אבטחת מערכות תוכנה

ממ"ן 16

מרצה: יצחק בייז, פרופסור אהוד גודס
בודק תרגילים: אלעד לוי

מגיש: אוראל דמרי
תעודת זהות: 204739825

תאריך: 24/07/2022

האוניברסיטה הפתוחה
The Open University of Israel

תוכן עניינים

אבטחת מערכות תוכנה- ממ"ן 16

הקדמה

בחרתי לבנות מערכת Runtime Protection עבור ubuntu-22.
הרעיון של Runtime Protection הוא שבמידה ופורץ אכן הצליח לקבל כבר גישה למערכת ההפעלה- זו תהיה התוכנה האחרונה שתנסה לעצור בעדו לבצע פעולה זדונית.
כיום ישנם הרבה מערכות Runtime והן מגנות מסוגים שונים של התקפות, ועל סוגים שונים של תשתיות.
לדוגמא twistlock היא אחת מהמערכות שיודעות לבצע Runtime Protection על גבי containers וגם על גבי hosts.
גם aqua פועלת בצורה דומה.
בעבר אנטי וירוסים היו מסתכלים על חתימה של קובץ מסויים וכך מזהים אם הוא טרויאני. כיום הסוסים הטרויאנים והתולעים יותר מתוחכמים. בנוסף הובן שפעולה זדונית כיום היא לא רק סוס טרויאני, אלא למשל הפעלה של קריפטומיינר על מכונה על מנת להרוויח כסף על חשבון משאבים.
בנוסף, ניתן כיום לזהות התנהגות זדונית גם בעזרת Machine Learning.
מערכות Runtime משלבות את כל סוגי ההגנות שהזכרתי, לפעמים אפילו משלבות בתוכן מערכות נוספות כמו למשל מערכות סריקת קוד, או סריקת imageים שבשימוש.

המערכת שבניתי

המערכת שלי יודעת לדווח על התנהגות חשודה שקורית על מערכת ההפעלה ubuntu-22.
ההתנהגויות שמערכת תדע לזהות הן:

- Reverse shell
- Bind shell
- זיהוי Malware ע"י שימוש ב Feed
- קריפטומיינר מותקן על המערכת
- פניה ל Domain חשוד
- שינוי בבינאריים של מערכת ההפעלה

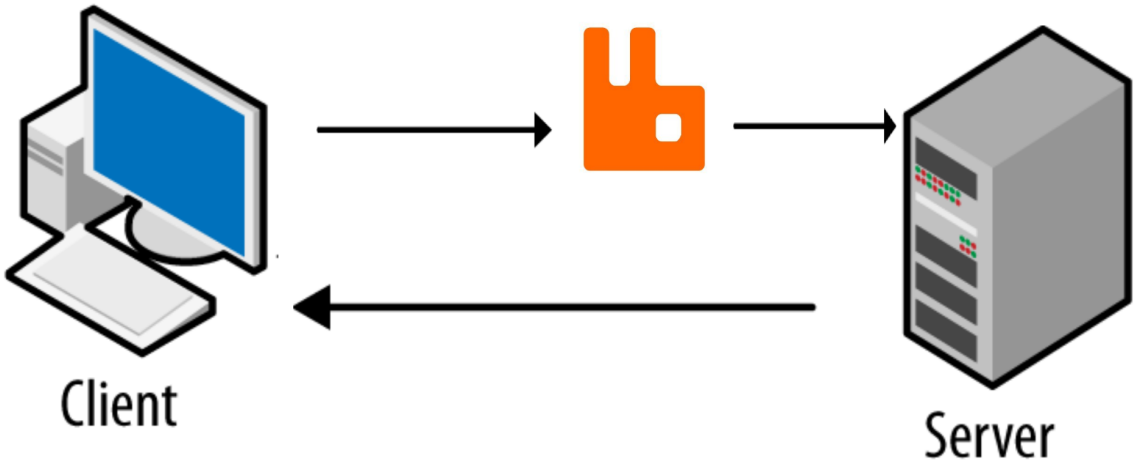
כמובן שישנן מגבלות למערכת שבניתי בגלל הגודל ומורכבות הפרוייקט.
בחרתי להתמקד אך ורק בשינויים בבינאריים בתיקיית הבינארים המרכזית של מערכת ההפעלה - /bin .
בנוסף- יכול לקרות מקרה של shortlive שבו לא אספיק לתפוס את הevent- למרות שאני מסתכל על הeventים באופן תמידי.
הפרוייקט מומש בPython.

