

תיעוד ומידע

YARA

במהלך העבודה כחלק ממחקר בנושא אבטחת מידע וכלים של אבטחת מידע, נחשפנו ל-YARA. YARA הוא כלי בתחום הסייבר סקיוריטי, שנועד לזיהוי ואפיון קבצים ומערכות ולאיתור איומי אבטחה. הכלי הזה שימושי מאוד לחוקרי אבטחת מידע ולמומחי סייבר בזיהוי ומניעה של איומים.

YARA מאפשרת ליצור חוקים מותאמים אישית המגדירים תבניות של נתונים או התנהגויות שקשורים לנזקות. כשקובץ או מערכת עומדים בתנאים של חוק כזה, ניתן לזהות אותם כנזקה או כמאיומים. באמצעות הכלי, ניתן לכתוב חוקים שמזהים מחרוזות ועל ידי כך לזהות אותם כנזקות. יתר על כן, אפשר לשלב חוקים של YARA בסקריפטים וכלים אוטומטיים, מה שמאפשר אינטגרציה עם מערכות אבטחה קיימות.

CLAMSCAN

כלי OPEN-SOURCE המיועד לאיתור והסרה של נזקות ווירוסים. ClamScan הוא הרכיב של ClamAV שמבצע את הסריקה עצמה.

תכונות עיקריות של ClamAV ושל ClamScan:

1. מנוע סריקה חזק: משתמש במנוע סריקה מתקדם שיכול לאתר מגוון רחב של איומים, כולל וירוסים, תולעים, סוסים טרויאניים ותוכנות זדוניות אחרות.
2. תמיכה בסקריפטים: ניתן להשתמש ב-ClamScan יחד עם סקריפטים וכלים אחרים לצורך אוטומציה וניהול מרכזי של סריקות.

חקרנו על שני הכלים המדוברים, ראינו את השימושים והמימושים שלהם, והחלטנו לבנות 2 תוספים - אחד לכל כלי, כך שאם אחד מהכלים (לפחות) מזהה איום כלשהו, נוכל לסווג את הקובץ כולו כקובץ חשוד כזדוני.

הרצות

במסגרת בניית התוסף וחקירת כלי ה-ClamScan ניסינו להריץ את התוסף שלנו על פרוייקט עבר שלנו שהעלנו לגיטהאב.







ראשית, לאחר בניית התוסף, צירפנו את הקובץ שבנינו כחלק מה-REPOSITORY, אותו רצינו לסרוק. כחלק מהגדרת התוסף, בכל בקשת PUSH או PULL, מתבצעת סריקה של התוסף לכל התיקיות בפרוייקט.

מצורף בזאת הפלט שהתקבל כחלק מהרצה (ראשונית) של התוסף על הפרוייקט שלנו. במסגרת ההרצה הצלחנו לסווג את התוסף שלנו כלא תקין (ולאחר מכן עשינו בו התאמות ושינויים):

Some checks haven't completed yet





1 in progress and 2 successful checks


		Security Scan / scan (push) <i>In progress - This check has started...</i>	Details
		Vercel – code-mentor - Deployment has completed	Details
		Vercel – code-mentor-server - Deployment has completed	Details

Annotations

1 error and 2 warnings




 **scan**
Process completed with exit code 2.

 **scan**
The following actions uses node12 which is deprecated and will be forced to run on node16: actions/checkout@v2. For more info: <https://github.blog/changelog/2023-06-13-github-actions-all-actions-will-run-on-node16-instead-of-node12-by-default/>
[Show less](#)

 **scan**
The following actions uses Node.js version which is deprecated and will be forced to run on node20: actions/checkout@v2. For more info: <https://github.blog/changelog/2024-03-07-github-actions-all-actions-will-run-on-node20-instead-of-node16-by-default/>
[Show less](#)

scan		
failed 1 minute ago in 20s		
<div> <input type="text" value="Search logs"/> <input type="button" value="Refresh"/> <input type="button" value="Settings"/> </div>		
>	✔ Set up job	0s
>	✔ Checkout repository	5s
>	✖ Set up ClamAV	13s
	⌚ Run ClamAV scan	0s
>	✔ Post Checkout repository	0s
>	✔ Complete job	0s

