Technologic Papers

"כמה גרוע זה כבר יכול להיות" - מחקר חולשות על נתב ביתי

מאת 0x3d5157636b525761

הקדמה

במאמר זה אחשוף מחקר עצמאי אשר ביצעתי ובמסגרתו מצאתי מספר חולשות הרצת קוד על ראוטר Netgear DGN2200. נתבים אלו נפוצים במיוחד בארץ עקב כך שחברת בזק מחלקת / חילקה אותן כחלק מחבילות השירות שלה. אני מציין בכוונה כאן את חברת בזק מכיוון שחקרתי firmware שלהם ולא firmware של Netgear, אף על פי שככל הנראה גם נתבים "טבעיים" של Netgear גם פגיעים בגרסאות מסויימות.

עד כמה שאני מבין, החולשה תוקנה כבר על ידי "בזק", אך חשוב לציין שישנם המון נתבים לא מעודכנים ולפיכך צפוי שמספר גדול של ראוטרים עדיין פגיע.

אציין כי חברת בזק שיתפה פעולה באופן מלא (בניגוד לחברת Netgear שניסתה בעיקר לזרוק אחריות), ומודעים לשחרור מאמר זה.

השתלשלות האירועים

- 10.02.2016 - גילוי 3 חולשות ראשוניות המצויינות במאמר זה (חולשות #1, #2 ו-#4).
- 10.02.2016 - פנייה אל חברת Netgear ומסירת הפרטים הטכניים, כולל PoC.
- 12.02.2016 - תגובה של חברת Netgear שלצערי גלגלו אחריות אל "בזק".
- 13.02.2016 - מציאת חולשה נוספת (חולשה #3) ודיווח נוסף עליה.
- 14.02.2016 - פנייה אל חברת "בזק".
- 21.02.2016 - תגובה ראשונית של "בזק" ומסירת פרטים טכניים.
- 03.03.2016 - תיקון ראשוני של חולשה #2 של בזק.
- 20.03.2016 - שחרור קוד ה-PoC לאינטרנט תוך מעקב צמוד של בזק.

תחילת המחקר

מטרת העל היא למצוא דרך להריץ קוד על ראוטר מתוך רגל ה-WAN. מטרה זו תתבצע במספר חלקים, כאשר החולשות "מועמסות" אחת על גבי השנייה. המחקר שלנו יתחיל בשאלה: "מה ניתן לעשות מתוך ה-LAN?", ולאחר מכן יתפשט לשאלה: "האם ניתן להרחיב את היכולות אל ה-WAN?".

באופן טבעי, הדבר הראשון שבוצע הוא סריקת פורטים שגרתית, שבוצעה על ידי nmap:

```
root@godmode:~/research/netgear_from_their_ws# nmap 10.0.0.138

Starting Nmap 6.40 ( http://nmap.org ) at 2016-02-17 06:44 IST
Nmap scan report for 10.0.0.138
Host is up (0.0090s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
53/tcp   open  domain
80/tcp   open  http
5000/tcp open  upnp
MAC Address: 4C:60:DE:29:1D:7F (Netgear)

Nmap done: 1 IP address (1 host up) scanned in 8.43 seconds
root@godmode:~/research/netgear_from_their_ws#
```

הכיוון שלי היה לבדוק את המימושים של כל השירותים הללו, ובאופן טבעי התחלתי עם ה-http. ממשק ה-http מאפשר ניהול נוח של הראוטר, לאחר סיפוק שם משתמש וסיסמא תחת HTTP basic authorization. מכיוון שמדובר ב-http plaintext, תוקף מתוך ה-LAN יכול היה לנסות ולהסניף את תעבורת ה-Administrator ומתוכה לחלץ את הסיסמא בקלות, אך לעת עתה נתעלם מבעיה זו ונניח שלא ניתן לבצע הסנפה שכזו. אם כן, השלב הבא במחקר הוא לחשוף את ה-filesystem.

חשיפת מערכת הקבצים

את גרסת ה-firmware האחרונה הורדתי מהאתר של בזק - ניתן גם היום להוריד גרסאות firmware שונות בהתאם לראוטר שיש לכם. ה-firmware הוא קובץ יחיד המכיל בתוכו את כל מה שהראוטר צריך, וביניהם מערכות קבצים. הכלי הטוב ביותר לכך הוא binwalk:

```
root@godmode:~/research/netgear_bezeq_firmware# binwalk -e ./*

DECIMAL       HEX             DESCRIPTION
--------------------------------------------------------------------------------
58            0x3A            JFFS2 filesystem, big endian

root@godmode:~/research/netgear_bezeq_firmware#
```

הידד! נראה שמצאנו מערכת קבצים מסוג JFFS2, שהיא מערכת קבצים די פופולרית עבור embedded.

