את המדיניות הישנה ולא במדיניות החדשה. הזמן המומלץ הוא 30 דקות.

מתן אמון יתר בשותף

ישנם שני צדדים ב-XDR. הצד שמבקש את המידע והצד שמוסר את המידע. צריכה להיות כמות מסוימת של אמון בין השניים. האתר המבקש סומך על כך שהמידע שיקבל חזרה הוא אמין ובטוח, בעוד האתר שמוסר את המידע סומך על כך שהצד השני מוסמך לבצע את הבקשה ולקבל את המידע.

גם אם שני הצדדים הם לגיטימיים (וגם אם אותו אדם כתב את שניהם), יש להפקיד את כמות האמון המינימלית במידע שמספק הצד שני. זאת אומרת שעדיין עלינו לבדוק שהמידע אינו מכיל סקריפטים זדוניים, מידע מטעה (כאשר אפשר) וכו'. זאת מכוון שגם אם הצד השני לגיטימי, יכול להיות שתוקף הצליח לקבל אליו גישה וכעת משתמש בו בכדי לתקוף אותנו.

כעת נביא דוגמה דמיונית למקרים שכאלו: נניח ש-DigitalWhisper היה מציג את עדכוני הטוויטר של תוכנית החדשות האהובה עליכם בעמוד הראשי.

אתר DigitalWhisper יבצע בקשות XDR אל טוויטר בכדי לדעת מה היו העדכונים האחרונים, וכמובן שטוויטר מאפשר ל-DigitalWhisper לבצע בקשות אלו, שהרי אלו בקשות לגיטימיות. טוויטר בחזרה ישלח את העדכונים האחרונים שתוכנית החדשות פרסמה בקידוד HTML. בין השניים יש אמון, ולכן

DigitalWhisper לא מבצע בדיקות קלט ופשוט מדפיס את העדכונים לעמוד וטוויטר מאפשר ל-DigitalWhisper לבצע מגוון רחב של פעולות.

תחריש מספר 1 - לתוקף יש שליטה על טוויטר

כמו שאמרנו, מכיוון ש-DigitalWhisper סומך על טוויטר שישלח מידע מקודד HTML הוא מכניס את המידע לעמוד ללא בדיקות קלט. כעת, מכיוון שטוויטר בשליטת התוקף, הוא יכול לשלוח מידע זדוני המכיל תגיות HTML, וסקריפטים אשר DigitalWhisper יציג למשתמשים ובכך יסכן אותם.

תחריש מספר 2 - לתוקף יש שליטה על DigitalWhisper

נניח שטוויטר סומך על DigitalWhisper ולכן נותן לו גישה למגוון רחב של אופציות. DigitalWhisper יכולים לקרוא ציוצים, לשנות ציוצים, להוסיף משתמשים, למחוק משתמשים וכו'. כעת, מכוון ש-DigitalWhisper בשליטת התוקף, הוא יכול לבצע בקשות זדוניות על שרתי טוויטר בשם DigitalWhisper.

חשוב מאוד שהצד המבקש יוודא את תקינות התגובה שהוא קיבל, ושהצד המוסר יחשוף רק את האפשרויות הכרחיות עבוד הצד השני. גם אם תוקף קיבל שליטה על צד אחד, אין זה אומר שהוא צריך לקבל שליטה על הצד השני באופן אוטומטי.

DDoS בעזרת XDR

יש לזכור כי קיימים שני סוגים של בקשות XDR: בקשה פשוטה ובקשה עם preflight. ישנם רק כמה מקרים בהם משתמשים ב-preflight, ולרוב הבקשות משתמשים במנגנון הפשוט.

נזכיר, במנגנון הפשוט אנו שולחים בקשה לשרת, מקבלים את המידע ואיתו את הכותרים, ואנו בודקים על פי הכותרים האם לדומיין אשר יצר את הבקשה מותר לראות את המידע. אם כן, ניתן לו גישה, אחרת לא.

