Digital Whisper

גליון 68, ינואר 2016

מערכת המגזין:

מייסדים:	אפיק קסטיאל, ניר אדר
מוביל הפרויקט:	אפיק קסטיאל
עורכים:	אפיק קסטיאל, ניר אדר
כתבים:	ניר נטר, עדן ברגר, עידו קנר וים מסיקה.

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

ברוכים הבאים לגליון ה-68, הגליון המסכם את שנת 2015.

נראה כי שנת 2015 הייתה שנה מעניינת לרב הדעות, עושה רושם שאירועים הקשורים למקצוע שלנו בפרט ולכל עניין הפרטיות בעולם האינטרנט בכלל עברו אסקלציה די רצינית בשנה האחרונה, החל מחולשות שפורסמו ברמה תשתיתית - כזו שגורמת גם לאדישים ביותר בתחום להרים גבה, ולהבין שכל ההגנות שלנו הן אשלייה אחת גדולה (או כמה אשליות קטנות, תחליטו אתם, זה לא באמת משנה), וכלה בקצוות קרחוני ענק שמעידות (כביכול?) על התערבות של שחקן בסדר גודל מדינתי ברמה שנוגעת ברובנו יום-יום.

אירועים כאלה יכולים ללמד אותנו לא מעט תובנות, וביניהם קיימות שלוש חשובות מאוד:

- **הראשונה**: תמיד נופתע, האירועים האחרונים מלמדים אותנו שלאינטרנט יש יותר שכבות ממה שנראה על פני השטח, וחשוב שנצא מנקודת ההנחה ששום דבר לא בטוח במאה אחוז. ברגע שנשכח את זה - פספסנו. תחשדו בהכל, אל תמסכו על כלום.

- **השניה**: טוב שאנחנו משקיעים באבטחה, וחשב לעשות זאת, אבל חשוב יותר לזכור שהכל או ככל הנראה - הרוב - לא יהיה רלוונטי במידה ויסמנו את הארגון שלנו כמטרה. ואם נצא מנקודת ההנחה הזאת בעת פריסת תשתית ההגנה שלנו, או בעת בניית טופולוגיית הרשת שלנו - הרווחנו.

- **השלישית**: תחשבו תמיד מחוץ לקופסא ותמיד צעד אחד קדימה, למרות שנראה שהעניין חשוב מאוד לצד התוקף, הוא חשוב מאוד (ואולי אף יותר) לצד המגן. על מנת באמת להגן על ארגון שלנו ועל הנכסים היקרים בו עלינו להניח שכבר פרצו לנו, או שההצפנות שלנו לא מספיק חזקות בשום מקרה. עלינו לסדר את מערך ההגנה שלנו כך שגם אז ובכל מקרה - הנכסים שלנו יהיו מוגנים.

כמובן שאפשר להמשיך ולדבר ולהציג עוד נקודות, אך אלו החשובות שבחרנו לסמן. אז מה תוליד שנת 2016? שאלה קשה, כנראה שלא נוכל לדעת, אבל מה שכן - ברור שיהיה לזה IP, שזה יידע לרוץ על טוסטר, ושנופתע בצורה מביכה מכמה שזה לא מאובטח...

וכמה מילים בפאן האישי ממש לפני שנגמרת לנו השנה: כל זמן הוא זמן טוב להוכיר תודה למי שעזרו לך, ובחלוף שנת 2015 ברצוננו להגיד תודה רבה לכל מי שעזר לנו השנה, ולכול מי שתרם מזמנו וממרצו, ולכל מי שהשקיע ולכל מי שבזכותו הפרוייקט הזה ממשיך לרוץ ולבעוט - תדעו שזה לא מובן לנו מאליו, תדעו שאנו מעריכים כל שעה ושעה מזמנכם שהקדשתם לפרוייקט ודרכו לקהילה.

אז תודה רבה ל: ארי'ה קסטיאל, שילה ספרה מלר, 5Fingers, יובל (tisf) נתיב, מתן אביטן, מתן הרט, יגאל סולימני, שחף אלקסלסי, ליאור אופנהיים, יניב בלמס, שחק שלו, יובל סיני, גל תא שמע, d4d, מנחם ברויאר, עידו קנר, שמואל (sub) ירוחם, ליאור ברש, ישי גרסטל, 0x3d5157636b525761, רזיאל בקר, נתנאל רובין, תומר זית, יהודה גרסטל, איאן מילר, עמית סרפר, אלכס פרייזר, עו"ד יהונתן קלינגר, ישראל (Sro) חורז'בסקי, דימה פשול, עידו קנר, עדן אלון, ניר (hyprnir) עופר, גל ביטנסקי, עופר גייר, אור וילדר, יגאל זייפמן ו-OGRose.

וכמובן לפני הכל - נרצה להגיד תודה רבה לכל מי שהשקיע מזמנו ובזכותו אתם קוראים שורות אלו: תודה רבה ל**ים מסיקה**, תודה רבה ל**ניר נטר**, תודה רבה ל**עדן ברגר** ותודה רבה ל**עידו קנר**!

קריאה מהנה!

ניר אדר ואפיק קסטיאל.

תוכן עניינים

דבר העורכים	2
תוכן עניינים	4
אז מה קרה החודש?	5
Windows Scripting - חלק א': Introduction	15
איך ליצור שכבת אנונימיות לבית ע"י בידוד מערכות ורשת TOR	27
הכרת SCTP - חלק שני	37
דברי סיכום	78

אז מה קרה החודש?

מאת ים מסיקה

צנזר- אות-

בין הוויכוחים הלוהטים ביותר שיוצא לי לנהל בשנים האחרונות הוא על נייטרליות הרשת ועל החופשיות של האינטרנט. ממשלה צריכה להתערב? לספקיות מותר להתערב? התוכן צריך לעבור צנזורה או להישאר ללא מגע ידם של גופים ושל ממשלות?

נרצה או לא, כיום המצב הוא שממשלות אכן משפיעות על דברים שנמצאים באינטרנט. אם מדובר בחוקי זכויות־יוצרים שגורמים לסגירתם של אתרי הורדות ואתרים אחרים, אם אלו חוקי הימורים שגורמים לחסימת אתרי הימורים בעולם ואם אלו חוקי הצנזורה שמגבילים את התבטאויותיהם של כלי התקשורת.

מדינות קיצוניות יותר אף מגדילות לעשות וחוסמות אתרים שמכילים תוכן שמבקר את הממשלה או מפלגות מסוימות. באותן מדינות הגישה לעבודות עיתונאיות ושל קבוצות בעלות דעה שונה משל השלטון עלולה להיאסר.

מאחר שכיום בתי־המשפט בעולם מסוגלים לדרוש מספקיות האינטרנט להוריד אתר מסיבה כזו או אחרת, נוצר HTTP Status code שמספרו 451 ומטרתו לציין "חוסר זמינות עקב סיבות חוקיות". בפועל: נחסם עקב הוראת בית־משפט. שרת שמגיש דף שכזה עלול לכלול את כל המידע החשוב הנוגע לחסימה, כולל קישורים לצו בית־המשפט הרלוונטי, איך "לאתגר" את החסימה והסבר על החוק שמאפשר לבית־המשפט לחסום אתרי אינטרנט במדינה. ההצעה ל־Status code אושרה על ידי ה־IETF ב־18 בדצמבר.

מספר ה־Status code נבחר לא במקרה, ויש בו ריח חזק של אמירה נוקבת. הוא קורץ לספרו הדיסטופי של ריי ברדבורי ששמו "451 פרנהייט", המתאר עולם שבו ספרים הם מחוץ לחוק.

מדובר ברעיון מבורך, אבל במקומכם לא הייתי ממהר לבזבז את בקבוקי האלכוהול שקניתם לנובי־גוד. בשבועיים האחרונים הספקתי לראות את הצהלה ההזויה במדיה שאומרת ש"ממשלות כבר לא יוכלו להסוות שהן צָנזרו דפים באינטרנט". כמובן שמדובר בשטויות מוחלטות, שכן בתי־המשפט תמיד יוכלו לכפות על אתרים להחזיר שגיאת 404 עבור עמודים שצונזרו.

שני רווחים בביטים... והופ, אתם בפנים

ב־14 לדצמבר התפרסמה פרצה ב־Grub2 (ה־Boot loader הנפוץ במערכות Linux) שהצליחה לעשות הרבה מאוד באזז. הגרסאות הפגיעות הן כל הגרסאות מ־1.98 (דצמבר 2009) ועד 2.02 (דצמבר 2015).

הדבר שאולי הקנה למתקפה הכי הרבה פרסום הוא הפשטות המגוחכת להפליא שלה: בזמן שאתם במסך ה־GRUB2 והוא מבקש מכם שם משתמש, לחצו 28 פעמים על מקש ה־Backspace. במידה ואתם פגיעים - מזל טוב, יעלה לכם מסך של GRUB Rescue ומפה אתם יכולים לעשות מה שבא לכם.

הפרצה ניתנת לניצול רק בעזרת גישה פיזית למכונה, ומאפשרת לתוקף לעקוף כל סוג של אימות, בין אם הסיסמה נשמרה כטקסט או מוצפנת. תוקף שיבצע את המתקפה בהצלחה יזכה בגישה מלאה למחשב דרך ה־GRUB Rescue shell.

על איך עובדת החולשה מאחורי הקלעים, בקצרה[1]: על ידי לחיצה על Backspace 28 פעמים אתם גורמים ל־Overflow וכותבים על ה־Return address של הפונקציה שבה אתם נמצאים, מה שגורם לקוד שלכם לקפוץ לכתובת 0x0 בזיכרון, שם יש שם לולאה שעורכת את עצמה ובסופו של דבר קופצת לפונקציה grub_rescue_run, שמריצה את ה־Grub rescue.

אם כל העניין הזה עושה לכם Déjà vu, יכול להיות שזה בגלל הבאג המוזר שהתפרסם לפני כשנה וחצי ב־Unity, מעטפת הממשק של הפצת הלינוקס Ubuntu, בזכותו אדם עם גישה פיזית למכונה יכול היה ללחוץ ארוכות על מקש ה־ENTER ולעקוף את מסך הנעילה.

באג מיליון הדולר

במאמר שפרסמו בלומברג על תכנית ה־Bug bounty של Facebook ב־2012 כתבו ש"אם יהיה תקל ששווה מיליון דולר, אנחנו נשלם את זה". החודש יצא לבחור אחד, וסלי, לבחון את האמת של האמירה הזו.

בשיחה ב־IRC, חבר נתן לו טיפ. הוא מצא כתובת מעניינת, תת־דומיין של Instagram בכתובת https://sensu.instagram.com שמאחוריה יש שירות Sensu-Admin שרץ מעל Ruby On Rails ועלול להיות פגיע. הבחור הסתכל על הקוד של Sensu-Admin ומצא שם מפתח סודי שמוגדר כברירת מחדל, ושאם Instagram לא שינו אותו הוא עלול לעזור לזיוף Session ואפילו להרצת קוד מרוחקת. למרבה ההפתעה, כך קרה - והוא הצליח להריץ קוד מרחוק ודיווח על כך.

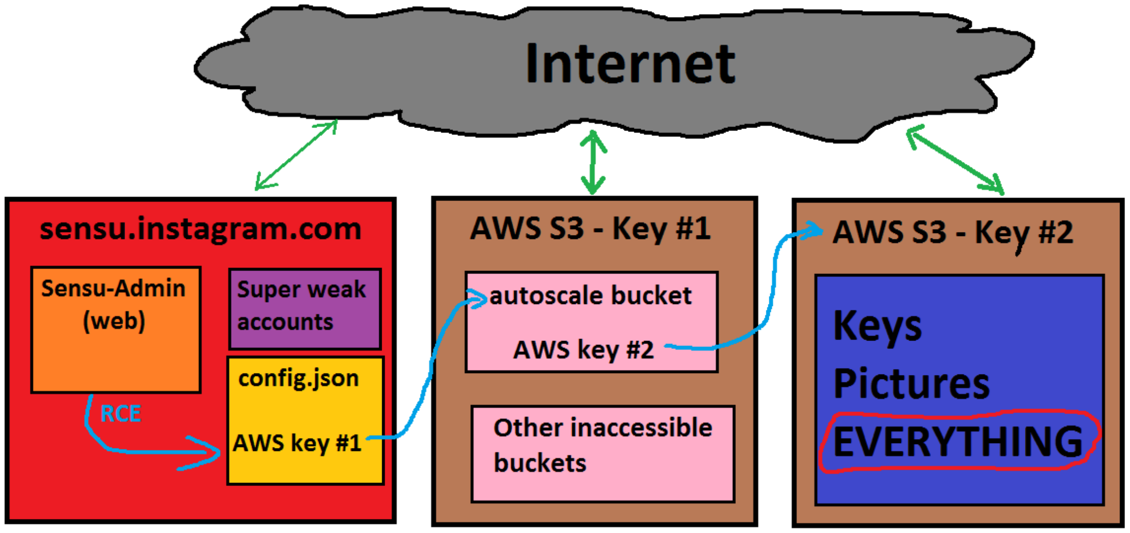
למרות זאת, הוא עדיין לא הצליח להתחבר לממשק בכתובת הזו. הוא השתמש ב־RCE כדי לקבל גישה למסד־הנתונים, ומשם שלף את שמות המשתמש והסיסמאות לממשק (שהיו כמובן מוצפנות; אפילו עם

[1] לפרטים טכניים מקיפים אמליץ לכם לקרוא את הפרסום המלא.

bcrypt). אחרי ממש מעט זמן הוא הצליח בעזרת JTR לגלות 12 סיסמאות שהיו מובנות מאליהן: password, changeme, Instagram וסיסמאות שזהות לשם המשתמש. הוא שלח דיווח שני.

במקביל, הוא בדק מה יכול להיות ב-/etc/sensu/config.json. הוא מצא שם הרשאות לא רק למסד הנתונים, אלא גם לחשבונות דואר אלקטרוני ול-Pagerduty[2], ואפילו מפתח לחשבון AWS.[3] הוא התחיל בלבדוק מה הוא יכול לגלות אם הוא יסתכל על חשבון ה-S3[4] שלהם, וגילה שהוא לא יכול לגשת לשום דבר - פרט לתיקייה אחת. התיקייה הזו הכילה קובץ tar.gz בו לא היה שום דבר מעניין, אבל בגרסאות הקודמות של אותו קובץ היה קובץ Vagrant עם עוד מפתח לחשבון AWS, שהפעם כלל גישה לכל שאר תיקיות ה-S3.

מה היה שם, אתם שואלים? מעבר לתמונות של משתמשי Instagram: קוד המקור של גרסה די חדשה של Instagram, SSL Certificates ומפתחות פרטיים, מפתחות API לכל שירות אפשרי (Twitter, Facebook, Thumblr, Flickr, OAuth, reCAPTCHA) ומפתחות שנועדו לייצר את העוגיות ב-Instagram. וסלי שלח דיווח שלישי, הפעם עם 7 סעיפי חולשה חדשים. בדיווח שלו, הוא כלל את הסעיף "אם ישנם כלי תיעוד מופעלים, הרי שאף אחד לא בודק אותם שכן בשום שלב לא זיהו את הנתונים אליהם ניגשתי".



עד מהרה הדיווח התגלגל להיות ויכוח של ממש. בתגובה לדיווחים של וסלי ענו בחברת Facebook: "שלום וסלי. תודה שדיווחת לנו. אנחנו שולחים את המידע לצוות המוצר הרלוונטי למחקר נוסף. אנו נעדכן אותך בנוגע להתקדמות שלנו. אנא קח בחשבון שנקיטת צעדים נוספים אחרי שמצאת פרצה מפרים את תכנית הפרסים שלנו. בעתיד אנחנו מצפים שמתוך רצון טוב תעשה מאמץ להימנע מהפרות של פרטיות, הרס מידע והפרעה או פגיעה לשירותים שלנו במהלך המחקר שלך".

---

[2] שירות שמספק כלי מעקב אחרי הזמינות והביצועים של השירות שלך.
[3] שירות Amazon Web Services. מספקים הרבה שירותים מגניבים כמו S3 (אחסון קבצים ב-Cloud) ו-EC2 (שרתי Cloud).
[4] שירות אחסון הקבצים בענן של Amazon.

משם החלו חילופי דברים שגרמו מהר מאוד לסגירת הדיווח של וסלי בלי לתת לו פרטים נוספים. כשפתח וסלי קריאה נוספת וביקש תשובות, ענו לו "שלום וסלי. אנו מעריכים את הדיווח שלך, אך הוא אינו זכאי להיות חלק מתוכנית הפרסים שלנו. ניצור עמך קשר אם יהיו לנו שאלות נוספות". כששאל האם הוא יכול לפרסם את ממצאיו, ענו לו: "ההחלטה האם לפרסם בידיך, אנו לא נותנים או מונעים אותה באופן מפורש".

נשמע שכאן היה יכול להסתיים הסיפור, אבל בבוקר שלמחרת הדיווח השלישי, כך לפי דבריו של וסלי, הוא קיבל טלפון מהמנהל שלו. המנהל אמר שהוא היה בשיחה עם מנכ"ל החברה בה וסלי עובד. מסתבר שאל המנכ"ל התקשר בחור בשם אלכס סטמוס, ה־CSO של Facebook. הוא אמר שוסלי השתמש במידע רגיש שהוא השיג על ידי פרצת אבטחה, ובזמן שהיא הייתה "טריוויאלית ובעלת ערך זעום" היא גרמה לדאגה גדולה בקרב עובדי החברה.

לפי דבריו של וסלי, אלכס אמר שהוא לא בטוח אם הוא רוצה לפנות לרשויות החוק, ושהוא לא רוצה לערב את הצוות המשפטי של Facebook. הוא דרש שוסלי לא יפרסם את פרטי החולשה, שישמור את כל הגילויים לעצמו ולא יפרסם אותם, יאשר שהוא לא ניגש לפרטי משתמשים ושימחק את כל המידע שהוא השיג ממערכות Instagram.

וסלי כתב בבלוג שלו שבקשותיו של אלכס הן ההפוכות מהכתוב במדיניות של Facebook: "אם תאפשר לנו זמן מספיק להגיב לדיווח שלך לפני שתהפוך מידע לפומבי, ומתוך רצון טוב תעשה מאמץ להימנע מהפרות של פרטיות, הרס מידע והפרעה או פגיעה לשירותים שלנו במהלך המחקר שלך, אנחנו לא נפתח בשום תהליך משפטי כנגדך או נבקש מרשויות אכיפת־החוק לחקור אותך". הוא ציין את האיומים לכאורה של Facebook ואת הפנייה למעבידים שלו ישירות לרעה.

אלכס הגיב בעצמו לדבריו של וסלי ואמר שהציעו לו 2,500$ למרות שהוא לא היחיד שמצא את הפרצה, ושהפנייה למעסיק נעשתה מכיוון שחשבו שוסלי מייצג את החברה. הוא אמר שהם ביקשו לא לפרסם רק את הפרטים של הגישה ל־S3 וכל מה שקרה אחר־כך, מכיוון שמדובר בניצול לרעה של תכנית הפרסים.

את טענות כל אחד מהצדדים תוכלו לקרוא כאן (אלכס) וכאן (וסלי).

ערער מעורער

חברת ציוד התקשורת Juniper Networks יצאה בהודעה דרמטית משהו שפורסמה ב־17 לחודש. עיקר ההודעה, שנוגע למוצר ה-Firewall שלה, היא פִּסְקָה שהצליחה להצית גל תקשורתי רחב של ידיעות וספקולציות:

> "במהלך סקירת קוד פנימית שביצענו לאחרונה, Juniper גילתה קוד לא מורשה ב־ScreenOS שיכול לאפשר לתוקף בעל ידע להשיג הרשאות ניהול למכשירי ®NetScreen ולפענח חיבורי VPN מוצפנים. מרגע שזיהינו את הפגיעות הזו, פתחנו בחקירה לעניינם של הדברים, ועבדנו על־מנת לפענח ולשחרר גרסאות מתוקנות עבור הגרסאות האחרונות של ScreenOS."

אם לקחת את התוכן מהפסקה הזו ולהסתכל על ה־Security advisory, Juniper אומרת בהודעה שישנן שתי פרצות, שתיהן מזוהות כ־CVE-2015-7756: הראשונה מדברת על כך שניתן לפענח את כל התעבורה שעוברת ב־VPN שהגדרנו ל-Firewall, והשנייה אומרת שניתן לגשת לאפשרויות ניהול במוצר גם אם אין לכם באמת הרשאות לכך.

בפרסומיה Juniper כמעט ולא נותנת לנו מידע קונקרטי על הפרצה ואיך היא עובדת, ומשתדלת לשמור על נוסח כמה שיותר מעורפל בכל הקשור לפרטים טכניים. בנוגע לפרצה שנוגעת לכניסה כמנהל, כתוב בכמה מילים על כך שאם ניצלו אותה התיעוד יראה לנו[5] שמשתמש בשם system התחבר למערכת. בנוגע לפרצה שנוגעת לפענוח תעבורה שעוברת מעל VPN הם לא מרחיבים מעבר לכך ש"שהתוקף צריך להאזין לתעבורה המוצפנת".

ה"קוד הבלתי מורשה" שהם מדברים עליו, כך נראה, הוכנס למערכות לראשונה בספטמבר 2012, וטלאי שמתקן את הפרצות שהוכנסו לקוד שוחרר ביום פרסום ההודעה. בכל מקרה, הסיפור הזה מדיף ריח מוזר מאוד שגרם לקהילת ה־Security לשאול הרבה שאלות:

1. "קוד לא מורשה"? האם זה אומר שמישהו חיצוני הצליח לגשת ל־Code repository של Juniper ולערוך אותו? מי יכול לבצע דבר כזה?
2. בהודעה מאוחרת יותר מיקדו Juniper את הניסוח: "חולשת פענוח ה־VPN עלולה לאפשר לתוקף בעל ידע[6] שיכול להאזין לתעבורת VPN לפענח את התעבורה הזו". מה זה אומר "תוקף בעל ידע"? אנשים רבים ברחבי האינטרנט טענו שמדובר ב־NOBU קלאסי.[7]
3. למה Juniper נמנעה מלהכניס פרטים טכניים לתוך כל אחת מההודעות שלה?

מיד אחרי הפרסום, ואולי בעקבות הניסוח המעורפל וסימני השאלה הרבים שצצים ממנו, החלו חוקרים למיניהם עטים על המציאה ומנסים לנתח כל דבר אפשרי שקשור לאותה הודעה של Juniper. אנשים

---

[5] אם לא שינו אותו. לא סביר בשום צורה שהיא. אם אתם הייתם כותבים Backdoor, זה לא בדיוק מה שהייתם מחפשים להעלים מהתיעוד?
[6] במקור: "Knowledgeable Attacker".
[7] Nothing But Us. פרצות קטנות שמושתלות על ידי ארגוני ביון בצורה שאף־אחד לא יוכל לזהות או להשתמש בהן חוץ מהם.

