

דוח מחקר – מטלה תכניתית – קורס 20940

מספר 16, מבוא לאבטחת המרחב המקוון

עידו קלמן (331771535) ואורי סטרוק (338054042)

תוכן עניינים

1.....	מבוא
2.....	מתודולוגיה
5.....	תוצאות
9.....	ניתוח ודיון
15.....	שיקולים אתיים
15.....	מקורות
15.....	נספחים

מבוא

מנגנוני אימות מבסיסי סיסמאות הם שכבות אבטחה מרכזיות ברוב המערכתיות המודרניות, אבל היעילות שלהם תלויה בזרה ישירה באיכות והגדירות האלגוריתמים המשומשים במערכת להצפנה ולגיבוב של הסיסמאות.

בחירות לא נכונות של פרמטרים קריפטוגרפיים או בהגנות מסוימות, כמו Salt, Pepper ואחרים יכולות במידה רבה לחושף את הארגון והמערכת למתקפות, אפילו טיפשות, כמו Brute-Force ו- Spraying, למרות השימוש באלגוריתמים מוכרים.

גיבוב, או באנגלית Hashing הוא תהליך קריפטוגרפי חד ציווני שבו קלט באורך שרירתי, כמו סיסמה, ממופה לפلت באורך קבוע שנקרא ערך גיבוב. אחד המאפיינים החשובים והמרכזיים של גיבוב הוא חוסר יכולת לשחזר את הקלט המקורי מתוך הפלט. מהסיבה הזאת, מערכת מאובטחת אף פעם לא תשמור סיסמה גלויה במסד הנתונים, אלא תמיד תשמור את ערך הגיבוב של הסיסמה במקום. בזמן האימות, נפעיל את אלגוריתם הגיבוב הנבחר על הסיסמה שהתקבלה ע"י המשמש, ונשווה את ערך הגיבוב המתקבל עם הערך השמור.

לעומת זאת, הצפנה היא תהליך דו ציווני שבו מידע מומר לצורה מוצפנת באמצעות **מפתח הצפנה**, ונינתן לשחזרו חוזרת לצורתו המקורי באמצעות מפתחות מתאימים. הצפנה מיועדת לשמירה על סודיות של המידע שנדרש לפענוח עתידי ולכך היא שיטה לא מתאימה לאחסון של סיסמאות, לאחר וחשיפה של מפתח ההצפנה שהייתה בשימוש להצפין את הסיסמאות תאפשר לתקוף לשחזר בזרה מלאה את כל הסיסמאות.

אחד השיטות המוכרות לתקיפה של מערכות מבוססות סיסמה היא **"חיפוש מממצה"**. בשיטה הזאת התוקף מנסה באופן שיטתי מספר רב של סיסמאות אפשרית עבור חשבון מסוים, עד להשגת התאמה. היעילות של ההתקפה תלויות בכמה גורמים –

- גודל מרחב החיפוש** (למשל – אם במערכת מסוימת סיסמה היא רצף של מספרים באורך של לפחות 3 ספרות, מרחב החיפוש הוא לפחות $1000 = 10^3$ סיסמאות).
- מהירות ביצוע ניסיונות האימות** (מהירות זו היא כמובן תוצאה של זמן התגובה בין התוקף והשרת, מהירות העיבוד והבדיקה של השרת, וכלי).
- קיומן של הגנות** – כמו שנראה בהמשך, הגנות שהשרות מפעיל כמו הגבלה של קצב הניסיונות, נעילת חשבון ומנגנוני TOTP ו-CAPTCHA יכולים למנוע מהתוקף להצלחת.

שיטה נוספת לתקיפה של מערכות כאלו נקראות Password-Spraying, והיא וריאציה מתוחכמת יותר של Brute-Force, שבה נעשה ניסיון להשתמש במספר מוגבל של סיסמות, על פני מספר רב של חשבונות שונים. השיטה זו נועדה לעקוף מנגנוןים שמוממשים על ידי השירות כדי לנטר ולמנוע פריצה לחשבון היחיד.

המטרה של העבודה שלנו היא לבצע סדרה של ניסויים שיבחנו את ההשפעה של אלגוריתמי גיבוב שונים ומנגנון הגנה מסוימים על העמידות של מערכות אימוטס מבוססות סיסמות, ולהעריך את היעילות של המנגנון והאלגוריתמים האלו להתמודד עם שני סוגי התקפות שתיארנו עד כה בסביבת הניסוי.

אלגוריתמי הגיבוב שנבדוק בדוח זה:

.1 **SHA-256** – אלגוריתם גיבוב קריפטוגרפי מהמשפחה SHA-2, המפיק פלט באורך קבוע של 256 bites. האלגוריתם תוכנן להיות מהיר ויעיל חישובית, תכונה שהופכת אותו לאלגוריתם שנמצא בשימוש נרחב. למרות זאת, דוקא המהירות הגבוהה שלו יכולה להפוך לחיסרונו מאחר והתוקף יכול לבצע כמוות גבוהה של ניסיונות גיבוב בשנייה. לכן שימוש באלגוריתם זה נחשב בטוח רק כשהמשלבים אותו עם salt.

.2 **Bcrypt** עם הפרמטר cost = 12 – אלגוריתם גיבוב נוסף לאחסון סיסמות שmbוסס על צופן سنקרוא Blowfish, עם מנגנון האטה מובנה. הפרמטר cost קובע את מספר סבבי החישוב האקספוננציאליים שמתבצעים, כך שככל עלייה של יחידה בערך זה מכפילה את זמן החישוב.

.3 **Argon2id** עם הפרמטרים time = 1, memory = 64MB, parallelism = 1 – האלגוריתם האחרון שנבחן הוא אלגוריתם מורכב יותר שמשלב הגנה ממתקפות זמןון והתקפות שבוססות GPU. הפרמטר time מגדיר את מספר איטרציות החישוב, memory קובע את כמות הזיכרון שנדרשת לכל פעולה גיבוב, ו- parallelism קובע את מספר התהליכיונים שפועלים במקביל.

מנגוני אבטחה שנבדקו בדוח זה:

.1 **Pepper – Pepper** – הוא ערך סודי נוסף ששולב בתהליך גיבוב הסיסמה, בדומה ל-Salt, אך בניגוד אליו הוא לא נשמר במאזן הנתונים אלא במקום מסוון פרט, למשל, משתנה סביבה. ה- Pepper משותף לכל המשתמשים במערכת ומתווסף לסיסמה לפני פעולה הגיבוב. מטרתו להקשות על התוקף גם במקרה של דליפת מסד הנתונים עצמו. ביל ה- Pepper קשה לבצע מתקפות Bruteforce או להשתמש ב- Rainbow tables על ערכי הגיבוב שנחשפו. כMOVN שחשיפה של ה- Pepper עצמה פוגעת ביעילות ההגנה ולבן האבטחה שלו חשובה מאוד.

.2 **Rate Limiting** – מנגנון הגנה מצד השירות שمبرיל את מספר ניסיונות האימוט שניתן לבצע בפרק זמן נתון, לפי כתובת IP, חשבון משתמש, או שילוב של שניהם. לאחר חריגה מהסף שהוגדר, השירות יכול להשוחות תגבות, לחסום זמני ניסיונות נוספים או לדורש אימוט נוסף כמו CAPTCHA. המנגנון הזה מצמצם שימושותיו את האפקטיביות של מתקפות Bruteforce ו- Password Spraying בכלל החסימות וזמן ההשחיה הארוך.

