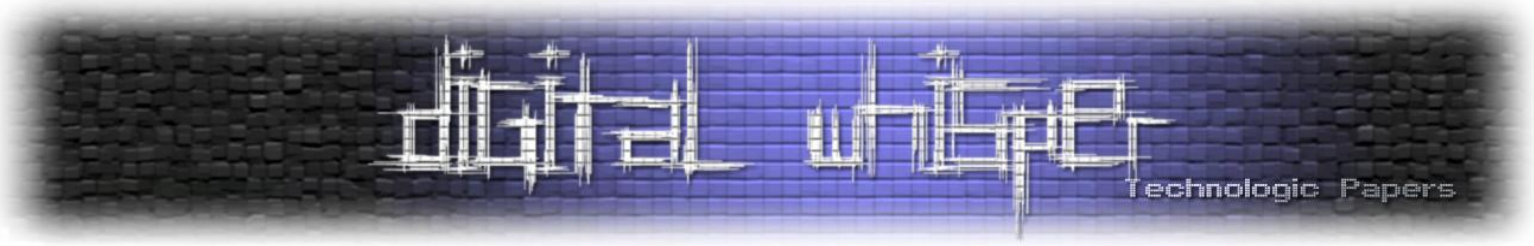


Protocol Tunneling

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 ביננו 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:

- מצב רגיל: רק הפאקטה עצמה / המטעד (הידוע בכינויו "שלם-טען") מוצפנים.
- מצב Tunnel: 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 ☺

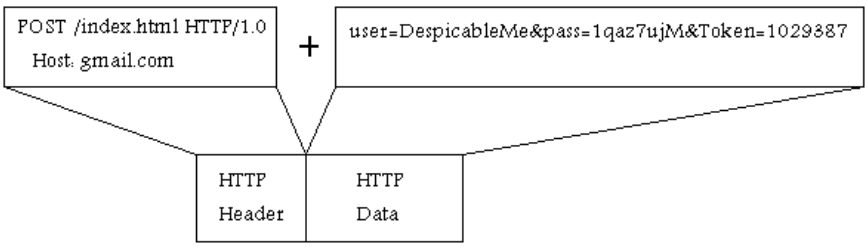
איך זה עובד? או מה בדיוק עושה כלי ה-PTUNNEL?

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:

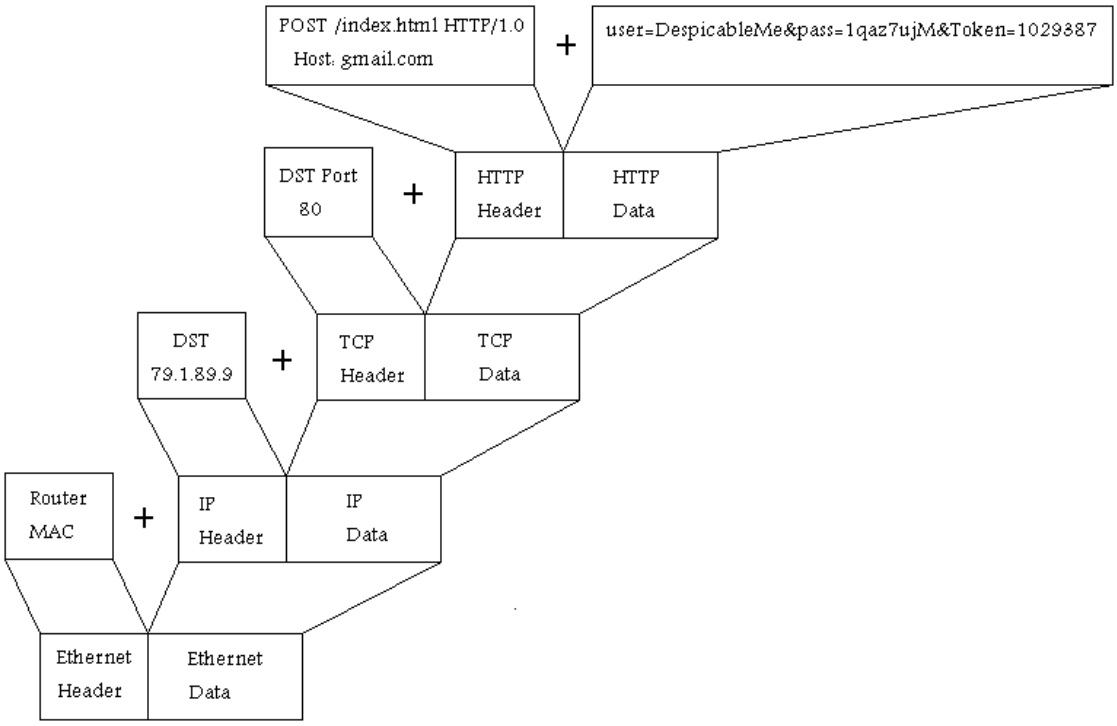


הדפדפן, שאחראי גם על השכבה הבאה, מחלק את החבילה לחבילות קטנות יותר במידת הצורך ועוטף כל חבילה מחדש בתצורה שונה עם כותר חדש מסוג TCP. בכותר יוגדרו כמה דברים: פורט היעד, פורט המקור, מיקום ברצף החבילות, דגלים ועוד ועוד.

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 המקסימלי 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:

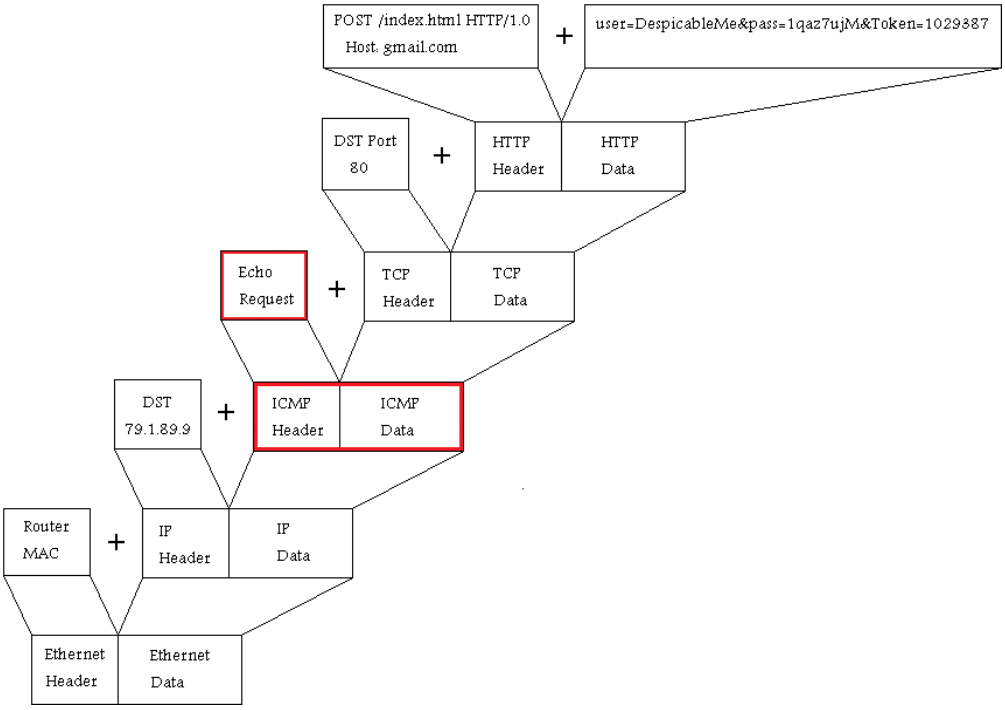
פרוטוקול TCP? הבוחן מבצע בדיקה נוספת האם התחנה מאושרת.

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.