החלו להשוות את ההבדלים בין הגרסה של לפני התיקון לזו שאחריה, HD Moore פרסם ניתוח שמפרט בדיוק איפה הושתלה הדלת־האחורית ומה הסיסמה שהושתלה ובעזרתה ניתן להתחבר למוצרי Juniper(!), ראלף פיליפ פרסם ניתוח שמראה איך אפשר לפענח את התקשורת מעל ה־VPN, ומת'יו גרין דיבר על הקריפטוגרפיה שמאחורי הדלת־האחורית ב־VPN.

ניסיתי לסכם את זה בקצרה (כי אפשר לקרוא על זה המון - כמעט - בכל - מקום), אבל לא כזה הלך לי. אז הנה הסיפור הכמה־שיותר־מלא[8]: כש־Jupiter רצו ליצור מפתח שיצפין את התקשורת של המוצרים שלהם, הם השתמשו בתקן שה־NSA קידמו. לתקן קוראים Dual_EC DRBG ומטרתו יצירת מספרים אקראיים. ב־2007 רשות התקינה הלאומית של ארצות־הברית (NIST) הכירה בו ועודדה את השימוש בו.

כדי להשתמש ב־Dual_EC DRBG לצורך יצירת מספרים אקראיים, יש צורך לספק לו שני פרמטרים, P ו־Q, כאשר אותם P ו־Q הן שתי נקודות על עקום אליפטי כלשהו. NSA סיפקו 2 נקודות "ברירת מחדל" כחלק מהתקן שקידמו (תודה רבה באמת). זמן קצר לאחר שהוכר כתקן על ידי ה־NIST, מצאו בו שני חוקרים ממיקרוסופט חולשה משמעותית.

הם הראו שאם P ו־Q נבחרים באופן מאוד מסוים, כך שיש משוואה שמקיימת Q=P*e עבור e כלשהו (שנשמר בסוד), מי שיצר את אותם פרמטרים חשודים (P, Q) יכול לחזות מה יהיה הפלט ה"אקראי". בשביל לגלות את e יש צורך בכוח חישובי רב מאוד. בשלב הזה אולי כדאי לציין שלפי הדלפות של סנודן משנת 2013, נראה שהחולשה לא הפתיעה מדי את ה־NSA. למעשה, היא די הייתה שם בכוונה תחילה.

אחרי אותה הדלפה ב־2013, Juniper מיהרו לצאת בהצהרה שהם אכן עושים שימוש בתקן Dual_EC DRBG, אך הם אינם עושים שימוש בפרמטרים המומלצים (בפועל הם השתמשו ב־Q משלהם). יותר מזה, המערכת משתמשת בתקן יצירת מספרים אקראיים נוסף, שנקרא ANSI X.9.31 ושהפלט של Dual_EC עובר דרכו.

הפתעה! עקב באג מוזר שנמצא במערכת, האלגוריתם השני לא באמת עבד. התוצאה הסופית הייתה בפועל הפלט של ה־Dual_EC, מה שחוקרים הגדירו אחר־כך כ"תקל קטסטרופלי".

אבל רגע! טכנית, אנחנו אמורים להיות בטוחים! יהיה בלגאן רק אם ה־Q הוא Q שנבחר בקפידה על ידי התוקף, ו־Juniper הרי השתמשו ב־Q משלהם, לא?

אז... כן. ולא. מסתבר שאותו Q מיוחד שהוגדר על ידי Juniper שונה ב־2012 על־ידי אותו תוקף מסתורי, שכנראה יודע מה ה־e המתאים. בתיקון שהם הוציאו החודש, Juniper החזירו את Q למה שהיה לפני 2012, וכך פתרו את הבעיה.

---

[8] המתמטיקאים שביניכם יאלצו לסלוח לי על חוסר־הדיוק והשמטת הפרטים הרבים. מי שבאמת מעוניין, כדאי שיקרא את המאמר של ראלף.

למרות זאת, גם אחרי התיקון עולות שאלות רבות לגבי התנהלות Juniper. למה הם משתמשים באלגוריתם שיש בו סיכון ועוטפים אותו באלגוריתם אחר לצורך הגנה? האם מדובר בצירוף מקרים ש"בטעות" לא עשו שימוש באותו אלגוריתם שני? למה התיקון שלהם כולל רק את החזרת ה־Q למצבו הקודם, ולא טיפול בטעויות המימוש הפונקציונליות? ובכלל, למה Juniper מסרבת לגלות איך היא יצרה את ה־Q? יכול להיות שהיא בעצמה מחזיקה ב־e מסוים ורוצה לעקוב אחרי המשתמשים שלה?

בחלק מהעיתונים ואמצעי המדיה נפוצה הסברה שמדובר ב־FEEDTROUGH, יכולת של ה־NSA שדלפה בעבר ומאפשרת לה לשתול דלתות־אחוריות במוצרי חומת־אש של Juniper, ושלפי המסמכים שדלפו "הופעלה במערכות רבות". הדבר לא סביר מכיוון שבמסמכים שדלפו נראה שמדובר בהשתלת חומרה שמתבצעת במכשירים לפני שהם נשלחים ללקוח.

מסמך חדש שהודלף על ידי סנודן מופץ גם הוא בתקשורת. במסמך, שנכתב בשיתוף פעולה בין ה־NSA לבין ה־GCHQ ומסווג כ־"TOP SECRET", כתוב בבירור שה־NSA מכירה פרצות אבטחה קיימות במוצרי Juniper עוד מאז שנת 2011. נוסף על־כך, מסתמן שיתוף פעולה פורה בין שתי סוכנויות הביון בכל הנוגע לניצול המוצרים של Juniper לצורכי מודיעין: "השיתוף של ה־NSA בכל הקשור בטכנולוגיה של Juniper שופר בצורה דרמטית במהלך שנת 2010 לצורך ניצול מספר רשתות יעד ש־GCHQ הייתה נגישה אליהם קודם לכן".

חשוב לזכור שנראה שהפרצות הקשורות למאמר הושתלו רק ב־2012, מה שמעיד על כך שהמסמך המודלף אינו קשור למקרה הנוכחי, לפחות לא באופן שהוא יותר מהצהרת כוונות.

עוקרים מהשורש

ב־30 בנובמבר בשעה 6:50 (UTC) קיבלו שרתי השורש (DNS Root Servers)[9] בעולם כמות בקשות בלתי סבירה בעליל, במה שנראה כמתקפת DDoS על השרתים. הכמות עלתה ועלתה במשך כמעט 3 שעות, עד שהגיעה בשיאה לכ־5 מיליון בקשות עבור כל שרת בכל שנייה, ופסקה לחלוטין בשעה 9:30. ההתקפה חודשה למשך שעה בדיוק ב־1 בדצמבר בשעה 5:10.

ב־30 בנובמבר כל השאילתות שנשלחו כחלק מהמתקפה היו על־אודות דומיין מסוים, ובהתקפה ב־1 בדצמבר השאילתות היו על־אודות דומיין אחר. הבקשות הגיעו מכמות מרשימה של כתובות IP, כך שקשה לנחש מי הוא שעומד בבסיס המתקפה[10], והמניעים העומדים מאחוריה אינם ברורים. ברוס שנייר העלה את ההשערה שמדובר בניסוי ליכולת תקיפה.

[9] תאשימו את איגוד האינטרנט הישראלי.
[10] אף שייתכן שכתובות אלו זויפו.

רשת האינטרנט תלויה במידה רבה בשרתי ה־DNS, והארכיטקטורה שלהם חסינה ויציבה מאוד על־מנת לתת מענה למקרים שבהם מנסים להזיק להם או להשבית אותם בהתקפות שכאלו. כך, לדוגמה, בעולם יש 13 שרתי שורש[11], הם מופעלים על ידי 12 ארגונים שונים ונמצאים ב־6 יבשות שונות. ה"שרתים" האלו הם לא באמת שרתים, אלא חוות שרתים גדולות שמורכבות משרתים פיזיים רבים - שרת L הועתק ל־128 מקומות שונים שנמצאים ב־127 ערים שונות ב־68 מדינות, מארגנטינה ועד תימן.

למרות הכול, יש חדשות רעות: בשלב מסוים, מספר פניות לגיטימיות לשרתי השורש אכן הסתיימו ב־Timeout מצד חלק מהם. מנגד, החדשות הטובות הן שחלק מהשרתים היו נגישים מכל מכונות הבקרה לכל אורך המאורע, כך שלא היו דיווחים מצד משתמשי הקצה על בעיות, והבלגאן הסתכם בעיכובים כמעט לא מורגשים עבור שאילתות מסוימות.

בעקבות המתקפה חסרת התקדים הוציאו root-servers.org, האחראיים על שרתי השורש, הודעה לכלל הציבור, בה פרסמו פרטים בסיסיים על ההתקפה (שהובאו כאן במלואם) והמליצו לספקיות לאמץ את BCP-38, הצעה שמטרתה אימות כתובת המקור של התעבורה היוצאות ממכונה מסוימת וכבר מיושמת ברוב גדול של הספקיות.

למרות שעדיין לא יצא לנו לחזות בהתקפות בסדר גודל שכזה על שרתי ה־DNS, זו בהחלט לא פעם ראשונה שמהלך שכזה מתבצע. מתקפות דומות קרו עוד ב־2002 וב־2007, ואיום משעשע להחשכת האינטרנט שוגר מצד אנונימוס בשנת 2012.

Дигмтал Уиспэр

החודש זכינו במאמר חדש מהחבר'ה ב־ESET, שמספרים לנו שהם גילו משהו מוזר שקרה בחודש אוקטובר באתר של Ammyy, תוכנת שליטה מרחוק. אלו שבחרו להוריד את הגרסה החינמית זכו לקבל תוספת במתנה.

בשביל הגיוון והעניין, כמעט כל יום המורידים של התוכנה זכו לקבל תוספת אחרת: ב־26 הם קיבלו את Lurk, ב־29 את Corebot, את Buhtrap ב־30 ואת Ranbyus או את Netwire RAT ב־2 בנובמבר. נראה שאותו אחד שפרץ לאתר מכר גישה לבעלי הכלים השונים, ונראה שהיו די הרבה קופצים על המציאה...

Buhtrap נשמע לכם מוכר? זה לא במקרה. בשלהי שנת 2014 זיהו חברת ESET מספר מחשבים ברוסיה, שנראה שמתנהגים מוזר במשך השנה האחרונה. היה נראה שכלי כלשהו שמוגדר לפעול רק על מחשבים רוסיים עושה כמה דברים לא נחמדים בכלל. הוא מנצל תוכנה פופולארית ברוסיה בשם Yandex Pluto[12]

---

[11] שמו של כל שרת מיוצג על ידי אות. A, B, C... עד M.

[12] תוכנה שמטרתה לשנות את השפה של המקלדת בהתאם למילים שהמשתמש מקליד.

לצורך רחרוח אחרי דברים שהמשתמש מקליד, הוא דואג שיהיה backdoor על המכונה, ועל הדרך הוא גם מתקין רכיב זדוני שמרגל אחרי פעילות המחשב ובין היתר יודע לקרוא כרטיסים חכמים.

הכלי, שמטרותיו היו בעיקר עסקים ובנקים רוסיים, הוא כלי חתום שנראה שבמסגרת כתיבתו נעשה מאמץ רציני כדי להתחמק מזיהוי. "מבצע בוח'טראפ"[13], כך מדווחים ESET, מעניין במיוחד מכיוון שהוא נראה כניסיון להפיק *רווח כלכלי* על ידי *תקיפות ממוקדות* - שילוב של שני דברים שקיים אצל מעטים מהכלים שאנחנו מכירים.

מהצד של הפרטים הטכניים, הכלי השתמש ב־CVE-2012-0158, פִּרצה בתוכנת Microsoft Word שהחברה סגרה לפני 3 שנים. הפתיון הוא מסמכי Word ברוסית (קבלות וחוזי טלפוניה) שנשלחו לאנשים מסוימים בבנקים ובחברות. כשאותם אנשים פתחו את המסמך, תוכנת ה־Word הורידה והריצה טרויאני שארוז במנהל ההתקנות של Nullsoft, ודואג להשמיד את עצמו אם הוא מזהה שהוא רץ בתוך מכונה וירטואלית או על מכונה שמותקנים בה כלי מחקר Maleware למיניהם. בצעד מעניין למדי, הוא גם בודק שמותקנת חבילת שפה רוסית למערכת ההפעלה (חלונות), ואפילו בודק בעזרת ה־Registry האם הוקלדו בדפדפן כתובות שקשורות לבנקים או שמותקנות על המכונה תוכנות הקשורות לבנקאות[14].

אם הכול הלך חלק, הוא מוריד קובץ נוסף שנועד לרגל אחרי המחשב של הקרבן. הסיפור המגניב באמת פה הוא שבהתאם לתוצאות של הסריקות, עלול לרדת אחד משני קבצים: קובץ תמים או קובץ מזיק. לדוגמה, אחד מהקבצים התמימים היה התקנה של Windows Live Toolbar, שבעזרתו ככל הנראה קיוו התוקפים להסתיר ולהפוך את הפעילות שלהם ללגיטימית מול כלי הגנה שונים (שכן נעשה שימוש בפִּרצה, אבל בסופו של דבר הותקן כלי לגיטימי).

אחרי שהורידו למחשב הקרבן כלי זדוני שהריץ את install.cmd. עכשיו הגיע הזמן לבדוק באיזה הרשאות אנחנו. אם הכלי מתבאס על זה שהוא רץ בהרשאות נמוכות, הוא הולך להשתמש ב־2 טריקים: הראשון הוא CVE-2013-3660, והשני הוא הטריק שראינו בקוד שהודלף מ־Carberp.

בגדול, כדי לקבל הרשאות גבוהות באופן אוטומטי במערכת ההפעלה חלונות, אתם צריכים שקובץ ההרצה שלכם ישב בתיקייה בטוחה (כמו system32), שהוא יהיה חתום ושב־manifest שלו יוגדר המאפיין autoElevate. על כל ההגדרות האלו עונה הקובץ שאחראי על העדכונים של מערכת ההפעלה, wusa.exe, שזכאי לקבל הרשאות גבוהות אוטומטית. הכלי מנצל את זה, מעתיק את הקובץ cryptbase.dll ל־%USERPROFILE%, משנה אותו כך שיגרום לכלי עצמו לעלות בזמן ההרצה, יוצר ממנו קובץ cab, משנה לו את הסיומת ל־msu ואז משתמש ב־wusa.exe על קובץ ה־msu כדי לחלץ את עצמו לתוך תיקיית המערכת.

---

13 באנגלית: Buhtrap. שילוב של Buhgalter (Бухалтер; "רואה חשבון" ברוסית) ו־Trap.

14 הרשימה מקיפה באופן מרשים מאוד, וכוללת גם תוכנות אחרות, כמו כאלו שנועדו לקריאת כרטיסים, לדוגמה.

הכלים שהותקנו במחשב השתנו בהתאם לפרופיל שזוהה במכונה - נראה שעל מכונות שונות הותקנו כלים שונים, ביניהן גרסה ערוכה של התוכנה mimikatz שמשמשת לגניבת סיסמאות, התוכנה LiteManager שמאפשרת שליטה מרחוק על המכונה, ואת הקובץ pn_pack.exe - שמתקין את התוכנה המדוברת של Yandex ומשתמש ב-DLL Side Loading כדי לצרף מודולים של Keylogging, קריאת כרטיסים חכמים וטיפול בתקשורת מול ה-C&C.

מעניין לראות שכמעט שנה אחרי הזיהוי של מבצע Buhtrap, הקבוצה שאחראית לו עדיין רצה וממשיכה לשנות את הכלי שלה לעיתים תכופות. נראה שלאט לאט הם מתקדמים לכיוון הדבקת שוק רחב, ומתרחקים מיצירת APT, סוג העבודה בה הם התחילו.

Windows Scripting - חלק א': Introduction

מאת ניר נטר

הקדמה

ברוכים הבאים למאמר הראשון בסדרת המאמרים על Windows Scripting! בסדרת המאמרים אציג ואסביר שפות Scripting שונות שמובנות ב-Windows בדגש על PowerShell. במאמר הנוכחי, אציג סקירה כללית על ההסטוריה של שפות Scripting שונות ב-Windows ואף אציג קצת יותר לעומק את חלקן.

כשאני אומר Windows Scripting אני מתכוון לשפות Scripting שמובנות ב-Windows. Windows היא מערכת הפעלה נפוצה, היא נמצאת כמעט בכל בית או עסק. אם נדע לשלוט במערכת ההפעלה על ידי אותן שפות Scripting בצורה טובה, תהיה לנו יכולת מיידית לעבוד ולנצל את המיטב ממערכת ההפעלה בהתראה של רגע, גם אם אין לנו Python או את ה-IDE האהוב עלינו. נוכל בהתראה של רגע לפתוח את ה-CMD, PowerShell או לכתוב VBScript מהיר כדי לעשות מה שאנחנו רוצים - ללא צורך בתוכנות חיצוניות!

אז מה אנחנו יכולים לעשות עם אותן שפות Scripting?! הן ישנות, הן גרועות! תנו לי פייתון עכשיו! אוקיי, אז זה נכון. פייתון זה מגניב וחזק (ואפילו מגיע מראש עם רוב הפצות הלינוקס, אבל אנחנו ב-Windows עכשיו). כבר עברנו על זה, יש המון מערכות Windows ואנחנו רוצים ללמוד להשתמש בהן בצורה הכי טובה שאפשר. פייתון לא מותקנת מראש ב-Windows, Deal with it. אז בכל זאת, מה אנחנו יכולים לעשות? Well, די הרבה. לאותן שפות Scripting יש התממשקות מאוד טובה עם Windows. הן שפות

שמעולות לאוטומציות, ניהול המחשב, הן מעולות ל-System Administrators או כמעט לכל שימוש אחר שמתבצע אל מול מערכת ההפעלה. שלא נדבר על האקרים. אם לתוקף יש גישה למחשב שלך, יש לו מגוון אפשרויות נוחות כדי להריץ לוגיקות קצת יותר מסובכות על המחשב שלך ללא צורך בקימפול, העברת הבינארי למחשב הנתקף והרצתו. ממש נוח, אה?

מיקרוסופט דאגו שיהיה לנו נוח עם מגוון רחב של שפות סקריפטים או שירותים שמנגישים לנו את המחשב ומאפשרים לנו גמישות, כמה נחמד מצדם. הם הביאו לנו את Batch, VBScript ואפילו גרסה שלהם ל-JavaScript, JScript! השיא היה ב-PowerShell, שהוא סוג של CMD על הרבה כוסות קפה, עליו נפרט בהמשך ובסדרת המאמרים לעומק. מיקרוסופט דאגו לנו לשירותים כמו WMI, Windows Management Instrumentation שמנגיש את מערכת ההפעלה בצורה של אובייקטים ככה שנוכל לנהל את מערכת ההפעלה בקלות ולקבל עליה מידע. WMI גם מספקת מנגנון טריגרים עשיר, הרצת Batch/VBScript/Jscript אם קופץ אירוע (קבצי MOF). הם דאגו לנו לעבודה נוחה מול COM בשפות כמו VBScript או PowerShell והם אפילו סיפקו לנו את WSF, Windows Scripting File שמאפשר לנו לכתוב סקריפט אחד בכמה שפות שונות. לדוגמא, נוכל לכתוב פונקציה ב-VBScript ולגשת אליה בפייתון! מגניב, לא?

שרדתם עד כאן? אתם בטח נלהבים כמוני. לפני שנכיר את השפות השונות ואת היכולות השונות, נעבור קצת לשיעור היסטוריה כדי להבין מאיפה הכל התחיל, תחזיקו חזק.

היסטוריה

מערכת ההפעלה DOS הביאה לנו את COMMAND.COM שזו התוכנה הראשונה שרצה כשהמחשב עולה ב-DOS. התוכנה הביאה לנו Shell איתו נוכל לעבוד אל מול המחשב ומערכת ההפעלה. באותו Shell כתבנו פקודות שבעזרתן אנחנו יכולים לסייר במחשב, לקבל עליו מידע ולפקד עליו. רצף פקודות כאלה יוצר לנו סקריפט שכתבנו למערכת ההפעלה כדי לבצע אוטומציות, לממש לוגיקות וכו'... אלה הם קבצי Batch.

הגיעו שנות ה-90 וחברת Microsoft שחררה סדרת מערכות הפעלה, Windows 9x כשבהן הוסיפו את cmd.exe שלמעשה היה הרחבה ל-COMMAND.COM. הם הוסיפו מגוון רחב של אפשרויות ופקודות חדשות וכיפיות. לאורך שחרורים של גרסאות Windows, מיקרוסופט הוסיפו כלים נוספים כמו wmic.exe, netsh.exe וכו'... שרק האיצו את כוחו של cmd.exe ושל Batch Scripts. בעזרת כלל הכלים שקיימים במערכת ההפעלה ניתן לקבל מידע רב על המחשב, ליצור אוטומציות, לתפעל את המחשב בצורה הרבה יותר מהירה ויעילה! Cmd.exe נמצא במקביל ל-GUI של מיקרוסופט שמאפשר לנהל את המחשב, אבל CLI הרבה יותר שווה מ-GUI.

בשנת 1996 מיקרוסופט שחררה את VBScript, שפה שמבוססת על Visual Basic שמיד היוותה תחליף לקבצי Batch. עבור Administrators זו הייתה קפיצת מדרגה מבחינת כתיבת אוטומציות וניהול הרשת/מחשבים שלהם. השפה הרבה יותר גמישה, בעלת יותר פונקציונאליות והרבה יותר מובנת. היא מאפשרת לנו התממשקות נוחה עם COM, תכונה שמוסיפה לשפה כוח רב. כמו כן, השפה לוקחת חלק בפיתוח אתרים, בצד לקוח ובצד שרת, אך ללא הצלחה רבה. Well, אפשר להבין... רק Internet Explorer תומכת בשפה.

לאורך השנים, VBScript פותחה ומגרסה 1.0 בשנת שחרורה היא הגיעה לגרסה 5.8. אך אל תטעו, כולנו יודעים שמיקרוסופט לא מאוד טובים בספירת הגרסאות למוצרים שלהם. היא קפצה מגרסה 3.0 לגרסה 5.0. לאורך השחרורים של VBScript, מיקרוסופט הוסיפו תמיכה ב-Classes, COM, פונקציות שונות, לולאות מורכבות וכו'...