.3 **TOTP – Time-based One-Time Password** – מנגנון אימוט דו-שלבי שmbוסס על קוד-חד פעמי שימוש כל פרק זמן, לרוב כל 30 שניות. הקוד משותף באמצעות סוד משותף בין השירות למישיר המשמש ובשילוב הזמן הנוכחי. גם אם התוקף מצליח להציג את סיסמת המשתמש הוא לא יוכל להשלים את תהליך ההתחברות ללא הקוד הזמן, ולכן TOTP הוא אמצעי אפקטיבי מאד כדי למנוע את המתקפות שנבעו בדוח זה.

מתודולוגיה

מבנה כללי של המערכת

הניסוי בוצע באמצעות שפת Python, שנועדה לצורך סימולציה של מתקפת הזדהות ומורכבת משלושה רכיבים עיקריים:

- .1. שירות אימוט
- .2. ליקוי תוקף

3. מנגנון הרצת ניסויים אוטומטי

השילוב של כל הרכיבים מאפשר לנו לשולט בצורה מלאה בפרמטרי ההגנה לצד השירות, להריץ בצורה אוטומטית סט של מתקפות שונות לצד הלקווח, ולאסוף את המדדים הcompanions לצורך ניתוח והשוואה בין תרחישים בשלב מאוחר יותר.

שרת האימוט

השירות מבוסס על Flask ומספק ממשק HTTP בסיסי לאיומות משתמשים. הוא מדמה מערכת היזדהות ריאלית שכוללת בתוכה גם אפשרות למנגנון אבטחה נפוץ.

בעת רישום משתמש, השירות שומר את פרטי המשתמש למסד נתונים מקומי מבוסס SQL. כמו כן בחישום האימייטיים, סיסמת המשתמש נשמרת בצורה הגלולה, אלא רק לאחר הפעלת אלגוריתם הגיבוב המוגדר עלייה. בהתאם לצורכי השירות בעת הרצת התקיפה, ניתן להפעיל Pepper גלובלי – ערך סודי שנשמר לצד השירות ומשולב בתהליך הגיבוב כדי להקשות על תוקפים.

נקודות קצה שמוגדרות בשרת:

רישום משתמש – POST /register

נקודות קצה זו מאפשרת יצירת חשבון חדש. הבקשה כוללת שם משתמש וסיסמה, ובמידת הצורך גם הגדרת TOTP. במהלך הרישום מתבצעת הצפנה של הסיסמה בהתאם לצורכי השירות הנוכחי.

השימוש שלנו בנקודות קצה זו בניסוי היא יצירת משתמשים בצורה סיננטית בצורה אוטומטית לפני הרצת המתקפות.

כניסה – POST /login

נקודות הקצה המרכזיות בניסוי. מטפלת בניסיון התחרבות באמצעות שם משתמש וסיסמה, ומישמת את כל מנגנוני ההגנה שהוגדרו בתצורת האבטחה של השירות: בדיקת Rate limiting לפי חלון זמן, בדיקת נעלית חשבון לאחר מספר כשלונות, דרישת CAPTCHA לאחר מספר ניסיונות כושלים, דרישת אimoto דו-שלבי, כموן – במידה ומופעל.

התגובה משתנה בהתאם למצבי המשתמש ומדיניות האבטחה.

כניסה דו-שלבי – POST /login_totp

נקודות קצה זו מופעלת לאחר Girişות סיסמה מוצלח כאשר TOTP פועל. היא מקבלת קוד חד פעמי ומבצעת Girişות מול הסוד ששומר במסד הנתונים עבור אותו משתמש. היא כמובן בשימוש רק כאשר TZורת השירות מאפשרת TOTP.

CREATE – GET /admin/captcha_token

נקודות קצה זו משמשת ליצור טוקן CAPTCHA. על מנת לקבל טוקן כזה יש להעביר ערך סודי, במקורה שלנו הוא ה- Group seed, כך שרק אדםין יוכל להשתמש בנקודה כדי לקבל טוקן אמיתי.

בניסוי נשתמש בנקודה זו על מנת ליצור טוקן מבלי להפסיק את התקיפה, כאשר השירות דרש זאת במידה ואפשרות ה-CAPTCHA דולקה.

ניהול TZורת אבטחה – POST /admin/config

גם נקודת קצה זו היא נקודת שימושה שמיועדת לשימוש האוטומציה, והיא מאפשרת לשנות בצורה דינמית את הגדרות האבטחה של השירות השמורים בזיכרון התהילך. ניתן לשנות כל אחד מהפרמטרים הבאים:

```
{
  "bcrypt_cost": 12,
  "captcha_afterfails": 5,
  "captcha_enabled": false,
```

```
"hash_mode": "sha256",
"lockout_enabled": false,
"lockout_threshold": 10,
"lockout_time": 300,
"pepper": "",
"pepper_enabled": false,
"rate_limit_enabled": false,
"rate_limit_max": 30,
"rate_limit_window": 60,
"totp_enabled": false
}
```

שימוש בנקודה הזאת מאפשר לאוטומציה שלנו להריץ ניסויים ברצף, בזמן שמשנים בצורה מבוקרת משתנה אחד בכל פעם כדי לבחון את ההשפעה שלו על התוצאות.

- – שלייפת תצורה נוכחית: GET /admin/config

נקודות קצה זו מאפשר קרואת תצורה האבטחה הנוכחית של הרשות. היא משמשת אותו כדי לוודא שהתצורה מוגדרת כראוי לפני כל ניסוי ולשמריה של התצורה כחלק מתייעוד התוצאות.

יצירת סיסמות

לצורך הניסוי נוצרו מראש שלושה מאגרי סיסמאות ברמות קושי שונות:

- .1 סיסמאות קלות
 - .2 סיסמאות בינוניות
 - .3 סיסמאות קשות

מאגרי היסמאות שלו נוצרו באמצעות סקירה שיכרנו לשם כך, ונשמרו כקבצים סטטיים. סיסמאות אלו משמשות לרשום המשתמשים בשרת.

בצד התוקף, למשל במתකפות Bruteforce, נעשה שימוש במחולל סיסמאות אחר, שאיננו מזמן ממאגר הסיסמאות (בצורה הזאת המתקפה איננה הייתה Bruteforce מאחר והתוקף היה חשוב לסיסמאות). מחולל הסיסמאות הזה סורק באופן שיטתי את מרחב הפתרונות האפשרי, והוא לא תלוי במאגרים הסטטיים.

חוקים ליצירת הסיסמות:

סיסמה קלה – רצף בין 4 ל-6 תווים הלקוחים מຕוך הקבוצה:

$$\{a, b, c, d, 0, 1, 2, 3\}$$

סיסמה בינונית – רצף בין 7 ל-8 תווים הלקוחים מתוך הסט האלפאנומי: אותיות קטנות וגדלות באנגלית והספרות 0 עד 9.

