digital whisper Technologic Papers

LAIR - מאורת המטמון של אליבאבא - חלק ב'

מאת ישי גרסטל וליאור ברש

הקדמה

אחרי שדיברנו על המערה ששומרת לנו על האוצר היקר שלנו, כל המידע שלנו כבר מחכה להיכנס למערה, מה שחסר לנו הוא רק שלב הזנת המידע למערכת.

תקציר הפרקים הקודמים (מהו LAIR)?

LAIR היא פלטפורמה, שנותנת מענה לצורך שעולה במבדקי חדירות מתמשכים ומבדקים עם מספר בודקים שרוצים לשמור על מיטביות כלל הבדיקות לאורך זמן ולמנוע עבודה כפולה וחוסרי מידע.

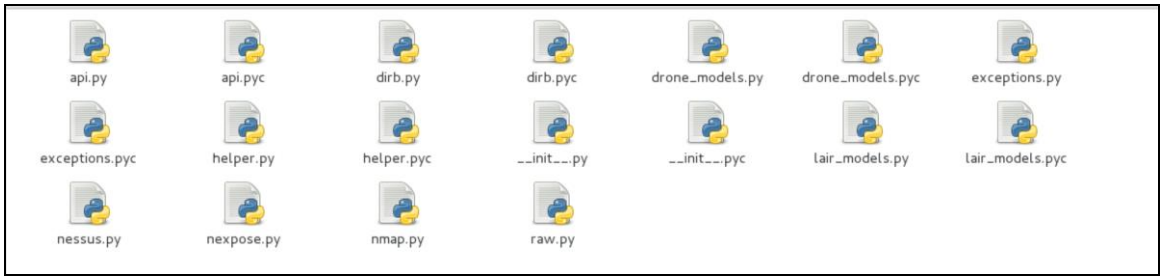
הפעם אנחנו רוצים להרחיב מעט על מכלול האלמנטים בפלטפורמה, איך הם עובדים ומתקשרים אחד עם השני וכיצד הם יוצרים הרמוניה במערה השיתופית.

במאמר הקודם התמקדנו בפונקציונאליות של המערכת עצמה, כיצד היא בנויה ומה הן הפונקציות בתוך ממשק הניהול. הפעם אנחנו רוצים לבחון יותר על הרכיב שמזין ומסדר את כל השלל במקום שאנחנו רוצים שיהיה. נעים להכיר - Drone.

ה-Drone בנוי כולו בפייטון והוא מורכב משתי תיקיות, האחת תיקיית קוד ראשית והשניה מכילה את הסקריפטים שישמשו אותנו ב-CLI.

האופן שבו אנחנו מבצעים את הזנת המידע למערכת הוא על ידי שימוש בפקודה מתוך תיקיית ה-bin שפונה אל תקיית ה-lairdrone שבעצמה מכילה קבצי קוד בפייטון שמחולקים באופן עקרוני לשתי

קטגוריות, הקטגוריה הראשונה היא השמירה על מבנה, סכמה אם תרצו של מבנה המידע הכללי של הפרויקט, מעין תבנית של סדר קבוע שאינה תלויה במי מזין את המידע. הקטגוריה השנייה אחראית על המרת התוצרים שייצאנו מכל אחד מהכלים שהשתמשנו בו בכדי שיתאים לסדר שיצרנו.



אפשר לראות בתמונה את הקבצים מהתיקיה ואת מבנה השמות שלהם המקשר אותם לכלי העבודה כמו גם קבצים השייכים לארכיטקטורה, כגון api.py, drone_modules.py וכו'.

