Digital Whisper Technologic Papers

Detection Engineering בארץ הקודש

(לקחים, תובנות ושלל אוצרות נוספים)

מאת דניאל קויפמן

הקדמה

Detection Engineering הינו תחום מרתק. זה לא סתם תחום מרתק, זאת גם ההתמחות שלי, ואני באמת יכול לכתוב ספר שלם בנושא. המקצוע הזה נכון להיום הוא די נישתי בעולם (אם כי המודעות אליו עולה בהדרגה), על אחת כמה וכמה בישראל. חלק מהסיבות למה המקצוע הזה נישתי יכולות להסתכם בנקודות הבאות:

- דרישות סף גבוהות לכניסה
- שמור לארגונים וחברות בעלות תוכנית אבטחת מידע בוגרת
- שמור לארגונים שיכולים להרשות לעצם כוח אדם מוכשר, וכתוצאה מכך, יקר

רק בהתבסס על הנקודות האלו, אפשר להבין מדוע המקצוע הזה עדיין בשלביו הראשונים. זה מזכיר לי את התקופה שבה טכנולוגיות ענן היו חדשות, אף אחד לא באמת הבין מהיכן באו ולהיכן הן הולכות, הכשרות וחומרי לימוד היו דלים וכו'. אני מאמין שהתחום הזה נמצא כרגע באותו מצב, אם כי מגמת המשרות המפורסמות ב-Linkedin לתחום הזה עולה בקצב איטי (במיוחד באירופה ובארה"ב). חברות מבינות את הנחיצות בתוכנית Detection שמתאימה להתמודד עם מטריית האיומים המורכבת של העולם המודרני. במאמר הזה, אנחנו נבין יחד מה, למה ואיך, מבלי להיכנס לעומקים טכניים יתר על המידה.

אז מה זה Detection Engineering ו-Detection-as-Code?

Detection Engineering הוא תחום העוסק ביצירה, פיתוח, ויישום של מערכות וטכנולוגיות לגילוי איומים, בעיקר בהקשרים של אבטחת מידע וסייבר. המטרה היא לזהות בצורה מהירה ויעילה ניסיונות חדירה, מתקפות סייבר או איומים פנימיים על מערכות המידע הארגוניות.

בתחום זה מתמקדים בבנייה של חוקים, אלגוריתמים וכלים אוטומטיים שמסוגלים לזהות התנהגות חשודה או אנומליות במידע שמתקבל מרשתות, ממערכות הפעלה, מאפליקציות או ממכשירים מחוברים.

הדבר נעשה באמצעות ניתוח נתונים, שימוש בכלי ניטור, ושיטות אוטומטיות כמו ניתוח לוגים, כלים לזיהוי מבוסס חתימה (Signature-based Detection) או ניתוח מבוסס התנהגות (Behavioral Detection).

על אבני הבסיס האלו, יושב לו תת-תחום שנקרא "זיהוי כקוד" (Detection-as-Code), ופה העניינים מתחילים לצבור תאוצה. זיהוי כקוד זאת גישה חדשנית לפיתוח ויישום מערכות זיהוי איומים בסייבר, **שבה חוקי הזיהוי והתגובה לאיומים נכתבים, מתוחזקים ומנוהלים כקוד תוכנה.** במקום להסתמך על כלים ידניים או חוקים סטטיים שמתקשים לעמוד בקצב של איומי סייבר המשתנים במהירות, גישה זו מאפשרת להשתמש בשיטות מודרניות של פיתוח תוכנה כמו בקרת גרסאות, בדיקות אוטומטיות ופריסה מתמשכת (Continuous Deployment).

בזיהוי-כקוד, צוותי אבטחה יכולים לכתוב ולהגדיר חוקי זיהוי בהתאמה אישית עבור סביבת הארגון, לבדוק את האפקטיביות שלהם לפני הפריסה, ולשפר את החוקים על בסיס שינויים ותובנות חדשים. תהליך זה הופך את מערכת הזיהוי לגמישה יותר, מאפשר שיתוף ושימוש חוזר בחוקים, ומבטיח ניהול שיטתי ומסודר של כללי הזיהוי.

הגישה משלבת בין עקרונות הנדסת תוכנה לבין אבטחת מידע, ומסייעת לצוותי הגנה לפתח מערכות זיהוי שיכולות להתמודד בצורה מהירה ויעילה עם איומים דינמיים.

מה נדרש מ-Detection Engineer לדעת?

ברמת הבסיס, Detection Engineer יבין עקרונות רבים מעולמות הסייבר, התכנות וה-DevOps:

- <u>הבנה מעמיקה בתחום הסייבר</u>: היכרות עם סוגי התקפות נפוצות, שיטות פעולה של תוקפים (TTPs), וכלים המשמשים אותם.
- <u>היכרות עמוקה עם מערכות SIEM:</u> כמו QRadar, Splunk, Sentinel, Elastic, Chronicle
- <u>היכרות אינטימית עם Logging:</u> Detection Engineer חייב להכיר לוגים כמו שהוא מכיר את כף ידו. החוקים שהוא כותב, והרמה שלהם, יהיו תלויים בהבנה שלו ב-Telemetry שמוצרי ורכיבי אבטחת מידע שונים ומגוונים מייצרים, כמו Okta, Entra ID, Firewalls, EDR, Office 365 וכו'. בנוסף לכך, נדרשת הבנה ב-Regex על מנת לפרסר שדות מהלוגים עצמם.
- <u>כתיבת קוד</u>: יכולת לכתוב קוד על מנת לבנות CI/CD Pipelines שתומכים בבדיקה והפצה של חוקי זיהוי למערכות הרלוונטיות.
- <u>בקרת גרסאות</u>: שליטה במערכות ניהול קוד כמו Git לצורך פיתוח ושיתוף של חוקי זיהוי כקוד.
- <u>ניסיון באוטומציה:</u> שימוש בכלים שתומכים בהפצה כלל-ארגונית, כמו Jenkins או GitLab CI/CD.
- <u>הכרה ויכולת לעבוד עם APIs:</u> על מנת לכתוב סקריפטים שיודעים להפיץ חוקים ישירות ל-API של המוצר.

הסף הוא אכן גבוה, והוא רק עולה כל הזמן.

דוגמה לתוכנית Detection-as-Code בארגון

יישום של תוכנית זיהוי-כקוד בארגון כולל מספר שלבים המשלבים כלים ותהליכים מודרניים של פיתוח תוכנה עם אבטחת סייבר. חשוב לציין שכל ארגון כמובן מתמודד עם הקשיים והמגבלות שלו, אך השלבים האלו יכולים לשמש בתור בסיס סולידי:

שלב 1 - איסוף דרישות והבנת האיומים:

בשלב זה, צוות אבטחת המידע (SOC) מנתח את הסיכונים, מאפייני המערכת והאיומים הרלוונטיים לארגון. הם מזהים סוגי התקפות פוטנציאליות, התנהגויות חשודות ואנומליות שצריך לגלות.

