SHAMOON COLLEGE OF ENGINEERING

קורלציה בין משתמשי קצה ברשת Tor

פרויקט גמר בהנדסת תוכנה

הוכן לשם השלמת הדרישות לקבלת תואר ראשון .B.Sc בהנדסת תוכנה

מאת: לאוניד מלמוד וכפיר בן-זקן

מנחה: ד״ר מיכאל אורלוב

הוגש למחלקה להנדסת תוכנה

המכללה האקדמית להנדסה ע״ש סמי שמעון

אשדוד

אוקטובר 2014

תשע״ד

קורלציה בין משתמשי קצה ברשת Tor

מס׳ פרוייקט: _____

הוגש למחלקה להנדסת תוכנה

המכללה האקדמית להנדסה ע״ש סמי שמעון

מאת:

לאוניד מלמוד

כפיר בן-זקן

אישור המנחה: _____

אישור ראש המחלקה: _____

תאריך לועזי	תאריך עברי
אוקטובר 2014	תשרי תשע״ה

אשדוד

הבעת תודה

כאמור, אחד הדברים העיקריים אותם למדנו בפרויקט הוא כיצד להתמודד עם בעיות.
ברצוננו להודות לכל האנשים אשר סייעו לנו לאורך הדרך ותרמו לנו מהידע ומהניסיון שלהם.
ראשית נודה למנחה שלנו מיכאל אורלוב . מיכאל תרם לנו ידע רב אודות מערכות מתוחכמות אלו אשר דורשות ידע רב באבטחת מידע וסייבר, נתן לנו ביטחון לבצע את המטלות בצורה הטובה ביותר ותמיד עודד אותנו בזאת שהפרויקט פותח לנו דלתות רבות בתעשייה .
תודה לשלמה מארק שתמך בנו מתחילת הפרויקט ושימש אוזן קשבת לכל הבעיות גם אלו שמעבר לחומר הלימודי.

העבודה נעשתה בהדרכת:

ד״ר מיכאל אורלוב

במחלקה להנדסת תוכנה

המכללה האקדמית להנדסה סמי שמעון

תוכן עניינים

מבוא

רשת Tor הינה תוכנה חופשית בעלת קוד פתוח המאפשרת למשתמשי אינטרנט לגלוש בצורה אנונימית.
התוכנה מעבירה את תעבורת הרשת דרך רשת חופשית של מתנדבים מרחבי העולם המכונים ממסרים (Relays) או צמתים (nodes), כך שכל משתמש שמתקין את התוכנה יכול להוות צומת ברשת.
Tor נועדה להגן על התכתבות של עיתונאים, פעילים פוליטיים בארה״ב, אירופה וגם בסין, איראן וסוריה.
לשם כך השירות מקבל 60% מהמימון שלו מממשלת ארה״ב, בעיקר ממשרד החוץ וממשרד הגנה, שאחראי גם על ה NSA.

למרות חשיבותו של Tor לפעילי זכויות אדם ומתנגדי משטר, ה NSA-והארגון הבריטי המקביל ה GCHQ הקדישו מאמצים ניכרים בתקיפה של השירות, שסוכנויות אכיפת חוק אומרות שהוא משמש גם גורמי טרור, סוחרים בתמונות של פורנוגרפיה פדופילית, סחר בסמים ובנשק.
ארגוני זכויות אדם היו מודאגים כי סוכנויות המודיעין הצליחו לפצח את ההגנות של תור בעקבות הגילויים בגרדיאן, בניו יורק טיימס וב ProPublica על המאמצים הנרחבים של ה- NSA לפצח תוכנות להגנה על פרטיות ואבטחה. בדיווח של הגלובו בברזיל נרמז שהסוכנויות מחזיקות ביכולות מסוימות נגד הרשת.
בעוד שנראה שNSA לא הצליחה לפגוע בליבת האבטחה של תוכנת תור או ברשת, המסמכים חושפים התקפות שמוכיחות היתכנות ,(proof-of-concept) כולל כמה שמתבססות על מערכות הריגול הנרחבות של ה NSA והGCHQ ש״יושבות״ על תעבורת האינטרנט.
לפי פרסומים הNSA מפעילה בחשאי כמה צמתים בתוך רשת תור, אולם המסמכים לא חושפים בכמה צמתים מדובר או אם ההתקפה המדוברת נבחנה בעולם האמיתי.
ניסיונות נוספים כוללים הפניית תעבורה לשרתים המופעלים באמצעות ה NSA או תקיפה של תוכנות אחרות המותקנות על מחשביהם של משתמשי תור, בשיטה שזכתה לשם הקוד Egotistical Giraffe.
מסמכים סודיים של הNSA שחשף אדוארד סנודן, מגלים כי הסוכנות לא הצליחה לפצח לגמרי את ההגנות של תור, ובמקום זה היא נשענת על פרצות בתוכנות אחרות במחשביהם של משתמשים. שיטה אחת שפיתחה הסוכנות כרוכה בתקיפה של דפדפן הפיירפוקס שמשמש תשתית לתור.
כך הסוכנות מצליחה למעשה להשתלט כליל על מחשב המטרה, לקבל גישה לקבצים שלו ולעקוב אחרי כל הקשות המקלדת ופעילותו ברשת (באמצעות תוכנה שמכונה Keylogger)
שימושים מוכרים ברשת Tor כגון אתר דרך המשי (Silk Road) הוא אתר מכירת הסמים הגדול ביותר באינטרנט, אשר פעל בDark Net המתנהלת על גבי שרתים שאינם מוכרים לגוגל או לשירותי אינטרנט אחרים.
דרך המשי, כמו אתרים נוספים פועלים על גבי רשת Tor שמצריכה שימוש בדפדפן Tor כדי לגשת אליהם לדברי הFBI היקף הפעילות של האתר עצום. ביולי 2013, היו רשומים יותר מ-950 אלף משתמשים באתר, 30% מהם מארה״ב.

בין פברואר 2011 ליולי 2013 בוצעו יותר מ-1.2 מיליון עסקאות שבהן היו מעורבים כמעט 147 אלף חשבונות של רוכשים ו-3,877 חשבונות של מוכרים. התשלום נעשה באמצעות המטבע הדיגיטלי ביטקוינט, וסך התשלומים עמד על 9,519,664 ביטקוינים.

לדברי הFBI, סוכניו ביצעו מעל 100 רכישות של סמים ובהם אקסטזי, קוקאין, הרואין ואחרים.

במידה והממשל האמריקני היו מצליחים לנטר את ההודעות ברשת או מפסיק להבין מי משתמש הקצה שמבצע עסקאות באתר היו יכולים להגיע ישירות למפעיל האתר ולמשתמשי הקצה הרוכשים, אך מכיוון שלפי פרסומים עדיין לא נמצא הדרך לניטור חבילות ברשת Tor הממשל האמריקני נאלץ לבצע עסקאות סמים מפוברקות בכדי להגיע למשתמשי הקצה.

וזה רק קצה הקרחון, בנוסף ישנם אתרים בכל התחומים הפליליים המוכרים החל מסחר בבני אדם ועד סחר בנשק, הזמנות רצח, פורומים המופעלים על ידי פדופילים ועוד.

המערכת משתמשת בטכניקה המכונה "Onion Routing" (ניתוב בצל) בה המידע המקורי מוצפן מספר פעמים ונשלח בנתיב המורכב מתחנות ממסר שנבחרו באופן רנדומאלי אך ישנה בדיקה באלגוריתם של רשת Tor הבודק שכל תחנת ממסר תהיה ממוקמת במדינה שונה.

בכל תחנת ממסר מפוענחת שכבה אחת של הצפנה ובכך נחשפת תחנת הממסר הבאה עד שהחבילה מגיעה לתחנה האחרונה המפענחת את השכבה האחרונה של הצפנה ושולחת אותו ליעד הסופי ללא דרך לדעת מיהו השולח.

תהליך זה מאפשר למנוע התערבות בתוכן לאורך הדרך ואף להסתיר את מקור המידע/הגולש.

המערכת עצמה משמשת מגוון עצום של ארגונים ומשתמשים פרטיים הזקוקים להגנה ופרטיות.

מסיבה זאת אין רישום מסודר של משתמשים אך אנו יכולים להסיק על השימוש בה מפעילות של משטרים כנגדה.

דוגמה טובה לכך היא חסימת השרתים הפומביים של Tor על ידי סין ואיראן וניסיונות לחסום את השירות על ידי סוריה, טוניסיה ועוד.

מנגד, ניתן גם לראות מסמכים המעודדים גורמים כמו שליחים דיפלומטים, אזרחים סורים, פעילי זכויות אדם וחושפי שחיתויות להשתמש במערכת בכדי להגן על התקשורת והפרטיות שלהם מפני גורמים עוינים.

ככל שרשת Tor הולכת וגדלה ממשתמשים הפעילים בכל רחבי העולם , הסבירות הולכת ופוחתת למערכת אוטונומית אחת (AS) בכדי להיות מסוגלת לראות את שתי הקצוות לחיבור אנונימי.

ניתוח רשת טור הוא אפשרי ואף יכול לסכן משתמשים רבים ברשת Tor כאשר יריב בעל אמצעיים וידע מקיף בביצוע אנליזה של הרשת ,וגם בעל כלים לביצוע קורלציה עם המידע שצבר מאותם צמתים (Entry node, Guard node, Exit node). ידוע גם כי השימוש בביטורנט הוא לא בטוח במיוחד, ואנו מראים כי יציאות (Exit Node) בעל תוחלת חיים ארוכה יכולה להוות בעיות אבטחה גדולות עבור הצרכים של המשתמשים וגם של הביצועים שלהם.

נכון להיום לפי פרסומים, עדיין אף גוף או אדם לא הצליח לבצע קורלציה בהסתברות גבוהה ברשת Tor, וכאשר אפשרות זאת תהיה ברת ביצוע ,רשת Tor תצטרך לבצע התאמות גדולות ולדעתנו המקצועית תהיה בלתי אפשרית ,בגלל שהקורלציה לא מבצעת בתוך הרשת האנונימית אלה בקצוות הקשר בין המשתמשים

ושם אין כל השפעה של TOR,וכמו שציינו לפני הרשת תפסיק להיות רלוונטית, ויצטרכו למצוא חלופות לביצוע התקשרות אנונימית בין משתמשים.

עם האנונימיות בגלישה, בא גם הצורך באנונימיות באתרים עצמם ובא צורך באתרים שלא יהיו נגישים לציבור הרחב.

רשת ה TOR מאפשרת גישה לשירותים מוחבאים שהם בעצם אתרים שהגישה אליהם מתאפשרת באמצעות פרוטוקולים מסוימים שרק TOR עובדת איתם, באמצעות הקשת כתובת מיוחדת המסתיימת ב Onion. סיומת onion אינה סיומת 'אמתית'. לא מדובר בשרת DNS ואי אפשר לגשת לכתובות האלה מהדפדפן שאתם גולשים בו עכשיו (במידה וזה לא TOR כמובן). מדובר בכינוי שניתן ליעד מסוים בשרת שרק ה TOR יודעת לפענח.

TOR במהותה, לא שומרת שום מידע שעובר דרכה ובכך בעצם לא יודעת מי מבקש לגשת לאן אלא פשוט עושה את זה. בנוסף כתובת הIP של השרת לא נחשפת במהלך הגלישה של המשתמש אליו בשימוש בסיומת Onion.

TOR מאפשר לקבוע מדיניות עבודה לכל Onion router שתגדיר מה כמות המידע שתעבור דרכו, רוחב הפס שהוא מאפשר, הגבלות ברמת כתובות IP וכן הגבלות ברמת פורטים, כל למשל אם נגדיר מדיניות שבה אנו מאפשרים את כל כתובות ה IP בכל הפורטים, ה Onion router שלנו ישמש את כל התפקידים האפשריים , Entering node; Relay node; Exit nod לחילופין אם נגדיר מדיניות שלא מאפשרת אף IP באף פורט ה Onion router שלנו יוכל לשמש רק כ Entering node; Relay node ולא ישמש כ Exit node אשר מעביר את התעבורה ליעד הסופי.

כל המידע שמופיע ב Description חתום בעזרת המפתח הפרטי של אותו ה Onion router חשוב להבין שכל התקשורת בין ה Onion routers לבין עצמם ובינם לבין ה TOR directory מבוססת גם היא על TLS.

מטרת הפרויקט

מטרת הפרויקט היא להגיע לשבירת אנונימיות של התעבורה ברשת Tor תוך כדי השתלת צמתים אמתיים ברשת בעלי Sniffer אשר ינתר את כל התעבורה שעוברת בצומת שעליה הותקן.
בנוסף לניתור ההודעות ישלחו הנתונים ישירות לשרת מנתח אשר מקבץ את כל הנתונים שמקבל מכל הצמתים ובעזרת אלגוריתם של סטטיסטיקה נמצא את משתמשי הקצה.
למטרה זאת נגיע בעזרת תוכנת ריגול (Sniffer) שנתקין על השרתים ומשתמשים, שנועדה לזהות במדויק את זמני כניסת החבילות וזמני יציאת החבלות וגודל החבילות, מכיוון שבכל רגע נתון נכנסים ויוצאים מהשרת אלפי חבילות, אנו נעביר את החבילות סינון לפי זמני כניסה וזמני יציאה מדויקים במילי שניות, ובעזרת זה נוכל להעריך בהסתברות גבוהה מי משתמש הקצה.

כיום לא ניתן לגלות מי מתקשר עם מי ברשת ,התוכנה Tor יכולה להיות שימושית למי שרוצה לשמור על פרטיות מכל גורם ממשלתי או פלילי ,אבל כאשר זה מגיע למצב של פגיעה בחיי אדם כגון טרור ,סחר בסמים ,עבריינים וכו' ,ולכן במקרים קיצוניים יש צורך לפגוע בפרטיות האדם לשם הצלת חיי אדם, כאן אנו יוצרים כלי שיוכל לזהות את האנשים האלו , אך כלי כזה גם יעשה רעש גדול בעולם אבטחת תוכנה כי הוא יצליח לעקוף את האבטחה שעד היום במשך כמה שנים לא הצליחו לפרוץ,
לדעתנו גם התוכנה(Tor) תפסיק להיות רלוונטית אם הכלי יהיה מופץ ברשת.

מטרת העל היא ליישם את המערכת על רשת Tor תוך כדי השתלת צמתים אמיתיים שישתמשו במערכת שפותחה על ידינו.

קורלציה ברשת Tor

לאחר סקירה מקיפה של הרשת הגענו למסקנה שהדרך היחידה לזהות התקשרות היא לנטר את החבילות שזורמות ברשת דרך שרתים ולזהות קורלציה בין משתמשי קצה בעזרת זמני ההשהיה בין החבילות העוברות ברשת.

רשת Tor בנויה על ניתוב בצל ומאוד פגיעה ליריב שיכול לפקח על התנועה של משתמש כפי שהוא נכנס ויוצא מרשת האנונימיות. ע״י ביצוע קורלציה בעזרת מערכת לניתוח חבילות העוקבת אחר ההתקשרות של השולח והמקבל .

Murdoch and Danezis הראו שאפשר לעשות התקפות מתואמות תנועה ביעילות לא גבוהה נגד רשת Tor.

ההשהיה הנמוכה של רשת Tor הופכת אותה מתאימה מאוד לדפדפנים ומשימות כגון גלישה באינטרנט , אבל לא בטוח נגד ניתוח תעבורה, התקפות של יריב פסיבי גלובלי.

הם הציגו טכניקות חדשות המאפשרות ליריבים עם תצוגה חלקית בלבד של הרשת להסיק דרך אילו צמתים נעשה שימוש ברגע נתון, ובעזרת מידע זה הצליחו לצמצם במידה ניכרת את האנונימיות שמספקת רשת Tor.

שיטות שחוקרים השתמשו :

פאול סייברסון העריך את האבטחה של רשת Tor ביחס ליריבים ולמדדים שהגדיר מראש

זה דורש הערכות של ההסתברויות של אירועים ברשת. לשם כך, הם השתמשו בשיטת מונטה קרלו כדי לדגום כמה תנועה של משתמש זורמת על הרשת ומדדו פעילות של משתמשים שונים .

עבור כל דגימה, הם השתמשו במודל של רשת Tor, לדמות התנהגות של משתמש, ולדמות את פעולות תוכנת לקוח Tor וכתוצאה מכך הם העריכו את אנונימיות המשתמש ולפי דגימות אלה עשו ניסוי נגד יריבי רשת Tor.

הם בנו את ה-TorPS סימולציה לבחירת נתיב , שעושה שימוש בנתונים היסטוריים ברשת, כדי לשחזר את המצב לפיו לקוחות פעלו בעבר ולאחר מכן מבצע בחירת אלגוריתם נתיב על פי תנאי הפעולות של המשתמש.

מודלים ברשת TOR

- משתמש רגיל (typical) –מודל זה נועד לייצג שימוש Tor ממוצע. זה משתמש בארבע אפשריות שניתן לעקוב 1. ב-Gmail / גוגל צ׳אט 2. לוח השנה גוגל / מסמכים 3. בפייסבוק 4. פעילות חיפוש באינטרנט.
- IRC- מודל זה מייצג את השימוש ב Tor לצורך חוזר ונשנה , אך בלעדי של צ׳אט IRC.
- ביטורנט(BitTorrent)- מודל זה מייצג באמצעות ביטורנט על Tor . הוא מורכב מפעילות במהלך ההורדה של קובץ יחיד ממשתמשים רבים ,אם מגדירים בחירת פורט רנדומלי בתוכנה ייווצר מצב שביטורנט יעבור דרך פורטים רבים יותר .
- WorstPort- מודל זה משנה את המודל אופייני על ידי החלפת מספרי הפורטים ל 6523 , שהוא פורט ברירת מחדל של ״ gobby ״.

  פורט מספר 6523 נוסף לאחרונה לפורט בעל תוחלת חיים ארוכה במיוחד.

  חיבור לExit Node שנקבעו על ידי Tor כ-פורט בעל תוחלת חיים ארוך אשר דורש שימוש בממסרי Exit Node יציבים, שחייבת להיות ״חיה״(Uptime) הרבה זמן ברשת.
- BestPort- מודל זה משנה את המודל אופייני על ידי החלפת מספרי הפורטים ב HTTPS 443 , אשר נתמכת על ידי הכמות הגדולה ביותר של Exit Node.

