digital whisper
Technologic Papers

מבוא ל-Fuzzing

נכתב ע"י בנישתי איל

הקדמה

תחום מחקר החולשות הוא תחום שצובר תאוצה בימים אלה, יותר ויותר חברות מוכנות לשלם היום לחוקרים (Bounty Programs) על מנת שיחשפו חולשות במוצרים שלהם ובכך בעצם יעזרו להם לשמור על המוצרים שלהן בטוחים יותר.

חברות גדולות כמו גוגל אף משיקות ועידות מתוזמנות ומזמינות האקרים וחוקרים מכל העולם לנסות את מזלם ולזכות בפרסים של עד כ-60K$.

המאמר הקודם שכתבתי התרכז בתחום פיתוח האקספלוייטים ובעצם כרונולוגית שייך לאותו הרגע שבו נמצאה איזושהי חולשה, לרוב חולשה שקשורה בניהול הזיכרון של התוכנה, ואז החוקר נדרש להוכיח שהחולשה ניתנת לניצול על מנת להריץ קוד זדוני או במקרה הפחות נעים לגרום למניעת שירות.

השלב שקודם לפיתוח האקספלוייט (או הוכחת ההיתכנות) הוא בעצם השלב של מחקר החולשות.

השיטות בתחום מחקר החולשות מתחלקות לשני תחומים עיקריים:

א. ניתוח סטטי - באמצעות שימוש בכלים לסריקת קוד (כולל שלב ההידור) ואו באמצעות ניתוח ידני.
ב. ניתוח דינמי - מתקשר ישירות לנושא שלנו, ניתוח התנהגות התוכנית בזמן ריצה ע"י הזנה של קלטים שונים ומשונים.

הוויכוח שקיים על איזו שיטה היא היעילה יותר, הינו ארוך, נוקב ומגיע לו התייחסות משלו. לכל הדעות השימוש בשתי השיטות גם יחד יניב תוצאות טובות יותר.

מבוא לפאזינג

אז מה זה בעצם פאזינג?

שיטת בדיקה אוטומטית שעיקרה בהזנה של קלטים אקראיים ו\או לא צפויים עבור תוכנה מסוימת בציפייה לגרום לה לקריסה, שבמקרה שלנו, תוביל למציאת חולשת אבטחה.

השיטה הוצגה לראשונה באוניברסיטת וויסקונסין שבארה"ב ע"י פרופ' ברטון מילר, יש הטוענים שהעבודה במקור החלה בעקבות השראה שהתקבלה מקריסה של תוכנות בשעת סופת ברקים בעקבות רעשי קו שנוצרו בעת חיבור מודם.

מכיוון שמדובר בשיטת בדיקה, גם הטרמינולוגיה מגיעה מעולם הבדיקות ולכן עיקרי השלבים בפאזינג הם:

- חילול אוטומטי של מקרי בדיקה (Test Cases).
- הרצה של המון מקרים אל מול המטרה.
- ניטור התוכנה הנמצאת תחת בדיקה וחיפוש שגיאות או קריסות.

הקלטים אותם אנו שולחים אל התוכנית יכולים להיות מסוגים שונים, בין אם קבצים בפורמטים שונים (כגון PDF) או מבוססי רשת, למשל SNMP או FTP.

בעוד השלבים האחרונים הם טכניים לחלוטין, השלב הראשון של יצירת מקרי הבדיקה הוא בעצם השלב המכריע שבו מתרכז המאמץ, שכן הצלחה או כישלון במאמץ לחשוף חולשת אבטחה טמונה ביצירת מקרה הבדיקה "הנכון" שיחשוף את הבאג, הפאזר צריך להצליח היכן שהמפתח והבודקים של התוכנה במקור בעצם נכשלו.

לפנינו שתי שיטות עיקריות ליצירת מקרי הבדיקות, "החכמה" ו-"הטיפשה", לכל אחת יתרונות וחסרונות בולטים, על פי רוב כדאי להשתמש בשתיהן כדי להשיג כיסוי נרחב יותר ובעצם להגדיל את הסיכוי למצוא חולשה חדשה.

הערת אגב: את תחום הפאזינג או השיטה ניתן ליחס גם למחקר חולשות בתחום אפליקציות האינטרנט, כגון SQLi ו Xss.

פאזינג "טיפש" (אקראי\מוטציוני)

שלא כשמה, היא שיטה יעילה למציאת חולשות אבטחה ולראיה הבאג הזה (שנמצא בנגן הפלאש של אדובי).

מקור השם הוא בעובדה ששיטה זו מסגירה את העובדה שמקרי הבדיקה לא מתבססים על שום ידע מוקדם לגבי הפרוטוקול או מבנה הקובץ בו משתמשים כקלט.

האנומליות בקלט מתבססת לרוב על קלטים קיימים שעוברים שינוי מוטציוני אקראי, הן ע"י שינוי ביטים והן על ידי הוספה אקראית של תוכן לקלט, למשל:

<html>Fuzz me</html> → Bit Flipping → <htm1>Fuzz m3</htm2>

<html>Fuzz me</html> → Random content → <html>Fuzz me33333</html>

בדוגמא ניתן לראות, כי יצירת מקרי הבדיקה מתבססת על דוגמה (לרוב תמימה וחוקית) של HTML שממנה אנו מנסים ליצר קלטים שעלולים להכשיל את הדפדפן (במקרה הספציפי הזה).