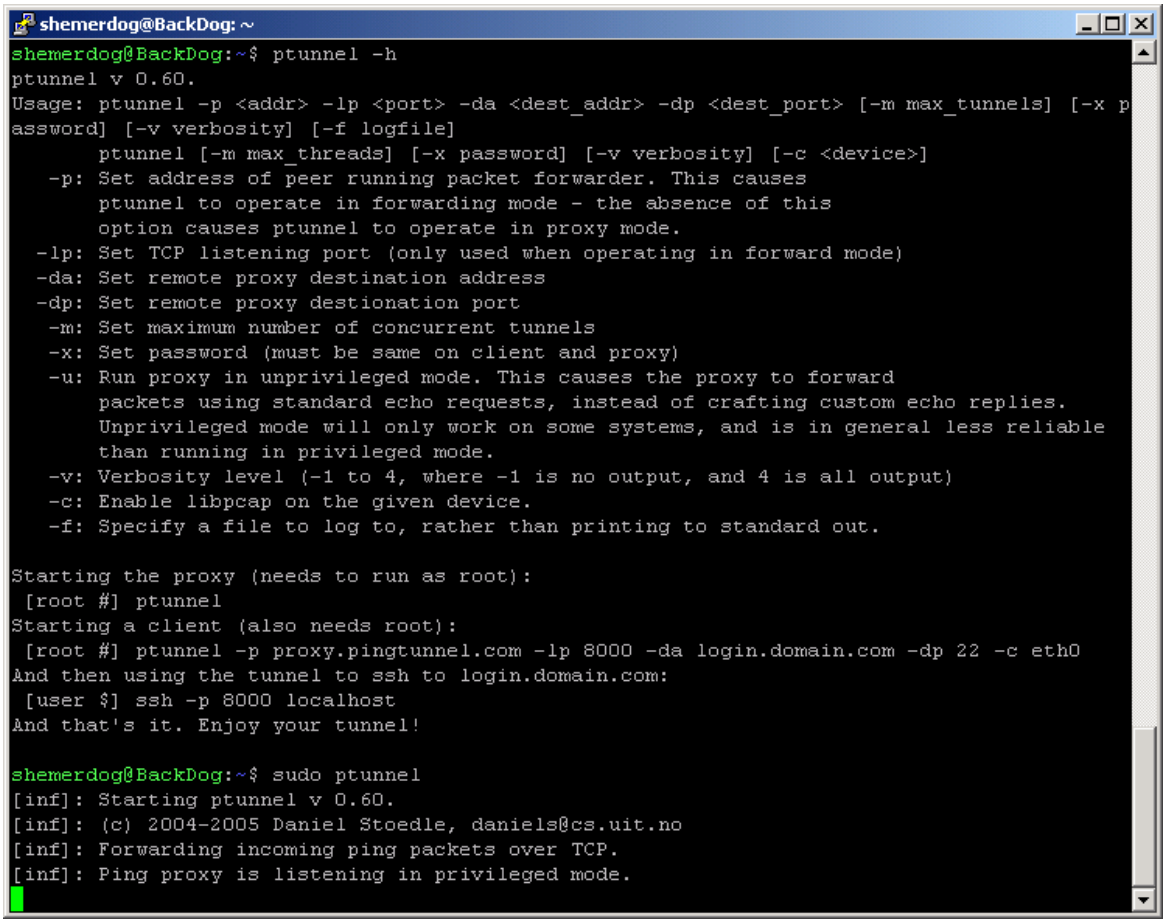
זה נראה בערך כך:



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:

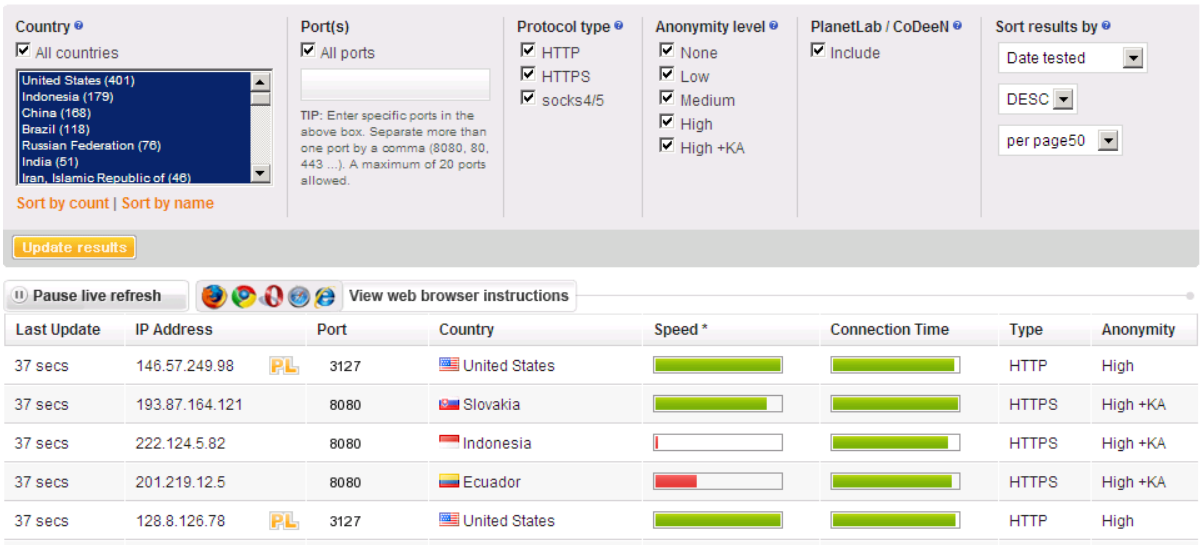


האמת שההסבר של הכלי עצמו די ממצא בנתינת דוגמא ליצירת חיבור SSH. נתרגם את זה לדוגמא שלנו:

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:

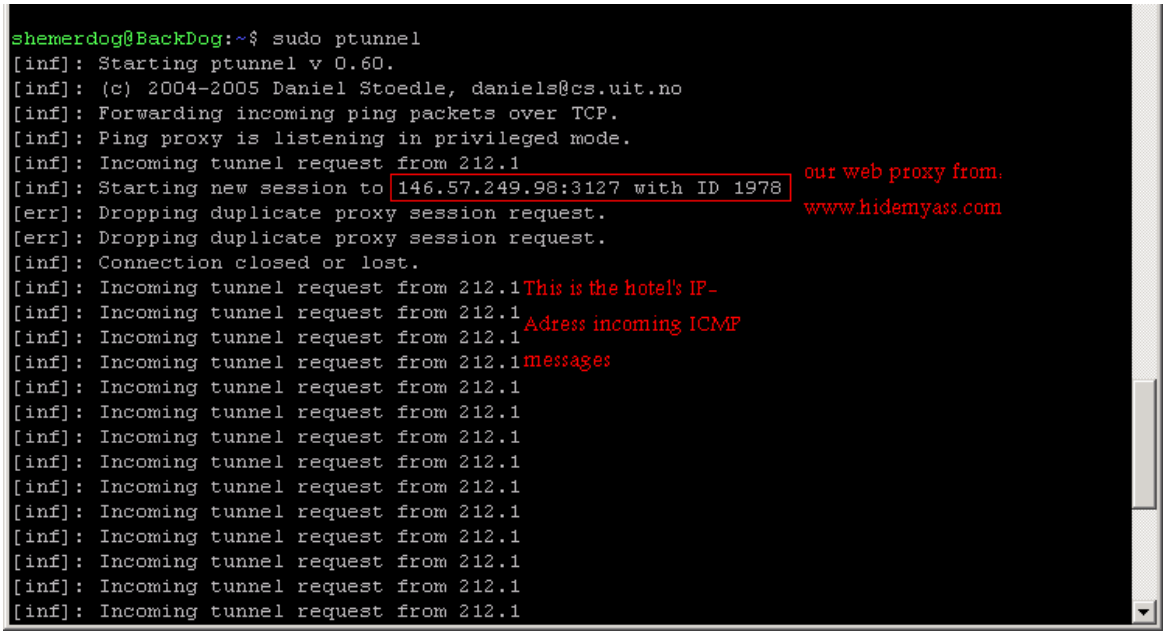


כדי ליצור חיבור מהמחשב הנייד שלכם אל שרת ה-BackTrack המאזין בבית וממנו לשרת הפרוקסי:

```
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

תרחיש: 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.

מצב מוגבל: כל פורטי היציאה חסומים.

אפשרות היציאה: דרך שרת הפרוקסי הארגוני לאתרים מסויימים בלבד.

הגדרה כללית: ניתן להוציא תקשורת מסוג HTTP בלבד ובאופן לא ישיר.

