

# דוח מחקר – מטלה תכנותית – קורס 20940

ממ"ן 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 מבוססי סיסמאות הם שכבת אבטחה מרכזית ברוב המערכות המודרניות, אבל היעילות שלהם תלויה בצורה ישירה באיכות והגדרות האלגוריתמים המשומשים במערכת להצפנה ולגיבוב של הסיסמאות.

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

גיבוב, או באנגלית Hashing הוא תהליך קריפטוגרפי **חד כיווני** שבו קלט באורך שרירותי, כמו סיסמה, ממופה לפלט באורך **קבוע** שנקרא ערך גיבוב. אחד המאפיינים החשובים והמרכזיים של גיבוב הוא חוסר היכולת 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 המשתמש, ונשווה את ערך הגיבוב המתקבל עם הערך השמור.

לעומת זאת, הצפנה היא תהליך דו כיווני שבו מידע מומר לצורה מוצפנת באמצעות **מפתח הצפנה**, וניתן לשחזר חזרה לצורתו המקורית באמצעות מפתח מתאים. הצפנה מיועדת לשמירה על סודיות של המידע שנדרש לפענוח עתידי ולכן היא שיטה לא מתאימה לאחסון של סיסמאות, מאחר וחשיפה של מפתח ההצפנה שהיה בשימוש להצפין את הסיסמאות תאפשר לתוקף לשחזר בצורה מלאה את כל הסיסמאות.

אחד השיטות המוכרות לתקיפה של מערכות מבוססות סיסמה היא Brute-Force, או בעברית "חיפוש ממצה". בשיטה הזאת התוקף מנסה באופן שיטתי מספר רב של סיסמאות אפשריות עבור חשבון מסוים, עד להשגת התאמה. היעילות של ההתקפה תלויה בכמה גורמים –

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

שיטה נוספת לתקיפה של מערכות כאלו נקראת Password-Spraying, והיא וריאציה מתוחכמת יותר של Brute-Force, שבה נעשה ניסיון להשתמש במספר מוגבל של סיסמאות, על פני מספר רב של חשבונות שונים. השיטה הזו נועדה לעקוף מגננונים שממומשים על ידי השרת כדי לנתר ולמנוע פריצה לחשבון יחיד.

המטרה של העבודה שלנו היא לבצע סדרה של ניסויים שיבחנו את ההשפעה של אלגוריתמי גיבוב שונים ומגננוני הגנה משלימים על העמידות של מערכות אימות מבוססות סיסמאות, ולהעריך את היעילות של המגננונים והאלגוריתמים האלו להתמודד עם שני סוגי ההתקפות שתיארנו עד כה בסביבת הניסוי.

### אלגוריתמי הגיבוב שנבדקו בדוח זה:

1. **SHA-256** – אלגוריתם גיבוב קריפטוגרפי מהמשפחה SHA-2, המפיק פלט באורך קבוע של 256 ביט. האלגוריתם תוכנן להיות מהיר ויעיל חישובית, תכונה שהופכת אותו לאלגוריתם שנמצא בשימוש נרחב. למרות זאת, דווקא המהירות הגבוהה שלו יכולה להפוך לחיסרון מאחר והתוקף יכול לבצע כמות גבוהה של ניסיונות גיבוב בשנייה. לכן שימוש באלגוריתם זה נחשב בטוח רק כשמשלבים אותו עם salt.
2. **Bcrypt** עם הפרמטר  $\text{cost} = 12$  – אלגוריתם גיבוב נוסף לאחסון סיסמאות שמבוסס על צופן שנקרא Blowfish, עם מגננון האטה מובנה. הפרמטר cost קובע את מספר סבבי החישוב האקספוננציאליים שמתבצעים, כך שכל עלייה של יחידה בערך זה מכפילה את זמן החישוב.
3. **Argon2id** עם הפרמטרים  $\text{time} = 1$ ,  $\text{memory} = 64\text{MB}$ ,  $\text{parallelism} = 1$  – האלגוריתם האחרון שנבחן הוא אלגוריתם מורכב יותר שמשלב הגנה ממתקפות תזמון והתקפות שמבוססות GPU. הפרמטר time מגדיר את מספר איטרציות החישוב, memory קובע את כמות הזיכרון שנדרשת לכל פעולת גיבוב, ו-parallelism קובע את מספר התהליכונים שפועלים במקביל.