נתקדם קצת בזמן... החל מ-Windows 98 מיקרוסופט הוסיפו למערכת ההפעלה שלה את WSH, Windows Scripting Host שהיוותה מנוע להרצת סקריפטים. היא מריצה כברירת מחדל רק VBScript ו-Jscript. אך ניתן להריץ דרכה עוד שפות סקריפטינג כמו Perl או Python (רק אם הן מותקנות על המחשב). WSH אפשרה להעיף את קבצי ה-Batch מהעולם והביאה מקום לשפות יותר מורכבות ככה שניתן יהיה לכתוב סקריפטים יותר מוצלחים ועם לוגיקות יותר מורכבות. בנוסף, WSH הכניסה ל-Windows את קבצי WSF, Windows Script File, פורמט קבצי XML שמאפשר הרצת כמה שפות סקריפטינג יחד.

כש-Windows 2000 שוחרר, הוא הביא איתו את WMI שמעניק יכולות רבות לשפות סקריפטינג ב-Windows ובאופן כללי לפיתוח ב-Windows (עד היום). מיקרוסופט לא שכחה את מערכות ההפעלה האחרונות שלה והוסיפה תמיכה במנגנון ל-Windows 98 ו-Windows NT 4.0 החל מ-SP4. במשפט אחד, WMI משקפת את מערכת ההפעלה דרך אובייקטים ומחלקות (נרחיב בהמשך). WMI האיצה שפות כמו VBScript מבחינת פונקציונאליות ואינטרקציה מול Windows.

באוגוסט 2002, מיקרוסופט החלו בפיתוח Shell חדש, פרויקט שנקרא Monad שמטרתו לייצר שפת Scripting חדשה ל-System Administrators. השפה תתבסס על NET Framework. שפורסמה זמן קצר לפני כן, בפברואר של אותה שנה. ב-יוני 2005 שוחררה גרסת הבטא הראשונה של הפרויקט עד שבאפריל 2006 שונה השם ל-Windows PowerShell. בנובמבר 2006, PowerShell שוחררה למערכות הפעלה החל מ-Windows XP והגיעה מובנת החל מ-Windows 7. כיום PowerShell נמצאת בגרסה 5.0.

שימושים ודוגמאות

אז הלהבתי אתכם בהתחלה (לפחות ניסיתי) ואחר כך שרדתם שיעור היסטוריה קצר. זה הזמן להראות קצת פרקטיקה! עכשיו אראה לכם קצת שימושים ודוגמאות לשפות סקריפטים ב-Windows. נעבור על כמה שפות ומנגנונים של Windows כדי לקבל תחושה על הפוטנציאל שטמון בהם.

Batch

הדוגמא הראשונה היא העתקת ה-System Event Log של מחשב כלשהו לתיקיה יעודית על ה-DC ברשת כלשהי. קבלו סיטואציה, אתם אנשי IT נחמדים שרוצים לתחקר מה קורה במחשבים שלכם ברשת. אבל אתם עצובים כי אתם לא רוצים לצאת מהמשרד שלכם עם דיסק און קי ולגשת פיזית לכל מחשב כדי להעתיק את הקבצים. אתם גם לא רוצים לעשות עבודה שחורה ולהתחבר לכל מחשב מרחוק וכך להעתיק את הקובץ. אז מה אתם עושים? Group Policy! וואו! ככה תוכלו להריץ קוד על כל המחשבים ברשת ביחד. ומה תריצו? קובץ Batch נחמד ויפה שיעשה לכם את העבודה! כמה נוח:

```
@echo off
net use \\DC\Share\Logs /USER:DOMAIN\USER RandomPassword
for /f %%a in ('wmic nicconfig where "IPEnabled='True'" get MACAddress^|findstr :') do @(
        mkdir \\DC\Share\Logs\%COMPUTERNAME%_%%a
        copy %windir%\System32\winevt\Logs\System.evtx \\DC\Share\Logs\%COMPUTERNAME%_%%a
/Y
)
net use \\DC\Share\Logs /d
```

<u>אז מה עשינו פה?</u>

השתמשנו ב-net use כדי להתחבר ל-Share שיצרנו מבעוד מועד עם User בעל הרשאות מתאימות. לאחר מכן, אנחנו משתמשים ב-for של batch (שהוא עקום במיוחד) כדי לאתחל משתנה %%a עם הפלט של הרצת wmic (שהוא CLI Shell של WMI) שיביא לנו את ה-MACAddress של ה-NIC שפועל כרגע (<u>אני מניח שיש רק אחד</u>).

לאחר מכן, אני משתמש ב-mkdir כדי ליצור תיקיה ששמה הוא "שם המחשב_MAC". לדוגמא: PC_00:11:22:33:44:55. לבסוף, אני משתמש ב-copy כדי להעתיק את הקובץ של System Event Log (עם Flag /Y כדי לבצע overwrite במידה והקובץ כבר קיים).

בסוף, אנחנו מוחקים את החיבור שיצרנו עם net use.

דוגמא נוספת, שהיא די דומה לקודמת לה. הסיפור דומה לסיפור הקודם, אתם אנשי IT נחמדים שרוצים להכיר מקרוב את כל המחשבים שלכם ברשת. לשם כך, אתם רוצים להריץ systeminfo שיספק לכם מידע

כמו hotfixes מותקנים, זיכרון, מערכת הפעלה, תאריך התקנה של המחשב, boot time וכו'... בנוסף, אתם רוצים מידע מלא על רכיבי הרשת שנמצאים במחשב. לשם כך, אתם משתמשים ב-ipconfig /all.

בדוגמא הבאה אנחנו עושים דבר זהה לדוגמא הקודמת, רק שהפעם אנחנו כותבים קובץ ל-Temp Folder אליו אנחנו כותבים מידע על המחשב ומיד לאחר מכן מעתיקים את הקובץ עם המידע שלנו ל-Share המתאים על ה-DC:

```
@echo off
net use \\DC\Share\Info /USER:DOMAIN\USER RandomPassword
for /f %%a in ('wmic nicconfig where "IPEnabled='True'" get MACAddress^|findstr :') do @(
        mkdir \\DC\Share\Info\%COMPUTERNAME%_%%a
        echo. > %temp%\info.txt
        systeminfo >> %temp%\info.txt
        ipconfig /all >> %temp%\info.txt
        copy %temp%\info.txt \\DC\Share\Info\%COMPUTERNAME%_%%a /Y
        del %temp%\info.txt
)
net use \\DC\Share\Info /d
```

VBScript

אז עלינו רמה מסתם פקודות ב-Batch Files. השפה מאפשרת לנו לבצע לוגיקות יותר מורכבות, אנחנו פחות מוגבלים בכתיבת הקוד שלנו. בקיצור - הרבה יותר כיף לנו!

הפעם אביא לכם דוגמא של סקריפט קצרצר שידפיס לכם את כתובת ה-IP החיצונית של המחשב שלכם, ועל הדרך ידפיס לכם האם החיבור שלכם תקין.

```
On Error Resume Next

Sub Main()
    On Error Resume Next

    Dim httpConnection : Set httpConnection = CreateObject("Microsoft.XMLHTTP")
    httpConnection.open "GET", "https://wtfismyip.com/text", False
    httpConnection.send ""

    If httpConnection.status = 200 Then
        WScript.Echo httpConnection.responseText
    Else
        WScript.Echo "Dude... Something is wrong."
    End If
End Sub

Main()
```

אז מה עשינו פה? השתמשנו ב-COM Object שנקרא XMLHTTP כדי לשלוח GET HTTP ל-wtfismyip.com/text. לשם כך, השתמשנו בפונקציה CreateObject. האתר אמור להדפיס את כתובת ה-IP שממנו אנחנו יוצאים. לאחר מכן אנחנו בודקים את ה-HTTP Code שחזר ובודקים האם הוא 200. אם כן, אנחנו לוקחים את ה-Response שאמור להיות רק כתובת ה-IP ואנחנו מדפיסים אותה. אם הייתה בעיה, אנחנו מדפיסים הודעה שמשהו לא בסדר. ממש לא מסובך ☺

האפשרות לגשת לאובייקטי COM שונים על ידי VBScript ממש מחזקת את השפה. הרבה תוכנות שאתם מתקינים על המחשבים מייצאים COM Interfaces כך שניתן להשתמש בהם. לדוגמא:

```
skype.TREGraphicObject
Skype4COM.Application
Skype4COM.ApplicationStream
Skype4COM.ApplicationStreamCollection
Skype4COM.Call
Skype4COM.CallChannel
Skype4COM.CallChannelCollection
Skype4COM.CallChannelManager
Skype4COM.CallChannelMessage
Skype4COM.CallCollection
Skype4COM.Chat
Skype4COM.ChatCollection
Skype4COM.ChatMessage
Skype4COM.ChatMessageCollection
Skype4COM.Client
Skype4COM.Command
Skype4COM.Conference
Skype4COM.ConferenceCollection
Skype4COM.Conversion
Skype4COM.Group
Skype4COM.GroupCollection
Skype4COM.IEProtocolHandler
Skype4COM.Participant
Skype4COM.ParticipantCollection
Skype4COM.PluginEvent
Skype4COM.PluginMenuItem
Skype4COM.Profile
Skype4COM.Settings
Skype4COM.Skype
Skype4COM.SmsChunk
Skype4COM.SmsChunkCollection
Skype4COM.SmsMessage
Skype4COM.SmsMessageCollection
Skype4COM.SmsTarget
Skype4COM.SmsTargetCollection
Skype4COM.User
Skype4COM.UserCollection
Skype4COM.Voicemail
Skype4COM.VoicemailCollection
```

VBScript? לעבוד מול Skype? מי היה מאמין... ניתן למצוא ולעבוד אל מול מגוון רחב של COM Interfaces, חלקם מגיעים מובנים במערכת ההפעלה וחלקם מותקנים עם תוכנות.

WSF

טוב אז כאן אני פחות אפרט, כי אין כל כך על מה לפרט. אבל צריך לציין שזה קיים כי זה פשוט מגניב. Windows Script File מאפשר לך להריץ כמה שפות במקביל כאילו הן שפה אחת. כמו שכתבתי בתחילת המאמר, ניתן לכתוב פונקציה ב-VBScript ולגשת אליה ב-Python.

ויקיפדיה כבר הכינו דוגמא, אז מי אני שלא אשתמש בה?

```
<?xml version="1.0" ?>
<!-- Mixing JScript and VBScript -->
 <job id="SORT-VBScriptWithJScript">
   <script language="JScript">
     function SortVBArray(arrVBArray) {return arrVBArray.toArray().sort();}
   </script>
   <script language="VBScript">
     <![CDATA[
     '** Fastest sort: call the Jscript sort from VBScript
     myData = "a,b,c,1,2,3,X,Y,Z,p,d,q"
     wscript.echo "Original List of values: " & vbTab & myData
     starttime = timer()
     sortedArray = SortVBArray(split(myData,","))
     endtime=timer()
     jscriptTime = round(endtime-starttime,2)
     wscript.echo "JScript sorted in " & jscriptTime & " seconds: " & vbTab & sortedArray
     ]]>
   </script>
 </job>
```

וכמובן, הם לא שכחו לצרף פלט כדי שלא תחשדו שמותחים אתכם:

```
Original List of values:        a,b,c,1,2,3,X,Y,Z,p,d,q
JScript sorted in 0 seconds:    1,2,3,X,Y,Z,a,b,c,d,p,q
```

[מקור: https://en.wikipedia.org/wiki/Windows_Script_File]

WMI

אז... WMI! קיצור של Windows Management Instrumentation. זהו ממשק שמאפשר לך לנהל את המחשב, לקבל עליו מידע, לעדכן מידע ואף להריץ קוד. לא רק על המחשב שלך, אלא גם על מחשבים אחרים ברשת! המנגנון למעשה מנגיש לך את המחשב בצורה של אובייקטים ומחלקות שמחולקים לפי Namespaces שונים. לדוגמא, תוכלו למצוא תחת ה-Namespace root\cimv2 את המחלקה Win32_Process שאם תריצו שאילתה על המחלקה (WMI Query Language - WQL) תקבלו את כל ה-Instances של המחלקה עם הרבה פרמטרים על כל Process. דוגמא נוספת, תוכלו למצוא ב-Namespace root\SecurityCenter2 מחלקות בשם AntiSpywareProduct, AntiVirusProduct, FirewallProduct שה-Instances שלהן יכיל מידע על מוצרי האבטחה שמותקנים על המחשב.

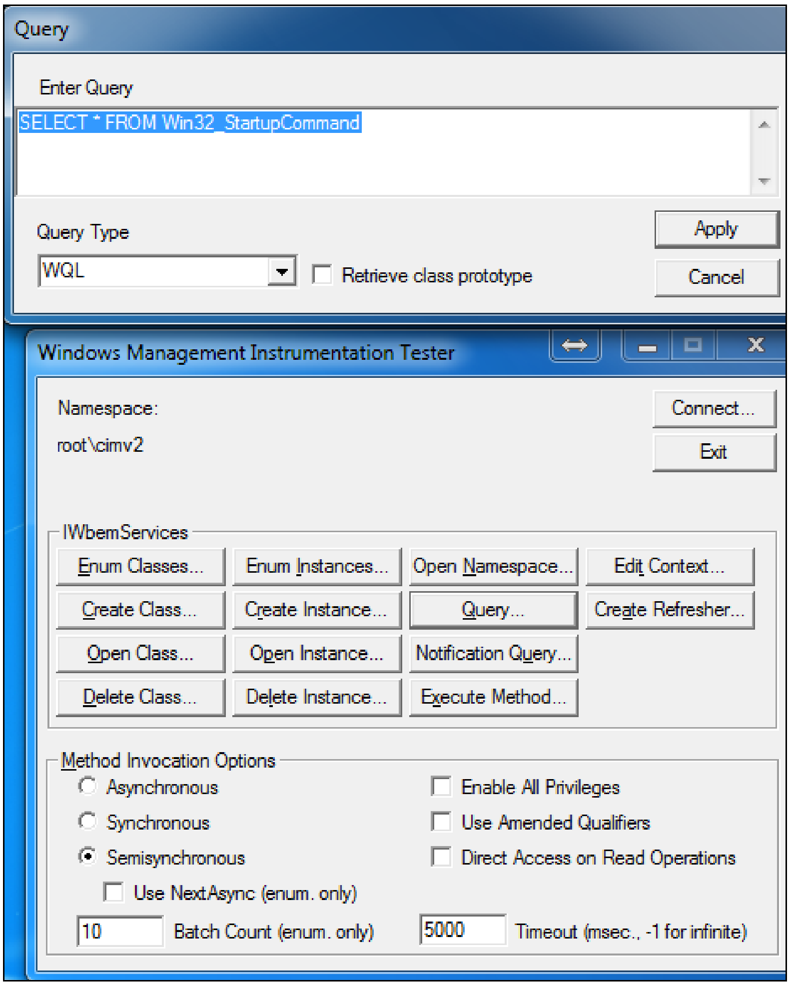
אז יש לנו די הרבה מחלקות שנגישות לנו ויכולות לספק לנו המון מידע! לא אציג פה את כולן, בשביל זה יש לכם תוכנות כמו WMI Explorer. אנו נתרכז במחלקות שנמצאות ב-Namespace שנקרא root\cimv2, זהו ה-Namespace הדיפולטי שם יש מחלקות רבות שמנגישות מידע רב על מערכת ההפעלה, חומרה וכו'... מחלקות שימושיות:

- **Win32_Process** - מחלקה המכילה מידע על תהליכים במחשב.
- **Win32_LogicalDisk** - מחלקה המכילה מידע על כוננים.
- **Win32_ComputerSystem** - מחלקה המכילה מידע על המחשב כמו שם מחשב, דומיין, דגם וכו'...

- **Win32_Service** - מכיל מידע על כל ה-services הקיימים במחשב.
- **Win32_StartupCommand** - מכיל מידע על תוכנות שעולות כשהמחשב עולה.

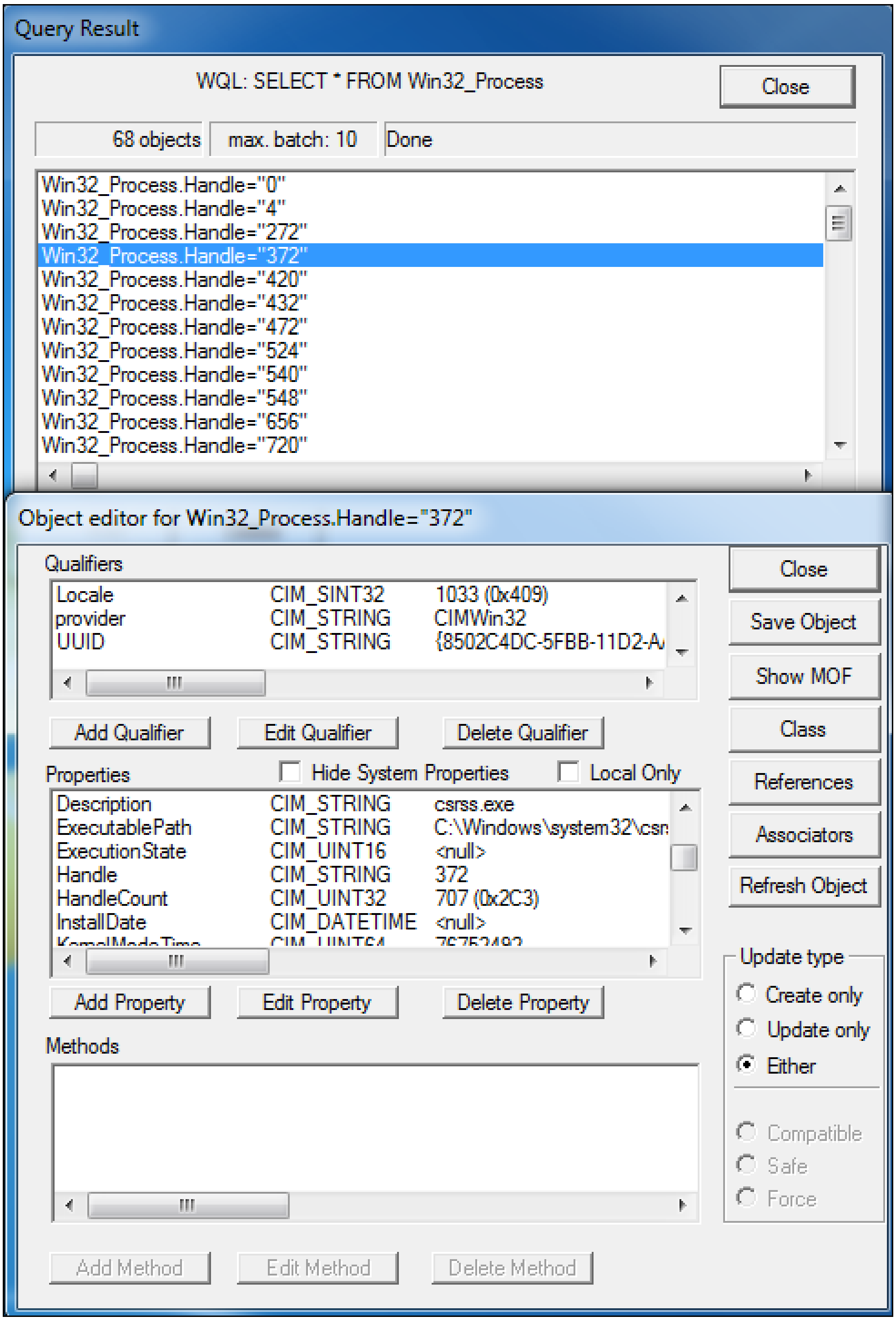
כמובן שהרשימה נמשכת ונמשכת ונמשכת ונמשכת ונמשכת...

ל-Windows יש ממשק מובנה כדי לעבוד עם WMI שנקרא wbemtest והוא נראה ככה (כשחלון הרצת שאילתה פתוח):



כאשר נתחבר ל-Namespace הרלוונטי במחשב כלשהו, נוכל להשתמש בממשק כדי לתשאל אובייקטים (Query), לנהל מחלקות, Instances ועוד...

דוגמא למידע מ-Win32_Process עבור Instance של csrss.exe:



WMI מציעה לכם דבר נחמד ויפה שנקרא Notification Query שמאפשר לכם לקבל "התראה", ברגע שמתרחש משהו. ניתן להאזין על אירועים שמתרחשים על סמך מחלקות שונות כשקיימים 4 טריגרים שונים: Creation, Deletion, Modification, Operation. לדוגמא, ניתן להגדיר Notification Query של Creation על Win32_Process כששם התהליך הוא notepad.exe. ככה ניתן לקבל "התראה" כש-notepad.exe נפתח:

```
SELECT * FROM __InstanceCreationEvent WITHIN 1 WHERE TargetInstance ISA
'Win32_Process' and TargetInstance.Name = 'notepad.exe'
```

זה למעשה מנגנון Events מאוד חזק שמערכת ההפעלה מציעה לנו. אז... מה נעשה איתו? הכירו את MOF! Managed Object Format זהו פורמט לניהול אובייקטים של WMI. בעזרת הפורמט תוכלו ליצור Instances של מחלקות קיימות, ליצור מחלקות, ליצור Namespaces וכו'... בעזרת הפורמט תוכלו לכתוב קובץ שיריץ קוד בהינתן טריגר. מגניב, לא? הנה דוגמא:

```
#pragma namespace("\\\\.\\root\\subscription")

instance of __EventFilter as $EventFilter
{
    EventNamespace = "Root\\Cimv2";
    Name  = "New Process Instance Filter";
    Query = "Select * From __InstanceCreationEvent Within 2"
            "Where TargetInstance Isa \"Win32_Process\" "
            "And Targetinstance.Name = \"notepad.exe\" ";
    QueryLanguage = "WQL";
};

instance of ActiveScriptEventConsumer as $Consumer
{
    Name = "TestConsumer";
    ScriptingEngine = "VBScript";
    ScriptText =
    "Set objFSO = CreateObject(\"Scripting.FileSystemObject\")\n"
    "Set objFile = objFSO.OpenTextFile(\"c:\\log.txt\", 8, True)\n"
    "objFile.WriteLine Time & \" \" & \" Notepad started\"\n"
    "objFile.Close\n";
};

instance of __FilterToConsumerBinding
{
    Consumer   = $Consumer;
    Filter = $EventFilter;
}
```

[מקור: http://www.codeproject.com/Articles/28226/Creating-WMI-Permanent-Event-Subscriptions-Using-M#8.ActiveScriptEventConsumerclass7]

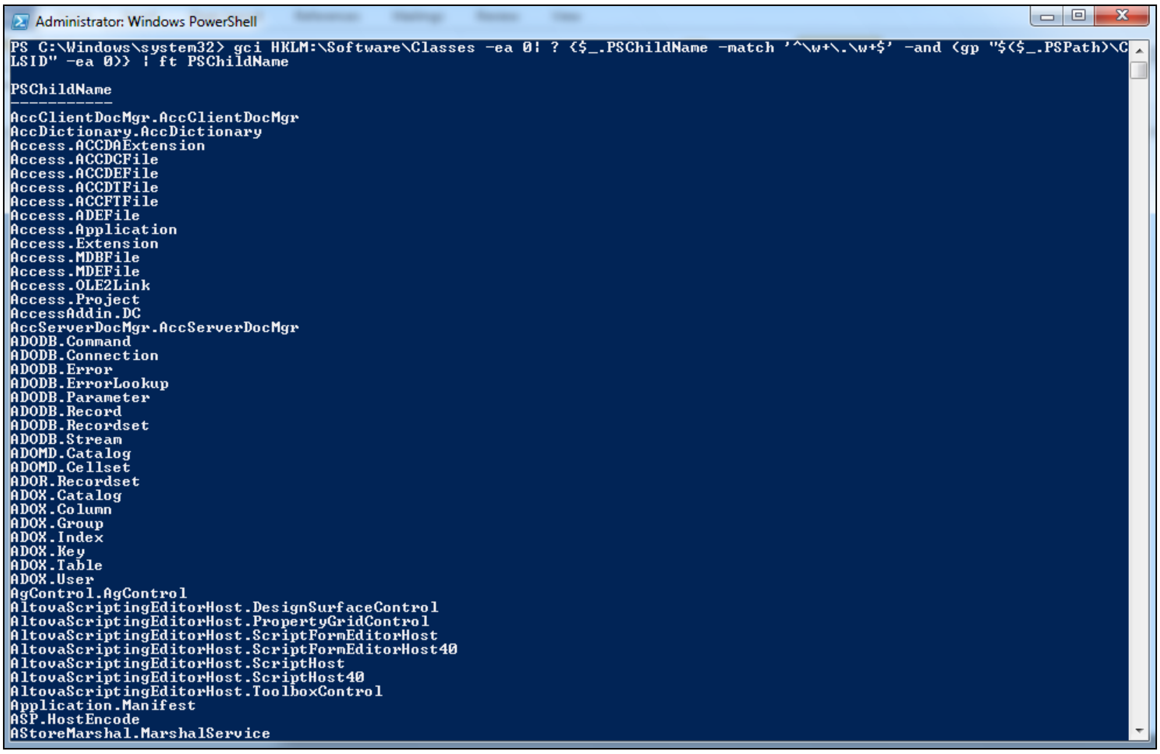
יצרנו מופע של המחלקה __EventFilter שהוא למעשה Notification Query שמאזין על פתיחת Process של notepad.exe. יצרנו מופע של המחלקה ActiveScriptEventConsumer שמאפשר לנו להריץ קוד (ספציפית, VBScript) וקישרנו את ה-Filter ל-Consumer על ידי מופע של המחלקה __FilterToConsumerBinding.

