**מסמך איפיון למיני פרוייקט - נושאים במערכות הגנה לרשת**

מודול לשרת פרוקסי לחסימת קבצים עוינים

מגישים:

אברהם נתן – ת.ז 317306736

ארז כרמל – ת.ז 302491600

ספיר גאלי – ת.ז 307907436

**מסמך אפיון – מיני פרוייקט בנושא מערכות הגנה ברשת**

במסמך זה יינתן תיאור מפורט של הפרוייקט – מודול שמהווה תוספת לשרת אפאצ'י 2 במוד פרוקסי לצורך הגנה מפני קבצים עוינים.

המסמך יציג עקרונות מנחים, אופן פעולת המערכת, התקנה וקינפוג ודוגמאות לפעולת המערכת.

**מבוא**

בארגונים רבים כיום קיימת רשת ארוגנית שעל בסיסה מתבצעת תעבורת המידע ואחסון המידע בתוך הארגון וגם כלפי חוץ. למרות שמבחינת שמירה על בטיחות המידע, הפתרון האידיאלי הוא ניתוק לחלוטין של הרשת הארגונית מרשת האינטרנט, על מנת שהארגון יוכל להתנהל בצורה מיטבית עליו לאפשר חיבור לעולם החיצון. עצם חיבור זה יוצר נקודת תורפה ופותחת פתח לנוזקות אפשריות למצוא את דרכן לתוך הרשת הארגונית.

לצורך מתן מענה לסכנות כאלה יש צורך בבקרה על תעבורת המידע שמגיע אל או מתוך הרשת בכלל זה סינון קבצים שניתן להוריד מהרשת האינטרנט לרשת הארגונית, ובכך למנוע הורדה של קבצים עוינים, ניתורם, והתראות על ביצוע ניסיונות כאלה.

**אופן פעולה:**

אפאצ'י 2 הינו שרת ורסטילי וסקלבילי שמשתמש בכלים הנקראים מודולים לצורך הגדרת תצורות עבודה שונות. מודולים אלה הינם קבצים מהצורה some\_module.so המגיעים עם החבילה של אפאצ'י, שניתן לטעון לתוך השרת לפני או בזמן עבודתו, ומופעלים ברגע שהשרת הופעל או מאותחל מחדש. בנוסף למודולים שמגיעים עם החבילה של אפאצ'י, ישנה אפשרות ליצור מודולים מותאמים אישית ולטעון אותם לשרת. זה האופן שבו מימשנו את הכלי.

ניטור:

הכלי יעבוד עם רשימת שמות הקבצים שמוגדרים כעוינים כפי שמופיעים ברשת, וישווה כל בקשת הורדה עם רשימה זו. במידה והקובץ שבבקשת ההורדה תואם ל"רשימה שחורה" זו, תיחסם ההורדה.

תגובה:

במקרה של תפיסת בקשה להורדה של קובץ עוין המודול יתריע זאת בפני המשתמש, יפנה אותו לאתר בטוח, ויבצע תיעוד של המקרה.

**התקנה וקינפוג:**

דרישות קדם:

1. מערכת לינוקס שעליה מותקן אפאצ'י 2.
2. הכלי apxs מותקן על המערכת.

התקנה:

פקודות ההתקנה מפורטות בקובץ Installation Notes.txt ובנוסף מצורף הקובץ installer.sh אשר ניתן להרצה דרך הcommand line ומבצע את כל ההתקנות הנדרשות באופן אוטומטי.

קינפוג:

בדפדפן יש להגדיר את תצורות הגישה של שרת ה – Proxy להגדרות proxy ידניות, יש להגדיר את הip של שרת הפרוקסי וport 8080, עבור הפרוטוקולים http, ftp, ssl.

בשרת הפרוקסי יש להגדיר את השמות של הקבצים העוינים – בקובץ /var/www/html/BlackList.txt יש להזין את שמות הקבצים שנרצה לחסום הורדה שלהם.

**מימוש התוכנית:**

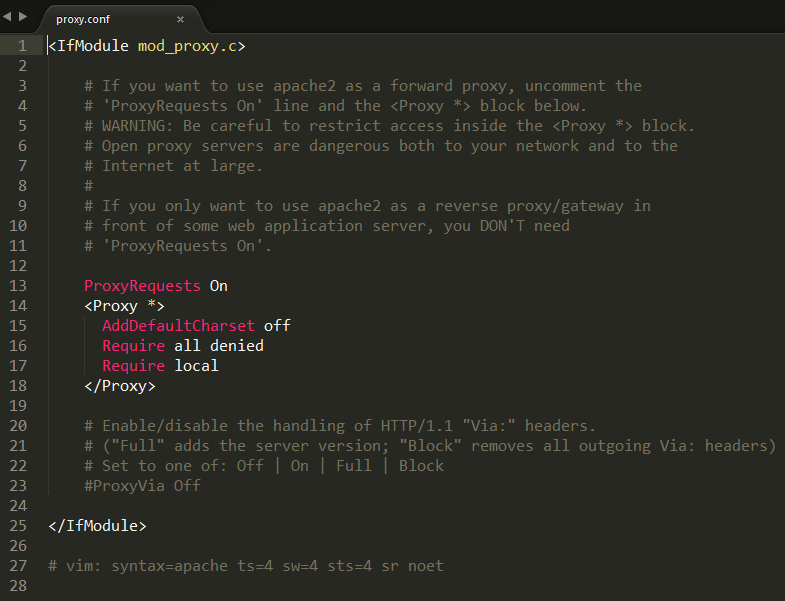
הקובץ המרכזי נכתב בשפת C ומכיל את הקוד של המודול ונוספים אליו 6 קבצי הגדרות:

* proxy.conf
* ports.conf
* forward\_proxy.conf
* virus\_block.conf
* mod\_virus\_block.c
* BlackList.txt
* virus\_block\_log.txt