קובץ ה-api.py אחראי על כל מה שקשור בהתחברות לבסיס הנתונים, החל מאימות שם המשתמש והסיסמא, החזקת כתובת ה-IP איתה אנו מתקשרים וכלה בהפרדה וזיהוי של החלקים בקובץ אותו אנו רוצים להעלות לתוך המערה שלנו המכילים מידע כמו מי הנתקף ומה הם התוצרים שמצאנו. כאשר ברקע של כל זה מופעל תהליך המוודא שבמערכת שלנו לא קיים כבר התוכן שאנו מעלים כעת כדי שלא ליצור כפילויות מידע. אותו התהליך אחראי גם לזהות מה כבר הזנו למערכת וממיין את המידע כך שמידע קיים יישאר כמו שהוא ושהדלתאות יעודכנו מבלי לפגוע בקיים.

בשלב הזו שבו אנו מזינים את המידע, אנחנו נדרשים להזין פרטי חיבור כמו כתובת IP, פורט ושם משתמש וסיסמה, טריוויאלי וחיוני. לא פחות חשוב מכך זו הגדרת הגישה של ה-drone למערכת על גבי טווח מוצפן, בכל זאת מדובר בגישה לאחד מבסיסי הנתונים אולי הכי רגישים שלנו ושל הלקוחות שלנו.

```
[redacted]:~$ export MONGO_URL=mongodb://[redacted]@192.168.1.100:110
14/lair?ssl=true
```

עכשיו, על גבי ערוץ מאובטח ומאומת הגיע הזמן להזין את המידע למערכת וישירות לתוך הפרויקט הספציפי עליו אנחנו עובדים. את ההבחנה בפקוירט הספציפי אגב, עושים בעזרת הגדרת PID רלוונטי.

```
[redacted]:~/lairdrone-1.0.0/bin$ ./drone-nmap CWNcuBfQLn7nNP6fc [redacted]
/lair.xml
[+] Attempting connection to database '192.168.1.100:11014/lair'
[+] Connection successful.
[+] Processing project CWNcuBfQLn7nNP6fc
[+] Processing completed: 1 host(s) processed.
```

במקרה הזה כמו שאפשר לראות, אנחנו מעלים תוצאות סריקה של NMAP, כאשר עוקב אחרי שם הקריפט האחראי להעברת תוצאות NMAP אפשר לראות את ה-PID ומייד אחריו את הנתיב לקובץ ה-XML שיצאנו עם תוצאות הסריקה.

אז המידע הוזן למערכת, הקפדנו להעביר אותו באופן מאובטח והקפדנו להזין את המידע לפרויקט המתאים, עכשיו הגיע הזמן לבחון כיצד המידע מסודר לתוך הפרויקט באופן עקבי ותואם. בעניין הזה הגדרנו קודם כאחד התפקידים של ה-api.py כמי שאחראי לניתוח המידע והחלוקה שלו לקטגוריות.

על האחריות לסדר הכללי, מבנה היצוג, טאבים, משתנים הקשורים לכל מחשב, כל הפרטים אודותיו כמו פורט, שירות, נתונים מזהים וכן הלאה, אחראים drone.models & lair.models. התפקיד שלהם הוא ליצור למענינו מעין תבנית קבועה שלא משנה מי מכניס מידע, או גם עם אילו כלים הוא השתמש, המידע יגיע למקומו בשלום, ולא, ישמור השם, ישבר ויהרס בדרך.

עד כאן יש לנו ביד מערכת קולבורציה לתקופים ומנגנון הזנת מידע, עכשיו השאלה היא איך מקימים אותה אילו כלים נתמכים במערכת הכדי לאפשר הזנה של תוצאות. הנה זה בא...

לעבודה!

בשלב הראשון צריך להוריד את התוכן של ה-LAIR framework אל השרת עליו נרצה להקים את המערכת שלנו מתוך סביבת ה-GIT של LAIR בכתובת:

https://github.com/lair-framework/lair

מההורדה נקבל קובץ 7 שדורש מאיתנו לחלץ מתוכו את המידע כדי להתחיל לעבוד. החילוץ עצמו מתבצע בעזרת הפקודה:

```
7za x lair-v1.0.5-linux-x64.7z
```

השלב הבא הוא הפעלת השירות בפעם הראשונה, כך שיתקין את כל השירותים הנחוצים שחולצו כבר קודם כדוגמת MongoDB אנחנו משתמשים בסקריפט שהכינו לטובת הנושא מראש והוא נקרא start.sh שמתפקידו להעלות את כל השירותים הנחוצים כדי שהכל ינגן בהרמוניה.