סימנה חזקה - רצף בין 8 ל-10 תווים הלקוחים מຕוך הסט האלפאנומי: אותיות קטנות וגדלות באנגלית והספרות 0 עד 9.

כל אחד מהרציפים נבחר בצוותה אחידת, בהירה ונדומלית, אחריתו מתוך הקבוצה הרלוונטית.

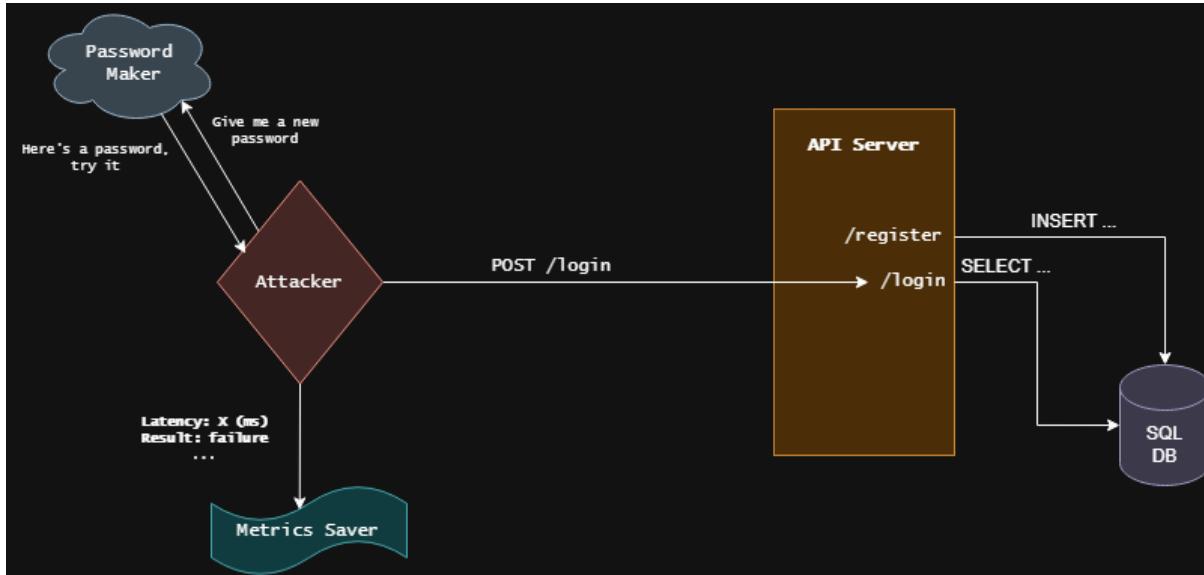
הרצת בדיקות בצורה אוטומטית

הבדיקות שנעשו בנייסוי בצוות אוטומטיות כדי להבטיח עקביות ויכולת לחזור את התוצאות בעלי הטעברות של בני אדם. האוטומציה מאפשרת לנו לחזור על הניסויים תחת תנאים זהים

מערכת הבדיקה מופעלת באמצעות סקורייפט מרכזי אשר:

- .1. מתחילה את סביבת הניסוי ע"פ דרישות הבדיקה
- .2. מפעיל את כלי הבדיקה (Attacker client)
- .3. אוסף מדדים בזמן אמת ומפיק דוח תוצאות

תרשים סיכום :

**דיון בתוקף הניסוי**

התוקף בניסוי מוגבל מכמה סיבות וגורמים טכניים. קודם כל, גודל המדגם היה מצומצם וככל מספר קטע של משתמשים וסימאות שלא בהכרח ריאליים למציאות האנושית. חומרת הניסוי וההיקף שלו יהיה מוגבל – הניסוי בוצע בפרק זמן קצר, עם עומס נמוך, ובסביבה בדיקות מבוקרת שלא מייצגת מערכת פרודקشن עמוסה ואמיתית שחייה תחת תנאי רשות מציאותיים שבהם קיימים עומס שנוצר כתוצאה משימוש טבעי בשרת האימוט על ידי משתמשים לגיטימיים.

תוצאות**ניסוי 1 : תקיפת Bruteforce, ללא הגנות, sha256, משתמש #1 –**

- בוצעו 35,919 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן הריצה הכללי היה 14.5 דקות, כאשר בוצעו בערך 41 ניסיונות בשניה בממוצע.
- זמן התגובה הממוצע של השרת היה 13.1 מילישניות, ונע בין 0 מ"ש ל- 2087 מ"ש.
- השימוש הממוצע במעבד היה 9.55%, ושימוש הזיכרון הממוצע היה בערך 35 מגה-בייט.
- סטטוס : הצלחה

ניסוי 2 : תקיפת Bruteforce, ללא הגנות, sha256, משתמש #2 –

- בוצעו 24,931 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן הריצה הכללי היה כ-12 דקות, כאשר בוצעו בערך 33 ניסיונות בשניה בממוצע.
- זמן התגובה הממוצע של השרת היה 18.67 מילישניות, ונע בין 0 מ"ש ל- 18.6 מ"ש.
- השימוש הממוצע במעבד היה 8%, ושימוש הזיכרון הממוצע היה בערך 33 מגה-בייט.
- סטטוס : הצלחה

ניסוי 3 : תקיפת Bruteforce, ללא הגנות, sha256, משתמש #3 –

- ביצעו 225,88 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן היריצה הכוללת כשעה ו-53 דקות, כאשר ביצעו ערך 33 ניסיונות בשנייה בממוצע.
- זמן התגובה הממוצע של השרת היה 18.96 מילישניות, ונע בין 0 מ"ש ל- 2821 מ"ש.
- השימוש הממוצע במעבד היה 6.69%, ושימוש הזיכרון הממוצע היה בערך 26 מגה-בייט.
- סטטוס : הצלחה

ניסוי 4 : תקיפת Bruteforce, ללא הגנות, argon2id – משתמשים #1 ו-#2משתמש #1 :

- ביצעו 35,919 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן היריצה הכוללת כ-48 דקות, כאשר ביצעו ערך 12 ניסיונות בשנייה בממוצע.
- זמן התגובה הממוצע של השרת היה 70.27 מילישניות, ונע בין 44 מ"ש ל- 2178 מ"ש.
- השימוש הממוצע במעבד היה 2%, ושימוש הזיכרון הממוצע היה בערך 33 מגה-בייט.
- סטטוס : הצלחה

משתמש #2 :

- ביצעו 35,919 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן היריצה הכוללת כ-33 דקות, כאשר ביצעו ערך 12 ניסיונות בשנייה בממוצע.
- זמן התגובה הממוצע של השרת היה 70.05 מילישניות, ונע בין 0 מ"ש ל- 1429 מ"ש.
- השימוש הממוצע במעבד היה 2%, ושימוש הזיכרון הממוצע היה בערך 32 מגה-בייט.
- סטטוס : הצלחה

ניסוי 5 : תקיפת Bruteforce, ללא הגנות, bcrypt –משתמש #1 :