את הקובץ שומרים כקובץ mof ומשתמשים ב-mofcomp.exe כדי להכניס את מה שיצרנו ל- WMI Repository. מגניב! מעכשיו, כל פעם שנפתח notepad, ירוץ הקוד שלנו שירשום את השעה כשנפתח תהליך בשם notepad.exe ☺

PowerShell

טוב אז PowerShell זו ה-שפת סקריפטינג של Windows כרגע. יש לה התממשקות עם .NET, עם WMI, COM ומה לא? השפה נוחה לתפעול, בעלת מגוון רחב של אפשרויות. כמו כן, תוכנות רבות משתמשות בשפה כממשק ניהול. לדוגמא, על שרתי Exchange קיים Console של PowerShell עם cmdlets כדי לנהל את השרת. על השפה עצמה אסביר במאמר הבא, אז בואו נראה כמה דוגמאות כדי לטעום קצת מיכולות השפה ☺

זוכרים את התמונה של ה-COM Interfaces של Skype שהראתי לכם בחלק של VBScript? היא הגיעה מ- Powershell ☺



ה-Oneline הנחמד שמצולם למעלה מתשאל את הרג'יסטרי במקום שנמצאת רשימה של כל ה- COM Interfaces ומבצע סינון לכל מה שרלוונטי להצגה.

בנוסף, יש ל-PowerShell התממשקות ממש טובה עם WMI על ידי Get-WmiObject:

```
PS C:\Windows\system32> Get-WmiObject Win32_Process | ForEach { Write-Host Name: $_.Name `t`t PID: $_.ProcessID }
Name: System Idle Process                PID: 0
Name: System                PID: 4
Name: smss.exe              PID: 272
Name: csrss.exe                     PID: 372
Name: wininit.exe                   PID: 420
Name: csrss.exe                     PID: 432
Name: winlogon.exe                  PID: 472
Name: services.exe                  PID: 524
Name: lsass.exe                     PID: 540
Name: lsm.exe               PID: 548
```

ממש נוח לגשת לפלט של הפקודה בגלל התכונה החזקה של Powershell להנגיש לך את הפלט כרשימה של **אובייקטים**, ככה ניתן לגשת לתכונות שונות בקלות, כמו בדוגמא. את הפלט העברנו ללולאת ForEach וביצענו איטרציה על כל Instance שחזר, שם הדפסנו רק את הפרמטרים Name ו-ProcessID.

כמו כן, תשאול של Event Logs ב-PowerShell הוא די פשוט. ניתן בקלות לתשאל ולפלטר, לדוגמא:

```
PS C:\Windows\system32> Get-EventLog -log System | Where { $_.EventID -eq 1074 } | Measure-Object

Count    : 13
Average  :
Sum      :
Maximum  :
Minimum  :
Property :
```

ככה לדוגמא אנחנו שולפים מה-System Event Logs על EventID 1074 שמייצג כיבוי של המחשב ולאחר מכן משתמשים ב-Measure-Object כדי לקבל מידע סטטיסטי על מה שחזר.

סיכום

אז התחלתי בלנסות להלהיב אתכם קצת על Windows Scripting ועשינו קצת שיעור היסטוריה. לאחר מכן, הראתי לכם שימושים ודוגמאות של שפות ומנגנונים שונים ב-Windows כדי שנקבל קצת מושג על הפוטנציאל ולהבין קצת איך הדברים נראים ומרגישים. אני מקווה שתפסתם את הכוח הגדול שיש בשפות והמנגנונים ש-Windows מציעה לכם. במאמר הבא נחפור קצת יותר ב-PowerShell, נבין את השפה, נלמד איך לעבוד איתה ונכיר את ה-Syntax של השפה. נכיר cmdlets נפוצים וטריקים נחמדים. בקיצור, נראה כמה חזקה השפה.

מקווה שנהנתם מהקריאה!

ליצירת קשר ניתן לפנות ב: neter.nir@gmail.com.

איך ליצור שכבת אנונימיות לבית ע"י בידוד מערכות ורשת TOR

מאת עדן ברגר

הקדמה

בדומה לרשת ה-WWW או ל-"Clearnet", TOR הינה רשת לכל דבר, והיא נועדה לספק אנונימיות ע"י שליחת חבילות המידע בין רשת מחשבים לפני הגעתן ליעד - לדוגמה שאילתת חיפוש אל השרתים של גוגל.

אפשר לדמות את TOR להלבנת הון מהבחינה שככל שהכסף עובר יותר ידיים ככה יהיה קשה יותר לאתר את מקורו, במקרה של TOR, נראה כי שלוש ידיים זה מספיק. TOR יצאה לאור בשנת 2002 ולתקופה מאוד ארוכה הייתה איטית ולא ניתן היה להסתמך עליה בשביל גלישה יומיומית.

בשנים האחרונות תחום האנונימיות צמח בהרבה, למיטב הבנתי, בעקבות הצעדים העיקריים:

- בזכות הפיתוח של התוכנה Vidalia שמציגה באופן גרפי את החיבור אל רשת TOR.
- השילוב של Vidalia עם פיירפוקס שיצא תחת השם החדש TORBrowser.
- זרקורי המדיה פנו אל אידיאולוגיית אנונימוס וציינו שהקבוצות האנונימיות משתמשות ברשת TOR או בכנויה: "ה-Darknet", בשביל לבצע את המחאות שלהן, לאחר מכן גם TOR נשארה בכותרות בזכות האתר Silkroad שהתפרסם מהמוצרים היחודיים שלו.

המדיה הציגה את אותן הקבוצות והמעשים שלהן, אך לא דאגה להסביר את האידיאולוגיה שעומדת מאחוריהן. מאוד בקצרה: מדובר בזכות להיות לא מנוטר בפעולות אינטרנטיות ע"י הממשלות, וכחוק גלובלי לכל קבוצת האקרים אנונימית לחתום את שמה כ-Anonymous בשביל להקשות על הזיהוי, זו גם הסיבה שאי אפשר להכליל אותן, לדוגמה: קבוצות אנונימוס שונות שתוקפות את הודו וגם את פקיסטן. הקבוצות בדר"כ מחפשות צדק וחופש לפי הזווית ראייה שלהן, אם זה למען הכלל ולא מלחמה עם מדינה, אז הם יקראו Hacktivist-ים, שהם לרוב מגנים על עוד זכויות, כמו חופש המידע (פגיעה בזכויות יוצרים, WhistleBlowers וכו').

בזכות גידול המשתמשים והטכנולוגיה החדישה כיום אפשר לצפות ב-Streaming בנוחות של בערך 200KB.

הבעיה והפתרון

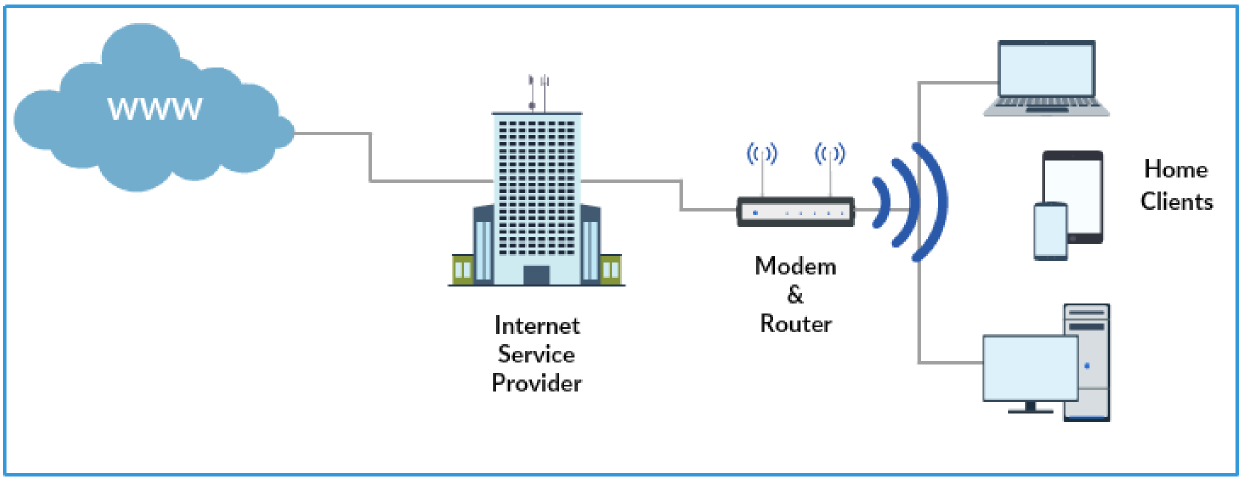
כעת, ניקח מצב שבו אנחנו מתקינים TOR על מערכת ההפעלה שלנו ומגדירים את הדפדפן המועדף עלינו להעביר את התקשורת שלו דרך ה-Proxy המקומי אל TOR, במצב כזה, אנו מיד ניצבים בפני שלוש בעיות אנונימיות:

- ה-Useragent של הדפדפן חשוף.
- תוכנות ושירותים שמותקנים במחשב לאו דווקא מוגדרים לעבור דרך הפרוקסי המקומי של TOR.
- קיים קוד JavaScript המסוגל ליצור חיבור ישיר עם המחשב וככה לגלות את האייפי האמיתי או לקרוא מאפיינים נוספים מהדפדפן.

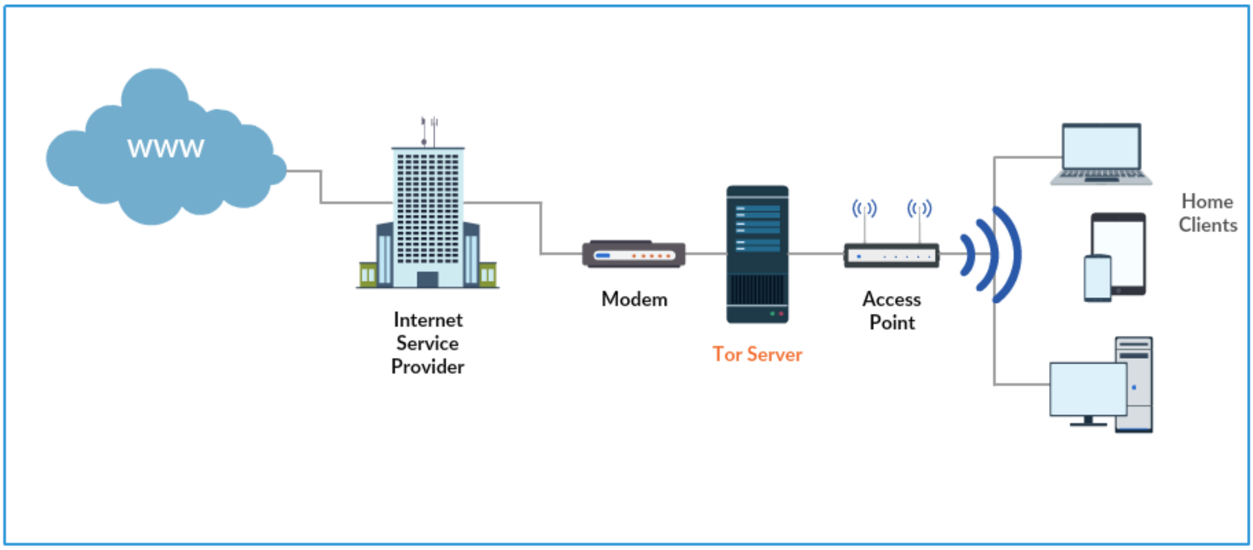
בשביל לפתור את בעיות האנונימיות האלו נפתחו מספר פרויקטים, TailsOS לדוגמה היא מערכת מבוססת דביאן שמטרתה היא להתחבר אל TOR ולהעביר את תעבורת המחשב לשם, ולהזהיר אותך לפני שאתה שולח מידע ישיר אל רשת ה-ClearNet.

בנוסף נפתחו גם הפרויקטים TorVM ו-Whonix, שמהם נצטרך לבחור ונרחיב עליהם בהמשך. אני הולך להציג כאן מימוש של ניתוב מידע מהרשת הבייתית אל רשת TOR, כך שלא יהיה צורך בהתעסקות בטלפונים או עם מחשבי המשתמשים.

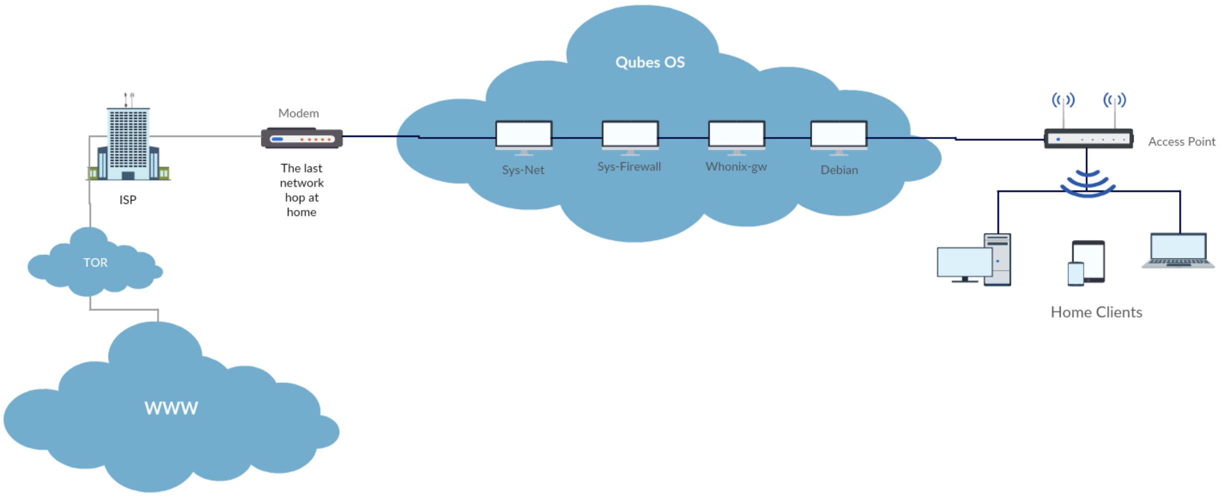
רשת בייתית נראת (בדרך כלל) כך:



לאחר שנרכיב את הפתרון שלנו, אותה רשת בייתית, תראה באופן הבא:



על ה-TOR Server נתקין מערכת ניהול וירטואליזציה בשם QubesOS. ומתחתיה את המערכת האנונימית Whonix בהמשך אפרט על שתי מערכות ההפעלה. מערך בשילוב עם מערכות ההפעלה יראה כך:



- נחבר את המודם אל המערכת sys-net.
- את ה-AP נחבר אל המכונת Debian.
- מכונת ה-Debian תספק ל-AP אינטרנט.
- שאר המחשבים יתחברו אל ה-AP, המידע של כולם ינותב דרך רשת TOR ללא צורך בהתעסקות נוספת, ונוכל למנוע DataLeaks למניהם.

תיכנון ראשוני

הדברים שנצטרך הם:

- שני ראוטרים או מודם אחד ו-switch אחד
- מחשב יעודי עם שני חיבורי רשת, Onboards או אחד חיצוני

בנוסף, חשוב לי להדגיש כי אני לא שם דגש על אבטחת מידע או על אנונימיות מוחלטת מכיוון שהנושא רחב מדי לסקר במסמך, אלו לינקים אל עמודי הבית של המערכות להמשך קריאה ונקודות בסיסיות:

1. בשביל לוודא שהמערכות הפעלה והעדכונים שלהן תקינים, נבצע בדיקת אמינות בעזרת GPG.
2. בהנחה שזו הרשת של הבית והמכשירים מחוברים, המון מידע אישי עובר, לדוגמה מהאפליקציות שמותקנות על הטלפונים (facebook).
3. כל מערכת שאנחנו מתקינים חשוב לשנות את הסיסמה של היוזר root והיוזר user.
4. עוד המון מידע מעניין על אבטחה ואנונימיות בדוקומנטציה של QubesOS ו-Whonix.

- www.qubes-os.org/doc
- www.whonix.org/wiki/Documentation

המערכות הפעלה שהשתמשתי בהן הן:

- www.qubes-os.org
- www.whonix.org
- www.ddwrt.com

חשוב להבין את שמות הכתובות ברשת TOR, אם הלינק שאנחנו עכשיו בתוכו מכיל קוד hash ארוך ולא מובן ומסתיים ב-onion. סביר להניח שאנחנו גולשים באתר שיושב בתוך רשת TOR, ולאנשים הגולשים ב-Clearnet אין route בשביל להגיע לאותם האתרים. ישנם אתרי proxy לגלישה אל תוך רשת TOR.

Qubes OS

Qubes הינה סביבת וירטואליזציה, היא מיועדת לספק אבטחה ע"י הפרדה, היא משתמשת בטכנולוגיית xen לווירטואליזציה, וכאשר נתקין ונפעיל אותה לראשונה, יהיו מולנו שלוש מערכות הפעלה וירטואליות מבוססות פדורה:

- **DOM0** - אחראית על ניהול המשאבים (זכרונות ומעבד) ועל יצירת מערכות וירטואליות חדשות, חוץ מעדכונים, אין למערכת גישה לאינטרנט.
- **Sys-net** - מערכת שהיעוד שלה הוא לנתב את תעבורת האינטרנט מהראוטר אל תוך Qubes.
- **Sys-firewall** - מספקת שכבת הגנה בין sys-net אל שאר המכונות.

בהתקנה דיפולטיבית של Qubes יגיעו איתה שלוש מערכות בנוסף:

- פרטית
- עבודה
- בנק

כל אחת כמובן עם כתובת MAC משלה ואין להן תקשורת אחת עם השנייה. ככה שנניח שנדבקתי בוירוס דרך המערכת הפרטית, אז לאו דווקא שהוירוס יצליח להדביק את שאר המערכות. ובנוסף אליהן, קיימת גם מכונה בשם DisposableVM שזו מכונה וירטואלית שנוצרת מחדש כשמפעילים אותה ונמחקת כשמכבים אותה.

Whonix

המערכת תמיד תגיע בשתי מערכות, אחת בשם Whonix-GW ואחת בשם Whonix-WS.

- תפקידה של ה-Gateway הוא להתחבר לרשת TOR
- תפקידה של ה-Workstation הוא להעביר את תעבורת האינטרנט שלה דרך ה-Gateway, וככה אנחנו נמנע Data leaks מה-Workstation.

שיטה ממולצת ע"י היוצרים שלה היא להתקין אותה בתוך Qubes.

Debian instead of whonix-ws

מצאתי שיותר פשוט לנתב את התעבורה ממכונת Debian מאשר מ-Whonix-ws, אני מניח שזה בגלל ההקשחה שהמערכת עברה בשביל למלא את מטרתה.

זו היא אינה הדרך המומלצת ע"י היוצרים של Whonix, אך אני חושב שהיא מספקת טוב שכבת TOR מעל מכשירים אישיים של הבית, לאנונימות מוחלטת צריך לעבוד קשה מהבית או קל מחוצה לו.

שלב ההתקנה

המסך הראשון של שלב ההתקנה הינו בחירת שפה - לטובת שמירת האחידות עם יחידות אנונימיות כמו שלנו, בכללי כדאי להשאיר הגדרות בדיפולט. כאן אפשר להיות יצירתיים ולהחליף את השפה לסינית, או שפה אחרת בעלת קהל משתמשים רב. במסך השני, אחרי קביעת השפה אני ממליץ לשנות ב- Software selection ל-Qubes OS with Xfce בשל אי צורך ב-KDE או בשניהם יחדיו. ובנוסף - תוסיפו את ההתקנה של Debian מצד ימין, אנו נשתמש בה בהמשך.

בשלב הבא אתם מתבקשים להגדיר את סיסמת ה-root, אין לנו צורך בליצור חשבון ליוזר root, תהיה לנו גישה ל-root בעזרת:

```
[user@dom0 ~]$ sudo -i
```

לאחר שההתקנה הושלמה והמחשב אותחל מחדש, תבחרו שם משתמש וסיסמה ואת הזמן הרצוי עליכם.

לאחר מכן תגיעו למסך: "Create Service VMs", אני ממליץ לבחור באפשרות השנייה:

Just create default service VMs.

האפשרות הראשונה תתקין לנו את המערכות Personal, Work ו- Banking שאנו לא נשתמש במערכות האלו כאן. והאפשרות השלישית תתן לנו מכונה ריקה.

