Technologic Papers

אבולוציה של רף אבטחה

מאת ליאור ברש

TL;DR

אוטומציה לסריקות אבטחה והפקת דוחות זמינה ובמסלול האצה לכיוון אוטונומיה מוחלטת של מחקר, מימוש ומניעת התקפה. ואז מתקינים golismerO עם חברים.

בין אבולוציה ואוטונומיה

אחד הגורמים המשפיעים ביותר על הרף התחתון של מידת האבטחה בסייבר-ספייס הוא מידת או רמת ההבנה והיכולת של אנשי המקצוע. כאשר השוק מספק מערכות בעלות רמת אבטחה נמוכה או כאשר אנשי אבטחה מספקים מערכות או ניתוחי מערכת שטחיים ומורכבים להבנה ע״י הלקוחות, לצרכני השרות לא נותר אלא להמצא מבלי שרצו בכך, בסיכון גבוהה מאשר הם מאמינים שהם חשופים לו. כחלק ממענה לבעיה הזו, השוק אימץ מודל שרות של מבדקי חדירות וסריקות אבטחה כדי לקבל תמונה יחסית מייצגת למידת הסיכון אליה חשופות תשתיות הארגון ומערכות המידע.

על אף שהמודל מצטייר מבטיח, בשל העלויות הגבוהות הכרוכות בביצוע מבדקים מקיפים, נוצרה מגמה שמטרתה להפוך את מרבית העבודה הרפיטטיבית שאינה דורשת ניתוחים מתקדמים והתערבות משמעותית, לאוטומטית. כך בעצם אפשר בשעות ספורות לסיים עבודה שבעבר לקחה ימים ושבועות ולהשאיר את התקציבים הפנויים לבחינת פוטנציאלים משמעותיים וביצוע בדיקות מתקדמות שנכון להיום לא ממוכנות ולא מאפשרות אוטומציה מלאה.

כחלק מהמגמה הזו, לפני מספר שנים בדדות הוקם פרויקט קוד פתוח בשם golismerO, הפרויקט מפותח ע״י שלושה תורמים עקריים שגם הקימו אותו ומטרתו בעצם להוות פלטפורמת אינטגרציה ואוטומציה למערך רחב של כלים המאפשרים אוטומציה מלאה למרבית או לכלל יכולות הסריקה והניתוח שלהם. הפלטפורמה כולה בנויה פיתון ומתוקף כך היא נתמכת על כל מערכות ההפעלה העיקריות, בנוסף המערכת מאפשרת שילוב כלים נוספים או חדשים לתהליך העבודה שלה כך שבהחלט אפשר להתאים את הפלטפורמה לצורכי המשתמש.

לפני שנראה כיצד מקימים מערכת כזו וכיצד מרחיבים את יכולותיה עם כלים נוספים בהם היא יכולה להשתמש לצורך ניתוח מרחב האיומים של הסביבה אותה היא בוחנת, נבחן את הסטאטוס שלנו אל מול היעד - מערכות מחקר, תקיפה והגנה אוטונמיות לחלוטין.
אגב, golismerO לא מממשת מחקר, מניעה או מימוש התקפה כלל, מדובר על מערכת מסגרת (framework) שמאפשרת אינטגרציה ואוטומציה של סריקות אבטחה.

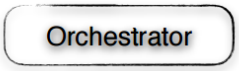
אמצע אוגוסט 2016, ממש מעט אחרי הסבב השני של תחרות ה-CGC ב-DEFCON אותה הניעה ומימנה DARPA, הרבה מאוד מהערפל סביב היכולת לקיים מערכת אוטונומית לחלוטין שתבצע מחקר חולשות, בחינת יתכנות מימוש החולשות ויצירת טלאי מיידי למניעת החולשה התפוגג, ה"עתיד" ממש כאן, אפשרי, מרגש ומאיים. המצב היום בשוק מאפשר מידה רבה של אוטומציה לתהליכים רבים, אולם אחת המגבלות המשמעותיות לאוטומציה מלאה, היא היכולת למדל תהליכי התנהגות אנושית לשפה שתוכל לשמש מערכת ממוחשבת ולהביא, אפילו רק למידה קרובה של אפקטיביות המאפיינת תהליך עבודה אנושי. כל זה עוד לפני שבכלל בחנו את האפשרות והממשות של הוספת ממדי בינה מלאכותית לתהליכים אוטומטיים והפיכה שלהם לתהליכים אוטונומיים.