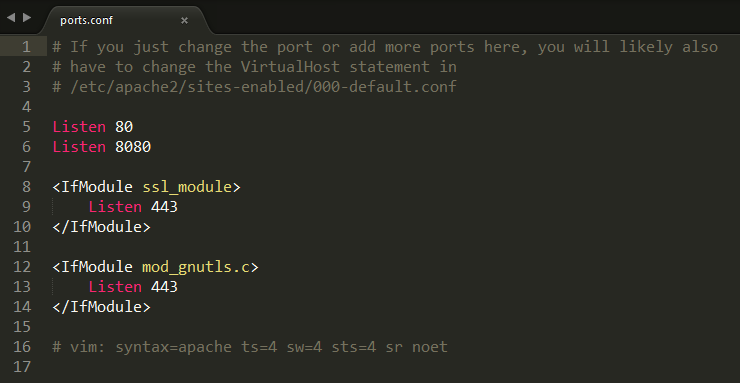
נפרט כעת על כל אחד מהקבצים.

**proxy.conf:**

קובץ הקונפיגורציות שמאפשר את מודול הפרוקסי של אפאצ'י. ברירת המחדל בקובץ זה הוא לא להעביר אף בקשה דרך הפרוקסי.

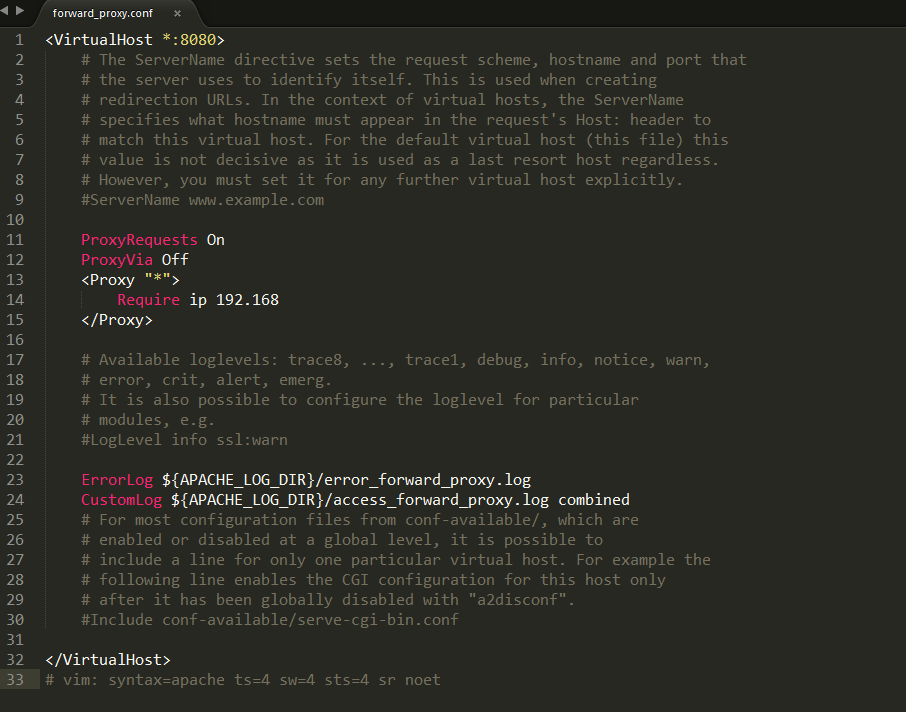


**ports.conf:**

בקובץ זה מוגדר לשרת הפרוקסי להקשיב לפורט 8080.

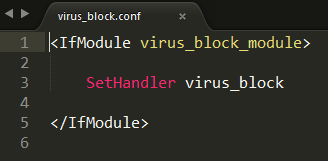
**forward\_proxy.conf:**

קובץ הגדרות למודול הפרוקסי עבור תצורת עבודה של forward\_proxy – פרוקסי שמסתיר את הרשת הארגונית מהרשת הגלובלית (local proxy). מקובץ זה מוגדר שכל url שמקורו ברשת הארגונית (ip שמתחיל ב 192.168) יעבור דרך שרת הפרוקסי. מוגדר בשורה 1 שמודול פרוקסי זה יקשיב לפורט 8080.

כמו כן מוגדרים תיעודי בקשות שעוברים דרך הפרוקסי (logging).

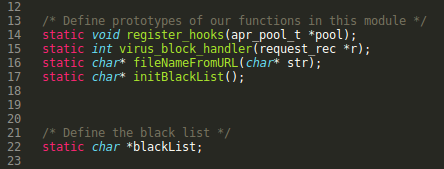
**virus\_block.conf:**

קובץ הקונפיגורציות עבור הכלי שמסנן את הקבצים. בקובץ זה מוגדר לאפאצ'י לשלוח בקשות שעוברות דרכו ל handler של הכלי (virus\_block\_module) במידה וכלי זה מופעל.

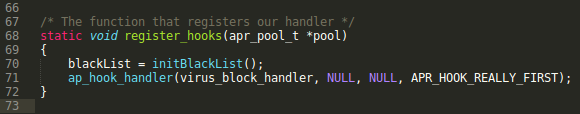


**mod\_virus\_block.c:**

קובץ קוד המקור של הכלי. מורכב מ4 הפונקציות הבאות ומהרשימה הבאה:



