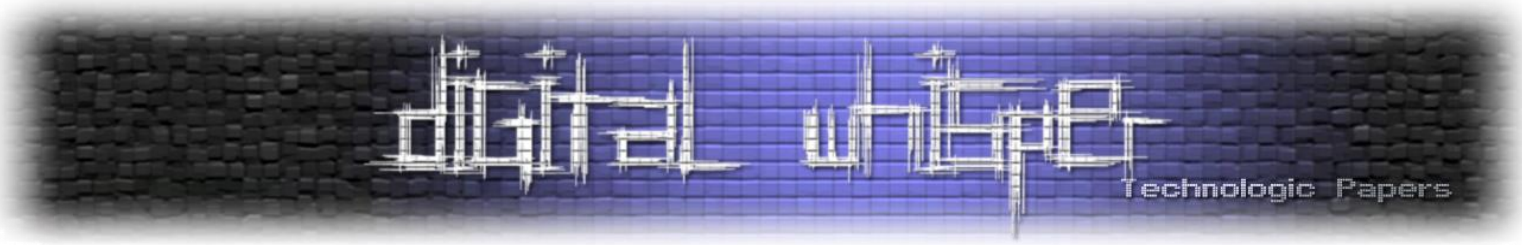


מהגנה למודיעין: בניית Honeypot מבוסס Cloudflare לזיהוי והטעיית תוקפים

מאת גיא תותחני

תקציר מנהלים

במהלך מספר ימים של ניטור שגרתי החלו להופיע בלוגים בקשות חוזרות לנתיבים כגון /wp-login.php, /.env ו-xmlrpc.php/. במבט ראשון הן נראו כמו רעש רקע רגיל של האינטרנט, אך ניתוח מעמיק יותר חשף פעילות אוטומטית רחבת היקף שבוצעה באמצעות תשתיות ענן ציבוריות ברחבי העולם. במקום לחסום את התוקפים מיד, הוחלט לנצל את ההזדמנות כדי להבין כיצד הם פועלים, אילו חולשות הם מחפשים ואילו כלים הם מפעילים. ההחלטה הזו הפכה בסופו של דבר לפרויקט שקיבל את השם **Operation GhostEdge.**

מאמר זה מתעד את פרויקט Operation GhostEdge. הקמת מערכת הגנה, זיהוי והטעיה (Deception) עבור אתר אינטרנט ציבורי, אשר נבנתה כולה על גבי תשתית הקצה של Cloudflare. הפרויקט נולד מתוך תצפית פשוטה אך נפוצה: כל שירות Web החשוף לאינטרנט, ללא קשר לגודלו או לחשיבותו, הופך כמעט מיד ליעד של תעבורה עוינת אוטומטית. בקשות חשודות מגיעות מסורקים, בוטים ותשתיות ענן ציבוריות, ומחפשות בשיטתיות חולשות מוכרות, ממשקי ניהול חשופים וקבצים רגישים.

מטרת המאמר היא לתאר את מלוא מחזור החיים של הפרויקט: החל מזיהוי הפעילות העוינת וניתוחה, דרך מידולה במונחי איום (Threat Model) ומיפויה ל-MITRE ATT&CK, וכלה בתכנון והטמעה של ארכיטקטורת הגנה רב-שכבתית. הארכיטקטורה משלבת בין סינון ב-Edge, מנגנון Threat Scoring דינמי, שכבת Honeypot המדמה שירותים פגיעים ומנגנון Tarpitting להאטת סורקים. הדגש המרכזי של הפרויקט אינו חסימה בלבד, אלא הפיכת התקיפה עצמה למקור Threat Intelligence מבצעי.

המאמר נכתב עבור קהל יעד מגוון. חשוב להדגיש כבר בפתח הדברים כי המערכת נבנתה למטרות הגנה, מחקר ולימוד בלבד. לא נעשה בה שימוש התקפי, לא בוצע ניסיון לפגוע במערכות צד שלישי, וכל המידע שנאסף שימש לצורכי Detection, Threat Intelligence ושיפור יכולות ההגנה בלבד.

רקע

עבור מפעילי אתרים רבים, המפגש הראשון עם תוקפים אינו מתרחש בעקבות אירוע אבטחה משמעותי או ניסיון פריצה מתוחכם, אלא כבר בשעות הראשונות לאחר העלאת השירות לאוויר. מערכות סריקה אוטומטיות פועלות באופן רציף ברחבי האינטרנט, מחפשות שירותים חדשים, תתי-דומיינים שנחשפו זה עתה ויישומים אשר טרם הוקשחו. במקרים רבים ניתן לראות ניסיונות גישה לנתיבים מוכרים זמן קצר לאחר פרסום האתר, גם כאשר הוא עדיין אינו מופיע במנועי חיפוש ואינו מוכר לציבור הרחב.

תופעה זו אינה חריגה למעשה, היא הפכה לחלק בלתי נפרד מהמציאות של כל שירות Web ציבורי. האינטרנט המודרני מלא בבוטים, סורקים אוטומטיים ותשתיות תקיפה מבוזרות המבצעים ללא הפסקה פעולות Reconnaissance, כלומר איסוף מידע ומיפוי מטרות פוטנציאליות. חלק מהפעילות מבוצע על ידי מנועי חיפוש ושירותי אינדוקס לגיטימיים, אך חלק משמעותי ממנה מגיע ממערכות שמטרתן לאתר חולשות, שירותים חשופים וטעויות תצורה שניתן לנצל.

בניגוד לדימוי המקובל של תוקף היושב מול מחשב ובוחר יעד ספציפי, רוב הפעילות העונת המופנית כיום כלפי שירותי Web מבוצעת באופן אוטומטי לחלוטין. התוקפים אינם מכירים את הארגון, אינם יודעים מי מפעיל את האתר ולעיתים אף אינם יודעים מהו תוכן השירות. במקום זאת הם מפעילים מערכות סריקה רחבות היקף אשר בודקות מיליוני כתובות אינטרנט ביום, בניסיון לאתר מטרות פגיעות.

גישה זו מכונה Mass Scanning. מטרתה פשוטה: לבצע מספר עצום של ניסיונות זיהוי בעלות נמוכה ככל האפשר, מתוך הנחה שגם אם רק אחוז קטן מאוד מהיעדים יתגלה כפגיע, ההשקעה תחזיר את עצמה. מבחינת התוקף, אין משמעות מיוחדת לאתר מסוים כל יעד הוא עוד כתובת ברשימת הסריקה.

הסריקות האוטומטיות מכוונות בדרך כלל למטרות נפוצות אשר קיימת סבירות גבוהה למצוא אותן ברשת. בין היעדים השכיחים ניתן למצוא התקנות WordPress, ממשקי ניהול חשופים, קבצי Environment המכילים סודות מערכת, חולשות PHP ידועות, תוספים פגיעים, קבצי גיבוי שנשכחו על השרת, ממשקי מסדי נתונים וכלי פיתוח אשר נותרו זמינים בסביבת הייצור.

דפוס פעולה זה מסביר תופעה שעלולה להיראות מבלבלת במבט ראשון. במסגרת הפרויקט זוהו ניסיונות גישה חוזרים ונשנים לנתיבים כגון:

```
/wp-login.php
/.env
/xmlrpc.php
/vendor/phpunit/eval-stdin.php
/shell.php
/adminer.php
```

למרות שהאתר שנבדק כלל לא התבסס על WordPress, לא השתמש ב-PHP ולא הכיל את הרכיבים הללו, הסורקים המשיכו לבדוק אותם שוב ושוב. הסיבה לכך פשוטה: הסורק אינו "יודע" באיזו טכנולוגיה האתר משתמש. הוא פועל לפי רשימת מטרות קבועה מראש. אם אחד הנתיבים קיים ומחזיר תגובה מעניינת, הוא יסומן להמשך בדיקה או לניסיון ניצול. אם הנתיב אינו קיים, הסורק ימשיך מיד אל היעד הבא.

כל אחד מהנתיבים הללו מייצג יעד תקיפה מוכר. חלקם משמשים כממשקי התחברות, חלקם עלולים לחשוף מידע רגיש, וחלקם קשורים לחולשות אשר נוצלו בעבר בקמפיינים רחבי היקף. לכן עצם הניסיון לגשת אליהם מהווה לעיתים אינדיקציה ראשונית לפעילות סריקה או לניסיון Exploitation.

מאחורי רוב קמפייני הסריקה האלו עומד היגיון כלכלי פשוט. עלות הסריקה נמוכה במיוחד. תוקף יכול לשכור שרת וירטואלי (VPS) בעלות של דולרים בודדים לחודש, להריץ עליו כלי סריקה אוטומטיים ולכסות בתוך זמן קצר מיליוני יעדים. גם אם רק חלק קטן מאוד מהמערכות שייסרקו יימצא פגיע, התשואה הפוטנציאלית עדיין מצדיקה את המאמץ. מאפיין נוסף של פעילות זו הוא השימוש הנרחב בתשתיות ענן ציבוריות. ספקי ענן מודרניים מאפשרים הקמה מהירה של שרתים, החלפת כתובות IP בתוך דקות בודדות והרחבת פעילות בהיקפים גדולים מאוד.

במהלך החקירה המתוארת במאמר זה זוהה כי חלק משמעותי מהתעבורה החשודה הגיע מתשתיות ענן ציבוריות מוכרות, ובהן Oracle Cloud, Microsoft Azure, DigitalOcean וספקים נוספים. עובדה זו אינה מעידה בהכרח על מעורבותם של הספקים עצמם, אלא על השימוש שעשו התוקפים במשאבי הענן הזמינים להם. מעבר לניסיונות מיפוי ואיתור נתיבים, פעילות אוטומטית מסוג זה כוללת לעיתים קרובות גם ניסיונות Credential Stuffing ו-Brute Force. מתקפות אלו מתבססות על מאגרי סיסמאות וחשבונות שדלפו בעבר ממערכות אחרות, מתוך הנחה שמשתמשים רבים עושים שימוש חוזר באותם פרטי התחברות במספר שירותים שונים. כאשר ניסיון כזה מצליח, התוקף עשוי לקבל גישה לחשבון תקין מבלי לנצל חולשה טכנית כלשהי

במהלך הניטור השוטף של אתר הלקוח זוהו שלושה דפוסי פעילות מרכזיים:

- **Reconnaissance** - ניסיונות מיפוי ואיתור נתיבים ושירותים
- **Credential Stuffing** - ניסיונות התחברות באמצעות פרטי הזדהות גנובים
- **Exploit Scanning** - חיפוש פעיל אחר חולשות הניתנות לניצול

השילוב בין שלושת הדפוסים הללו הצביע על כך שלא מדובר ברעש רקע שגרתי בלבד, אלא בפעילות אוטומטית בעלת מאפיינים התקפיים ברורים. בשלב זה עלתה השאלה המרכזית שהובילה להקמת הפרויקט: האם נכון פשוט לחסום את הבקשות ולהמשיך הלאה, או שניתן לנצל את הפעילות כדי ללמוד על התוקפים, לאסוף מודיעין ולשפר את יכולות ההגנה של המערכת?