אם כך המצב, אז האתגר מורכב לא רק מהיכולת הטכנולוגית אלא גם מהיכולת למדל את תהליכי ההתנהגות ובפרט מנגנוני שיקול הדעת האנושית לתבנית מורכבת יותר מאשר מערך החלטות בינארי, בין אם מדובר במערכי חוקים לינארים או בתהליכי ניתוח מושכל מבוסס מכונה דוגמאת מרבית מודלי הבינה המלאכותית כיום או אם מדובר בתהליכי למידה מבוססים רשתות נוירוניות או מודלים קוגנטיבים, ישנו לפחות ברמת הביצוע כיום פער משמעותי בין הניתן לקיים.

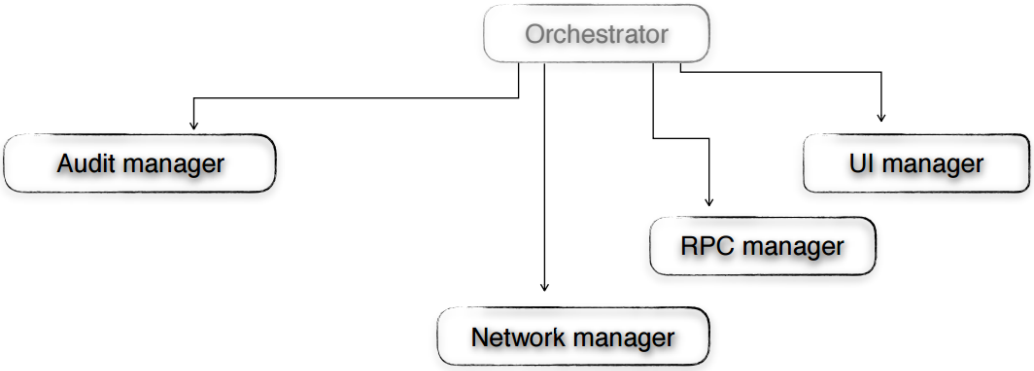
על אף הפער בין הניתן לקיים ישנו פער קטן יותר וקל יותר לגישור, פער הידע כהכרות עם קיומן של אפשרויות. לטובת גישור הפער הזה, נכתבות שורות אלו, כדי שיכל כל מי שרוצה, להקים לעצמו מערכת אוטומטית למימוש סריקות אבטחה והפקת דוחות סיכון.

איך זה עובד?