הערה: לעיתים ניתקל במערכות קבצים לא סטנדרטיות ונצטרך לבצע מעט אבחון: טריק פופולרי של מתכנני embedded הוא לשנות ערכי *magic* ממערכת הקבצים על מנת שכלים כגון binwalk לא יזהו אותם. כמובן שישנם גם טריקים אחרים.

השלב הבא הוא כמובן לבצע mount ולבחון את התוכן.

היכן ה-cgi שלי?

ישנם כמה מקומות מעניינים לבחון:

1. סקריפטי אתחול - בדרך כלל תחת etc/init.d או etc/rc.d.
2. קונפיגורציות - בדרך כלל כל קובץ שמסתיים ב-conf, במיוחד תחת etc.
3. תיקיית www עבור ה-http שלנו.

אמנם ישנו סקריפט אתחול תחת init.d, אך הוא לא עושה דברים מיוחדים. לצערי, גם הקונפיגורציות לא היו משהו... והכי גרוע - בתיקיית www מצאתי רק קבצי gif ו-html! היכן לוגיקת צד השרת?

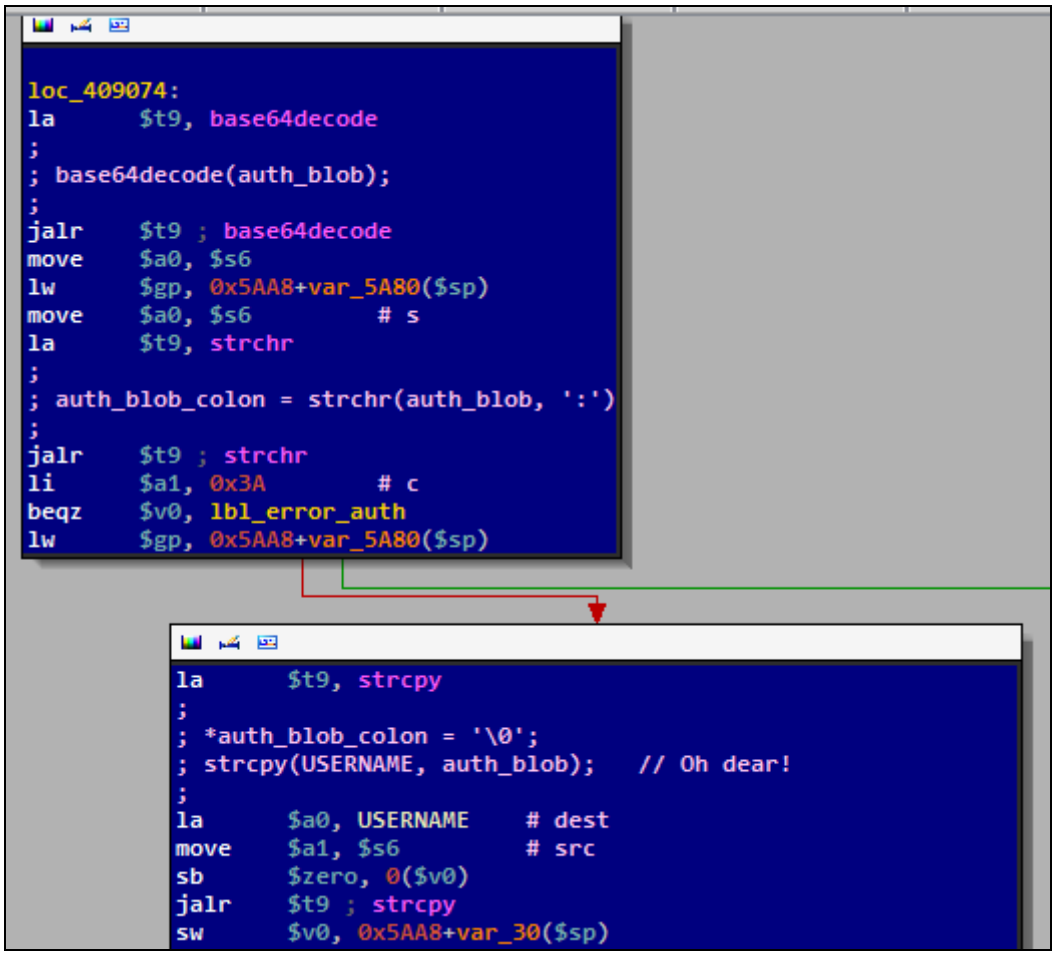
נדמיין ממשק http - תהליך ("httpd") פתח socket אשר מאזין על פורט 80, והוא משרת בקשות. תהליך זה אחראי על הגשת דפי ה-html מתוך תיקיית ה-www ואחראי גם על לוגיקת ה-cgi... מה שאומר שאם ה-cgi לא נמצאים בתוך מערכת הקבצים, הם כנראה "אפויים" לתוך תהליך כלשהו, והימור טוב הוא ה-httpd. נבחן אותו:

```
root@godmode:~/research/netgear_from_their_ws/squashfs-root/usr/sbin# strings httpd | grep
"\.cgi" | head
wsw_summary.cgi
fw_check.cgi
upgrade_check.cgi
strtblupgrade.cgi
dpf_backup.cgi
upgrade.cgi
fwLog.cgi
fwEmail.cgi
pforward.cgi
fwSchedule.cgi
root@godmode:~/research/netgear_from_their_ws/squashfs-root/usr/sbin#
```