לאחר עליית המערכת, ביחרו ב-Use default config במסך ה-Panel:

Welcome to the first start of the panel

אחרת נגדיר פאנל עליון בעצמנו.

דבר ראשון שנעשה יהיה להיכנס לטרמינל של שתי המערכות ולשנות את הסיסמאות של המשתמשים:

Qubes logo on toolbar → ServiceVM: sys-firewall → sys-firewall: Terminal

נקליד את הפקודות הבאות:

```
[user@sys-firewall ~]$ sudo -i
[root@sys-firewall ~]$ passwd
[root@sys-firewall ~]$ passwd user
```

את אותו הדבר נעשה גם ל-Sys-net ולכל שאר המכונות שנתקין.

כעת, ניגש להתקין Template של Whonix-gw. בטרמינל של dom0, נקליד:

```
[user@dom0 ~]$ sudo -i
[root@dom0 ~]$ qubes-dom0-update --enablerepo=qubes-templates-community qubes-
template-whonix
[root@dom0 ~]$ exit
```

אחרי שההורדה הסתיימה, אנחנו יכולים ליצור את שתיהן:

```
[user@dom0 ~]$ qvm-create -l purple gw --proxy --template=whonix-gw
[user@dom0 ~]$ qvm-prefs -s gw netvm sys-firewall
```

ו:

```
[user@dom0 ~]$ qvm-create -l gray ws -template=debian-8
[user@dom0 ~]$ qvm-prefs -s ws netvm gw
```

השלב הבא יהיה להוריד מ-Sys-net את אחד מה-Interfaces שהתווסף אליה (בהנחה שבחרנו ב-Create default service VMs) בשלב ההתקנה.

מ-dom0:

```
[user@dom0 ~]$ qvm-pci -l sys-net
[user@dom0 ~]$ lspci |grep Eth
```

כעת ננסה למחוק אחד מהכרטיסים (לא ניתן להבדיל איזה מהם, אנחנו אמורים למחוק ע"פ הפלט של qvm-pci וlspci) ונבדוק אם יש למכונה אינטרנט. אם אין אינטרנט סימן שהוצאנו את הכרטיס השגוי, אז נוסיף אותו בחזרה ונמחק את השני.

נכבה את כולן בשביל שנוכל לערוך את רכיבי החומרה שלהן:

```
[user@dom0 ~]$  qvm-shutdown -all
```

נמחק מ-Sys-net את ההתקן (06:00.0 זו דוגמה מהמחשב שלי):

```
[user@dom0 ~]$  qvm-pci -d sys-net 06:00.0
```

ונוסיף אל המכונה שתהיה מקושרת אל ה-AP:

```
[user@dom0 ~]$  qvm-pci -a ws 06:00.0
```

ונדליק אותן בחזרה:

```
[user@dom0 ~]$  qvm-start sys-net
[user@dom0 ~]$  qvm-start sys-firewall
[user@dom0 ~]$  qvm-start gw
[user@dom0 ~]$  qvm-start ws
```

דבר מעניין שטוב לדעת: במערכות בתוך Qubes, אחרי כל ריסטרט המערכת חוזרת להגדרות הראשוניות שלה ומוחקת קבצים וערכים חדשים, הדרך בה מוסיפים למערכת הגדרות היא ע"י הוספת סקריפטים וחבילות אל תיקיית /rw/.

כעת ניצור את הסקריפט שיכין את ההגדרות רשת בזמן עליית המערכת, בחלון טרמינל של WS נקליד:

```
user@ws:~$ sudo -i
root@ws:~# cd /rw/config
```

ונערוך את rc.local בעזרת העורך טקסט המועדף:

```
root@ws:~# nano rc.local
#!/bin/bash
```

נוסיף הגדרות לנוחות:

```
# Aliases
echo 'alias ls="ls --color"' >> /etc/bash.bashrc
echo 'alias ll="ls -l"' >> /etc/bash.bashrc
```

נשנה את ההגדרות של ה-Interface שמחובר אל ה-AP:

```
# Etherenet & IP.
/sbin/ifconfig eth1 192.168.0.1 netmask 255.255.255.0

# Fails if run more than once
set +e
/sbin/route add -net 192.168.0.0 netmask 255.255.255.0 gw 192.168.0.2
set -e
echo 1 > /proc/sys/net/ipv4/ip_forward
```

נשחזר הגדרות שהגדרנו מראש ל-IPTables:

```
# Iptables
/sbin/iptables-restore -c < /rw/config/rules.iptables
```

וכאן נסיים עם הסקריפט.

השתמשתי בסט פקודות הבא (שימו לב שזה ימחוק את טבלאות ה-IPTables):

```
root@ws:~# iptables -F
root@ws:~# iptables -t nat -F
root@ws:~# iptables -table nat -append POSTROUTING -out-interface eth0 -j
MASQUERADE
```

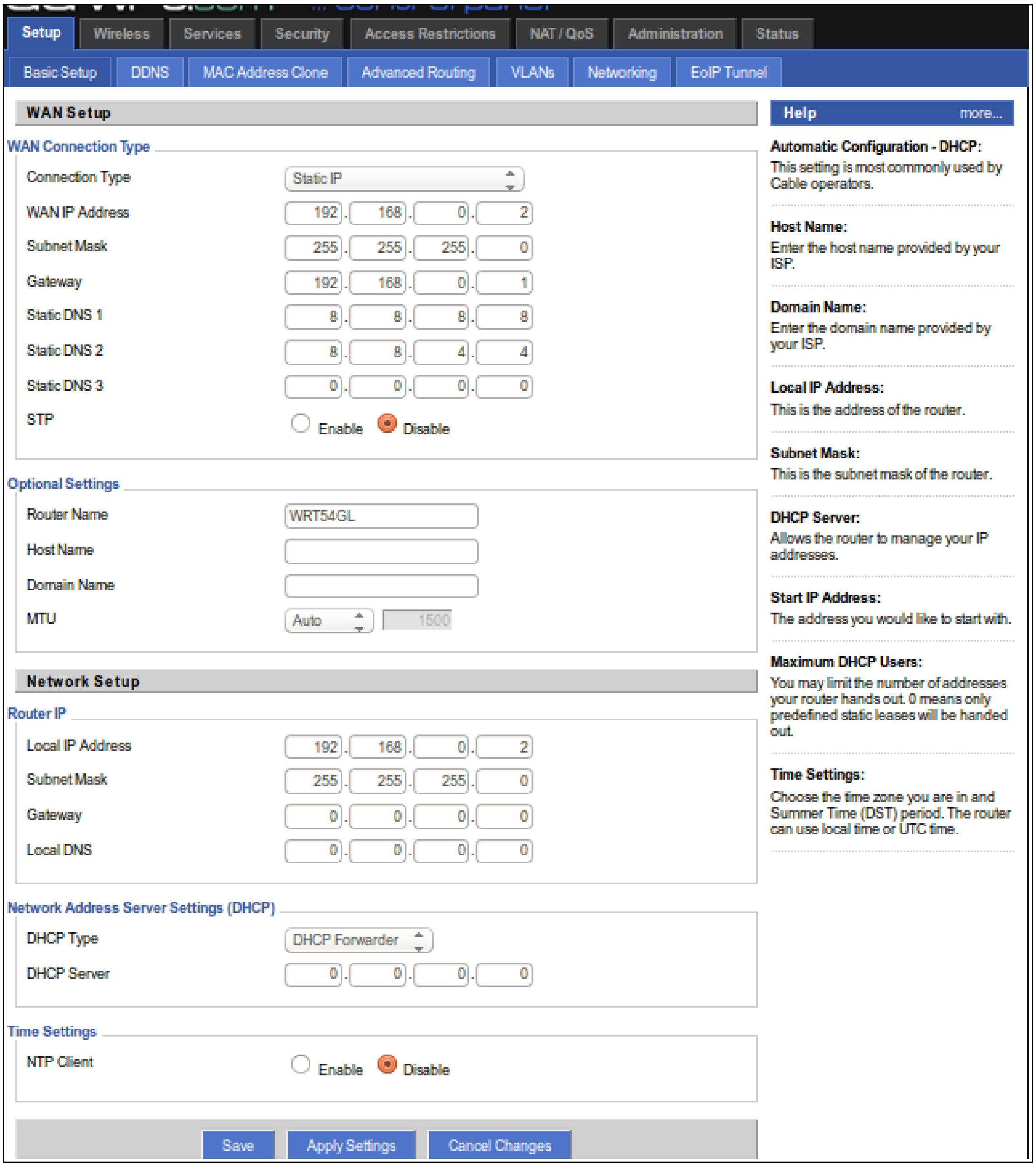
על מנת שנוכל לייצא אל הקובץ בעזרת iptables-save, נקליד:

```
root@ws:~# iptables-save > /rw/config/rules.iptables
```

בשלב הזה TOR Server צפוי לחכות לתעבורה, עכשיו נשאר לנו להגדיר את ה-AP, יש לנו שתי אפשרויות עיקריות:

- ה-AP משמש כ-Switch ולשם כך נצטרך להתקין DHCP על TOR Server, לינק להתקנת dhcp ב-Debian למטה.
- ה-AP משמש כ-Router ומחלק כתובות עם DHCP, נצטרך להגדיר לו את יציאת ה-WAN בתור IP סטטי באותו ה-Subnet עם TOR server.

הגדרות ה-switch שהשתמשתי בהן מופיעות בתמונה הבאה:



סיכום

בחרתי להשתמש ב-QubesOS בשילוב עם Whonix מפני שכשהתחלתי לחקור את הנושא זה הפתרון המלא הראשון שהכרתי. את אותו הפתרון אפשר לממש בדרכים אחרות, לדוגמה - RaspberyPi עם כרטיס Ethernet נוסף. יתרון טוב בלהשתמש ב-QubesOS הוא הקלות להוסיף הגנה על הרשת הבייתית עם Snort שירחרח את התעבורה, להוסיף מכונת Honeypot או להתקין קושחת חומת אש וירטואלית במקום Sys-firewall.

זו היא דרך אחת לממש נתב אל רשת TOR על שרת יעודי, את Whonix-gw אפשר להתקין גם על VirtualBox לדוגמה, או להחליף ב-TorVM בקלות לפי המדריך הזה:

www.qubes-os.org/doc/privacy/torvm

מה עושים ב-TOR?

דבר ראשון יהיה לבחון את Hidden Wiki, אתר וויקיפדיה שאנשים אנונימים מפרסמים לינקים אל האתרים שלהם, עם תקציר לאיזה סוג שירות הם מציעים, יש שם מגוון מאוד רווח ומעניין של השירותים שאנשים מציעים. בנוסף ל-Hidden Wiki ישנם גם מנועי חיפוש בתוך רשת TOR, עם מגוון של כתובות אתרים.

אפשר לכוון יותר ממכונה אחת להעביר את המידע שלה דרך Whonix-gw, לדוגמה מכונת Whonix-ws או את ה-DisposableVM.

אני מקווה שנהנתם, כמובן שלא סיקרתי את כל הנושא ככה ששאלות, תהיות, הארות והערות יתקבלו בברכה כאן בתגובות או בכתובת האימייל: Eden2036@gmail.com

לינקים

Routing:

- https://wiki.debian.org/DHCP_Server
- https://openwrt.org

Switch:

- http://www.dd-wrt.com/wiki/index.php

TORcheck:

- https://check.TORproject.org

Copy to dom0:

- https://www.qubes-os.org/doc/copy-to-dom0

הכרת SCTP - חלק שני

מאת עידו קנר

הקדמה

בחלק הראשון הסברתי בקצרה יחסית על פרוטוקול ה-SCTP, אשר מאפשר להכיל בתוכו הרבה מאוד הודעות המיועדות לגבי יעד או מספר יעדים מוגדרים, לרוב פנייה לשרת אחר כאשר השרת הראשי נכשל. על מנת להצליח להעביר מידע שונה, מבנה הבקשה חייבת להיות סוג של קפסולה, המאפשרת בעצם להכניס בתוכה מידע המתכנס בתוך מבנה מוגדר מראש. כל חלק מכיל מבנה מוגדר, ובחלקים אלו הם אגע במאמר זה...

חשוב לי להדגיש כי אין מאמר זה מגיע להיות שלם, או ממצא אודות הפרוטוקול, אך הוא כן מנסה לעשות סדר ולאסוף מידע ממספר מקורות בנושאים המדוברים ולרכזם במקום אחד.

במאמר לא תמצאו את החוקים השונים, כיצד ומתי יש לעשות דברים מעבר להגדרת סוג הבקשה, היות ונושא זה דורש סדרת מאמרים שלמה בפני עצמה.

קיצורים ומונחי יסוד

כאשר מדברים על SCTP ישנם מספר קיצורים ומונחי יסוד, אשר חלקים גם מופיעים כאן במאמר:

מונח	פירוש
MAC	Message Authentication Code - גישה לבדיקת תקינות המבוססת על פונקציות Hash קריפטוגרפיות (בגישה דומה ל-HMAC - RFC2104 אך לרוב לא זהה) באמצעות מפתח פרטי - סימטרי - כלומר משותף לכל הצדדים הנוגעים בדבר, על מנת לוודא את אמינות המידע בין הצדדים.
RTO	Retransmission Timeout
RTT	Round-Trip Time
SCTP	Stream Control Transmission Protocol
RTTVAR	Round-Trip Time Variation
SRTT	Smoothed RTT
TCB	Transmission Control Block
TLV	Type-Length-Value coding format

TSN	Transmission Sequence Number
ULP	Upper-Layer Protocol
User	כאשר מדברים על משתמש, מדברים על התוכנה (שכבה שבע) - User Application .
Message	הודעה מצד המשתמש המוגשת ל-SCTP על ידי ה-ULP.
Multiplexing	איגוד מספר פעולות שונות תחת אותה קורת גג.
Bundling	פעולה אופציונלית של Multiplexing אשר יכולה להכיל יותר מהודעת המשתמש אחת באותה פקטת SCTP.
Congestion Windows	משתנה האחראי על הגבלת גודל המידע בבתים שניתן לשלוח ליעד מסוים לפני שמקבלים אישור.
Chunk	"חתיכה", יחידה של מידע בתוך פקטת SCTP אשר מורכבת מ"ראש" המתאר את החתיכה והתוכן של החתיכה (Payload).
Cumulative TSN Ack Point	מספר הסופי לחתיכה האחרונה, והגיע עליה אישור.
Inactive destination	יעד אשר מכיל שגיאות, או אינו זמין לקבל הודעות.
Network Byte Order	השימוש ב-Big Endian - שהבתים החשובים (MSB) נמצאים בהתחלה.
Ordered Message	הודעת משתמש אשר נשלחה לפי הסדר, עם התחשבות לכל הודעות העבר של המשתמש שנשלחו.
Outstanding TSN	מספר TSN אשר שייך לחתיכה אשר נשלחה ליעד, אך טרם התקבל עליה אישור.
Path	הנתיב אשר נלקח על ידי פקטת SCTP אשר נשלחה ליעד מסוים לפי כתובת. כאשר, שליחה ליעד אחר, אינה מבטיחה נתיב שונה.
PMTU	נתיב MTU, היכולת להבטיח נתיב לפי גודל ה-MTU.
rwnd	**Receiver Window** - משתנה של SCTP המחזיק בתוכו את החישוב העדכני ביותר לחלון השליחה של היעד, ונשמר בבתים. זה מאפשר לשולח הבנה של הגודל הקיים אצל היעד המקבל את הבקשה.

Primary Path	הנתיב המרכזי וברירת המחדל ליעד ולמקור ההודעות. ההגדרה מחזיקה בתוכה את כתובת המקור, ולא רק את כתובת היעד, היות והמימוש ירצה לרוב לדעת כיצד לחזור למקור על ידי שליטה בנתיב התעבורה עצמו.
ECN	**Explicit Congestion Notification** - מונח אשר נלקח מ-RFC 3168, ומתאר תוספת עבור IP אשר מחזיקה מתודה אשר "יודעת" מתי יש בעיית congestion כאשר יש איבוד מידע במצב של datagram. זוהי תוספת עבור SCTP שלא חובה למימוש, ויש בה הבדלים קטנים מ-RFC3168, אך מתבססת על הRFC המקורי.

מבנה SCTP

הרעיון הכללי במבנה של SCTP נראה כך:

```
                    SCTP User Application

      -----------------------------------------------------
       _____________                  ____________________
      |             |                | Sequenced Delivery |
      | Association |                |   within Streams   |
      |             |                |____________________|
      |   Startup   |
      |             |         ____________________________
      |     and     |        |    User Data Fragmentation |
      |             |        |____________________________|
      |   Takedown  |
      |             |         ____________________________
      |             |        |     Acknowledgement        |
      |             |        |          and               |
      |             |        |    Congestion Avoidance    |
      |             |        |____________________________|
      |             |
      |             |         ____________________________
      |             |        |        Chunk Bundling      |
      |             |        |____________________________|
      |             |
      |             |     ________________________________
      |             |    |      Packet Validation         |
      |             |    |________________________________|
      |             |
      |             |     ________________________________
      |             |    |     Path Management            |
      |_____________|    |________________________________|

        Figure 2: Functional View of the SCTP Transport Service
```

[התרשים נלקח מ-RFC 4960]

בחלק הראשון הסברתי כי ישנם מספר הודעות, בהם התחלה וסיום תקשורת. ניתן לראות כי בקשה שכזו משורטטת בנפרד מהשאר.

בהודעה שאינה עומדת בפני עצמה, ישנם מספר חלקים עליהם אני הולך לדבר על המבנה שלהם:

- סדר פעולות שליחה של זרימה - כלומר הודעות המשתמש הנשלחות על ידי ULP לפי הסדר.
- פרגמנטצית הודעות משתמש - כלומר חלוקת הודעות משתמש על מנת לעמוד בחוקי PMTU.
- קבלה ומניעת גודש - כלומר יצירת TSN לכל מידע משתמש באשר הוא.
- חבילות מידע - כלומר המידע בפועל שעובר, בין אם מידע משתמש או הודעת פרוטוקול.
- אימות פקטות - כלומר אחזקת ה-checksum לוודא תקינות של הפקטה.
- ניהול נתיב - המידע אודות הניתוב של הבקשה

מבנה של פקטת SCTP:

מבנה של פקטת SCTP בגישה הכוללת, יחסית פשוטה כאשר לא נכנסים לפרטים:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Common Header                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Chunk #1                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           ...                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Chunk #n                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

תפקיד חלק זה של המאמר הוא כן ללמד על הפרטים השונים במבנה, וזה מהות מאמר זה, מיד לאחר שאסביר בקצרה מה רואים בתרשים.

התרשים מציג לנו הגדרות בסיס כהתחלה. לאחר מכן אפשר להחזיק חתיכות מידע שונות, אשר הכמות שלהם תשתנה בהתאם לסוג הבקשה, או לגודל ה-MTU.

כפי שהזכרתי בחלק הקודם, רק בקשות מסוג INIT, INIT ACK ו-SHUTDOWN COMPLETE יהיו לגמרי לבד, בעוד ששאר הבקשות יכולות להיות משורשרות עם מידע נוסף, ובכך לבצע multiplexing של המידע. במידה והמידע גדול מידי לגודל ה-MTU, יבוצע פרגמנטציה של המידע, וישלח בחלקים אשר יתאימו לגודל הנכון.