לעיתים, פאזרים "טיפשים" מתוחכמים יותר מנסים יוריסטיקות שונות כגון: הזזת תווים או העלמת ערכים מאוד מסוימים כמו NULL או 1-.

יתרונות

מאוד קל לכתוב פאזר "טיפש", אין צורך להכיר את מבנה הקובץ או הפרוטוקול ולהתחקות אחריהם. כל מה שצריך זה אוסף גדול של קלטים חוקיים. חיפוש פשוט בגוגל יכול להגריל אלפי קבצים מסוגים שונים, מאוד פשוט להשיג אלפי דוגמאות של קבצי PDF למשל ע"מ לבדוק קוראים שונים.

חסרונות

- מוגבל לפרמוטציות השונות על אוסף הקלטים הראשוני - יכול להיות שדווקא המקרים המאוד מעניינים לא מופיעים בקבוצת הקלט, בעיקר אם הם לא בשימוש נרחב (לא מתועדים ו\או לא נפוצים), ושם למרבה הפלא (או שלא) מתחבאים הבאגים המעניינים.
- פרוטוקולים מסוימים וקבצים אוכפים את תכפות התוכן ע"י CRC ו-CHECKSUMS, פאזר "טיפש" יפספס את הנקודה הזו עבור הרוב המוחץ של הקלטים ובעצם לא יעבור את שלב האימות (שאלת כיסוי הקוד היא שאלה מאוד מעניינת ונדון בה בהמשך).

פאזינג "חכם" (מחולל)

מקרי הבדיקה מחוללים על סמך RFC ו\או מסמכים שונים שמתארים את מבנה הקובץ או הפרוטוקול. כותב המחולל מאתר את נקודות ההתערבות החכמות לדעתו ושם הפאזר בעצם ישתול ערכים "רעים" (ארוכים ו\או מיוחדים כגון NULL, סימנים כמו % וכו').

במאמר מוסגר, פאזר "חכם" אמור לתת תוצאות טובות יותר מפאזר "טיפש", כמובן שהוא יפספס מקרים מאוד מסוימים שהראשון ימצא.

גם בעולם הפאזרים החכמים נולדים כל הזמן פתרונות טובים יותר ויותר, כגון פתרונות מבוססי אינסטרומנטציה שבאים על מנת להשיג כיסוי קוד נרחב יותר ולעזור לפאזר "לחלחל" עמוק יותר בקוד, ולעקוף תנאים קשים שלולא העזרה הזו לא היו קורים, בפתרונות האלה הפאזר שותל מעין סוכן חכם בקוד של תוכנת היעד ועובד איתו בשיתוף מלא, פתרונות כאלה קיימים גם בעולם הקופסא השחורה וגם הלבנה וחברות מובילות כמו גוגל ומיקרוסופט משתמשות בהן בשילוב עם מחוללי קוד מבוזרים על מנת להשיג תוצאות טובות עוד יותר (ואף בשיטות כמו עיבוד מבוזר).

מכיוון שהמאמר הנוכחי הוא ברמת מבוא אני אשאיר את ההתעמקות בנושא האחרון לקורא.

יתרונות

- שלמות, מכיוון שיש בידינו ידע על המבנה אנחנו יכולים לחולל בעצם את כל מקרי הבדיקה כולל אלה שנמצאים בשימוש נמוך שם בחוץ, אנחנו לא מסתמכים על איסוף דגימות!
- אנחנו יכולים להתמודד עם דרישות מורכבות יותר כמו CHECKSUM.

חסרונות

- חייבים גישה למסמכים או כל מקור אחר שמתארים את המבנה.
- כתיבת מחולל כזה עלולה להיות עבודה מאוד קשה במיוחד אם הפרוטוקול מסובך.
- בשורה התחתונה מסמכים הם לא קוד, דברים ממומשים אחרת בפועל ולפעמים נכתב קוד שלא מתועד בדיעבד.

מתי להפסיק? או "כמה פאזינג זה מספיק פאזינג?"

די ברור שבפאזינג "טיפש" אפשר ליצר כמות כמעט אין סופית של בדיקות ולעומת זאת בפאזינג "חכם" כמות הבדיקות היא סופית אבל האם עדיין מתחבא שם באג שלא מצאנו? השאלות האלה מתעוררות במיוחד כשהבאגים מסרבים לצוץ ולהעביר אותנו לשלב הבא של הניתוח והוכחת ההתכנות. אחת התשובות האפשריות לשאלה הזו טמונה בין היתר בכיסוי הקוד אותו הצלחנו להשיג בסבב הנוכחי.

יש כלים רבים שיכולים לעזור בשאלת כיסוי הקוד כגון GCOV, חלק עומדים בפני עצמם וחלקם כבר כלולים בסביבות פיתוח פאזרים.