אכן נראה שלפחות שמות ה-cgi "אפויים" לתוך ה-httpd. מצאתי גם קוד html ושלל ירקות, מה שאומר ששווה לחקור את ה-httpd יותר לעומק!

מחקר ה-httpd

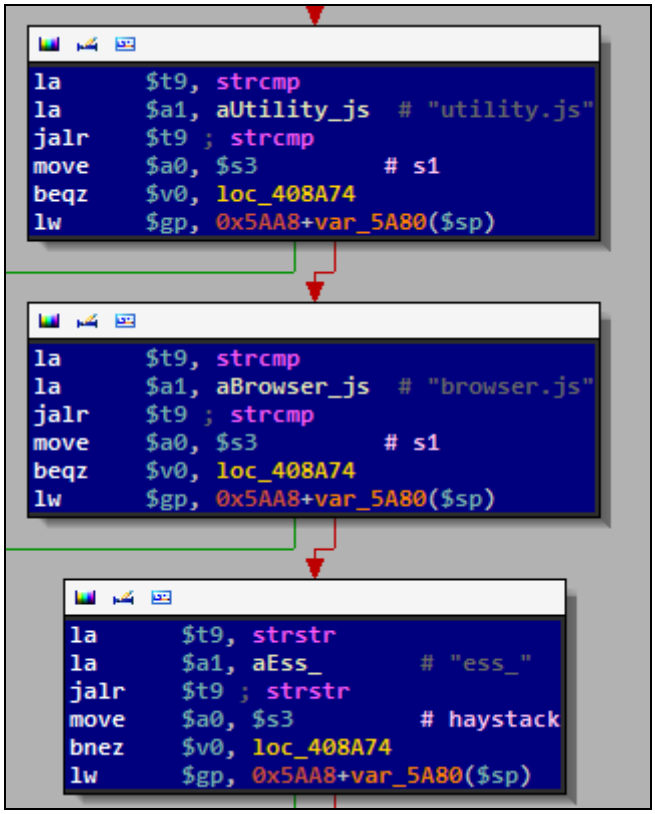
ה-httpd הוא קובץ ELF שמקומפל לארכיטקטורת MIPS (כן, זה מה שהראוטר שלכם מריץ). למזלנו, IDA יודעת להתמודד יפה עם MIPS. הדבר הראשון שמציק בממשק ה-http בראוטר הוא שהוא דורש שם משתמש וסיסמא לכל דף (גם דפים לא קיימים!), ולכן היינו רוצים לבחון את הלוגיקה שמטפלת ב-basic authorization. למצוא קטע קוד שמטפל בכך היה טריוויאלי (xref פשוט), ולצערנו מתגלה פונקצייה די גדולה. החלטתי להתמקד אך ורק בפרסור שם המשתמש והסיסמא:



בשלב זה בקוד, "auth_blob" מחזיק את ה-base64 שנשלח כחלק מתוך ה-HTTP headers ומייצג את שם המשתמש והסיסמא. לאחר Base64 decoding (שנעשה in-place) הקוד מחפש נקודותיים, שאמורות להפריד בין שם המשתמש והסיסמא. אם אכן נמצאו נקודותיים, מחליפים אותן ב-NULL terminator (פעולה סבירה) ולאחר מכן משתמשים ב-strcpy כדי להעתיק את שם המשתמש אל באפר גלובאלי (הו, לא!). אורך הבאפר הוא 20 בתים, וסיפוק של שם משתמש ארוך מדי מתחיל לדרוס גלובאליים אחרים. אם כן, חולשה #1: memory corruption בעת סיפוק שם המשתמש והסיסמא מתוך ה-LAN.