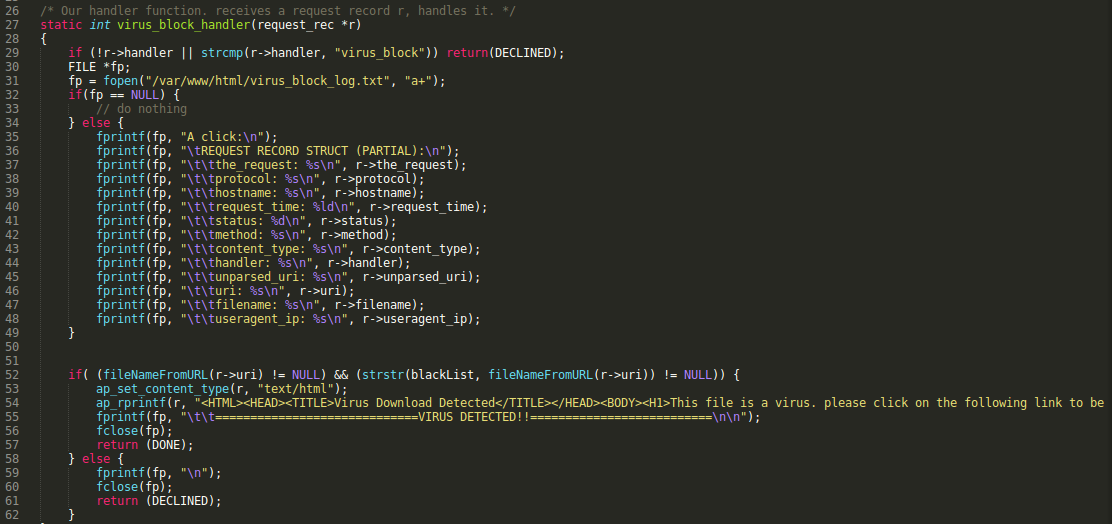
הblackList שבשורה 22 תהיה רשימת הקבצים העוינים אשר אנחנו נקרא מקובץ בשם BlackList.txt ואליה נשווה שמות של קבצים.

הפונקציה register\_hooks מפעילה את הhandler שמטפל בבקשות שעוברות דרך הפרוקסי ומפעילה את הפונקציה שמאתחלת את הblackList.

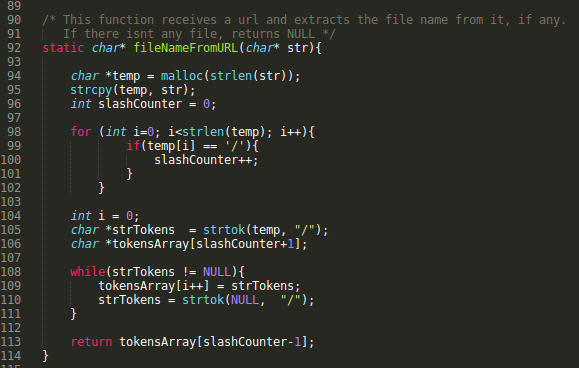
הפרמטר APR\_HOOK\_REALLY\_FIRST נועד לוודא שמודול זה יהיה הראשון שיעבוד על בקשת ה GET של דפדפן המשתמש.

הפונקציה virus\_block\_handler מבצעת את הדברים הבאים:

1. פתיחה של קובץ לוג ששם יתועדו הבקשות שעוברות דרך שרת הפרוקסי.
2. בודקת שאכן קיים uri לבקשה (בקשות מסוגים מסויימים לא מכילות uri).
3. מחפשת את שם הקובץ בuri ברשימה השחורה.
4. במידה ונמצאה התאמה, כלומר הקובץ מוגדר כעוין, המערכת מתריעה בפני המשתמש ונותנת את האפשרות לעבור לאתר בטוח.
5. בנוסף מבצעת תיעוד של ניסיון ההורדה של הקובץ העוין.

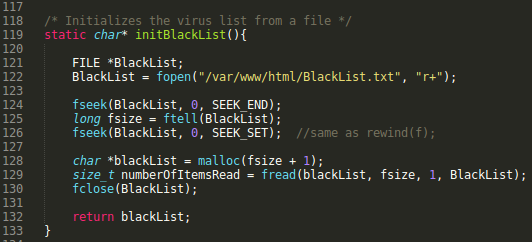


הפונקציה fileNameFromURL מקבלת url ומוציאה מתוכו את שם הקובץ שאותו נרצה לבדוק. אם לא קיים קובץ אז מוחזר NULL ואין מה לבדוק.

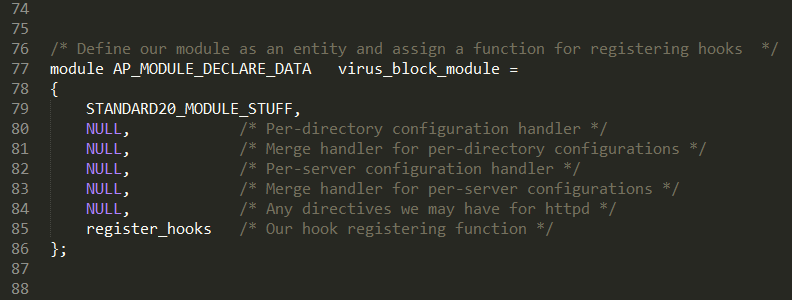


הפונקציה initBlackList קוראת את הקובץ עם רשימת הקבצים העוינים ובונה מחרוזת שבה נחפש את שמות הקבצים.

במידה שהתווספו שמות של קבצים עוינים, צריך רק להוסיף אותם לקובץ ולקרוא אותו מחדש.



כמו – כן ישנו המבנה שמגדיר כי כלי זה הינו מודול בשם virus\_block ומשייך אליו את הפונקציה register\_hooks אשר יודעת אילו handlers להפעיל עבור מודול זה.



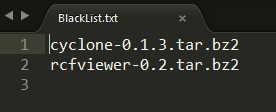
**BlackList.txt:**

קובץ זה הינו הקובץ אשר יכיל רשימת שמות של קבצים שהוגדרו עויינים ע"י מנהלי הרשת הארגונית (רשימה שחורה).

לתוך קובץ זה יוקלדו שמות קבצים שנמצאו עויינים ידנית, והכלי יסתמך על קובץ זה כאשר יסנן הורדות של קבצים מהאינטרנט.