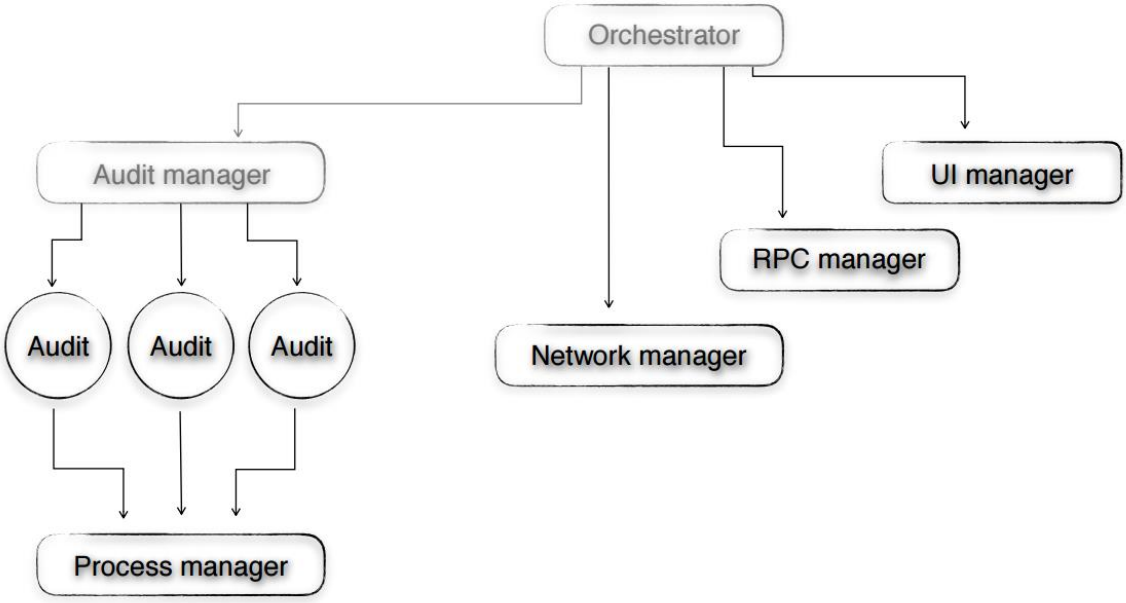
מרכז הפעילות של המערכת הוא ה-orchestrator שממנו הכל מתחיל ובו הכל מסתיים.



הוא עצמו אחראי ישירות על ארבעה מנהלי דומיינים שונים.

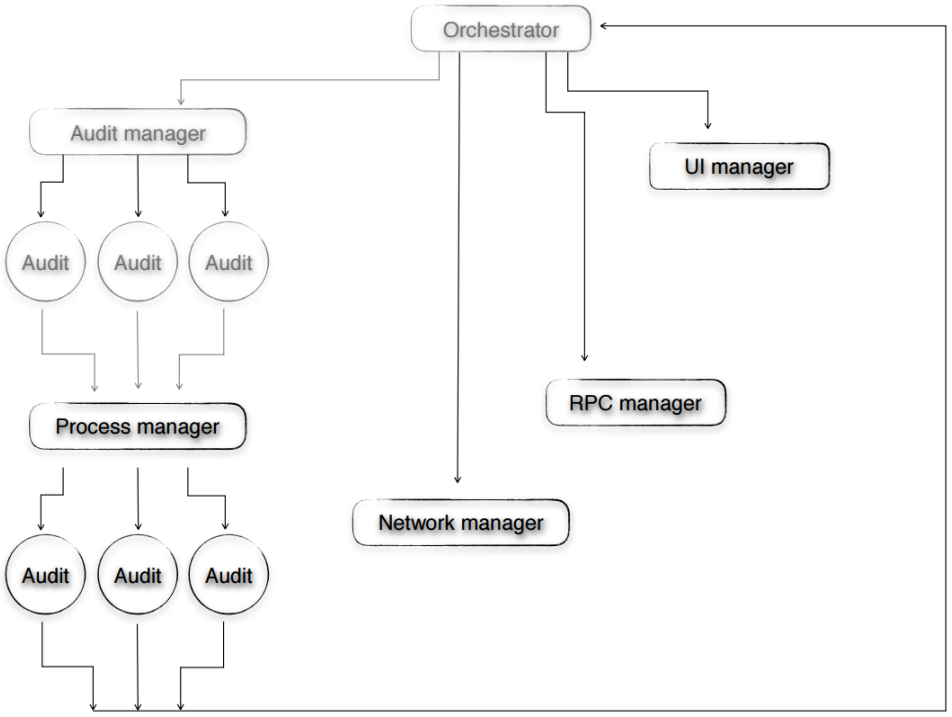


כל קלט במערכת מגיע לתור-קלט ב-orchestrator והוא זה מייצר את כל מנהלי הדומיינים והאובייקטים תחתיו. קלט שרלוונטי לניהול תהליכי ה-audit מופנה בהתאמה למנהל הדומיין כדי להתחיל או להפסיק סריקה מול ערוץ יעודי שאחראי מאותו הרגע לעקוב ולנהל את כל האובייקטים שלו עצמו, בין אם מדובר על כתיבה למסד נתונים או מעקב סטאטוס אחר תנועת האובייקטים מול הפלאגינים השונים, את המידע הזה מעבירים ל-למהל דומיין חדש, מנהל התהליכים.