מטרות

לאחר שהתגבשה ההבנה כי האתר אינו מתמודד עם אירוע נקודתי אלא עם זרם קבוע של סריקות אוטומטיות, עלתה השאלה כיצד נכון להגיב לפעילות זו. האפשרות הפשוטה ביותר הייתה להסתפק בחסימה באמצעות חוקי Firewall ו-WAF, אך גישה זו הייתה פותרת רק חלק מהבעיה. היא אמנם הייתה מונעת חלק מהבקשות החשודות, אך לא הייתה מספקת תשובות לשאלות החשובות באמת: מי עומד מאחורי הפעילות? אילו חולשות מחפשים התוקפים? האם מדובר בקמפיין רחב היקף? ומה ניתן ללמוד ממנו כדי לשפר את ההגנה בעתיד? מתוך שאלות אלו גובשו מטרות הפרויקט, אשר השפיעו על כל החלטה ארכיטקטונית שהתקבלה בהמשך הדרך.

המטרה הראשונה הייתה **זיהוי וניתוח של פעילות עוינת**. במקום לראות בכל בקשה חשודה עוד אירוע בלוגים, המערכת נדרשה לזהות דפוסים חוזרים, להבחין בין פעילות לגיטימית לפעילות עוינת ולספק תמונה ברורה של סוגי האיומים המופנים כלפי האתר

במקביל, היה צורך **להגן על שרת המקור** ולצמצם ככל האפשר את החשיפה שלו לאינטרנט הציבורי. אחת ההחלטות המרכזיות בפרויקט הייתה לעצור חלק משמעותי מהתנועה החשודה כבר בשכבת ה-Edge של Cloudflare, עוד לפני שהיא מגיעה לתשתית האמיתית.

מטרה נוספת הייתה **למנוע ניסיונות Exploitation** של חולשות Web מוכרות. גם כאשר האתר אינו משתמש בטכנולוגיות מסוימות, תוקפים ממשיכים לחפש נתיבים, קבצים ורכיבים פגיעים. המערכת נדרשה לזהות ניסיונות אלו מוקדם ככל האפשר ולמנוע מהם להגיע לסביבת הייצור.

אולם בשונה ממערכות הגנה מסורתיות, מטרת הפרויקט לא הייתה חסימה בלבד. אחד המרכיבים המרכזיים בתכנון היה הקמת **סביבת Honeypot מבוקרת**, אשר תדמה שירותים פגיעים ותאפשר לתוקפים להמשיך בפעילותם בתוך סביבה מבוקרת. גישה זו נועדה להפוך את ניסיון התקיפה עצמו למקור מידע בעל ערך.

כתוצאה מכך הוגדרה גם מטרה נוספת: **איסוף Threat Intelligence בזמן אמת**. המערכת תוכננה לתעד מידע על כתובות IP, User-Agents, Payloads, דפוסי תקיפה, תדירות פעילות ומאפיינים נוספים אשר יכולים לסייע בהבנת האיום ובשיפור מנגנוני ההגנה.

מטרה נוספת הייתה **צמצום שטח התקיפה (Attack Surface)** של האתר. ככל שפחות שירותים, נתיבים וממשקים חשופים לתוקפים, כך קטן הסיכון לפגיעה במערכת. לכן הארכיטקטורה תוכננה כך שחלק ניכר מהאינטראקציה עם התוקפים יתבצע בשכבת ה-Edge ולא מול שרת המקור.

בסופו של דבר, כל המטרות הללו התכנסו לרעיון מרכזי אחד: להפוך את שכבת ההגנה ממנגנון תגובתי אשר רק חוסם תוקפים, לפלטפורמה המסוגלת לזהות, לנתח, ללמוד ולאסוף מודיעין מתוך הפעילות העוינת עצמה.

Threat Model

לפני שניתן להחליט כיצד להגן על מערכת, יש להבין מפני מה היא נדרשת להתגונן. מודל איום (Threat Model) הוא כלי המשמש לזיהוי הנכסים הדורשים הגנה, סוגי התוקפים הפוטנציאליים ומטרות התקיפה האפשריות. הגדרה מסודרת של מודל האיום מאפשרת להתאים את מנגנוני ההגנה לסיכונים הממשיים ולא לאיומים תאורטיים בלבד.

נכסים מוגנים

המערכת נועדה להגן על מספר נכסים מרכזיים:

- אתר הלקוח
- שרת המקור (Origin Server)
- פרטי הזדהות (Credentials)
- מידע רגיש
- זמינות השירות

חשוב להבחין בין נכסים גלויים לנכסים סמויים. שרת המקור הוא יעד ברור שניתן לנסות לתקוף באופן ישיר, אך גם מידע כגון מבנה הנתיבים הפנימי, גרסאות תוכנה, מפתחות גישה וקבצי תצורה עשוי להיות בעל ערך רב עבור תוקף.

אחת ממטרות שכבת ה-Edge היא לצמצם ככל האפשר את כמות המידע שהתוקף מסוגל ללמוד על נכסים אלו, ובכך להקטין את יכולתו לבצע Reconnaissance ולתכנן מתקפה ממוקדת.

סוגי תוקפים

הפעילות שנצפתה במהלך החקירה הצביעה בעיקר על תוקפים אוטומטיים והזדמנותיים, ולא על גורם אשר מיקד את פעילותו בארגון מסוים.

בין סוגי הפעילות שזוהו:

- Automated Scanners
- Opportunistic Attackers
- Credential Stuffing Bots
- Distributed Reconnaissance Infrastructure

ההבחנה בין תוקף ממוקד (Targeted) לבין תוקף הזדמנותי (Opportunistic) משמעותית במיוחד. תוקף ממוקד עשוי להשקיע זמן, משאבים ויצירתיות כדי לעקוף מנגנוני הגנה. לעומתו, תוקף הזדמנותי מחפש מטרות קלות ונוטה לעבור ליעד הבא ברגע שהעלות או המאמץ הנדרשים ממנו עולים.

מרבית הפעילות שנצפתה בפרויקט תאמה לפרופיל ההזדמנותי. מסיבה זו, מנגנונים כגון Tarpitting ו-Deception צפויים להיות אפקטיביים במיוחד, שכן הם מגדילים את עלות התקיפה ומפחיתים את הכדאיות שלה.

מטרות התקיפה

ניתוח הפעילות העלה מספר מטרות אפשריות של התוקפים:

- Account Takeover
- Credential Harvesting
- Environment Disclosure
- Remote Code Execution
- Web Shell Deployment

ניתן לראות מטרות אלו כשלבים שונים לאורך שרשרת תקיפה טיפוסית. בשלב הראשון התוקף מנסה לאסוף מידע על המערכת ולזהות חולשות אפשריות. לאחר מכן עשויים להגיע ניסיונות השגת גישה באמצעות Credentials גנובים או חשיפת מידע רגיש. במקרים חמורים יותר, המטרה עשויה להיות הרצת קוד מרחוק או השתלת Web Shell לצורך יצירת אחיזה מתמשכת במערכת.

שרשרת תקיפה זו מתחילה בדרך כלל ב-Reconnaissance, ממשיכה לניסיון גישה או ניצול חולשה ומסתיימת בהשגת שליטה או התמדה (Persistence). כפי שיוצג בהמשך המאמר, מיפוי הפעילות ל- MITRE ATT&CK משקף באופן ברור את המעבר בין שלבים אלו.

סביבת העבודה והטכנולוגיות

הפרויקט נבנה כולו על גבי שכבת הקצה של Cloudflare, ללא צורך בשרתים ייעודיים נוספים. פרק זה מתאר בקצרה את הרכיבים המרכזיים ואת תפקידם, ומפנה לתיעוד הרשמי עבור הרחבה.

רכיב	תיאור התפקיד
Client Server (Origin)	שרת המקור של האתר הנכס שעליו מגנים
Cloudflare	Reverse Proxy ושכבת הגנת Edge מול האתר
Cloudflare Workers	מנוע ה-Honeypot, הזיהוי וה-Scoring
Cloudflare WAF	סינון וחסימת תקיפות לפי חוקים
Rate Limiting	מניעת Brute Force ו-Flooding
Firewall / Security Events	מקור ה-Telemetry לניתוח

[רכיבי הסביבה ותפקידם]

Cloudflare ושכבת ה-Edge

Cloudflare היא רשת תשתית עולמית הפועלת כ-Reverse Proxy בין המבקרים לבין שרת המקור. כל בקשה עוברת תחילה דרך נקודת קצה של Cloudflare הקרובה גאוגרפית למבקר, ושם ניתן לבדוק, לסנן או להסיט אותה עוד לפני שהיא מגיעה לשרת. ארכיטקטורה זו היא הבסיס לכל יכולות ההגנה בפרויקט, משום שהיא מאפשרת לעצור פעילות עוינת ב-Edge ולחשוף את שרת המקור באופן מינימלי. מידע נוסף בדף המוצר: cloudflare.com/products/workers.

Cloudflare Workers

Cloudflare Workers היא פלטפורמת Serverless המריצה קוד JavaScript ישירות ב-Edge, בתוך מנגנון Isolates קל-משקל. לכל בקשה נכנסת מופעל מטפל **fetch()** המקבל את הבקשה ומחזיר תגובה. בפרויקט שלנו, ה-Worker הוא הלב של המערכת: הוא מבצע את הזיהוי, מחשב את ה-Threat Score, מנתב בקשות חשודות ל-Honeypot ואוסף Telemetry. תיעוד רשמי: developers.cloudflare.com/workers.

היתרון של מודל ה-Isolates על פני מודל מסורתי מבוסס Containers או מכונות וירטואליות הוא שהקוד רץ קרוב מאוד למבקר, ללא "Cold Start" משמעותי וללא צורך בניהול תשתית. עבור מערכת אבטחה זהו יתרון כפול: ראשית, ההחלטה (לחסום, לאתגר, להטעות) מתקבלת במהירות וב-Edge, לפני שהבקשה מגיעה לשרת שנית, הלוגיקה ניתנת לעדכון מהיר ומופצת גלובלית בתוך שניות. כך ניתן להגיב לדפוס תקיפה חדש כמעט בזמן אמת, מבלי לגעת בשרת המקור.