הגדרות ראש

מבנה Common Header נראות כך:

```
                    SCTP Common Header Format

     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     Source Port Number        |     Destination Port Number   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      Verification Tag                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                           Checksum                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

Source Port Number - שש עשרה הביטים הראשונים (מספר מ-1 ועד 65,535) שייכים לפורט המקור, אשר ביחד עם פורט היעד יכולים לזהות את הבקשה וסוג הפניה.

Destination Port Number - שש עשרה הביטים השניים (מספר מ-1 ועד 65,535) שייכים לפורט היעד. פורט זה מייצג את הפורט אליו הבקשה זו מיועדת. הפורט יסייע למקבל לבצע פתיחה של ה-multiplexing ליעד הנכון, בין אם כתובת מסוימת או האפליקציה הנכונה.

Verification Tag - שלושים ושניים הביטים הבאים (מ-0 ועד 4,294,967,295) הם עבור טאג מזהה אשר מסייע לדעת מי השולח של פקטת ה-SCTP. בעת שליחת הבקשה, הערך של השדה הזה חייב להיות זהה לערך שנשלח בזמן האתחול (לא פעולת הINIT) של הבקשה, למעט המקרים הבאים:

- פעולת ה-INIT - הערך חייב להיות 0
- פעולת SHUTDOWN COMPLETE עם ביט T חייבת להיות זהה לטאג הנשלח בעת פעולת ה-SHUTDOWN ACK.
- פעולת ABORT אשר נשלחת כאחת מההודעות, יכולה להעתיק את הטאג של הפקטה עליה מבצעים את פעולת ה-ABORT.

פעולת INIT חייבת להיות ה-chunk היחיד בפקטת SCTP אשר תחזיק את הערך 0.

Checksum - שלושים ושניים הביטים הבאים (מ-0 ועד 4,294,967,295) מחזיק את חתימת פקטת ה-SCTP. כפי שהזכרתי בחלק הקודם, האלגוריתם למימוש הוא CRC32c.

שדה ה-Chunk

שדה ה-chunk מוגדר בצורה הבאה:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Chunk Type  | Chunk  Flags  |        Chunk Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                          Chunk Value                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

כל חלק מסוג Chunk יהיה במבנה הבא:

Chunk Type - שמונה ביטים (מ-0 ועד 254), השדה מחזיק בסוג המידע שחלק זה מחזיק. כאשר השדה יכול להחזיק ערכים מ-0 ועד 254, כאשר 255 הוא ערך שמור לשימוש עתידי.

סוגי חתיכות לוקחים בחשבון את הרעיון של ביטים גבוהים ונמוכים. 2 הביטים הגבוהים מציינים את סוג הפעולה שתתבצע במידה ולא ידוע מה סוג החתיכה.

סוגי הביטים הם:

סוג הביט	תאור
00	הפסק לטפל בבקשה זו והתעלם ממנה, ואל תטפל יותר בבקשות של אותה סוג חתיכה
01	הפסק לטפל בבקשה זו והתעלם ממנה, אל תטפל יותר בבקשות של אותה סוג חתיכה, ודווח על כך
10	דלג על החתיכה והמשך הלאה
11	דלג על החתיכה והמשך הלאה, אבל דווח על החתיכה הזו ERROR, עם ההודעה של 'Unrecognized Chunk Type'.

סוגי הודעות/חתיכות:

קוד	סוג הודעה	הסבר
0	DATA	המידע עצמו (payload)
1	INIT	התחלת שליחה
2	INIT ACK	אישור קבלת השליחה
3	SACK	אישור חלקי בדבר קבלת השליחה
4	HEARTBEAT	בקשת HEARTBEAT
5	HEARTBEAT ACK	אישור על בקשת ה-HEARTBEAT
6	ABORT	בטל פעולות
7	SHUTDOWN	בקשה לסיום מסודר של תשדורת
8	SHUTDOWN ACK	אישור על קבלת SHUTDOWN
9	ERROR	שגיאת פעילות (operation error) - אינה נחשבת למשהו שלא ניתן להמשיך דרכו, בניגוד ל-ABORT, אך יכולה לבוא בשימוש גם של ABORT לשם כך.
10	COOKIE ECHO	התחלה של עוגיות (ארחיב על הנושא בחלק אבטחת המידע)
11	COOKIE ACK	אישור קבלת עוגייה
12	ECNE	שמור עבור הכרזת congestion
13	CWR	שמור עבור הכרזת הפחתת חלון שליחה
14	SHUTDOWN COMPLETE	סיום פעולת הכיבוי
15-62	פנוי	
63	שמור	
64-126	פנוי	
127	שמור	
128-190	פנוי	
191	שמור	
192-254	פנוי	
255	שמור	

Chunk Flags - שמונה ביט (מ-128 ועד 127). הדגלים משתנים בהתאם לסוג החתיכה, ובמידה ולא צוין אחרת, הערך תמיד יעמוד על 0 מצד המשדר, ותמיד יזכה להתעלמות מצד המקבל.

Chunk Length - שש עשרה ביט (מ-0 ועד 65,535). השדה מייצג את גודל המידע בבתים וכולל בתוכו גם את הגודל של כל החבילה, שזה אומר Chunk Type, Chunk Flags, Chunk Length ו-Chunk Value.
כך שאם אין מידע ב-Chunk Value, הגודל שלו יהיה 4 בתים.

האורך אינו מחשב איזשהו "ריפוד" (padding) של אפסים, וזה למרות שהשדות יכולים להחזיק ריפוד שכזה, במידה ולא כל המקום שלהם תפוס, ולא יעלה על ארבעה בתים, כאשר הריפוד עצמו לא יעלה על 3 בתים. כלומר בית אחד שמור לערכים, והשאר (עוד שלושה בתים) יהיו הריפוד. בנוסף, המידע על האורך של Chunk Value יהיה למעט הערך האחרון בו, המציין את הגבול שבו המידע נגמר.

במידה ובונים נכון את התמיכה, ניתן לעבוד עם Chunk Length גם כאשר האורך מציין רק את גודל המידע ולא כולל בתוכו את הגודל הכולל של הכל.

האורך המקסימלי של חתיכה (כולל את כל השדות) חייב להיות כפולה של ארבע בתים. במידה והאורך של חתיכה אינו כפולה של ארבע בתים, השולח חייב לבצע את פעולת הריפוד עם אפסים והריפוד הזה אינו נכלל כחלק מחישוב Chunk Length. השולח כאמור חייב לא לכלול יותר משלושה בתים בריפוד. כאשר המקבל כאמור, חייב להתעלם מהריפוד.

Chunk Value - גודל משתנה. שדה זה מכיל את המידע עצמו אותו יש להעביר בחתיכה. השימוש ופורמט המידע נקבע על ידי Chunk Type.

המידע בנושא יורחב בהמשך המאמר.

פרמטר אופציונלי

Chunk Value ב-SCTP יכול לכלול בתוכו פרמטר אופציונאלי, בעל המבנה הבא:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Parameter Type       |       Parameter Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                       Parameter Value                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

Chunk Parameter Type - שש עשרה ביט (מ-0 ועד 65534, כאשר 65535 שמור לתוספי IETF). הערכים עצמם מוגדרים בטבלת ה-SCTP Chunk שמורים לשימוש של IETF.

סוגי חתיכות לוקחים בחשבון את הרעיון של ביטים גבוהים ונמוכים. 2 הביטים הגבוהים מציינים את סוג הפעולה שתתבצע במידה ולא ידוע מה סוג החתיכה.

סוגי הביטים הם:

סוג הביט	תאור
00	הפסק לטפל בבקשה זו והתעלם ממנה, ואל תטפל יותר בבקשות של אותה סוג חתיכה
01	הפסק לטפל בבקשה זו והתעלם ממנה, אל תטפל יותר בבקשות של אותה סוג חתיכה, ודווח על כך
10	דלג על החתיכה והמשך הלאה
11	דלג על החתיכה והמשך הלאה, אבל דווח על החתיכה הזו ERROR, עם ההודעה של 'Unrecognized Chunk Type'.

Chunk Parameter Length - שש עשרה ביט (מ-0 ועד 65535). השדה מחזיר את האורך בבתים, כאשר האורך כולל את Parameter Type, את Parameter Length ו-את Parameter Value. הגודל המקסימלי הוא ארבעה בתים, כאשר לא כוללים בתוכם את ה-padding שיש בשביל לשמור על הגודל. כמו בכל שאר המקומות בפרוטוקול בהם מוגדר אורך, גם כאן, ה-padding מתבצע לימין לערך, ולא יעלה על 3 בתים סה"כ.

Chunk Parameter Value - גודל משתנה. המידע בפועל שנשלח, בהתאם כמובן לשאר הפרמטרים.

חתיכות

ישנם מספר סוגי חתיכות:

Payload - כאשר רוצים לשלוח מידע בפועל, יש להשתמש במבנה הבא:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0    | Reserved|U|B|E|    Length                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              TSN                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Stream Identifier S      |   Stream Sequence Number n    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Payload Protocol Identifier                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                 User Data (seq n of Stream S)                 /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type תמיד יהיה אפס, והוא מתאר למעשה שמדובר ב-Payload.
- Reserved מחזיר בחמישה ביטים. לרוב הוא יהיה בערך של '0', ויזכה להתעלמות מהמקבל.

לכל ביט בשדה, יש הגדרה:

סוג הביט	תאור
U	פירוש הביט הוא Unordered bit. במידה והוא מחזיק בערך '1', הוא מציין כי החתיכה מחזיקה מידע ללא סדר מסוים. לכן גם לא יהיה Stream Sequence Number אשר ישויך לחתיכה, ולכן המקבל חייב להתעלם מהשדה של Stream Sequence Number. כאשר מרכיבים חזרה את המידע (במידה ויש צורך לכך), החתיכה חייבת להישלח ל-ULP על ידי המקבל, ללא ניסיון לבצע סדר מסוים במידע. במידה ומידע לא מסודר הוא גם מחולק (fragmented), כל חלק של ההודעה חייב להכיל את הביט עם הערך '1'.
B	ביט זה מציין האם החלק הוא התחלת המידע, כאשר מדובר בשליחה מחולקת.
E	ביט זה מציין האם החלק הוא סוף המידע, כאשר מדובר בשליחה מחולקת.

כאשר המידע אינו מחולק (fragmented) ישתמש בביטים של B ו-E בצורה הבאה:

1/0		תאור
1	0	התחלת המידע המחולק
0	0	אמצע המידע המחולק (כל שאר המידע למעט הסוף שנשלח)
0	1	סוג המידע המחולק
1	1	מידע לא מחולק

כאשר המידע מחולק למספר חתיכות, הTSN יהיה בשימוש על ידי המקבל בשביל להרכיב מחדש את המידע. דבר זה מחייב את השולח לשים כל מידע מחולק תחת TSN שהוא במספר סידורי עולה עבור כל חלק שנשלח.

Length - שש עשרה ביט. השדה מחזיק את גודל חתיכת ה-DATA בבתים, מהתחלת השדה, ועד לסיום החתיכה, לא כולל padding.

אורך של חתיכה עם בית בודד תהיה בעלת הספרה 17 - כלומר 17 בתים.

חתיכת DATA עם מידע משתמש בעל אורך של L תהיה באורך של L+16, אשר מציינת למעשה את הגודל של שש עשרה פלוס L בבתים. האורך של מידע המשתמש חייב להיות גדול יותר מ-0, כלומר L חייבת להיות גדולה מ-0 בנוסחה.

TSN - שלושים ושתיים ביט. הערך מייצג את ה-TSN עבור חתיכת ה-DATA. האורך המותר ל-TSN הוא מ-0 עד 4,294,967,295 (כלומר $32^2 - 1$). כאשר הערך האחרון של 4,294,967,295 מגיע, השדה של TSN חייב להתאפס ולהחזיק את הערך '0'.

Stream Identifier S - שש עשרה ביט (מ-0 ועד 65,535). זהו מזהה ייחודי לסוג הstream אשר המידע שייך אליו.

Stream Sequence Number - שש עשרה ביט (מ-0 ועד 65,535). זהו מזהה חד חד ערכי עבור ה-stream, גם כאשר מדובר במידע המחולק על פני מספר בקשות, עד לסיום השליחה של הערך האחרון.

Payload Protocol Identifier - שלושים ושניים ביט (unsigned). המידע מייצג את התוכן של האפליקציה (או ULP) לפי התאור של protocol identifier.

המידע מועבר ל-SCTP על ידי ה-ULP, ונשלח ליעד. זהו מזהה אשר אינו בשימוש של SCTP, אך ישויות אחרות יכולות להשתמש בו בנוסף לאפליקציה עצמה, על מנת לזהות את סוג המידע שנשלח כחלק מהמידע עצמו.

השדה חייב להישלח גם כאשר מדובר במידע מחולק, וזאת על מנת להבטיח כי כל מי שרוצה לדעת להשתמש במידע הזה יוכל.

היות והמידע של שדה זה אינו חלק מה SCTP, לכן אינו מנוהל על ידו, התוכן יכול להיות כל מה שרוצים, כמו למשל מספר שהוא Little Endian (למרות שמקובל בתקשורת שהערכים הם Big Endian), וכיו"ב. האחראי למידע הוא רק ULP.

הערך '0' לשדה זה, אומר כי אין מזהה לאפליקציה ל-Payload הזה.

User Data - אורך משתנה. זהו המידע בפועל של המשתמש. המימוש חייב לשים padding של ארבעה בתים בגודל 0 בסוף. אסור לpadding להיות כלול באורך השדה. אסור לשולח לשים יותר מ-3 בתים ל-padding. כלומר הבית הראשון יהיה המידע עצמו.

הדגמת wireshark לבקשת DATA SCTP (מהקישור הבא):

```
Frame 25: 94 bytes on wire (752 bits), 94 bytes captured (752 bits)
Ethernet II, Src: Vmware_69:f8:e7 (00:0c:29:69:f8:e7), Dst: Vmware_5b:32:ba (00:0c:29:5b:32:ba)
Internet Protocol Version 4, Src: 192.168.10.105, Dst: 192.168.10.104
Stream Control Transmission Protocol, Src Port: 12345 (12345), Dst Port: 12345 (12345)
    Source port: 12345
    Destination port: 12345
    Verification tag: 0x6ad4dac4
    [Assocation index: 9]
    Checksum: 0x72adaeb4 (not verified)
    DATA chunk(ordered, complete segment, TSN: 3096428048, SID: 0, SSN: 0, PPID: 1234, payload length: 31 bytes)
        Chunk type: DATA (0)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x03
            .... ...1 = E-Bit: Last segment
            .... ..1. = B-Bit: First segment
            .... .0.. = U-Bit: Ordered delivery
            .... 0... = I-Bit: Possibly delay SACK
        Chunk length: 47
        Transmission sequence number: 3096428048
            [This chunk is acked in frame: 27]
                [The RTT to SACK was: 0.001208000 seconds]
        Stream identifier: 0x0000
        Stream sequence number: 0
        Payload protocol identifier: Unknown (1234)
        Chunk padding: 00
Data (31 bytes)

0000  68 65 6c 6c 6f 20 77 6f 72 6c 64 20 66 72 6f 6d   hello world from
0010  20 31 39 32 2e 31 36 38 2e 31 30 2e 31 30 35       192.168.10.105
    Data: 68656c6c6f20776f726c642066726f6d203139322e313638...
    [Length: 31]
```

INIT

חתיכת ה-INIT נמצאת בשימוש על מנת לאתחל את התקשורת של SCTP בין הנקודות תקשורת השונות (כפי שהוסבר בחלק הקודם).

מבנה ה-INIT הוא כזה:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 1    |  Chunk Flags  |      Chunk Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Initiate Tag                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Advertised Receiver Window Credit (a_rwnd)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Number of Outbound Streams   |  Number of Inbound Streams    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Initial TSN                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/              Optional/Variable-Length Parameters              /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

חתיכת ה-INIT מכילה בתוכה את הפרמטרים הבאים רק פעם אחת, אלא אם צוין אחרת:

האם חובה	פרמטר
✓	Initiate Tag
✓	Advertised Receiver Window Credit
✓	Number of Outbound Streams
✓	Number of Inbound Streams
✓	Initial TSN
✗	IPv4 Address[1]
✗	IPv6 Address[1]
✗	Cookie Preservative
✗	Reserved For ECN Capable[2]
✗	Host Name Address[3]
✗	Supported Address Types[4]

1 - חתיכת ה-INIT יכולה להחזיק מספר כתובות IP שיכולות להיות במבנה של IPv4 ו-IPv6 באפשרויות שונות בהתאם לצורך.
2 - כל שדה התומך ב-ECN הוא "שמור" לשימוש עתידי עבור Explicit Congestion Notification.
3 - אסור לחתיכת INIT יותר מכתובת Host Name Address אחת. בנוסף, לשולח בקשת ה-INIT אסור לצרף עוד כתובות תחת Host Name Address בבקשת ה-INIT עצמה. המקבל של בקשת ה-INIT חייב להתעלם מכל תוספת של Host Name Address, במידה ושדה זה קיים.
4 - במידה ופרמטר זה קיים, הוא מחזיק ברשימה של כל סוגי הכתובות שהשולח יכול לתמוך בהם. העדר של השדה, מייצג שהשולח מסוגל להתמודד עם כל סוג של כתובת.

הערת מימוש:

במידה וחתיכת INIT מתקבלת עם פרמטרים מוכרים שהם לא אופציונאלים, בחתיכת ה-INIT, אז מומלץ למקבל לעבד את בקשת ה-INIT ולשלוח תשובה של INIT ACK. המקבל של בקשת ה-INIT יכול במידת הצורך לשלוח חתיכה של ERROR עם COOKIE ACK מאוחר יותר. אך מימוש מגביל יכול לשלוח חזרה חתיכת ABORT כתשובה לבקשת INIT.

- Type - תמיד יחזיק בערך 1.
- Chunk Flags - תמיד יהיה בערך 0, היות והוא שמור במצב חתיכת INIT. בנוסף, על המקבל להתעלם שדה זה. סדר הפרמטרים בINIT יכול להגיע בכל סדר.
- Initiate Tag - שלושים ושניים ביט (מ-1 ועד 4,294,967,295). מקבל בקשת ה-INIT שומר את ערך שדה זה. הערך של השדה חייב להיות בשדה Verification Tag של כל בקשת SCTP אשר נשלחת חזרה ממקבלי בקשת ה-INIT. למעט הערך 0, השדה יכול להחזיק כל ערך בטווח. במידה והמקבל גילה כי הערך הוא 0, מקבל הבקשה חייב להתנהג אל הבקשה כאילו יש בה שגיאה, ולדווח חזרה ABORT.

הערות:

- זהו שדה המסייע להתמודדות עם פעולות של MiTM ולכן חשוב שהערך בו יהיה רנדומלי, על מנת לסייע בהגנה מהתקפות של MiTM ו-Sequence Number. ניתן להשתמש בגישה של RFC4086, המתארת כיצד ניתן לבצע רנדומיזציה של Initiate Tag.
- בחירה זהירה של Initiate Tag, תסייע בהתמודדות של חוסר שכפול פקטות ממידע שכבר נשלח בעבר.
- אסור ש-Initiate Tag ישתנה על ידי איזשהו צד, כל עוד הבקשות בנושא עדיין חיות.

Advertised Reciver Window Credit (a_rwnd) - שלושים ושניים ביט (מ-0 ועד 4,294,967,295). שדה זה מייצג גודל באפר יעודי לפי בתים אשר שולח בקשת ה-INIT שמר עבור שיוך החלון.

לאורך זמן החיים של השיוך, גודל האפר אינו יכול להיות מוסר מהשיוך, אך הקצה יכול לשנות את הגדול על ידי שליחה של בקשת SACK.

Number of Outbound Stream (OS) - שש עשרה ביט (מ-1 ועד 65,535). מציין את הערך של outbound stream של בקשת ה-INIT על ידי השולח, אשר מעוניין לשייך לבקשה. אסור שהערך יהיה 0. במידה ומקבל בקשת ה-INIT, קיבל את OS עם הערך 0, עליו לבטל כל שיוך בנושא.

<u>Number of Inbound Streams (NIS)</u> - שש עשרה ביט (מ-1 ועד 65,535). מציין את הערך המקסימלי אשר שולח בקשת ה-INIT מרשה למקבל הבקשה לשייך. אסור שהערך של שדה זה יהיה 0. במידה ונשלח ערך 0 בשדה הזה, על המקבל בטל כל שיוך בנושא. חשוב לציין כי אין באמת משא ומתן למספר ה-stream בנושא, אך במקום משא ומתן, שני הצדדים ישתמשו בגישה הבאה: min(requested, offered). בחלק ה-INIT ACK אסביר זאת יותר לעומק.

<u>Initial TSN (I-TSN)</u> - שלושים ושניים ביט (מ-0 ועד 4,294,967,295). שדה זה מגדיר TSN התחלתי אשר השולח ישתמש בו. שדה זה יכול להחזיק גם את הערך הקיים בשדה Initiate Tag.

<u>Optioanl/Variable-Length Parameter</u> - גודל משתנה. השדה הבא הוא למעשה מבנה בפורמט המשתנה בהתאם לסוג המבנה:

- IPv4
- IPv6
- Cookie Preservative
- Host Name Address
- Supported Address Types

כתובת IPv4 במבנה הבא:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Type = 5               |      Length = 8               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        IPv4 Address                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- <u>Type</u> - תמיד יהיה 5.
- <u>Length</u> - תמיד יהיה 8 (בתים).
- <u>IPv4 Address</u> - שלושים ושניים ביט - לפי הגדרת RFC791. השדה מחזיק בערך המקודד בינארית.

כתובת IPv6 תהיה במבנה הבא:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 6           |          Length = 20          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                         IPv6 Address                          |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד מחזיק בערך 6.
- Length - תמיד מחזיק בערך 20 (בתים).
- IPv6 Address - מבנה של IPv6 לפי RFC2460 בקידוד בינארי.

אסור לשולח להשתמש בגישה של IPv4 ממופה לכתובת IPv6 (אשר מתואר ב-RFC2491), ובמקום, יש להשתמש במבנה של IPv4 שתואר למעלה.

הערה:
במידה ואין שימוש ב-IPv4 ו/או IPv6 בשדה של Optioanl/Variable-Length Parameter בפעולות של INIT ו-INIT ACK, יהיה קל יותר לתפעל את הבקשות תחת NAT.

Cookie Preservative - המבנה הבא מציין כמה זמן עוגייה תהיה בחיים:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type = 9             |          Length = 8           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Suggested Cookie Life-Span Increment (msec.)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד יהיה 9.
- Length - תמיד יהיה 8 (בתים).
- Suggested Cookie Life-Span Increment - שלושים ושניים ביט (מ0 ועד 4,294,967,295). הפרמטר מייצג מה הערך להוספה במילי-שניות אשר השולח מעוניין שהמקבל יוסיף לאורך זמן החיות של העוגייה.

פרמטר אופציונאלי זה, יתווסף ל-INIT על ידי השולח כאשר השולח מנסה לחדש שיוך עם המקבל אשר נכשל בעבר בשל הודעת שגיאה של cookie operation error. המקבל יכול להתעלם את הערך המוצע, בשביל לא להיכנס לבעיית אבטחה כלשהי.

שדה אופציונאלי של Host Name Address - שולח בקשת ה-INIT ישתמש בפרמטר הזה על מנת לספק שם מתחם במקום כתובות IP למקבל הבקשה. מקבל הבקשה אחראי על ביצוע פעולת ה-resolving לשם המתחם. שימוש בפרמטר זה, יכול לסייע בעבודה מול NAT.

המבנה שלו נראה כך:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type = 11            |          Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                          Host Name                            /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד יהיה 11.
- Length - ישתנה בהתאם להגדרות Host Name.
- Host Name - גודל משתנה. השדה מכיל שם מתחם לפי הגדרות של RFC1123 חלק 2.1.
- SCTP אינו אחראי על ביצוע פעולת ה-resolving.

חשוב לציין כי לפחות תו null אחד יהיה קיים ב-Host Name, וחייב להיכלל גם באורך.

שדה אופציונאלי של Supported Address Types, מחזיק ברשימה של כתובות IP ושמות מתחם בהם השולח תומך:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type = 12            |          Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Address Type #1        |        Address Type #2        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            ......                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-++-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד יהיה 12.
- Length - משתנה בהתאם לכמות הכתובות שיש.
- Address Type - שש עשרה ביט. מחזיק בתוכו את ה-Type של 4IPv ו-Host Name שנתמכים.

הדגמת wireshark לבקשת SCTP INIT (מהקישור הבא):