לפרוייקט ישנן 2 צורות-
On prem- הפעלה של הagent שבניתי- הוא מדפיס למסך על events חשודים
Saas - בניתי אתר בflask שיושב על aws שמציג events חשודים. בצד השני כל agent שמתחבר יודע לשלוח לאתר את הevents. הכל עובר בדרך ב RabbitMQ למיתון העומסים. כל השירותים יושבים על AWS. נעשה שימוש ב AmazonMQ ו EC2.

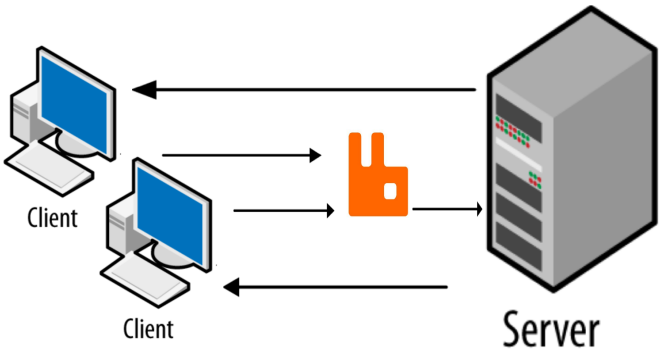
ארכיטקטורת SAAS

המערכת מורכבת משלושה קומפוננטות עיקריות:

- סרבר שמציג את האירועים החשודים
- קליינט ששולח אירועים חשודים למחסנית מסוג RabbitMQ
- מחסנית שמחזיקה את כל האירועים עד שהסרבר שולף אותם



השרת יושב על גבי EC2.
הקליינט הוא פרוסס של ubuntu22.
המחסנית היא שירות serverless על גבי AmazonMQ- בחרתי להרים RabbitMQ.
ניתן לחבר כמה קליינטים שרוצים לסרבר.



ארכיטקטורת On Premise

בארכיטקטורה זו ישנו רק קליינט - שיודע להדפיס למסך במידה ויש אירוע חשוד.

הוראות הפעלה:

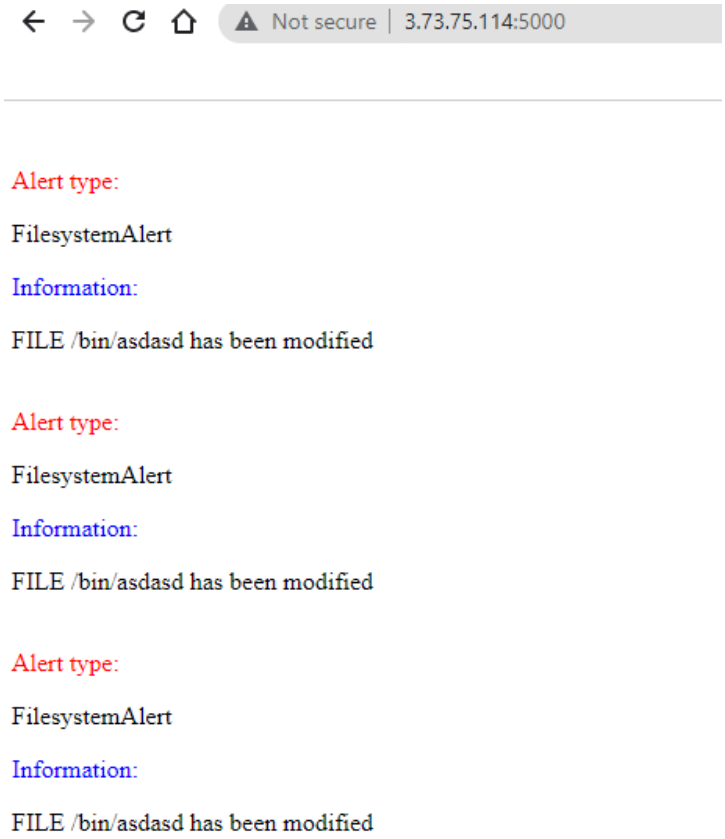
כפי שנאמר- בניתי למערכת 2 צורות: אחת היא SAAS והשנייה היא OnPrem.
השארתי לנוחיותך ubuntu22 כEC2 שתוכל להתחבר אליו- ועשיתי deployment ל rabbitmq
ולserver כבר כדי שתוכל פשוט להתחבר אליהם.