Cloudflare WAF

Cloudflare WAF בודק בקשות Web ו-API נכנסות ומסנן תעבורה לא רצויה לפי קבוצות חוקים (Rulesets). ה-WAF מבוסס על שפת ביטויים גמישה המאפשרת לסנן לפי מאפייני בקשה כגון כתובת IP, נתיב URL, Headers ותוכן ה-Body. בפרויקט שולב ה-WAF עם מנגנון ה-Scoring כדי לאפשר חסימה, Managed Challenge או ניתוב ל-Honeypot. תיעוד: developers.cloudflare.com/waf.

Rate Limiting

מנגנון ה-Rate Limiting של Cloudflare מגדיר תקרה לכמות הבקשות העומדות בביטוי מסוים בתוך חלון זמן נתון, ופועל כאשר התקרה נחצית. הוא חיוני לבלימת Brute Force, Credential Stuffing ו-POST Flooding, משום שהוא פוגע ישירות בקצב הפעולה של כלי Automation. תיעוד:
developers.cloudflare.com/waf/rate-limiting-rules

ניתוח המידע

מקורות הנתונים

הנתונים שעליהם התבסס הניתוח נאספו ממספר מקורות בתוך סביבת Cloudflare, אשר יחד סיפקו תמונה רחבה של התעבורה החשודה:

מקור נתונים	תיאור
Firewall Events	בקשות שנחסמו, אותגרו או סווגו כחשודות
Security Events	אירועי אבטחה ממערכת הניטור של Cloudflare
Bot Protection Logs	מידע על בוטים, Crawlers ותעבורה אוטומטית
Rate Limiting Logs	חסימות והגבלות קצב בעקבות פעילות חריגה
Worker Logs	Telemetry מתוך מערכת ה-Honeypot

[מקורות הנתונים לחקירה]

מתודולוגיית הניתוח

בשלב הראשוני בוצע ניתוח של Firewall Events ו-Worker Logs במטרה לזהות פעילות עוינת ודפוסים חוזרים. הניתוח כלל בחינה של כמות הבקשות והיקף הפעילות (Requests Count), זיהוי מדינות ותשתיות המקור (Source Countries), ניתוח כתובות IP חשודות ותשתיות ענן (Source IP Analysis), ניתוח כלי

תקיפה ובוטים לפי User-Agent, מיפוי הנתיבים שנבדקו (Path Enumeration), וסיווג שמפריד בין Crawlers לגיטימיים לבין פעילות עוינת (Bot Classification).

עיקרון מנחה בניתוח היה לבחון אינדיקציות בהקשר, ולא במנותק. בקשה בודדת לנתיב .php אינה מעידה בהכרח על תקיפה אך כאשר אותה כתובת IP פונה לעשרות נתיבי WordPress ברצף, בתדירות גבוהה ועם User-Agent של כלי Automation - התמונה המצטברת ברורה. לכן הניתוח התמקד ב-Correlation: קישור בין מאפייני בקשה שונים (User-AgentIP, נתיב, שיטה, תזמון) לכדי פרופיל פעילות אחד, שאותו ניתן לסווג ולנתח.

ממצאים מרכזיים

החקירה העלתה כי האתר היה יעד לסריקות אוטומטיות רחבות היקף, אשר בוצעו ברובן באמצעות תשתיות ענן ציבוריות. הטבלה הבאה מסכמת את הממצאים העיקריים:

פרמטר	ממצא
סוגי בקשות	GET / POST
מדינות שזוהו	ארה"ב ותשתיות ענן בינלאומיות
סוגי תנועה	Suspicious Reconnaissance לצד SEO Crawlers
נתיבים שנבדקו	wp-login.php / .env / shell.php
תשתיות שזוהו	AWS / Azure / Oracle Cloud / OVH

[ממצאים מרכזיים]

ניתוח IOC

במהלך החקירה זוהו מספר Indicators of Compromise (אינדיקטורים לפעילות עוינת) אשר הצביעו על Automation. הטבלה הבאה מציגה מבחר כתובות IP שנצפו, יחד עם ה-ASN והספק שמאחוריהן. ה-ASN הוא מזהה ייחודי לרשת המנוהלת על-ידי גורם אחד, ומאפשר לקשר כתובת IP לספק התשתית:

IP	מדינה	ASN	ספק	סוג פעילות
161.153.99.197	US	AS31898	Oracle Cloud	Credential Stuffing
161.118.254.192	SG	AS31898	Oracle Cloud	Automated Recon
150.136.244.33	US	AS31898	Oracle Cloud	Brute Force
174.138.54.2	US	AS14061	DigitalOcean	XMLRPC Recon
20.226.103.253	BR	AS8075	Microsoft Azure	PHP Enumeration
140.245.121.218	SG	AS31898	Oracle Cloud	XMLRPC Enumeration
193.143.1.112	RU	AS198953	Proton66 OOO	ENV File Recon

IP	מדינה	ASN	ספק	סוג פעילות
173.239.214.211	US	AS62240	Clouvider	WP Login Probe
103.230.178.122	IN	AS136634	Navkar Netsol	XMLRPC POST
172.202.0.52	US	AS8075	Microsoft Azure	PHP Path Enum
88.151.33.24	NL	AS41608	NextGenWebs	Git Repo Enum
176.65.139.232	NL	AS214472	Offshore LC	ENV Enumeration
195.178.110.159	NL	AS48090	TECHOFF SRV	Git Enumeration

[מבחר כתובות IP חשודות שזוהו (IOCs)]

רוב הכתובות החשודות השתמשו בתשתיות ענן ציבוריות דבר המאפיין קמפיינים אוטומטיים של Reconnaissance ו-Credential Stuffing. בנוסף זוהו דפוסים של User-Agent Rotation, שימוש ב-VPS ציבוריים, וניסיונות הסתרת Fingerprint.

ניתוח הפיזור הגאוגרפי וה-ASN מלמד שתי תובנות. ראשית, ריכוז ניכר של כתובות שייך ל-AS31898 Oracle Cloud דבר המעיד על קמפיין שעושה שימוש אינטנסיבי בספק ענן יחיד, ככל הנראה באמצעות מספר מכונות שהוקמו במקביל.

שנית, הפיזור על פני מדינות רבות (ארה"ב, סינגפור, ברזיל, רוסיה, הודו, הולנד) אינו מעיד בהכרח על מיקומו האמיתי של התוקף, אלא על מיקום מרכזי הנתונים של ספקי הענן. במילים אחרות, ה"מדינה" היא מאפיין של התשתית, לא של היריב. תובנה זו חשובה למדיניות החסימה: חסימה לפי מדינה (Geo-Blocking) תהיה גסה מדי ועלולה לחסום משתמשים לגיטימיים, בעוד שחסימה או ניתוב מבוססי ASN, יחד עם Threat Scoring ברמת הבקשה, מספקים איזון טוב בהרבה בין הגנה לבין שמירה על זמינות לתעבורה אמיתית.

מודיעין User-Agent

שדה ה-User-Agent מאפשר להבדיל בין פעילות לגיטימית לבין פעילות עוינת. הטבלה מציגה דוגמאות לסיווג שבוצע:

User-Agent	סיווג
Googlebot	Legit Search Engine
AhrefsBot	SEO Crawler
meta-webindexer	AI / Meta Indexer
curl	Suspicious Automation
sqlmap	Offensive Security Tool
Go-http-client	Automation Framework

[סיווג User-Agents]

השימוש ב-curl, ב-sqlmap ובכלי Automation נוספים הצביע על סריקה אוטומטית ולא על גלישה אנושית. בהמשך מובאים תיאורים קצרים של הכלים הפומביים הרלוונטיים.

על הכלים הפומביים שזוהו

curl - כלי שורת פקודה להעברת נתונים בפרוטוקולים שונים (בהם HTTP/S). לגיטימי לחלוטין בשימוש יומיומי, אך נפוץ מאוד גם בסקריפטים אוטומטיים של סריקה. אתר הכלי: curl.se.

wget - כלי GNU להורדת קבצים מהאינטרנט דרך שורת הפקודה, אף הוא לגיטימי אך נפוץ ב-Automation. אתר: gnu.org/software/wget.

sqlmap - כלי בדיקות חדירה (Offensive Security) קוד פתוח לאוטומציה של זיהוי וניצול חולשות SQL Injection. הופעתו ב-User-Agent מעידה כמעט תמיד על סריקה התקפית. אתר: sqlmap.org.

nikto - סורק שרתי Web בקוד פתוח המחפש קבצים מסוכנים, גרסאות מיושנות וקונפיגורציות בעייתיות. מאגר הפרויקט: github.com/sullo/nikto.

masscan - סורק פורטים מהיר במיוחד המסוגל לסרוק טווחי כתובות עצומים בזמן קצר כלי מרכזי בקמפיינים של Mass Scanning. מאגר: github.com/robertdavidgraham/masscan.

Googlebot / AhrefsBot - בוטים לגיטימיים: הראשון הוא ה-Crawler של מנוע החיפוש Google, והשני סורק SEO מסחרי. אלו זוהו בנפרד והוכנסו ל-Allowlist כדי לצמצם False Positives.

חשוב לזכור ששדה ה-User-Agent ניתן לזיוף בקלות תוקף יכול להציג כל מחרוזת שירצה, כולל התחזות ל-Googlebot. לכן ה-User-Agent לבדו אינו מספיק לסיווג, והוא משמש כאינדיקציה אחת מני רבות במנגנון ה-Scoring. אימות "בוטים מאומתים" (Verified Bots) מתבצע באמצעות בדיקות נוספות, כגון התאמה בין כתובת ה-IP לבין טווחי הכתובות הרשמיים של הספק, ולא על סמך מחרוזת ה-User-Agent בלבד.

דוגמאות מייצגות מתוך הלוגים

להמחשת אופי הפעילות, מובאות כאן מספר דוגמאות מייצגות של רשומות לוג, לאחר Sanitization..

ניסיון Credential Stuffing

```
POST /wp-login.php HTTP/1.1
Host: <target>
User-Agent: Mozilla/5.0 (compatible)
Content-Type: application/x-www-form-urlencoded
log=admin&pwd=********&wp-submit=Log+In
-> classification: HIGH (credential_stuffing)
```

הדוגמה ממחישה ניסיון התחברות אוטומטי לנתיב WordPress. שדה הסיסמה הוצג ממוסך (Masked) המערכת שמרה Hash בלבד לצורכי Correlation, בהתאם לעקרונות ה-Data Minimization.