בפועל, לא הצלחתי להגיע להרצת קוד מתוך הבאג הזה, אז לעת עתה נזכור שהוא קיים ונמשיך הלאה.

לאחר שמיצינו את חקירת הטיפול בשם המשתמש והסיסמא, הכיוון היה ללכת אחורה בפונקציה - לראות אילו code paths מביאים אותנו למצב של דרישת אותנטיקציה מראש. מהר מאד גיליתי את הלוגיקה שמסננת דפים מאותנטיקציה, הנה חתיכה קטנה ממה שהולך שם:



הלוגיקה מסננת דפים על ידי strcmp של הדף המבוקש (כפי שהגיע ב-HTTP GET), כאשר אם יש התאמה אז מוותרים על אותנטיקציה. חדי העין שביניכם הבחינו בוודאי ב-strstr בבלוק האחרון - והוא אכן הבאג כאן. שימו לב שבמקרה של strstr, מספיק שיהיה רשום איפשהו (למשל, בתוך משתנה get) את ה-string המבוקש על מנת לדלג על אותנטיקציה.

הערה: אף על פי שזה עלול להיראות כמו backdoor, בפועל זה יותר דומה לבאג. ישנה לוגיקה דומה שמחפשת סיומות של jpg (שאותן ניתן לנצל באופן דומה) וכנראה סתם המתכנת היה גרוע.

אם כן כל מה שעלינו לעשות הוא לספק את המחרוזת הרצויה בתור משתנה GET. חולשה #2: מעקף אותנטיקציה מתוך ה-LAN על כל דף שמוגש על ידי הראוטר.

הרחבת היכולות מתוך ה-LAN

אז, מה נותנת לנו היכולת לעבור אותנטיקציה מתוך ה-LAN? כמובן, ניתן לעשות דברים מאד מרושעים כמו לעדכן firmware, לנתק משתמשים ולמעשה לשלוט בכל אספקט של ה-LAN, אבל אנחנו מעוניינים להגיע למצב של הרצת קוד באופן חשאי. האם אנחנו יכולים לעשות זאת? הרצת קוד נעשית בדרך כלל על ידי קריאה אל system, ולכן נחפש xref-ים אל system בתוך ה-httpd. מהר מאד נגיע אל תוצאה נראית מבטיחה:

```
move    $a2, $s2
jalr    $t9 ; websGetVar
la      $a1, aPing_ipaddr  # "ping_IPAddr"
lw      $gp, 0x228+var_218($sp)
addiu   $s0, $sp, 0x228+var_110
la      $t9, sprintf
lui     $a1, 0x49
lui     $a3, 0x49
move    $a2, $s2
la      $a3, aTmpDiag_conf  # "/tmp/diag.conf"
la      $a1, aPingC4SS   # "ping -c 4 %s > %s"
jalr    $t9 ; sprintf
move    $a0, $s0         # s
lw      $gp, 0x228+var_218($sp)
la      $t9, system
jalr    $t9 ; system
move    $a0, $s0         # command
lw      $gp, 0x228+var_218($sp)
lui     $a0, 0x49
la      $t9, sendPage2Client
move    $a1, $s1
jalr    $t9 ; sendPage2Client
la      $a0, aDiag_ping_htm  # "DIAG_ping.htm"
lw      $ra, 0x228+var_4($sp)
move    $v0, $zero
lw      $s2, 0x228+var_8($sp)
lw      $s1, 0x228+var_C($sp)
lw      $s0, 0x228+var_10($sp)
jr      $ra
addiu   $sp, 0x228
 # End of function sub_41CC90
```

זהו אזור קוד המטפל בדיאגנוסיטקה של הראוטר, וספציפית יודע לעשות ping אל host לבחירת האדמין.