```
Frame 1: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)
Ethernet II, Src: Vmware_69:f8:e7 (00:0c:29:69:f8:e7), Dst: Vmware_5b:32:ba (00:0c:29:5b:32:ba)
Internet Protocol Version 4, Src: 192.168.10.105, Dst: 192.168.10.104
Stream Control Transmission Protocol, Src Port: 12345 (12345), Dst Port: 12345 (12345)
    Source port: 12345
    Destination port: 12345
    Verification tag: 0x00000000
    [Assocation index: 0]
    Checksum: 0xeb971a79 (not verified)
    INIT chunk (Outbound streams: 1, inbound streams: 65535)
        Chunk type: INIT (1)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
        Chunk length: 36
        Initiate tag: 0xbe65a7ef
        Advertised receiver window credit (a_rwnd): 57344
        Number of outbound streams: 1
        Number of inbound streams: 65535
        Initial TSN: 864639500
        Supported address types parameter (Supported types: IPv4)
            Parameter type: Supported address types (0x000c)
                0... .... .... .... = Bit: Stop processing of chunk
                .0.. .... .... .... = Bit: Do not report
            Parameter length: 6
            Supported address type: IPv4 address (5)
            Parameter padding: 0000
        ECN parameter
            Parameter type: ECN (0x8000)
                1... .... .... .... = Bit: Skip parameter and continue processing of the chunk
                .0.. .... .... .... = Bit: Do not report
            Parameter length: 4
        Forward TSN supported parameter
            Parameter type: Forward TSN supported (0xc000)
                1... .... .... .... = Bit: Skip parameter and continue processing of the chunk
                .1.. .... .... .... = Bit: Do report
            Parameter length: 4
```

INIT ACK

בקשת ה-INIT ACK היא בקשה חוזרת המגיעה כאישור לבקשת ה-INIT:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Type = 2    |  Chunk Flags  |          Chunk Length         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         Initiate Tag                          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              Advertised Receiver Window Credit                |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Number of Outbound Streams   |  Number of Inbound Streams    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          Initial TSN                          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 \                                                               \
 /              Optional/Variable-Length Parameters              /
 \                                                               \
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

חתיכת ה-INIT ACK מכילה בתוכה את הפרמטרים הבאים רק פעם אחת, אלא אם צוין אחרת:

האם חובה	פרמטר
✓	Initiate Tag
✓	Advertised Receiver Window Credit
✓	Number of Outbound Streams
✓	Number of Inbound Streams
✓	Initial TSN
✓	State Cookie
✗	IPv4 Address[1]
✗	IPv6 Address[1]
✗	Unrecognized Parameter
✗	Reserved For ECN Capable[2]
✗	Host Name Address[3]

1 - בקשת INIT ACK מספר כתובות IP שיכולות להיות במבנה של IPv4 ו-IPv6 באפשרויות שונות בהתאם לצורך.
2 - שדה התומך ב-ECN הוא "שמור" לשימוש עתידי עבור Explicit Congestion Notification.
3 - אסור לחתיכת INIT יותר מכתובת Host Name Address אחת. בנוסף, לשולח בקשת ה-INIT אסור לצרף עוד כתובות תחת Host Name Address בבקשת ה-INIT עצמה. המקבל של בקשת ה-INIT חייב להתעלם מכל תוספת של Host Name Address, במידה ושדה זה קיים.

חשוב לדעת כי מימוש של SCTP חייב לקחת בחשבון שמקבל בקשת INIT ACK, יכול לקבל בקשות מאוד גדולות (גדולות יותר מ-1500 בתים) בשל העבודה שיש שדה משתנה שיכול להכיל מספר ערכים בפנים, בהם State Cookie ורשימת כתובות משתנה.

למשל הצד שמגיב לבקשת INIT יכול לספק רשימה של 1,000 כתובות IPv4 שהוא מעוניין לשלוח אליהם, והגודל של בקשה כזו תהיה כ-8,000 בתים עבור INIT ACK.

עוד חשוב לדעת, כי אם חתיכת INIT ACK מתקבלת עם פרמטרים שהם חובה של בקשת ה-INIT ACK, אז המקבל צריך לעבד את הבקשה ואז לשלוח חזרה חתיכת COOKIE ECHO. המקבל של חתיכת ה- INIT ACK עלול לשלוח גם חתיכת ERROR ביחד עם חתיכת ה-COOKIE ECHO.
מימוש שהוא שמרני עלול לשלוח בקשת ABORT חזרה כתגובה ל-INIT ACK שכזה.

בצירוף של כל פורט מוצא אשר נשלח כחלק מבקשת SCTP הראשית, כל כתובת IP בבקשת ה-INIT ACK מציינת למקבל של INIT ACK כתובת תעבורה חוקית ותקינה על ידי השולח של בקשת ה-INIT ACK, אשר חייבת להמשיך להתקיים לכל אורך החיים של שיוך הבקשה.

שדות:

- <u>Initiate Tag</u> - שלושים ושניים ביט (מ-1 ועד 4,294,967,295). המקבל של בקשת INIT ACK מתעד את הערך של הפרמטר. הערך חייב להיכנס לתוך שדה ה-Verification Tag של כל בקשת SCTP שעליה בוצע ה-INIT ACK, כחלק מהשיוך לבקשה זו. כפי שציון למעלה, המטרה של הערך הזה הוא להיות כמה שיותר רנדומאלי, על מנת להצליח להגן על התקפות כדוגמת MITM, Replay ו- Sequence Number. אסור שהערך ל-Initiate Tag יהיה 0. במידה והערך הוא 0, המקבל של הבקשה חייב להרוס את השיוך של הבקשה. ובמידה והמקבל מעוניין, הוא ישלח פעולת ABORT בשביל להבין מה קרה.

- <u>Advertised Reciver Window Credit (a_rwnd)</u> - שלושים ושניים ביט (מ-0 ועד 4,294,967,295). שדה זה מייצג את גודל הבאפר בבתים אשר השולח של INIT ACK ביצע כשיוך לחלון זה. כל עוד יש שיוך לבקשה, לבאפר אסור להשתחרר, והוא חייב להישאר כבאפר יעודי לבקשה המשויכת.

- <u>Number of Outbound Streams (OS)</u> - שש עשרה ביט (מ-1 ועד 65,535). שדה זה מגדיר את מספר ה"זרמים" (streams) אשר שולח ה-INIT ACK מעוניין ליצור לשיוך זה. אסור לשלוח את הערך 0 בשדה זה, או להיות גדול יותר מהערך של MIS אשר נשלח בבקשת ה-INIT. במידה והערך 0 ישלח, המקבל חייב להרוס את השיוך לבקשה ואת השיוך של TCB.

- MIS (Maximum Number Of Inbound Streams) - שש עשרה ביט (מ-1 ועד 65,535). שדה זה מחזיק את הכמות המקסימליות שהשולח של INIT ACK מורשה לצד המקבל ליצור עבור השיוך. אסור שהערך 0 ישלח בשדה זה. במידה והערך 0 ישלח, המקבל חייב להרוס את השיוך לבקשה ואת השיוך של TCB. חשוב להדגיש כי אין משא ומתן בנושא של כמות הזרמים, אך במקום, הצדדים של הבקשה ישתמשו בחישוב של (min(requested, offered.

- Initial TSN (I-TSN) - שלושים ושניים ביט (מ-0 ועד 4,294,967,295). הגדרה זו מגדירה את שדה ה-TSN אשר שולח בקשת ה-INIT ACK ישתמש. ניתן להגדיר שדה זה יהיה עם הערך של Initiate Tag.

- Optional/Variable-Length Parameters. זהו מבנה הזהה לאותו סוג נתונים של בקשת ה-INIT.

State Cookie

- Type - תמיד יהיה 7.
- Parameter Value - שדה זה חייב להכיל את כל המצבים השונים והפרמטרים שלהם אשר נדרשים כחלק מ-INIT ACK על מנת ליצור שיוך ביחד עם MAC, כפי שתואר בחלק הראשון של המאמר.

Unrecognized Parameter

- Type - תמיד יהיה 8.
- Parameter Value - אורך משנה. הערך ישלח חזרה ליוצר בקשת ה-INIT, כפעולה המחזירה את הערך הלא מוכר חזרה לשולח, על מנת שזה יחזור חזרה לשולח להבנה מלאה יותר למה שקרה.

הדגמת wireshark לבקשת ACK INIT SCTP (מהקישור [הבא](#)):

```
Frame 19: 306 bytes on wire (2448 bits), 306 bytes captured (2448 bits)
Ethernet II, Src: Vmware_5b:32:ba (00:0c:29:5b:32:ba), Dst: Vmware_69:f8:e7 (00:0c:29:69:f8:e7)
Internet Protocol Version 4, Src: 192.168.10.104, Dst: 192.168.10.105
Stream Control Transmission Protocol, Src Port: 12345 (12345), Dst Port: 12345 (12345)
    Source port: 12345
    Destination port: 12345
    Verification tag: 0x88423abf
    [Assocation index: 9]
    Checksum: 0x3151b119 (not verified)
    INIT_ACK chunk (Outbound streams: 1, inbound streams: 65535)
        Chunk type: INIT_ACK (2)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
        Chunk length: 260
        Initiate tag: 0x6ad4dac4
        Advertised receiver window credit (a_rwnd): 57344
        Number of outbound streams: 1
        Number of inbound streams: 65535
        Initial TSN: 3429330720
        State cookie parameter (Cookie length: 228 bytes)
            Parameter type: State cookie (0x0007)
                0... .... .... .... = Bit: Stop processing of chunk
                .0.. .... .... .... = Bit: Do not report
            Parameter length: 232
            State cookie: 000000000000000000000000000000000000000000000000...
        ECN parameter
            Parameter type: ECN (0x8000)
                1... .... .... .... = Bit: Skip parameter and continue processing of the chunk
                .0.. .... .... .... = Bit: Do not report
            Parameter length: 4
        Forward TSN supported parameter
            Parameter type: Forward TSN supported (0xc000)
                1... .... .... .... = Bit: Skip parameter and continue processing of the chunk
                .1.. .... .... .... = Bit: Do report
            Parameter length: 4
```

Selective Acknowledgment (SACK)

בקשת SACK נשלחת ליוצר הדיאלוג לאשר כי התקבלו חתיכות של DATA, וכן לעדכן במידה והיו פערים בין השליחה לקבלה בסדר של חלק ה-DATA. שולח בקשת ה-SACK יודע זאת, כאשר יש בעיה בסדר של TSN הנשלח ב-DATA.

שליחת SACK חייבת להכיל בתוכה:

- סיכום של TSN Ack
- Advertised Receiver Window Credit (a_rwnd)
- Number of Gap Ack Block
- מספר כפילויות של שדות ה-TSN

לפי ההגדרה, הערך של Cumulative TSN Ack הוא הערך האחרון של TSN שהתקבל לפני שהתקבלה הפסקה ברצף של מקבל ה-TSN, והערך הבא של ה-TSN עדיין לא התקבל חזרה לפני שליחת בקשת ה-SACK. כלומר שדה זה מאשר קבלה של כל שאר החלקים עד למספר זה ומשליחת ה-SACK שלפניו.

שליחת SACK יכולה להכיל בתוכה שדות של Gap Ack Blocks. שדה זה מכיל בעצם אישור על קבלת מידע של TSN עד להפסקה ברצף. במידה ואין הפסקה ברצף תוך כדי, לא ישלח השדה הזה.

לפי ההגדרה, כל שדות ה-TSN, אשר זוכים לאישור על ידי Gap Ack Blocks יהיו גדולים יותר מאשר הערך של Cumulative TSN Ack:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 3    |Chunk  Flags   |        Chunk Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Cumulative TSN Ack                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Advertised Receiver Window Credit (a_rwnd)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Number of Gap Ack Blocks = N  |  Number of Duplicate TSNs = X |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Gap Ack Block #1 Start       |   Gap Ack Block #1 End        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                                                               /
\                              ...                              \
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Gap Ack Block #N Start      |  Gap Ack Block #N End         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Duplicate TSN 1                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                                                               /
\                              ...                              \
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Duplicate TSN X                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד יהיה 3.
- Chunk Flags - שמונה ביט. תמיד יהיה 0 מרופד, והמקבל יתעלם משדה זה.
- Cumulative TSN Ack - שלושים ושניים ביט (מ-0 ועד 4,294,967,295). הפרמטר מכיל את ה-TSN האחרון שהגיע לפני הסדר לפני שנהיה פער בשליחה.
- במידה ולא נשלח עדיין מידע, הערך יהיה ערך ה-Initial TSN פחות אחד.

- <u>Advertised Receiver Window Credit (a rwnd)</u> - שלושים ושניים ביט (מ-1 ועד 4,294,967,295). השדה מכיל את הערך המעודכן לבאפר בבתים שאליו ישלח ה-SACK.

כל בקשת SACK מכילה לצד המקבל ערך ל-a_rwnd. הערך מייצג את גודל הבאפר בבתים שהתקבל מהמידע בזמן שליחת ה-SCAK, ומחזיק במה שנשאר מסה"כ המידע שהתקבל כחלק מבקשת ה-INIT/INIT ACK. כאשר יש שימוש ב-Cumulative TSN Ack ו-Gap Ack Block, שולח המידע יכול לפתח סוג של מנגון חזרה של המידע שלא נשלח, תוך שימוש בבאפר a_rwnd.

אחת הבעיות אבל בשימוש ב-SACK, הוא שהשולח חייב לקחת בחשבון כי עיבוד בקשות ה-SACK יכול להתבצע לא בסדר הנכון. כלומר בקשת SACK יכולה להישלח על ידי מקבל המידע בצורה מאוחרת יותר מהמידע הראשוני שכבר התקבל.

התקן מסביר כיצד להתמודד ולהתנהג עם התופעות האלו, תוך שימוש גם עם שדה ה-a_rwnd, אך מאמר זה לא מכסה את הנושא.

- <u>Number of Gap Ack Blocks</u> - שש עשרה ביט (מ-0 ועד 65,535). מציין את כמות הפערים בין שנכללים בבקשת SACK זו.

- <u>Number of Duplicate TSN</u> - שש עשרה ביט (מ-0 ועד 65,535). שדה זה מכיל את מספר ה-TSN שהגיעו יותר מפעם אחת. כל שכפול TSN מגיע לאחר מכן עם רשימה של Gap Ack Block.

- <u>Duplicate TSN</u> - שלושים ושניים ביט (מ-0 ועד 4,294,967,295). שדות אלו מייצגים את מספר הפעמים ש-TSN התקבל יותר מפעם אחת מאז שליחת ה-SACK האחרון. בכל פעם כאשר המקבל עותק של TSN (לפני שנלח SACK), הוא מוסיף אותו לרשימה של כפילויות. כאשר בוצעה שליחה של SACK, המספר הזה מתאפס חזרה.

Gap Ack Blocks

השדות הבאים נמצאים תחת ההגדרה של Gap Ack Blocks. הם ממשיכים אחד אחרי השני, בכמות המצויינת ב-Number of Gap Ack Blocks. כל חתיכות ה-DATA עם TSN השווה או גדול מ- Cumulative TSN Ack + Gap Ack Block Start, ושווה או קטן יותר מ-Cumulative TSN Ack + Gap Ack Block End של כל Gap Ack Block יוצאים מהנחה כי המידע התקבל כמו שצריך.

<u>Gap Ack Block Start</u> - שש עשרה ביט (מ-1 ועד 65,535). שדה זה מצייג את התחלת הקיזוז בין TSN לבין Gap Ack block זה. על מנת לחשב את מספר ה-TSN, שדה ה-Cumulative TSN Ack נוסף למספר קיזוז זה. חישוב מספר ה-TSN המחושב הוא המזהה את ה-TSN הראשון תחת Gap Ack Block שהתקבל.

<u>Gap Ack Block End</u> - שש עשרה ביט (מ-1 ועד 65,535). שדה זה מיצג את הקיזוז במידע ועד לTSN האחרון שהתקבל בפער, תחת בלוק ה-Gap Ack Block הנוכחי. על מנת לחשב את מספר ה-TSN, ערך שדה ה-Cumulative TSN Ack נוסף למספר זה. חישוב מספר TSN זה, הוא המזהה של ה-TSN האחרון תחת שדה ה-DATA עבור Gap Ack Block זה.

הדגמה לשימוש ברעיון, הוא כאשר יש הנחה שהמקבל יש את המידע הבא בחתיכות ה-DATA שרק הגיעו בזמן שהוחלט לשלוח את ה-SACK:

```
----------
| TSN=17 |
----------
|        | <- still missing
----------
| TSN=15 |
----------
| TSN=14 |
----------
|        | <- still missing
----------
| TSN=12 |
----------
| TSN=11 |
----------
| TSN=10 |
----------
```

[מתוך RFC 4960]

חלק של הפרמטר של בקשת SACK חייב להיות בנוי בצורה הבאה (בהנחה כי a_rwnd חדש מוגדר לערך של 4660 על ידי השולחת):

```
+--------------------------------+
|   Cumulative TSN Ack = 12      |
+--------------------------------+
|        a_rwnd = 4660           |
+----------------+---------------+
| num of block=2 | num of dup=0  |
+----------------+---------------+
|block #1 strt=2 |block #1 end=3 |
+----------------+---------------+
|block #2 strt=5 |block #2 end=5 |
+----------------+---------------+
```

[מתוך RFC 4960]

הדגמת wireshark לבקשת SACK SCTP (מהקישור הבא):

```
Frame 27: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
Ethernet II, Src: Vmware_5b:32:ba (00:0c:29:5b:32:ba), Dst: Vmware_69:f8:e7 (00:0c:29:69:f8:e7)
Internet Protocol Version 4, Src: 192.168.10.104, Dst: 192.168.10.105
Stream Control Transmission Protocol, Src Port: 12345 (12345), Dst Port: 12345 (12345)
    Source port: 12345
    Destination port: 12345
    Verification tag: 0x88423abf
    [Assocation index: 9]
    Checksum: 0xe14725f9 (not verified)
    SACK chunk (Cumulative TSN: 3096428048, a_rwnd: 57344, gaps: 0, duplicate TSNs: 0)
        Chunk type: SACK (3)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
            .... ...0 = Nounce sum: 0
        Chunk length: 16
        Cumulative TSN ACK: 3096428048
            [Acknowledges TSN: 3096428048]
                [Acknowledges TSN in frame: 25]
                [The RTT since DATA was: 0.001208000 seconds]
        Advertised receiver window credit (a_rwnd): 57344
        Number of gap acknowledgement blocks: 0
        Number of duplicated TSNs: 0
```

Heartbeat Request (Heartbeat)

הודעה זו נשלחת מצד אחד לשני על מנת לוודא כי היעד חי וקיים.

החלק המיועד לפרמטרים מכיל בתוכו את המידע עבור חתיכת Heartbeat, והוא בגודל משתנה ושקוף במבנה שלו, והוא בנוי בצורה שרק השולח יבין את התוכן:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 4    | Chunk  Flags  |      Heartbeat Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/            Heartbeat Information TLV (Variable-Length)        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד יכיל את הערך 4.
- Chunk Flags - שמונה ביט. יכיל את הערך 0 ויזכה להתעלמות על ידי המקבל.
- Heartbeat Length - שש עשרה ביט (מ-1 ועד 65,535). שדה זה מכיל את הגודל בבתים של חתיכת המידע של Heartbeat הכוללת גם את גודל התוכן של השדה Heartbeat Information.

- Heartbeat Information - גודל משתנה. זהו שדה בגודל משתנה, והוא שדה חובה, אשר מכיל בתוכו את המבנה הבא:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Heartbeat Info Type=1      |         HB Info Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                  Sender-Specific Heartbeat Info               /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Heartbeat Info Type - תמיד עם 1.
- HB Info Length - שש עשרה ביט (מ-1 ועד 65,535). הגודל של כל המידע של בקשת ה- Heartbeat Information.
- Sender-Specific Heartbeat Info - מידע משתנה שהשולח בלבד יודע כיצד לפרש אותו.
- לרוב המידע הזה יכיל מידע על זמן השולח כשהחתיכה נשלחה, וכן את כתובת היעד לשליחה.
- הבקשה הזו, בנויה על מנת להבטיח כי היעד קיים, וגם כי הנתיב תקין.
- כאשר מדובר בבדיקת תקינות של נתיב, המידע חייב להיות ערך רנדומאלי לגמרי בגודל של 64 ביט.

הדגמת wireshark לבקשת SCTP Heartbeat (מהקישור הבא):

```
Frame 3: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
Ethernet II, Src: ArtesynE_0c:06:96 (00:01:af:0c:06:96), Dst: TattileS_00:5e:46 (00:a0:80:00:5e:46)
Internet Protocol Version 4, Src: 10.28.6.42, Dst: 10.28.6.44
Stream Control Transmission Protocol, Src Port: 2905 (2905), Dst Port: 2905 (2905)
    Source port: 2905
    Destination port: 2905
    Verification tag: 0x00000e50
    [Assocation index: 1]
    Checksum: 0x53c3055f (not verified)
    HEARTBEAT chunk (Information: 20 bytes)
        Chunk type: HEARTBEAT (4)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
        Chunk length: 24
        Heartbeat info parameter (Information: 16 bytes)
            Parameter type: Heartbeat info (0x0001)
                0... .... .... .... = Bit: Stop processing of chunk
                .0.. .... .... .... = Bit: Do not report
            Parameter length: 20
            Heartbeat information: 40e44b920a1c062c1b66af7e00000000
```

Heartbeat Acknowledgment (HEARTBEAT ACK)

יעד אשר מקבל בקשת HEARTBEAT ישלח את בקשת HEARTBEAT ACK ליוזם בקשת ה-HEARTBEAT. הבקשה תמיד תישלח ל-IP המקור של בקשת ה-HEARTBEAT שתשובה זו היא עבורו.

מבנה הבקשה דומה מאוד לבקשת HEARTBEAT:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 5    | Chunk  Flags  |    Heartbeat Ack Length       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/            Heartbeat Information TLV (Variable-Length)        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד יכיל את הערך 5.
- Chunk Flags - שמונה ביט. יכיל את הערך 0 ויזכה להתעלמות על ידי המקבל.
- Heartbeat Ack Length - שש עשרה ביט (מ-1 ועד 65,535). שדה זה מכיל את הגודל בבתים של חתיכת המידע של Heartbeat הכוללת גם את גודל התוכן של השדה Heartbeat Information.
- Heartbeat Information - גודל משתנה. זהו שדה בגודל משתנה, והוא שדה חובה, אשר מכיל בתוכו את המבנה הבא:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Heartbeat Info Type=1      |         HB Info Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                  Sender-Specific Heartbeat Info               /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Heartbeat Info Type - תמיד עם 1.
- HB Info Length - שש עשרה ביט (מ-1 ועד 65,535). הגודל של כל המידע של בקשת ה- Heartbeat Information.
- Sender-Specific Heartbeat Info - מידע משתנה שהשולח בלבד יודע כיצד לפרש אותו.
- לרוב המידע הזה יכיל מידע על זמן השולח כשהחתיכה נשלחה, וכן את כתובת היעד לשליחה.
- הבקשה הזו, בנויה על מנת להבטיח כי היעד קיים, וגם כי הנתיב תקין.
- כאשר מדובר בבדיקת תקינות של נתיב, המידע חייב להיות ערך רנדומאלי לגמרי בגודל של 64 ביט.

הדגמת wireshark לבקשת SCTP Heartbeat Ack (מהקישור הבא):