שלב 2 - כתיבת חוקי זיהוי כקוד:

בשלב זה, בדרך כלל ארגונים משתמשים במאגרי חוקים פתוחים, כמו SIGMA, למרות שיש גם חברות שבנו את עצמן בדיוק על התחום הזה, כמו SOCPrime ו-Snapattack. SIGMA היא מאגר במקור פתוח שנוצר על ידי חברת Nextron. היא הפכה להיות מאגר שהמון חברות ומומחים נעזרים בו על מנת לפתח חוקי זיהוי חדשים. חוקי SIGMA כתובים ב-YAML. ניקח לדוגמה חוק מהמאגר: חוק SIGMA בעצם אמור לתת לנו אינדיקציה לאיך אנחנו אמורים לזהות איום מסוים. בדוגמה למעלה, השם של החוק הוא "Suspicious Download via Certutil.exe":

```yaml
title: Suspicious Download Via Certutil.EXE
id: 19b08b1c-861d-4e75-a1ef-ea0c1baf202b
related:
    - id: 13e6fe51-d478-4c7e-b0f2-6da9b400a829
      type: similar
status: test
description: Detects the execution of certutil with certain flags that allow the utility to download files.
references:
    - https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/certutil
    - https://forensicitguy.github.io/agenttesla-vba-certutil-download/
    - https://news.sophos.com/en-us/2021/04/13/compromised-exchange-server-hosting-cryptojacker-targeting-other-exchange-servers/
    - https://twitter.com/egre55/status/1087685529016193025
    - https://lolbas-project.github.io/lolbas/Binaries/Certutil/
author: Florian Roth (Nextron Systems), Jonhnathan Ribeiro, oscd.community, Nasreddine Bencherchali (Nextron Systems)
date: 2023-02-15
tags:
    - attack.defense-evasion
    - attack.t1027
logsource:
    category: process_creation
    product: windows
detection:
    selection_img:
        - Image|endswith: '\certutil.exe'
        - OriginalFileName: 'CertUtil.exe'
    selection_flags:
        CommandLine|contains:
            - 'urlcache '
            - 'verifyctl '
    selection_http:
        CommandLine|contains: 'http'
    condition: all of selection_*
falsepositives:
    - Unknown
level: medium
```

לפי התקציר של החוק, ניתן להבין שמטרתו היא לזהות שימוש ב-LOLBIN שנקרא Certutil, עם דגלים מסוימים המאפשרים לכלי הזה הורדה של קבצים. יוצר החוק גם דאג לספק לנו מיפוי ל-MITRE ATT&CK Framework, ורמת חומרה של Medium. החלק החשוב ביותר של חוק SIGMA נמצא תחת תגית ה-"Detection", בשורה 22, ותגית ה-"Condition", בשורה 32. תגית ה-Detection מגדירה לנו בעצם את מה שהחוק בא לזהות בצורה מפורטת. בחוק דוגמה שלנו, אנחנו יכולים להבין שמדובר באירוע Windows Event ID 4688 או באירוע Sysmon event ID 1, מכיוון שבשורה 20 מוגדר לנו Category של process_creation.

החוק מחפש כל אירוע שבו ה-Process name הוא "certutil.exe", ושה Commandline מכיל בתוכו את המחרוזת "urlcache" **או** את המחרוזת "verifyctl". השילוב של שלושת הנתונים האלו היא אינדיקציה לכך שתהליך Certutil מנסה להוריד קובץ - וזוהי פעולה חשודה. לאחר שיש לנו מושג מה החוק שאנחנו רוצים להטמיע, אנחנו נצטרך להבין איך אנחנו ממירים אותו כך שיתאים למערכת ה-SIEM שאיתה אנחנו עובדים, ואז מפיצים אותו אליה. נושא זה הוא עמוק מדי ולכן לא יהיה חלק מהאמר הנוכחי.

שלב 3 - Unit Testing:

לפני פריסה, החוקים נבדקים בתהליך של בדיקות אוטומטיות כדי לוודא שהחוקים מגיבים נכון לאירועים. באמצעות בדיקות יחידה (unit tests), ניתן להפעיל חוקים על אירועים סמליים ולהבטיח שהזיהוי עובד כראוי. ניתן להשתמש גם ב-Atomic על מנת לבצע סימולציה on-demand לחוקים כדי להבטיח שהם מייצרים אירוע לפי כוונת המשורר.

שלב 4 - Version Control:

הקוד מנוהל במערכת בקרת גרסאות כמו Git, מה שמאפשר לצוותי אבטחת המידע לעקוב אחרי שינויים, לשמור על היסטוריה של שיפורים בחוקים, ולעבוד בצורה מסודרת ומבוקרת על פיתוח והטמעה של חוקים חדשים.

שלב 5 - Deployment:

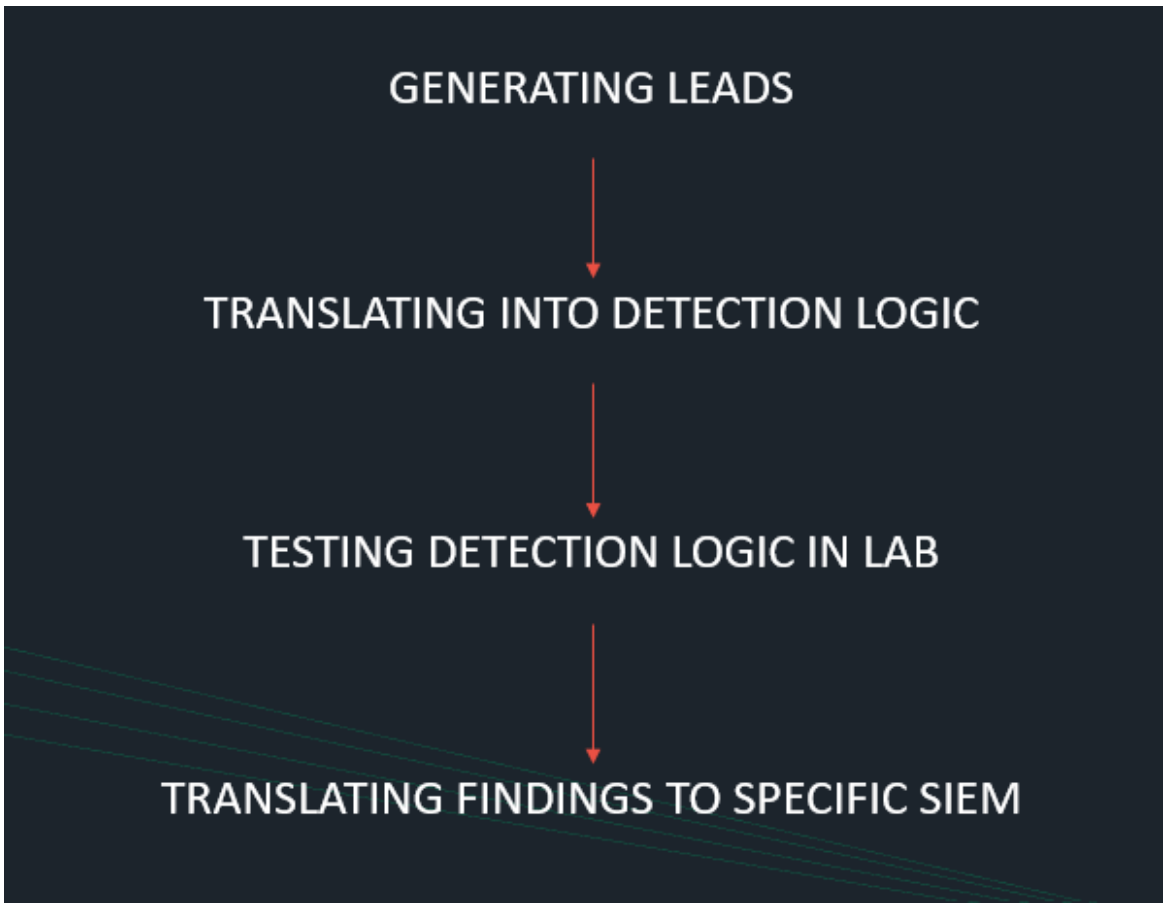
לאחר שעברו בדיקות, החוקים נפרסים בסביבת Production. כאן ניתן להשתמש בכלים אוטומטיים כדי להטמיע את החוקים ב-SIEM, בדרך כלל דרך ה-API שלהם.