### מגננוני אבטחה שנבדקו בדוח זה:

1. **Pepper – Pepper** הוא ערך סודי נוסף שמשולב בתהליך גיבוב הסיסמה, בדומה ל-Salt, אך בניגוד אליו הוא לא נשמר במסד הנתונים אלא במקום מאובטח נפרד, למשל, משתנה סביבה. ה-Pepper משותף לכל המשתמשים במערכת ומתווסף לסיסמה לפני פעולת הגיבוב. מטרתו להקשות על התוקף גם במקרה של דליפת מסד הנתונים עצמו. בלי ה-Pepper קשה לבצע מתקפות Brute force או להשתמש ב-Rainbow tables על ערכי הגיבוב שנחשפו. כמובן שחשיפה של ה-Pepper עצמו פוגעת ביעילות ההגנה ולכן האבטחה שלו חשובה מאוד.
2. **Rate Limiting** – מגננון הגנה בצד השרת שמגביל את מספר ניסיונות האימות שניתן לבצע בפרק זמן נתון, לפי כתובת IP, חשבון משתמש, או שילוב של שניהם. לאחר חריגה מהסף שהוגדר, השרת יכול להשהות תגובות, לחסום זמנית ניסיונות נוספים או לדרוש אימות נוסף כמו CAPTCHA. המגננון הזה מצמצם משמעותית את האפקטיביות של מתקפות Brute force ו-Password Spraying בגלל החסימות וזמן ההשהיה הארוך.
3. **TOTP – Time-based One-Time Password** – מגננון אימות דו-שלבי שמבוסס על קוד-חד פעמי שמשתנה כל פרק זמן קבוע, לרוב כל 30 שניות. הקוד משותף באמצעות סוד משותף בין השרת למכשיר המשתמש ובשילוב הזמן הנוכחי. גם אם התוקף מצליח להשיג את סיסמת המשתמש הוא לא יכול להשלים את תהליך ההתחברות ללא הקוד הזמני, ולכן TOTP הוא אמצעי אפקטיבי מאוד כדי למנוע את המתקפות שנבצע בדוח זה.

## מתודולוגיה

### מבנה כללי של המערכת

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

1. שרת אימות
2. לקוח תוקף

## 3. מנגנון הרצת ניסויים אוטומטי

השילוב של כל הרכיבים מאפשר לנו לשלוט בצורה מלאה בפרמטרי ההגנה בצד השרת, להריץ בצורה אוטומטית סט של מתקפות שונות בצד הלקוח, ולאסוף את המדדים הכמותיים לצורך ניתוח והשוואה בין תרחישים בשלב מאוחר יותר.

## שרת האימות

השרת מבוסס על Flask ומספק ממשק HTTP בסיסי לאימות משתמשים. הוא מדמה מערכת הזדהות ריאלית שכוללת בתוכה גם אפשרות למנגנוני אבטחה נפוצים.

בעת רישום משתמש, השרת שומר את פרטי המשתמש למסד נתונים מקומי מבוסס SQL. כמו בחיים האמיתיים, סיסמת המשתמש לא נשמרת בצורתה הגלויה, אלא רק לאחר הפעלת אלגוריתם הגיבוב המוגדר עליה. בהתאם לתצורת השרת בעת הרצת התקיפה, ניתן להפעיל Pepper גלובלי – ערך סודי שנשמר בצד השרת ומשולב בתהליך הגיבוב כדי להקשות על תוקפים.

## נקודות קצה שמוגדרות בשרת:

- POST /register – רישום משתמש:

נקודת קצה זו מאפשרת יצירת חשבון חדש. הבקשה כוללת שם משתמש וסיסמה, ובמידת הצורך גם הגדרת TOTP. במהלך הרישום מתבצעת הצפנה של הסיסמה בהתאם לתצורת השרת הנוכחית.

השימוש שלנו בנקודת קצה זו בניסוי היא יצירת משתמשים בצורה סינתטית בצורה אוטומטית לפני הרצת המתקפות.

- POST /login – אימות משתמש:

נקודת הקצה המרכזית בניסוי. מטפלת בניסיון התחברות באמצעות שם משתמש וסיסמה, ומיישמת את כל מנגנוני ההגנה שהוגדרו בתצורת האבטחה של השרת: בדיקת Rate limiting לפי חלון זמן, בדיקת נעילת חשבון לאחר מספר כשלונות, דרישת CAPTCHA לאחר מספר ניסיונות כושלים, דרישת אימות דו-שלבי, כמובן – במידה ומופעל.

התגובה משתנה בהתאם למצב המשתמש ומדיניות האבטחה.