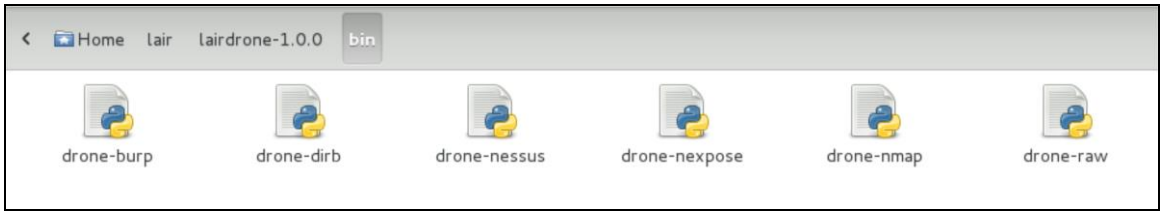
את הפקודה אנחנו מריצים בצירוף כתובת ה-IP שאנחנו רוצים לקשורת את השרות אליה ושתשמש לגישה לממשק ה-Web. במהלך השלב הזה יש שני דברים חשובים שצריך לשים לב אליהם. בפעם הראשונה אנחנו יוצרים שני משתמשים, האחד תפקידו להיות משתמש מנהל לבסיס הנתונים אליו נזין את המידע אודות הלקוח, והמשתמש השני אמון על בסיס הנתונים של ה-LAIR עצמו.

על כל פנים, אותנו כרגע יותר מעניין המשתמש שתפקידו לנהל את בסיס הנתונים של האפליקציה, זה כיוון שבכל פעם שנרצה לכבות או להפעיל את ה-LAIR אנו חייבים להכניס את המשתמש והסיסמא שלו כדי שיתאפשר לנו לקבל את נתוני האמת שכבר הכנסנו למערכת ולחסוך מאיתנו את הצורך להזין כל סריקה למערכת נפרדת. בנוסף המשתמש הזה משמש גם לצורך שליחת מידע על ידי ה-Drone.

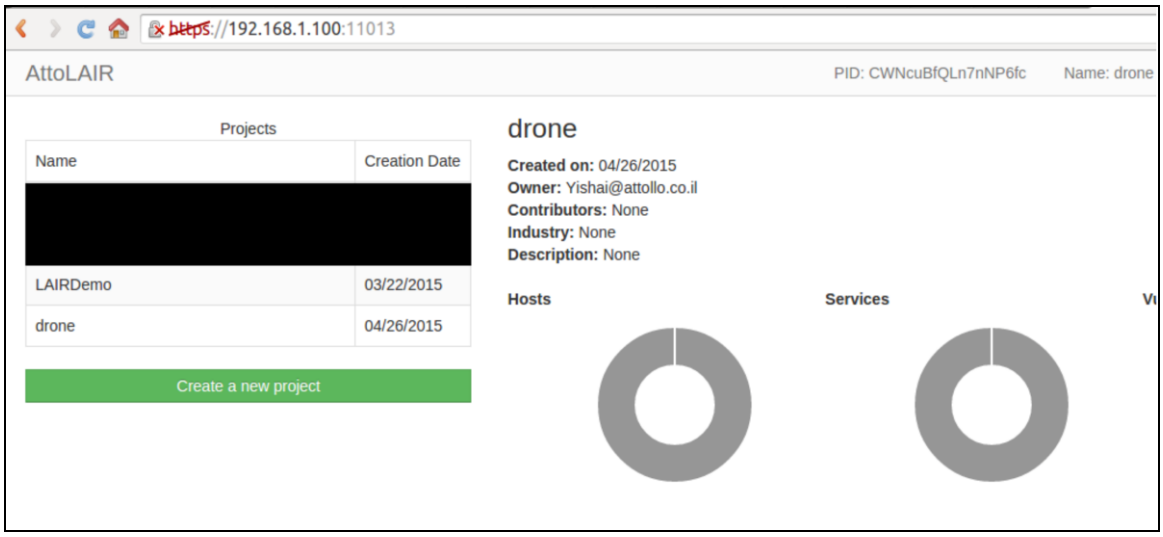
מייד לאחר הקמת המערכת בפעם הראשונה נוכל להתחבר עם שם משתמש וסיסמה של מנהל המערכת, שזה טוב ונפלא אבל המשתמש הזה יכול לראות את כל הפרויקטים שרצים במערכת וזה כבר פחות טוב. מה שנרצה לעשות זה לפתוח משתמש לכל תוקף ולהגדיר את הרשאות הגישה למערכות הרלוונטיות שעליהן התוקף עובד.