- ביצעו 35,919 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן היריצה הכוללת כשעתיים ועשרים דקות, כאשר ביצעו ערך 4.2 ניסיונות בשנייה בממוצע.
- זמן התגובה הממוצע של השרת היה 221.55 מילישניות, ונע בין 188 מ"ש ל- 2058 מ"ש.
- השימוש הממוצע במעבד היה 0.75%, ושימוש הזיכרון הממוצע היה בערך 24 מגה-בייט.
- סטטוס : הצלחה

משתמש #2 :

- ביצעו 24,931 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן היריצה הכוללת כשעה ו- 36 דקות, כאשר ביצעו ערך 4.3 ניסיונות בשנייה בממוצע.
- זמן התגובה הממוצע של השרת היה 220 מילישניות, ונע בין 186 מ"ש ל- 2003 מ"ש.
- השימוש הממוצע במעבד היה 0.76%, ושימוש הזיכרון הממוצע היה בערך 22 מגה-בייט.
- סטטוס : הצלחה

ניסוי 6 : תקיפת Bruteforce, sha256, עם Pepper – משתמש #1 –

- ביצעו 35,919 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן היריצה הכוללת כ-15 דקות, כאשר ביצעו ערך 39 ניסיונות בשנייה בממוצע.
- זמן התגובה הממוצע של השרת היה 14.3 מילישניות, ונע בין 2 מ"ש ל- 1827 מ"ש.
- השימוש הממוצע במעבד היה 6.95%, ושימוש הזיכרון הממוצע היה בערך 33 מגה-בייט.
- סטטוס : הצלחה

ניסוי 7 : תקיפת Bruteforce, sha256, עם TOTP – משתמש #1 –

- ביצעו 35,919 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן היריצה הכוללת כ-16 דקות, כאשר ביצעו ערך 37 ניסיונות בשנייה בממוצע.
- זמן התגובה הממוצע של השרת היה 15.3 מילישניות, ונע בין 0 מ"ש ל- 1307 מ"ש.

- השימוש הממוצע במעבד היה 7%, ושימוש הזיכרון הממוצע היה בערך 33 מגה-בייט.
- סטוס : **כישלון בתוצאה מ-TOTP דлок**

ניסוי 8: תקיפת Bruteforce, sha256, עם #2 – משתמש #2

- בוצעו 24,931 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן הריצה הכלול היה כ-49 דקות, כאשר בוצעו בערך 8 ניסיונות בשנייה בממוצע.
- זמן התגובה הממוצע של השרת היה 16.07 מילישניות, ונע בין 1 מ"ש ל- 1254 מ"ש.
- השימוש הממוצע במעבד היה 5.14%, ושימוש הזיכרון הממוצע היה בערך 33 מגה-בייט.
- סטוס : **הצלחה**

ניסוי 9: תקיפת Bruteforce, sha256, עם CAPTCHA

משתמש #1

- בוצעו 35,919 ניסיונות בסך הכל עד למציאת הסיסמה
- זמן הריצה הכלול היה כ-16 דקות ו-24 שניות, כאשר בוצעו בערך 36 ניסיונות בשנייה בממוצע.
- זמן התגובה הממוצע של השרת היה 16.17 מילישניות, ונע בין 0 מ"ש ל- 2345 מ"ש.
- השימוש הממוצע במעבד היה 6.93%, ושימוש הזיכרון הממוצע היה בערך 34 מגה-בייט.
- סטוס : **הצלחה**

ניסוי 10: תקיפת Bruteforce, sha256, עם Account Lockout

במהלך ניסוי זה, מנגנון ה- Account lockout הוגדר לחסום חיבור לחשבון אחרי 1000 ניסיונות כושלים, על אף שזו כנראה הגדרה מתירנית יותר בהשוואה לכזו שתהיה בסביבה של שירות אימות אמיתי.

משתמש #1 : תוצאה : החשבון נחסם

אחרי 1000 ניסיונות, התוקף לא הצליח לנחש את הסיסמה

אחרי 1000 ניסיונות, התוקף לא הצליח לנחש את הסיסמה

משתמש #2 : תוצאה : החשבון נחסם

אחרי 1000 ניסיונות, התוקף לא הצליח לנחש את הסיסמה

בשימוש בהגנה זו, התוקף לא הצליח לפרוץ את הסיסמה של אף אחד ממחשבונות

ניסוי 11: תקיפת Bruteforce, sha256, ללא הגנות – משתמש עם סיסמה בחזקת "ビノニティ"

משך התקיפה עבר את 7200 השניות שהן שעתיים, הרף העליון, ללא הצלחה בפיצוח סיסמה בינוונית.

עד לסיום שעתיים אלו, בוצעו 191,223 ניסיונות. לעומת זאת, בוצעו כ-26 ניסיונות בשנייה. שימוש המעבד הממוצע היה כ-6 אחוז ושימוש הזיכרון הממוצע היה כ-35 אחוז. הסיסמה الأخيرة שנמצאה הייתה `Yj.aaaaaL82Kb7M`.

לאחר שביצענו אקסטרපולציה על פי נתונים אלו, בಗלל סיסמאות בינוניות וחזקות לקוחות מtower סט אלפבית אלפאנומי עם 62 תוויות (אותיות קטנות וגדלות באנגלית ומספרים), והסיסמה שאיתה ניסינו לפצח הייתה $2.15 * 10^{12}$ ניסיונות שהיו לוקחים לנו $8.3 * 10^{10}$ שנים, שהם בערך 2,600 שנים, מספר לא ריאלי עבור תוקף.

ניסוי 12: תקיפת Bruteforce, sha256, ללא הגנות – משתמש עם סיסמה בחזקת "חזקת" – משך התקיפה עבר את 7200 שניות שהן שעתיים, הרף העליון, ללא הצלחה בפיקוח סיסמה בגיןית.

עד לסיום שעתיים אלו, בוצעו 191,223 נסיעונות. ככלומר בוצעו כ-26 נסיעונות בשנייה. שימוש המעבד המומוצע היה כ-6 אחוז ושימוש הזיכרון המומוצע היה כ-35 אחוז. הסיסמה האחורונה שמוצצת הייתה ".aaaaaaXUO".

לאחר שביצענו אקסטרופולציה על פי נתונים אלו, בגלל שישנם מאות בגיןיות וחזקות לckoות מתוך סט אלפבית אלפאנומי עם 62 תוויות (אותיות קטנות וגדלות באנגלית ומספרים), והסיסמה שאותה ניסינו לפצח הייתה Zwi1GNKo73, היינו צריכים לבצע עוד $7.09 * 10^{17}$ נסיעונות שהיו לורחים לנו $8.3 * 10^8$ שנים, שהם בערך 865 מיליון שנים, מספר מאוד לא ריאלי עבור תוקף שבו ניתן להזכיר שבנסיבות גבואה, החשבון חסין מפני מתקפות Bruteforce.

ניסוי 13: תקיפת Password Spray, ללא הגנות – על גבי כל משתמשי המערכת –

בניסוי זה, שהוא הפעם הראשונה שבו נסקור מתקפת Password Spray, השתמש ברשימה גדולה של סיסמאות שכוללת את חמשת אלף הסיסמאות הכי נפוצות למשתמשים בראשת.