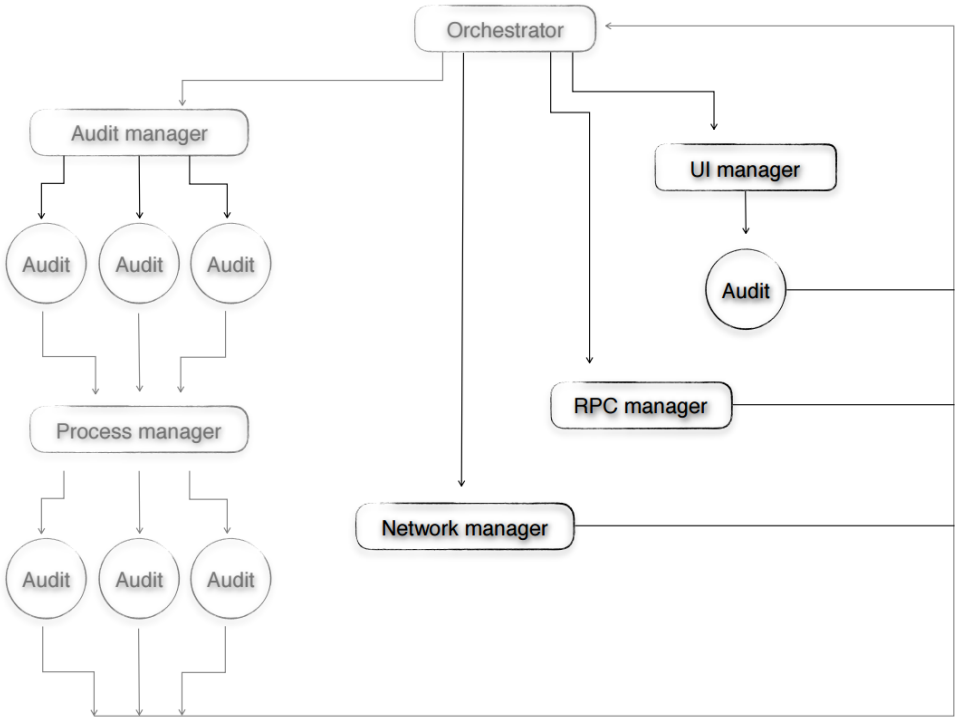


מנהל התהליכים, מנהל, כצפוי, תת-תהליכים. אותו מנהל תהליכים מחזיק שני תורים, אחד למידע שמגיע מאוביקטיי ה-audit והשני למידע שמועבר ממנו לתוספים (plugins) אותם הוא מנהל. המידע שמגיע מאובייקטי ה-audit מגדיר איזה מידע הולך לאיזה תוסף ותפקידו של מנהל התהליכם לקבוע איזה תת

תהליך ינהל את התוסף הרלוונטי ויקים עבורו את סביבת הביצוע שלו. כאשר סיים תוסף את הפעילות שלו או לחילופין כאשר תוסף רוצה להעביר מידע לרכיב אחר במערכת הוא פונה ישירות ל-orchestrator.



תפקידם של שלושת מנהלי הדומיינים הנותרים דיי תואם את שמם: מנהל ממשק התקשורת אחראי על ניהול כלל הקישוריות לרשת. מנהל ה-RPC אחראי על מסירת ההודעות במרחב ה-API. ומנהל ממשק המשתמש מקבל עותק של כל ארוע במערכת אותו הוא מזרים דרך תוסף יעודי לו לממשק המשתמש.