המערכת באוויר. מה שחסר זה מידע... כאן אנחנו פונים למגוון הכלים שנתמכים היום על ידי המערכת. כבסיס המערכת מגיעה עם מספר מצומם של drone-ים לכלים מוכרים, הבחירה בפיתוח ה-drone-ים דווקא לכלים אלו היא שרירותית ובאופן עקרוני ניתן לבנות תהליך יבוא לכל כלי שתרצו.

את אפשרויות היבא הנתמכות נכון לעכשיו אפשר לראות בתמונה.



הקוד בקבצים האלו, מטרתו להמיר את המידע שנשמר בתצורת xml מכל אחד הכלים הנתמכים היישר אל המקום הנכון במערכת וכמובן שבמבנה הנכון.



מכאן השליטה היא שלנו, באחריותו של מנהל המערכת למעשה לאפשר גישה לכל אחד מהתוקפים להזדהות מול המערכת. משתמשי המערכת מצידם יכולים ליצור להם פרויקטים כאשר באחריותם להחליט אם הפרויקט שיצרו הוא עצמאי והגישה אליו מוגבלת להם בלבד או שיבחרו לשתף משתמשים אחרים במערכת ולאפשר להם לגשת למידע הרלוונטי לפרויקט שיצרו.

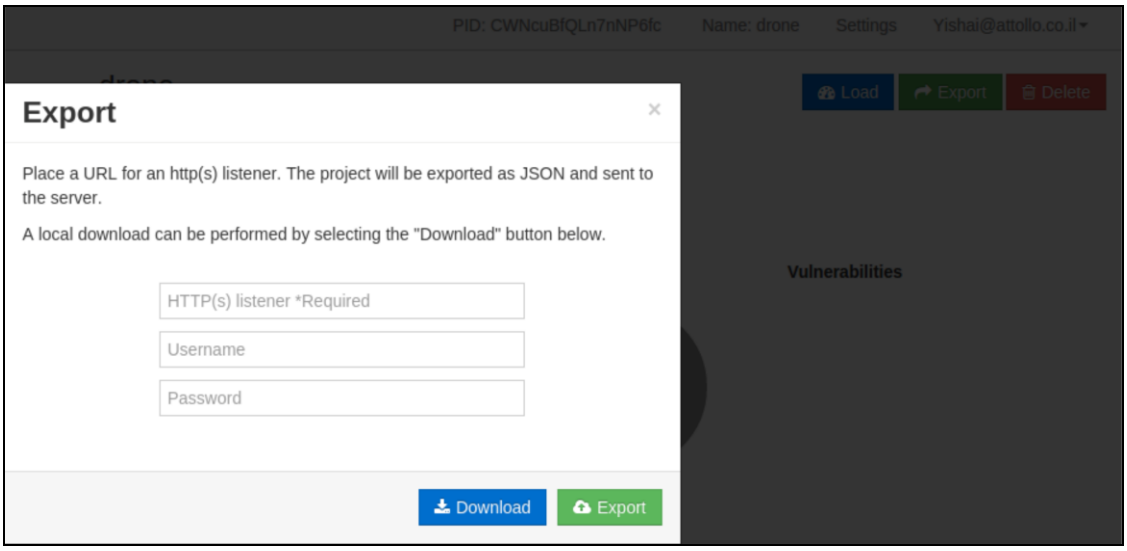
כאן חשוב להזכיר שוב כי מנהל המערכת, הלא הוא המשתמש הראשי של בסיס הנתונים מלפני מספר פסקאות יכול לראות את כל הפרויקטים ולצפות בלוגים של בסיסי הנתונים שהלכה למעשה מכילים את כל המידע במערכת. אם נמשיך את אותו הקו, נראה שישנה בעיה קלה בשיטה... אף משתמש לא יכול להעלות מידע למערכת מבלי לדעת את הסיסמא לבסיס הנתונים. ובמערכת רגישה כל כך אנחנו לא בהכרח רוצים שהפרטים האלו יהיו בידהם של כל המשתמשים במערכת. כאן יש לנו שלוש אפשרויות עקרוניות:

- **האפשרות הראשונה** היא שכל אחד ממשתמשי המערכת יעביר את תוצאות העבודה שלו למנהל המערכת והוא בתורו יזין את המידע לבסיס הנתונים. החסרון העיקרי של שיטת העבודה הזו הוא יצירת נקודת כשל אחת. שאלה נוספת שעלה היא האם נרצה בכלל שתהיה בידיו של משתמש כל שהוא גישה למערכת באופן מלא לצורך שימוש קבוע ויומיומי.

- **האפשרות השניה** מסבכת מעט את התמונה, יצירת בסיס נתונים נפרד לכל משתמש אליו המשתמש יכול לשלוח את המידע, אך אותו משתמש, או כל משתמש אחר במערכת לצורך העניין לא יקרא משם את המידע. אם לדייק, הוא יכול אבל אין בכך צורך. לאחר שהמידע הוזן לבסיס הנתונים הזמני, מנהל המערכת יכול להתחבר לשם בעצמו ולהעתיק את כל החומרים של המשתמשים השונים ולהעלות אותם לבסיס הנתונים המרכזי ואפשר אם תרצו להפוך את התהליך לאוטומטי.

- **האפשרות השלישית** היא יצוא של פרויקט שלם, אפשר ליצא את הפרויקט כולו ישירות למערכת או לחילופין לשמור את הפרויקט כולו למחשב עליו אנחנו עובדים. על היכולת הזו אחראי drone-raw כאשר הוא מיצא את הפרויקט לבתנית JSON שנוכל להעלות לתוך בסיס נתונים לפי בחירתנו או כפרויקט חדש.

כך זה נראה:



סיכום

המערכת עצמה כמו שאפשר לראות בנויה באופן יחדית פשוט MongoDB, משמש כבסיס הנתונים הראשי, האפליקציה עצמה בנויה על NodeJS וכוללת בתוכה את כל השירותים הנדרשים. ההתקנה מאוד פשוטה ומסתכמת כמעט כולה בהרצה של סקריפט בודד. ה-drone-ים השונים מסדרים את כל המידע במקום ואחראים על שני תפקידים עיקריים, האחד פנייה לבסיס הנתונים והשני הוא סידור המידע לפי במערכת כך שכל המידע שהוצאנו מכלל הכלים יגיע בדיוק למקום המיועד לו ובאחידות.

כאשר כל הדברים מתנהלים על מי מנוחות, לא אמורות להיות תקלות והכל מנגן ביחד את שירת התקיפה של אליבבא והמערה השיתופית.

בפרק הבא: כותבים Drone!

על הכותבים

- **ישי גרסטל** - שד טזמני וחובב קוד פתוח, מחבר וחוקר מערכות בעיקר כדי לפרק אותן.
- **ליאור ברש** - חופר 24/7 ומעניש על זה את המקלדת.