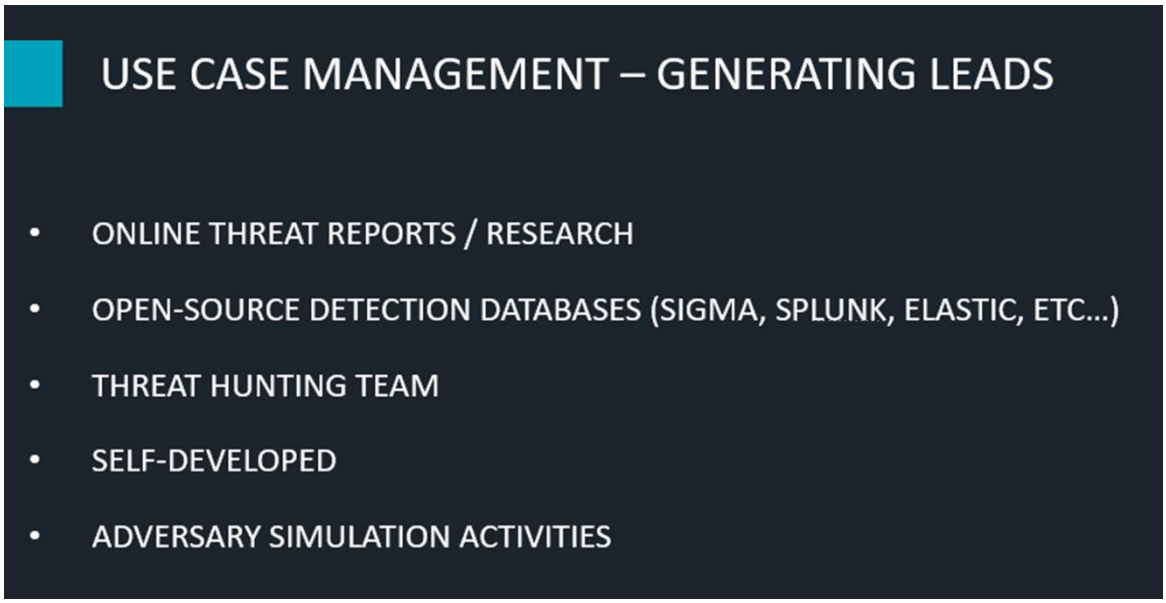
שלב 6 - ניטור והתאמה:

במהלך הפעילות השוטפת, החוקים שנפרסו מנוטרים כדי לוודא שהם מזהים איומים בצורה אפקטיבית. אם יש יותר מדי אזעקות שווא או אם מתגלות מתקפות חדשות, החוקים מעודכנים בהתאם. ה-SOC הוא החבר הכי טוב שלנו בשלב הזה, כי הם עובדים ב-"שטח".

איך אני מיישם Detection-as-Code במקום העבודה שלי?

לא אחשוף יותר מדי פרטים לעומק, אבל התהליך הוא פחות או יותר כזה (נלקח ממצגת שהכנתי לא מזמן):



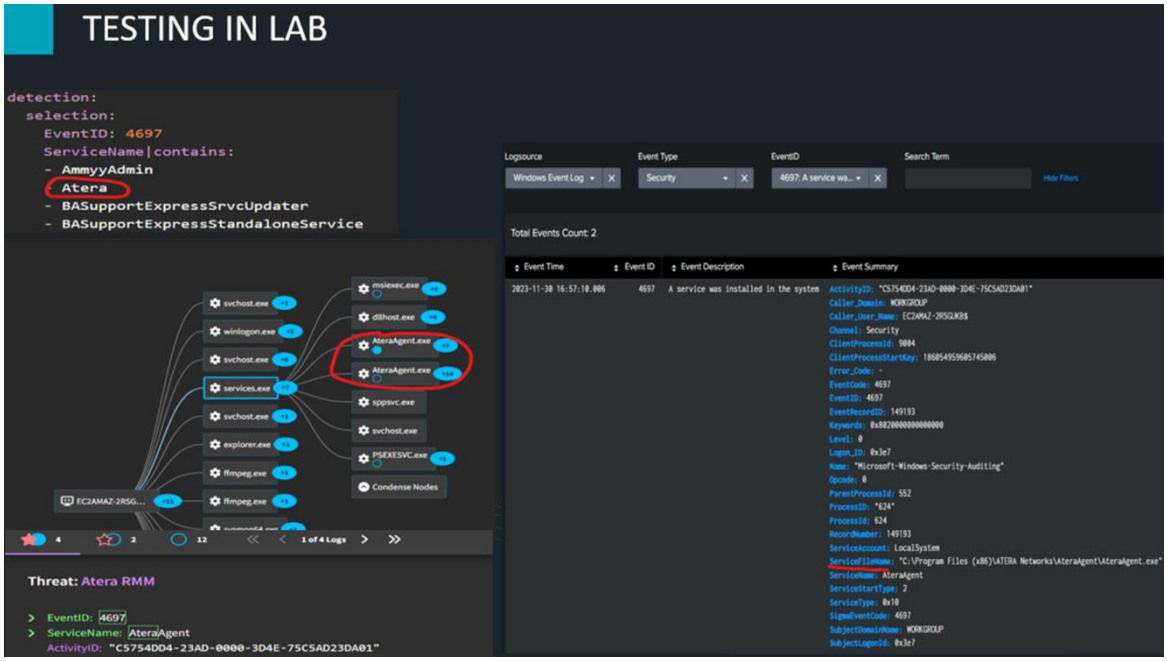


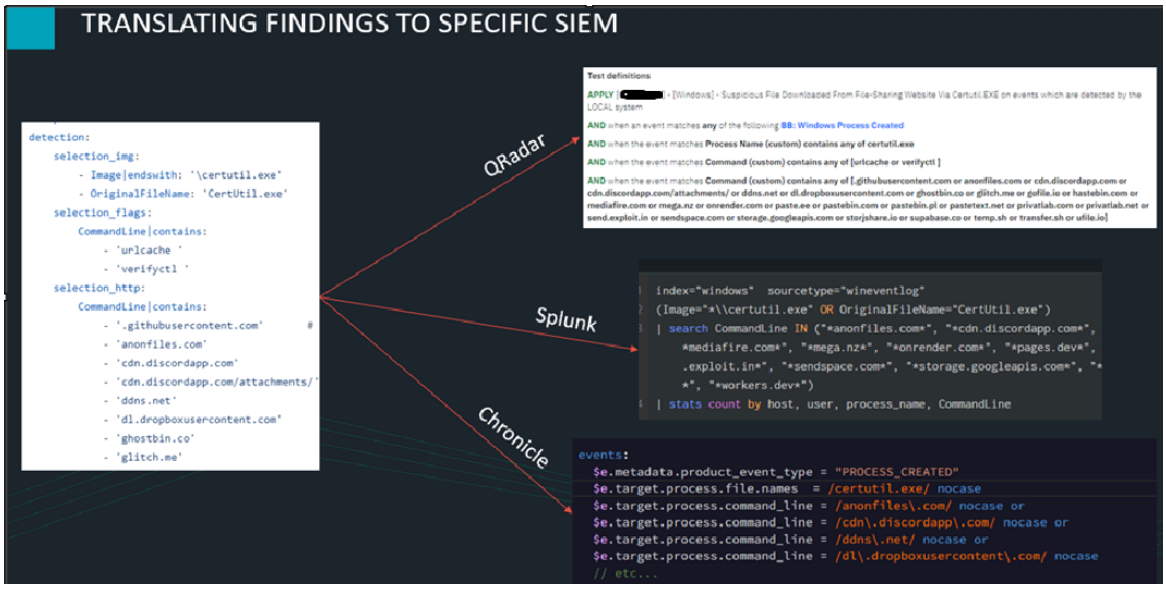
TRANSLATING INTO DETECTION LOGIC

```
detection:
    selection_img:
        - Image|endswith: '\certutil.exe'
        - OriginalFileName: 'CertUtil.exe'
    selection_flags:
        CommandLine|contains:
            - 'urlcache '
            - 'verifyctl '
    selection_http:
        CommandLine|contains:
            - '.githubusercontent.com'        #
            - 'anonfiles.com'
            - 'cdn.discordapp.com'
            - 'cdn.discordapp.com/attachments/'
            - 'ddns.net'
            - 'dl.dropboxusercontent.com'
            - 'ghostbin.co'
            - 'glitch.me'
```