- POST /login\_totp – אימות דו-שלבי:

נקודת קצה זו מופעלת לאחר אימות סיסמה מוצלח כאשר TOTP פעיל. היא מקבלת קוד חד פעמי ומבצעת אימות מול הסוד ששמור במסד הנתונים עבור אותו משתמש. היא כמובן בשימוש רק כאשר תצורת השרת מאפשרת TOTP.

- GET /admin/captcha\_token – יצירת CAPTCHA:

נקודת קצה זו משמשת ליצירת טוקן CAPTCHA. על מנת לקבל טוקן כזה יש להעביר ערך סודי, במקרה שלנו הוא ה- Group seed, כך שרק אדם יוכל להשתמש בנקודה כדי לקבל טוקן אמיתי.

בניסוי נשתמש בנקודה זו על מנת ליצור טוקן מבלי להפסיק את התקיפה, כאשר השרת דורש זאת במידה ואפשרות ה-CAPTCHA דלוקה.

- POST /admin/config – ניהול תצורת אבטחה:

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

```
{
  "bcrypt_cost": 12,
  "captcha_after_fails": 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.

- GET /admin/config – שליפת תצורה נוכחית:

נקודת קצה זו מאפשר לקרוא את תצורה האבטחה הנוכחית של השרת. היא משמשת אותנו כדי לוודא שהתצורה מוגדרת כראוי לפני כל ניסוי ולשמירה של התצורה כחלק מתיעוד התוצאות.

## יצירת סיסמאות

לצורך הניסוי נוצרו מראש שלושה מאגרי סיסמאות ברמות קושי שונות:

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

מאגרי הסיסמאות שלנו נוצרו באמצעות סקריפט ייעודי שיצרנו לשם כך, ונשמרו כקבצים סטטיים. סיסמאות אלו משמשות לרישום המשתמשים בשרת.

בצד התוקף, למשל במתקפות Bruteforce, נעשה שימוש במחולל סיסמאות אחר, שאיננו מוזן ממאגר הסיסמאות (בצורה הזאת המתקפה איננה הייתה Bruteforce מאחר והתוקף היה חשוף לסיסמאות).

מחולל הסיסמאות הזה סורק באופן שיטתי את מרחב הפתרונות האפשרי, והוא לא תלוי במאגרים הסטטיים.

חוקים ליצירת הסיסמאות:

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

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

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

**סיסמה חזקה** – רצף בין 8 ל-10 תווים הלקוחים מתוך הסט האלפאנומרי: אותיות קטנות וגדולות באנגלית והספרות 0 עד 9.

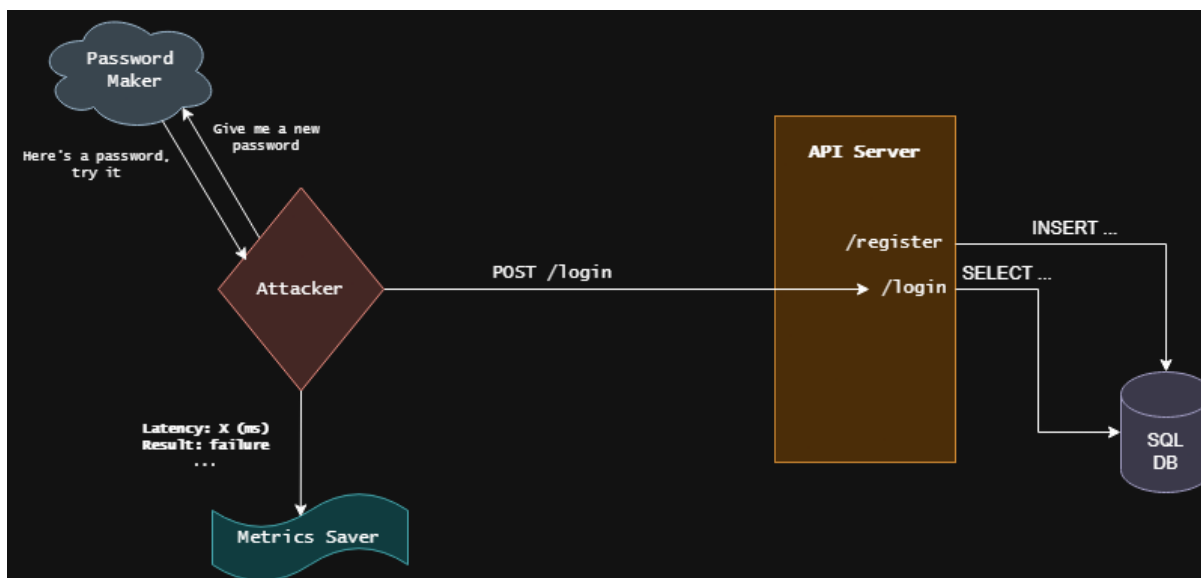