לעומת זאת, במנגנון ה-preflight הדפדפן ישלח בקשת Options עם הכותרים המתאימים, כך שהשרת יידע באיזו שיטה ועם אילו כותרים הבקשה האמיתית תתבצע, מקבל תשובה מן השרת עם הכותרים המתאימים ובודק האם הבקשה המקורית יכולה להתבצע. נזכור גם שבעת בקשת XDR פשוטה (ללא preflight) השרת מבצע את כל החישובים הנחוצים בכדי ליצור את התגובה (ומתייחס לבקשה כבקשה לגיטימית). זהו רק הדפדפן אשר מחליט האם הבקשה היא באמת לגיטימית או לא.

ניתן להשתמש בזה בכדי לבצע התקפות DDOS בשכבת האפליקציה.

נניח לדוגמה שקיים עמוד המבצע פעולות חישוביות רבות ולבסוף שולח בחזרה את תוצאות חישוביו. בעת ביצוע XDR לעמוד, העמוד יבצע את החישובים וישלח את התוצאות, רק לאחר מכן הדפדפן יחליט האם לאפשר לעמוד שביצע את הבקשה לראות את תוכן התשובה או לא. אך זה כבר לא משנה, מכיוון וגרמנו לביצוע הפעולות על מחשבי השרת.

כעת נוכל לחפש עמודי אינטרנט המכילים Persistent XSS ולהזריק לתוכם קוד JS אשר מבצע את בקשת ה-XDR הזו.

בעוד אנשים ייכנסו לאתרים אלו, הדפדפנים שלהם יבצעו את בקשות ה-XDR, ובכך בעצם יעמיסו על מחשבי האתר.

היינו יכולים להשיג את אותה התוצאה בעזרת החדרת תג של תמונה בעל תכונת src מתאימה לעמודי האינטרנט המכילים Persistent XSS, אך ב-XDR נוכל להשתמש גם במקרה של בקשת POST/DELETE/PUT. היינו יכולים גם להטמיע iframe עם src מתאים או לשלוח טופס (כאשר צריכים לשלוח את הבקשה ב-POST לדוגמה) מתוך iframe, אך XDR נותן ביצועים טובים יותר, בעזרת XDR ניתן לשלוח כ-100,000 בקשות בדקה (בכרום וספארי).

פתרון חלקי לבעיה זו הוא לבצע בדיקות על כותרת ה-Origin בצד השרת, וכאשר ה-Origin אינו אתר שאנו רוצים שיבצע לנו XDR לא נבצע שום עיבוד בצד השרת. יש לזכור כי למרות שהתוקף יכול לזייף בעצמו את כותרת ה-Origin, הוא לא יכול לעשות זאת אצל משתמשים תמימים הנכנסים לאתר הזדוני, ולכן, מכוון שב-DDOS עסקנו, כל עוד אם התוקף בעצמו ישלח את הבקשות עם ה-Origin המזויף זו כבר לא תהיה מתקפת DDOS, לפחות לא מאסיבית (יכול להיות שלתוקף יש כמה מחשבים) אלא מתקפת DOS רגילה (או DDOS קטנה) שספק שתשפיע על השרתים שלנו.

בנוסף, פתרון זה רק מקשה עלינו לנצל את הבעיה, שכן אם נשלחות בקשות רבות מידי גם רכיבי הרשת בדרך אלינו יהיו עמוסים (כמו Switchs, Routers, וכו').

אפשרויות ניצול חדשות למתקפות ישנות

למרות שחלק מהתוספות החדשות ב-HTML5 אינן גוררות בעיות אבטחה חדשות, הן יוצרות וריאציות חדשות לבעיות ידועות.

XSS

באמצעות תגים חדשים

מנגנון הגנה יעיל על מנת למנוע (או לפחות לצמצם) מתקפות XSS, כש-encoding לא אפשרי (כאשר מקבלים תוכן עשיר מה-user) זהו מנגנון של פילטרים.

כמו בכל מנגנון פילטרים, יכולה להיות רשימה לבנה ורשימה שחורה. לצערנו (או שלא), רוב המנגנונים משתמשים בגישה של רשימה שחורה.