דרוש: 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 והמינהרות 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 (PPTP ,UDP).
- 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).

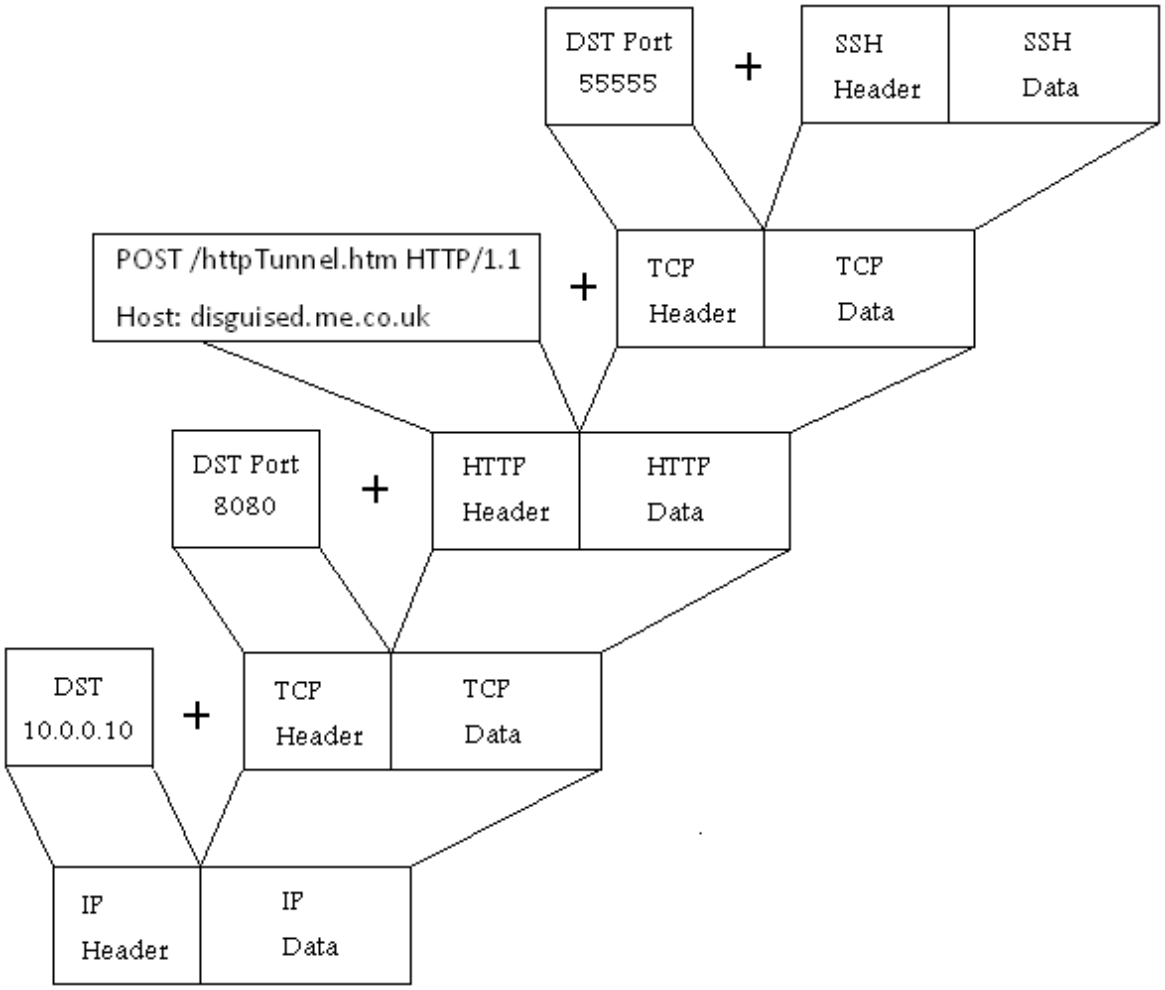
נתבונן רגע בדוגמא הלוגית הבאה שתציג את העיקרון בבקשות ה-HTTP שיתקבלו על ידי השרת:

```
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:

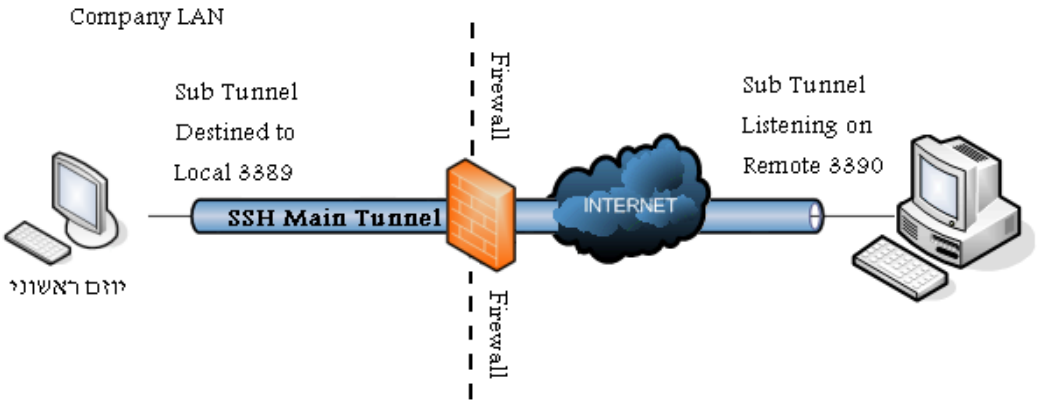
הגדרת הכיוון: המיקום שבו ייקבע הפורט שישמש להאזנה יקבע את כיוונה של תת המנהרה החדשה.