כל אחד מהרצפים נבחר בצורה רנדומלית, בבחירה אחידה של תו אחרי תו מתוך הקבוצה הרלוונטית.

## הרצת בדיקות בצורה אוטומטית

הבדיקות שנעשו בניסוי בוצעו בצורה אוטומטית כדי להבטיח עקביות ויכולת לשחזר את התוצאות בלי התערבות של בני אדם. האוטומציה אפשרה לנו לחזור על הניסוי תחת תנאים זהים

מערכת הבדיקות מופעלת באמצעות סקריפט מרכזי אשר:

1. מאתחל 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 וכלל מספר קטן של משתמשים וסיסמאות שלא בהכרח ריאליים למציאות האנושית. חומרת הניסוי וההיקף שלו יהיה מוגבל – הניסוי בוצע בפרק זמן קצר, עם עומס נמוך, ובסביבת בדיקות מבוקרת שלא מייצגת מערכת פרודקשן עמוסה ואמיתית שחייה תחת תנאי רשת מציאותיים שבהם קיים עומס שנוצר כתוצאה משימוש טבעי בשרת האימות על ידי משתמשים לגיטימיים.

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

- בוצעו 35,919 ניסיונות בסך הכול עד למציאת הסיסמה
- זמן הריצה הכולל היה 14.5 דקות, כאשר בוצעו בערך 41 ניסיונות בשנייה 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 הכול עד למציאת הסיסמה
- זמן הריצה הכולל היה כ-12 דקות, כאשר בוצעו בערך 33 ניסיונות בשנייה 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 –**

- בוצעו 225,88 ניסיונות בסך הכול עד למציאת הסיסמה
- זמן הריצה הכולל היה כשעה ו-53 דקות, כאשר בוצעו בערך 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-#2****משתמש #1:**

- בוצעו 35,919 ניסיונות בסך הכול עד למציאת הסיסמה
- זמן הריצה הכולל היה כ-48 דקות, כאשר בוצעו בערך 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:**

- בוצעו 35,919 ניסיונות בסך הכול עד למציאת הסיסמה
- זמן הריצה הכולל היה כ-33 דקות, כאשר בוצעו בערך 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:**

- בוצעו 35,919 ניסיונות בסך הכול עד למציאת הסיסמה
- זמן הריצה הכולל היה כשעתיים ועשרים דקות, כאשר בוצעו בערך 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:**

- בוצעו 24,931 ניסיונות בסך הכול עד למציאת הסיסמה