TESTING IN LAB

- GENERATING ON-DEMAND TEMPORARY VIRTUAL MACHINE
- PERFORMING THE ACTIVITY THAT THE DETECTION LOGIC ATTEMPTS TO CAPTURE
- ALL TELEMETRY (NETWORK, FILES, PROCESS) IS RECORDED ON ATTACKED MACHINE
- ON-DEMAND SPLUNK INSTANCE IS GENERATED, LOGS ARE PUSHED TO IT AND ARE AVAILABLE FOR BROWSING
- RECORDED ACTIVITY GETS RAN AGAINST FULL DETECTION LIBRARY FOR HISTORICAL CORRELATION


TESTING IN LAB
detection:
selection:
EventID: 4697
ServiceName|contains:
- AmmyyAdmin
- Atera
- BASupportExpressSrvcUpdater
- BASupportExpressStandaloneService
Threat: Atera RMM
EventID: 4697
ServiceName: AteraAgent
ActivityID: "C5754DD4-23AD-0000-3D4E-75C5AD23DA01"
Logsource
Event Type
EventID
Search Term
Total Events Count: 2
Event Time
Event ID
Event Description
Event Summary
2023-11-30 16:57:10.006
4697
A service was installed in the system


TRANSLATING FINDINGS TO SPECIFIC SIEM
detection:
selection_img:
- Image|endswith: '\certutil.exe'
- OriginalFileName: 'CertUtil.exe'
selection_flags:
CommandLine|contains:
- 'urlcache '
- 'verifyctl '
selection_http:
CommandLine|contains:
- '.githubusercontent.com'
- 'anonfiles.com'
- 'cdn.discordapp.com'
- 'cdn.discordapp.com/attachments/'
- 'ddns.net'
- 'dl.dropboxusercontent.com'
- 'ghostbin.co'
- 'glitch.me'
QRadar
Test definitions
Splunk
index="windows" sourcetype="wineventlog"
(Image="*\\certutil.exe" OR OriginalFileName="CertUtil.exe")
| search CommandLine IN ("*anonfiles.com*", "*cdn.discordapp.com*", *mediafire.com*", "*mega.nz*", "*onrender.com*", "*pages.dev*", .exploit.in*", "*sendspace.com*", "*storage.googleapis.com*", "* *", "*workers.dev*")
| stats count by host, user, process_name, CommandLine
Chronicle
events:
$e.metadata.product_event_type = "PROCESS_CREATED"
$e.target.process.file.names = /certutil.exe/ nocase
$e.target.process.command_line = /anonfiles\.com/ nocase or
$e.target.process.command_line = /cdn\.discordapp\.com/ nocase or
$e.target.process.command_line = /ddns\.net/ nocase or
$e.target.process.command_line = /dl\.dropboxusercontent\.com/ nocase
// etc...

לקחים חשובים שלמדתי בתפקיד

בשלב מסוים, כל תוכנית אבטחה מצליחה לפתח את היכולת "לזהות" אירועים מזיקים במערכות שלה. אך הדבר יוצר עומס של ניתוח ידני ואסקלציה של הממצאים, וזה תהליך קשה לביצוע מוצלח. לאחרונה בקריירה שלי, אני מתעניין יותר ויותר ב**איך** צוותי אבטחה טובים מתפקדים.

הדבר אפשר לי חשיפה מרתקת לאיכויות השונות של גישות לגילוי חדירות, הן הטובות והן הפחות טובות, ואלו הן הדעות שלי לגבי המאפיינים של צוותי Detection מתפקדים ברמה גבוהה:

צוותים נהדרים לא פותרים בעיות זיהוי עם אנליסטים

כאשר יש צורך באדם שיקבל התרעה ידנית, יבצע הקשר, יחקור אותה ויסיר את האיום - **מדובר בהכרזה על כישלון**. מוצרים חדשים מציעים הרבה אפשרויות לניהול עומסים אלו ולהפחתת עייפות האנליסטים הנובעת מכך בצורה משמעותית.

צוותים חלשים יותר רואים את התהליך הידני והמכאני הזה בתור נכס, במקום לזהות אותו כסימפטום לבעיה. לעומת זאת, צוותים חזקים יותר לא מתמקדים בפיתוח SOC מגודר שמוביל אותם לנתיב התרעות לניתוח ידני, אלא מעדיפים שהאדם לא יהיה חלק מהפתרון כלל. משמרות כוננות של קבלת התרעות מאפשרות להמשיך לגייס מספר קטן של מהנדסים מיומנים במקום הרבה אנליסטים. האתגר הוא להבטיח שמשמרות אלו לא יהיו עמוסות בזבל.

צוותים נהדרים מודעים לאיפה ולאיך עבודת Analysis נוצרת

אדם אחד, עם המנוף הנכון, יכול להרים את המשקל של מספר אנשים. בדומה לכך, אדם אחד יכול לייצר עבודה עבור רבים אחרים. נקרא לזה מנוף העבודה. אפשר לראות אבטחת מידע במונחים של עבודה אישית המתחלקת לגילוי סיכונים ולצמצום סיכונים. בנייה והרס. התרעות וניתוח. גילוי סיכונים בדרך כלל מייצר הרבה עבודה לצמצום הסיכון.

בדרך כלל קל יותר לגלות פגיעות מאשר לתקן אותה. לדוגמה, שעה אחת של Threat Hunting שעשית בעצמך יכולה להטיל שבועות של עבודת תיקון על מישהו אחר. עבודת הערכת סיכונים היא גם דומה. התוצר של הערכת סיכונים (שנעשית על ידי מעטים) צריך לייצר מפת דרכים ארוכה ויסודית של פרויקטים לצמצום הסיכון (עבור רבים). החוק הזה תקף גם בזיהוי ובהתראות.

הזמן המוקדש ליצירת חוק זיהוי באיכות נמוכה ייצר ככל הנראה כמות משמעותית של עבודה למי שמגיב להתרעות שמגיעות לאחר מכן. אדם אחד ומערכת זיהוי גרועה עלולים להציף אחרים עם חוקים רעים ורעש מיותר. באותו אופן, אדם אחד יכול להציף את מערכת ה-Logging המרכזית עם נתונים חסרי תועלת, בהנחה שמישהו אחר יפרש אותם או שיקנו מוצר AI קסום שיעניק להם ערך כלשהו.

תשתית זיהוי טובה יותר מפותחת על ידי מהנדסים שמודעים מאוד ל"חוסר האיזון" הזה שנוצר בין גילוי/שבירה, ועושים את המיטב כדי להימנע מיצירת עבודה מיותרת בהמשך. הם יגבילו את זרם הסיכונים הנכנסים כדי שיהיה להם סיכוי טוב יותר לצמצם את מה שכבר גילו.

הסרת ההפרדה בין מהנדס לאנליסט תסייע להימנע מתמריצים סותרים בנוגע ליצירת חוקים וסגירת התראות. קיום משמרות כוננות על בסיס רוטציה יגרום לצוות לעבוד עם המערכות שהם עצמם יצרו. כל המשתתפים יהיו אחראים על גורלם, ולא ירגישו קורבנות של השפעות חיצוניות. תחושת חוסר שליטה זו היא תורם משמעותי לשחיקה ב-SOC, כיוון שאנשים לא מרגישים שהם שולטים בעבודה שלהם.

צוותים נהדרים שואבים מתרבות ההנדסה של הארגון

רוב כלי האבטחה שמציעים ספקים מאפשרים יצירת חוקים דרך ממשקים גרפיים עם שדות טקסט, לרוב בידי מספר קטן של אנשים. גישה זו מדכאת ביקורת עמיתים. מנוף העבודה מחמיר לרעה כאשר החוקים עוברים ללא ביקורת לקבוצה אחרת או לצוות כוננות, ללא לולאת משוב או בקרת איכות.