יש כמה סוגים של בדיקת כיסוי והם מתחלקים ל:

- כיסוי שורות - כמה שורות קוד רצו.
- כיסוי ענפים - כמה הסתעפויות שונות נבדקו, נכון בעיקר לכיוונים שונים שנבחרו במשפטי התניה (IF).
- כיסוי נתיבים - נתיבים שלמים שכוללים מספר הסתעפויות.

דוגמא:

```
If (x > 2)
    x=2
if (y > 2)
    y=2
```

כמה בדיקות צריך ע"מ לכסות כל סוג?

תשובה - אחת בשביל כיסוי שורות, 2 עבור כיסוי ענפים ו-4 עבור כיסוי נתיבים.

האם כיסוי קוד מלא מבטיח שנמצא את כל הבעיות? כמובן שלא! הרי בסופו של דבר רוב השגיאות הן לוגיות ומצריכות קלטים מאוד מסוימים, אבל זו דרך טובה להשוות בין פאזרים ולקבל מושג כללי לגבי כמה קוד בדקנו והכי חשוב האם נתקענו אי שם בהתחלה או בדקנו נתיב מאוד ספציפי.

כלים נפוצים

כלים נפוצים רבים ובמיוחד סביבות פיתוח לפאזרים לעיתים מספקים לנו בין היתר סביבות מנוטרות וכלי עזר לפיתוח הפאזר בלי צורך לרדת לעומקי המימוש של הכלי עצמו תוך התמקדות בפרוטוקול אותו אנו בודקים, הם כולם מלווים בעצם את התהליך הכולל של פאזינג והוא:

1. זיהוי המטרה - מהי האפליקציה אותה אנו בודקים.
2. זיהוי וקטורי התקיפה - מה הקלט עבור מושא הבדיקה? קובץ? בקשה על הרשת?
3. הכרת הקלט - מהו סוג הקלט ומהי הפלטפורמה המתאימה בשבילו.
4. חילול מקרי הבדיקה.
5. שליחת הקלטים אל האפליקציה.
6. ניטור - חיבור דיבאגר בצורה ידנית או שימוש ביכולת מובנית של הפלטפורמה ו\או ניטור לוגים.
7. ניתוח - מעבר על דו"ח הקריסות וההחלטה האם זוהי בעיה הניתנת לניצול.

הרשימה להלן אינה מלאה, ישנם מספר רב של כלים, לכל אחד היתרונות והחסרונות שלו, צריך פשוט להתנסות בהם ולאמץ כמה מהם ע"מ להשיג תוצאות טובות, מחקר כלים ובחירתם הוא שלב מאוד חשוב בתהליך ולא כדאי לזלזל בו, הרבה מאוד חוקרים כותבים כלים יעודיים משלהם שעונים על צרכים מאוד מסויימים.

ZZUF - אחד הפאזרים המוטציונים הנפוצים ביותר, משתמש בשיטת הפיכת ביטים אקראית. הוא לא גרפי אבל מאוד יעיל ופשוט לשימוש.

Peach Fuzzer - פאזר "חכם" שהוא גם מחולל וגם מוטציוני, עובד בעזרת קבצי PIT בפורמט XML שמגדירים את מבנה הקלט. מגיע עם כלי ניטור ולוגים ובעצם משאיר לנו רק את הצורך למדל את הקלט (הדוגמאות והמדריך באתר מאוד ממצים). לצורך ניטור הפלטפורמה משתמשת בשני רכיבים - סוכן ומנטר, הסוכן יכול להיות מותקן בצורה מקומית או מרוחקת ולהגדיר מנטר על מנת לבצע פעולות כמו הסנפת הרשת, חיבור דיבאגר לתהליך וניטור זיכרון.