- זמן הריצה הכולל היה כשעה ו-36 דקות, כאשר בוצעו בערך 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 –**

- בוצעו 35,919 ניסיונות בסך הכול עד למציאת הסיסמה
- זמן הריצה הכולל היה כ-15 דקות, כאשר בוצעו בערך 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 –**

- בוצעו 35,919 ניסיונות בסך הכול עד למציאת הסיסמה
- זמן הריצה הכולל היה כ-16 דקות, כאשר בוצעו בערך 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.
- סטטוס: **כישלון כתוצאה מ-TOTP דלוק**

#### ניסוי 8: תקיפת BruteForce, sha256, עם Rate limiting – משתמש #2

- בוצעו 24,931 ניסיונות בסך הכול עד למציאת הסיסמה
- זמן הריצה הכולל 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.
- סטטוס: **הצלחה**

#### ניסוי 9: תקיפת BruteForce, sha256, עם CAPTCHA – משתמש #1

- בוצעו 35,919 ניסיונות בסך הכול עד למציאת הסיסמה
- זמן הריצה הכולל 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.
- סטטוס: **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 שרת אימות אמיתי.

<u>משתמש #1:</u>	תוצאה: <b>החשבון נחסם</b>
אחרי 1000 ניסיונות, התוקף לא הצליח לנחש את הסיסמה	<u>משתמש #3:</u>
תוצאה: <b>החשבון נחסם</b>	אחרי 1000 ניסיונות, התוקף לא הצליח לנחש את הסיסמה
<u>משתמש #2:</u>	תוצאה: <b>החשבון נחסם</b>
אחרי 1000 ניסיונות, התוקף לא הצליח לנחש את הסיסמה	

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

#### ניסוי 11: תקיפת BruteForce, sha256, ללא הגנות – משתמש עם סיסמה בחזקה "בינונית" –

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

עד לסיום שעתיים אלו, בוצעו 191,223 ניסיונות. כלומר בוצעו כ-26 ניסיונות בשנייה. שימוש המעבד הממוצע היה כ-6 אחוז ושימוש הזיכרון הממוצע היה כ-35 אחוז. הסיסמה האחרונה שמוצתה הייתה aaaaYjY.

לאחר שביצענו אקסטרפולציה על פי נתונים אלו, בגלל שסיסמאות בינוניות וחזקות לקוחות מתוך סט אלפבית אלפאנומרי עם 62 תווים (אותיות קטנות וגדולות באנגלית ומספרים), והסיסמה שאותה ניסינו לפצח הייתה L82Kb7M, היינו צריכים לבצע עוד  $2.15 \times 10^{12}$  ניסיונות שהיו לוקחים לנו  $8.3 \times 10^{10}$  שניות, שהם בערך 2,600 שנים, מספר לא ריאלי עבור תוקף.