לכל בעיה בכל שעה אני זמין בטלפון 0525363208.

Github: https://github.com/Oreldm/RuntimeDefender
SAAS Website: http://3.73.75.114:5000/
RabbitMQ:
https://b-1a801484-0559-42b6-af8b-d16dd2535eef.mq.eu-central-1.amazonaws.com/
User: myuser
Password: mypassword1mypassword1

לשימוש ב SAAS (מומלץ):

1. בצע ssh לכתובת 3.73.75.114 עם המשתמש ubuntu ועם הpem בשם mine שנמצא בתיקיית הקוד. הפקודה המלאה היא ***ssh ubuntu@3.73.75.114 -i mine.pem***
2. הפעל את הקליינט:
   a. לך לתיקייה /home/ubuntu/RuntimeDefender/saas
   b. כבר שמתי PythonPath אך במקרה ישנה בעיה תוכל לרשום
      i. export PYTHONPATH="/home/ubuntu/RuntimeDefender"
   c. הפעל את הקליינט:
      i. python3 client.py
3. לך לAlertServer:
   a. בדפדפן גלוש ל http://3.73.75.114:5000/
   b. אם השרת לא למעלה אפשר להרים אותו בעזרת python3 server.py באותה תיקייה.
4. נסה את אחת האפשרויות- למשל:
   a. לך לתיקייה /bin ותערוך שם קובץ
   b. תעשה reverse shell:
      i. /bin/sh -i
5. ייצר קובץ בשם xmrig.
6. עבור כל אחד מהמקרים אתה תקבל התראה בדפדפן בלייב (זה מתעדכן אוטומטית).



לשימוש במערכת הOnPrem:

1. התקן ubuntu 22
2. התקן את ה requirements
   a. pip install -r requirements.txt
3. הפעל את הקובץ main.py
   a. python3 main.py
4. נסה את האפשרויות שמתוארות בסעיף 4-5 בפסקה הקודמת
5. עבור כל התראה תודפס הודעה למסך

תיאור הפיצ'רים ושיטות זיהוי

כאמור, המערכת יודעת לזהות ולהתריע למשתמש שיסתכל בשרת Alerts על התנהגויות חשודות.
בכדי להצליח לזהות את כל אחת מההתקפות הייתי צריך לבצע מחקר לא קטן.
ההתנהגויות החשודות בהן אנו תומכים:

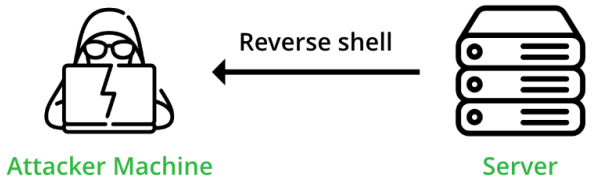
Reverse Shell:

הסבר:

במצב זה מה שיקרה זה שהמכונה תתחבר לתוקף וכל פקודה שהיא תקבל היא תעביר לבינארי של הshell, וכך התוקף יוכל לשלוח פקודות ל shell של המכונה.

שיטת זיהוי:

הקליינט מבצע ps -ef ורואה האם נעשה ניתוב ל /bin/sh כלומר יש -i כפרמטר.
אני בודק את זה במספר צורות על ידי מספר substrings שיכולים להופיע ב ps -ef.



Server tries to connect to Attacker machine

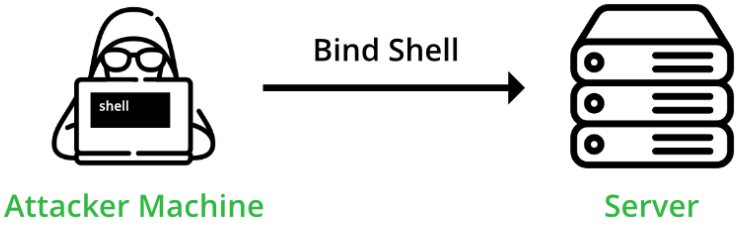
Bind Shell:

הסבר:

מצב זה הוא בדיוק הפוך ל Reverse Shell- כלומר במקום שהמכונה תתחבר לתוקף- התוקף יתחבר למכונה.

שיטת זיהוי:

בדומה ל Reverse Shell.