מיקומו של הקובץ יהיה ב /var/www/html/

הקובץ ייראה כך:

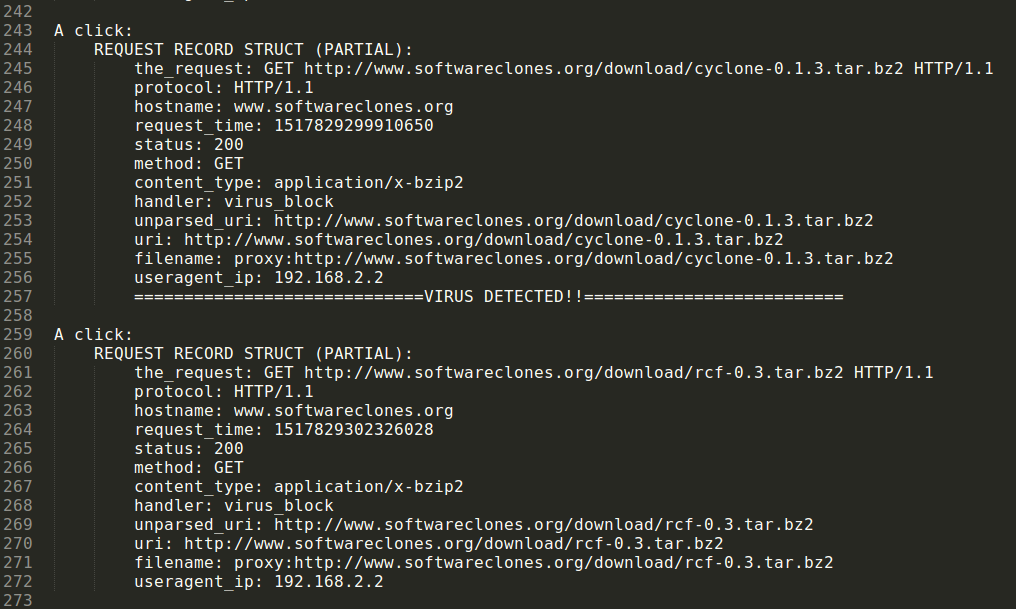


**virus\_block\_log.txt:**

קובץ אשר יתעד את פעולת הכלי ובכלל זאת אירועים שבהם היה ניסיון להוריד קבצים עויינים. להלן דוגמה שבה הכלי תיעד ניסיון הורדה של קובץ זדוני, ולעומתו תיעוד הורדה של קובץ נקי:

(הרצה זו השתמשה ב BlackList.txt המוגדר לעיל, לכן הקובץ

cyclone-0.1.3.tar.bz2 נחסם)



**מגבלות הכלי:**

הכלי מספק את הצרכים של רשת אירגונית בה המשתמשים אינם זדוניים ושבה מנהלי הרשת מסננים אתרים שנחשבים לא בטוחים. עם זאת, לכלי מספר מגבלות אשר הופכות אותו לפתרון לא אידאלי במקרה של ניסיון מכוון להוריד קבצים זדוניים. את המגבלות האלו נפרט כאן:

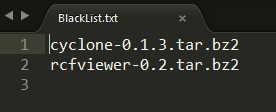
* קריטריון הכלי לסינון קבצים הינו שם הקובץ בלבד. דבר זה דורש עדכונים תכופים יחסית של קובץ ה BlackList.txt במקרים בהם שמו של קובץ זדוני שונה. בנוסף, במידה ומישהו רוצה במכוון להוריד קובץ עוין, אם הוא יוכל לגרום לכך ששם הקובץ ישונה, הוא יוכל להוריד את הקובץ.

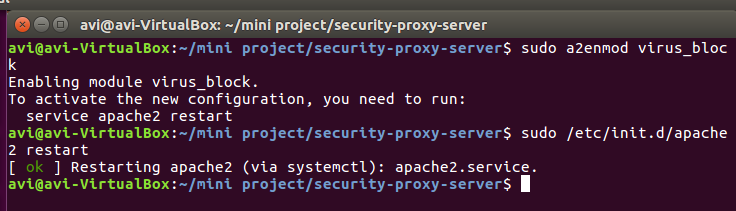
פתרון אפשרי לבעיה זו הינה הוספת קריטריונים נוספים לסינון קבצים, כגון שמירת רשימה שחורה של חתימות קבצים, אך באותה מידה ניתן לעקוף גם פתרון זה, למרות שדבר זה יהיה יותר מאתגר.

* חוסר יכולת לטפל בבקשות מוצפנות. כלי זה מתמודד היטב עם בקשות בפרוטוקול HTTP, מה שמאפשר לו להציץ לתוכן הבקשה ולשלוף משם את שם הקובץ. בבקשות מוצפנות, אין אפשרות לכלי לגשת לתוכן הבקשה, ולכן הוא לא יכול לשלוף שם של קובץ להשוואה. למשל, במידה ולקוח ינסה להוריד קובץ מפייסבוק, הכלי לא יוכל לסנן קובץ זה.