WordPress של Fingerprinting

```
GET /wp-includes/js/wp-emoji-release.min.js HTTP/1.1
User-Agent: Go-http-client/1.1
-> classification: MEDIUM (wp_fingerprinting)
```

האירוע הצביע על ניסיון Fingerprinting של סביבת WordPress באמצעות גישה לנתיב סטטי מוכר. השימוש ב-Go-http-client הצביע על Automation ולא על גלישה אנושית.

XMLRPC Abuse עם החלפת User-Agent

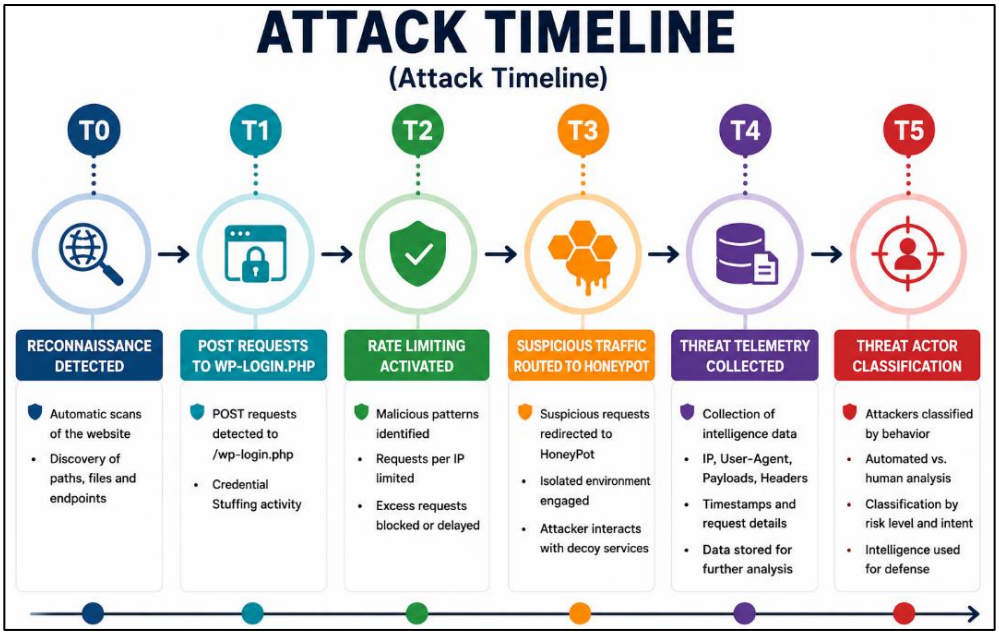
```
POST /xmlrpc.php HTTP/1.1
User-Agent: Go-http-client/1.1
<methodCall><methodName>metaWeblog.newPost</methodName> ...
-> classification: CRITICAL (xmlrpc_abuse)
   (repeated from same IP with rotated User-Agent)
```

המערכת זיהתה ניסיון XMLRPC Abuse באמצעות קריאה ל-metaWeblog.newPost, עם Payload בפורמט XML מלא. כאשר אותו מקור חזר על הניסיון תוך החלפת ה-User-Agent, סומן הדבר כ-User-Agent Rotation וכ-Bot Obfuscation ניסיון לעקוף זיהוי מבוסס Fingerprinting. שילוב של POST, XMLRPC Abuse Credential Attempt ומבנה Payload חשוד הוביל לסיווג Critical.

מסקנות Threat Intelligence ראשוניות

ניתוח האירועים העלה כי לא מדובר בתוקף בודד, אלא בפעילות אוטומטית רחבת היקף שבוצעה באמצעות תשתיות ענן ציבוריות. הממצאים הצביעו על קמפיינים של Credential Stuffing, על Reconnaissance מבוזר, על Enumeration של WordPress, על תשתית תקיפה מבוססת ענן, ועל ניסיונות Exploitation אוטומטיים. דפוסי הפעילות תאמו קמפיינים של Mass Scanning המבוצעים כנגד שירותי Web ציבוריים.

ציר זמן של האירועים



מיפוי MITRE ATT&CK

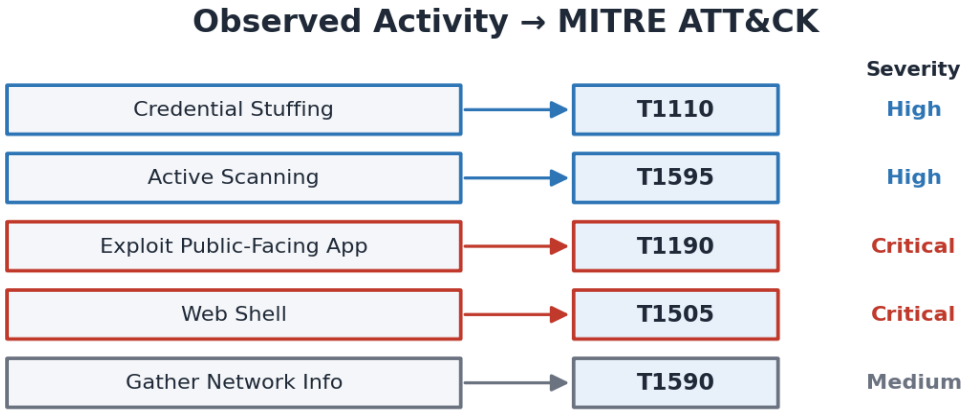
כדי לתאר את הפעילות בשפה אחידה ומקובלת בתעשייה, מופתה הפעילות שנצפתה למסגרת MITRE ATT&CK בסיס ידע פתוח של טכניקות תקיפה. המיפוי מסייע להבין את שלבי התקיפה ולתעדף תגובה:

Technique	MITRE ID	Evidence	Severity
Credential Stuffing	T1110	POST ל-/wp-login.php	High
Active Scanning	T1595	XMLRPC / ENV Enum	High
Exploit Public-Facing App	T1190	phpunit/eval-stdin probes	Critical
Web Shell	T1505	בקשות shell.php	Critical
Gather Network Info	T1590	בקשות .git/config	Medium

[מקור: MITRE ATT&CK. attack.mitre.org/techniques/]

הפניות ישירות לעמודי הטכניקות: T1110 , T1595 , T1190, T1505, T1590

המיפוי משקף את שלבי שרשרת התקיפה כפי שנצפו בפועל. TGather Network Information (1590) ו-TActive Scanning (1595) הם שלבי איסוף המידע הראשוניים התוקף ממפה את היעד ומחפש נקודות כניסה. TExploit Public-Facing Application (1190) ו-TWeb Shell (1505) מייצגים את שלב הניצול וההשתלטות, שבו התוקף מנסה להריץ קוד או להשתיל אחיזה מתמשכת. TBrute Force (1110), הכולל Credential Stuffing, מכוון להשגת גישה דרך פרטי הזדהות. הצגת הפעילות בשפה המשותפת של ATT&CK מאפשרת לתקשר את הסיכון לגורמים שונים בארגון ולתעדף תגובה לפי שלב התקיפה וחומרתו.



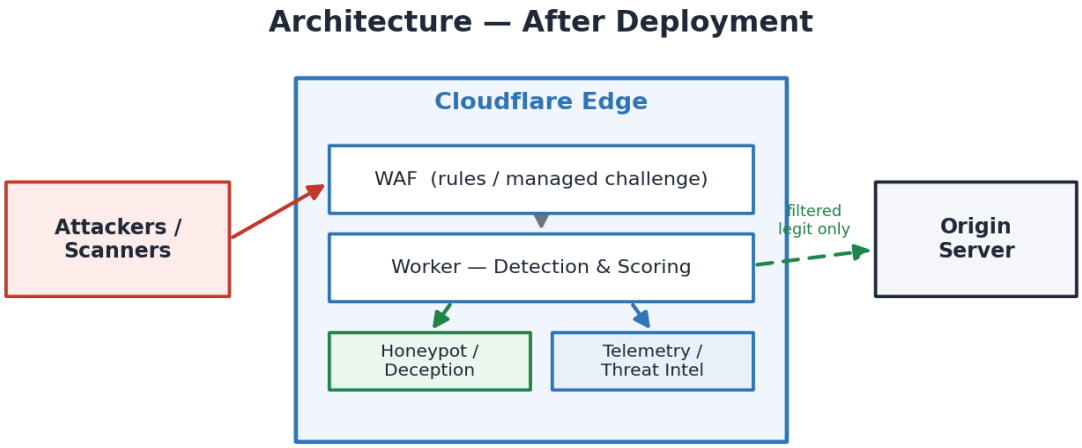
ארכיטקטורת ההגנה

לפני ההטמעה

במצב ההתחלתי, בקשות חשודות הגיעו ישירות לשרת המקור. לא נאסף מודיעין מתקדם, לא בוצעה Deception, ולא התקיים Threat Classification. שטח התקיפה היה חשוף, וכל בקשה לגיטימית או עוינת הגיעה אל השרת באופן זהה:

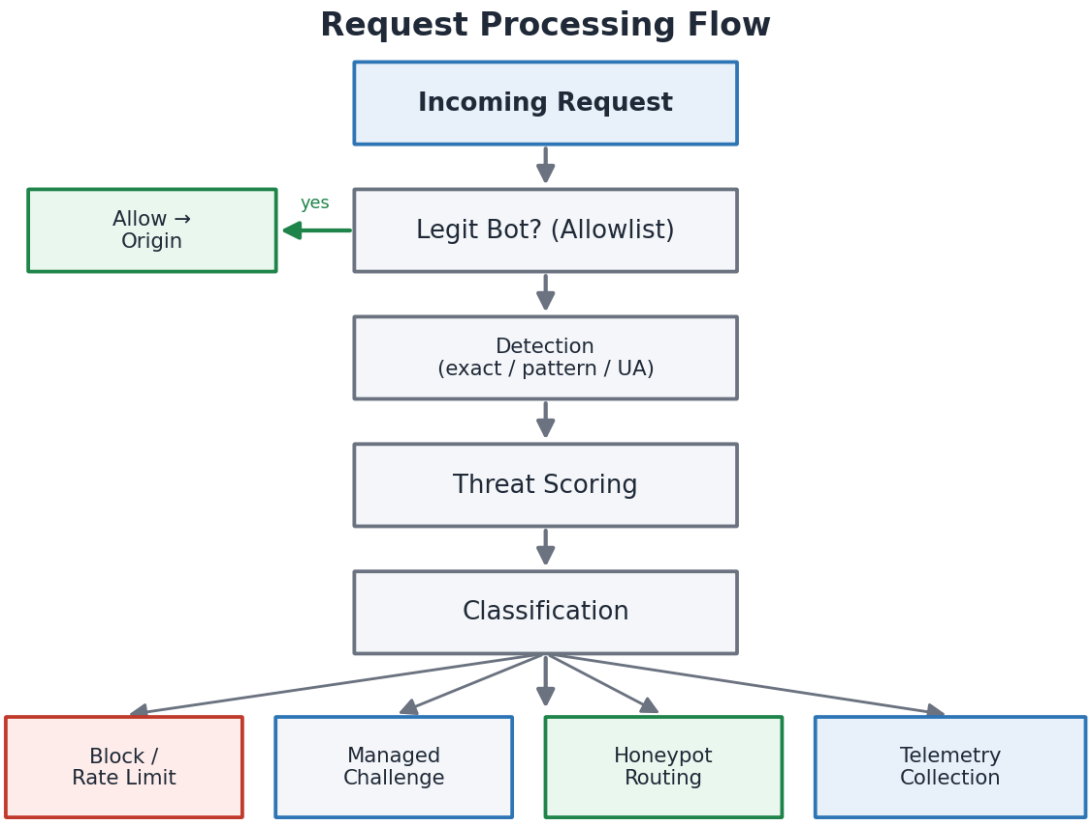
לאחר ההטמעה

הארכיטקטורה החדשה ממקמת את Cloudflare כשכבת Edge בין התוקפים לבין שרת המקור. בקשות עוברות תחילה דרך ה-WAF, משם ל-Worker שמבצע זיהוי ו-Scoring, ובהתאם לסיווג מנותבות לחסימה, ל-Managed Challenge, ל-Honeypot או ל-Telemetry. רק תעבורה לגיטימית מסוננת ממשיכה אל שרת המקור. כך מתקבלת עצירת איומים ב-Edge, הפחתת חשיפת השרת, איסוף מודיעין בזמן אמת, שכבת Deception ו-Threat Classification מובנה.



מנוע ה-Worker וה-Honeypot

ה-Cloudflare Worker מהווה את מנוע ה-Detection וה-Deception של המערכת. עבור כל בקשה הוא מבצע רצף פעולות: זיהוי (Detection), חישוב סיכון (Threat Scoring), ניתוב ל-Honeypot במידת הצורך, איסוף Telemetry, החזרת תגובות מזויפות (Fake Responses) וסיווג (Classification). מושג ה-Honeypot מערכת מלכודת המדמה יעד פגיע כדי למשוך ולנתח תוקפים מתואר בהרחבה בערך Honeypot computing בוויקיפדיה.



סדר הפעולות בצינור העיבוד (Pipeline) אינו מקרי. תחילה נבדק האם מדובר בבוט לגיטימי מוכר אם כן, הבקשה מועברת ישירות לשרת המקור ללא עיכוב. בקשות שאינן ב-Allowlist עוברות לשלב הזיהוי, ומשם ל-Scoring ולסיווג. רק לאחר שהבקשה סווגה נבחרת הפעולה המתאימה: חסימה, Managed Challenge, ניתוב ל-Honeypot, או רישום Telemetry. סדר זה מבטיח שתעבורה לגיטימית כמעט אינה מושפעת מהמערכת, בעוד שתעבורה חשודה מטופלת באופן מדורג ופרופורציונלי לרמת הסיכון שלה.

מנגנוני זיהוי

Exact Match Detection

המערכת בדקה נתיבים חשודים בהתאמה מדויקת, ובהם /, /adminer.php, /.env, wp-login.php, /shell.php ו-/server-status. נתיבים אלו אינם קיימים באתר הלגיטימי, ולכן גישה אליהם היא אינדיקציה חזקה לסריקה. היתרון בשיטה זו הוא דיוק גבוה ומיעוט False Positives: מבקר אנושי לגיטימי כמעט לעולם לא יפנה ישירות אל /wp-login.php באתר שאינו מבוסס WordPress. החיסרון הוא נוקשות הרשימה סטטית, ותוקף שמשתמש בנתיב חדש שאינו ברשימה לא ייתפס. לכן Exact Match הוא רק שכבה אחת, והוא משולב עם זיהוי גמיש יותר מבוסס דפוסים.

Pattern Matching

מעבר להתאמות מדויקות, בוצע זיהוי לפי דפוסים (Patterns) המופיעים בנתיב או בבקשה, כגון php., phpunit, shell, git.wordpress ו-obfuscate. גישה זו תופסת גם וריאציות חדשות של נתיבים שלא נצפו קודם.

זיהוי מבוסס-דפוסים מרחיב משמעותית את הכיסוי, אך מחייב כיול זהיר. דפוס רחב מדי עלול לתפוס תעבורה לגיטימית (False Positive), בעוד שדפוס צר מדי יחמיץ וריאציות (False Negative). האיזון בין השניים הוא אחת ההחלטות המרכזיות בעיצוב מערכת זיהוי, ובפרויקט זה הוא נתמך על-ידי מנגנון ה-Scoring שמשקלל את הדפוס יחד עם אינדיקציות נוספות במקום לחסום אוטומטית על כל התאמה.

User-Agent Analysis

נותחו User-Agents המעידים על Automation, כגון curl, wget, sqlmap, masscan ו-nikto, וכן בקשות חסרות מאפייני דפדפן (Headless). שילוב של נתיב חשוד עם User-Agent חשוד מעלה משמעותית את רמת הסיכון המחושבת.

כפי שצוין, ה-User-Agent ניתן לזיוף, ולכן הוא נמדד תמיד בהקשר. בקשה עם User-Agent ריק או חריג, היעדר Headers מקובלים של דפדפן (כגון Accept-Language), ותזמון מכני המדויק מדי כל אלה מצטרפים לפרופיל שמבדיל בין כלי Automation לבין משתמש אנושי. עיקרון ה-Correlation הזה הוא שמאפשר זיהוי אמין גם כאשר התוקף מנסה להתחזות.

סינון בוטים לגיטימיים (AllowList)

ניתוח הלוגים חשף תנועה לגיטימית של Crawlers כגון Googlebot, AhrefsBot, Bingbot ו- Meta Web Indexer. כדי שמנגנוני הזיהוי לא יסווגו אותם בטעות כעוינים, הוטמע מנגנון Allowlist המאפשר את מעברם התקין. מנגנון זה מצמצם False Positives, משפר את איכות ה-Telemetry ומונע "זיהום" של

הלוגים בתעבורה לגיטימית. ההפרדה בין Crawlers לגיטימיים לבין פעילות עוינת היא תנאי הכרחי לאיכות ה-Threat Intelligence שמופק מהמערכת.

עם זאת, ה-Allowlist עצמו מהווה משטח תקיפה פוטנציאלי: תוקף שמתחזה ל-Googlebot מקווה לקבל מעבר חופשי. לכן ההכללה ב-Allowlist אינה מבוססת על מחרוזת ה-User-Agent בלבד, אלא על אימות נוסף למשל בדיקה שכתובת ה-IP אכן שייכת לטווחים הרשמיים של הספק. כך נשמר האיזון בין הימנעות מחסימת בוטים מועילים (כמו מנועי חיפוש, שחשובים ל-SEO של האתר) לבין מניעת ניצול ה-Allowlist להתחמקות.

Threat Scoring

מנוע ה-Scoring

מנגנון ה-Threat Scoring נבנה כדי להעריך את רמת הסיכון של כל בקשה. לכל Request מחושב Score בהתאם למספר פרמטרים: סוג הנתיב שנבדק, ה-User-Agent, שיטת ה-HTTP, אינדיקציות ב-Payload, דפוסי תקיפה מוכרים, אינדיקציות Reconnaissance וניסיונות Exploitation. הניקוד מאפשר להבדיל בין סריקה בסיסית, פעילות Automation חשודה, ניסיון Exploitation, ומתקפה בסיכון גבוה.

לוגיקת הניקוד ו-Correlation

הניקוד מחושב לפי חומרת האינדיקציה ופוטנציאל הסיכון שלה, אך הכוח האמיתי טמון ב-Correlation בין אינדיקציות. לדוגמה: בקשת GET בודדת לנתיב .php מקבלת ניקוד נמוך יחסית אולם שילוב של גישה ל-.env, יחד עם User-Agent חשוד, בקשת POST ותדירות גבוהה מוביל לעלייה חדה בניקוד ולסיווג מחמיר. גישה זו מאפשרת לזהות Credential Stuffing, Reconnaissance אוטומטי, Exploit Scanning ו-Web Shell Enumeration גם כאשר כל אינדיקציה בנפרד אינה חד-משמעית.

כדי להמחיש את העיקרון, נבחן דוגמה (Illustrative) של חישוב Score עבור בקשת POST ל-/wp-login.php המגיעה מתשתית ענן עם User-Agent של כלי Automation.

כל אינדיקציה בנפרד אינה מספיקה לחסימה ודאית אולם הצטברותן יחד מובילה ל-Score גבוה ולסיווג High או Critical. זהו ההבדל המהותי בין חוק חסימה בודד לבין מנוע Scoring: המנוע מודד את התמונה המצטברת, ולכן הוא עמיד יותר בפני ניסיונות התחמקות שבהם התוקף משנה מאפיין אחד בכל פעם.

מנוע הסיווג (Classification)

לאחר חישוב הניקוד, כל בקשה סווגה לרמת סיכון. בקשות בסיכון נמוך נרשמו ל-Log בלבד בקשות בסיכון בינוני הופנו ל-Managed Challenge ובקשות בסיכון גבוה נחסמו או נותבו ל-Honeypot. סיווג זה הוא שמתרגם את ה-Score המספרי לפעולה מבצעית:

SCORE RANGE	CLASSIFICATION	SEVERITY	DESCRIPTION
50+	low_confidence_probe	LOW	Basic reconnaissance Low-confidence activities or weak indicators. Minimal risk to the environment.
100+	suspicious_probe	MEDIUM	Suspicious activity Multiple indicators detected. Potential reconnaissance or probing behavior.
180+	active_scanner	HIGH	Active scanning Aggressive scanning or automation detected. High probability of attack attempts.
250+	critical_exploitation_attempt	CRITICAL	Critical exploitation attempt High-risk exploitation behavior detected. Immediate threat to the system.

NOTE: This classification helps prioritize alerts, automate response actions, and improve detection accuracy by identifying the most dangerous threats in real time.