בנוסף, כל המתקפות שנריצ' בדוח' זה מהסוג הנ'"יל יובצעו על גבי כל המשתמשים במערכת, ככלומר: סיסמאות קלות, בגיןיות וקשות. ההחלטה הזאת נובעת מהעובדת שהרשימה כוללת אחד סיסמאות קצורות וארוכות, וכך אפשר לבדוק את סיכוי הצלחה של התקיפה על גבי כל המשתמשים.

תקיפה מסוג Password spray נראית בצד התוקף בצורה הבאה –

Testing password 'password' (1/.....)

Testing password '123456' (2/.....)

סיכום התוצאות:

- בוצעו 142,861 נסיעונות התחברות
- משך זמן התקיפה הכלול היה 1 שעות ו-8 דקות
- במהלך התקיפה נפרצו 4 חשבונות שונים

ניסוי 14: תקיפת Lockout, הגנת Password Spray – על גבי כל משתמשי המערכת –

אחד ההגנות שראינו בשלבים קודמים יותר בניסוי הינה הגנת Lockout, Account Lockout, שנעילה את הכניסה לחשבון אחריו מספר מסויים של נסיעונות כושלים להתחברות לחשבון.

בניסוי זה נראה כי הגנה זו היא אפקטיבית גם במקרה למתתקפת Password Spray ולא רק ביחס ל-Bruteforce. תוצאות הניסוי מראות כי על אף שתקיפה מבוססת Password Spray ללא הגנת הצלחה לפחות 4 מהמשתמשים שהשתמשו בסיסמאות נפוצות, מאוחר ונדרשו לכך יותר מכמות הניסיונות שהוגדרו בתצורת ההגנה בשרת 10000 נסיעונות עד לחסימת החשבון, ככל שהזמן עבר לאורך התקיפה, כמוות החשבונות שניתנו היה לנסוט את הסיסמאות עליהם הלק' קטן – מאוחר וכל חשבון שנוסע עליו יותר מ-1000 סיסמאות נחסם.

ככלומר, החל משלב מסוים על אף העובדה שנשארו סיסמאות נפוצות בראשיה, כל החשבונות היו חסומים והפיקוח נכשל.

תוצאה: 0 מתוך 30 חשבונות פוצחו.

ניסוי 15: תקיפה, Password Spray – על גבי כל משתמשי המערכת –

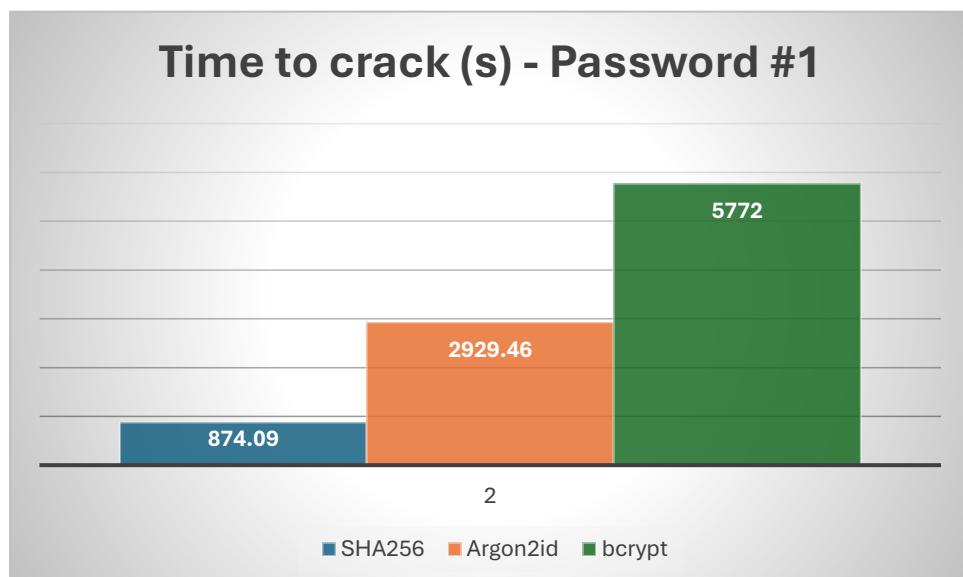
בעת שילוב הגנתה – Rate limiting, משך התקיפה התארך מאחר ונדרשים לבצע הפסוקות יזומות בעת התקיפה על משתמשים שעברו את ההגבלה לכמויות ניסיונות ההתחברות שאפשר לבצע עליהם בפרק הזמן הנדרש. בຕוצאה מכך, משך זמן הניסוי עבר את הרף העליון של שעתיים, כשבפרק זמן זה 3 סיסמאות פוצחו, במקומות 4 בתצורה הרגילה

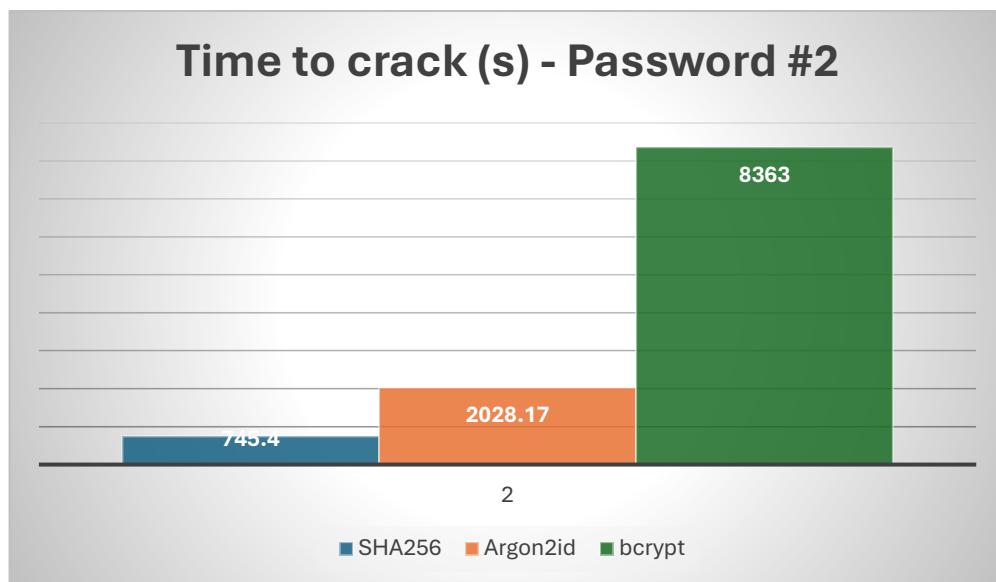
ניתוח ודיון

1. השוואת היעילות והעמידות של מנגןוני גיבוב שונים:

בחולק זה של הדיון נרצה להשוות ולהגיע למסקנות לגבים ליעילותם ולחסינותו של אלגוריתמי גיבוב השונים שבדקנו : SHA256, argon2id, bcrypt . בגרפים הבאים השווינו את הזמן בשניות של שתקוף Bruteforce לפצח את סיסמת המשתמש, על גבי שני משתמשים שונים, כאשר השתמשנו בתצורת השרת הבסיסית ללא הגנות נוספות, ושינינו בכל פעם רק את מנגןון הגיבוב של הסיסמאות :

עבור משתמש #1



עבור משתמש #2 :

התוצאות מצביעות על המסקנה הבאה – על אף שהאלגוריתם SHA256 מהיר ופשוט, הוא האלגוריתם הכי פגוע מהשליטה למתקפת Brute force. האלגוריתם argon2id במקומות השני – איטי יותר ועמיד יותר, והאלגוריתם bcrypt איטי עוד יותר ובעל העמידות הגבוהה ביותר למתקפת-h-Bruteforce מהשליטה שבחנו בניסוי.