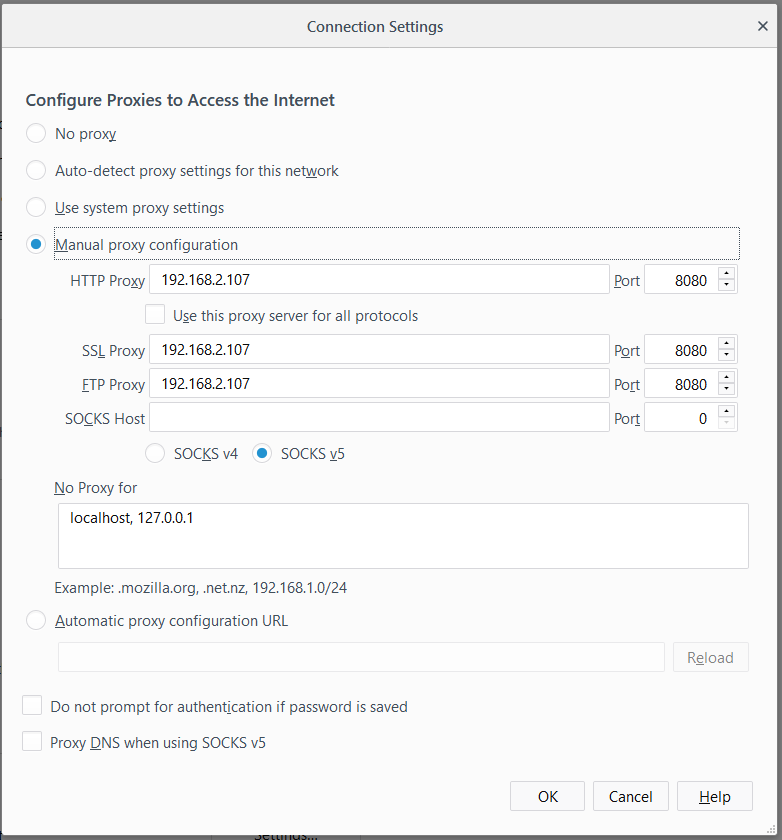
פתרון לבעיה זו הינו מסובך היות והוא כרוך בפיצוח הבקשה המוצפנת ע"י זיוף certificates או ע"י הנפקה של certificates לשרת עצמו, כך שיוכל לראות תכני בקשות מוצפנות.

**דוגמת הרצה:**

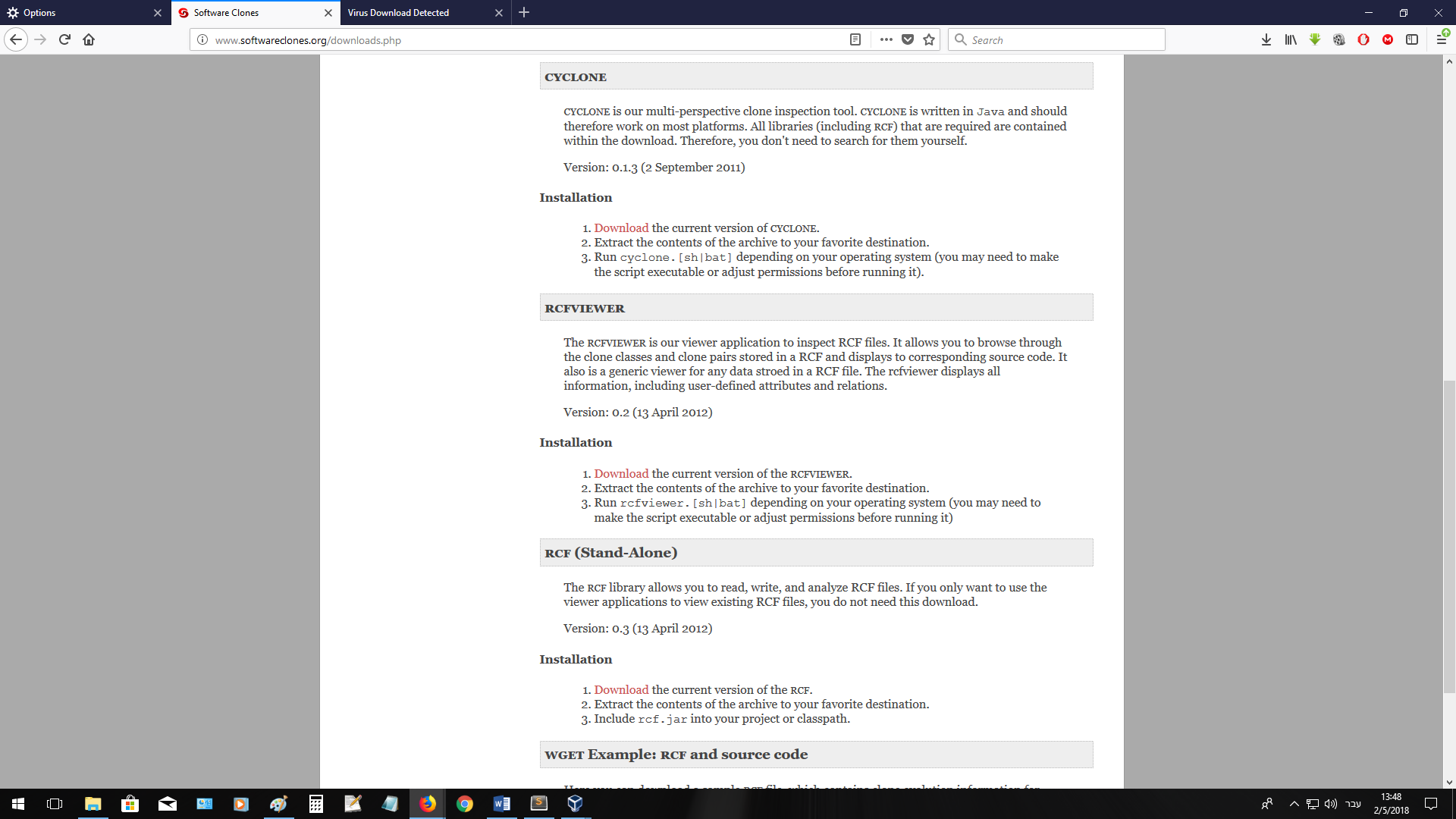
נראה את פעולת הכלי עבור ה BlackList.txt שראינו לעיל. נזכיר:

לאחר שהכלי מותקן, והרשימה השחורה מעודכנת, נפעיל את השרת:

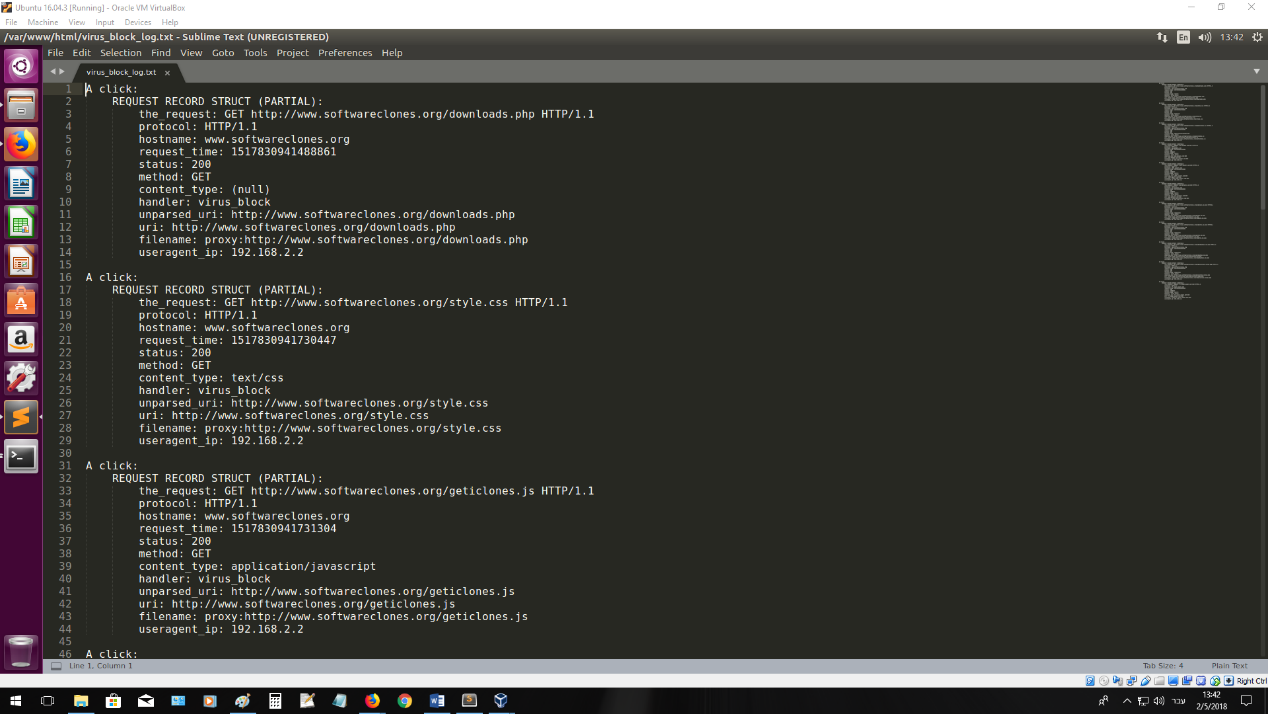
לאחר מכן נבדוק מה ה ip של המכונה עליה רץ השרת:

ובדפדפן הלקוח נגדיר את הגדרות הפרוקסי בצורה זו:

נגלוש כעת לאתר http מסויים:

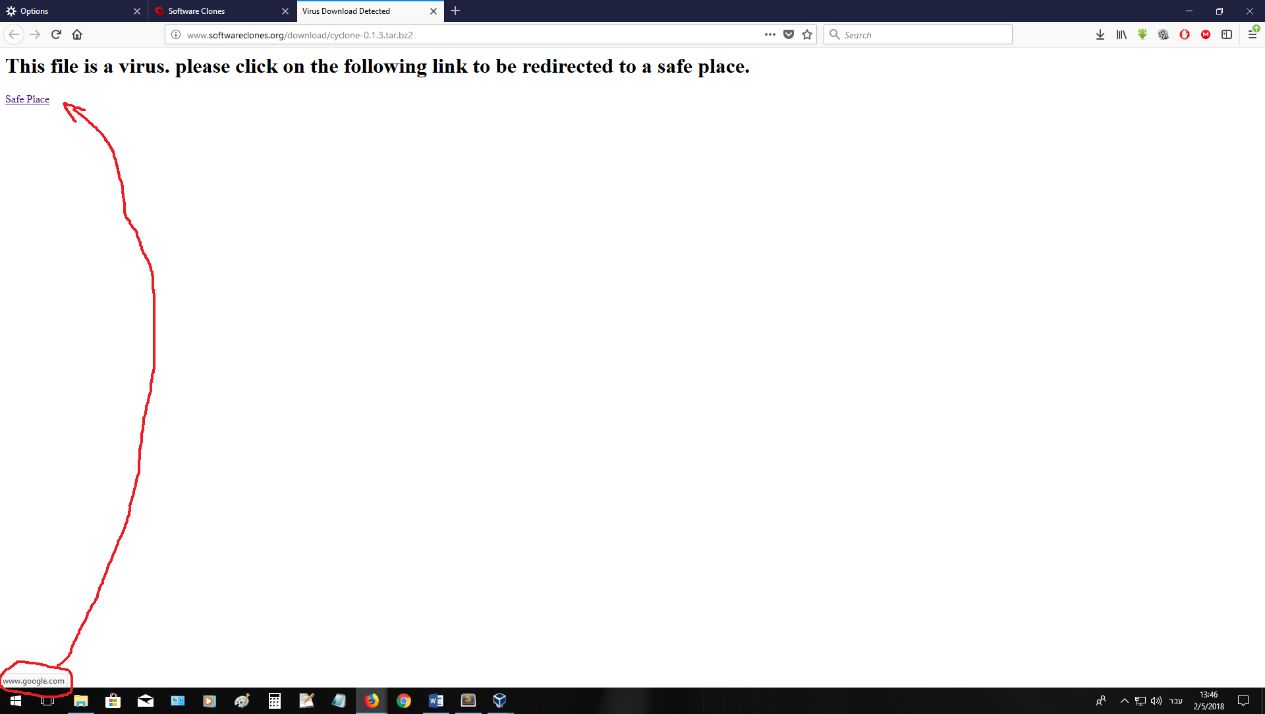


וכפי שנשים לב, יתחיל תיעוד של בקשות ה GET בקובץ virus\_block\_log.txt

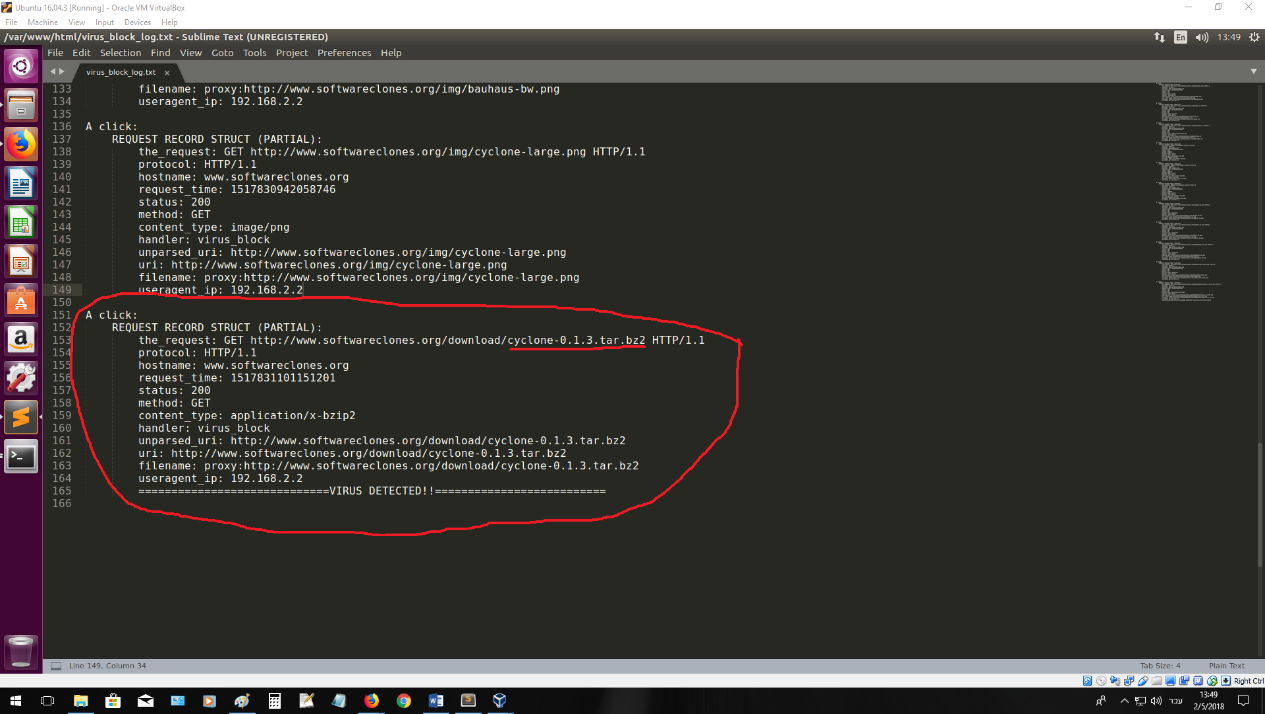


אם ננסה להוריד כעת את cyclone, אשר מוגדר כקובץ עוין, לא נצליח.

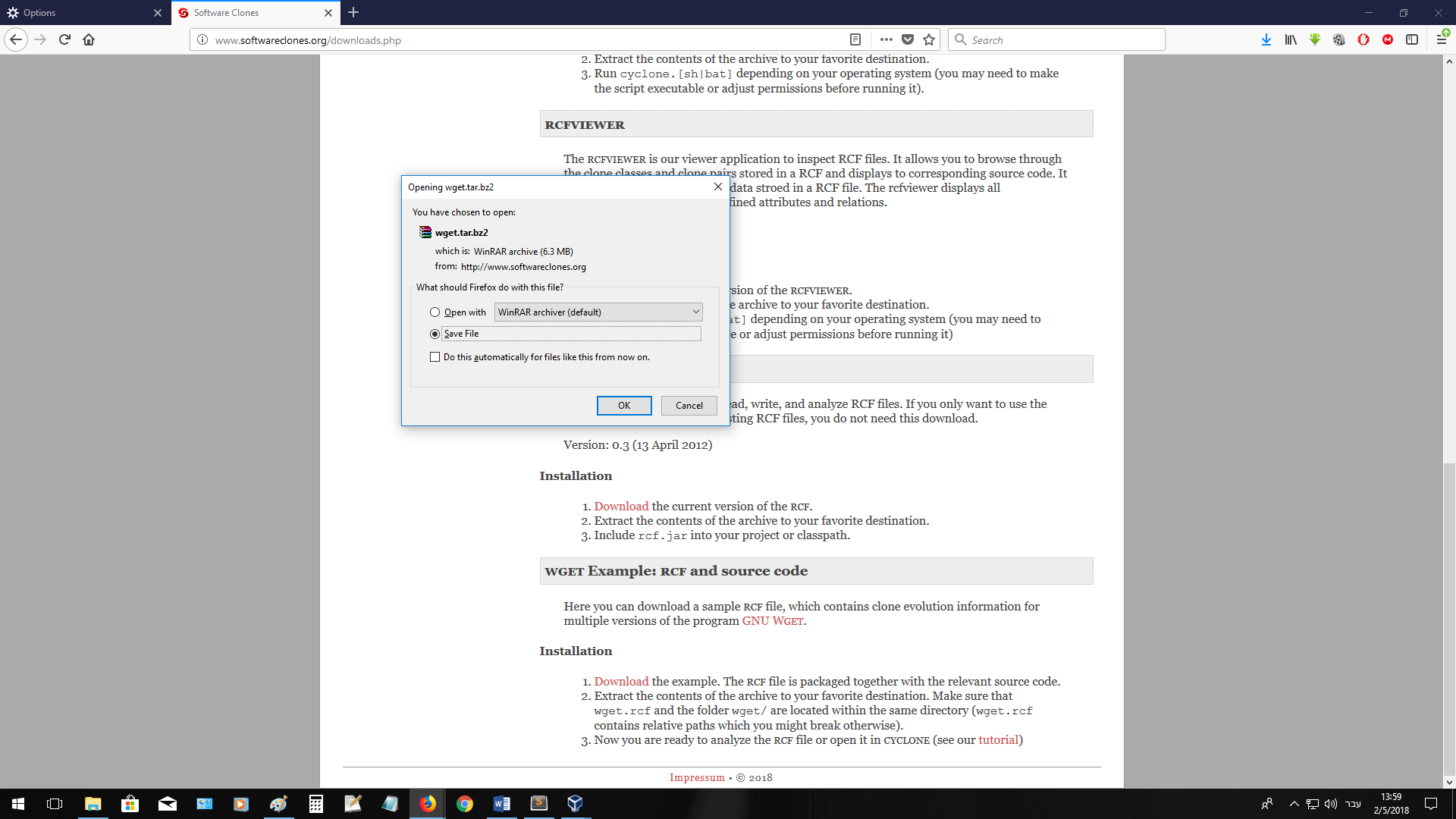
תחילה, השרת יציג הודעה על כך בדפדפן שלנו, עם לינק הפנייה לאתר הבית של גוגל ("אתר בטוח").