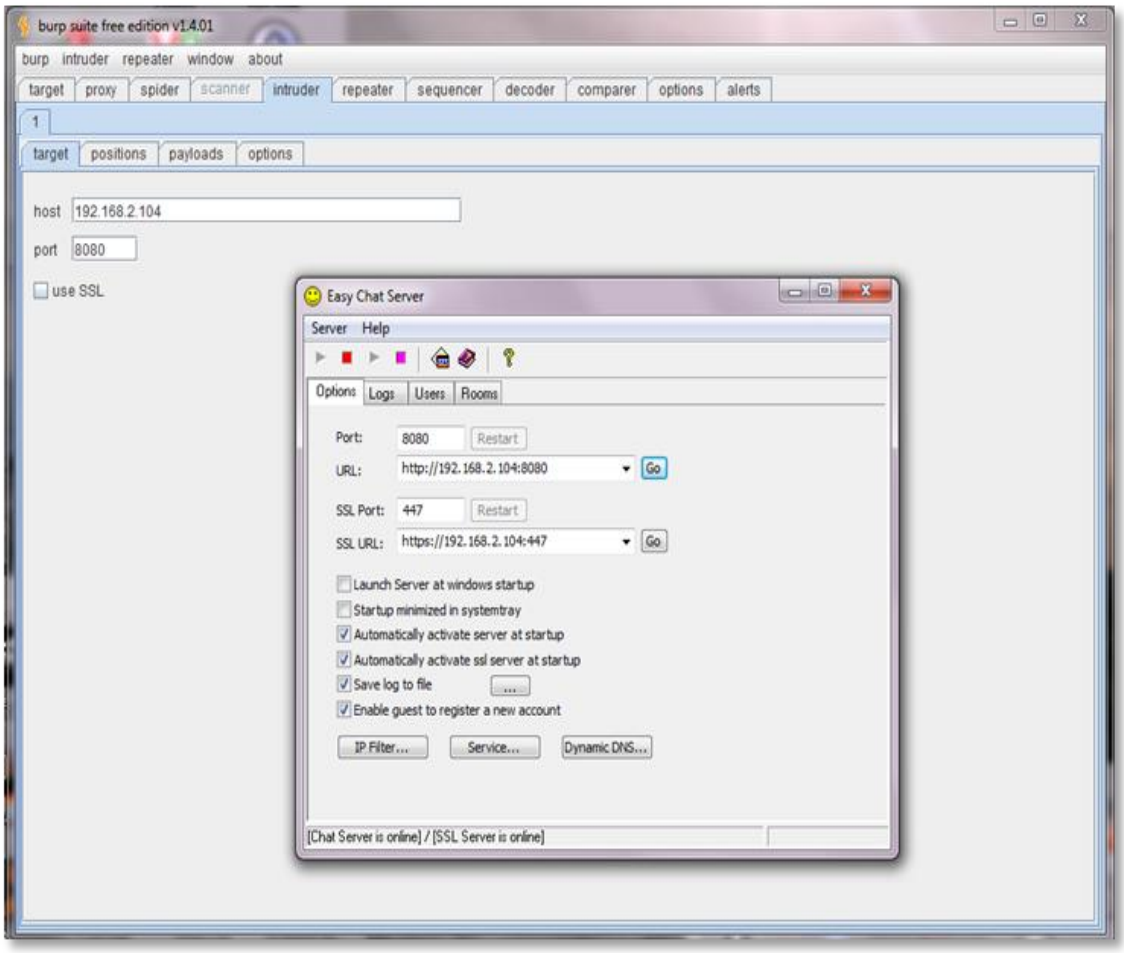
Sulley - הפאזר לאיתור חולשות מרוחקות הטוב ביותר שקיים לטעמי כיום, אומנם מצריך את היכולת לכתוב סקריפטים בפייטון אך מביא עימו יכולת רבות כולל ניטור התהליך, איתחול והמשך הפאזינג במידה וקרס, ממשק ווב שמציג מצב, ניתוח קריסות ועוד. אפשר למצוא דוגמא מצויינת לפאזינג של שרת FTP כאן.

Burp Suite - פלטפומה משולבת לבדיקת אפליקציות אינטרנט, מגיעה עם שרת פרוקסי מובנה למיפוי בקשות ושליחתן לכלים השונים שמגיעים עם הפלטפורמה לצורך בדיקה, לפלטפורמה כלים שונים והיא ניתנת להרחבה.