**ניסוי 12: תקיפת Bruteforce, sha256, ללא הגנות – משתמש עם סיסמה בחזקה "חזקה" -**

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

עד לסיום שעתיים אלו, בוצעו 191,223 ניסיונות. כלומר בוצעו כ-26 ניסיונות בשנייה. שימוש המעבד הממוצע היה כ-6 אחוז ושימוש הזיכרון הממוצע היה כ-35 אחוז. הסיסמה האחרונה שמוצתה הייתה aaaaaXUo.

לאחר שביצענו אקסטרפולציה על פי נתונים אלו, בגלל שסיסמאות בינוניות וחזקות לקוחות מתוך סט אלפבית אלפאנומרי עם 62 תווים (אותיות קטנות וגדולות באנגלית ומספרים), והסיסמה שאותה ניסינו לפצח הייתה Zwi1GNKo73, היינו צריכים לבצע עוד  $7.09 \times 10^{17}$  ניסיונות שהיו לוקחים לנו  $8.3 \times 10^8$  שנים, שהם בערך 865 מיליון שנים, מספר מאוד לא ריאלי עבור תוקף שבו ניתן להכריז שבסבירות גבוהה, 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, נשתמש ברשימה גדולה של סיסמאות שכוללת את חמשת אלפים הסיסמאות הכי נפוצות למשתמשים ברשת.

בנוסף, כל המתקפות שנריץ בדו"ח זה מהסוג הנ"ל יבוצעו על גבי כל המשתמשים במערכת, כלומר: סיסמאות קלות, בינוניות וקשות. ההחלטה הזאת נובעת מהעובדה שהרשימה כוללת כאחד סיסמאות קצרות וארוכות, ולכן אפשר לבחון את סיכויי ההצלחה 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 ניסיונות התחברות
- משך זמן התקיפה הכולל היה 1 שעות ו-8 דקות
- במהלך התקיפה נפרצו 4 חשבונות שונים

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

אחת ההגנות שראינו בשלבים קודמים יותר בניסוי היא הגנת Account Lockout, שנועלת את הכניסה לחשבון אחרי מספר מסוים של ניסיונות כושלים להתחברות לחשבון.

בניסוי זה נראה כי הגנה זו היא אפקטיבית גם ביחס למתקפת Password Spray ולא רק ביחס ל-Bruteforce. תוצאות הניסוי מראות כי על אף שתקיפה מבוססת Password Spray ללא הגנות הצליחה לפרוץ 4 מהמשתמשים שהשתמשו בסיסמאות נפוצות, מאחר ונדרשו לכך יותר מכמות הניסיונות שהוגדרו בתצורת ההגנה בשרת (1000 ניסיונות עד לחסימת החשבון), ככל שהזמן עבר לאורך התקיפה, כמות החשבונות שניתן היה לנסות את הסיסמאות עליהם הלך וקטן – מאחר וכל חשבון שנוסו עליו יותר מ-1000 סיסמאות נחסם.