ביקורת עמיתים היא בעלת ערך כאשר תשתית הזיהוי מאפשרת זאת. במקרה הזה, מטרת ביקורת העמיתים אינה בהכרח למנוע קוד עם באגים, אלא לשמור על סטנדרט גבוה לגבי מה הופך חוק לחוק טוב, וכך למנוע עומס מיותר בקצה השני. קל יחסית ליישם זאת בתשתית זיהוי המשתמשת ב-Elastalert. מכיוון שכל החוקים שלה מבוססים על קבצי yml. Splunk יכול לקרוא תצורות התרעה מקובץ savedsearches.conf.

תשתית זיהוי באיכות גבוהה צריכה להתנהל במאגרים ולפעול לפי סטנדרטים הנדסיים מוכרים. רוב התשתיות עוברות לתצורת Configuration-as-Code בכל מקרה, ולכן אבטחה אינה צריכה להיות יוצאת דופן. אחריות על פיתוח החוקים משתפרת, ואפשר גם לבצע בדיקות יחידה (Unit Testing) ובדיקות תחביר (Linting) באופן מרכזי.

צוותים נהדרים מכינים את האנליסט עם כמות המידע המירבית שניתן

התרעות שנשלחות לאנליסט ללא כל סוג של עיבוד מקדים יהיו פגיעות מאוד לפתרון False positive וישחקו את האנליסט בהמשך. יש כמה דרכים לשקול סטנדרטים לעיבוד מקדים ולהתרעות.

יש להעשיר התרעות. זה מתאר סטנדרט של פירוט שבו התרעה מביאה מידע נוסף לאנליסט מבלי לדרוש עבודה נוספת. זה עוזר להימנע מ"גיהינום טאבים" שבו אנליסט צריך להיות מחובר לכמה כלים כדי לעקוב אחרי אירוע, רק כדי לדעת מה קורה.

לדוגמה, אפשר לכלול צילום מסך של הפישינג, לבדוק מידע מרשימות שחורות של דפדפנים, לענות אוטומטית על השאלה "כמה מחשבים אחרים ביקרו באתר הזה" מתוך נתוני DNS או netflow, וכדומה.

התרעה אינה שימושית אלא אם היא נושאת את כל ההקשר שהאנליסט יצטרך כדי להבין האם להכריז על אירוע או לא. צריך להפעיל אוטומציה שמביאה מידע תואם, כולל קטעי לוג, תרגום מזהים או שמות עובדים, שמות מחשבים, דעות ממודיעין איומים, וכו'.

יש למזער, ככל האפשר, באופן אוטומטי. לדוגמה, חוקים יכולים להפעיל מיד צעדי מזעור, כמו כתובת "דווח על פישינג" שמפעילה תהליך "האם הזנת את פרטי ההתחברות שלך" שמוביל לנעילת החשבון או איפוס סיסמה. זה יפחית את פעולות המעקב של כונן, את זמן הטיפול באירוע ואת זמן התגובה הממוצע.

בנוסף, ההעשרה הקודמת עשויה לאפשר לנו לשקול מחדש את חומרת ההתרעה ולסגור אותה אם היא לא עומדת בסטנדרטים.

צוותים נהדרים יעדיפו להדריך Threat Hunts, מאשר להפעיל כוננים

כפי שחזרתי ואמרתי לאורך המאמר הזה, קל ליצור חוק שמעמיס עבודה על מספר אנליסטים. מעבר לכך, קיימת תפיסה שגויה שכל התרעה חייבת להפוך ל"משימה" שאנליסט צריך לבדוק ולסגור. צוותי זיהוי טובים יותר מחזיקים פונקציה כלשהי של "ציד", בין אם זה סבב כוננות או מפגש האקאתון עם פיצה ובירה.

אם צוות האבטחה שלכם מספיק מתקדם כדי לבנות תוכנית זיהוי, אני חושב שציד חצי-מונחה באופן קבוע הוא קריטי מספיק כדי שיהפוך לרשמי כחלק מהתוכנית. במקום לשלוח את כל ההתראות לאנליסטים לחקירה בסדר FIFO (ראשון נכנס, ראשון יוצא), שקלו להפנות חלק מהחוקים לציד שוטף וחוזר שמתקיים על בסיס סבב קבוע.

אם צייד לא ימצא את ההתראה מועילה, הוא יהיה מוכן היטב לספק משוב על העשרה נוספת, מזעור אוטומטי, דרכי אסקלציה אחרות, או מחיקה ושכתוב של החוק.

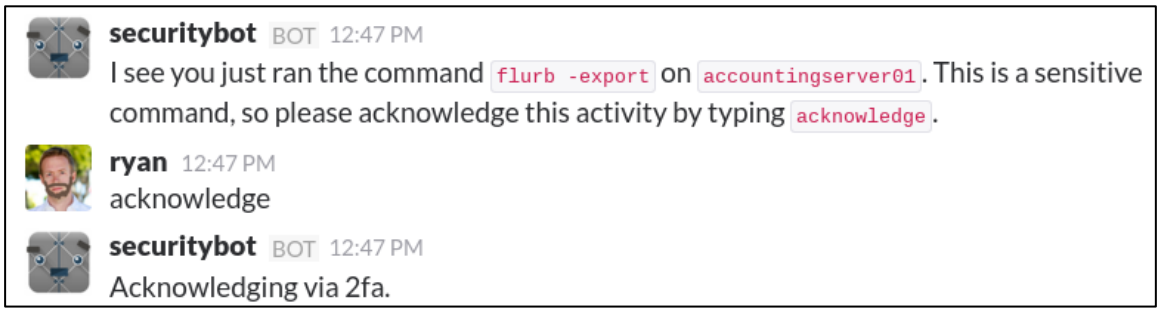
לצוותים נהדרים יש עמיתים נהדרים לעבודה שיכולים לעזור להגיב להתראות

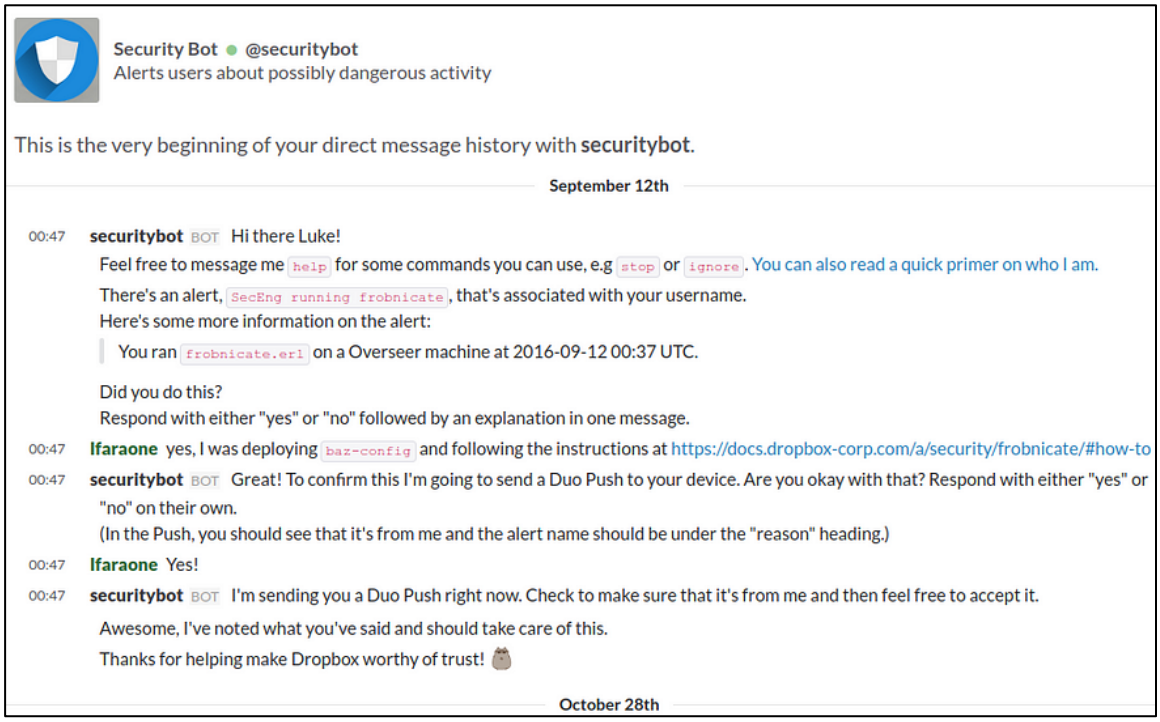
שימושי לשלב עובד בחקירות מסוימות אם הוא יכול לספק הקשר במהירות רבה יותר מאשר חקירה ידנית או חקירה המבוססת על כלים. לעובד יש מנוף עבודה חזק יותר מאשר לאנליסט כאשר התרעה כוללת אותו. לדוגמה, דמיינו בוט Slack שיכול, אוטומטית או בהנחיית כונן, לשאול את העובד שאלה כמו:

- (כאשר הם התחברו ל-VPN ממיקום או מדינה חשודה): "היי, אתה בנסיעה"?
- (כאשר הם עשו משהו בניגוד למדיניות): "היי, היית צריך להשתמש ב-sudo בסביבת Production?"

מנוף העבודה כאן הוא משמעותי, בעיקר משום שלעובד בדרך כלל יש יותר מידע מאשר לאנליסט. כמובן, יש מקרים חריגים שבהם הדבר פחות מתאים, כמו במקרה של איום פנימי. אך באופן כללי, דרך האסקלציה הזו להוספת הקשר לחוק או התראה היא רעיון מצוין.

יש הרבה דוגמאות באינטרנט בנושא הזה, כמו:





צוותים נהדרים לא מעבירים התראות מאחד לשני

הסיבה לסגירת התרעה צריכה להיאסף לצורך מדידה. זהו היבט חשוב בלולאת המשוב שמקרבת אותנו לכיוון של התרעות איכותיות. יש לפתח שפה שתאפשר להבין מדוע התרעה נסגרה. לדוגמה:

- האירוע לא זיהה את מה שהיה אמור לזהות.
- האירוע שזוהה נראה זדוני, אך לא היה.
- האירוע זיהה התנהגות זדונית, אך זה לא הצדיק התרעה.

ישנן סיבות רבות לסגירת התרעה, ולולאת המשוב הזו תסייע ללמד סטנדרטים שיוכנסו לתהליך הפיתוח באמצעות בדיקות תחביר או ביקורת עמיתים.

העברת הנתונים הללו לכונן הבא תעזור למנוע שחיקה, שכן הכונן החדש יונחה על ידי המידע שנמסר מהכונן הקודם בנוגע לדיוק ההתרעה. התרעה גרועה לא צריכה להשפיע על יותר מרוטציה אחת.

צוותים נהדרים לא מחכים לאירוע או לצוות אדום כדי לראות איפה הם עומדים

מערך הבעיות של זיהוי חדירות הוא אתגר אינטגרציה מורכב. אז מדוע לא יותר צוותי זיהוי מתמקדים בבדיקות אינטגרציה? זהו תחום מרתק בתחום גילוי חדירות, להתייחס לזיהוי באותו אופן שבו מתייחסים ל-Pipeline הנתמך בפלטפורמות CI/CD כמו Jenkins. יש כמה דרכים שבהן ניתן לבדוק את זה כיום:

תרגילי שולחן (Tabletops): פשוט מאוד לבדוק את צינורות הזיהוי המרכזיים עם תרחישים פשוטים. למשל, "אני אעקוב אחרי הלוגים בזמן שאתה מתקין סקריפט זדוני".

Red Teaming: זהו תהליך יקר. ניתן לעשות זאת אולי פעם או פעמיים בשנה. זה שימושי יותר עבור בדיקת תגובה לאירועים מאשר כבדיקת אינטגרציה. עם זאת, זה יכול לעזור לעורר יצירתיות ביצירת חוקים.

כלים לזיהוי: במקרה זה אנחנו כותבים תוכנה שתתקוף ישירות את מנגנוני הזיהוי שלנו, ממש כמו בדיקת יחידה אמיתית. רבים מהם יהיו פשוטים לכתיבה, ויש חברות שיש להן מוצר בדיוק לזה, כמו AttackIQ.

השימוש בגישה הזו לכתיבת כלים ייעודיים שתוקפים ישירות את תשתית הזיהוי שלנו מאפשר לכם לבדוק בצורה שיטתית ואוטומטית אם החוקים והמערכות שלנו עובדים כפי שמצופה מהם תחת מצבים שונים, בדומה לבדיקות אינטגרציה במערכות פיתוח תוכנה.

תעדוף נכון של Detection Engineering

Detection Engineering הוא מושג חדש יחסית. הוא מכיר במורכבות מערך הזיהוי ובנפח העבודה שהוא יוצר במהירות עבור אחרים. הנדסת זיהוי היא נקודת מפגש בין מודיעין, יכולות טכניות, הזדמנויות פלטפורמה ועומס ניהולי. אני כותב על החלק האחרון, העומס הניהולי.

הנדסת זיהוי היא יותר מקוד ופלטפורמות; זו מודעות לזרימה של עבודה חדשה שנוצרת פתאום על ידי המאמצים שלנו.

הסדירו את ענייני ה-Logging שלכם

החזון לטווח הארוך עבור לוגים מרכזיים צריך להיות אסטרטגיית זיהוי, אך בטווח הקצר יש להתמקד בעיקר ביכולת ביצוע שאילתות ובמענה לצרכי הלוגים של צוותים אחרים. בשלב זה, לא צריכים להיות תפקידים ייעודיים להנדסת זיהוי.

ראשית, עלינו להיות בררנים ולאסוף את הלוגים המינימליים הנדרשים כדי להגיב בצורה אמינה לתרחישים נפוצים. איסוף נתוני התחברות במערכות הארגוניות ובמערכות ה-Production, יחד עם לוגים של שימוש בתשתיות (לוגים של IaaS) בדרך כלל מקבלים עדיפות כאן.

בנוסף, ניקח כל לוג אפליקטיבי ששאר הארגון משתמש בו לצורך פתרון בעיות, בדרך כלל עד רמת המוצר. בשלב זה, נימנע מגישה חמדנית לגבי לוגים. אין לנו צורך לאחסן כל קריאת מערכת (syscall) על כל שרת (לפחות לעת עתה!). ייקח לא מעט זמן עד שלגישה חמדנית תהיה תשואה כלשהי על ההשקעה.

שיחות אלו כופות דיונים על תקציב, אחסון, שאילתות, שימור והעברת לוגים. עם זאת, עצרו את השיחה סביב התראות. אל תלכו רחוק מדי בנושא אסטרטגיות התראה מתוחכמות בשלב זה. לפחות, אל תחזיקו בדעות חזקות בנושא זה לזמן מה, ואל תתחייבו למצב שבו אנו מעירים את כולם ויוצרים מרכז SOC או סבב תורנויות לקריאות חירום.

שלב זה תומך בגורם השני באבטחת מערכות. הוא מגביר את קצב הפיתוח, יוצר תמיכה מרובת בעלי עניין, לא יעמיס על משימות זיהוי, ומעניק יכולת תגובה חשובה תוך שיפור ההבנה של סביבת העבודה.

כעת, לכל הפחות, תוכלו להבין מה קרה בתקרית. הפסקה טבעית מתרחשת כאן לפני שנמשיך הלאה לעבר אסטרטגיית זיהוי.