פילטרים שחוסמים למשל תגים כמו 'script>' או 'img>' וכו', על מנת למנוע מאיתנו להכניס סקריפט שלנו שירוץ. אך כבר דברנו על התגים החדשים של HTML5, ובעזרתם נוכל לעקוף מנגנוני פילטור אלו, ולהריץ את הסקריפט הנחשק.

למשל, תגי המולטימדיה החדשים עבור video ו-audio, להם יש מאורע של מה לעשות בזמן שגיאה (onerror).

וכמו תמיד, שגיאות אנחנו יכולים ליצור גם בעצמנו. כאשר נכניס src לא לגיטימי, מאורע ה-onerror של תג ה-video\audio יתרחש. לדוגמה:

```
<video src=”0” onerror="javascript:alert(1)"></video>
```

```
<audio src=”0” onerror="javascript:alert(1)"></audio>
```

כאשר שמים כמה תגי source מוקפים בתג video/audio, הדפדפן יעבור על כל רשימת ה-source-ים עד שיימצא source בו הוא תומך ואותו יפעיל. גם לתג ה-source קיים מאורע onerror אשר יקרה כאשר ה-source הספציפי לא נתמך, ולכן עוד דרך לגרום למאורע ה-onerror היא לגרום לכך שכל הפורמטים של תגי ה-sources ששמנו יהיו לא נתמכים, לדוגמה:

```
<video><source onerror="javascript:alert(1)">
```

```
<audio><source onerror="javascript:alert(1)">
```

כעת נעלה רמה אחת למעלה, פילטרים אשר חוסמים את התווים '>' '<'. לא נוכל להזריק תגיות, אך נניח (הנחה סבירה יחסית) כי הטקסט שנכנס כקלט נכנס כערך של תכונה של תג, ולכן נעבור לאפשרויות הבאות.

באמצעות תכונות חדשות

נסתכל לדוגמה על הקוד הבא:

```
<input type=’text’ value=’someUserInput’ />
```

לפני HTML5, על מנת לנצל חולשת XSS זו, היינו משתמשים למשל במאורע onmouseover, וכאשר המשתמש היה עובר עם הסמן על מקום זה, היה מורץ הסקריפט:

```
someUserInput = ' onmouseover='alert("XSS HTML4 version")
```

באמצעות HTML5, המספק לנו תכונה חדשה לתג ה-input, תכונת autofocus שמטרתה לשים את שדה הטקסט בפוקוס עם טעינת הדף אוטומטית, אנו מקבלים כוח רב יותר, ונוכל בקלות לגרום לסקריפט לרוץ ישר עם טעינת הדף ללא צורך במעבר ידני של הגולש עם הסמן, כלומר ללא אינטראקציה עם הקורבן:

```
someUserInput = ' onfocus='alert("XSS HTML5 version")'
autofocus='autofocus
```

ציינו קודם כי תכונת ה- autofocus לא חייבת לקבל ערך. כלומר, השורה הבאה שקולה:

```
someUserInput = ' onfocus='alert("XSS HTML5 version")' autofocus
custom='
```

שימו לב ששמנו תכונת custom, שהיא custom attribute, סתם בכדי להיפטר מהגרש האחרון שסוגר את הערך של value בלי שגיאות.

כמו כן:

```
someUserInput = ' onfocus=alert("XSS HTML5 ++ version") autofocus
custom='
```

(לא חייבים להקיף את ערכי התכונות בגרשיים)

מה שיקרה, זה ששמנו את השדה בפוקוס אוטומטי, וכל עוד הוא בפוקוס, ירוץ ה-alert.

משום שאנחנו לא רוצים ליצור אינסוף דיאלוגים נוספים, כלומר שהסקריפט ירוץ כל עוד השדה בפוקוס, ננצל את העובדה שרק שדה אחד יכול להיות בפוקוס בזמן נתון. כאשר שני תגי input מכילים autofocous, הדפדפן בסופו של דבר יעביר את הפוקוס מהראשון שהוגדר לשני. כלומר, הראשון בשלב הזה 'יפסיד אותו'. מזכיר לנו- event כלשהו? (אהמ-onblur)

```
someUserInput = ' onblur='alert("I love Israel")' autofocus /><input
autofocus custom='
```

ונקבל הרצה פעם אחת ללא אינטרקציית המשתמש. כמובן שאם הוא יחזור לפוקוס על השדה ויצא ממנו שוב אז הסקריפט שוב ירוץ. אך נוכל בהרצת הסקריפט בפעם הראשונה לדאוג לכך שמצב כזה לא יקרה.