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.

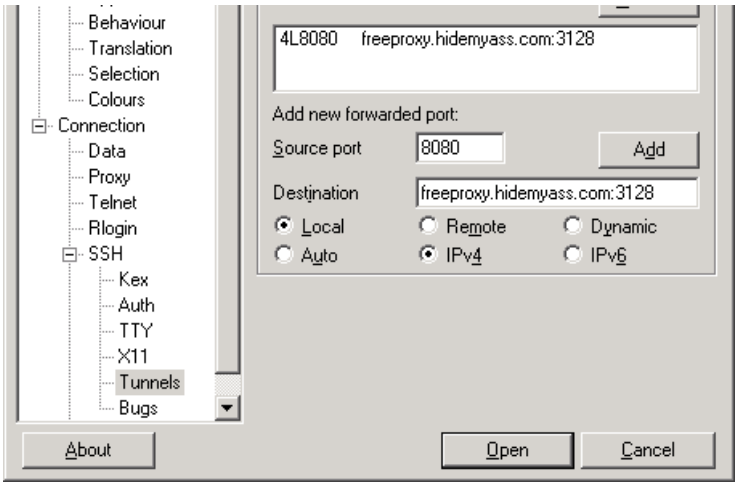
מה שיפה וגמיש במנהרות SSH, מעבר לפשטות ולזמינות שלהן, הוא שניתן ליצור קשר גם אל תחנות אחרות, כך שכל אחד מצידי המנהרה משמש לגמרי כמו Gateway. יתכן מצב שבו תהיה לנו גישה לשרת בסביבת ה-DMZ בארגון שממנו ניזום מנהרה לתחנה חיצונית. בתוך המנהרה הזו עצמה ניצור תת-מנהרה מבחוץ אל שרת פנימי אחר בארגון שמקבל חיבורי 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
```

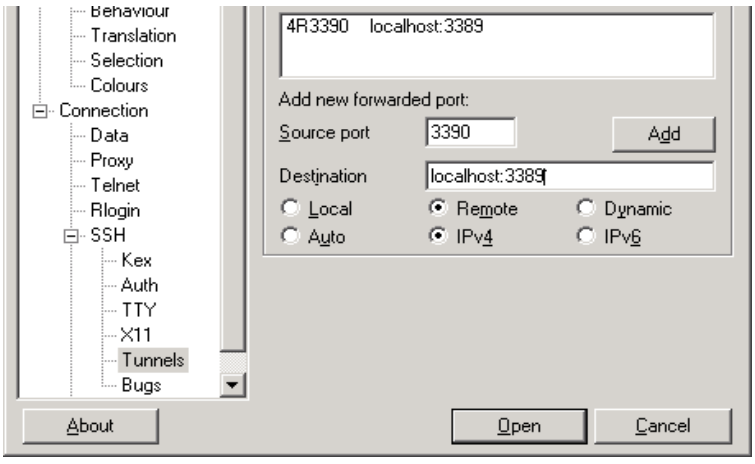