השקיעו זמן בהקשחות, ותכננו חזרה בעתיד לעבר Detection

לזיהויים מותאמים אישית יש תשואה גבוהה יותר בסביבה עם כמות מסוימת של בקרות והפרדה. צוותים קטנים לא צריכים להגדיר בצורה רשמית את אסטרטגיית הזיהוי והתראות שלהם מוקדם מדי. הבקרות שתוכלו להכניס ישפיעו באופן משמעותי על מה שתגלו. במקום זאת, יש לקבוע את ציון הדרך שלכם כיכולת לבצע שאילתות ושיתוף פעולה לפני שתתחילו בהתראות וזיהויים מותאמים אישית.

בשלב זה אנחנו לא מאיישים צוות ייעודי להנדסת זיהוי. עדיין יש עבודה רבה הנדרשת לתמוך ביצירת סביבה צפויה שמבוססת על משתנים קבועים (Invariants). התקדמות בזיהוי תתבצע היטב כאשר משתנים קבועים אלו מתחילים להיווצר.

משתנים קבועים הם ציפיות מוצקות לגבי איך אתם מצפים שהסביבה שלכם תתנהג. זה יכול לכלול הפחתת הרשאות, צורות ריכוזיות של אימות זהויות ותכנון מכוון של הזדהות עובדים.

ישנם כמה יוצאים מן הכלל: כלי אבטחה מוקדמים עשויים להגיע עם יכולות זיהוי והתראה מובנות. כלים אלו הם די בסדר, בדרך כלל ניתנים לניהול, כל עוד אינכם שקועים יתר על המידה בניסיון להאיץ את כמות העבודה שהם מייצרים. הימנעו מהפיתוי ליצור תורנות חירום עם SLA מחמירים ברגע שזיהויים הופכים זמינים.

בשלב זה, הסתכלו על זיהוי בספקנות. דמיינו תוכנית זיהוי בטרם עת כמו באר כבידה. משימות החיזוק (hardening) שקודמות לתוכנית זיהוי יהפכו את התוצאה הסופית למיוחדת מאוד. אבל עד אז, זיהוי מוקדם מדי יגזול זמן וימנע מכם להגיע לתוצאה הרצויה.

יישמו רק זיהויים והתראות באיכות גבוהה

ראשית, נרצה ללמוד וליישם את ה-ADS. זו רמה גבוהה מספיק לאיכות זיהוי והתראות. מי שמתעורר בעקבות התראה צריך לצפות לסטנדרטים המחמירים שמכתיבה ה-ADS מאחורי הזיהוי שגרם להתראה.

אנו רוצים למנוע הצטברות של התראות בינוניות. התחילו בתיעוד ושיתוף פעולה כבר מההתראה הראשונה. ההתראה הראשונה שלכם היא התראת הייחוס: מתועדת בצורה מושלמת, עם מעט מאוד או ללא התראות שווא, תרחישי תגובה ברורים וניתנת לפענוח על ידי כל מי שמקבל אותה. השקיעו באופן יוצא דופן בהתראה הראשונה והשוו כל התראה אחרת אליה. זו ההתראה המובילה של התוכנית שלכם.

ה-ADS מציעה הנחיות לגבי מה צריך לתעד לפני שהתראה נכנסת למערכת זיהוי ב-Production.

הסט הראשון של ההתראות צריך להתמקד בזיהוי משתנים קבועים (Invariants). התשתית שלכם לעולם לא צריכה להיתקל במצבים אלו בהתבסס על הנחות שאתם מגבים עם בקרות ומדיניות.

דוגמה: שירות ניהול הסודות לעולם לא צריך להיפתח.

הסודות של ה-Prodcution עשויים להיות מופרדים באופן עמוק עם גישה מוגבלת. כן, יש סיבות לפתוח כספת, אבל יכולה להיות ציפייה שכל פתיחה של כספת תפעיל תקרית, מה שהופך את ההתראה לכשרה.

דוגמה: אין שימוש מרוחק ב-API של IaaS.

אתם (בתקווה) דורשים שכל פעילות מאומתת תגיע מתתי רשתות מסוימות. אם התשתית שלכם תומכת בכך) מדיניות IAM, הפרדת רשתות, שרתי ביניים (שימוש לא מורשה מחוץ לרשתות אלו יעיד חזק על דליפת Credentials), מדוע משתמשים במפתחות מ-VPN אקראי? (שוב, זו התראה לגיטימית אם המשתנה הקבוע תומך בכך. אחרת, יש עבודה בסיסית שעלינו לבצע.

דוגמה: אף אחד לא צריך לגעת ב-honeytokens.

כן, זה ברור. אין לגעת ב-honeytokens מכיוון שהם ממוקמים בצורה חכמה מחוץ למסלול המפתחים הצפוי. לאחר מכן, יש להבטיח את יכולת התגובה של הפלטפורמה שלכם.

יכולת תגובה נבנית כשניתן להכניס אינדיקטור (IoC) כמו hash זדוני של קובץ הרצה, דומיין או כתובת IP יוצאת, או TTP (טכניקה מסוימת) למערך הזיהוי שלכם, והתראה על כך מיידית לצוות התגובה. זה הכל. פלטפורמת הזיהוי שלכם מספקת עין פקוחה בזמן תגובה לתקריות באמצעות מעקב אחר אינדיקטורים ידועים.

לבסוף, כל זיהוי שאתם רוצים להימנע מלקדם להתראה יכול להיחשב כנתוני ציד. חלק מנתוני הציד אולי לעולם לא יקודמו להתראה, אך הם עדיין עשויים להיות שימושיים. דוגמה נהדרת לציד חוזר היא הרצת פקודת sort | uniq על כל ה-User-Agents שהופיעו בתשתית לאורך זמן. אם מופיע סוכן משתמש חדש, הוא או יחשוף מה שקורה בתשתית או יגלה כלי ממשק גרפי יוצא דופן שמשתמש בו תוקף כדי לגלוש

בתשתית, כמו Cloudberry, Elasticwolf או s3 browser. User-Agents חדשים לא שווים התעוררות בלילה, אבל הם כלי אינפורמטיבי עבור מי שמחפש לצוד דברים מוזרים. (אה, תראו, מוצר נתונים חדש נכנס ל-Prod).

אנחנו עדיין לא מוכנים לצוות ייעודי להנדסת זיהוי. אנחנו עדיין נאבקים על כוח אדם כללי. ההגבלה הזו מכוונת, כמו שתבינו בקרוב.

אימוץ של גישת הנדסה לזיהוי באופן מלא

הנדסת זיהוי מלאה נכנסת לתוקף בצורה האפקטיבית ביותר כאשר ניתן לשלוט בקצב העבודה שהיא מייצרת, להאיץ או להאט אותה, מבלי לשבש את המטרות האחרות שלכם. כל שלב נלקח במכוון כדי לנהל את היקף העבודה הכרוך בכך. הנה איך אנחנו מאיצים:

- הכנסת זיהויים ומוצרים שנוצרו מבחוץ.
- קידום רעיונות מאומתים מציד (hunt) לזיהויים הניתנים להתראה.
- גיבוש תרחישים מצוותים: התקפה, מודיעין איומים, ציות ובדיקות סיכונים.

...או מאטים:

- הורדת דרגת התראות שמייצרות False Positives או עבודה מיותרת.
- עריכת רטרוספקטיבה כפויה על False Positives עם יוצרי ההתראה.
- שיתוף הערות בביקורת עמיתים על יצירת התראות.

לא ניתן להגיע לשלב הזה לפני שנרגיש ביטחון בניהול קצב העבודה שהמערכת הזו תייצר באופן בלתי נמנע.

תיעדוף גיוס עבור Detection Engineering

מדוע הייתי כל כך שמרן בנושא כוח אדם?