מה קורה אם יש פילטור ספציפי על ארועים (events), כמו למשל onerror, onload ואירועים נפוצים אחרים, וכעת לא נוכל להשתמש בהם?

לשם כך נשתמש באירועים החדשים שבאים יחד עם HTML5 כמו למשל onforminout ו-onformchange.

לדוגמה:

```
<form id=formid onforminout=alert(1)><input></form>
```

```
<button form=formid onformchange=alert(2)>
```

ואפילו אם נעלה עוד שלב, פילטור חזק יותר באמצעות ביטוי רגולרי אשר מזהה תחילית של 'on', לאחריו מילה ואז הסימן =. כלומר, הביטוי הרגולרי:

```
on\w+=/i
```

פילטר חזק, אך גם זה לא ימנע מאיתנו להשתמש בתכונות החדשות של HTML5 אשר יעברו את הפילטר הזה בקלות, כמו למשל:

```
<form id="formid" /><button form="formid"
formaction="javascript:alert(1)">
```

עוד וקטורי תקיפה חדשים תוכלו למצוא ב- http://www.html5sec.org

ביצוע התקפות CSRF בעזרת XDR

מתקפת CSRF - Cross Site Request Forgery, מתבססת על העובדה שהמשתמש סומך על הדפדפן (וגם על האתר הנתקף). זו נגרמת מכיוון שבכל פעם שהדפדפן שלח בקשה לעמוד מסוים הוא מצרף את ה-Credentials (עוגיות, SSL Certificate, HTTP Authentication) אשר מזוהות עם אותו מקור לבקשה.

כך לדוגמה אם היינו מחוברים לאתר של הבנק שלנו ובאותו הזמן היינו נכנסים לאתר זדוני המכיל את תג התמונה הבא:

```
<img src="www.bank.com/transfer.php?amount=1000&to=EvilBankAccount" />
```

הדפדפן שלנו היה מבצע בקשת GET בכדי לקבל את התמונה, ובקשה זו הייתה גורמת להעברה כספית אל חשבוננו (שכן באותו הזמן הבנק זיהה אותנו על פי ה-Credentials שהדפדפן צירף).

פתרונות לבעיה זו הגיעו ממעבר לשיטת ה-POST (שלא פותרת את הבעיה לגמרי - בעזרת יצירת טופס מוחבא ב-iframe ושליחתו אל אתר הבנק ניתן לשלוח בקשה ב-POST, אך זה טיפה יותר מסורבל) ועד שליחת Challange (או Token רנדומלי) בחזרה עבור כל פעולה שכזו.

בעזרת XDR, ניתן לבצע כעת CSRF גם בבקשות DELETE ו-PUT!

לכן שרתים אשר לא מוגדרים טוב ומאפשרים בקשות שכאלו ללא אותנטיקציה (ואפילו עם אותנטיקציה) עלולים להיות פגיעים, תוקף יהיה מסוגל לנצל את ה-Credentials של משתמש בעל הרשאות כתיבה ומחיקה ולבצע פעולות אלו במקומו, וכך להשתיל, לשנות ולמחוק עמודים מן האתר. כדי שזה יקרה, על השרת לאפשר בקשות XDR עבור שיטות אלו באופן ידני (זה לא נעשה אוטומטית, ובברירת המחדל זה לא מתאפשר). אך נזכור ש-XDR יכול לבצע בקשות גם לרשת הפנימית, ושם יש סיכוי גדול יותר למצוא פרצות אבטחה שכאלו, וקיבלנו איום ממשי.