דוגמאות אמיתיות מתוך הלוגים

Active Scanner Detection

```
{
  "path": "/obfuscate.php",
  "score": 180,
  "classification": "active_scanner",
  "matchedRules": [
    "pattern:obfuscate",
    "pattern:.php"
  ]
}
```

Critical Exploitation Attempt

```
{
  "path": "/vendor/phpunit/eval-stdin.php",
  "classification": "critical_exploitation_attempt",
  "score": 290,
  "matchedRules": [
    "pattern:phpunit",
    "pattern:eval-stdin"
  ]
}
```

מערכת ה-Honeypot

מטרות המערכת

מערכת ה-Honeypot נבנתה כחלק משכבת ה-Deception וה-Threat Intelligence. מטרתה אינה רק לחסום תוקפים, אלא ליצור סביבת Interaction מבוקרת המאפשרת לאסוף מודיעין בזמן אמת. המערכת נועדה להטעות תוקפים באמצעות שירותים ונתיבים מזויפים, לאסוף Payloads ו-User-Agents, לזהות מתקפות אוטומטיות, לנתח ניסיונות Credential Stuffing, ולהאריך את זמן הפעילות של התוקף בסביבה מבוקרת לצורך Telemetry. כך הופכת התקיפה ממקור סיכון למקור מודיעין.

תגובות מזויפות ושכבת Deception

במקום להחזיר שגיאות רגילות, המערכת החזירה תגובות מזויפות המדמות שירותים פגיעים, כדי לגרום לתוקף להמשיך באינטראקציה:

שירות מזויף	מטרה
Fake WordPress Login	זיהוי Credential Stuffing ו-Brute Force
Fake ENV File	זיהוי ניסיונות חשיפת Secrets
Fake Adminer Panel	זיהוי Database Enumeration ו-Admin Probing
Fake phpinfo	זיהוי PHP Reconnaissance ו-Exploit Scanning

שכבת ה-Deception אפשרה להאריך את זמן החשיפה של התוקף, לאסוף Payloads ודפוסי תקיפה, לזהות כלי Automation, לשפר את איכות ה-Threat Intelligence ולצמצם את חשיפת שרת המקור.

הרעיון המנחה בבסיס ה-Deception הוא היפוך יחסי הכוחות: בעוד שתוקף אוטומטי מצפה לתגובה דיכוטומית (הנתיב קיים או לא), המערכת מספקת לו תגובה שנראית "מבטיחה" דף Login לכאורה אמיתי, קובץ Environment לכאורה חשוף וכך גורמת לו להמשיך ולהשקיע מאמץ. כל בקשת המשך מספקת מידע נוסף: אילו שדות התוקף מנסה למלא, אילו Payloads הוא שולח, ובאיזה קצב. המידע הזה הופך את ה-Honeypot ממנגנון הגנה פסיבי למקור מודיעין פעיל, מבלי שהתוקף נחשף לשרת האמיתי כלל.

Credential Telemetry

כאשר תוקף ניסה להתחבר דרך דף ה-Login המזויף, נאסף מידע מוגבל לצורכי Telemetry בלבד. מנגנון זה איפשר לזהות קמפיינים של Password Spraying ו-Credential Stuffing, שימוש בסיסמאות נפוצות, Bot Automation וכלים התקפיים מבלי לפגוע בפרטיות.

ההבחנה בין Credential Stuffing לבין Password Spraying משמעותית לניתוח. ב-Credential Stuffing התוקף משתמש בזוגות שם-משתמש/סיסמה שדלפו ממאגרים אחרים, בהנחה שמשתמשים ממחזרים סיסמאות ב-Password Spraying התוקף מנסה מספר סיסמאות נפוצות מול חשבונות רבים, כדי להימנע מנעילת חשבון. דפוס הניסיונות מספר החשבונות מול מספר הסיסמאות וקצב הניסיון מאפשר להבחין בין השניים, וכל אחד מהם מצביע על סוג קמפיין שונה ועל תשתית תקיפה שונה.

שיקולי אבטחה ואתיקה

מערכת ה-Honeypot תוכננה תוך הקפדה על עקרונות Privacy ו-Data Minimization. לא נשמרו סיסמאות מלאות בוצע Masking לפני שמירה נשמר Hash בלבד לצורכי Correlation לא בוצע Replay של Credentials ולא בוצע ניסיון התחברות למערכות צד שלישי. נשמר Telemetry מינימלי בלבד. המערכת נבנתה למטרות הגנה, מחקר ולימוד של טכניקות Detection ו-Deception ולא נעשה בה כל שימוש התקפי או זדוני.

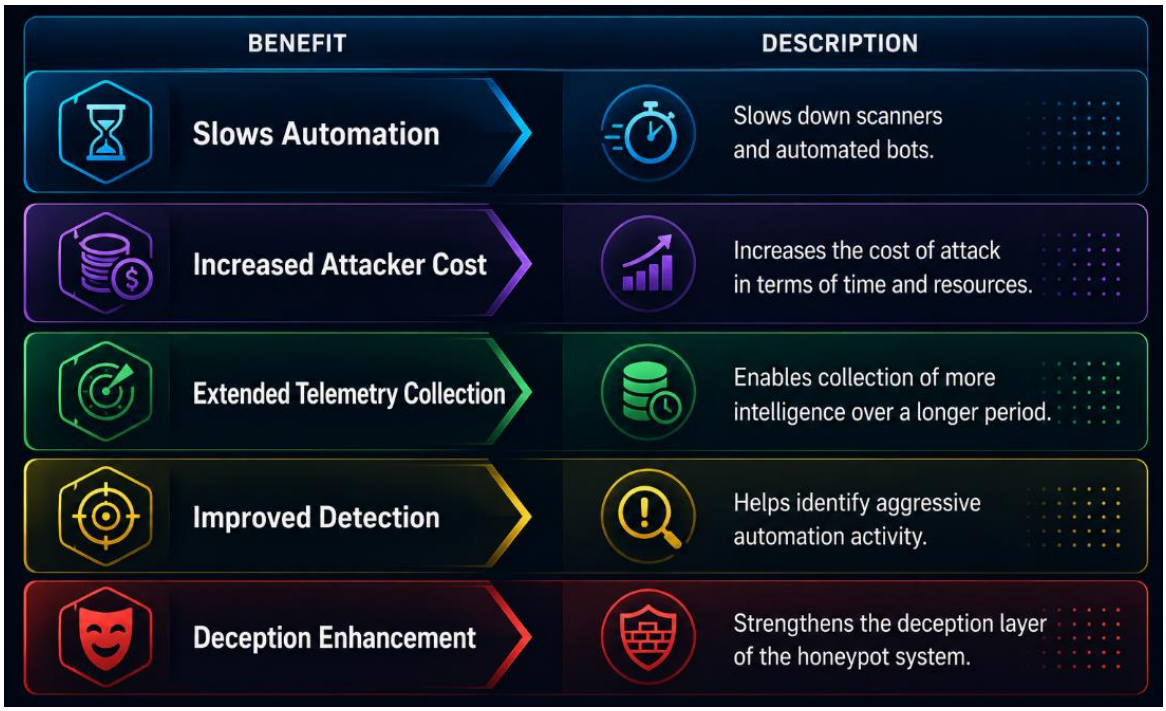
Tarpitting

במסגרת ה-Honeypot הוטמע מנגנון Tarpitting שנועד להאט פעילות אוטומטית ולפגוע ביעילות של סורקים. במקום לחסום מיד בקשות חשודות, המערכת החזירה תגובות בעיכוב אקראי ומבוקר, וכך יצרה עומס על כלי ה-Automation וביזבזה את משאבי התקיפה. גישה זו האטה Scanners, הגדילה את זמן השהיית התוקף בסביבת ה-Honeypot, העלתה את עלות התקיפה במונחי זמן ומשאבים, ופגעה ביעילות של קמפיינים מבוססי Mass Scanning תוך איסוף Telemetry נוסף לאורך זמן.

גישה זו אפשרה:

- להאט Scanners אוטומטיים
- להגדיל את זמן הפעילות של תוקפים בסביבת ה-Honeypot
- להגדיל את עלות התקיפה מבחינת זמן ומשאבים
- לפגוע ביעילות של Mass Scanning Campaigns
- לאסוף Telemetry נוסף לאורך זמן
- לזהות כלי Automation אגרסיביים

ובתמונה:



איסוף מודיעין Intelligence Collection

מערכת ה-Honeypot ומנגנוני ה-Telemetry אפשרו לאסוף מידע מבצעי בזמן אמת מתוך הפעילות העוינת. הנתונים נאספו באמצעות Cloudflare Workers, Firewall Events, Security Events ו-Worker Observability Logs. הטבלה הבאה מסכמת את סוגי הנתונים שנאספו:

סוג נתון	תיאור
IP Address	כתובת המקור של הפעילות
Country	מדינת המקור של הבקשה
ASN	מספר ה-Autonomous System של התשתית
User-Agent	זיהוי דפדפן, בוט או כלי Automation
Request Path	הנתיב שנבדק או הותקף
HTTP Method	סוג הבקשה (GET / POST)
Headers	כותרות HTTP מלאות ל-Fingerprinting
Request Body	חלקי Payload מבקשות חשודות
Attack Score	ה-Threat Score שחושב
Classification	סיווג רמת הסיכון
Matched Rules	חוקי Detection שהופעלו
Fingerprint	מזהה ל-Correlation בין בקשות

בנוסף, המערכת אספה מידע מתקדם על XMLRPC Abuse, ניסיונות Credential Stuffing, User-Agent Rotation, Bot Obfuscation, Automation Frameworks, WordPress Fingerprinting, Campaign Markers ו-Exploitation Payloads. בין היתר זוהו ניסיונות שימוש ב-WordPress API Methods כגון metaWeblog.newPost, לצד Payloads חשודים של POST. המערכת הצליחה לבצע Correlation בין אותו IP, החלפת User-Agent, דפוסי Payload ותזמון התקיפה, וכך לזהות פעילות Automation רחבת היקף.

נתיב xmlrpc.php/ ראוי להסבר קצר: זהו ממשק XML-RPC של WordPress, שבעבר נוצל לרעה לצורך הגברת מתקפות (Amplification) ו-Brute Force מרוכז באמצעות שיטות כגון system.multicall. ניסיונות גישה אליו, ובמיוחד בקשות POST עם מבני XML מורכבים, הם אינדיקציה מובהקת ל-Automation ולא לגלישה אנושית. ה-Correlation בין כתובת IP יחידה לבין מספר User-Agents שונים לאורך זמן (User-Agent Rotation) הוא אחד הסימנים החזקים ביותר לניסיון התחמקות מ-Detection, והוא מאפשר לקשר בקשות לכאורה-נפרדות לאותו קמפיין תקיפה.