יש לציין שבוד שניתן לצפות מראש שהאלגוריתמים argon2id ו- bcrypt שניהם יהיו איטיים יותר מ- SHA256, התחרויות ביןיהם תלויות בפרמטרים שאיתם רצים. בכלל שבחרכנו באופן דיפולטיבי cost factor של 12 ל-.bcrypt, יש $2^{12} = 4096$ איטרציות במהלך החישוב שמאטאות את החישוב בצורה משמעותית, כאשר הפרמטרים שהוגדרו ל-.argon2id הם פרמטרים זולים יותר. וכן גם ייתכן שתוצאות שונות יהפכו את argon2id לאיטי יותר מ-.bcrypt.

המלצות לפועלה:

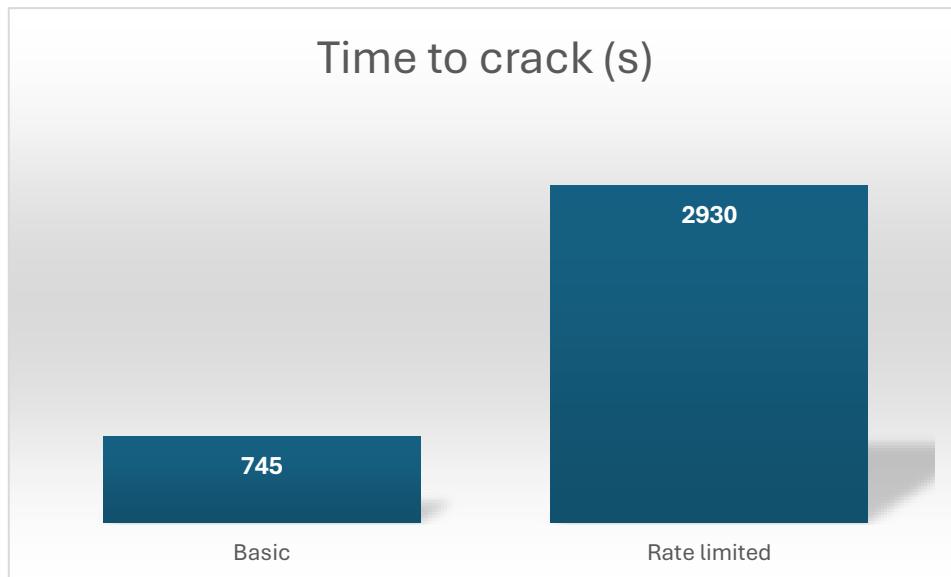
על מנת להגן על שרת האימוט בצורה המיטבית נציע להשתמש באלגוריתמי הגיבוב הבאים, על פי סדר יורד של רמת האבטחה שהם מציעים :

Bcrypt	-
Argon2id	-
SHA256	-

2. שימוש במנגנון הגנה והשפעתם על יכולת הפיצוח:

מנגנון Rate Limiting

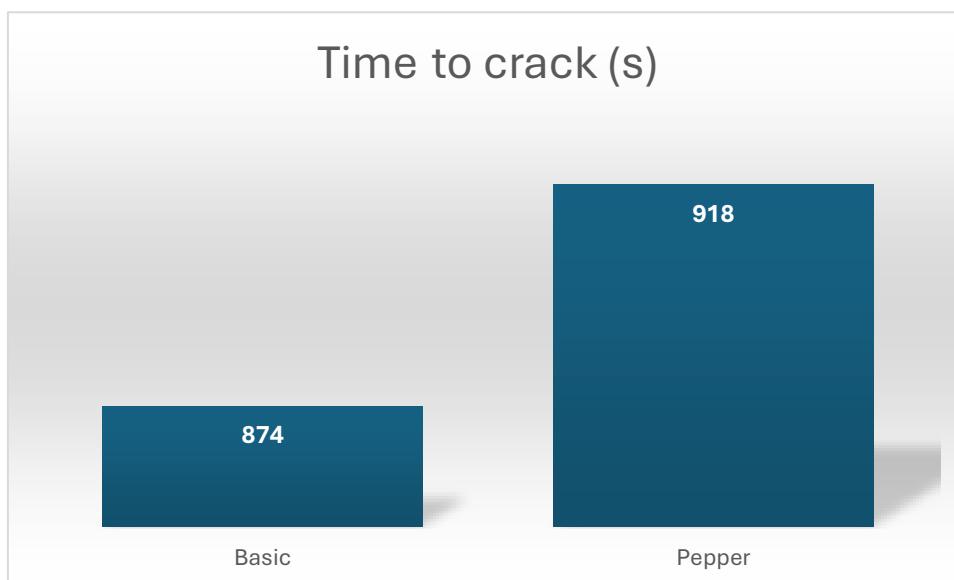
בגרף הבא ניתן לראות את ההשוואה בין 2 רמות הגנה של השרת במתכנת Brute force : הראשונה – ללא הגנות נוספות, והשנייה – כאשר הפעילו את מנגנון ה- Rate limiting עם ההגדרה (המקלה יחסית) של עד ל-500 בקשות בכל 60 שניות. לעומת זאת, בין כל 60 שניות התוקף יכול לבצע עד ל-500 בקשות התחברות.



ניתן לראות כיצד הפעלת המנגנון הגדיל את משך הזמן הדרוש על מנת לפצח את הסיסמה **פי כמעט 4**, מה שהופך אותו למנגנון הגנה מאוד אפקטיבי בצד השרת.

שילוב ערך Pepper

בגרף הבא ניתן לראות את ההשוואה בין רמת ההגנה הבסיסית של השרת, לבין שילוב של ערך Pepper סודי בזמן הגיבוב :



ההבדל הznich בין זמני הריצה (874 שניות ללא ההגנה מול 918 שניות עם שילוב ערך Pepper) נובע מכח השוספת-h-Pepper מוסיפה שלב חישובי נוסף לכל ניסיון, ולכן למורות שהפעולה עצמה יחסית פשוטה, כשהיא מתבצעת בכל ניסיון סיסמה ומצטברת לאלפי ניסיונות, היא מצטברת לעלות של כמה עשרות שניות.

עם זאת - התועלת האמיתית של Pepper מתחבطة בעיקר במתקפות Offline, שלא כיסינו בניסוי זה. במקרה כזה, התוקף מחזיק את מאגר הנתונים המוגוב (למשל, לאחר שהוא הודף משרת האימוט), אבל הוא לא מחזיק בערך-h-Pepper שמאוחסן בServerError, למשל במשתנה שביבה בתוך השירות. בלי הערך הזה, כל ניסיון פיצוח ייכשל גם אם ניחוש הסיסמה אכן נכון, וכך התוקף מחוויב לא רק לנחש את הסיסמה עצמה, אלא גם את הערך של ה-*Pepper*, ובכך מרחיב החיפוש גדל משמעותית והעלות החישובית של ההתקפה קטנה מאוד.