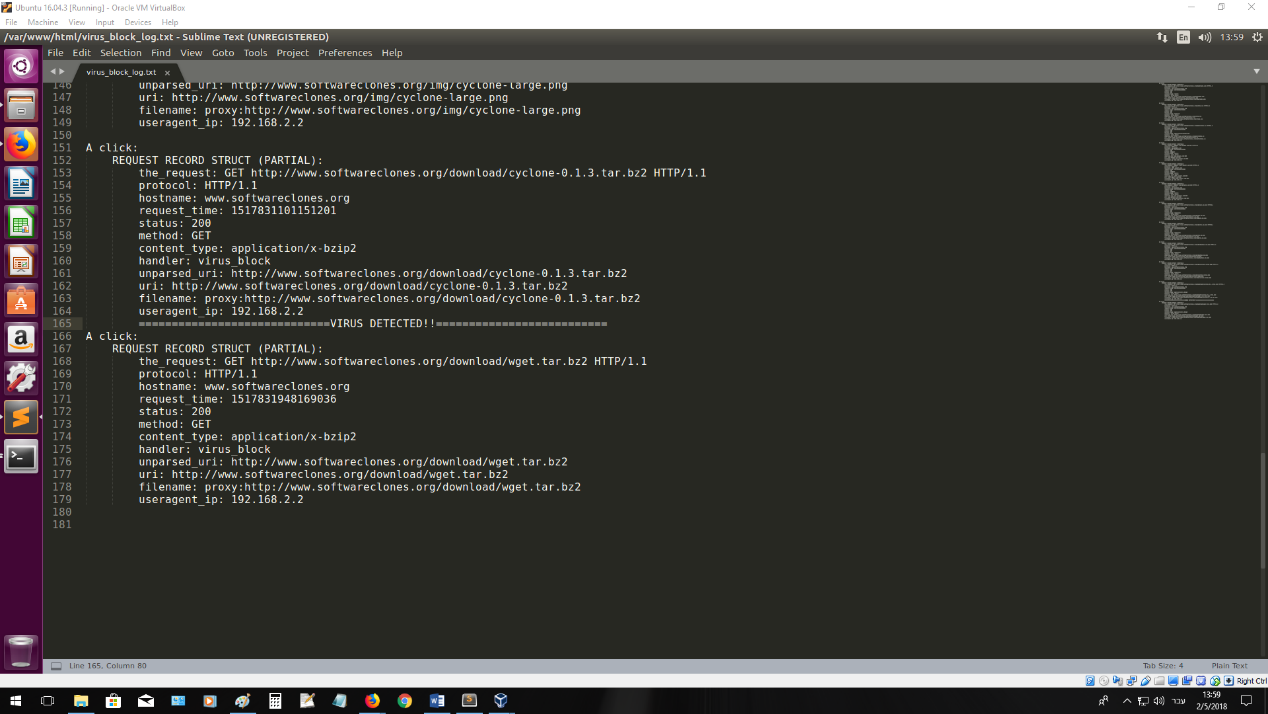
בנוסף, אנו נראה תיעוד של ניסיון ההורדה:



לעומת זאת, אם ננסה להוריד את WGET Example:



ופעולה זו תתועד כפעולה בטוחה:



**סיכום:**

הכלי שבנינו הינו כלי שמתאים לרשת ארגונית אשר מטרתו למנוע הורדה תמימה של קבצים עוינים, הוא מצליח להתמודד עם מקרים פשוטים, מבצע ניטור ותגובה ברמה בסיסית, אשר מתאימה לארגון בו למשתמשים אין ידע עמוק בנושאי אבטחת מידע.

הכלי מתממשק בצורה טובה לשרת אפאצ'י 2 במוד פרוקסי, פשוט להתקנה, וקל לתחזוקה שבאה לידי ביטוי בעדכונים לבסיס הנתונים שלו.

**ביבליוגרפיה:**

* לצורך הבנה של אופן עבודת מודולים באפאצ'י 2 ואיך לכתוב מודול נעזרנו באתר של אפאצ'י 2 ובעיקר באתר זה:

<https://httpd.apache.org/docs/2.4/developer/modguide.html>

* לצורך קינפוג שרת פרוקסי של אפאצ'י כשרת forward-proxy נעזרנו באתר זה:

<http://geek-university.com/apache/configure-apache-as-a-forward-proxy/>

* לצורך הדוגמאות שהצגנו השתמשנו באתר של softwareclones:

<http://www.softwareclones.org/>

* google, stackoverflow, Wikipedia עבור כל דבר נוסף שהיינו צריכים לדעת.