מודל ביטורנט יוצר פי 2.5 זרמים מהמודל אופייני ופי 50 ממודל ה-IRC.

הביטורנט גם גורם עומס ברשת וזה משפיע על כלל המשתמשים .

הסיבה שציינו את המודלים הנ״ל היא שכאשר נרצה לבצע קורלציה אנחנו נצטרך לבצע אותם על צמתים שחיים יחסית הרבה זמן ועל משתמשים אשר יצרו חיבורים רבים ברשת Tor וניתנים לזיהוי ביתר קלות.

היריב צריך לקבוע את הדרך הטובה ביותר להקצאת רוחב הפס שלו כדי למקסם את הסיכוי של גילוי המידע העובר ברשת. מכיוון שלא ניתן לבחור את אותו הממסר פעמיים במעגל, הוא חייב לרוץ לפחות שני ממסרים על מנת לבצע התקפת קורלציה לפיכך, אנו מניחים שמטרות היריב הם הGUARD והיציאה.

אנו מניחים שהGUARD שהשתלנו נמצא מספיק זמן פעולה כדי להשיג את דגל השמירה. אנו מניחים שצומת היציאה המושתל אינו רשאי לקבל את דגל GUARD ואת מדיניות היציאה המאפשרת יציאה לכל הכתובות והפורטים.

השגת GUARD הוא הרבה יותר חשוב כדי למקסם את הסיכוי של גילוי המידע העובר ברשת של משתמש נתון, משתמשים משנים GUARD חדש הרבה פחות מאשר יציאות חדשות.

תיאור המערכת:

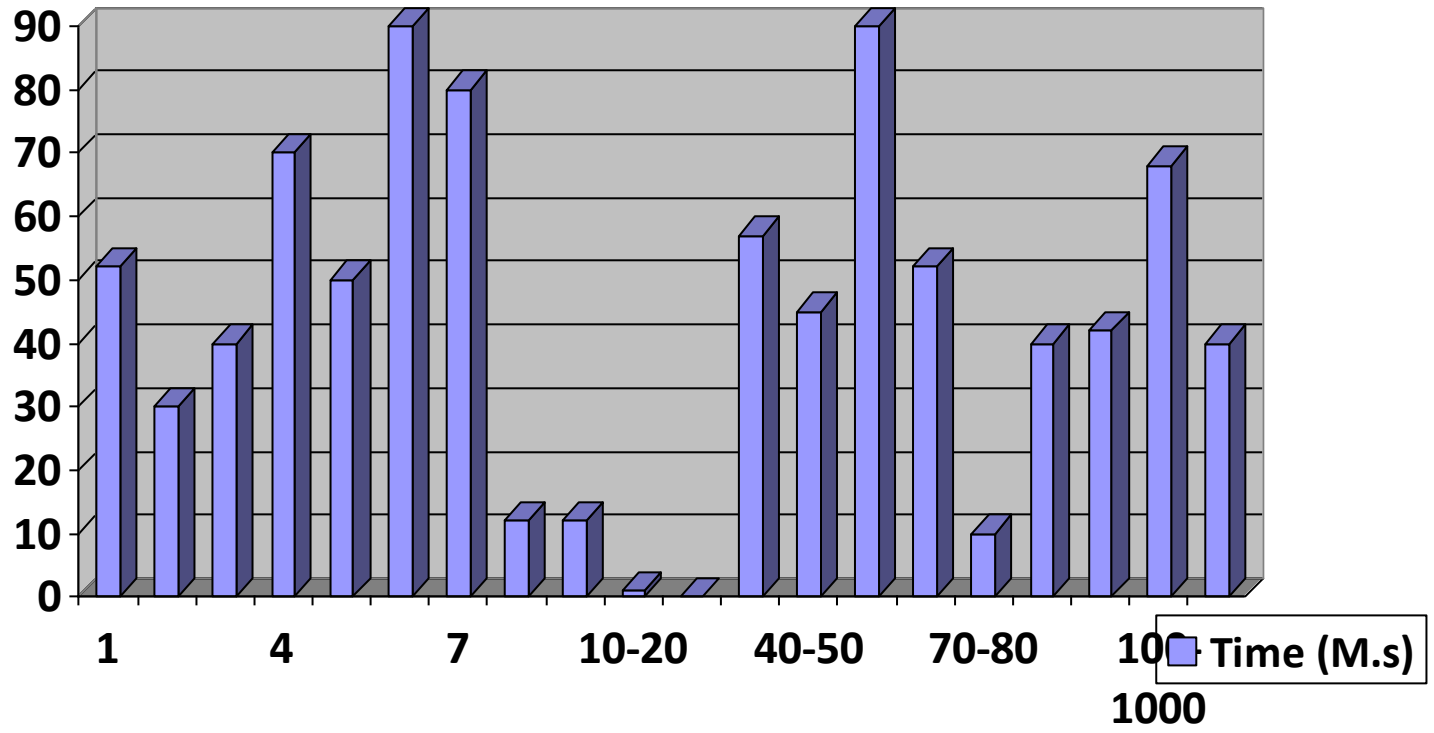
המערכת מורכבת ממספר מודולים עיקריים:

1) מודול לכידה- SNIFFER

a. מותקן בכמה שיותר צמתים ברשת TOR , ככל שיהיו יותר צמתים ביצוע הקורלציה תהיה טובה יותר ותוצאות המנתח יהיו מוצלחות יותר

b. קביעת סוג התעבורה אותה אנחנו מעוניינים לנתח, נבחר להתייחס רק לפקטות בעלות פרוטוקול Tcp ובנוסף נוודא שפקטות אלה לא יכילו את הדגל fin, כיוון שזה אומר סיום התקשרות וחבילות מסוג אלה יכולות להזיק ולבלבל אותנו בניתוח הנתונים

c. הכנסת מרווחי ההשהיות למערך Histogram שמייצג את התפלגות זמני ההשהיה בין פקטות, כך שכל איבר במערך מייצג את כמות החבילות שהגיעו באותו טווח זמן של השהייה בms



d. שליחת המידע למנתח Analyzer.

2) מודול שליחה -

a. קביעת כמות המידע שיש לצבור לפני השליחה למנתח. בחרנו לשלוח בכל פעם כ-1000 פקטות לניתוח.

b. לאיזה כתובת שולחים ובאיזה פורט, כיוון שהשתמשנו ברשת פנימית Local host המחשבים תקשרו בינהם ב IP של רשת פנימית.

c. שליחה לשרת מנתח בעזרת Object Stream את האובייקט וכך נשווה בין אובייקטים.

3) מודול ניתוח Analyzer -

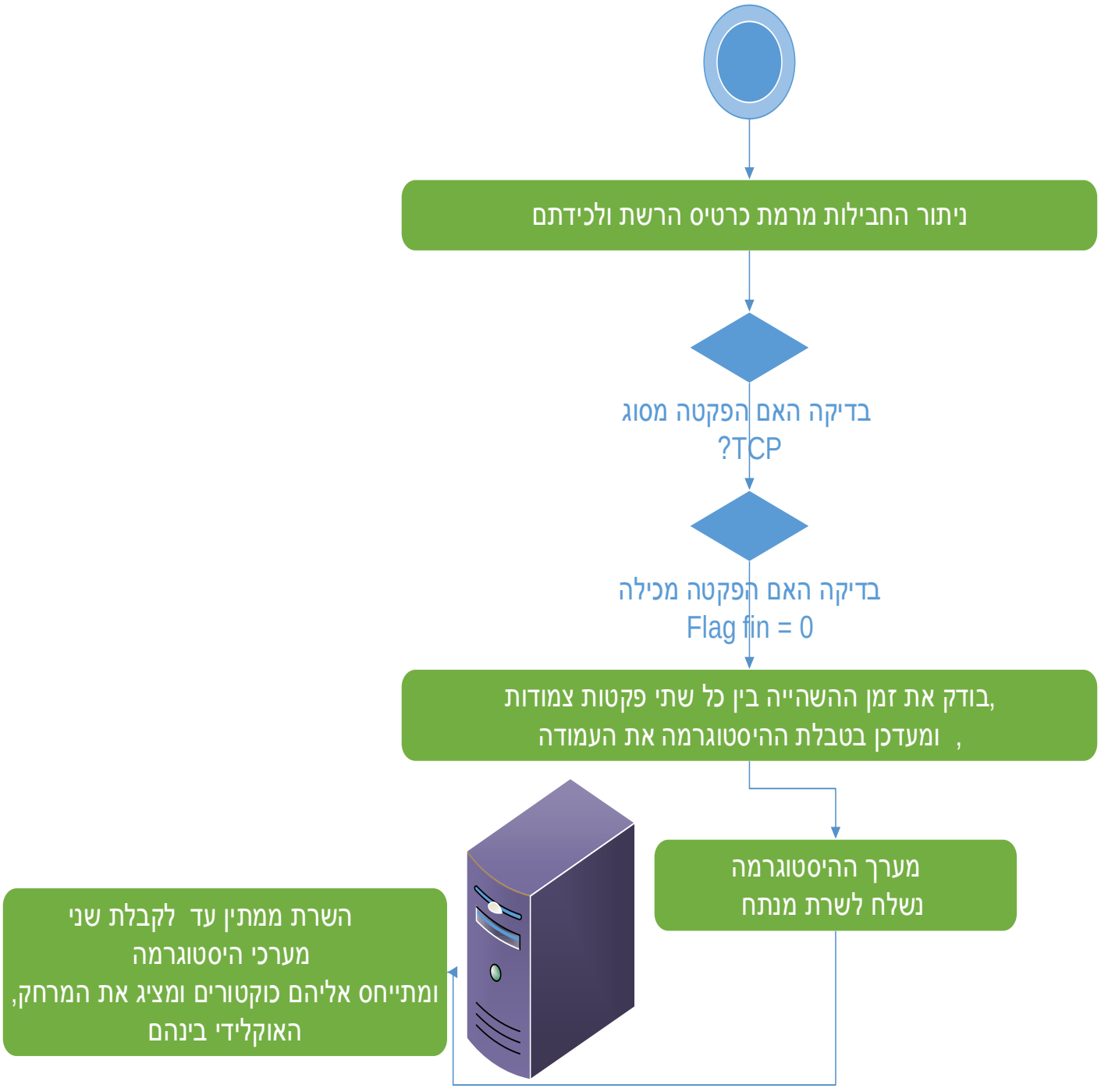
a. ניתוח זמן אמת בקבלת פקטות מצמתים שונים, המנתח מחכה לקבלת 2 מערכי Histogram מצמתים שונים, ומחשב את המרחק האוקלידי בינהים.

b. איסוף פקטות וניתוחם כל זמן קצוב

המערכת המתוארת לעיל מומשה בשפת Java כאשר כל אחד מן המודולים הנ״ל מומש כמחלקה או כמספר מחלקות.

מאופי החלוקה הנ״ל, ולאור העובדה כי מדובר על מערכת Real Time בה מבני נתונים מתעדכנים ושואבים מידע אחד מן השני באינטראקטיביות היה ברור כי על התוכנה להשתמש בריבוי תהליכונים. מודול ההרצה של התכנית יוצר מופע של כל התהליכונים ואלה רצים במקביל לאורך כלל חיי התכנית.

תרשים זרימה:


ניתור החבילות מרמת כרטיס הרשת ולכידתם
בדיקה האם הפקטה מסוג
?TCP
בדיקה האם הפקטה מכילה
Flag fin = 0
בודק את זמן ההשהייה בין כל שתי פקטות צמודות,
ומעדכן בטבלת ההיסטוגרמה את העמודה ,
מערך ההיסטוגרמה
נשלח לשרת מנתח
השרת ממתין עד לקבלת שני
מערכי היסטוגרמה
ומתייחס אליהם כוקטורים ומציג את המרחק,
האוקלידי בינהם

מימוש ה- Sniffer:

לאחר ההתקנה של WinpcP ו JnetPcap בנינו מנגנון סינון, שלוכד את החבילות שרצות על כרטיס הרשת בעלות פרוטוקול תקשורת Tcp ובנוסף הגדרנו שחבילת אלו לא יהיו חבילות סיום תקשורת, ניתן לזהות חבילות אלה בעזרת דגל Fin שמופיע בפרטי החבילה

את חבילות אלה הכנסנו למערך בזמן אמת בכדי שיהיה יותר נוח לקחת מהם את המידע. לאחר איסוף 1000 חבילות, בדקנו את הפרשי זמני ההשהיה במאית השנייה בין כל 2 חבילות עוקבות, ובזמן אמת עדכנו את מערך ה Histograma שמערך זה מייצג את התפלגות זמני ההשהיה בין הפרשי פקטות צמודות , כך שכל איבר במערך מייצג את כמות החבילות הצמודות שהגיעו בהשהיה אשר באותו טווח זמן בMs שמודגר באותו תא במערך.

סביבת העבודה

רשת TOR

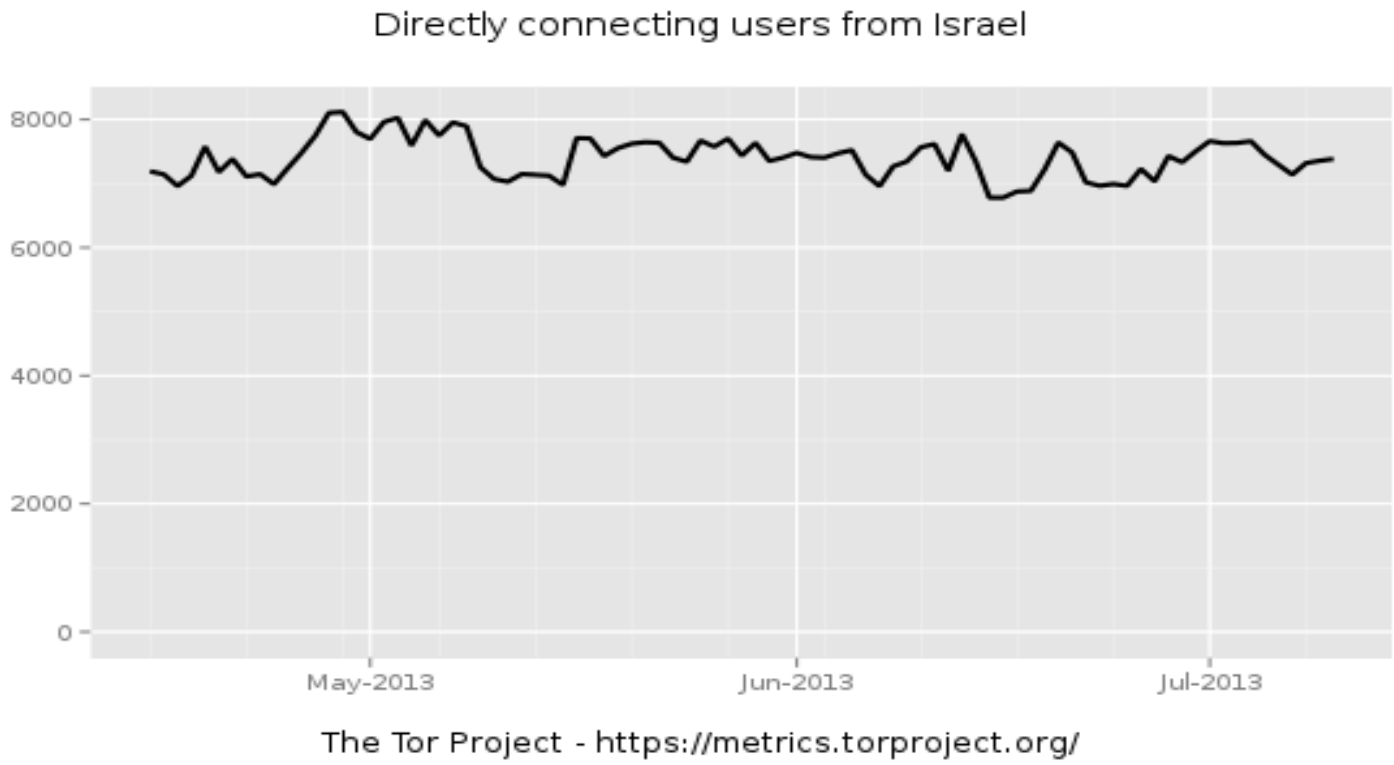
השימוש ברשת זו, אינו מבטיח חיסיון מוחלט, אך סביר בהחלט למשתמש הממוצע .

ל TOR כמה מאפיינים אשר הופכים אותה לכלי חזק מאוד בידיו של כל משתמש.

חיסיון – כאשר אנו שולחים חבילה (Packet) ברשת האינטרנט, החבילה יוצאת מאצלנו ונשלחת אל השרת אשר אותו שמנו ככתובת היעד, והחבילה הנ״ל תעבור דרך אוסף של רשתים ומחשבים אשר יפנו את החבילה עד אשר החבילה תגיע אל היעד. במצב זה, השרת אשר אליו מגיעה החבילה שלנו, יודע בדיוק מהכין הגיע החבילה ומקים חיבור ישיר אלינו. כאשר אנו מעבירים את החבילה דרך TOR החבילה אינה מגיעה ישירות לשרת ואין לנו בעצם קשר ישיר אל השרת. החבילה שלנו תעבור ותתפצל בין כמה Relays בדרך וכך לשרת לא תהיה את כתובת ה IP-הישירה שלנו.

הצפנה – החבילות שאנו נשלח אל השרת, לא ייצאו כמו שהם יוצאות באופן רגיל דרך האינטרנט. החבילה תהיה מוצפנת בכמה שכבות. ככל שהחבילה תתקרב יותר על היעד שלה, השכבות האלה יופשטו ממנה לאט לאט באופן עבודה זה, אנו מקשים מאוד על התקפות Man in the Middle אשר מנסות לתפוס את החבילות בדרך.

חסימה –רשת ה-TOR חוסמת באופן אוטומטי העברה של אותם cookies ומגבילה לאין שיעור את ההעברה של אותם cookies אל אתרים אשר אינם מורשים. בנוסף לכך, כאשר אנו מפעילים את TOR, התוכנה מסירה את cookies הקיימים כרגע על המחשב על מנת לבטל את המעקב.