הפעלה של Account Lockouts

ההגנה הבאה שנבחנו היא הגנה שמבצעת נעליה של החשבונות כאשר התוקף עבר את סף הניסיונות השגויים שאפשריים לכינסה לחשבון. על פי ההנחה, כאשר הרשת מבצע נעליה של החשבון, התוקף יעצור את הניסיונות לפריצה לחשבון. תרחיש כזה במציאות הוא סביר, למשל במקרים שבהם כאשר החשבון נחסם (כשמנסים יותר מדי סיסמאות שגויות) עד שבעל החשבון האמתי יאמת את זהותו, בדרך המילול המקשר לחשבון.

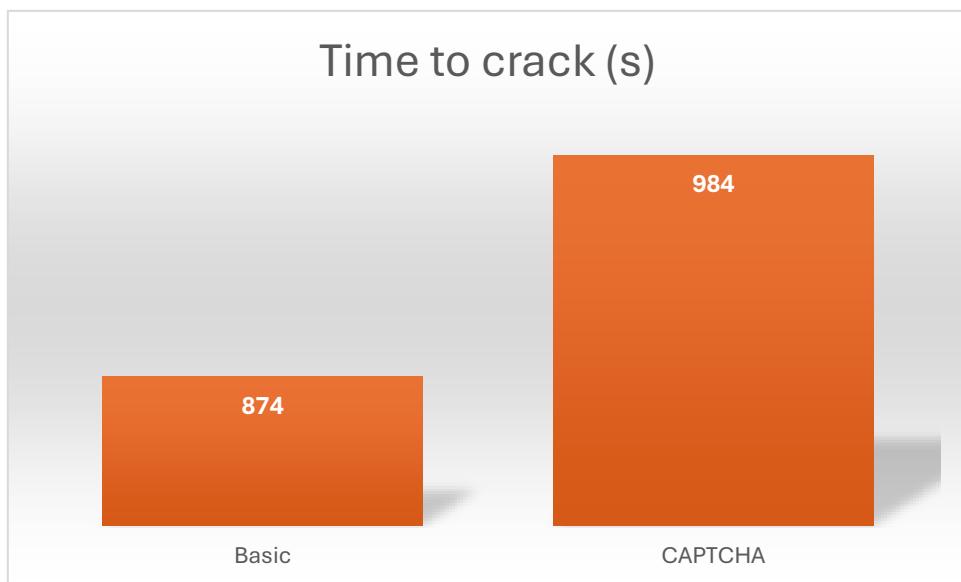
כתוואה מכך, הגנה זו הופכת לחזקה מאוד נגד מתקפות Bruteforce: **במהלך הניסוי לא הצלחנו לפצח אף סיסמה מהמאגר כאשר הרשות פעל עם הגדרה זו עם מתקפה מהסוג הנ"ל**

הפעלה של מנגנון CAPTCHA

נזכיר שכמוצופה מאייתנו בניסוי, לאחר ותווך אמתי יכול במקרים מסוימים לעبور אתגרי CAPTCHA גם במציאות, לאחר X ניסיונות כשהשרת מצפה מאייתנו לעبور אתגר CAPTCHA, אנחנו משתמשים ב-Endpoint בשירות שמיועד לשימוש שלנו בלבד, שמאפשר לקבל טוקן תיקון כזה. לאחר X ניסיונות נוספים, נדרש לקבל אחד נוספת.

לכן אנחנו מצפים שזמן הפיצוח יהיה ארוך יותר כתוצאה מתוקורה בבקשת הרשות שמייעדות מעבר לאתגר CAPTCHA.

השוינו בגרף הבא בין ריצה ללא הגנות, וריצה מבוססת CAPTCHA על אותו משתמש:



עיר, שעל אף שההבדל בזמן הפיצוח איננו ענק, הגדרות ניסוי זה דרשו אתגר CAPTCHA **כל 20 ניסיונות**. ככל שמספר זה ייקטן, תוקף פוטנציאלי יctrיך לעבר יותר אתגרים לעיתים יותר תכופות. בנוסף לכך, בנסיבות אפשר妾שההבדל יהיה גדול יותר ולאחר הזמן שולוקח לעבר אתגר מהצורה שמופיע בתמונה למיטה ארוך יותר מעוקף בקשה ה- HTTPS שהtopic ביצע בניסוי שלנו –

Match the characters in the picture Help

To continue, type the characters you see in the picture. [Why?](#)

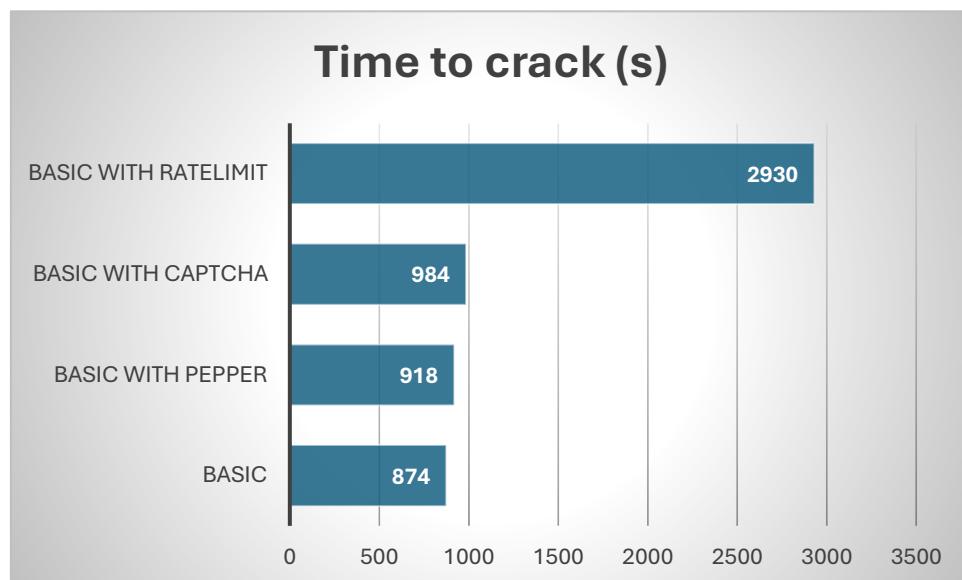


The picture contains 8 characters.