כלומר, החל משלב מסוים על אף העובדה שנשארו סיסמאות נפוצות ברשימה, כל החשבונות היו חסומים והפיצוח נכשל.

**תוצאה: 0 מתוך 30 חשבונות פוצחו.**



### ניסוי 15: תקיפת Password Spray, הגנת Rate limit – על גבי כל משתמשי המערכת –

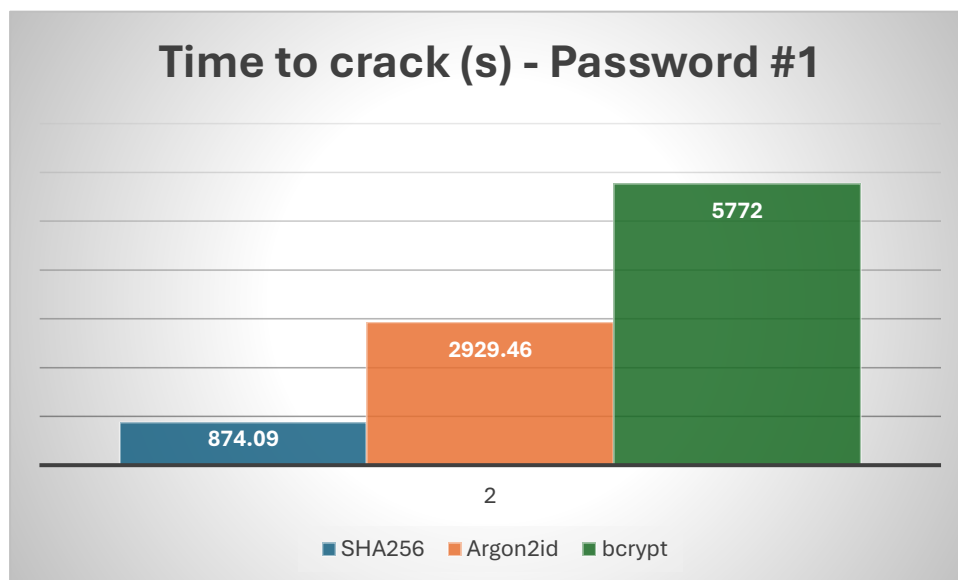
בעת שילוב הגנת ה-Rate limiting, משך התקיפה התארך מאחר ונדרשים לבצע הפסקות יזומות בעת התקיפה על משתמשים שעברו את ההגבלה לכמות ניסיונות ההתחברות שאפשר לבצע עליהם בפרק הזמן הנתון. כתוצאה מכך, משך זמן הניסוי עבר את הרף העליון של שעתיים, כשבפרק זמן זה 3 סיסמאות פוצחו, במקום 4 בתצורה הרגילה

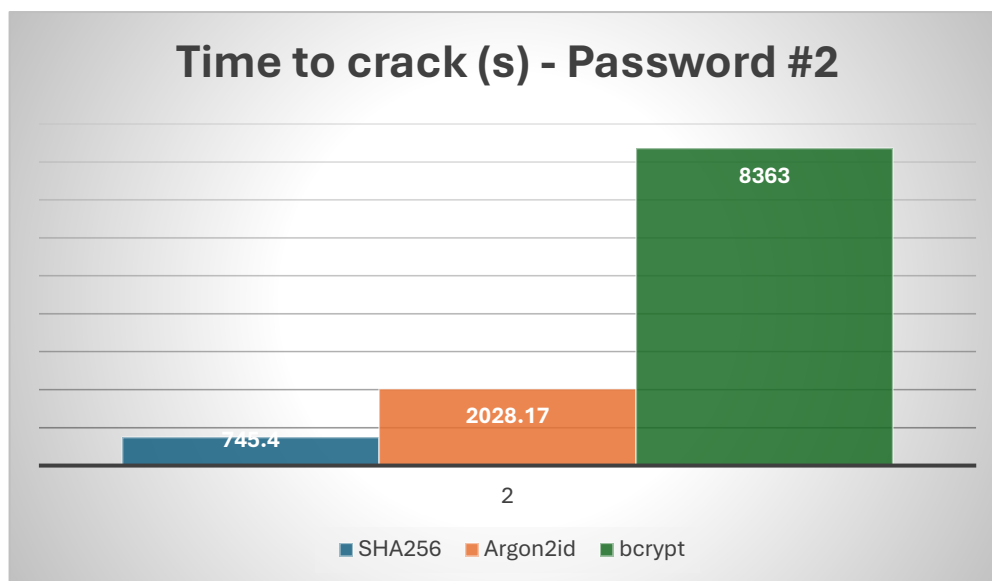
## ניתוח ודיון

### 1. השוואת היעילות והעמידות של מנגנוני גיבוב שונים:

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

עבור משתמש #1:



עבור משתמש #2:

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

יש לציין שבעוד שניתן לצפות מראש שהאלגוריתמים argon2id ו-bcrypt שניהם יהיו איטיים יותר מ-SHA256, התחרות ביניהם תלויה בפרמטרים שאיתם רצים. בגלל שבחרנו באופן דיפולטיבי cost factor של 12 ל-bcrypt, יש  $2^{12} = 4096$  איטרציות במהלך החישוב שמאטות את החישוב בצורה משמעותית, כאשר הפרמטרים שהוגדרו ל-argon2id הם פרמטרים זולים יותר. ולכן גם ייתכן שתצורות שונות יהפכו את argon2id לאיטי יותר מ-bcrypt.

**המלצות לפעולה:**

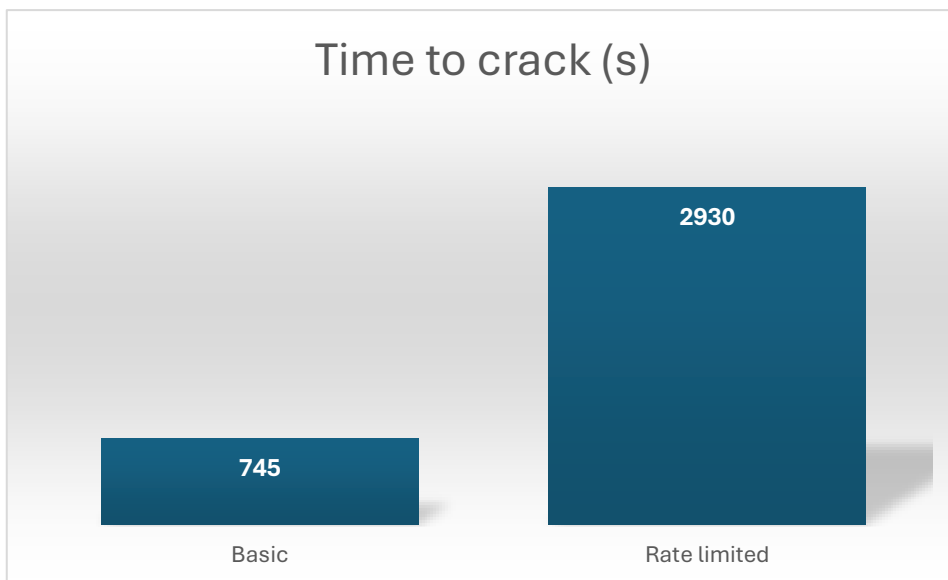
על מנת להגן על שרת האימות בצורה המיטבית נציע להשתמש באלגוריתמי הגיבוב הבאים, על פי סדר יורד של רמת האבטחה שהם מציעים:

- Bcrypt
- Argon2id
- SHA256

## 2. שימוש במנגנוני הגנה והשפעתם על יכולת הפיצוח:

### Rate Limiting מגננון

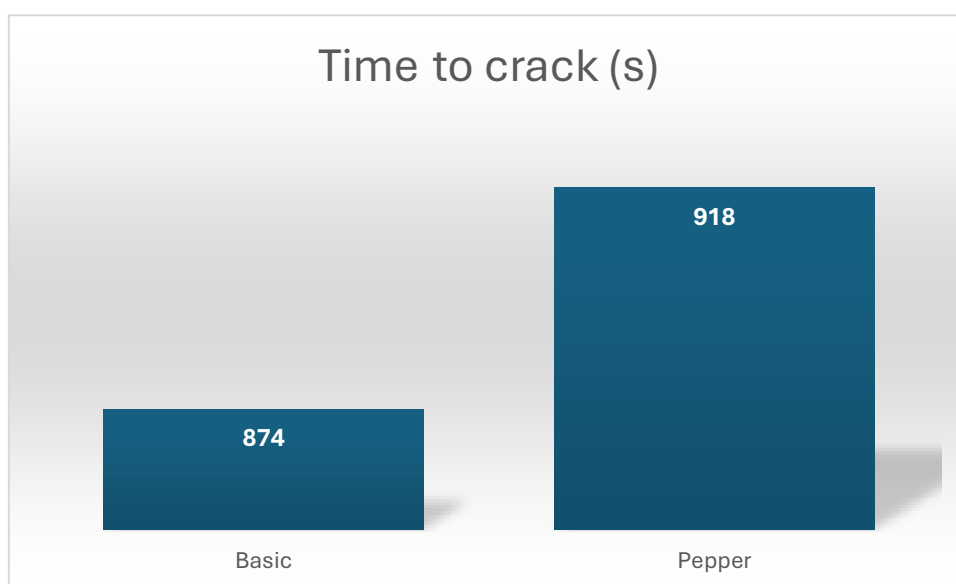