אינטגרציית WAF

עקרון האינטגרציה

לאחר ניתוח ה-Telemetry וה-IOC, עודכנו חוקי ה-WAF כך שיזהו ויחסמו דפוסים שחזרו באופן עקבי. בניגוד ל-WAF מסורתי המתמקד בחסימה בלבד, המערכת שילבה בין Detection, Threat Classification, Honeypot Routing, Deception ו-Telemetry. כך הפך ה-Threat Intelligence שנאסף בזמן אמת למדיניות הגנה מבצעית ב-Edge, ובקשות חשודות יכלו להיחסם, לעבור Managed Challenge, להיות מנותבות ל-Honeypot, או להיכנס לתהליך Telemetry.

חוקי WAF לדוגמה

Pattern	מטרת הזיהוי
.php	Reconnaissance-ו PHP Enumeration
shell	Web Shell Discovery
phpunit	ניסיונות RCE מוכרים
adminer	Database Enumeration
.git	מניעת Git Repository Exposure
.env	חסימת ניסיונות חשיפת Secrets
xmlrpc.php	WordPress Automation
wp-login.php	Credential Stuffing

[דוגמאות לחוקי WAF שהוטמעו]

ניתוב ל-Honeypot, Managed Challenge ו-Rate Limiting

חלק משמעותי מהפעילות התמקד בנתיבי WordPress נפוצים, ובמיוחד /wp-login.php ו-/xmlrpc.php. במקום לחסום מיד את כל הבקשות לנתיבים אלו, בוצעה התאמה ייעודית שאיפשרה ניתוב מבוקר של בקשות חשודות אל ה-Honeypot והפיכתן למקור מודיעין. עבור בקשות שסווגו כחשודות אך ללא ודאות מלאה, הופעל Managed Challenge של Cloudflare, אשר סינן תעבורה אוטומטית ובוטים תוך שמירה על חוויית משתמש תקינה. במקביל הופעל Rate Limiting עבור פעילות שתאמה Credential Stuffing, Brute Force, POST Flooding ו-XMLRPC Abuse.

השפעה מבצעית

האינטגרציה בין ה-WAF לבין ה-Honeypot יצרה שכבת הגנה מבצעית שעצרה איומים ב-Edge, צמצמה את חשיפת שרת המקור, שיפרה את ה-Visibility על פעילות עוינת, אספה Threat Intelligence בזמן אמת, וזיהתה תשתיות תקיפה אוטומטיות. כך הודגם כיצד ניתן להפוך מערכת WAF ממנגנון חסימה סטטי לפלטפורמת Detection ו-Threat Intelligence דינמית.

יתרון מהותי בגישה זו הוא לולאת המשוב (Feedback Loop): ה-Telemetry שנאסף ב-Honeypot מזין את חוקי ה-WAF, חוקי ה-WAF מנתבים תעבורה נוספת ל-Honeypot, וזה מזין שוב את הניתוח. כך המערכת משתפרת עם הזמן ומתאימה את עצמה לדפוסי תקיפה חדשים, במקום להישאר תלויה ברשימת חוקים סטטית שנקבעה מראש.

Risk Assessment

מתודולוגיה

הערכת הסיכון התבססה על שילוב בין Likelihood (הסתברות התקיפה), Impact (רמת הנזק האפשרית) ו-Threat Severity (חומרת הפעילות שנצפתה). נשקלו פוטנציאל הפגיעה, סבירות ההתממשות, רמת החשיפה, יכולת ה-Exploitation, ההשפעה על הזמינות והפוטנציאל לחשיפת מידע רגיש.

הגישה המקובלת מציגה את הסיכון כמכפלה של הסתברות בהשפעה. אירוע בעל הסתברות גבוהה והשפעה נמוכה (כמו סריקת Reconnaissance שגרתית) מדורג נמוך-בינוני אירוע בעל הסתברות נמוכה אך השפעה קריטית (כמו הרצת קוד מרחוק מוצלחת) מדורג גבוה למרות נדירותו. הצלבת שני הצירים הללו מאפשרת לתעדף את מאמצי ההגנה אל עבר התרחישים שבהם שילוב ההסתברות וההשפעה מסוכן במיוחד, במקום לפזר משאבים באופן אחיד.

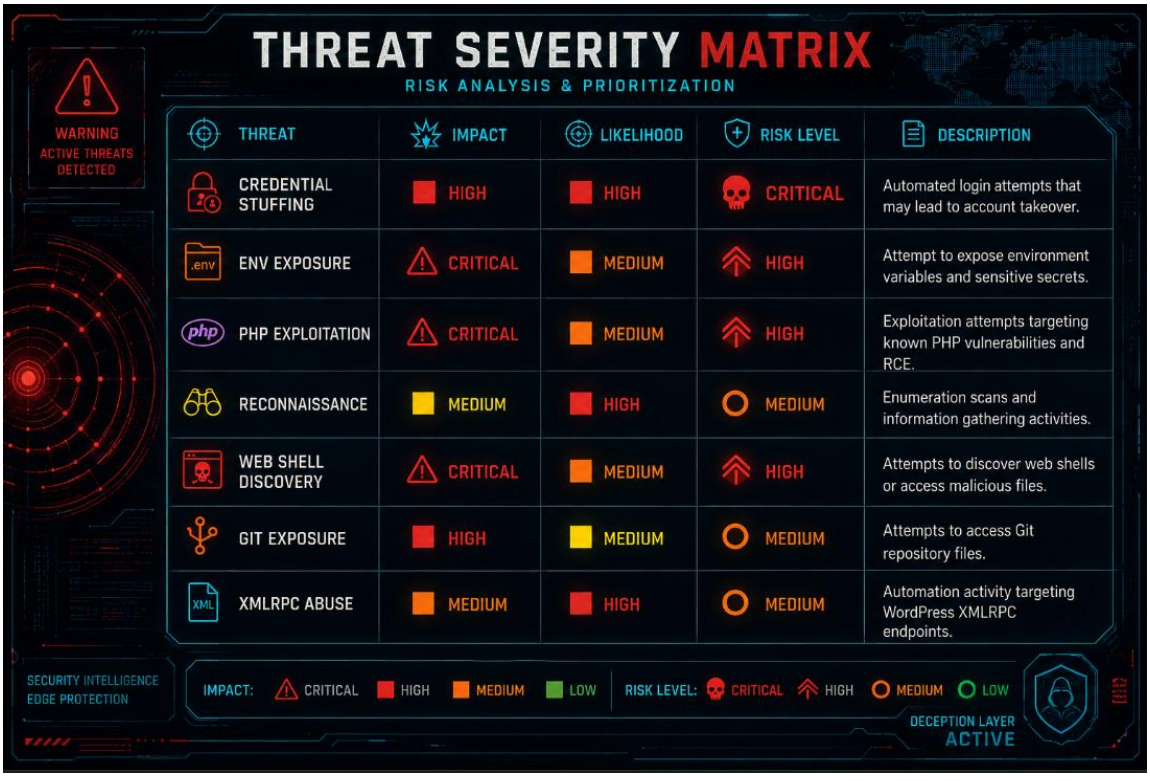
השפעה עסקית

למרות שהאתר אינו מבוסס WordPress או PHP, פעילות ה-Reconnaissance וה-Mass Scanning יצרה סיכונים ממשיים עומס מיותר על שרת המקור, ניסיונות גישה לנתיבים רגישים, סיכון לחשיפת מידע, פוטנציאל לניצול חולשות עתידיות, Credential Abuse, ופגיעה אפשרית בזמינות במקרה של Flooding. השימוש בתשתיות ענן וב-Automation Frameworks העלה את רמת התחכום ואת ה-Scale של הפעילות.

מעבר לנזק הטכני הישיר, קיימת גם עלות עסקית עקיפה משאבי שרת המבוזבזים על טיפול בתעבורה עוינת, פגיעה אפשרית בחוויית המשתמש בזמן עומס, וסיכון מוניטיני במקרה של דליפת מידע. עבור ארגון, גם תקיפה "לא ממוקדת" שנכשלת עלולה לגרור עלויות חקירה, ניטור ותגובה. לכן ההשקעה במניעה ובזיהוי מוקדם ב-Edge משתלמת גם כאשר לא אירעה פריצה בפועל.

הפחתת סיכון והערכה כוללת

הטמעת ה-Honeypot יחד עם WAF איפשרה לצמצם משמעותית את הסיכון באמצעות Edge Protection, Threat Classification, Managed Challenges, Rate Limiting, Honeypot Routing, Deception ואיסוף Threat Intelligence. אף שלא זוהתה פריצה מוצלחת, רמת הסיכון הכוללת הוערכה כ-High בשל היקף הפעילות האוטומטית, השימוש בתשתיות מבוזרות, ניסיונות Exploitation מוכרים ו-Credential Abuse מתמשך. המערכת הצליחה להפחית את ה-Exposure של שרת המקור ולשפר את יכולות ה-Detection.



תוצאות הפרויקט

הישגים מרכזיים

במסגרת הפרויקט הוקמה מערכת Detection ו-Deception מבוססת Cloudflare אשר זיהתה, ניתחה והטעתה פעילות עוינת בזמן אמת. המערכת עצרה ניסיונות Reconnaissance ו-Enumeration ב-Edge; זיהתה Credential Stuffing ו-XMLRPC Abuse באמצעות POST ל-/xmlrpc.php; זיהתה שימוש ב-WordPress API Methods כגון metaWeblog.newPost; מנעה גישה ישירה לשרת המקור באמצעות ניתוב ל-Honeypot; אספה Threat Intelligence מ-Worker Telemetry ו-Firewall Events; זיהתה תשתיות תקיפה מבוססות VPS וענן; וזיהתה User-Agent Rotation ודפוסי Bot Obfuscation. בין האירועים הבולטים זוהה מעבר מ-Reconnaissance בסיסי לניסיונות Exploitation פעילים, כולל שימוש ב-Go-http-client ובכלי Automation נוספים, ובקשות POST חוזרות ל-/xmlrpc.php עם החלפת User-Agent מאותו מקור דפוס מובהק של ניסיון לעקוף Detection.