Characters:

Continue

לסיכום ההגנות "המאtot" עד כה, ניתן לבצע השוואה בגרף הבא :



המלצות:

מהניסיונים שלנו עולה כי סדר העדיפויות בעת בחירת ההגנות שכוסו עד כה בשרת, בנוסף להגנות שיכולה למנוע לחולוטין את ההצלחה של מתקפות Bruteforce, Account Lockout ו-hgnet TOTP :

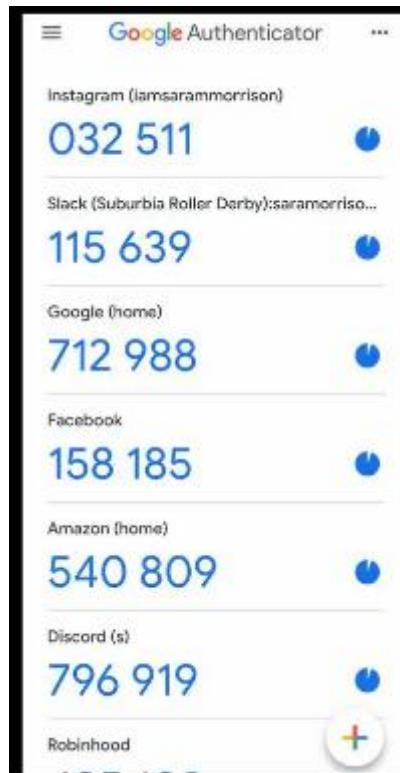
1. הגנת Rate limit, בשאייה למספר ניסיונות קטן ככל האפשר עד להפעלת הגבלה.
2. הגנת CAPTCHA

3. שילוב של מנגנון Pepper בעת הגיבוב ב景德ן הנתונים

הפעלה של מנגנון TOTP

TOTP הוא מנגנון אימות דו שלבי, כמו שציינו בדוח בחלקים מוקדמים יותר. אופן המימוש שלו פועל כך:

1. בעת הרישום, המשמש מעביר לשרת סוד, שמוחזק בעת אצל המשתמש ואצל השרת.
2. הסיסמה החדש פעמיית מחושבת כפונקציה של הסוד והזמן הנוכחי, בחלונות של לרוב 30 שניות.
3. התוצאה (= הסיסמה) היא בדרך כלל קוד קצר של 6 ספרות משתנה כל פרק זמן קבוע



מנגנון TOTP יכול לחסום מתקפות Bruteforce לחלווטין, כמו שמייד נראה, מאחר והסיכוי של התקוף לנחש את הסיסמה החדש פעמיית הוא אפסי. גם אם התקוף הצליח לגלוות באמצעות Bruteforce את הסיסמה של המשתמש, הוא נדרש לאמצעי זיהוי נוסף – כאשר מדובר לא סביר שהתקוף מסוגל לפצח את הסיסמה החדש פעמיית שנדרשת בתוך 30 שניות (ואחרת, גם אם יפוץ אותה – היא כבר תחלף מזו).

משמאל – דוגמה מותך האפליקציה Google Authenticator שבה ניתן לראות סדרה של סיסמות חד-פעמיות, כל אחת לשרת אימות אחר – כאשר כל אחד תחלף לאחר לכל היותר 30 או 60 שניות.

מאחר וכתוכפים, כאמור, איןנו מסוגלים לחשב את ה-TOTP, בעת זיהוי של הסיסמה במתקפת Bruteforce נעצור ונדווח: "הסיסמה זהה", אך המשתמש מאוקטב TOTP, והדבר ייחשב ככישלון.

תוצאת הניסוי:

User password was guessed, but user enabled TOTP. Stopping

Failed: password not found after 35919 attempts

Report saved: results/bf_sha256_totp_0_20260107_011757.json

על אף הצלחת התקוף לנחש את הסיסמה של המשתמש, בגלל השימוש ב-TOTP, התקיפה נכשלה והתקוף נעצר.

בצורה דומה, **כל ניסיון בדיקה שביצענו, בין אם על ידי Password Spray או על ידי Bruteforce, נכשל כאשר המשתמש הפעיל TOTP, אם או בלי קשר לרמת הקושי והמורכבות של הסיסמה.**

3. בחירת סיסמות נכונה -

כפי שראינו בניסויים שעשינו בדוח, בעוד שsistמאות קצות של כ-4 תווים, שלקוחות מותך מרחב פתרונות קטן יחסית, ניתנות לפיצוח במתקפות Bruteforce בתוך דקות או שעות בודדות

כאשר המשמשים בוחרים בסיסמאות ארוכות יותר, ומנסים להגדיל ככל שייתר את מרחב הפתרונות של הבעה, מתקפות Bruteforce הופכות להיות מגושמות מדי ודורשות פרק זמן לא ריאלי לפתרון הבעיה, שיכול להגיע למיליוני שנים.

המלצות: חשוב מאוד להשתמש בסיסמאות חזקות שמקילות מספר גדול ככל שייתר של תווים, אותיות קטנות וגדלות וסימנים מיוחדים.

סיכום ATIIM

1. **הגבלת התקיפה הניסויי:** הניסוי הוגבל במכoon לפרמטרים מבוקרים, בהם מספר ניסיונות, קצב שליחת הבקשות ומשך זמן התקיפה על מנת למנוע יצירה של עומס חריג על הרשת.
2. **אי-שימוש במידע רגיש:** לא נעשה שימוש בסיסמאות אמיתיות או במידע אישי שמוקדם ממשמשים אמיתיים. כל קלטי הניסוי נוצרו באופן מלאכותי, בשיטות שהוסבו בדרך כלל לצורכי הדוגמה.
3. **מניעת שימוש לרעה:** תוכרי הניסוי והנתונים לא מיועדים לשימוש בעולם האמיתי ולא יפורסמו או מפורטים באופן פומבי מעבר להקשרי בדיקת המטלה.

הצהרה ATIIM:

ניסוי זה בוצע במסגרת סביבת מעבדה מבוקרת וסגורת, שהוקמה למטרות לימוד בלבד. כל המערכת, המשמשים, הסיסמאות והנתונים שמעורבים בניסוי הם סינטטיים ויוצרים על ידינו, ואיןם שייכים למשמשים אמיתיים או מערכות אמיתיות.

במהלך הניסוי לא הוכחשה התקיפה של שום מערכת חיצונית, שירות ציבור או תשתיות שלא בבעלותנו, ולא נגרמה פגיעה בפרטיות של מידע אמיתי. כל הפעולות בוצעו על מנת לנתח את מנוגנוני ההגנה שמפורטים בניסוי זה ועל מנת להעריך את יעילותם.

מקורות

1. משפחת פונקציות הגיבוב SHA-2, ויקיפדיה - <https://en.wikipedia.org/wiki/SHA-2>
2. אלגוריתם פונקציית הגיבוב bcrypt, ויקיפדיה - <https://en.wikipedia.org/wiki/Bcrypt>
3. אלגוריתם פונקציית הגיבוב argon2, ויקיפדיה - <https://en.wikipedia.org/wiki/Argon2>
4. הדוגמה (יותר פשוטה מהדוח) של דימוי Bruteforce בפייתון - <https://youtu.be/HHOzhtrJJg0>

נספחים

חלק מקובץ ה-ZIP שהוגש, יחד עם דוח זה, הוגש:

1. סרטון קצר שמתאר את כל העבודה שלנו ומתאר איך הרצינו את הבדיקות.
2. מצגת שמתארת את עיקרי התוצאות והמסקנות של הדוח.
3. תוצאות גולמיות של הריצות שבוצעו בדוח

בנוסף לכל אלו, כל התשתיות שבהן השתמשנו בניסוי קיימות ב- GitHub, [כאן](#) – בהתאם לשיקולים האתיים, הגישה ל- repo תיסגר לאחר סיום הקורס

רשימת חממת אלפיים הsistמאות הcy נפוצות שבהן השתמשנו לצורך ביצוע מתקפות ה- Password Spray [לפחות מהאתר הבא](#)

סוו