סוגרים מעגל

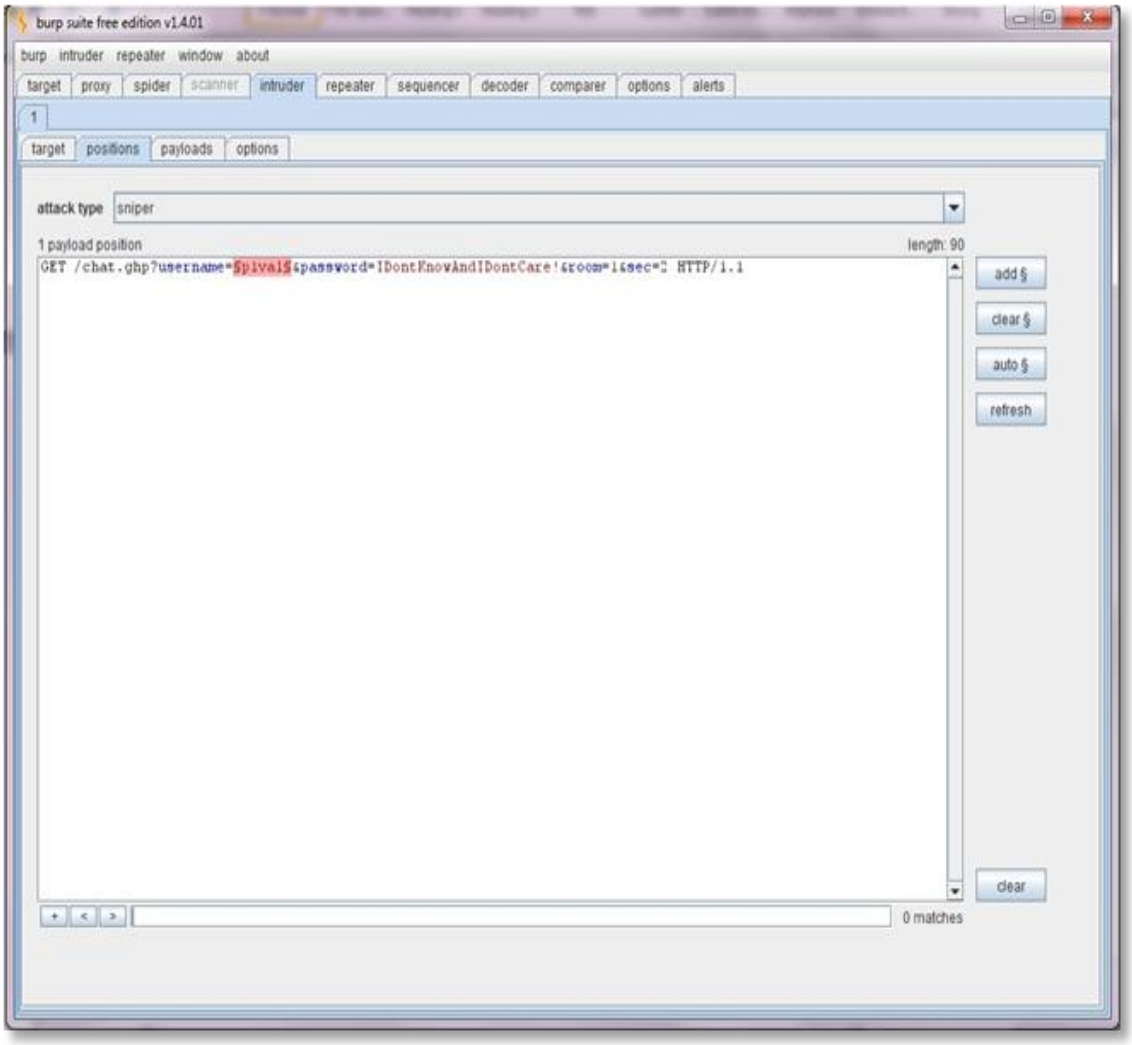
בחרתי ב-Burp Suite כדי להראות איך יכול היה להראות המחקר של שרת הצ'אט בו השתמשתי במדריך הקודם שכתבתי. דילגתי על שלב המיפוי עם שרת הפרוקסי משום שאני יודע מראש איך נראית בקשת ההזדהות, להזכירכם הבקשה שאותה אני הולך לבדוק אחראית על תהליך הזהוי של הלקוח של מול השרת, כפי שכבר ניתן להבין וקטור הקלט הוא בקשת HTTP ואם להיות יותר ספציפי אז הפרמטרים אשר מרכיבים את הבקשה.

בשלב הראשון הרצתי את שרת הצ'אט וביקשתי ממנו להאזין על פורט 8080. בשלב השני הרצתי את Burp Suite ובחרתי בלשונית 'Intruder' (הכלי שבעצם יעשה בשבילי את הפאזינג), בתת הלשונית 'Target' הגדרתי את כתובת השרת והפורט, ניתן לראות את שניהם רצים זה לצד זה בתמונת הבאה:

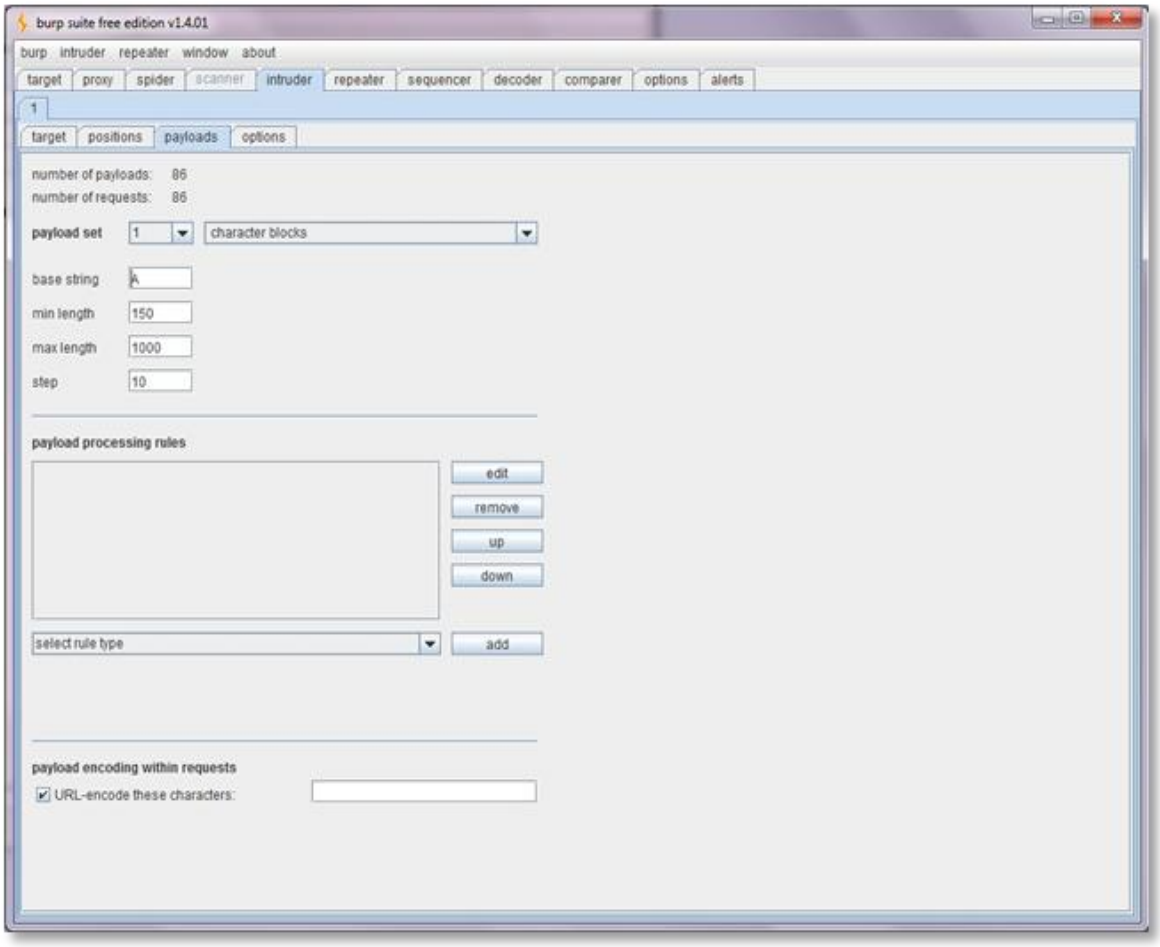


כשלב שלישי, בתת הלשונית 'Positions' הגדרתי איך בעצם נראית בקשת האימות (במקרה הזה הקלדתי ידנית אך זו יכלה להיות תוצאה של "חטיפה" באמצעות הפרוקסי ושליחה לאינטרוודר, מאוד אינטרקטיבי!)

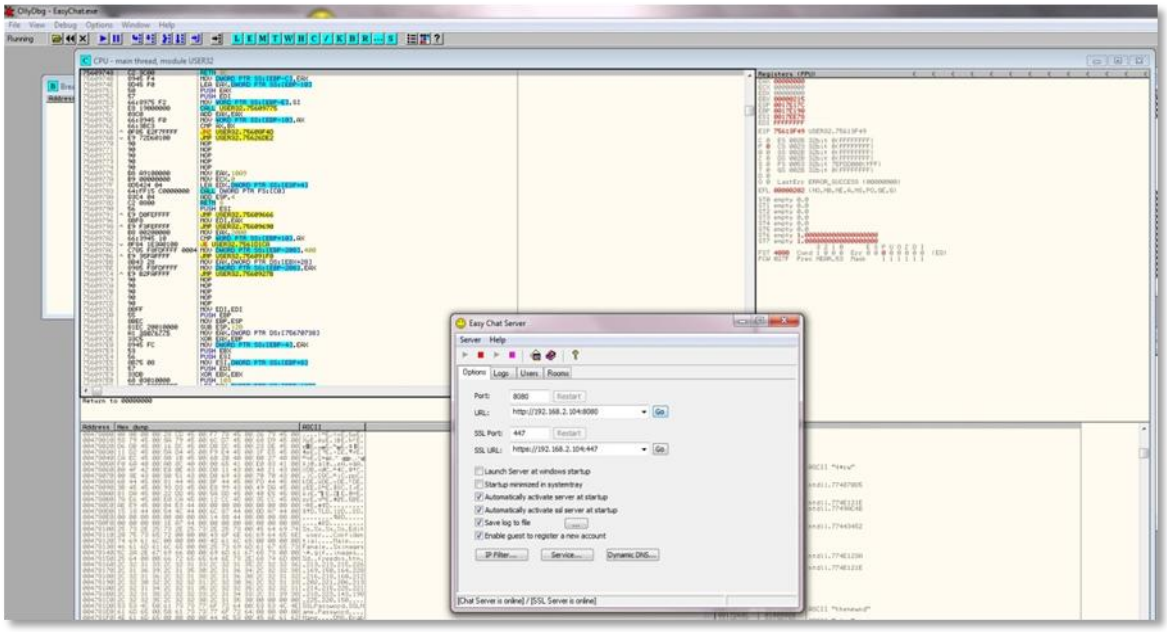
בחרתי בשיטת ההתקפה 'sniper' מכיוון שאני בודק רק פרמטר אחד (username) ואני רוצה לבדוק payload אחד בכל רגע נתון, ניתן ללמוד על השיטות השונות כאן. הוספתי את הבקשה וסימנתי את הערך של שם המשתמש (§p1val§) כמטרה עבור ה- payload:



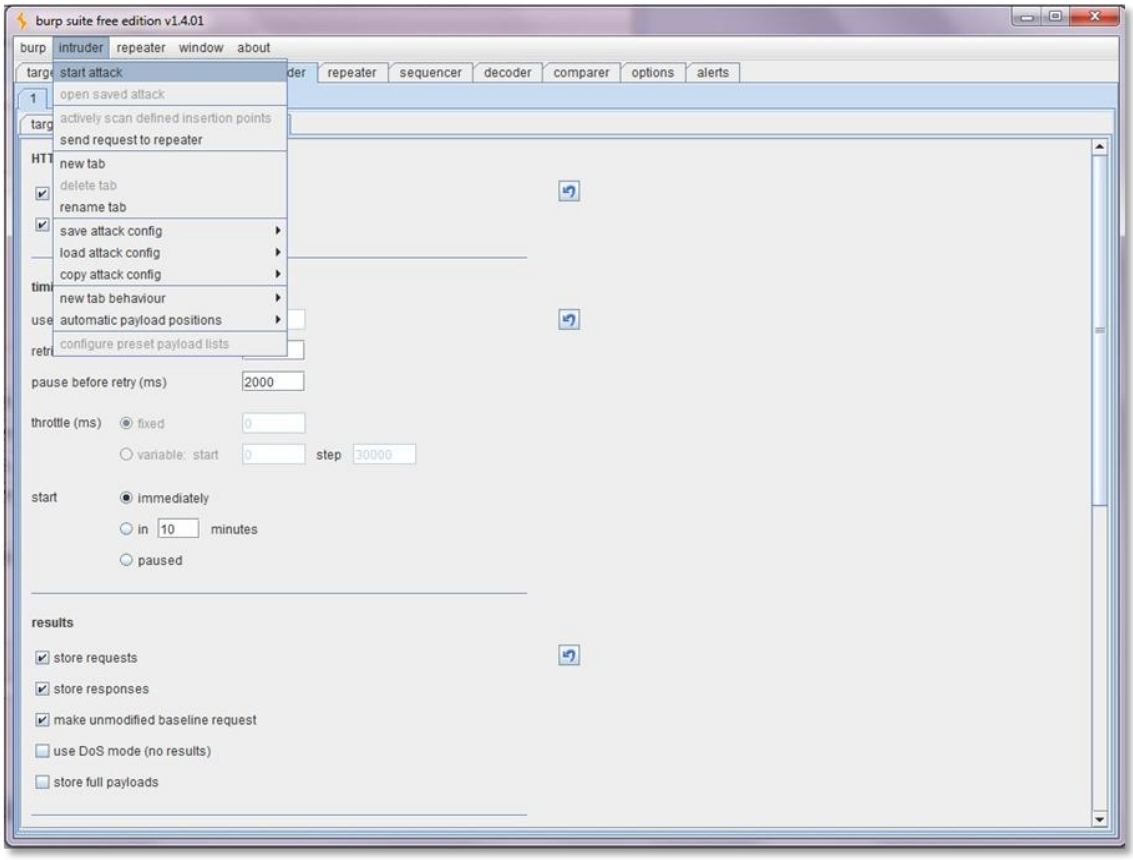
בשלב הרביעי הגדרתי את ה-payload, בחרתי בסט אחד של בלוק אותיות ('A') בגודל משתנה מ-150 ל-1000 בקפיצות של 10 (אני מחפש גלישת מחסנית):



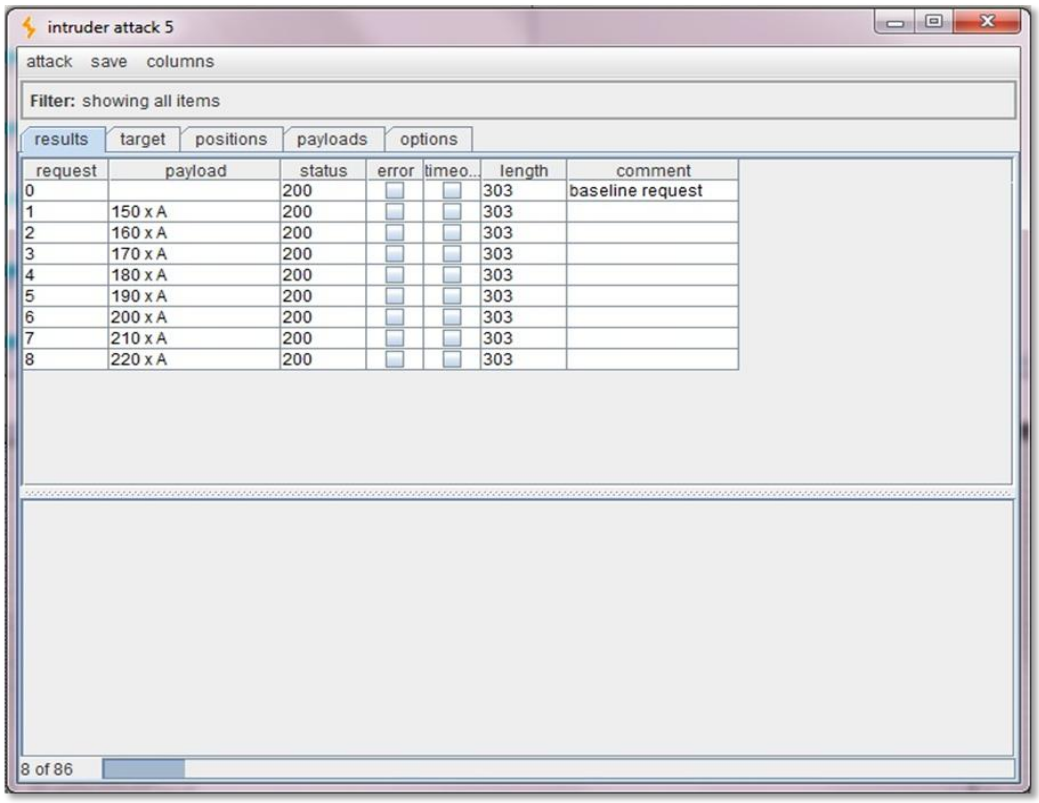
מכיוון ש-Burp Suite לא מגיע עם יכולות ניטור מובנות השתמשתי ב-OllyDbg וחיברתי אותו אל תהליך שרת הצ'אט שכבר רץ ברקע:



זהו, הכל מוכן ונשאר רק להתקיף:

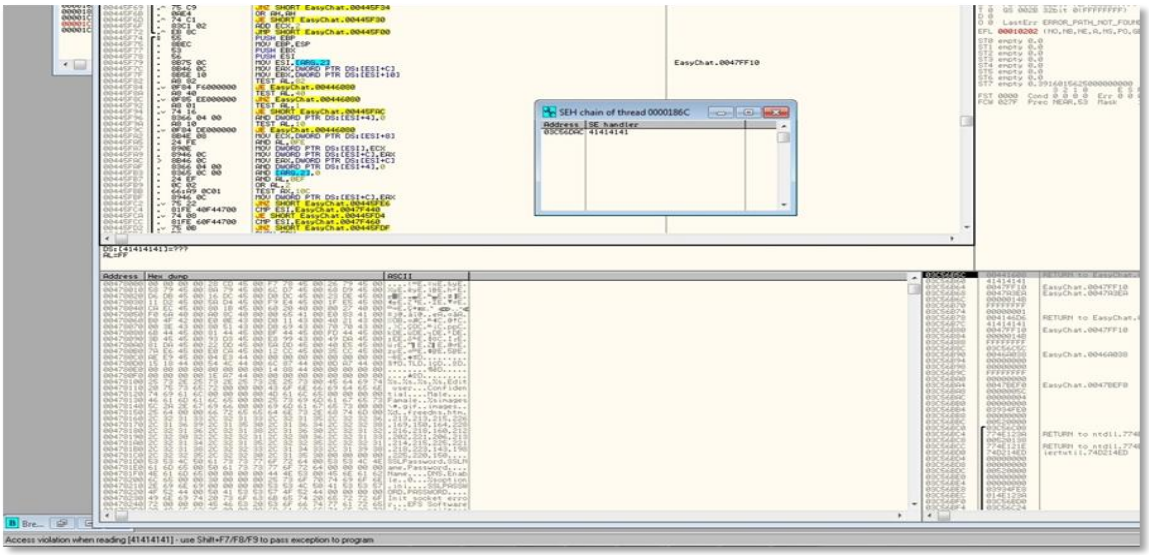


כעבור כמה שניות של ריצה דו"ח הריצה של האינטרודר נתקע על הבקשה השמינית:



request	payload	status	error	timeo...	length	comment
0		200			303	baseline request
1	150 x A	200			303	
2	160 x A	200			303	
3	170 x A	200			303	
4	180 x A	200			303	
5	190 x A	200			303	
6	200 x A	200			303	
7	210 x A	200			303	
8	220 x A	200			303	

גודל המחרוזת 220 כבר היה לי מוכר אז המשכתי כדי לבדוק בדיבאגר (OllyDbg) אם קרתה שגיאה וזה מה שראיתי:



שגיאת גישה לכתובת 0x41414141! בדיקה של ה-SEH Chain מגלה לנו שדרסנו את כתובת ה- Exception Handler במחסנית. מצאנו באג!

למי שלא יצא לקרוא את המדריך הקודם ומעוניין ללמוד יותר על SEH ועל ניצול הבאג הזה בפרט מוזמן להמשיך ולקרוא כאן.

הערת אגב: ניסיתי את אותה התקפה על URI חלקי שלא כולל את פרמטרי ה-room וה-sec, גלישת המחסנית לא קרתה! זה מתקשר ישירות לחלק שמדבר על כיסוי קוד ובדיקת ענפים וזה ניסוי מאוד מעניין ומחנך, בבדיקות קופסא שחורה הבאגים הם בפרטים הקטנים!

סיכום

פאזינג לא בא להחליף ניתוח סטטי של קוד אבל הוא בהחלט פעולה משלימה ונוחה ברוב המקרים. ככלל אצבע שימוש במספר פאזרים מסוג שונה יביא לתוצאות טובות יותר, גם זמן הריצה משפיע בצורה ישירה על הסיכוי למצוא חולשות.

יש סוגים שונים של פאזרים, לכל אחד היתרונות והחולשות שלו, יש לקחת זאת בחשבון כשאתם חושבים לכתוב פאזר משלכם.

חברות גדולות ורציניות מטמיעות היום פאזינג כחלק אינטגרלי במחזור הפיתוח שלהן, חלקן אף מרחיקות לכת ובודקות מוצרי צד שלישי בהן הן משתמשות במוצר שלהן.

כולי תקווה שפרק המבוא, עוזר לקבל מושג ראשוני על תחום מחקר החולשות האוטומטי ופאזינג בפרט.

אם אתם לא תבדקו את הקוד מישהו אחר יעשה את זה! אז...