מהנתונים עולה כי לאורך חודש יוני בשנת 2013 היו מחוברים לרשת Tor בממוצע כ-7814 משתמשים מישראל בממוצע כל יום. כפי שאפשר לראות בגרף המצורף נתון זה נשאר יחסית יציב לאורך מספר החודשים האחרונים.

נתון נוסף שאפשר למצוא באתר הינו כמות המשתמשים המתחברים לרשת באמצעות "גשרים", גשרים אלה הינם חיבורים ייעודיים שנועדו לעקוף חסימות ברמת ספק האינטרנט או המדינה שנועדו למנוע התחברות לרשת. מסיבה זאת חלק מכתובות ה IP-של הגשרים אינם פומביים ומופצים בצורות שונות למשתמשים ממדינות בהן הגישה לרשת חסומה. בחודש יוני היו מחוברים באופן זה כ-269 משתמשים מישראל בממוצע מידי יום.

בחודש ינואר 2013 פעלו מישראל בממוצע כל יום 14 תחנות ממסר וניתן לראות צמיחה לאורך מספר השנים האחרונות בתחנות הממסר הפועלות מישראל.

*תחנות הממסר הן הצמתים בהן מוסרות שכבות הצפנה ובפרויקט שלנו אנחנו מכוונים לצמתים האלו לשם הטמעת הSniffer.

תקשורת עם השרת:

הקמת הקישור:

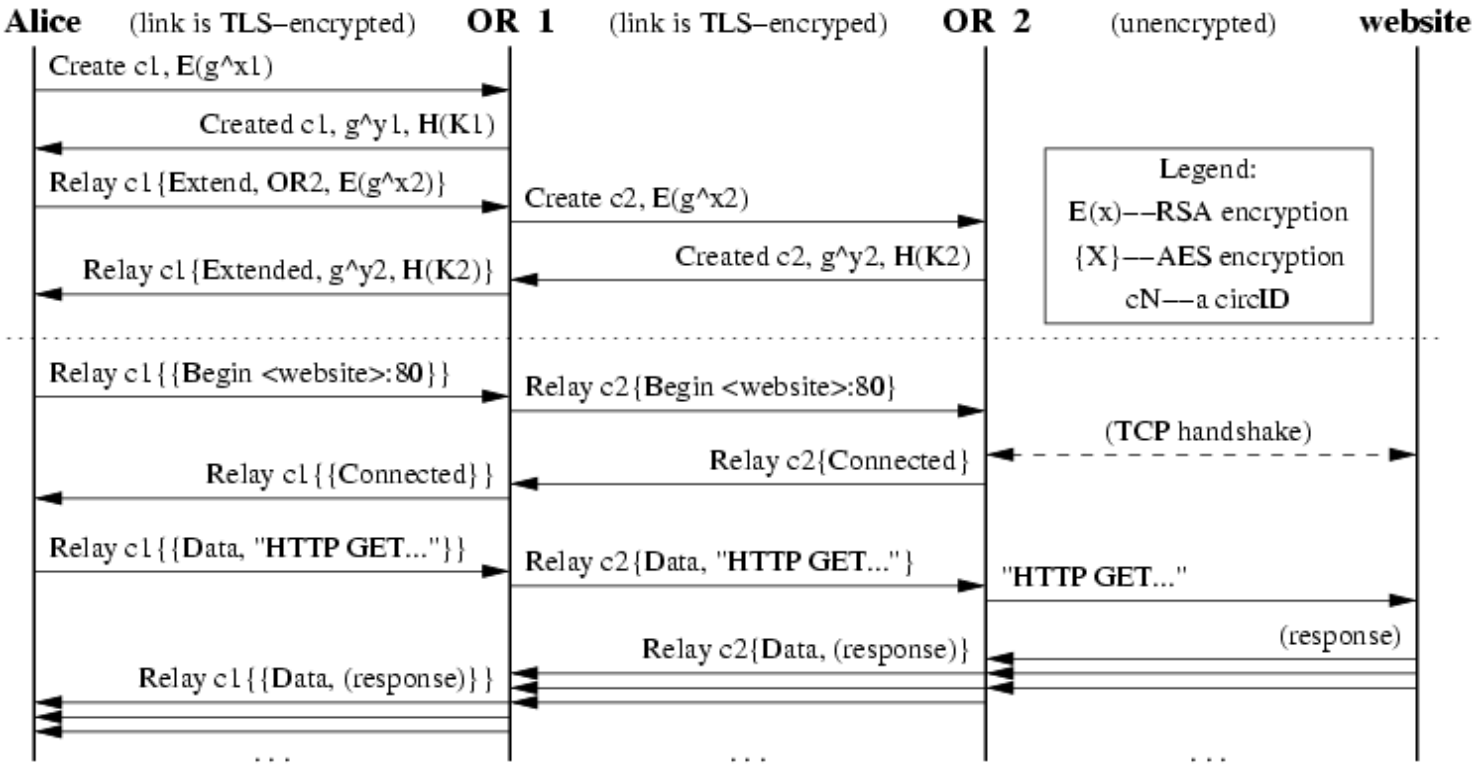
"בוב" שולח את בקשת החיבור שלו לאתר דרך הפרוקסי של TOR שנקראOnion proxy ולצורך העניין יודע לעבוד עם כל אפליקציה שתומכת ב SOCKS, הניתוב הקובע דרך אילו Onion Routers תעבור הבקשה, שמורכב מ Circuits נקבע מראש בצד המשתמש שיודע אילו Onion Routers זמינים.

בעזרת שרת ה TOR Directory שהוא שבעצם Onion Router מוכר ואמין שקיבל את האפשרות להיות שרת כזה יחד עם תעודה דיגיטלית שתחתום את הנתונים שהוא יודע להעביר כדי שאנחנו נהיה שמחים ורגועים.

דבר חשוב נוסף שכדאי לדעת הוא ש TOR משתמש ב Telescoping circuits, מה שאומר ש "בוב" מכיר את כל הדרך ובעצם מתבסס על מבנה שנקרא Leaky-pipe circuit topology המאפשר לו לקבוע מאיזה Circuit המידע יוצא ליעד הסופי מה שעוזר בהתמודדות עם התקפות שונות המבוססות על Traffic observation, הסשן שנפתח מול ה Onion Routers מבוסס TLS מטעמי יעילות ואבטחה, עבודה עם הצפנה א-סימטרית לאורך כל ההתנהלות לא ריאלית מהיבט של ביצועים ובעייתית מאוד ברמת ייתכנות היישום בפועל.

"בוב" מתחיל סשן TLS מול ה Onion router הראשון בדרך שנקרא גם Entering node לאחר מכן "בן" עושה relay דרך ה Entering node לכיוון ה Onion router הבא איתו יצור סשן TLS, במידה וה Onion Router הנוכחי ממשיך ומשרת את "בוב" כדי לעשות עוד Relay לעוד Onion Router הוא יקרא Relay node וכך ממשיך התהליך עד ה Onion router האחרון שתפקידו להעביר את המידע ליעד הסופי, ה Onion router הזה נקרא Exit node, בסופו של דבר "בוב" יוצר n - סשנים של TLS כאשר n מייצג את מספר ה Onion routers דרכם הוא עובר. שימו לב שבמתודולוגיה הזו כל Onion router מכיר רק את שכניו הקרובים ולא את כל ה Circuits.

כל ה Circuits יחדיו מוגבלים באופן אוטומטי למשך זמן של 10 דק' שאחריו הם משתנים. בנוסף , TOR מבצע גם End-to-end integrity checking



דוגמא ממשית:

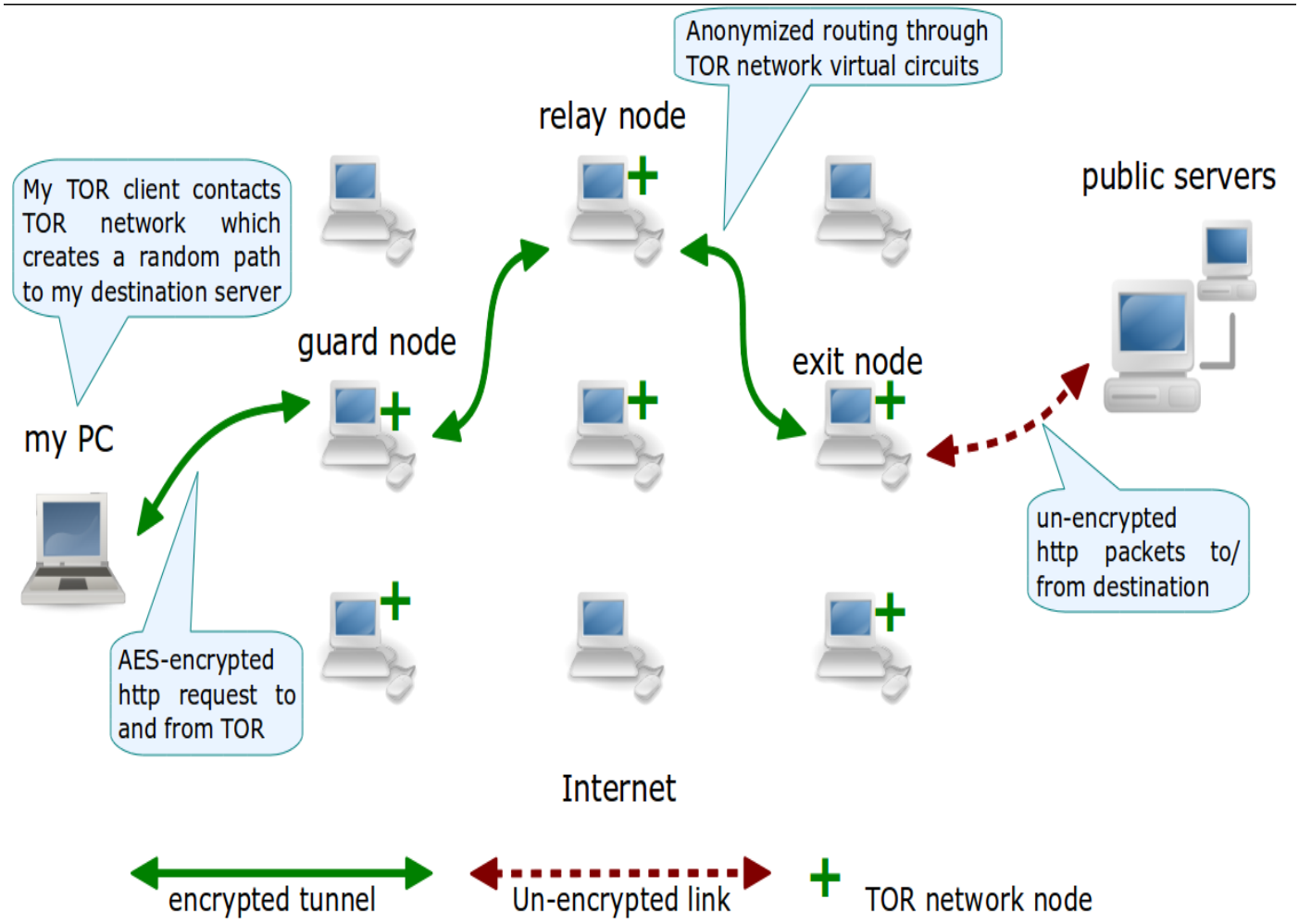
ראשית, על אליס להתקין במחשב שלה מערכת TOR כעת כאשר אליס רוצה לגלוש באינטרנט בצורה אנונימית, מערכת ה TOR המותקנת אצלה תבחר באופן אקראי 3 מתווכים ברשת. המתווכים ברשת הTOR הם משתמשים רגילים שלמען חופש הביטוי וזכויות האדם התנדבו להתקין תוכנה שתעביר תעבורת אינטרנט אנונימית עבור משתמשי TOR. כשאליס רוצה להשאיר תגובה באתר של בוב, תוכנת ה TOR-של אליס תיקח את התגובה ותעטוף אותה בשכבות, כפי שראינו בדוגמא מהעולם האמיתי. השכבה הפנימית ביותר תכיל את התגובה אותה אליס רוצה לשלוח ואת כתובת ה IP-של עלי (5.5.5.5) בתור השולח ואת כתובת ה IP-של בוב (2.2.2.2) בתור המקבל. בשכבה החיצונית הבאה התהליך חוזר על עצמו עם ה IP-של דבי (4.4.4.4) בתור השולחת וה IP-של עלי בתור המקבל. בשכבה החיצונית הבאה ה IP-של פרנק (3.3.3.3) הוא השולח וה IP-של דבי הוא המקבל. ובשכבה החיצונית ביותר ה IP-של אליס הוא השולח (1.1.1.1) ושל פרנק הוא המקבל. בנוסף, כמו בדוגמת העולם האמיתי, כל שכבה מוצפנת, ורק היעד של השכבה יכול לפענחה

ולדעת למי להעביר הלאה. מאחר וכל תחנה במסלול מקלפת את השכבה שלה ומעבירה לתחנה הבאה, עיקרון זה נקרא ניתוב

פרוטוקול TLS הוא היורש של פרוטוקול SSL שמשמש להצפנת התעבורה בין הדפדפן לבין אתרים שונים ובנוסף משמש כאמצעי לאימות זהות האתר. השיטה מבוססת על תעודות. הדרך הפשוטה ביותר להבחין שאתה משתמש בה היא שהקישור מתחיל ב-HTTPS במקום HTTP.

בכרום יוצג אייקון של מנעול, כאשר לעתים זה יעשה רקע של צבעים שונים .

איור דוגמא לזרימה ברשת Tor



hidden Service:

זהו מקרה שהמוען הנמען והמסר אנונימיים (אף אחד חוץ מהמוען והנמען אינו מודע למסר). במצב זה התקשורת לעולם אינה עוזבת את רשת Tor לכן מוצפנת מתחילתה ועד סוף ויהיה שימוש במעגל כפול.

תחילה צריך ליצור Hidden Service משמע פירסום שלו בתוך רשת Tor, לשם כך נבחרים מספר קשרים שונים אשר ישמשו כמאזינים עבורו, נוצר מעגל אליהם והם מתבקשים לשמש כ-Introduction Point, לקשרים הללו מועלה המפתח הפומבי של השירות הנסתר.

בעוד שהקשרים יהיו מודעים לתוכן שכן הועלה אליהם המפתח הפומבי שמותאם עם ה-Hidden Service הם לא יהיו מודעים לכתובת IP שכן התקשרות התבצעה בעזרת מעגל (תקשורת אשר עוברת דרך 3 קשרים באופן מוצפן).

השלב השני הוא יצירת Hidden Service Descriptor שיכיל את המפתח הפומבי ותקציר של קשרים אשר משמשים כ Introduction Points זאת בכדי שהשירות הנסתר יהיה נגיש ברשת Tor .

המידע הזה יועלה לרשת DHT כאשר הוא חתום במפתח הפרטי של יוצר ה Hidden Service כאשר קלינט כלשהו מבקש לגשת ל Hidden Service כלשהו, הוא ידלה את המידע שהועלה לרשתDHT שמותאם עם ה Hidden Service Descriptor ובכך הוא יזכה לדעת מי הם הקשרים אשר משמשים כ Introduction Pointsלכן קלינט שרוצה לגשת ל Hidden Service חייב להכיר את הכתובת של השירות מראש. כאשר הקליט מציג בקשה לסיומת onion הוא מבקשת מידע (מבקש את ה Hidden Service Descriptor מהרשת DHT בתקווה שקיים מידע כלשהו על הכתובת הזאת) ובכך הוא מקבל את המפתח הציבורי של השירות ואת הקשרים אשר פועלים Introduction Points .

השלב הבא שהקלינט מבצע הוא יצירת מעגל לקשר שישמש כ Rendezvous Point-שיתקשר בין המעגל של הקלינט למעגל של ה Hidden Service.

ל Rendezvous Point נמסר סיסמא חד-פעמית (Cookie), ולאחר מכן הקלינט יוצר מעגל תקשורת לאחד מה Introduction Points (על-ידי מעגל, לא קשר ישיר) ומוסר לו מידע בדבר ה Rendezvous Point-שימסר לHidden Service , ההודעה עטופה על-ידי המפתח הפומבי של השירות ובין היתר היא מכילה את תחילתה של לחיצת היד להחלפת מפתחות בעזרת פרוטוקול DH .

ה-Hidden Service יוצר מעגל תקשורת אל ה Rendezvous Point ותוך שימוש בסיסמא החד-פעמית מאשש את זהותו ונותן את המחצית השניה של לחיצת היד בפרוטוקול DH וזה המפתח אשר יצפין את התקשורת כאשר היא מגיעה ל Rendezvous Point, כל התקשורת בין הקילנט וה Hidden Service תעבור דרך ה Rendezvous Point כאשר כל אחד משתמש במעגל שיצר, וכאשר התקשורת יוצאת מהמעגל בכל צד, היא עדיין תהיי מוצפנת בעזרת המפתח שהוחלף בעזרת פרוטוקול דיפי-הלמן .

היישום הכפול של מעגל תקשורת ב Hidden Service מאפשר לכל צד לשמור על האנונימיות שלו, שכן כל צד בוחר בעצמו את המעגל תקשורת שלו. למרות שעם הגעת ההודעה ל Rendezvous Pointהמעגלי תקשורת כביכול הסתיימו, ההודעה עדיין מוצפנת על-ידי המפתח שנחלק בין הקלינט לבין ה Hidden Service כל צד אנונימי לצד השני, ובכך המוען, הנמען והמסר נותרים אנונימיים.

Java וה-APIים כל הפקטות

כללי:

הפרוייקט נכתב בשפת Java, בסביבת Eclipse, תוך שימוש בשתי ספריות (jars) חיצוניות.

Jar (Java Archive) הינו קובץ המאחד קבצי מחלקות וmetadata לקובץ אחד, בדומה לקובץ zip. מפתחים נוהגים ליצור קבצי Jar על מנת ליצור קבצי ספריה להם ניתן לעשות import ב-eclipse ולהשתמש בתוכנם.