אותו הדבר מבוצע ב-PuTTY:



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 מינהרות 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 מינהרות 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 מינהרות.

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 (1+10+26).
- 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.

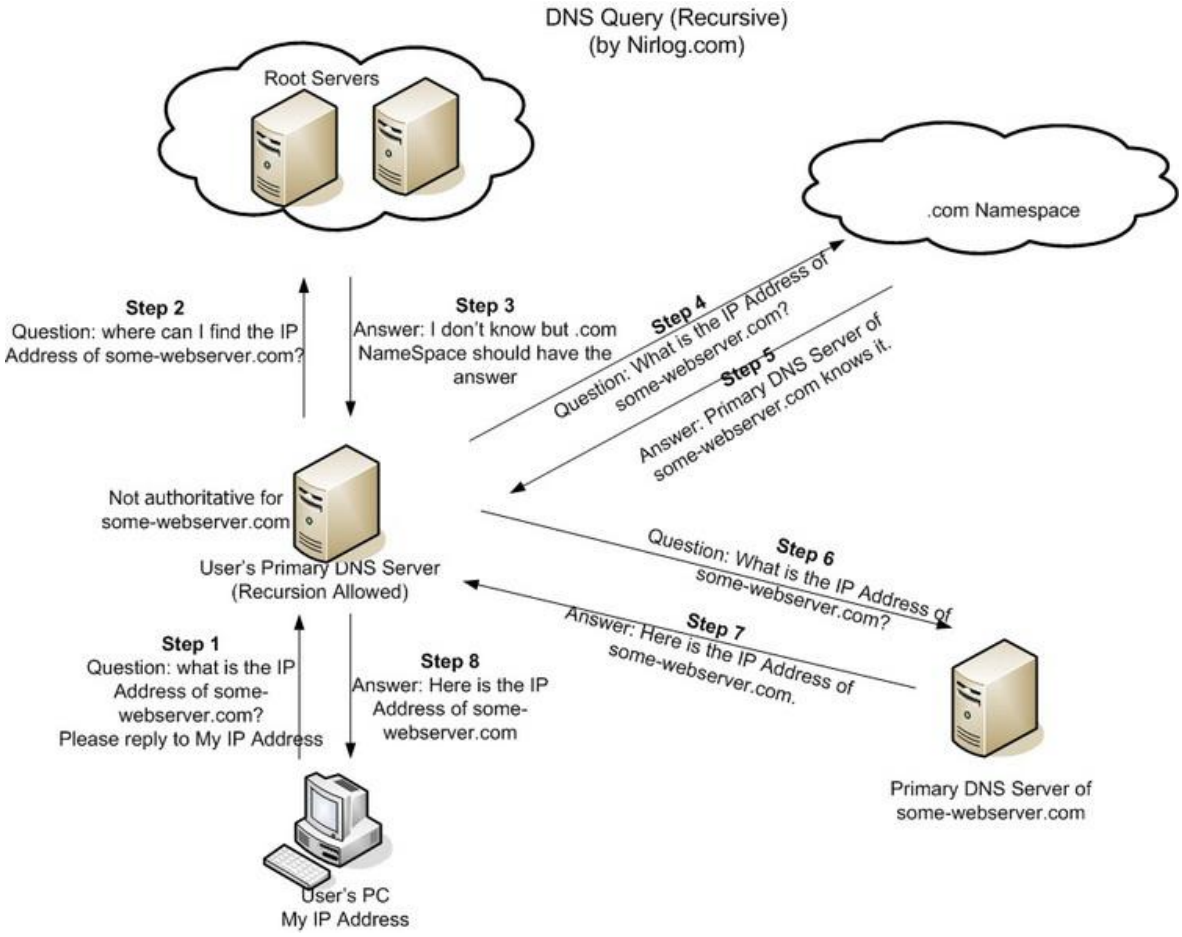
אצל רשם ה-DNS נגדיר שרת אחד בלבד בתור NS בלעדי של הדומיין שלנו ולא ניצור אף רשומה נוספת.

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.