ראשית, ייתכן שהנדסת זיהוי לעולם לא תדרוש יותר מאדם אחד במשרה מלאה. צינורות הזיהוי (Detection Pipelines) כבר לא מהווים את הסיוטים שהיו בחברות צעירות יותר, במיוחד בחברות שהן מבוססות ענן באופן מלא. כלים הופכים להיות פשוטים יותר.

יש לאייש צוות הנדסת זיהוי רק לאחר התקדמות משמעותית בבסיסים. הכלים לניהול סיכונים ניהוליים צריכים להיות גם הם בשלים. יש סיבות רבות להרחיב את כוח האדם בתחום הזה, אבל לא אם זה ייצור באופן מיידי סיכון ניהולי.

איזה סיכון ניהולי?

אם סביבה ניהולית בריאה עדיין לא הוקמה, הזיהוי ייצור מיד את סוג הלחץ הלא נכון. בתגובה, לא תרצו לבנות מבנה ניהול מקביל עם ניהול משימות נפרד, רמות חומרה שונות, דפוסי אסקלציה נפרדים וכו.'

ניהול משימות ותיעדוף הם אתגרים בכל מקום. מצאו את הדיונים הקיימים בארגון שלכם בנושא זה, שתפו פעולה ובנו עליהם במקום להמציא אותם מחדש. דברו עם מנהלים ונסו להבין איך העבודה זורמת בתוך החברה. הנדסת זיהוי יוצרת המון עבודה. הגדלת כוח האדם בשלב מוקדם תחריף את הבעיה מהר יותר ממה שתוכלו לתקן את בעיית הניהול הבסיסית.

הנדסת זיהוי סובלת מכמות עבודה ניהולית רבה יותר ממה שאנשים מבינים.

ניהול הוא האתגר המרכזי של תוכנית זיהוי. לכן, יש להעדיף את הבסיסים מחוץ לתחום הזיהוי תחילה. שימוש במשתנים קבועים (Invariants) מפשט את תחום הזיהוי הכולל ובונה ביטחון שהאסטרטגיה הניהולית שלכם תואמת את הפילוסופיה הניהולית הקיימת של החברה.

הדגימו את היכולות שלכם עם אבני הדרך ההדרגתיות שתיארתי. לאחר מכן, ראו כמה עבודה עליכם לנהל ואם תוכלו לכוון אותה ביעילות. אם עדיין יש צורך, אז איישו כוח אדם נוסף.

אולי תחשבו אחרת. ייתכן שתחשבו שמומחי הנדסת זיהוי יכולים להגיע ולבנות תוכנית זיהוי מלאה מאפס. זו גישה הפוכה. כן, ייתכן שזה אפשרי עם האנשים הנכונים, אבל סביר יותר שדפוסי פעולה שגויים ייווצרו עם כל מושג ניהולי חדש שהכנסתם לפתע, מבלי ששמתם לב שהוא חופף עם מושגים קיימים.

מדוע זאת גישת התיעדוף הנכונה?

העבודה בתחום הזיהוי יכולה להתגלגל ולהפוך למורכבת יותר ויותר; הנה הסיבה.

זיהוי היא בעיה שאני מתאר כאחת כזו שנראית ניתנת לפתרון, אך באופן מטעה. מספר גורמים מובילים אותי למסקנה הזו. מגוון של דפוסי פעולה שליליים (Anti-Patterns) מופיעים כאשר הופכים את הזיהוי למרכז מוקדם מדי בתוכנית אבטחה:

1. **עבודת זיהוי פועלת בצורה עצמאית**: פלטפורמת זיהוי לרוב נלקחת מתוך תקציב האבטחה בלבד ולא מצריכה אישור נרחב מהנדסת החברה. השבתה של פלטפורמת זיהוי לעיתים רחוקות תשפיע על Production, והארגון ההנדסי הגדול יותר בקושי יידע על קיומה.

2. **זיהוי מצריך פחות מומחיות ב-Production**: הרבה עבודה בזיהוי יכולה להתבצע ללא צורך במומחיות מלאה בתשתית, במוצר או בערמת היישומים. מהנדס זיהוי יכול להתמחות אך ורק בבעיות זיהוי ועדיין להיות מועיל בהרבה חברות. יכולת ההעברה הזו מאפשרת התקדמות בתחום הזיהוי מבלי להיכנס לעומקים שבהם נמצאים מהנדסי ה-Production.

3. **זיהוי מסמן ניצחון**: הידיעה שתפסתם מתקפה בזמן אמת היא סיבה מיידית לחגיגה. הציפייה לניצחון הזה היא אמיתית ומוסיפה הנאה לעבודה. אבטחה היא לרוב עבודה חסרת הכרת תודה וללא ניצחונות ברורים. זיהוי אינו כזה - אם תפסתם אותם, ניצחתם.

כל זה עשוי להישמע טוב כל כך שתחשבו להפסיק את כל שאר העבודות ולהתמקד אך ורק בזיהוי. זיהוי הוא אזור מושך ובטוח לעבוד בו, עם תרחיש ניצחון ברור. דמיינו ערוץ עבודה גרילה שבו צוות אבטחה שאינו מוצא מנופים או תקציב לעבודה על פתרונות ייצור יכול לזוז מהר בתחום מבודד היטב.

הבעיות הארגוניות לרוב דוחפות את הזיהוי לקדמת הבמה. יש מצבים מוצדקים לחלוטין שבהם זיהוי הופך לאסטרטגיה עיקרית. ייתכן שאין אפשרות או יעילות להקים משאבים הנדסיים ייעודיים ותפקודיים לכל תחום בעסק מורכב. גבולות הנדסיים עשויים להוציא את הסיכונים לחלוטין מתחום האחריות של צוות האבטחה, כך שהזיהוי הופך לאחת הדרכים הנדירות בהן צוות אבטחה יכול לתרום להפחתת הסיכון.

עם זאת, **ללא סיבה מוצקה**, התמקדות מוגזמת בזיהוי לוקחת משאבים מעבודת ההפחתה, שדורשת שיתוף פעולה נרחב, השלכות על ייצור ותיאום טכני עם המפתחים. ההנהלה עשויה להישאב לתוך זה, והגורל של הצוות עשוי להשתנות לאט לארגון הנדסי מוצל שמנותק משאר ההנדסה.

אל תעשו את הטעות הזו אם אתם בונים צוות אבטחה מהיסוד. העדיפו לעבוד עם המהנדסים על פתרונות הפחתה שווי ערך, והתייחסו לזיהוי כרכיב אחד מהתמונה הכוללת. אל תתנו לזיהוי להפוך לחלופה מקצועית בטוחה לעבודה על פתרונות ייצור, שדורשת לעיתים יותר אחריות ישירה מצידנו.

סיכום ומילות סיום

בעולם הפרוע של הנדסת זיהוי, למדנו שהצלחה לא מגיעה רק מפריסת כלים נוצצים או יצירת ערמות של התראות. במקום זאת, המפתח טמון בבניית גישה מחושבת ומדרגית שלא מציפה את צוות ה-SOC בהתראות שגויות או יוצרת כאוס. מהבטחת תשתית חזקה על בסיס משתנים קבועים, ועל ידי קפיצה הדרגתית לזיהויים מותאמים אישית, המסע הוא עניין של איזון, שיתוף פעולה, ולהיות פרנואיד *בדיוק במידה הנכונה*.

אז, קורא יקר, כשאתה יוצא אל השקיעה, זכור: לזהות איומים זה מגניב, אבל לא על חשבון השפיות של הצוות שלך. התחל בקטן, אוטומציה כשאפשר, ובבקשה - אף אחד לא רוצה להתעורר בשלוש לפנות בוקר בגלל אזעקת שווא.

על המחבר

דניאל קויפמן, עובד בתור Detection Research/Engineering ב-MSSP עולמי. עם ניסיון בבנק Fortune 500. מוזמנים להתחבר איתי!