כאמור, בפרוייקט נעשה שימוש ב Jar: JNetpcap

פירוט:

JNetpcap הינה ספריית קוד פתוח המשמשת ללכידה (capturing) של פקטות ברשת. לאחר ההקלטה ניתן לבצע עיבוד על הפקטות בשפת Java.

הספרייה מגדירה מחלקות רבות אשר הינן הכרחיות לביצוע פעולת הלכידה ביניהם:

- JNetpcapCaptor – המחלקה המרכזית המשמשת ללכידת פקטות
- NetWorkInterface – מחלקה המייצגת ממשק רשת
- NetWorkInterfaceAddress – מחלקה המייצגת כתובת של ממשק רשת
- Packet – מחלקת אב ממנה יורשות מחלקות המייצגות פקטות מסוגים שונים
- IPPacket – מחלקה שיורשת ממחלקת Packet ומייצגת פקטת IP. המחלקה מייצגת באופן מדוייק פקטת IP (גרסא 4 וכן גרסא 6) עם כל השדות הרלוונטיים.
- TCPPacket – מחלקה שיורשת ממחלקת IPPacket ומייצגת פקטת TCP. בדומה למחלקת האב שלה, המחלקה מייצגת באופן מדוייק פקטות TCP עם כל השדות הרלוונטיים.

- PacketReceiver- קובץ ממשק אשר באחריות המשתמש לממש ובו יוגדר מהי פעולת העיבוד אותה אנו מעוניינים לבצע על הפקטה.

באופן כללי נציין כי העבודה עם ה-jar הנ"ל היתה טובה אך יש להיזהר: ה-APIים טוענים כי ניתן לבצע פעולות נוספות (דוגמת שליחה באמצעות JNetpcap) אך אין זה תמיד המקרה. מדובר בספריות אשר פותחו לפני מספר שנים, ככה"נ לצורך ספציפי, ואינן מתוחזקות יותר וישנן פעולות אשר מתוארות ב-API ובפועל גוררות שגיאות.

בנוסף השתמשנו בתוכנה WinPcap:

WinPcap היא תוכנה המותקנת על המחשב ומשמשת עבור לכידת חבילת מידע או בלוק של מידע המועבר ברשת וצורפה לו כתובת השולח והמקבל וכן מידע לתיקון שגיאות, וניתוח הרשת לפלטפורמות, הוא מכיל פילטר (קובץ המאפשר לטעון ולשמור קבצים מסוג מסוים) של חבילת מידע רמת ליבה, רמה נמוכה של ספריית קישור דינמי ,(packet.dll) ברמה גבוהה, ספריית מערכת-עצמאית wpcap.dll) המבוסס על jnetpcap-1.4.r1425-.win64 מסנן חבילת המידע, הוא התקן שמוסיף לחלונות את היכולת ללכוד ולשלוח נתונים גולמיים מכרטיס רשת, עם האפשרות לסנן ולשמור במאגר את חבילות המידע שנשמרות . Packet.dll הוא ממשק לתכנות יישומים (חלק במערכת ההפעלה המספק ליישומים ממשק משתמש אחד) שיכול להיות משומש לגישה ישירה לפונקציות של חבילת המידע של ההתקן, המציעה ממשק תכנותי עצמאי ממערכת ההפעלה של מייקרוסופט .

Wpcap.dll מייצא רמה גבוהה של לכידה של הפקודות הבסיסיות שתואמים עם הlibpcap, ספריית היחידה של הLinux הפונקציות הללו מאפשרות ללכוד חבילות מידע בדרך עצמאית מחומרת הרשת המונחת ביסוד ומערכות ההפעלה .

Linux\Ubuntu

מלכתחילה היה ברור כי הפרוייקט ככל הנראה יפותח על גבי מערכת Linux, שכן התוכנה מבצעת פעולות לא אלמנטאריות מבחינת מערכת ההפעלה (האזנה לממשקי רשת, פתיחת sockets וקישור ה-socket לרכיב רשת ספציפי) אשר דורשות הרשאות גבוהות. בפרט, האזנה לכרטיס רשת לא היתה אפשרית לולא היינו מריצים את ה-Eclipse ללא הרשאת root.

על כן, נבחרה הפצת Ubuntu של Linux כמערכת ההפעלה. הסיבה המרכזית לבחירה ב-Ubuntu היא פשטות השימוש והאינטואיטיביות של מערכת ההפעלה מחד, ויכולות השליטה והרצת פקודות בתור root מאידך.

התקנת Ubuntu הינה פשוטה מאד. העבודה עם מערכת ההפעלה, לאחר חבלי לידה קצרים, הינה פשוטה ומומלצת.

נציין כי ניתן היה לפתח חלקים מהפרוייקט בסביבת חלונות אך היה זה מסובך פי כמה, כיוון שאת הסימולציה של הרשת נערוך על Linux ודבר זה היה דורש מאיתנו לעשות התאמות מיותרות.

תהליך הלמידה

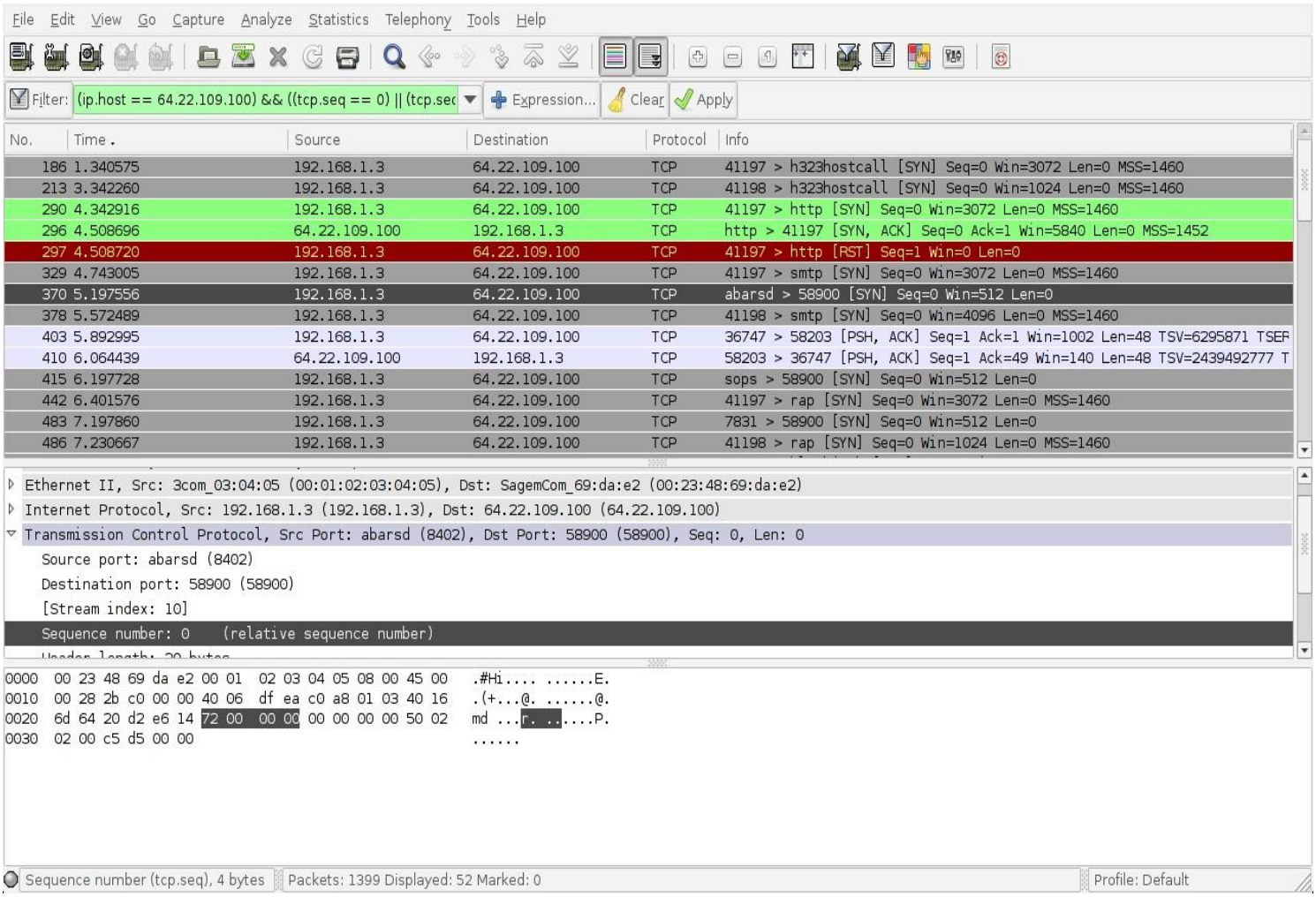
עקומת הלמידה שלנו במהלך הפרויקט הייתה חדה למדי. עברנו תהליך ארוך ומשמעותי תחת הנחיה מקצועית ויעילה.

בשלב הראשון למדנו להכיר דרך תאוריה את רשת Tor איך הממסרים עוברים, דרך כמה נקודות, איך נראה ממסר ועוד.

בשלב השני התחיל תהליך התכנות שכלל בניית Sniffer, Analyzer ומישוש סימולציה של רשת.

בתחילת הדרך, למדנו את עקרונות העברת נתונים ברשת, שם הבנו כי התמונה המלאה נמצאת בפרוטוקולי TCP/IP. לאחר הרחבת ידע בפרוטוקולים אלו, התחלנו להתוודע לסביבות העבודה אשר פורטו בפרקים הקודמים ,שלב ה Sniffing יתבצע בממסרים ברוחב פס גבוה ובממסרים אשר בעלי זמן חיים ארוך יחסית לממסרים האחרים, כמה שיותר ממסרים, חלוקת רוחב פס טובה יותר.

בשלבים הראשונים, התקנו wireshark והתחלנו לקבל "תחושה באצבעות" של העבודה עם פקטות והתבוננו בסוגי הפקטות השונות והפילטרים השונים. במקביל בחרנו להשתמש בשפת java, והתחלנו להתוודע לספריות העתידיות איתן נעבוד. איור של Wireshark



ברור היה כי בניית התוכנה שתשב על הצמתים תתחלק לשני שלבים מרכזיים: זיהוי הפקטות ה"חשודות" ושליחת המידע לשרת מנתח.

בהתייעצות עם המנחה בחרנו לעבוד עם ספרייה מסוג Rocksaw אשר תנתר את ההודעות ברמת כרטיס הרשת, אך לאחר נסיונות רבים לבניית תוכנה שתנתר הודעות בעזרת ספרייה זאת, הגענו למסקנה שזאת לא הדרך הנכונה ואנו לא מצליחים לקבל את כל הנתונים שעוברים ברשת כפי שציפינו.

לאחר התייעצות נוספת עם המנחה בחרנו לנסות להשתמש בספריית JNetpcap, שתתלבש על ה-DRIVER של כרטיס הרשת וכך, נקבל עותקים של הפקטות העוברות ברשת (במסגרת רמה שנייה במודל שבע הרמות). על מנת לממש את הממשק בין PNetcap ל-JAVA נעשה שימוש בספרית JNetpcap.

בהמשך דנו במנגנון פילטור הפקטות ודנו בחלופות: האם לבצע את הפלטור במסגרת ה"מוח" של התוכנה (האחראי על הוראות ה"חיסול") או מנגנון נפרד ומקדים.

בהמשך הדרך נתקלנו בלא מעט קשיים טכניים, לדוגמא: קישור ספריית JNetpcap ב-Eclipse.

השלב הבא היה לכתוב את החלק הראשון (sniffing). דנו במימושים אפשריים שונים, וכיצד כל אחד מהם נותן מענה לבעיית ה real-time. יש לזכור כי המערכת צריכה להתמודד עם זרימת אלפי פקטות בשנייה. היה ברור כי אין טעם להעתיק ממש כל פקטה לזיכרון המחשב על מנת לפלטר אותה, אלא רק את אלו שעברנו את סינון הjpcap. החלטנו לממש 2 מבני נתונים דינמיים המבוססים על עיקרון LIFO. בהתחלה ניסינו להשתמש במחסנית גנרית של java, אולם עקב קשיים טכניים החלטנו לממש את המחסניות בעצמנו.

בעייה נוספת שנתקלנו הייתה עבודה עם serialization שכמעט כל שינוי במבנה מחלקה המוגדרת -כ serializable מונע את האפשרות לקרוא אובייקטים שנכתבו ע"י גירסאות קודמות

שלה (תיזרק InvalidClassException). Java מחשבת עבור המחלקה קוד גירסה ייחודי, הנגזר מכל המשתנים והפונקציות שהיא מכילה. גם בשינוי שם של פונקציה במחלקה או מספר הפרמטרים שהיא מקבלת – משתנה קוד זה. אובייקטים נכתבים לקובץ עם קוד גירסת המחלקה, ונקראים חזרה מהקובץ רק אם קוד המחלקה זהה לקוד בקובץ. ניתן לעקוף זאת ע"י הגדרה עצמאית של קוד גירסת המחלקה, אך גם אז הדבר יעבוד רק אם מבנה השדות במחלקה לא השתנה – כאשר מוסיפים ומורידים משתנה מופע שאינו transient למשל – חובה להחליף את קוד הגירסה, אחרת הקריאה עשויה להחזיר מידע שגוי.

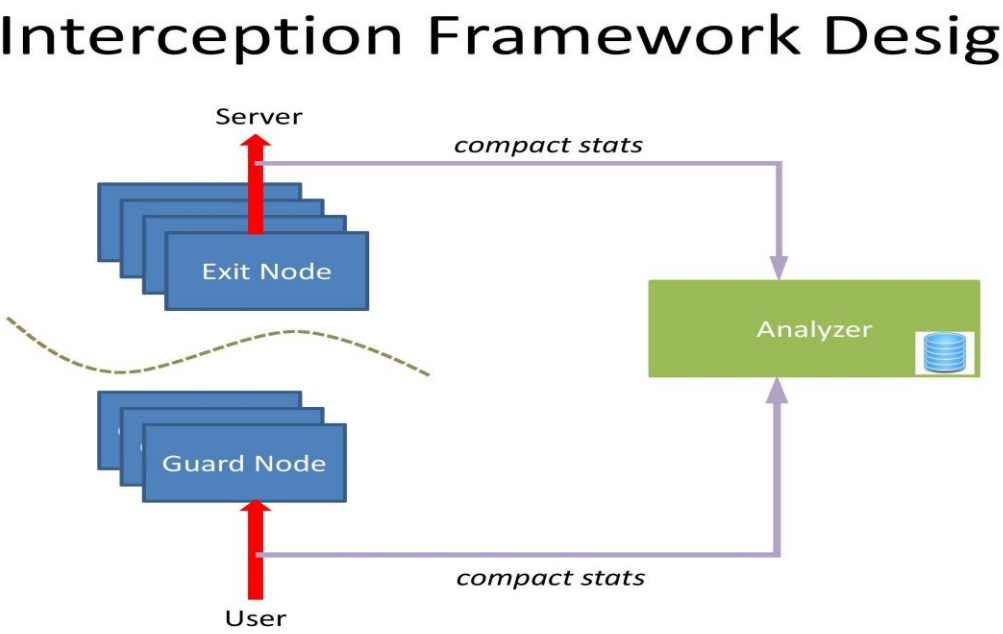
בתחילת הפרוייקט המחשבה הייתה לפתוח כמות Guard node פעילים ברשת Tor

ובכך לנסות לזהות קורלציה בינהם, לאחר בדיקה מעמיקה ושיחות מול המנחה הבנו שרעיון זה לא יישים, גם מבחינה כלכלית שדורשת קניית שרתים גדולים כיוון שהקורלציה צריכה **להתבצע בצמתים בעלות אורך** פס גבוה, וגם מבחינת זמן שהשרת צריך להיות ב"אוויר" בכדי להיות אמין ע"י רשת Tor ובכך יוכל להיות Guard node .

לאחר מכן הרעיון היה לבנות רשת Tor פנימית כאשר Sniffer יותקן על כל Guard node ו Exit node ומטרתו לשלוח את המידע שנאגר למחשב ייעודי (Analyzer) שמטרתו לנתח את המידע שהגיע מהצמתים ובעזרת סטטיסטיקות לנתח את תעבורת הרשת .

ראשית לשם בדיקת ייתכנות המערכת נעשתה בניית סביבת סימולציה במחשב יחיד בעל מערכת הפעלה Linux שתדמה תעבורה ברשת TOR בעלת משתמשים רבים, האופן בו זה בוצע הוא ע"י יצירת כתובות מדומות (127.0.0.X) כך ששינוי הX יהווה כתובת חדשה. לאחר השגת מטרה זאת אפשרי להרחיב את הניסוי לעשרות מחשבים שישמשו כצמתים שידמו את הרשת.

בוצעו שינויים בקובצי קונפיגורציה במערכת הפעלה Linux במטרה לאפשר ריבוי כתובות IP כמו כן בקוד של תוכנת Tor בוצעו שינויים כך שהתוכנה לא תזהה שאנו משתמשים בריבוי משתמשים מאותו מחשב. במקביל גרמנו לחבילות לבצע delays בין התהליכים כך שידמה משתמשים אמתיים ברשת.
השתמשנו בספריית JNetPcap שהיא ספריית קוד פתוח המשמשת לעבודה עם פקטות ברמת כרטיס הרשת בשפת Java. כל Guard node ישלח את הנתונים שאגר בעזרת הSniffer למחשב ייעודי אחת לכמה זמן שנגדיר.
איור המדמה את הרעיון שעליו עבדנו



לאחר חשיבה מרובה מול המנחה, הגענו למסקנה שבניית רשת פנימית של Tor תהיה שונה בתכלית מרשת Tor שפרוסה בכל העולם.

לכן הגענו למסקנה שבכדי לזהות קורלציה בין משתמשי קצה,
אנו נצטרך לזהות תקשורת בין משתמש קצה לבין שרת כך שהמשתמש מתחבר לשרת, ובשני הצדדים מותקן Sniffer, ושולח את הנתונים לשרת מנתח.
ליישום רעיון זה, נאלצנו לבנות שרת שעובד עם Hidden services מה שבעצם אומר בניית אתר ברשת Tor בעל סיומת onion.
משתמש הקצה מתחבר לאתר דרך רשת Tor, וכך השגנו את הקורלציה בין משתמש הקצה לשרת שיושב עליו האתר.