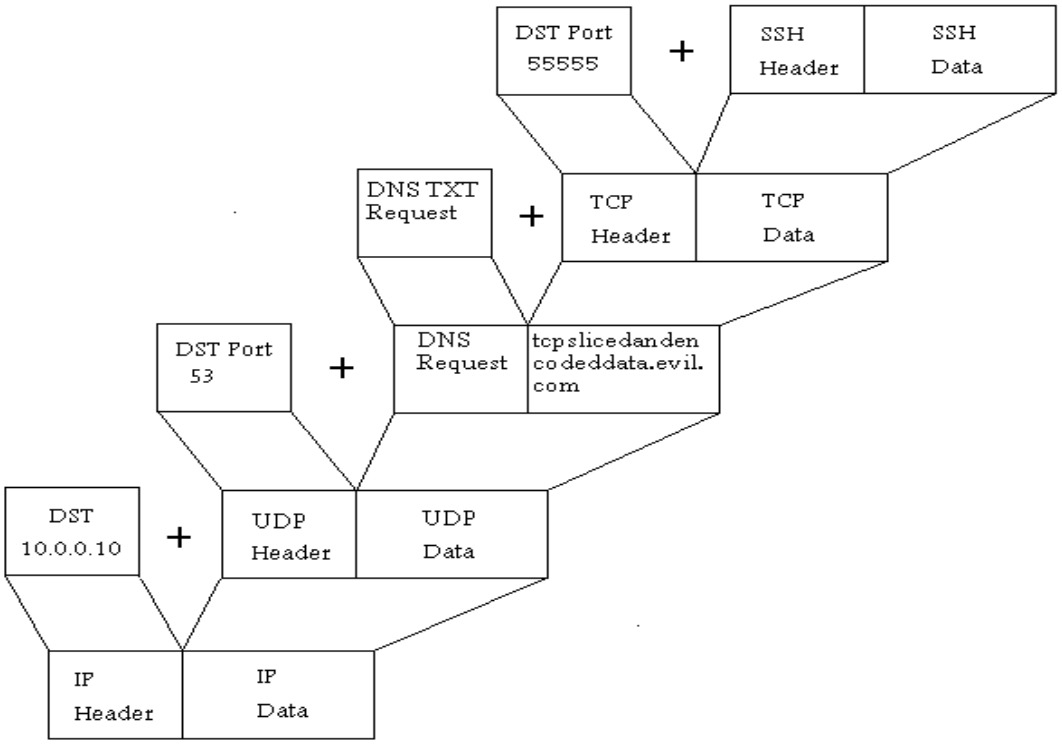
תרחיש DNS רגיל:



(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.

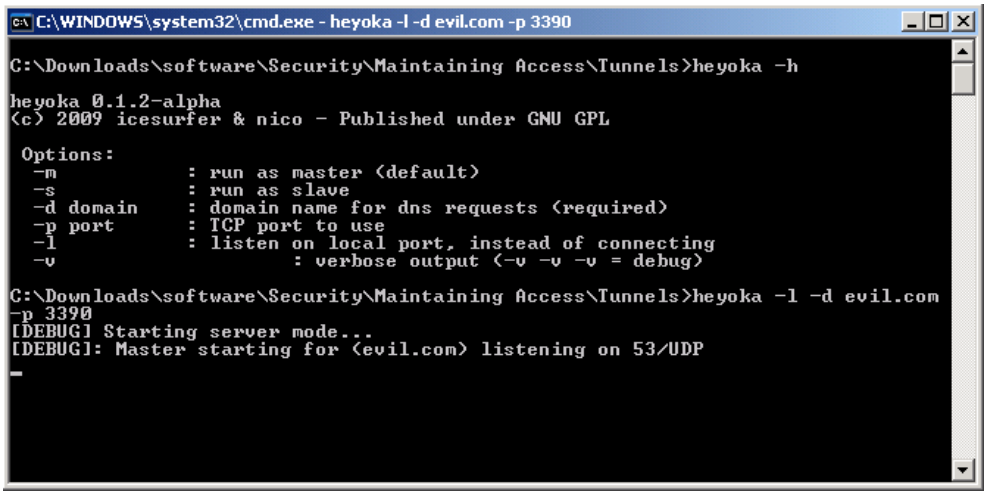
מלבד זאת, מופיע במצגת סיכום קצר וקולע בנושא המינהרות:

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

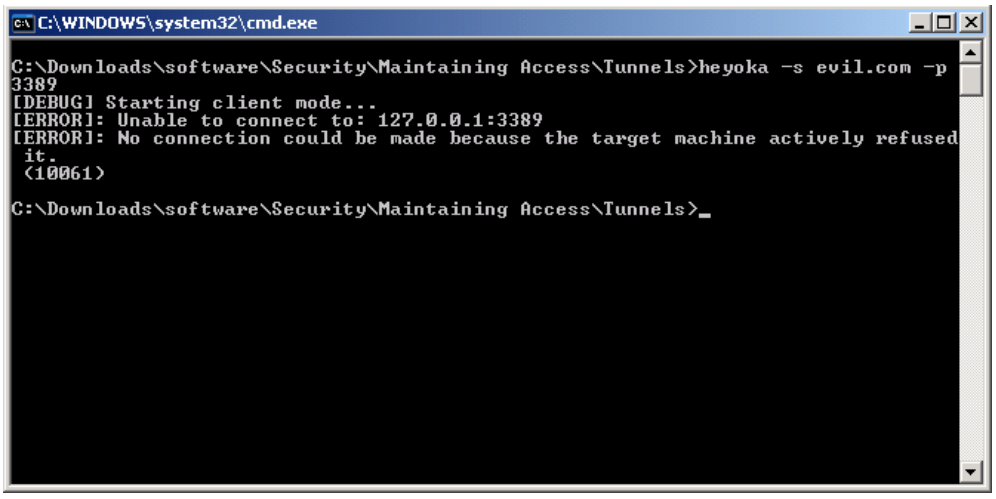
עדיין, מבין כל המנהרות, זו האיטית והרועשת ביותר, אולם כשאין מוצא, מעט עדיף מכלום.

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 וללליאור ברש שהיוו את הצלילה המשמעותית ביותר שלי לתוך עולם אבטחת המידע.