ישנם עוד שלושה מנהלי דומיינים שלא הוזכרו עד כה והם לא חלק מהלופ. מנהל תוספים שתפקידו לנהל את טעינת התוספים ולקרא מקבצי המערכת (golismero.). מנהל ייבואים שנטען עם הדגל import--בתחילת סריקה ותפקידו לייבא מידע מוקדם במידה וישנו. מנהל הדוחות שנטען עם הדגל report--ותפקידו לכתוב את תוצאות הסריקה לקובץ שהוגדר בפורמט שהוגדר.

בונים לגו כלים

כאמור golismerO היא מערכת מסגרת שממממשת כלים קיימים, ומאגדת את המידע המצטבר מכלל הכלים לכדי מאגר מידע אחוד, אפשר לייבא למערכת תוצאות קיימות ואפשר לבצע בעזרתה סריקות שונות שיניבו תוצאות שונות, כך או כך, תהליך הקמת המערכת והשימוש בה פשוט מאוד.

המערכת כאמור רצה על כל סביבה שמאפשרת הרצה של פיתון, אנחנו נראה איך מריצים אותה על סביבה מבוססת ubuntu 14.04. תחילה נתקין את כל הדרישות להרצת הסביבה ועוד כמה דרישות להרצת כלים שנוסיף בהמשך.

שלב ראשון:

```
sudo apt-get install python2.7 python2.7-dev python-pip python-docutils git perl nmap sslscan
```

נתקין את תשתית הפיתון הנדרשת, את git שאיתו נמשוך את קובצי המקור של המערכת, את perl שתדרש בהמשך ועוד שני כלים נוספים שאיתם נבצע סריקה בסיסית, nmap ו-sslscan.

שלב שני:

```
sudo git clone https://github.com/golismero/golismero.git
```

נמשוך בעזרת git את קוד המקור של המערכת, מומלץ להוריד אותו לתוך תקיה יעודית רק כדי לשמור על הסדר, אבל לא חובה. לאחר שסיימנו את השכפול נריץ את התקנת דרישות המערכת.

```
sudo pip install -r requirements.txt
```

במקרה שלנו כאשר מערכת ההפעלה מבוססת לינוקס נריץ גם את דרישות המערכת לסביבות לינוקס.

```
sudo pip install -r requirements_unix.txt
```

שלב שלישי:

```
sudo ln -s /opt/golismero/golismero.py /usr/bin/golismero
```

ניצור קישור לקובץ ההפעלה של המערכת.

זהו, המערכת מוכנה לעבודה ואפשר להתחיל לסרוק, רק בעיה אחת, זה היה פשוט מדי!

אז למי שרוצה כבר לראות איך הסריקה הבסיסית נראית כל מה שצריך לעשות זה להריץ את הפקודה:

```
golismero TARGET
```

הסריקה הבסיסית יכולה להיות מופעל גם ע"י הוספת הדגל --scan אבל זה לא חובה. ואפשר לראות את כל מגוון האפשרויות עם הוספת הדגל h-.

כיוון שלא הוגדר למערכת מה לעשות עם תוצאות הסריקה, ברירת המחדל שלה תהיה להדפיס את התקדמות הסריקה ואת התוצאות ישירות למסך.

בונים לגו גיבורים

עכשיו כשיש לנו מערכת עובדת, אפשר להרחיב אותה ע"י הוספת יכולות אינטגרציה מול מכלים נוספים. אחד הכלים המעניינים שהמערכת תומכת בהם הוא openVAS. בכמה מילים, openVAS הוא פיצול (fork) של nesus שמהווה היום את האלטרנטיבה המקיפה ביותר למערכת סריקת חולשות מבוססת קוד פתוח, מנעד היכולות שלו נע מסריקות תשתית לסריקות אפליקטיביות והוא מקבל עדכונים על בסיס שוטף. openVAS לעומת golismerO, פחות טריוויאלי להקמה. אז קדימה.