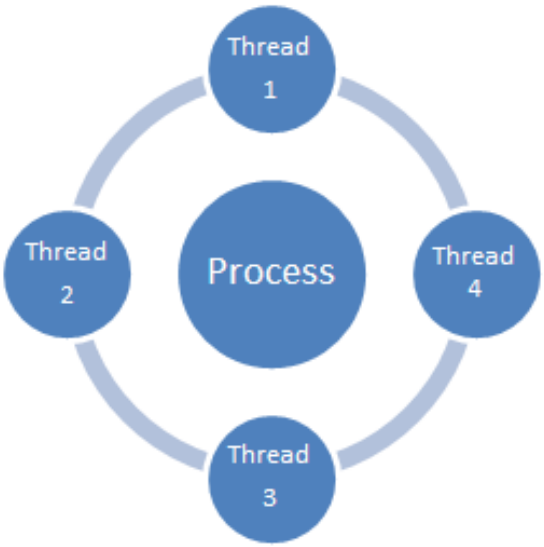
מקביליות

הרעיון היה כי מחסנית אחת תשמש לאחסון הפקטות שעברו ניתוח וזוהה איזה פקטות מתאימים לנו מבחינת המידע הכלול בו, לדוגמא אם הפקטה בעלת השהייה גדולה מהמשתנה שקבוע מראש , לא נתייחס אליו .

ואם ישנו fin (דגל בקשת הניתוק) אנחנו נפטרים ממנו.

חשוב לציין שנתייחס רק לפקטות בעלות פרוטוקול תקשורת TCP, כיוון שרשת Tor עובדת רק בפרוטוקול זה.

ומחסנית שנייה תאחסן את הפקטות. סנכרון בין 2 המחסניות, דורש שימוש בmulti-threading. היינו חייבים thread נוסף השתמשנו במנגנון הsynchronize של שפת java.



סימולציה

כלים שהשתמשנו

VMware–כדי לבצע סימולציה נאלצנו להשתמש במערכת הפעלה וירטואלית, היתרון העיקרי של ה-VMware הוא בכך שהיא מאפשרת שימוש במקביל במערכות ההפעלה חלונות, לינוקס ויוניקס על אותו מחשב.

השתמשנו ב ubuntu-14.04.1-desktop-i386 גרסה 32 ביט כדי לממש את הסימולציה של רשת TOR.(מערכת הפעלה Ubuntu תקרא Linux במאמר)

אחרי התקנת ה Linux ב VMware צריך לבדוק שהגדרות הרשת (Network Adapter) מוגדרים

כ NAT: used to share the host's IP Address , כדי למנוע בעיות תקשורת .

ARM – תוכנה שמצבעת ניטור על צמתים אנונימיים ברשת TOR, מאפשר תצפית גרפית של זרימת הנתונים ברשת ,מאפשרת לראות את כל הצמתים בהם הלקוח עובר עד הגעתו ליעד המבוקש ,ניתן דרך התוכנה להוות צומת יציאה ברשת או לקוח פשוט ,ניתן דרכו לראות עוד שלל אפשריות כגון :

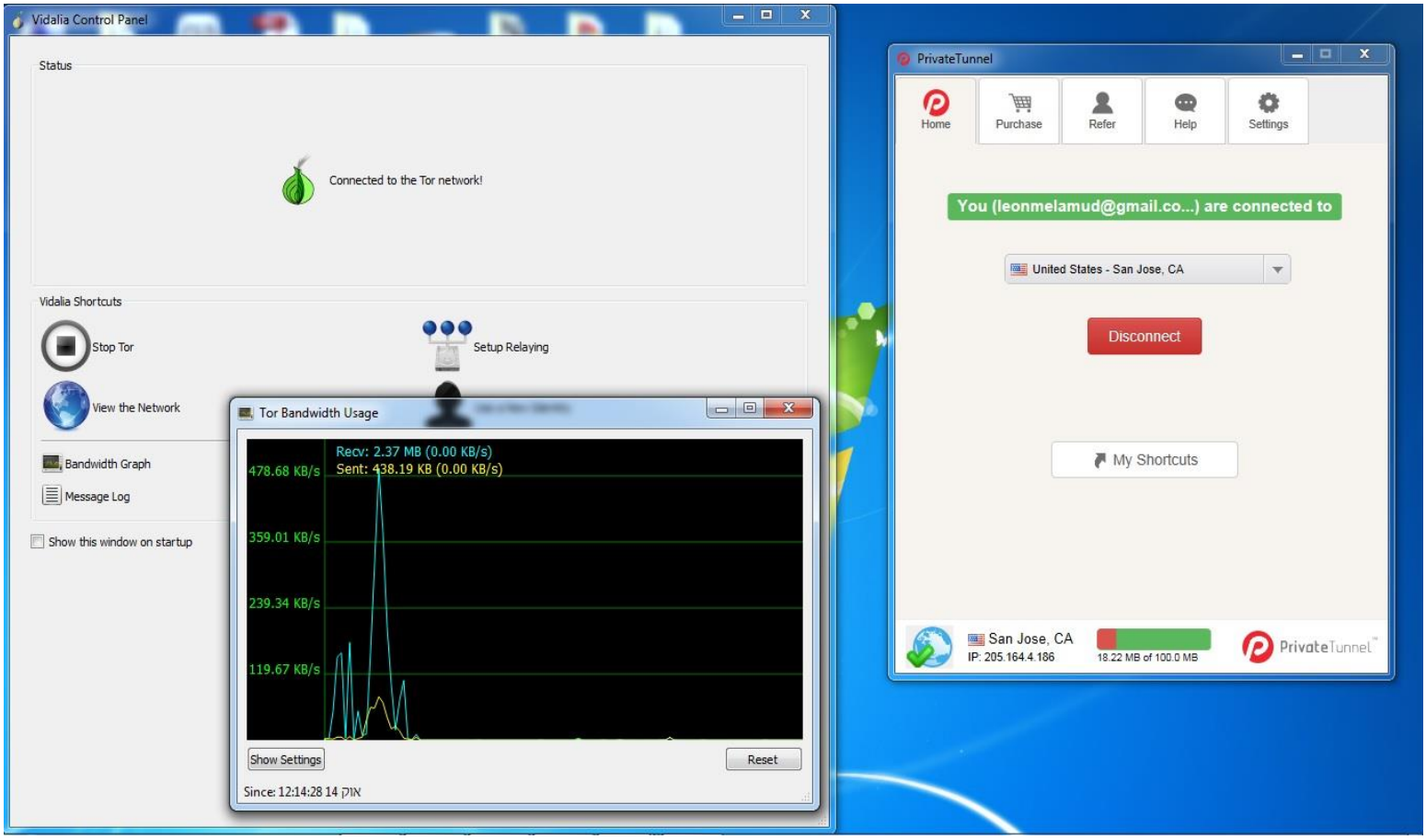
- שימוש במשאבים (רוחב פס, CPU, ושימוש בזיכרון)
- מידע ממסר כללי (כינוי, טביעת אצבע, דגלים, or/ dir / controlports)
- יומן אירועים עם סינון רגולרי אופציונאלי ומניעת כפילויות
- קובץ תצורת Torrc עם הדגשת תחביר ואימות

כלי מאוד יעיל ונוח כאשר עושים שינויים נרחבים בהגדרות של הרשת.

Lighttpd - הוא שרת אינטרנט מאובטח, מהיר, תואם, וגמיש מאוד שכבר מותאם לסביבות עתירות ביצועים. שימוש בזיכרון הפנימי נמוכה מאוד בהשוואה לשרתי אינטרנט אחרים והוא גם דואג שלא יהיה עומס על המעבד.

Nautilus- נאוטילוס תומך בגלישת מערכות קבצים מקומיות, כמו גם מערכות קבצים זמינות באמצעות מערכת, השימוש שלנו היה שהיה צורך לגישה גרפית לקבציי מערכת כאשר התעסקנו עם הקבצים של האתר.

Vidalia - היא פלטפורמה גרפית לתוכנת Tor התומכת ברוב מערכות ההפעלה ,Vidalia מאפשר לך להתחיל ולהפסיק את Tor, לראות כמה רוחב פס אנו צורכים, לראות כמה מעגלים יש לנו כעת , לראות איפה מעגלים אלה מחוברים על מפה עולמית, צפיית הודעות מTor על התקדמותו ומצבו הנוכחי, ולתת לך להגדיר את הלקוח Tor, הגשר(Bridge) שלך, או לעבוד עם ממשק פשוט. השימוש שלנו בתוכנה היה בצד לקוח במערכת הפעלה וינדוס. ניתן לראות שהיה צורך בשימוש בvpn במקומות שחוסמים גישה לרשת (לדוגמא במכללה)



FoxyProxy – תוסף לפיירפוקס שמאפשר לגלוש בדפדפן דרך רשת Tor .

בניית הסימולציה

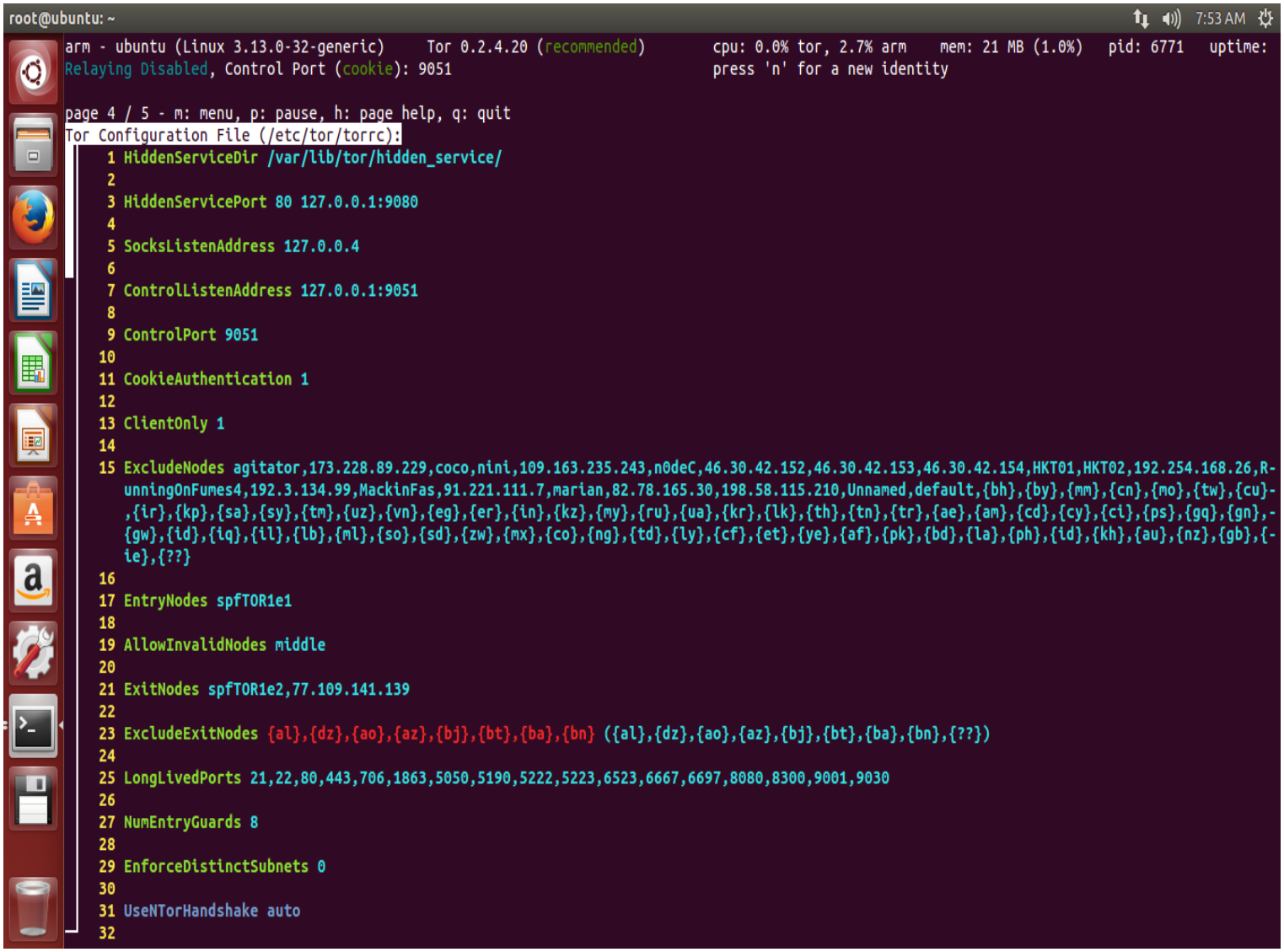
ראשית אנו נשתמש ב Terminal כדי לבצע את הסימולציה .

כדי להשתמש ב TERMINAL כמנהל יש צורך לרשום sudo bash (חוסך הרבה צרות).

התקנו את תוכנת TOR ללא ה Bundle שמתקין את הדפדפן ברירת מחדל של TOR

לאחר ההתקנה יש צורך לשנות את הקובץ TORRC שנמצא ב etc/tor/torrc/

קובץ ה TORRC(תצוגה ממסך התוכנה ARM) איור 1



הסבר על ההגדרות הקריטיות בקובץ

- SocksListenAddress 127.0.0.4

שינוי הכתובת האזנה ,כדי שלמחשבים שונים לא יהיו כתובות דומות ברשת הביתית(אחרת רק מחשב אחד יוכל להתחבר לרשת TOR

- HiddenServiceDir var/lib/tor/hidden_service/

בנתיב הבא הוא שומר את הקובץ hostname אפשר בתוכו יהיה כתובת האתר בסיומת ONION

- HiddenServicePort 80 127.0.0.1:9051

הקצאה של כתובת אשר תתקשר עם השרת במחשב ותתחבר דרכו לרשת TOR

- EnforceDistinctSubnets 0

אפשרות ששתי מחשבים בעלי IP דומה יתחברו לאותו מעגל, זאת אומרת שללא אפשרות זאת רק מחשב אחד יוכל להתחבר מIP מסוים.

- EntryNodes spfTOR1e1

הבחירה כאן הייתה להשתמש באותו GUARD NODE כדי לספק יציבות במערכת.

- LongLivedPorts 21,22,80,443,706,1863,5050,5190,5222,5223,6523,6667,6697,8080, 8300,9001,9030

כאן אנחנו מגדירים פורטים בעלי תוחלת חיים ארוכה כדי לתת יציבות למערכת ורצוי להגדיר כאשר יש hidden service (אתר).

- UseNTorHandshake auto

לחיצת היד "NTOR" היא בהרבה יותר מהירה ומאובטחת מהלחיצה הרגילה "TAP" , כאשר הוא על אוטו' הוא יחפש צמתים שעובדים עם "NTOR" ובמקרה הוא לא ימצא ישתמש ב"TAP" .

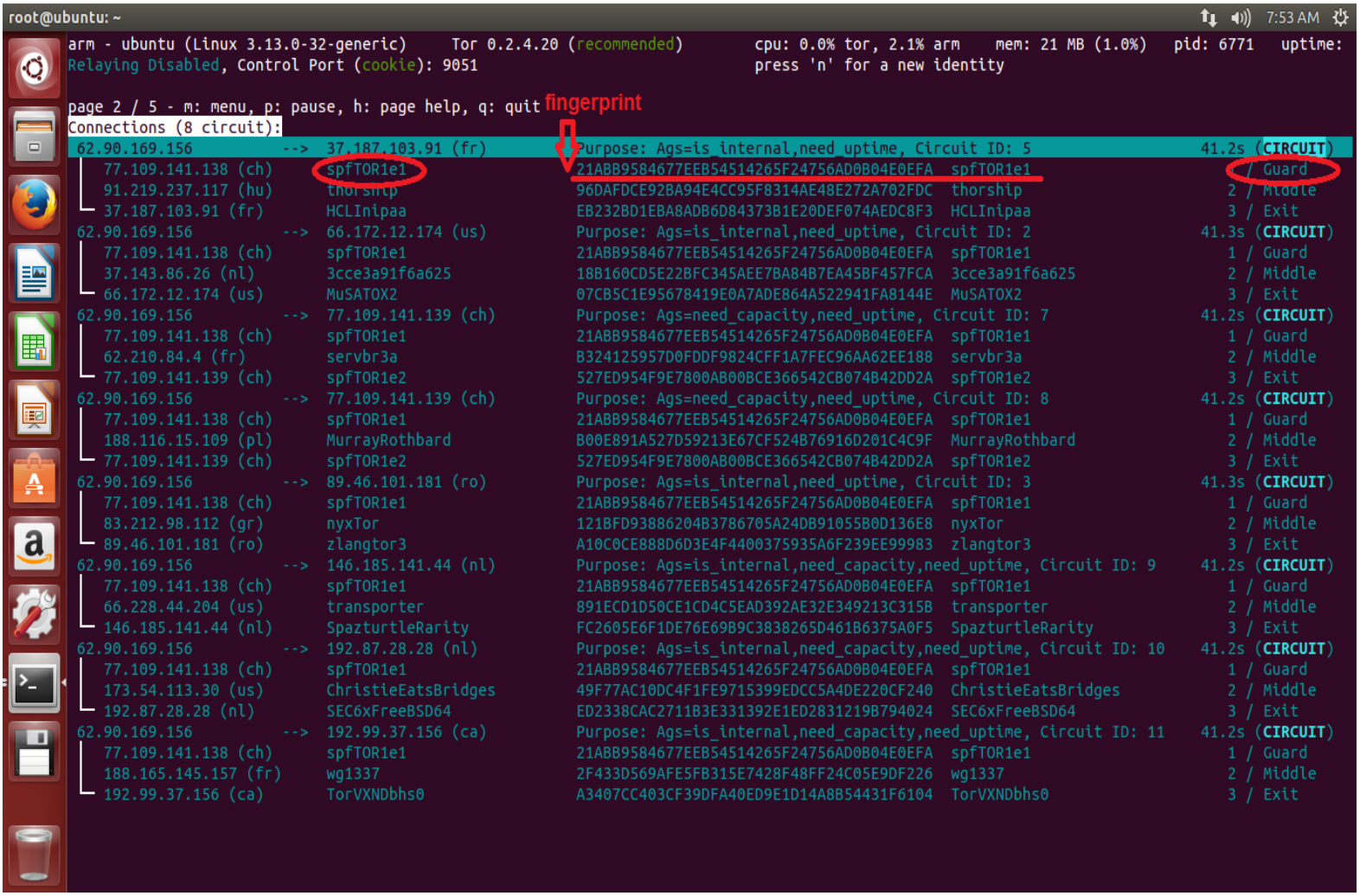
התקנת ARM ושימושו

השימוש הראשון נראה באיור 1 למעלה , יכול לזהות שגיאות בקובץ Torrc ומסמן את התחביר בצבעים.