באופן פשוט:

```
ping -c 4 %s > %s
```

מורחב על ידי sprintf ואז נשלח ישירות אל system. הקובץ שאליו תכתבנה התוצאות הוא קבוע, אבל ה-host שעליו יתבצע ה-ping נלקח מתוך פרמטר ה-GET בשם ping_IPAddr, בלי שום וידוא עליו! זה אומר שאפשר להשתמש ב-pipe או ב-backticks והם יפורשו על ידי ה-system.

הנה דוגמא משעשעת:

```
GET /ping.cgi?IPAddr1=8&IPAddr2=8&IPAddr3=8&IPAddr4=8&ping=Ping&ping_IPAddr=1|echo%201%
3E/tmp/pwned HTTP/1.1
Host: 10.0.0.138
Connection: keep-alive
Authorization: Basic QWRtaW46QWRtaW4=
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/48.0.2564.109 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,he;q=0.6

HTTP/1.0 200 OK
Content-length: 2248
Content-type: text/html
```

הערה: אני יודע שאפשר לראות כאן את שם המשתמש והסיסמא שלי. היא שונתה (ובכל מקרה את ה-firmware שלי פצ'פצ'תי), אז אל תטרחו...

אם נבדוק האם נוצר קובץ בשם pwned בתוך tmp, נגלה כי אכן נוצר כזה -- הצלחנו להריץ קוד על הראוטר! מכיוון ש-httpd רץ ב-root (ולמעשה, הכל רץ ב-root בראוטר), אנחנו יכולים לבצע הכל! חולשה #3: הזרקת פקודה מתוך ה-LAN. כמובן, שילוב של חולשה זו עם מעקף האותנטיקציה נותן למשתמש ב-LAN להריץ איזה קוד שבא לו על הראוטר באופן שקט וחשאי.

אם כן, השתלטנו לגמרי על ה-LAN, והמטרה הקרובה היא לראות האם אפשר להרחיב את שליטתנו ל-WAN.

אז מה קורה ב-WAN?

אז, משתמש מתוך ה-LAN יכול לעשות מה שבא לו על ידי בקשת HTTP פשוטה לראוטר. המטרה היא למצוא משתמש כזה שיעשה בשבילנו את העבודה - ולגרום לו לבצע את העבודה מתוך ה-WAN. נניח שמישהו בתוך ה-LAN גולש החוצה עם http (לאתר של DigitalWhisper, לדוגמא). זה אומר שאם תוקף יושב בין הראוטר ובין שרת ה-http (כלומר, התוקף הוא Man In The Middle) - התוקף יכול לשנות את תשובת השרת. בפרט, התוקף יכול להזריק iframe חבוי שיבצע פניית http אל השרת (עם כל הפרטים), והקורבן שלנו (שיושב בתוך ה-LAN) יגש אל הראוטר ויבצע עבורנו את העבודה! תקיפה זו, המוכרת בתור CSRF, כבר הוזכרה במספר גליונות קודמים, ולכן לא אכביר במילים.

שילוב החולשות

למעשה, כרגע יש לנו דרך מרוחקת להריץ קוד על ראוטר בהינתן man in the middle - הזרקת iframe שיבצע את הפקודה הרצויה (על ידי כלי ה-ping) ויעקוף את בקשת האותנטיקציה (על ידי הפעלת החולשה שמצאנו בהתחלה).

כתבתי קוד הדגמה (ב-html) שמדגים את שילוב החולשות הללו ויוצר קובץ על הראוטר. הוא פותח גם את ה-telnetd על הראוטר (לתוך ה-LAN), כך שיהיה קל לראות האם ההתקפה הצליחה או לא.

מסקנות

1. את רוב החולשות כאן ניתן היה למנוע בקלות. שימוש ב-strstr ו-strcpy היה אמור כבר לחלוף מן העולם, אך לצערנו עדיין קיים, במיוחד ב-embedded. צפו לראות עוד הרבה כאלה עם כל ה-hype של ה-"Internet of Things".
2. חולשת ה-command injection אף היא די פרימיטיבית, וניתן היה להמנע ממנה על ידי סניטיזציה פשוטה של הקלט.
3. התקפות מסוג CSRF אמורות להיות בראש של כל מתכנת צד-שרת, וגם כאן הייתה פאשלה לא קטנה.

קוד ה-PoC, הסברים קצת יותר מעמיקים יותר ותמונות של החתולה שלי ניתן למצוא בבלוג שלי, שיתעדכן בערך אחת לחודש, בכתובת: https://securitygodmode.blogspot.co.il.