גם במקרה הזה אנחנו נשענים על מערכת ההפעלה ubuntu בגרסא 14.04.

שלב ראשון:

```
sudo apt-get install -y build-essential devscripts dpatch libassuan-dev \ libglib2.0-dev libgpgme11-dev
libpcre3-dev libpth-dev libwrap0-dev libgmp-dev libgmp3-dev \ libgpgme11-dev libopenvas2 libpcre3-dev
libpth-dev quilt cmake pkg-config \ libssh-dev libglib2.0-dev libpcap-dev libgpgme11-dev uuid-dev bison
libksba-dev \ doxygen sqlfairy xmltoman sqlite3 libsqlite3-dev wamerican redis-server libhiredis-dev
libsnmp-dev \ libmicrohttpd-dev libxml2-dev libxslt1-dev xsltproc libssh2-1-dev libldap2-dev autoconf
nmap libgnutls-dev \ libpopt-dev heimdal-dev heimdal-multidev libpopt-dev mingw32 \ make git screen
rsync sudo software-properties-common alien nsis rpm \ libcurl4-gnutls-dev w3af-console python-
setuptools pnscan netdiag slapd ldap-utils snmp \ ike-scan zip texlive-latex-base texlive-latex-extra texlive-
latex-recommended htmldoc
```

מייד אפשר לראות את הפער בין שתי המערכות רק מכמות דרישות המערכת וכיוון שמדובר בכמות גדולה! של דרישות הפעם לא נמנה את כולן.

שלב שני:

```
git clone git://github.com/sstephenson/rbenv.git .rbenv
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(rbenv init -)"' >> ~/.bashrc
sudo apt-get install rbenv
exec $SHELL

git clone git://github.com/sstephenson/ruby-build.git ~/.rbenv/plugins/ruby-build
echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
exec $SHELL
```

```
git clone https://github.com/sstephenson/rbenv-gem-rehash.git ~/.rbenv/plugins/rbenv-gem-rehash
sudo apt-get install -y libreadline-dev

rbenv install 2.2.3
rbenv global 2.2.3
```

לאחר הקמת כל דרישות הבסיס מתקיניםruby ו-מקימים סביבה דרך rbenv.

שלב שלישי:

```
sudo add-apt-repository ppa:mrazavi/openvas
sudo apt-get update
sudo apt-get install openvas
```

מוספים את המקורות (repository) להתקנה ולאחר עדכון מריצים את ההתקנה עצמה של openVAS.

שלב רביעי:

```
sudo openvas-nvt-sync
sudo openvas-scapdata-sync
sudo openvas-certdata-sync

sudo service openvas-scanner restart
sudo service openvas-manager restart
sudo openvasmd --rebuild --progress
```

מעדכנים את מאגרי המידע של המערכת ומאתחלים את המערכת. קחו בחשבון שהשלב הזה עלול להיות די ארוך.

שלב חמישי וכמעט אחרון:

```
wget https://svn.wald.intevation.org/svn/openvas/trunk/tools/openvas-check-setup --no-check-certificate

chmod 0755 openvas-check-setup
sudo ./openvas-check-setup --v8 --server
sudo chmod +x /var/lib/openvas/plug
```

מוודאים שההתקנה עברה בהצלחה, אם יהיו שגיאות או חוסרים כל שהם הבדיקה הזו תצביעה עליהם ותציע דרכים אפשריות לתיקיון ולאחר מכן מעדכנים את כל התוספים של openVAS ל-x+.