תוכנת ARM באה לידי ביטוי בנוסף כאשר נרצה לראות דרך איזה צמתים תעבורת רשת TOR עוברת ב Guard,Middle,Exit NODE, ונרצה למשל להגדיר Guard node

ספציפי שיעבור דרכו תעבורת הרשת הראשונה מהלקוח זה יכול להתבצע על ידי הגדרה של הצומת בעזרת כינוי, טביעת אצבע או IP .

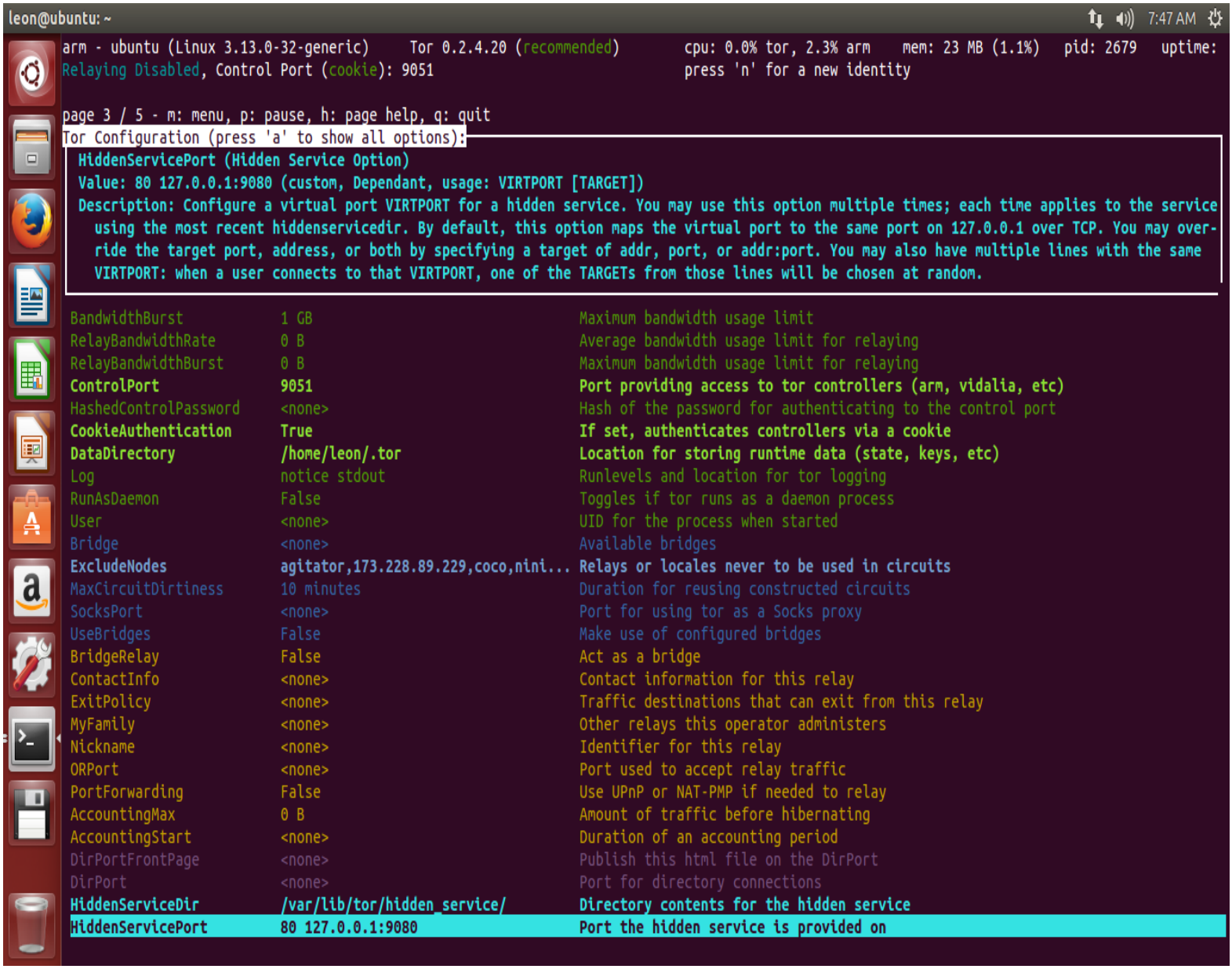
איור 2



מה שעוד שימושי מאוד בתוכנה הזאת שניתן לשנות את קובץ ה torrc מתוך התוכנית ולא דרך הTerminal ,

ניתן לראות בשתי השורות האחרונות את ההגדרה של ה Hidden Service ,בשלבים הבאים נעשה שימוש בנתונים האלו כדי לפתוח אתר ברשת Tor.

איור 3



בניית hidden Service

אחרי שהגדרנו בקובץ Torrc את האפשרות להקים hidden service אנחנו צריכים

להשיג את הכתובת האתר, הכתובת תהיה בקובץ hostname בתיקייה שהגדרנו

ב hiddenServiceDir. הכתבות שקיבלנו היא : **4n5cvihz5jqf4gm3.onion**

בכתובת זאת נשתמש לארוך כל הסימולציה , במקרה ואנו רוצים ליצור עוד אתרים ניתן ע"י הוספת שתי הפקודות שציינו עם כתובת שונה ונתיב לתיקייה אחרת.

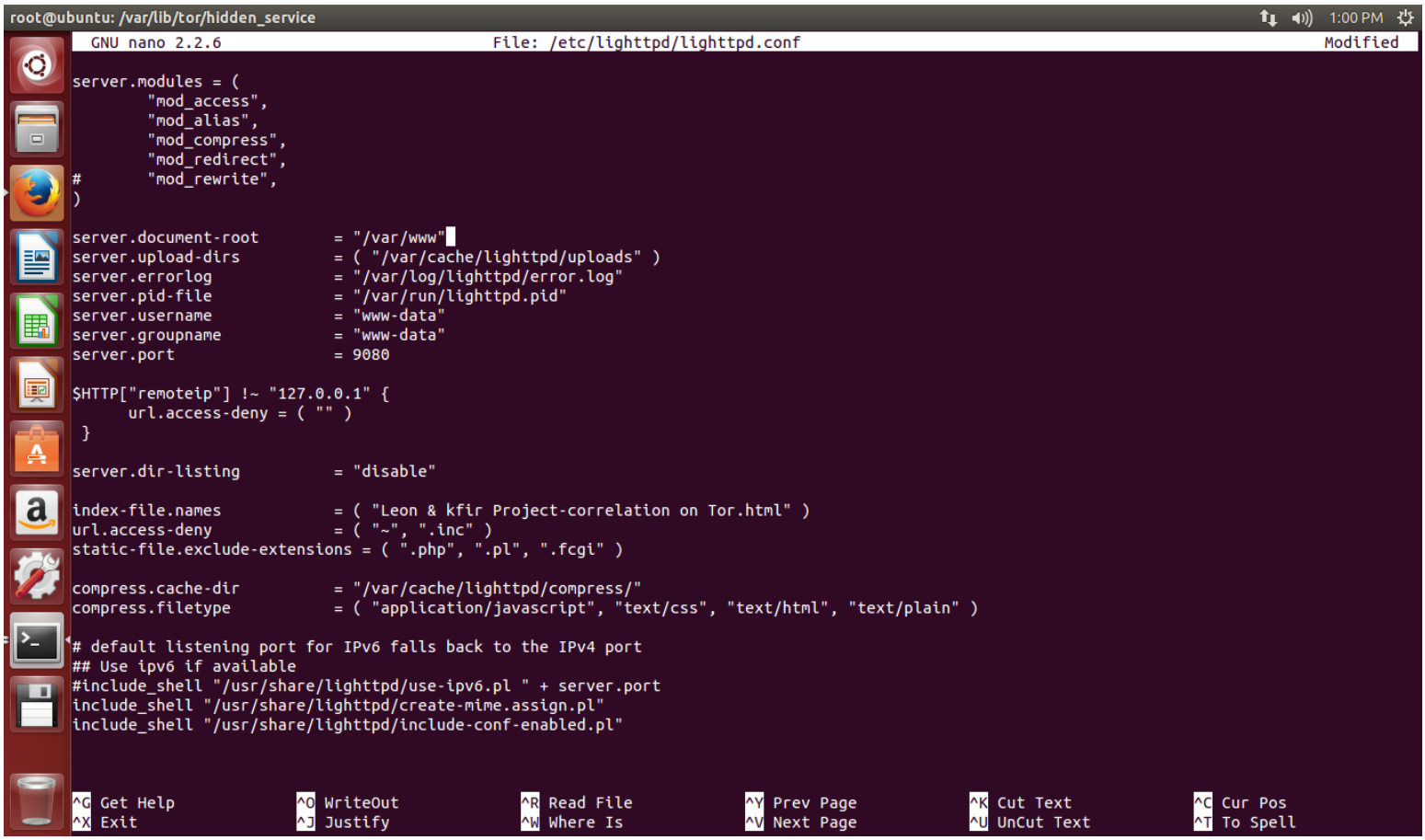
עכשיו שיש לנו את הכתובת אנחנו צריכים להקים אתר , בשביל זה יש צורך בשרת

שיעבוד על גבי רשת Tor ויאפשר לאנשים הגולשים ברשת Tor בלבד לגשת לאתר שהטמנו .

החלטנו להשתמש ב **lighttpd** במקום שרת **Apache HTTP Serve** שהוא השרת ה http הנפוץ בעולם משום שAPACHE דורש משאבים רבים ויחסית מסורבל להתקנה לעומת **lighttpd**.

לאחר התקנת ה Lighttpd היינו צריכים לשנות את הגדרות הקובץ lighttpd.conf

שנמצא בתיקייה /etc/lighttpd/. איור 4



יש צורך לעשות תיאום בין ההגדרות שעשינו בקובץ torrc לקובץ הנ"ל

1.הserverport שמוגדר 9080 כמו שמופיע באיור 3

2. Remoteip שמוגדר כ127.0.0.1 כמו באיור 3

עכשיו תקשורת השרת תעבור דרך רשת Tor ולא דרך הרשת הרגילה .

שלב הבא היא הטמעת אתר בתוך השרת , נתיב האתר יופיע ב

Server.Document-root -מופיע באיור 4.

ושם הקובץ בו נשתמש להעלאת האתר יופיע ב index-file.names -מופיע באיור 4.

היינו צריכים לבנות אתר בעל "משקל" כדי לקבל נתונים רבים לביצוע הקורלציה ,

שימוש כמו צ'ט או הודעות בודדות לא היה נותן לנו תוצאות , כי יש צורך לאסוף כ1000 פקטות כדי לבצע את הקורלציה.

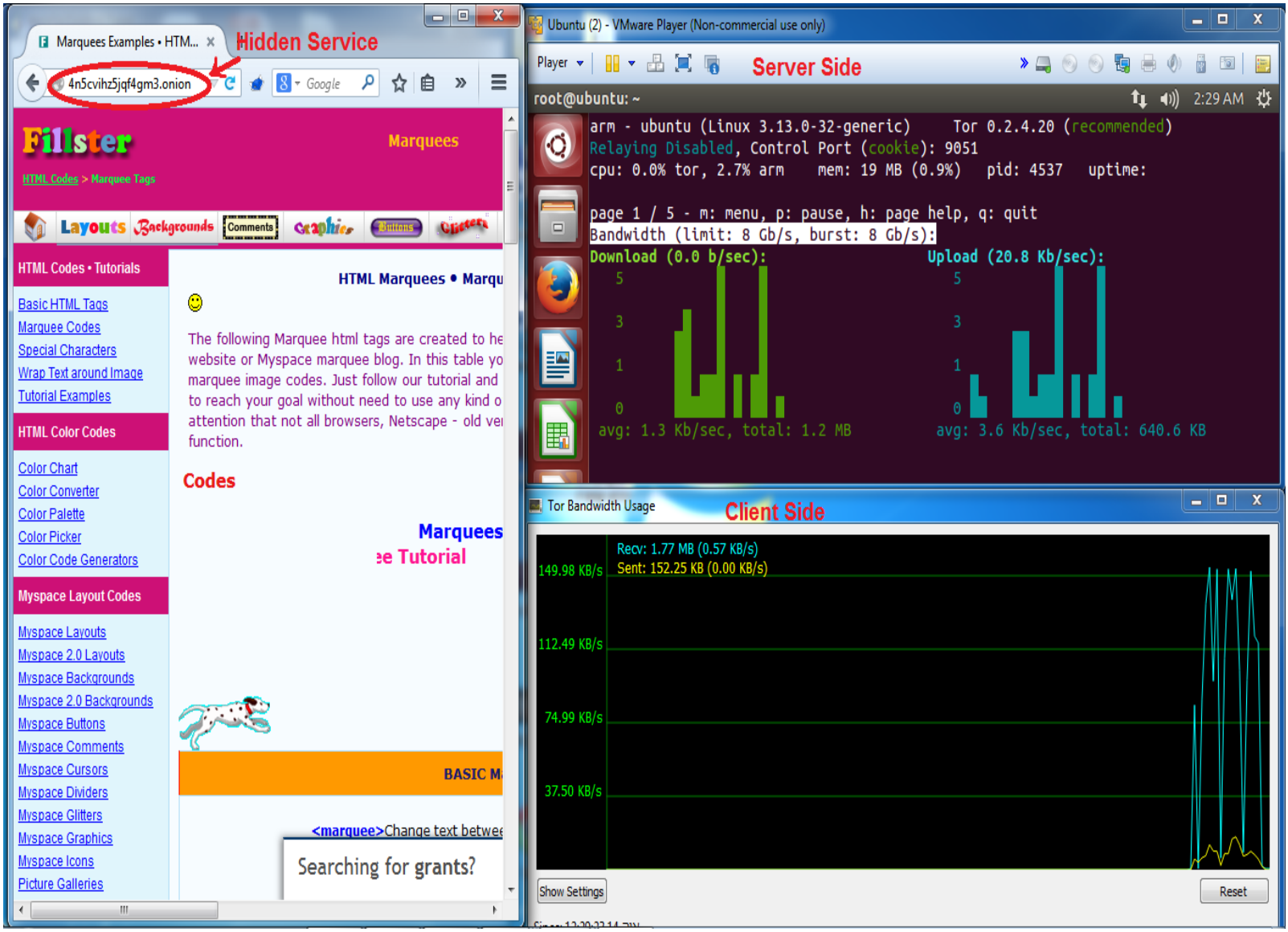
שימוש ב Vidalia בצד לקוח אפשר לנו לראות בצורה גרפית את זרימת הנתונים והחיבורים השונים ברשת Tor .

היה צורך להתקין תוספת ל Firefox בשם FoxyProxy כדי לגלוש ברשת Tor

בהגדרות היה צורך להגדיר את הHostname :port לפני מה שהגדרנו ב Vidalia

וב SockProxy לסמן את Socks V4/4A.