לאחר שינוי ועדכון של התוסף, והרצה נוספת של אותו הפרויקט (שהוא כמובן פרוייקט תקין שלא מכיל קוד זדוני), ראינו כי התוסף אכן סיווג אותו כבטוח לשימוש:

All checks have passed		
3 successful checks		
✔	 Security Scan / scan (push) Successful in 17s	Details
✔	 Vercel – code-mentor - Deployment has completed	Details
✔	 Vercel – code-mentor-server - Deployment has completed	Details

לאחר מכן, הרצנו גם את התוסף השני שלנו, שמשתמש ב-YARA על פרוייקט גיט שלנו, שאנו יודעים שהוא תמים, ואכן קיבלנו את ההדפסה:

```
$ python scan_repo.py
Enter the GitHub repository URL: https://github.com/omer2080/code-mentor
Cloning into 'temp_repo'...
remote: Enumerating objects: 188, done.
remote: Counting objects: 100% (188/188), done.
remote: Compressing objects: 100% (129/129), done.
remote: Total 188 (delta 92), reused 133 (delta 48), pack-reused 0
Receiving objects: 100% (188/188), 81.37 KiB | 661.00 KiB/s, done.
Resolving deltas: 100% (92/92), done.
Cloned repository to temp_repo
No malicious code detected.
```

כדי לוודא שאכן התוסף שלנו עומד במטרתו, "השתלנו" קוד זדוני לתוך אותו ה-REPOSITORY, ואכן במקרה הזה הוא זיהה את הקובץ כזדוני, ומיד ביצע מחיקה:

```
$ python scan_repo.py
Enter the GitHub repository URL: https://github.com/omer2080/code-mentor
Cloning into 'temp_repo'...
remote: Enumerating objects: 187, done.
remote: Counting objects: 100% (187/187), done.
remote: Compressing objects: 100% (128/128), done.
Receiving objects: 69% (130/187), 80.57 KiB | 660.00 KiB/s, done.
Receiving objects: 100% (187/187), 80.57 KiB | 660.00 KiB/s, done.
Resolving deltas: 100% (92/92), done.
Cloned repository to temp_repo
Malicious code detected:
File: Ransomware_Patterns - Match: [$ransomware_note]
Repository at 'temp_repo' has been deleted.
```

בתוסף הנ"ל, אנחנו מבקשים מהיור להכניס ל-TERMINAL את ה-URL של ה-REPOSITORY אותו הוא רוצה לסרוק. לאחר מכן מתבצע פקודת ה-GIT CLONE ואז אנו בודקים האם ה-YARA RULES שהגדרנו מתקיימים עבור אחד (או יותר) מהקבצים ב-REPOSITORY.

במידה והקובץ נמצא כמזיק, אנו מיד מבצעים מחיקה של תיקיית ה-REPOSITORY.

השוואת התוספים והמערכת המובנית של GitHub

על אף שאיננו מתיימרים לבנות תוסף שיוכל להתחרות ביכולות הסריקה ואיתור הוירוסים המובנית של GitHub, ישנו יתרון מובהק לבניית תוסף באופן עצמאי:

- סריקה מותאמת אישית: באמצעות בניית תוסף, נוכל לבצע סריקה מפורטת לאיתור תוכנות זדוניות, ולספק בדיקת אבטחה מותאמת אישית לצרכינו או לצרכי הלקוח.

לעומת זאת, כמובן של-GitHub יש יתרון שאיננו יכולים לספק:

- בניגוד לבניית תוסף באופן עצמאי, שדורש בניה ותחזוקה, האבטחה המובנית של GitHub מספקת חבילה מקיפה של תכונות אבטחה אוטומטיות עם הגדרה מינימלית, תוך התמקדות בפרצות תלות, סריקת קוד וזיהוי סודי.