שלב שישי ואחרון:

```
wget https://github.com/sullo/nikto/archive/master.zip
unzip master.zip
cd nikto-master/program
sudo cp * /usr/local/bin/
ln -s /usr/local/bin/nikto.pl /usr/bin/
```

```
wget -O wapiti-2.3.0.tar.gz "http://downloads.sourceforge.net/project/wapiti/wapiti/wapiti-2.3.0/wapiti-2.3.0.tar.gz?r=http://sourceforge.net/projects/wapiti/files/wapiti/wapiti-2.3.0/&ts=1391931386&use_mirror=heanet"

tar zxvf wapiti-2.3.0.tar.gz
cd wapiti-2.3.0
sudo python setup.py install
sudo ln -s /usr/local/bin/wapiti /usr/bin/
```

לגמרי לא חובה אבל אם תרצו אפשר להוסיף כלים נוספים ש-openVAS יודע לעבוד איתם, במקרה הזה אני מוסיף את nikto ואת wapiti להרחבת היכולות האפליקטיביות.

עכשיו כשכל החלקים במקום, כמעט טאפשר להתחיל לסרוק ולמפות, מה שנותר הוא לעדכן את קובץ ההגדרות של golismerO היכן הוא יכול למצא את openVAS וכיצד הוא יכול לגשת אליו.

```
touch ~/.golismero/user.conf
chmod 600 ~/.golismero/user.conf
vim ~/.golismero/user.conf

[openvas]
host = <openVAS_host>
user = <openVAS_username>
*password = <openVAS_pass>
```

נבחן מספר דוגמאות להרצה לפני שנעבור לסיכום.

את ההפעלה הבסיסית ראינו מייד אחרי ההתקנה הראשונית של golismerO, עכשיו נראה איך מוסיפים עוד כמה אפשרויות לשורת הפקודה כדי להפוך את הריקה למקיפה ומשמעותית יותר.

```
golismero scan -e openvas TARGET
```

הוספת הדגל e- תומר למערכת לבצע סריקה בעזרת התוסף שנציין ואפשר לשרשר את הדגל e- מספר פעמים כדי לטעון את כל התוספים שנרצה לטעון.

```
golismero plugins
```

יגרום למערכת להציג בפנינו את רשימת כל התוספים הנתמכים, ואם הוספנו אותם נוכל לקרא להם עם ישירות מפקודת ההרצה.

עוד יכולת מעניינת של golismerO היא האפשרות להעביר את הבדיקה דרך פרוקסי:

```
golismero scan -pu USER -pp PASS -pa ADDRESS -pn PORT TARGET
```

הפקודה דיי מסבירה את עצמה. ובמידה ונרצה לקבל את תוצר הסריקה כקובץ עצמאי נשתמש בדגל o- אותו נוכל להפנות למספר פורמטים כמו html, json, csv, xml, yaml, rst, txt ואולי המעניין מבניהם, הכתיבה ל-db.

מסקנות ומחשבות לעתיד

אולי המימד האהוב עלי ביותר בכל הנושא של אוטומציה ואוטונומיה של מערכות, הוא החופש שיכולות אלו מביאות איתן, בחופש אני מתכוון לזמן שמתפנה מפעולות רפיטטיביות שעשינו כבר אלפי פעמים והלימוד שלנו מהן נמוך אם בכלל קיים. הסוג הזה של חופש בעצם מהווה את הקרקע להתפתחות וזה תמיד מרגש. יש איזו תחושה מעורבבת בשנים האחרונות שנעה בין קצה אחד של הסקאלה בו החדשנות שקועה עמוק מתחת ערמות של buzzwords וטכנולוגיות מכובסות, לבין הצד השני של הסקאלה בו אנו עומדים מול מרחב יצירה חדש שמאפשר לנו ליצור דברים שכמותם לא היו.

לרוב בכל תקופה אפשר לחוש את המגמה בברור, אך כעת, לתחושתי ולתחושת רבים שאני משוחח איתם על הנושא נראה שאנו דורכים בשתי הקצוות גם יחד, זה אומר רק דבר אחד, דברים חדשים בדרך!

אז תפנו זמן ותתחברו ליצירה שבכם. זהו, מכאן כל מה שנותר זה לחגוג.