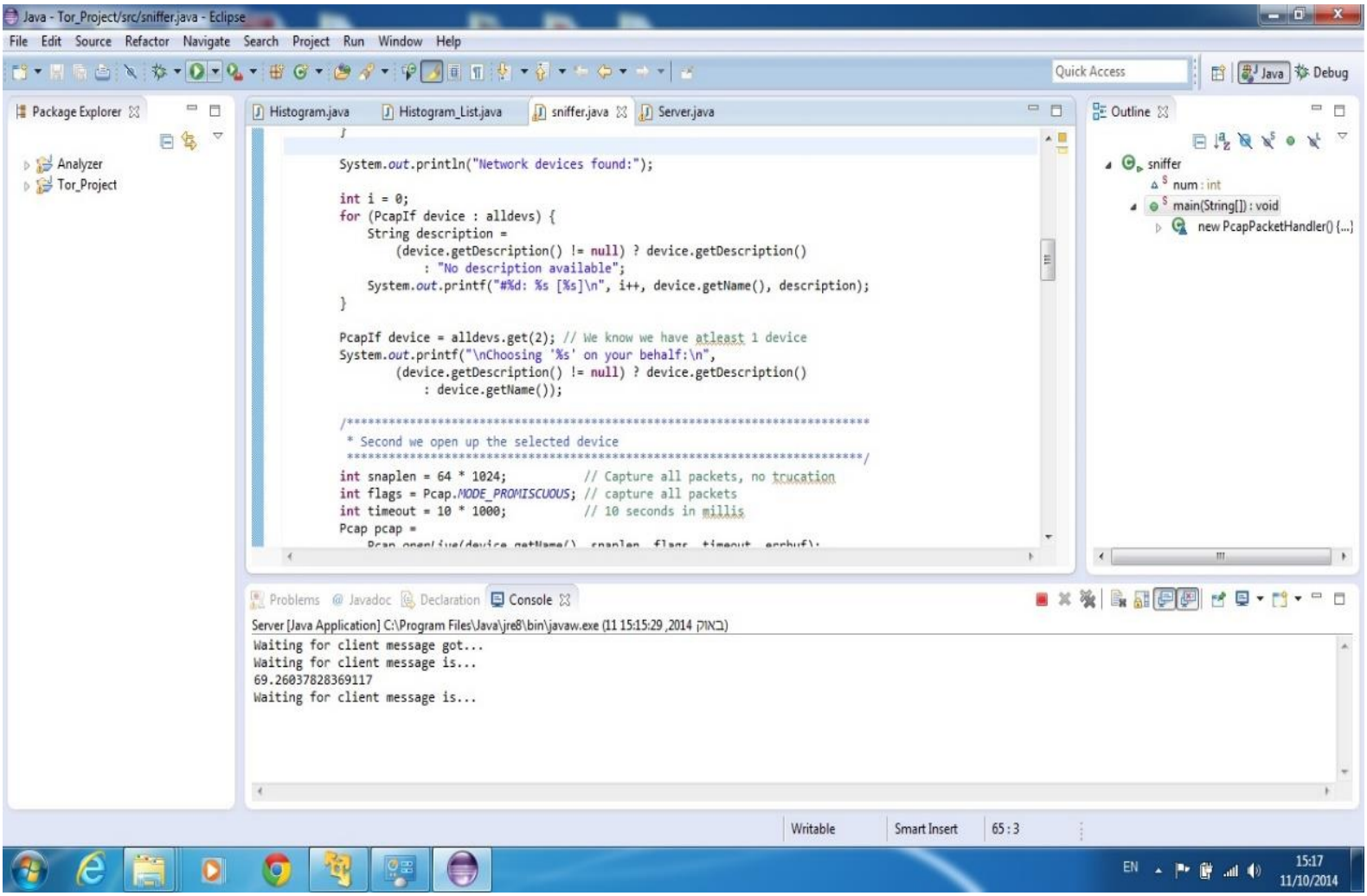
האתר בתמונה עם צד הלקוח וצד השרת :



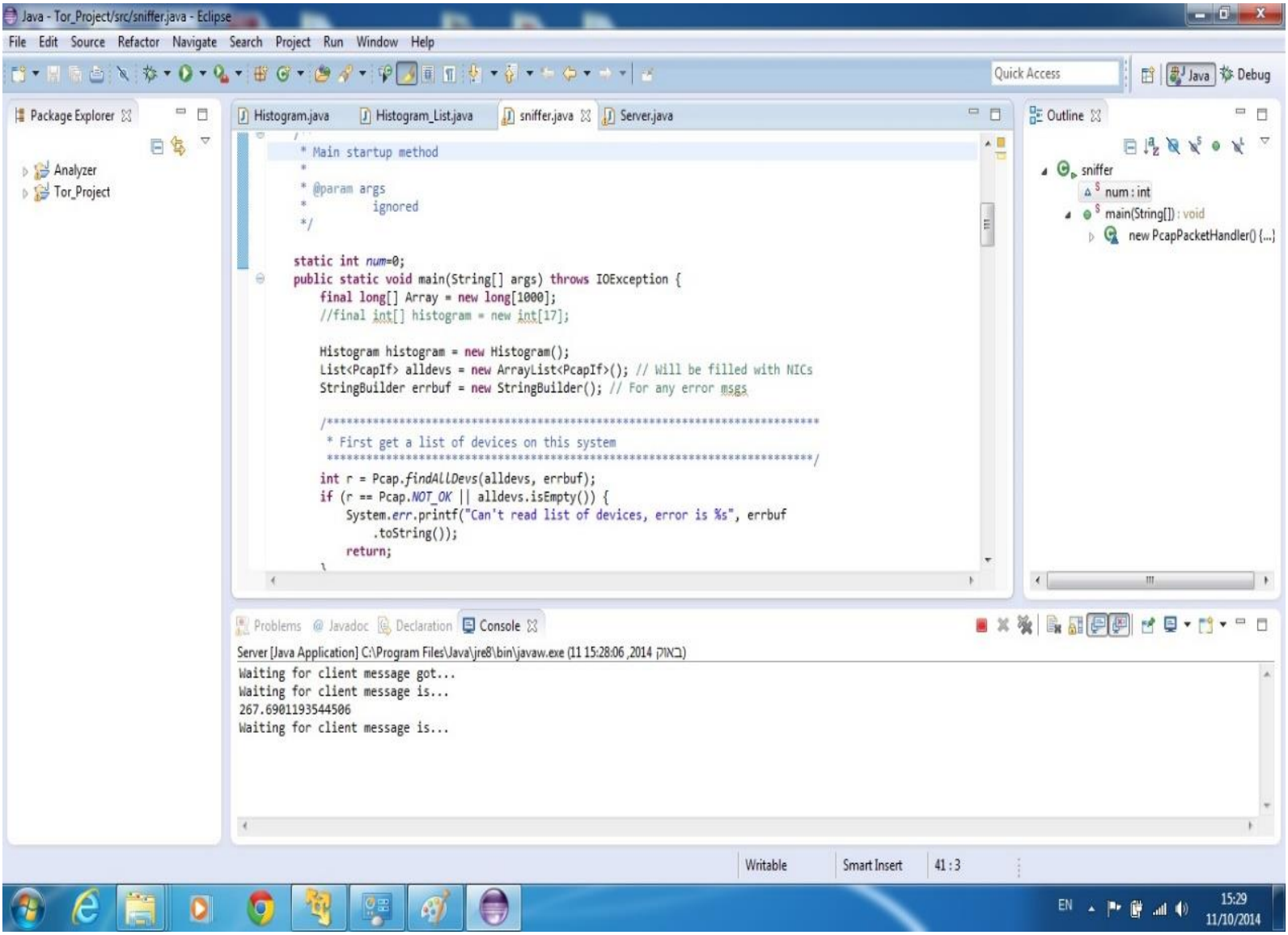
קורלציה בין משתמשי קצה ברשת Tor

תוצאות

1.התוצאות שהתקבלו לאחר תקשורת בין הלקוח והשרת כאשר מותקן עליהם הSniffer



2.התוצאות שהתקבלו לאחר תקשורת של הלקוח עם אתר אחר , ושליחת נתונים מהשרת ללקוח אחר שאנו לא מאזינים לנתונים שלו (בלי Sniffer)

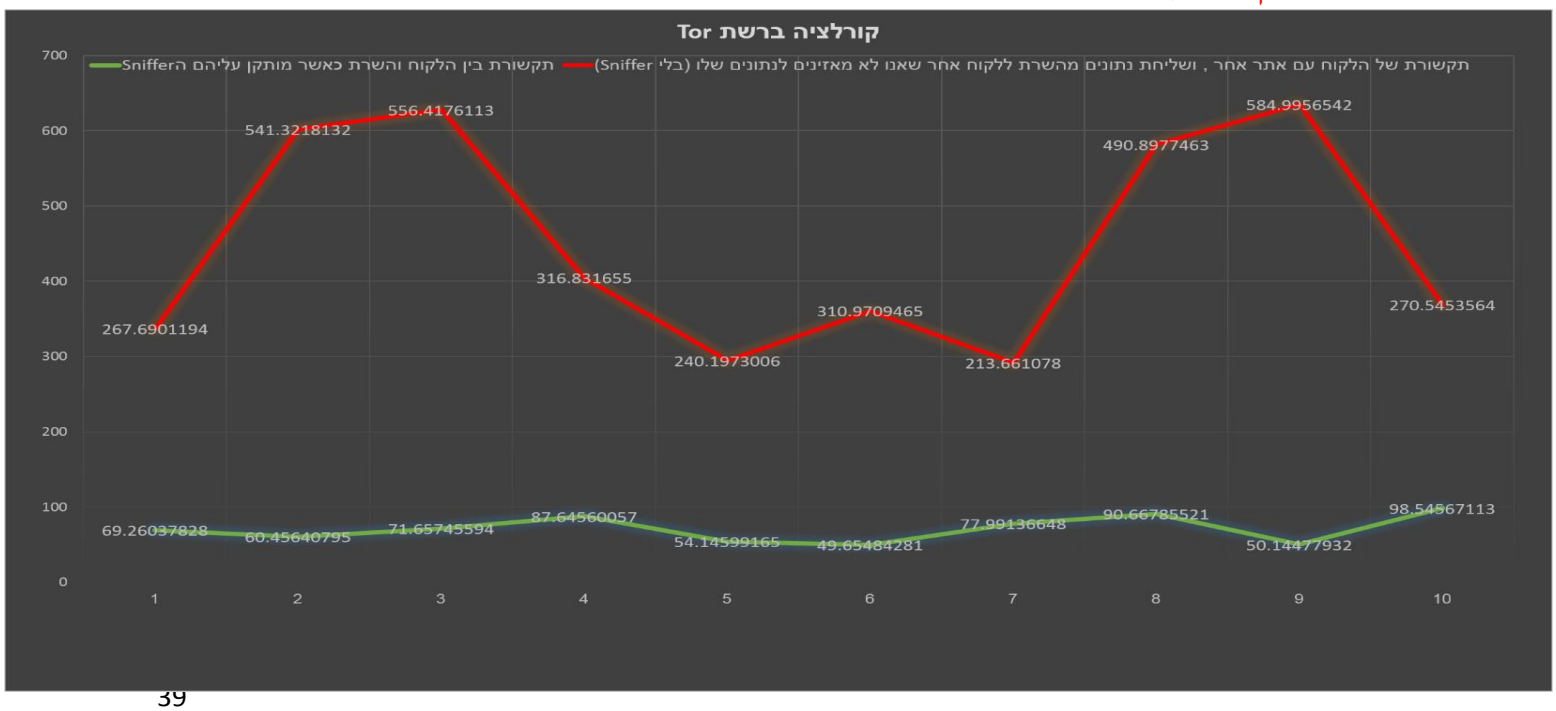


טבלת ניסיונות ותוצאות

אפשרות שניה	אפשרות ראשונה	ניסיון מספר
267.69011935	69.26037828	1
541.32181324	60.45640795	2
556.41761131	71.65745594	3
316.83165497	87.64560057	4
240.19730064	54.14599165	5
310.97094651	49.65484281	6
213.66107801	77.99136648	7
490.89774631	90.66785521	8
584.99565421	50.14477932	9
270.54535638	98.54567113	10

המרחקים האוקלידים מיוצגים בms

האפשרות הראשונה התקבלה לאחר תקשורת בין הלקוח והשרת כאשר מותקן על שניהם ה Sniffer האפשרות השנייה התקבלה לאחר תקשורת של הלקוח עם אתר אחר , ושליחת נתונים מהשרת ללקוח אחר שאנו לא מאזינים לנתונים שלו (בלי Sniffer), מה שאומר שאין תקשורת בין השרת והלקוח .
אנו יכולים לראות בבירור את ההפרשים בין האפשרות הראשונה והשנייה , והסבר לזה הוא שזמני ההשהיה בין פקטות באפשרות הראשונה יותר קצרות ,וכאשר חישבנו את המרחק האוקלידי בין 2 מערכי ההיסטוגרמות שהגיעו מהלקוח והשרת עליהם אנחנו מאזינים ומתקשרים ביניהם , התוצאות הראו מספרים קטנים מאלו של האפשרות השנייה כאשר לא היה תקשורת בין הלקוח והשרת. ניתן לראות בגרף באופן ברור את השוני בין האפשרויות



קורלציה בין משתמשי קצה ברשת Tor

סיכום

הפרויקט המרתק הזה גרם לנו להכיר ולהתמודד עם סביבות עבודה רבות, להתנסות במערכת תכנה מורכבת ורבת תהליכים,

נאלצנו להתמודד עם בניית מערכת Real Time שדורשת ניהול מדויק ונכון של מבני נתונים, תהליכים ותהליכונים. הצורך לעבוד במקביליות כדי לאפשר זרימת מידע בין מבני הנתונים והמודולים השונים, תוך מניעת deadlocks וstarvation של כל התהליכים גרם לנו לחקור את יכולותיה של שפת JAVA כשפת תכנות Object oriented וmulti-threads friendly.

התוודענו לעולם פרוטוקולי האינטרנט העיקריים כמו TCP/IP וגם פרוטוקולים של רשת TOR

וכיצד הם משתמשים בפרוטוקול TCP/IP ו"רוכבים" עליו.

למדנו כיצד להשתמש במערכות וחבילות תכנה רבות ושונות כגון RockSaw, JPcap, Java, Wireshark רשת TOR, ANALAYZER , יצירת מערכת אחת אשר מתפקדת בתנאי Real Time, להיעזר באינסוף מקורות המידע הקיימים ברשת, לנהל גרסאות קוד עדכניות ונגענו במושגים הכי עדכניים כרגע בענף אבטחת מידע והסייבר.

בנוסף נתבקשנו לבנות שרת שמתקשר ברשת Tor שבשרת זה נמצא אתר שמזרים נתונים בין המשתמש לשרת, למטרה היינו חייבים להעמיק את הידע עלhidden services.

בניית הסימולציה דרשה מאיתנו להכיר לעומק את מערכת ההפעלה LINUX מפקודות בסיסיות ועד שינוי הגדרות המערכת .

פרויקט זה בהחלט יעזור בנקודת פתיחה ועזרה לסטודנטים מהמכללה שירצו לחקור ולעשות פרויקטים בנושא Tor עם מיכאל אורלוב או מרצים אחרים.

מימושים עתידיים אפשריים

הטמעת המערכת ברשת הגדולה ע"י צבירת צמתים משלנו בשיטות שונות ,

המשך מחקר עם גורמים ביטחוניים שיש להם צורך לשבירת אנונימיות של גורם המסכן את טובת הציבור בצורה כלשהי.

ביבליוגרפיה

[1] M. Edman and P. Syverson. AS-Awareness in Tor Path Selection.
In ACM Conference on Computer and Communications Security (CCS), 2009.
[2] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor.
In IEEE Symposium on Security and Privacy (Oakland),2005.
[3] TorPS. TorPS: The Tor Path Simulator. http://torps.github.io, 2013.
[4] Tor Project, Inc. The Tor Project. https://www.torproject.org/, 2013.
[5] Tor Project, Inc. Tor Metrics Portal. https://metrics.torproject.org/, 2013.
[6] The Tor Project. Changelog Tor 0.2.4.12-alpha.https://gitweb.torproject.org/tor.git?a
[7] 0x539 Dev Group. Gobby: A Collaborative Text Editor.http://gobby.0x539.de, 2013.
[8] Paul Syverson and Rob Jansen .Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. U.S. Naval Research Laboratory, Washington DC.2013
[9] M. Akhoondi, C. Yu, and H. V. Madhyastha. LASTor: A Low-Latency AS-Aware Tor Client. In IEEE Symposium on Security and Privacy (Oakland), 2012.
[10] Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems, 2011
[11] באתר הארץ, גישת הבורר לשמירה על אנונימיות באינטרנט
[12]http://data.isoc.org.il/blog/429

[13]http://www.geektime.co.il/tor-part-3/

[14]http://www.binaryvision.org.il/?p=1002

[15] http://freehaven.net/anonbib/ , Selected Papers in Anonymity

[16] On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records by S. Chakravarty, M. V. Barbera, G. Portokalidis, M. Polychronakis, and A. D. Keromytis, March 2014.
[17] The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network by Rob Jansen, Florian Tschorsch, Aaron Johnson, and Björn Scheuermann. Proceedings of the Network and Distributed Security Symposium - February 2014.
[18] Challenges in protecting Tor hidden services from botnet abuse (PDF)
by Nicholas Hopper. In the Proceedings of Financial Cryptography and Data Security (FC'14), March 2014.

Tor- https://www.torproject.org/

Jpcap- https://github.com/jpcap/jpcap

Wireshark- https://www.wireshark.org

Rawsocket- https://www.npmjs.org/package/raw-socket

Ubuntu -http://releases.ubuntu.com/14.04

lighttpd-/http://www.lighttpd.net

Vidalia-https://www.torproject.org/projects/vidalia.html.en

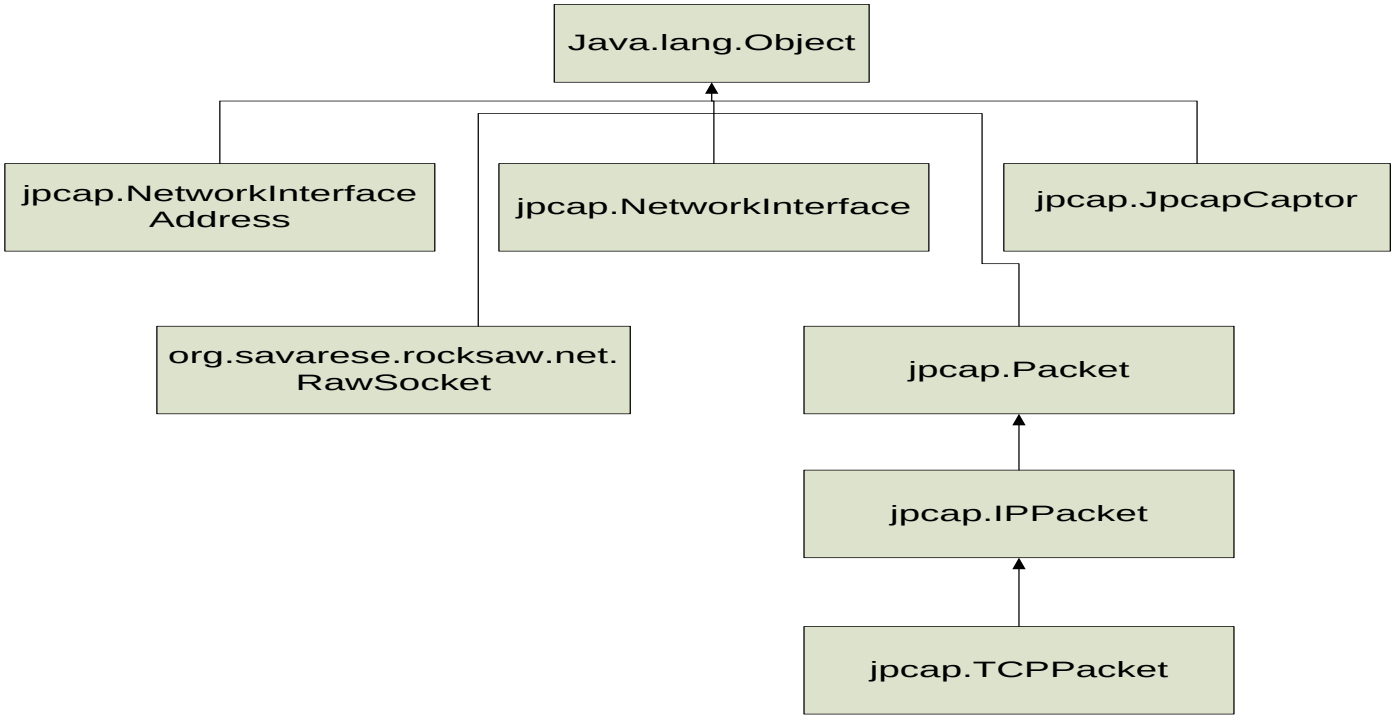
Eclipse- https://www.eclipse.org

Vmware- http://www.vmware.com

Ubuntu-/http://www.ubuntu.com

נספחים:

JPCAP



1. Go to the Jpcap download section and put the files, jpcap-0.6.zip and JPcapSetup-0.6.exe in a folder. Unzip and execute them.
2. Copy the Jpcap.dll file to the C:\Windows\System32 folder
3. Start Eclipse, make a new project, make new package, make a new class, write your code.
4. Right click on the project name --> Build Path --> Configure Build Path ... --> Java Build Path --> Add External JARs...
5. Add the jpcap.jar file.
6. Run your program.