```
Frame 4: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
Ethernet II, Src: TattileS_00:5e:46 (00:a0:80:00:5e:46), Dst: ArtesynE_0c:06:96 (00:01:af:0c:06:96)
Internet Protocol Version 4, Src: 10.28.6.44, Dst: 10.28.6.42
Stream Control Transmission Protocol, Src Port: 2905 (2905), Dst Port: 2905 (2905)
    Source port: 2905
    Destination port: 2905
    Verification tag: 0x0d53e6fe
    [Assocation index: 1]
    Checksum: 0x8c8e0746 (not verified)
    HEARTBEAT_ACK chunk (Information: 20 bytes)
        Chunk type: HEARTBEAT_ACK (5)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
        Chunk length: 24
        Heartbeat info parameter (Information: 16 bytes)
            Parameter type: Heartbeat info (0x0001)
                0... .... .... .... = Bit: Stop processing of chunk
                .0.. .... .... .... = Bit: Do not report
            Parameter length: 20
            Heartbeat information: 40e44b920a1c062c1b66af7e00000000
```

Abort Association (ABORT)

שליחת ABORT מתבצעת על ידי צד שקיבל במקור את ה-INIT כאשר הוא רוצה לסגור את השיוך לבקשה. הסיבה לכך יכולה להימצא בתוך המידע שנשלח. כאשר הודעת ABORT נשלחת, אסור לה להכיל גם חלק של DATA, אך פעולות ניהול כדוגמת INIT, INIT ACK, ו-SHUTDOWN COMPLETE) יכולות להגיע ביחד עם ABORT, אך הן חיבות להיות מסודרות לפני התיכה שמחזיקה את ABORT, או שמקבל פעולת ה-ABORT יתעלם מהם.

אם מקבל בקשת ה-ABORT מקבל אותה עם שגיאה או ללא TCB, המקבל חייב להתעלם בצורה שקטה מהבקשה. בנוסף, למקבל בקשת ה-ABORT, אסור לעולם להגיב לבקשה שכזו על ידי שליחת בקשת ABORT משל עצמו.

מבנה בקשת ABORT:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 6    |Reserved     |T|           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                   zero or more Error Causes                   /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד יהיה 6.
- Chunk Flags - שמונה ביט. שבעה הביטים הראשונים שמורים וחייבים תמיד להיות בערך 0. הביט האחרון (T) יכיל את הערך 0 במידה והשולח מילא את שדה Verification Tag אשר השולח ציפה ממנו לבצע. אם ה-Verification Tag משתקף מ-T, אז הערך חייב תמיד להיות 1. כאשר מדובר ב"השתקפות", הכוונה היא שה-Verification Tag שהתקבל ונשלח הם זהים. ישנם מספר כללים בנושא ה-Verification Tag כאשר נשלח ABORT, אך לא אכסה אותם במאמר זה.
- Length - שש עשרה ביט (מ-1 ועד 65,535). שדה זה מחזיק בתוכו את גודל החתיכה כולה בבתים, כולל את שדה ה-Error Cause.
- Error Cause - שדה זה יכוסה תחת החלק של שדה ה-Error.

הדגמת wireshark לבקשת ABORT SCTP (מהקישור הבא):

```
Frame 2: 50 bytes on wire (400 bits), 50 bytes captured (400 bits)
Ethernet II, Src: Vmware_5b:32:ba (00:0c:29:5b:32:ba), Dst: Vmware_69:f8:e7 (00:0c:29:69:f8:e7)
Internet Protocol Version 4, Src: 192.168.10.104, Dst: 192.168.10.105
Stream Control Transmission Protocol, Src Port: 12345 (12345), Dst Port: 12345 (12345)
    Source port: 12345
    Destination port: 12345
    Verification tag: 0xbe65a7ef
    [Assocation index: 7]
    Checksum: 0x7dc1b2fc (not verified)
    ABORT chunk
        Chunk type: ABORT (6)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
            .... ...0 = T-Bit: Tag not reflected
        Chunk length: 4
```

Shutdown Association (SHUTDOWN)

יעד שהוא תחת שיוך לבקשה, חייב להשתמש ב-SHUTDOWN על מנת לבצע סיום תקין של שיוך עם השרת.

מבנה בקשת SHUTDOWN הוא כזה:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 7    | Chunk  Flags  |      Length = 8               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Cumulative TSN Ack                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד יהיה בערך 7.
- Chunk Flags - שמונה ביט. תמיד יהיה 0.
- Length - שש עשרה ביט, תמיד יהיה 8. מציין את האורך של כל החתיכה.
- Cumulative TSN Ack - שלושים ושניים ביט (מ-1 ועד 4,294,967,295). פרמטר זה מייצג את מספר TSN האחרון שהתקבל על ידי שולח בקשת ה-SHUTDOWN לפני איזשהו פער.

חשוב לציין כי פעולת SHUTDOWN אינה מכילה איזשהו Gap Ack Block, ולכן אין היא יכולה לספק אישור על סדר קבלת TSN.

כאשר בפעולת SACK יש מחסור ב-Gap Ack Blocks, אשר נכללו מקודם, הדבר מציין כי המקבל חזר בו בנושא של שיוך חתיכות DATA. אך היות ואין לפעולת SHUTDOWN את שדות ה-Gap Ack Block, המקבל של בקשת ה-SHUTDOWN אינו מורשה להפריע למצב בו יש מחסור ב-Gap Ack Block כטווח.

הדגמת wireshark לבקשת SCTP SHUTDOWN (מהקישור הבא):

```
Frame 30: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
Ethernet II, Src: Vmware_5b:32:ba (00:0c:29:5b:32:ba), Dst: Vmware_69:f8:e7 (00:0c:29:69:f8:e7)
Internet Protocol Version 4, Src: 192.168.10.104, Dst: 192.168.10.105
Stream Control Transmission Protocol, Src Port: 12345 (12345), Dst Port: 12345 (12345)
    Source port: 12345
    Destination port: 12345
    Verification tag: 0x88423abf
    [Assocation index: 9]
    Checksum: 0x67754ba2 (not verified)
    SHUTDOWN chunk (Cumulative TSN ack: 3096428048)
        Chunk type: SHUTDOWN (7)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
        Chunk length: 8
        Cumulative TSN Ack: 3096428048
```

Shutdown Acknowledgment (SHUTDOWN ACK)

שליחת חתיכה זו, חייבת להגיע על מנת לאשר הגעה של פעולת SHUTDOWN, כאשר תהליך הכיבוי הסתיים. החתיכה של SHUTDOWN ACK אינה מחזיקה בשום פרמטרים:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 8    |Chunk  Flags   |      Length = 4               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד יהיה בערך 8.
- Chunk Flags - שמונה ביט. תמיד יהיה 0.
- Length - שש עשרה ביט, תמיד יהיה 4. מציין את האורך של כל החתיכה.

הדגמת wireshark לבקשת SCTP SHUTDOWN ACK (מהקישור הבא):

```
Frame 31: 50 bytes on wire (400 bits), 50 bytes captured (400 bits)
Ethernet II, Src: Vmware_5b:32:ba (00:0c:29:5b:32:ba), Dst: Vmware_69:f8:e7 (00:0c:29:69:f8:e7)
Internet Protocol Version 4, Src: 192.168.10.104, Dst: 192.168.10.105
Stream Control Transmission Protocol, Src Port: 12345 (12345), Dst Port: 12345 (12345)
    Source port: 12345
    Destination port: 12345
    Verification tag: 0x88423abf
    [Assocation index: 9]
    Checksum: 0xbcf42fee (not verified)
    SHUTDOWN_ACK chunk
        Chunk type: SHUTDOWN_ACK (8)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
        Chunk length: 4
```

Operation Error (ERROR)

כאשר נשלחת בקשת ERROR לרוב לשרת, הדבר מגיע בשביל להציג כי יש בעיה במידע שהתקבל. חתיכה של ERROR יכולה להכיל בתוכה מספר שגיאות שונות. Operation Error אינו נחשב לשגיאה שלא ניתן להמשיך ממנה והלאה, אך היא במידה והיא בשימוש תחת חתיכה של ABORT, אז היא תחשב לכזו.

כל סוג שגיאה מכיל מבנה שונה אשר יכנס למבנה הכללי הבא:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 9    | Chunk  Flags  |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                   one or more Error Causes                    /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Type - תמיד יהיה 0.
- Chunk Flags - שמונה ביט. תמיד יהיה 0 ותהיה התעלמות ממנו.
- Length - שש עשרה ביט. מכיל את הגודל בבתים של כל חתיכת ה-ERROR כולל את כל שדות ה-Error Causes.

Error Causes

סוגי השגיאות השונים מחולקים לתתי נושאים, אשר מכילים מבנה שונה בהתאם לצורך:

קוד שגיאה	תאור
1	Invalid Stream Identifier
2	Missing Mandatory Parameter
3	Stale Cookie Error
4	Out of Resource
5	Uresolvable Address
6	Unrecognized Chunk Type
7	Invalid Mandatory Parameter
8	Unrecognized Parameters
9	No User Data
10	Cookie Received While Shutting Down
11	Restart of an Association with New Address
12	User Initiated Abort
13	Protocol Violation

מבנה הדיווח הוא כזה:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Cause Code          |          Cause Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                    Cause-Specific Information                 /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Cause Code - שש עשרה ביט. יהיה אחד מהערכים בטבלה למעלה.
- Cause Length - שש עשרה ביט. יכיל את הגודל בבתים של כל מבנה ה-Error Cause, הכולל בתוכו גם את השדות המגיעים עם Cause-Specific Information.
- Cause-Specific Information - גודל משתנה. מכיל בתוכו מבנה שדות המתאימים לסוג השגיאה.

Invalid Stream Identifier - שגיאה זו מייצגת מצב בו מתקבל מידע על stream אשר אינו באמת קיים:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Cause Code=1              |       Cause Length=8          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Stream Identifier      |         (Reserved)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- <u>Stream Identifier</u> - שש עשרה ביט. מכיל בתוכו את ה-Stream Identifier אשר התקבל עבור חתיכת ה-DATA.
- <u>Reserevd</u> - שש עשרה ביט. השדה תמיד יחזיק בערך 0, ויזכה להתעלמות על ידי המקבל.

Missing Mandatory Parameter - שגיאה זו מייצגת מצב בו שדה אחד או יותר המוגדרים כשדות חובה אינם נשלחו במצב של INIT או INIT ACK.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Cause Code=2              |      Cause Length=8+N*2       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Number of missing params=N                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Missing Param Type #1       |   Missing Param Type #2       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Missing Param Type #N-1     |   Missing Param Type #N       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- <u>Number of Missing Parameters</u> - שלושים ושניים ביט. השדה מכיל את מספר הפרמטרים הנמצאים תחת ההגדרה של Cause Specific Information.
- <u>Missing Param Type</u> - שש עשרה ביט. כל שדה יכיל את מספר הפרמטר שחסר (לפי סדר המבנה).

State Cookie Error - שגיאה זו מייצגת מצב בו מקבל State Cookie קיבל עוגייה תקינה, אך שפג התוקף לשימוש בה.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Cause Code=3              |        Cause Length=8         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Measure of Staleness (usec.)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- <u>Measure of Staleness</u> - שלושים ושניים ביט. שדה זה מכיל את ההבדל במילי-שניות בין הזמן הנוכחי לבין פרק הזמן שהעוגיה פגה תוקף. שולח הבקשה אינו חייב לשלוח את פרק הזמן, ובמידה והוא בוחר שלא לעשות כן, הערך יהיה 0.

Out of Resource - שגיאה זו מייצגת מצב בו אין לשולח השגיאה מספיק משאבים לטפל בבקשות. שגיאה זו לרוב תגיע בשילוב של ABORT.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Cause Code=4             |       Cause Length=4          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

Unresolvable Address - שגיאה זו מייצגת מצב בו שולח השגיאה אינו מצליח לבצע פעולת resolving או להגיע ליעד של הכתובת שנשלחה אליו, או שהוא אינו תומך בסוג הכתובת שהגיעה אליו. לרוב הודעת שגיאה זו תגיע בשילוב של ABORT.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Cause Code=5             |       Cause Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                  Unresolvable Address                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- <u>Unresolvable Address</u> - גודל משתנה. מכיל את מבנה סוג הכתובת (כפי שהוגדר בINIT) אשר יש איתו בעיה.

Unrecignized Chunk Type

שגיאה זו מייצגת מצב בו שולח ההודעה אינו מבין את ההודעה שקיבל חזרה לשולח המקורי. בנוסף לכך הביטים של Chunk Type מוגדרים כ-01 או 11.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Cause Code=6             |       Cause Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                  Unrecognized Chunk                           /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- <u>Unrecognized Chunk</u> - אורך משתנה. השדה מחזיק בחתיכה אשר אינה מזוהה הכוללת את Chunk Type, Chunk Flags ו-Chunk Length.

Invalid Mandatory Parameter - שגיאה זו מייצגת מצב בו שולח בקשת INIT ACK לא שלח את אחד משדות החובה.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Cause Code=7             |       Cause Length=4          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

Unrecognized Parameters - שגיאה זו מייצגת מצב בו שולח בקשת INIT ACK שלח שדה או מספר שדות לא מוכרים על ידי המקבל תחת Optional TLV.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Cause Code=8              |      Cause Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                  Unrecognized Parameters                      /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Unrecognized Parameters - גודל משתנה. שדה זה מכיל רשימה של העתק חלק ה-INIT ACK בצורה שלמה, כולל Optional TLB. הודעה זו לרוב תהיה כחלק מסוג הודעת השגיאה המגיעה עם COOKIE ECHO, כאשר מגיבים ל-INIT ACK, וכאשר שולח בקשת ה-COOKIE ECHO מעוניין לדווח על הבעיה.

No User Data - הודעת שגיאה זו מייצגת מצב בו שולח חתיכת DATA ללא התוכן עצמו.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Cause Code=9              |      Cause Length=8           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                  TSN value                                    /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- TSN Value - שלושים ושניים ביט. שדה זה מכיל את תוכן ה-TSN של חתיכת ה-DATA אשר התקבל ללא תוכן.
  לרוב הודעה זו תגיע בצירוף של פעולת ABORT.

Cooise Recived While Shutting Down - שגיאה זו תגיע כאשר התחיל תהליך הכיבוי, ובקשת COOKIE ECHO התקבלה תוך כדי מצב שליחת SHUTDOWN ACK-SENT. הודעת שגיאה זו לרוב תגיע בתוך בקשת SHUTDOWN ACK.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Cause Code=10             |      Cause Length=4           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

Restart of an Association with New Address - שגיאה זו תגיע כאשר התקבלה בקשת INIT לשיוך שכבר קיים. פעולת הINIT הוסיפה כתובת לשיוך אשר מקודם לא היה קיים כחלק מפעולת השיוך. כחלק מהודעת השגיאה, תצורף רשימה של הכתובות המדוברות.

שגיאה זו תגיע לרוב כחלק מפעולת ABORT וסירוב לקבל את פעולת ה-INIT.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Cause Code=11         |      Cause Length=Variable    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                       New Address TLVs                        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

User Initiated Abort - שגיאה זו יכולה להכיל פעולת ABORT בתוכה, ונשלחת לבקשת ULP. חלק ה-ULP יכול לציין צורך בביצוע ABORT אשר תהיה חלק מתשדורת SCTP ויגיע ל-ULP בצד השני.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Cause Code=12         |      Cause Length=Variable    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                    Upper Layer Abort Reason                   /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

Protocol Violation - שגיאה זו יכולה להיות בתוך פעולת ABORT שנשלחת בגלל שבקשת ה-SCTP זוהתה על ידי המקבל כבקשה לא תקנית, ואין שום שגיאה אחרת אשר מתאימה להציג את המקור שלה. מימוש של הודעת שגיאה זו יכול להכיל גם מידע בנושא.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Cause Code=13         |      Cause Length=Variable    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                    Additional Information                     /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

Cookie Echo (COOKIE ECHO) - החתיכה של COOKIE ECHO נמצאת בשימוש רק בזמן אתחול השיוך. העוגייה נשלחת על מנת להשלים את השיוך בחלק האתחול. חתיכה זו חייבת להקדים כל חתיכה של DATA אשר נשלחת לשיוך. אך במידה ורוצים לשלוח כבר מידע, ניתן להכניס אותה כבקשה ראשונה מעל ה-DATA ולא לעמוד בפני עצמה.

COOKIE ECHO אינו מכיל בתוכו את השדה State Cookie. במקום, המידע בתוך State Cookie הופך להיות המידע של העוגייה. פעולה זו מאפשרת למימוש לשנות רק את שני הבתים הראשונים של State Cookie ולהפוך להיות COOKIE ECHO.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 10   |Chunk  Flags   |         Length                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                     Cookie                                    /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Chunk Flags - שמונה ביט. תמיד יהיה 0.
- Length - שש עשרה ביט. השדה מחזיק בגודל (בבתים) של כל חתיכת ה-COOKIE ECHO כולל שדה ה-COOKIE.
- Cookie - גודל משתנה. השדה חייב להכיל את העוגייה המדויקת שהתקבלה על ידי שדה ה-Cookie ב-INIT ACK.

מומלץ למימוש להחזיק בעוגייה הקטנה ביותר שרק ניתן, בשביל להבטיח תקשורת בריאה בין הצדדים.

הדגמת wireshark לבקשת COOKIE ECHO SCTP (מהקישור הבא):

```
Frame 21: 278 bytes on wire (2224 bits), 278 bytes captured (2224 bits)
Ethernet II, Src: Vmware_69:f8:e7 (00:0c:29:69:f8:e7), Dst: Vmware_5b:32:ba (00:0c:29:5b:32:ba)
Internet Protocol Version 4, Src: 192.168.10.105, Dst: 192.168.10.104
Stream Control Transmission Protocol, Src Port: 12345 (12345), Dst Port: 12345 (12345)
    Source port: 12345
    Destination port: 12345
    Verification tag: 0x6ad4dac4
    [Assocation index: 9]
    Checksum: 0x92c8fabf (not verified)
    COOKIE_ECHO chunk (Cookie length: 228 bytes)
        Chunk type: COOKIE_ECHO (10)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
        Chunk length: 232
        Cookie: 000000000000000000000000000000000000000000000000...
```

Cookie Acknowledgment (COOKIE ACK) - COOKIE ACK נמצאת בשימוש רק בזמן אתחול השיוך. החתיכה נמצאת בשימוש כתשובה ל-COOKIE ECHO. חתיכה זו חייבת להגיע לפני פעולת DATA או SACK כתוצאה משיוך, אך יכולה להגיע כחלק מאוסף הנתונים, כאשר פעולה זו קודמת לשאר הפעולות הנוספות באותה פקטת SCTP.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 11   |Chunk  Flags   |     Length = 4                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Chunk Flags - שמונה ביט. חייב תמיד להיות על 0.

הדגמת wireshark לבקשת COOKIE ACK SCTP (מהקישור הבא):

```
Frame 24: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
Ethernet II, Src: Vmware_69:f8:e7 (00:0c:29:69:f8:e7), Dst: Vmware_5b:32:ba (00:0c:29:5b:32:ba)
Internet Protocol Version 4, Src: 192.168.10.105, Dst: 192.168.10.104
Stream Control Transmission Protocol, Src Port: 12345 (12345), Dst Port: 12345 (12345)
    Source port: 12345
    Destination port: 12345
    Verification tag: 0x6ad4dac4
    [Assocation index: 9]
    Checksum: 0x6ca21c00 (not verified)
    COOKIE_ACK chunk
        Chunk type: COOKIE_ACK (11)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
        Chunk length: 4
```

Shutdown Complete (SHUTDOWN COMPLETE) - פעולת SHUTDOWN COMPLETE היא החייבת להגיע בשביל להציג אישור על כך שהמקבל של פעולת SHUTDOWN ACK מבצע בעצמו פעולת כיבוי.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 14   |Reserved     |T|      Length = 4               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[מתוך RFC 4960]

- Chunk Flags - שמונה ביט. שבעה ביט ראשונים שמורים וחייבים להיות עם הערך 0. ביט ה-T יכול להיות 0 או 1. במידה והוא בערך 0, אז אומר כי השולח מילא את שדה ה- Verification Tag כמצופה על ידי יעד השליחה. אם ה-Verification Tag משתקף מ-T, אז הערך חייב תמיד להיות 1. כאשר מדובר ב"השתקפות", הכוונה היא שה-Verification Tag שהתקבל ונשלח הם זהים. ישנם חוקים מוגדרים בנושא של Verification Tag בנושא, אך הם לא מכוסים במאמר זה.

הדגמת wireshark לבקשת SHUTDOWN COMPLETE SCTP (מהקישור הבא):

```
Frame 33: 50 bytes on wire (400 bits), 50 bytes captured (400 bits)
Ethernet II, Src: Vmware_5b:32:ba (00:0c:29:5b:32:ba), Dst: Vmware_69:f8:e7 (00:0c:29:69:f8:e7)
Internet Protocol Version 4, Src: 192.168.10.104, Dst: 192.168.10.105
Stream Control Transmission Protocol, Src Port: 12345 (12345), Dst Port: 12345 (12345)
    Source port: 12345
    Destination port: 12345
    Verification tag: 0x88423abf
    [Assocation index: 9]
    Checksum: 0xcee66a2a (not verified)
    SHUTDOWN_COMPLETE chunk
        Chunk type: SHUTDOWN_COMPLETE (14)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
            .... ...0 = T-Bit: Tag not reflected
        Chunk length: 4
```

סיכום

בחלק הקודם הסברתי על הרעיון מאחורי SCTP ומה בעצם הפרוטוקול מנסה לפתור. בחלק הזה הדגמתי כיצד התקשורת הזו תתבצע בפועל. מה שנשאר הוא למעשה מאמר מאוד ארוך או סדרה של מאמרים אשר מסבירים כיצד המימוש מתקיים בפועל.

בנוסף, להציג הרבה מאוד מקרים שונים, והתנהגויות שונות אשר ניתן להיתכל בהם בשימוש בפרוטוקול. כאשר מבינים פרוטוקול לעומק, קל ופשוט יותר לקרוא אותו להתמודד איתו, ואפילו למצוא בעיות אבטחה במימושים השונים.

אני מקווה כי המאמר הצליח לספק ולו במעט סקרנות בנושא, גם אם אינכם מגיעים מעולם הטלפוניה, ורואים את הפוטנציאל הגלום בפרוטוקול שכזה.

קריאה נוספת

מרבית המידע שנקלח בחלק זה של המאמר נלקח מ-RFC 4960, שהיא הגרסה העדכנית לתקן הכללי של SCTP נכון לכתיבת מאמר זה. אך מאז 2007, אשר התקן יצא, יצאו לו מספר תוספות:

- Stream Control Transmission Protocol (SCTP) Chunk Flags Registration (RFC 6096)
- IANA Procedures for the Management of the Service Name and Transport Port Number Registry (RFC 6335)
- SACK Immediately Extension for the Stream Control Transmission Protocol (RFC 7053)
- Authenticated Chunks For the Stream Control Transmission Protocol (SCTP) (RFC 4895)
- Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP) (RFC 4820)
- SCTP Parameters
- WebRTC Data Channels - draft 13

- במידה ואתם מעוניינים לראות עוד פעולות של Wireshark עם STCP (או בכלל עם פרוטוקלים שונים), אני ממליץ על הקישור הבא: https://wiki.wireshark.org/SampleCaptures.
- מאמר של גוגל על SPDY, אשר כיום מוכר כ-HTTP/2.0 ומדוע SCTP לא נבחר להיות חלק מ-SPDY. TL;DR - הפחד מלשנות את הפרוטוקולים שכבר קיימים במערכת.
- מאמר של צחי לוי על הבחירה של SCTP עבור WebRTC. הסבר עמוק יותר למהות השימוש במערכת תחת Html5 Rocks.

דברי סיכום

בזאת אנחנו סוגרים את הגליון ה-68 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 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 ינואר.

אפיק קסטיאל,

ניר אדר,

31.12.2015