המערכת הצליחה:

- לעצור ניסיונות Reconnaissance וסריקות Enumeration בשכבת ה-Edge
- לזהות ניסיונות Credential Stuffing ו-XMLRPC Abuse באמצעות POST Requests ל-/xmlrpc.php
- לזהות ניסיונות שימוש ב-WordPress API Methods כגון:
  - metaWeblog.newPost
- לזהות Payloads חשודים ו-Automation Frameworks
- למנוע גישה ישירה לשרת המקור באמצעות ניתוב Requests חשודים ל-Honeypot
- לאסוף Threat Intelligence בזמן אמת מתוך Worker Telemetry ו-Firewall Events
- לזהות תשתיות תקיפה מבוססות VPS ו-Cloud Providers ציבוריים
- לזהות User-Agent Rotation ודפוסי Bot Obfuscation
- לבצע Correlation בין Payloads, Attack Timing ו-Repeated Exploitation Attempts
- ליצור סביבת Deception מלאה אשר הצליחה לגרום לתוקפים לחשוף Payloads, Attack Patterns ו-Credential Attempts

במהלך הניתוח זוהו מספר אירועים משמעותיים אשר הצביעו על מעבר מפעילות Reconnaissance בסיסית לניסיונות Exploitation פעילים. בין היתר זוהו:

- גישה לנתיבי WordPress מוכרים לצורכי Fingerprinting
- שימוש ב-Go-http-client ובכלי Automation נוספים
- מספר בקשות POST ל-/xmlrpc.php אשר כללו XMLRPC Abuse וניסיונות Credential Abuse
- שימוש ב-Campaign Markers ו-Payload Patterns חוזרים
- החלפת User-Agent מאותו מקור תקיפה במטרה לעקוף מנגנוני Detection

יתרונות

המערכת סיפקה מספר יתרונות מבצעיים משמעותיים:

- **Edge Protection** - עצירת פעילות עוינת בשכבת ה-Edge לפני הגעתה לשרת המקור.
- **Reduced Attack Surface** - צמצום משמעותי של שטח התקיפה ושל חשיפת שירותי ה-Web.
- **Improved Visibility** - שיפור יכולות ה-Visibility וה-Telemetry על פעילות עוינת בזמן אמת.
- **Real-Time Detection** - יכולת זיהוי, Classification ו-Threat Scoring בזמן אמת.
- **Threat Intelligence Collection** - איסוף IOC, Payloads, User-Agents ודפוסי תקיפה מתוך פעילות אמת.
- **Deception Capabilities** - שימוש בטכניקות Honeypot ו-Fake Services לצורך הארכת זמן החשיפה של תוקפים ואיסוף מידע נוסף.
- **Credential Intelligence** - זיהוי ניסיונות Credential Abuse ו-XMLRPC Authentication Attempts.
- **Bot & Automation Detection** - זיהוי User-Agent Rotation, Automation Frameworks ודפוסי Bot Activity.
- **Low Deployment Cost** - הטמעה מהירה בעלות נמוכה יחסית באמצעות Cloudflare Workers ו-WAF.
- **Rapid Deployment** - יכולת פריסה מהירה ללא צורך בתשתיות מורכבות או שרתים ייעודיים.
- **Scalable Architecture** - יכולת הרחבה עתידית למנגנוני AI Detection, Threat Correlation ו-Behavioral Analytics.

מגבלות וכיווני המשך

לצד הישגיו, חשוב להכיר במגבלות המערכת. ראשית, זיהוי מבוסס דפוסים ו-Allowlist דורש תחזוקה שוטפת דפוסי תקיפה משתנים, וכלי Automation חדשים מופיעים כל הזמן. שנית, תוקף מתוחכם ובעל מוטיבציה גבוהה (Targeted) עשוי להאט בכוונה את קצב פעילותו ולחקות התנהגות אנושית כדי לחמוק מ-Scoring מבוסס תדירות. שלישית, ה-Deception יעיל בעיקר מול Automation הזדמנותי, ופחות מול תקיפה ידנית ממוקדת.

כיווני ההמשך הטבעיים כוללים שילוב Behavioral Analytics לזיהוי חריגות לאורך זמן, הוספת מנגנוני Machine Learning ל-Scoring דינמי המתעדכן מאליו, העשרת ה-Telemetry במקורות (Feeds) חיצוניים שלThreat Intelligence, והרחבת שכבת ה-Deception לשירותים נוספים מעבר ל-WordPress. כל אלה ניתנים למימוש על גבי אותה ארכיטקטורת Edge, ללא צורך בשינוי מבני.

Legitimate Bot Detection

This section identifies trusted crawlers such as Googlebot, Bingbot and AhrefsBot. Legitimate indexing traffic is logged separately and bypasses the honeypot workflow:

```
const legitBotRules = [
  { name: "Googlebot", category: "search_engine", match: "googlebot" },
  { name: "Bingbot", category: "search_engine", match: "bingbot" },
  { name: "AhrefsBot", category: "seo_crawler", match: "ahrefsbot" }
];

const matchedLegitBot = legitBotRules.find(
  bot => lowerUA.includes(bot.match)
);

if (matchedLegitBot) {
  return fetch(request);
}
```

Exact Match Detection

High-confidence attack paths commonly targeted during reconnaissance and exploitation attempts:

```
const exactBadPaths = [
  "/wp-login.php",
  "/xmlrpc.php",
  "/adminer.php",
  "/phpinfo.php",
  "/.env",
  "/.git/config"
];

if (exactBadPaths.includes(path)) {
  addScore(100, "exact_bad_path");
}
```

Pattern-Based Detection

Searches for suspicious keywords associated with known exploitation techniques:

```
const patterns = [
  { value: "xmlrpc", score: 60 },
  { value: "phpunit", score: 90 },
  { value: "shell", score: 90 },
  { value: ".env", score: 100 }
];

for (const p of patterns) {
  if (path.includes(p.value)) {
    addScore(p.score, `pattern:${p.value}`);
  }
}
```

User-Agent Intelligence

Identifies automation frameworks and offensive security tools:

```
if (
  lowerUA.includes("curl") ||
  lowerUA.includes("sqlmap") ||
  lowerUA.includes("nikto") ||
  lowerUA.includes("masscan")
) {
  addScore(90, "suspicious_user_agent");
}
```

Credential Honeypot Telemetry

Collects credential attack telemetry while masking sensitive data:

```
credentialTelemetry = {
  username: sanitizeText(username),
  maskedPassword: maskPassword(password),
  passwordHash: await sha256(password)
};

addScore(150, "credential_honeypot_submission");
```

Threat Classification Engine

Maps the accumulated score to a threat category:

```
if (score >= 300) {
  classification = "credential_attack_or_critical_exploitation";
} else if (score >= 250) {
  classification = "critical_exploitation_attempt";
} else if (score >= 180) {
  classification = "active_scanner";
}
```

Tarpitting

Introduces artificial delays for high-risk requests:

```
if (score >= 180) {
  await sleep(randomBetween(1500, 5000));
}
```

Threat Telemetry Collection

Logs suspicious activity for threat intelligence and correlation:

```
console.log(JSON.stringify({
  type: "HONEYPOT_HIT",
  ip,
  country,
  path: url.pathname,
  score,
  classification,
  matchedRules
}));
```

Deception Layer

Returns fake responses that simulate vulnerable services:

```
function generateFakeResponse(path, method) {
  if (path.includes("wp-login")) {
    return wordpressLoginPage("");
  }

  if (path.includes(".env")) {
    return "APP_ENV=production";
  }
}
```

סיכום

במסגרת Operation GhostEdge הוקמה מערכת הגנת Web מודרנית המבוססת על Cloudflare WAF, Cloudflare Workers, ארכיטקטורת Threat IntelligenceHoneypot וטכנולוגיית Deception. המערכת אפשרה לזהות, לנתח ולהטעות תוקפים בזמן אמת, תוך הגנה על שרת המקור וצמצום משמעותי של ה-Exposure. הפרויקט הדגים כיצד ניתן להשתמש ביכולות Edge Security להקמת סביבת Detection ו-Deception מודרנית המספקת Prevention, Detection, Threat Intelligence, Telemetry ו-Deception כאחד.

מעבר לכך, הפרויקט הוכיח כי גם תשתית Web קטנה יכולה לאסוף מודיעין סייבר מבצעי בזמן אמת, באמצעות שילוב של Threat Scoring, Worker Telemetry, Honeypot Routing, אינטגרציית WAF וזיהוי בזמן אמת.

ממצאי החקירה הצביעו על פעילות Automation מתקדמת: WordPress Fingerprinting, XMLRPC Abuse, Credential Stuffing, ניסיונות Exploitation, Bot Obfuscation ו-User-Agent Rotation - והמערכת הצליחה לגרום לכלי התקיפה לחשוף Payloads אמיתיים, Attack Workflows ו-Credential Attempts בתוך סביבה מבוקרת.

המסקנה המרכזית היא ששילוב בין Cloudflare Edge Security, ארכיטקטורת Honeypot ו-Threat Intelligence מאפשר ליצור שכבת Detection ו-Deception מבצעית גם ללא תשתיות Enterprise מורכבות - תוך שמירה על Deployment מהיר, Visibility גבוהה ועלות נמוכה יחסית. הפרויקט מהווה הדגמה מעשית לכך שהגנה אפקטיבית אינה נמדדת רק ביכולת לחסום, אלא גם ביכולת ללמוד מן התוקף ולהפוך את התקיפה עצמה לנכס מודיעיני.

על המחבר

גיא תותחני, עוסק ב-CTI ו-SecOps ניתוח תוקפים, Threat Hunting, Honeypots, Deception, Technologies, אבטחת יישומי Web ופיתוח מנגנוני Detection. מתמקד בחיבור בין עולמות ההגנה וההתקפה לצורך הפקת מודיעין סייבר מעשי ושיפור יכולות ההגנה של ארגונים.

מאמר זה מבוסס על מחקר ויישום מעשי של טכניקות Detection, Threat Intelligence ו-Deception בסביבת Web מודרנית.

אשמח לענות על כל שאלה ואם צרכים כל עזרה בנושא זה! זמין ב-linkedin

מקורות מידע

- T1595 - Active Scanning
- T1190 - Exploit Public-Facing Application
- T1505 - Server Software Component (Web Shell)
- T1590 - Gather Victim Network Information
- curl
- wget (GNU)
- sqlmap
- nikto
- Brute-force attack
- Reverse proxy
- Autonomous System (ASN)
- Credential stuffing
- Cloudflare Workers - תיעוד
- Cloudflare Workers - מוצר דף
- Cloudflare WAF - תיעוד
- Cloudflare WAF - Custom Rules / Managed Challenge
- Cloudflare WAF - Managed Rules
- Cloudflare - Rate Limiting Rules
- MITRE ATT&CK - הבית דף
- T1110 - Brute Force
- Masscan
- Honeypot (computing)
- Tarpit (networking)