Wireshark

capinfos - Prints information about capture files

dftest - Shows display filter byte-code, for debugging dfilter routines.

dumpcap - Dump network traffic

editcap - Edit and/or translate the format of capture files

idl2wrs - CORBA IDL to Wireshark Plugin Generator

mergecap - Merges two or more capture files into one

randpkt - Random Packet Generator

rawshark - Dump and analyze raw pcap data

reordercap - Reorder input file by timestamp into output file

text2pcap - Generate a capture file from an ASCII hexdump of packets

tshark - Dump and analyze network traffic

wireshark-filter - Wireshark filter syntax and reference

wireshark - Interactively dump and analyze network traffic

The following man page is part of the libpcap distribution and hosted here as a convenience for our users.

pcap-filter - Capture filter syntax

raw-socket

This module implements raw sockets for Node.js.

This module has been created primarily to facilitate implementation of the net-ping module.

This module is installed using node package manager (npm):

```
# This module contains C++ source code which will be compiled
# during installation using node-gyp.  A suitable build chain
# must be configured before installation.

npm install raw-socket
```

It is loaded using the require() function:

```
var raw = require ("raw-socket");
```

Raw sockets can then be created, and data sent using Node.js Buffer objects:

```
var socket = raw.createSocket ({protocol: raw.Protocol.None});
```

```
socket.on ("message", function (buffer, source) {
   console.log ("received " + buffer.length + " bytes from " + source);
});

socket.send (buffer, 0, buffer.length, "1.1.1.1", function (error, bytes) {
   if (error)
      console.log (error.toString ());
});
```

Lighttpd

Get the source You can either use a release from http://www.lighttpd.net/download or compile from svn:
Using a release/snapshot:
Extract the tar ball and enter the new directory:
tar -xf lighttpd-1.4.XXX.tar.gz
cd lighttpd-1.4.XXX
svn (you will need autoconf and automake for this):
First time:
svn checkout svn://svn.lighttpd.net/lighttpd/branches/lighttpd-1.4.x/
cd lighttpd-1.4.x
./autogen.sh

Next time from lighttpd-1.4.x/:
svn update
./autogen.sh
lighttpd 1.5 (development branch)
lighttpd 1.5 is not released yet, but you may checkout the source from:

svn checkout svn://svn.lighttpd.net/lighttpd/trunk/
cd trunk
./autogen.sh
See DevelSubversion and Devel.

Install dependencies
Depending on which features you want, you need other libraries; you will want at least libpcre and zlib, for more see OptionalLibraries.
On most systems you need to install the development version of the library packages, the library itself won't be enough!

On debian you can also use apt-get to install all build dependencies:

apt-get build-dep lighttpd
Here is the list used for the debian packages (without the packaging parts): libssl-dev, zlib1g-dev, libbz2-dev, libattr1-dev, libpcre3-dev, libmysqlclient15-dev, libfam-dev, libldap2-dev, libfcgi-dev, libgdbm-dev, libmemcache-dev, liblua5.1-0-dev, pkg-config, uuid-dev, libsqlite3-dev, libxml2-dev, libkrb5-dev.

Configure
Now you have to use the ./configure script - there is a help option for it:

./configure --help
Don't forget to set the --prefix if you don't want to install in /usr/local.

Build
If the configure step was successful, you can now build it:

make
Install
After a successful build you may install the package. This is not needed, but you will have to give lighttpd the correct location of the modules if you don't (see ./lighttpd -h).
As you hopefully didn't use root to build the package, you probably have now to get root access:

su make install
Init script
The init-script that is shipped with the source-tarball is meant to be run on a LSB conforming platform like SuSE, Fedora or Debian.
For 1.4.x versions from r2742/1.4.27 onwords , use doc/initscripts/ instead of doc/ in the commands below.

sed -e 's/FOO/lighttpd/g' doc/initscripts/rc.lighttpd > /etc/init.d/lighttpd
chmod a+rx /etc/init.d/lighttpd
cp -p doc/initscripts/sysconfig.lighttpd /etc/sysconfig/lighttpd
mkdir -p /etc/lighttpd

< 1.4.27
install -Dp ./doc/lighttpd.conf /etc/lighttpd/lighttpd.conf

> 1.4.28
cp -R doc/config/conf.d/ doc/config/*.conf doc/config/vhosts.d/ /etc/lighttpd/

chkconfig lighttpd on
If you're running CentOS or RHEL you might need to change the first line to this

sed -e 's/FOO/lighttpd/g' doc/initscripts/rc.lighttpd.redhat > /etc/init.d/lighttpd
In Debian / Ubuntu you use update-rc.d rather than chkconfig:

update-rc.d lighttpd defaults
Supervise
As an alternative to init scripts you can setup a "supervised" lighttpd with daemontools or runit, see LighttpdUnderSupervise Get the source
You can either use a release from http://www.lighttpd.net/download or compile from svn:

- Using a release/snapshot:
  Extract the tar ball and enter the new directory:

- tar -xf lighttpd-1.4.XXX.tar.gz
- cd lighttpd-1.4.XXX
- svn (you will need autoconf and automake for this):
  First time:

- svn checkout svn://svn.lighttpd.net/lighttpd/branches/lighttpd-1.4.x/
- cd lighttpd-1.4.x
- ./autogen.sh

Next time from lighttpd-1.4.x/:

svn update
./autogen.sh
lighttpd 1.5 (development branch)

lighttpd 1.5 is not released yet, but you may checkout the source from:
svn checkout svn://svn.lighttpd.net/lighttpd/trunk/
cd trunk
./autogen.sh
See DevelSubversion and Devel.
Install dependencies
Depending on which features you want, you need other libraries; you will want at least libpcre and zlib, for more see OptionalLibraries.
On most systems you need to install the development version of the library packages, the library itself won't be enough!
On debian you can also use apt-get to install all build dependencies:
apt-get build-dep lighttpd
Here is the list used for the debian packages (without the packaging parts): libssl-dev, zlib1g-dev, libbz2-dev, libattr1-dev, libpcre3-dev, libmysqlclient15-dev, libfam-dev, libldap2-dev, libfcgi-dev, libgdbm-dev, libmemcache-dev, liblua5.1-0-dev, pkg-config, uuid-dev, libsqlite3-dev, libxml2-dev, libkrb5-dev.
Configure
Now you have to use the ./configure script - there is a help option for it:
./configure --help
Don't forget to set the --prefix if you don't want to install in /usr/local.
Build
If the configure step was successful, you can now build it:
make
Install
After a successful build you may install the package. This is not needed, but you will have to give lighttpd the correct location of the modules if you don't (see ./lighttpd -h).
As you hopefully didn't use root to build the package, you probably have now to get root access:
su make install
Init script
The init-script that is shipped with the source-tarball is meant to be run on a LSB conforming platform like SuSE, Fedora or Debian.
For 1.4.x versions from r2742/1.4.27 onwords , use doc/initscripts/ instead of doc/ in the commands below.
sed -e 's/FOO/lighttpd/g' doc/initscripts/rc.lighttpd > /etc/init.d/lighttpd
chmod a+rx /etc/init.d/lighttpd
cp -p doc/initscripts/sysconfig.lighttpd /etc/sysconfig/lighttpd
mkdir -p /etc/lighttpd

< 1.4.27
install -Dp ./doc/lighttpd.conf /etc/lighttpd/lighttpd.conf

> 1.4.28
cp -R doc/config/conf.d/ doc/config/*.conf doc/config/vhosts.d/ /etc/lighttpd/

chkconfig lighttpd on
If you're running CentOS or RHEL you might need to change the first line to this
sed -e 's/FOO/lighttpd/g' doc/initscripts/rc.lighttpd.redhat > /etc/init.d/lighttpd
In Debian / Ubuntu you use update-rc.d rather than chkconfig:
update-rc.d lighttpd defaults
Supervise
As an alternative to init scripts you can setup a "supervised" lighttpd with daemontools or runit, see LighttpdUnderSupervise

5 minutes
Want to run a fast, low-resource server for static content? It's easy. Create a text file named lighttpd.conf with the following content:

```
server.document-root = "/var/www/servers/www.example.org/pages/"

server.port = 3000

mimetype.assign = (
  ".html" => "text/html",
  ".txt" => "text/plain",
  ".jpg" => "image/jpeg",
  ".png" => "image/png"
)
```

lighttpd will listen on TCP port 3000 and bind to all interfaces by default. The few important MIME types are assigned
and the document root (the base directory that is used for all requests) is set. The files in the document root have to be
readable by the user starting the web server.

First, check that your configuration is ok:

```
$ lighttpd -t -f lighttpd.conf
```

Now start the server for testing:

```
$ lighttpd -D -f lighttpd.conf
```

and point your browser to http://127.0.0.1:3000/

To stop the server, return to the command prompt and press ctrl-c.

A real daemon
Next you should familiarize yourself with some settings necessary for your server's security:

```
server.document-root = "/var/www/servers/www.example.org/pages/"

server.port = 80

server.username = "www"
server.groupname = "www"

mimetype.assign = (
  ".html" => "text/html",
  ".txt" => "text/plain",
  ".jpg" => "image/jpeg",
  ".png" => "image/png"
)

static-file.exclude-extensions = ( ".fcgi", ".php", ".rb", "~", ".inc" )
index-file.names = ( "index.html" )
```

Now the web server is listening on port 80, the default for HTTP traffic, and will switch to the user www and the group www.
The server has to be started as root to take control of port 80, but it's not necessary or a good idea to continue running as root after port acquisition, so the server switches to user www.

Lastly, access to view the contents of some types of files is forbidden as they are used for generating dynamic content. Requests directly to a directory are rewritten to the index.html file in that directory.

Assuming you have already created the /etc/init.d/lighttpd service as described in TutorialInstallation, place the config file in /etc/lighttpd/lighttpd.conf and start the server with:

/etc/init.d/lighttpd start
To stop it use:

/etc/init.d/lighttpd stop
10 minutes
Conditionals, conditionals, conditionals:

The most important part in Lighty's configuration is the use of conditionals. Using simple or regular expression conditions, default setting can be overridden.

```
server.document-root = "/var/www/servers/www.example.org/pages/"

server.port = 80

server.username = "www"
server.groupname = "www"

mimetype.assign = (
  ".html" => "text/html",
  ".txt" => "text/plain",
  ".jpg" => "image/jpeg",
  ".png" => "image/png"
)

static-file.exclude-extensions = ( ".fcgi", ".php", ".rb", "~", ".inc" )
index-file.names = ( "index.html" )

$HTTP["host"] == "www2.example.org" {
  server.document-root = "/var/www/servers/www2.example.org/pages/"
}
```

Now we have a new virtual server, www2.example.org, which uses the same settings as the first server, only the document root is different.

The following server configuration adds a download area and enables the built-in directory listing feature:

```
server.document-root = "/var/www/servers/www.example.org/pages/"

server.port = 80

server.username = "www"
server.groupname = "www"

mimetype.assign = (
  ".html" => "text/html",
  ".txt" => "text/plain",
  ".jpg" => "image/jpeg",
  ".png" => "image/png"
)

static-file.exclude-extensions = ( ".fcgi", ".php", ".rb", "~", ".inc" )
index-file.names = ( "index.html" )

$HTTP["host"] == "www2.example.org" {
  server.document-root = "/var/www/servers/www2.example.org/pages/"
  $HTTP["url"] =~ "^/download/" {
```

```
    dir-listing.activate = "enable"
  }
}
```

As you can see, conditionals can be nested: only the download folder and its sub-folders have the directory listings enabled.

There's also the else clause for conditionals. Despite the name, it's an else if construct similar to some programming languages, as it has to be followed by another condition.

Here's an example of conditional-based vhosts. The else is being used to configure behavior that should be present only in "default" vhost.

```
$HTTP["host"] == "example.org" {
  # options specific to example.org
  expire.url = ( "" => "access plus 25 hours" )
} else $HTTP["host"] == "static.example.org" {
  # options specific to static.example.org
  expire.url = ( "" => "access plus 2 weeks" )
} else $HTTP["host"] =~ "" {
  # options applied to any other vhosts present on this ip
  # ie. default options
  expire.url = ( "" => "access plus 3 hours" )
}
```

Now that we've covered the basics, you're ready to learn some more advanced topics like includes and configuring PHP with FastCGI

After 30 minutes
Now you know the basic set up, include files, and maybe even how to set up PHP or Ruby. But there's so much more to find out.

Most of it is described in examples in the default configuration file that you can find in the doc directory of the tarball or in the repository (source:trunk/doc/lighttpd.conf).

For the configuration file syntax and the complete list of configuration options, visit the reference section of the wiki: Reference Documentation

Also check out our community.

lighttpd.conf Magnifier (11.2 KB)

Tor

Introduction
Tor or The Onion Router is a toolset used to help anonymize your traffic. From the Tor website:

Tor is a toolset for a wide range of organizations and people that want to improve their safety and security on the Internet. Using Tor can help you anonymize web browsing and publishing, instant messaging, IRC, SSH, and other applications that use the TCP protocol. Tor also provides a platform on which software developers can build new applications with built-in anonymity, safety, and privacy features.
This guide is an adaptation of the official Tor installation method. It has been changed to reflect installation methods unique to Ubuntu, but may easily be used as a guide for other Debian based distros.

Installing Tor
Install Tor by issuing the following command or use System --> Administration --> Synaptic Package Manager:

sudo apt-get install tor
If your version of Ubuntu is not the latest, or does not have Tor, you may prefer to install from the Tor project's own repository: see Tor installation documentation for the latest instructions.

Starting Services and Checking Status

sudo /etc/init.d/tor start
Check that the Tor service is running on port 9050:

ss -aln | grep 9050
You should see either or both lines of the following output:

0 0 :::950 :::*
0 0 *:950 *:*
(Potential Issue on 8.04 =

Some of you following this will notice that there are no scripts in the /etc/init.d folder. This won't help you, but just lets you know it may not be there. Also attorproject manually installing with their deb package. Same thing. It isn't you.

sudo /etc/init.d/tor start
)

You can test if Tor is working by connecting to this web page: https://check.torproject.org

Install Vidalia (Optional)

Vidalia is a controlling Graphical User Interface for Tor. Tor must be installed for Vidalia to work. Once you have installed Tor and Vidalia you can configure client and relay settings through Vidalia. To install, type the following command in a terminal:

sudo apt-get install vidalia
Anonymizing Applications
Here are some common applications people configure for use with Tor on Ubuntu. The Tor FAQ lists more supported applications.

Mozilla Firefox

The Tor project recommends that you download their patched version of Firefox, the Tor Browser Bundle. If you prefer, you can use the Torbutton Firefox extension, but the Tor developers say they can no longer guarantee to keep up with the pace of Firefox development.

Note 1: You will notice that browsing through Tor is slower - this is normal due to the extra hops and encryption required.

Note 2: With older version of Ubuntu and Firefox, it may be necessary to first disable the proxy settings in Firefox's native options menu when using Torbutton; otherwise Torbutton will be unable to disable Tor.

FoxyProxy

Under Proxy Options select proxy type SOCKS v4
Enter 127.0.0.1 for the host
Enter 9050 for the port
Leave user/pass blank

VMware

Before creating a virtual machine, you must obtain the operating system and any necessary product keys for installation in that virtual machine. VMware Fusion does not come with any operating systems to install in virtual machines you create.
This method assumes that you are using a physical CD or a disk image (.iso / .cdr /.dmg file). You cannot create a Windows virtual machine by using .exe files downloaded from Microsoft, as those files need to be run on a Windows PC.
To create a new Windows virtual machine using the Easy Install method:

In Fusion, go to File > New. The New Virtual Machine Assistant launches.
If you have a physical CD/DVD of Windows, proceed to step 4.
If you are using an .iso/.cdr /.dmg image file:

Nautilus

Nautilus is a user authentication and management service.
sudo add-apt-repository ppa:gnome3-team/gnome3
sudo apt-get update && sudo apt-get install nautilus

Work In Progress
Please note, this is an on-going, incomplete project! While the core functionality should remain stable, some important features are not yet implemented and design is subject to change.

Setup
Currently Nautilus requires Riak 1.4 as its persistence layer. Ensure Riak is installed and running. By default, Nautilus looks for Riak on `localhost:8087`.

Usage
Nautilus is a standalone service which provides pre-authenticated, transient proxies to privileged backend services. For instance, a client application can use Nautilus to obtain a portal which maps to a user's profile page. Because portals are pre-authenticated, there is no need for an application to check credentials against the database on every request. After a set period of time, a portal expires and consuming applications may respond by transparently requesting a new portal.

A complete flow, from user creation through portal creation works like this:

1. A user is created via the user creation endpoint
2. An OAuth 2.0 Bearer Token is created on behalf of the user
3. A request for a portal against an existing backend service is made
4. Using the portal ID obtained in step 3, a proxied request is made

To clarify how this might work with a real client application, let's walk through the above flow using curl. (Assume we have an instance of Nautilus running on localhost port 3000.

```sh
$ curl -X POST http://localhost:3000/user \
      -d '{"email": "foo@bar.tld", "password": "hunter2"}' \
      -H 'Content-Type: application/json'

{}
```