לסיכום

במאמר זה סקרנו והצגנו חלק קטן משלל הפיצ'רים והתוספות ש-HTML5 מאפשר לנו.

ראינו כיצד פיצ'רים חדשים עלולים ליצור בעיות אבטחה חדשות, לחדש אפשרויות ניצול לבעיות ישנות וגם כיצד פיצ'רים חדשים באים לעזרתנו בניסיון לפתור בעיות אבטחה ישנות (sandbox), על אף שגם פתרונות אלו עדיין לא הגיעו לכדי שלמות.

ישנן הרבה תוספות ושינויים שהצטברו ויצרו את מצב האינטרנט כיום (וכנראה מצב האינטרנט בעתיד), ברובם לא נגענו, ובמאמרים הבאים נמשיך ונסקור את שאר התוספות שנעשו ל-HTML5, ונתעסק בשאלות חשובות כמו: האם מנגנון ה-Drag&Drop ייצור גל מתקפות חדשות מסוג Clickjacking, כיצד ניתן להשתמש ב-History API בכדי לבצע מתקפות פישינג.

נמשיך ונסקור גם את התוספות השונות שנעשו מסביב ל-HTML5, וגם הן מהוות חלק בלתי נפרד מהשינוי המסיבי שהאינטרנט עובר כיום, כמו Websockets, Geolocation, Client Storage וכן הלאה, שהן אמנם חלק מהחידושים שבאים יחד עם HTML5, אך בניגוד למה שנהוג לחשוב, הן אינן חלק מ-HTML5. גם פה נענה על שאלות חשובות כמו: האם מנגנון ה DatabaseStorage יהיה חשוף למתקפות מסוג SQL Injection? מה ההבדל בינו ובין Session Storage, Database Storage או Global Storage? ולמה אנחנו צריכים כל כך הרבה שיטות לשמור מידע בצד הלקוח?

נראה גם כיצד ניתן לבצע Port Scanning בעזרת בקשות XDR ו-WebSockets, ונציג דרכים חדשות לנצל משתמשים תמימים לצרכינו הזדוניים, כל אלו ועוד במאמרים הבאים.

קצת קישורים

- http://www.andlabs.org
- http://code.google.com/p/html5security
- https://developer.mozilla.org/en/HTML/HTML5
- http://mashable.com/2011/04/29/html5-web-security
- http://www.softwaremag.com/focus-areas/security/featured-articles/what-does-html5-mean-for-security

קוואבאנגהההה!!!!

על הכותבים

לירן בנודיס ואלעד גבאי, בני 19, סטודנטים בשנה ד' בהנדסת מחשבים באוניברסיטה העברית. סטודנטים במעבדת האינטרנט של האוניברסיטה העברית:

http://internetlabhuji.wordpress.com

תודות

תודה ל:

- ישראל חורז'בסקי (SRO), אפיק קסטיאל (cp77fk4r) ואוהד אסולין.

דברי סיום

בזאת אנחנו סוגרים את הגליון ה-24 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 (או בעצם - כל יצור חי עם טמפרטורת גוף בסביבת ה-37 שיש לו קצת זמן פנוי [אנו מוכנים להתפשר גם על חום גוף 36.5]) ואנשים המעוניינים לעזור ולתרום לגליונות הבאים. אם אתם רוצים לעזור לנו ולהשתתף במגזין Digital Whisper - צרו קשר!

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

ועכשיו בדיחה (נראה אם ניר ישים לב אליה בעריכה):

> מתמטיקאי פיזיקאי וביולוג יושבים במסעדה ומסתכלים מהחלון, פתאום עוברת אישה ונכנסת לבניין ממול, לאחר עשר שניות יוצאים מהבניין שני גברים. הפיסיקאי אומר: "לא הגיוני... כניראה שטעינו במדידה", אומר הביולוג: "מה זאת אומרת לא הגיוני? הם התרבו!", המתמטיקאי אומר: "מספיק שאדם נוסף יכנס לבניין והוא יהיה ריק!". ☺

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 ספטמבר 2011.

אפיק קסטיאל,

ניר אדר,

31.08.2011