Attacker executing bind from his machine to server

שינוי קבצי מערכת:

פעמים רבות האקרים ישנו קבצים שנמצאים תחת /bin בכדי שלצורך העניין כאשר נרשום "ls" יופעל הקוד של ההאקר. אנו נדע לזהות אם התבצע שינוי כזה / הוזרק או הוחלף קוד של אחד הבינאריים הללו.

שיטת זיהוי:

אני מבצע md5 לקבצים במכונה ובודק שהם אכן זהים ל md5 שאמור היה להתקבל כאשר הותקן הclient.

זיהוי Malware לפי Feed:

יש היום Feeds רבים של Malwares .

בחרתי להשתמש בחלק מהFeed של `https://virusshare.com`

אני טוען אותו בתחילת הריצה לזכרון (אני טוען כ40 אלף).

כיוון שאין לי מנוי לשם- אני נאלצתי לטעון לזכרון (אני עושה scrape על האתר בתחילת הריצה), ולכן אני לא טוען את כל הmd5 כי זה יקח הרבה זמן.

שיטת זיהוי:

אני מבצע md5 לקבצים במכונה ובודק אם הם זהים לאחד מהmd5 מהfeed.

קריפטומיינרים:

כיום האקרים מתקינים קריפטומיינרים על גבי מכונות על מנת להרוויח כסף תוך שימוש במשאבים שלא שייכים להם. אנו יודעים לזהות את הקריפטומיינר xmrig, וכן במידה ויש צריכה גדולה של משאבים גם קריפטומיינרים אחרים.

שיטת זיהוי:

1. נעשה strings על כל בינארי שמותקן ונבדוק אם יש שם string שקשור לאחד מהקריפטומיינרים שלנו.
2. נקרא את השם של כל תוכנה חדשה שמותקנת ונראה אם יש בה xmrig (אני יודע לזהות עוד הרבה סוגים - למשל monero).
3. נסרוק את המשאבים של המכונה ונראה אם יש בה שימוש קיצוני לאורך זמן (למשל עם הפקודה top).

פנייה לDNS חשוד:

לפעמים האקרים בכדי להשתלט על המכונה- פותחים session למחשב שלהם.

כיום ישנם דאטאבייסים של Ip's ו DNS's שהם חשודים.

בחרתי להתמקד בסיומות של `.zip', '.review', '.country', '.kim', '.cricket',.'` `''.sceince', '.work', 'party','.gq', '.link`

מכיוון שלפי המחקר הנ"ל:

https://www.xfer.com/newsletter-content/10-of-the-most-dangerous-domains-on-the-web.html

אלו הן הסימות הכי מסוכנות כיום.

שיטת זיהוי:

אנחנו נשלח tcp dump לשרת ושם נפרסר אותו ונראה האם ישנה פנייה לdns שהוא חשוד.

מימוש הקוד:

צד שרת:

למעשה אנחנו נאסוף Process Output, System Files, Files , שימוש ב Resources, ואת תעבורת הרשת.
לכן ראשית אייצר קלאסים שהם data model (חשוב לי לציין שלא בכולם השתמשתי בסופו של דבר מפאת קוצר הזמן, במקום זאת השתמשתי ישירות בסטרינג ועשיתי regex/substring- אך בכל זאת השארתי את הקוד שכתבתי, כי בעתיד אוסיף את זה לדיזיין טוב יותר):

Alert
- Name
- information

EventModel
- path
- Filename
- Event_type

FileModel
- Name
- Md5

NetworkRequestModel
- domain_dst
- Ip_dst
- Ip_src
- Domain_src

ProcessOutputModel
- Proccess_arr : list

ResourceModel
- Cpu
- Memory

לאחר מכן יצרתי את ה Server עצמו עם flask:

Server.py
- Main Using flask

- getevents()
- index()

צד קליינט:

Client.py
- main

Watcher
- init(path_to_watch)
- watch

Verifier
- verify_cryptominer
- Verify_reverse_shell
- Verify_request
- Verify_resources
- Verify_malware_dict
- Verify_malware
- Verify_filesystem_event

Tools
- temrminal_command
- Get_md5
- Get_malware_feed

Settings
- PATH_TO_WATCH

מחסנית באמצע:

בחרתי להשתמש ב RabbitMQ בתור המחסנית בין השרת לקליינטים מכיוון שזה יודע לעמוד בעומסים שאצטרך והשימוש בה יחסית פשוט.

RabbitMqController
- Send_alert
- get_alert