בגרף הבא ניתן לראות את ההשוואה בין 2 רמות הגנה של השרת במתקפת Brute force : הראשונה – ללא הגנות נוספות, והשנייה – כאשר הפעלנו את מנגנון ה- Rate limiting עם ההגדרה (המקלה יחסית) של עד ל-500 בקשות בכל 60 שניות. כלומר, בין כל 60 שניות התוקף יכול לבצע עד ל-500 בקשות התחברות.



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.

### Pepper שילוב ערך

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



ההבדל הזניח בין זמני הריצה (874 שניות ללא ההגנה מול 918 שניות עם שילוב ערך Pepper) נובע מכך שהוספת ה-Pepper מוסיפה שלב חישובי נוסף לכל ניסיון, ולכן למרות 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 מתבטאת בעיקר במתקפות Offline, שלא כיסינו בניסוי זה. במקרה כזה, התוקף מחזיק את מאגר הנתונים המגובב (למשל, לאחר שהוא הודלף משרת האימות), אבל הוא לא מחזיק בערך ה-Pepper שמאוחסן בנפרד, למשל במשתנה סביבה בתוך השרת. בלי הערך הזה, כל ניסיון פיצוח ייכשל גם אם ניחוש הסיסמה אכן נכון, ולכן התוקף מחוייב לא רק לנחש את הסיסמה עצמה, אלא גם את הערך של ה-Pepper, ובכך מרחב החיפוש גדל משמעותית והעלות החישובית של ההתקפה גדלה מאוד.

### הפעלה של Account Lockouts

ההגנה הבאה שנבחן היא הגנה שמבצעת נעילה של החשבונות כאשר התוקף עובר את סף הניסיונות השגויים שאפשריים לכניסה לחשבון. על פי ההנחיות, כאשר השרת מבצע נעילה של החשבון, התוקף יעצור את הניסיונות לפריצה לחשבון. תרחיש כזה במציאות הוא סביר, למשל במצבים שבהם כאשר החשבון נחסם (כשמנסים יותר מדי סיסמאות שגויות) עד שבעל החשבון האמיתי יאמת את זהותו, למשל דרך המייל המקושר לחשבון.

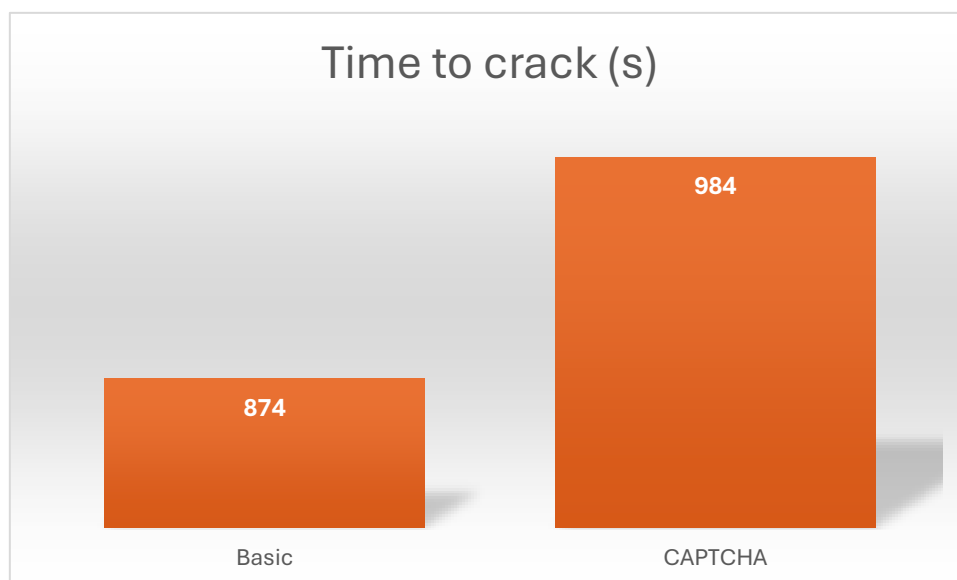
כתוצאה מכך, הגנה זו הופכת לחזקה מאוד נגד מתקפות Bruteforce: **במהלך הניסוי לא הצלחנו לפצח אף סיסמה מהמאגר כאשר השרת פעל עם הגדרה זו עם מתקפה מהסוג הנ"ל**

### הפעלה של מגננון CAPTCHA

נזכיר שכמצופה מאיתנו בניסוי, מאחר ותוקף אמיתי יכול במקרים מסוימים לעבור אתגרי CAPTCHA גם במציאות, לאחר X ניסיונות כשהשרת מצפה מאיתנו לעבור אתגר CAPTCHA, אנחנו משתמשים ב-Endpoint בשרת שמיועד לשימוש שלנו בלבד, שמאפשר לקבל טוקן תקין כזה. לאחר X ניסיונות נוספים, נצטרך לקבל אחד נוסף.

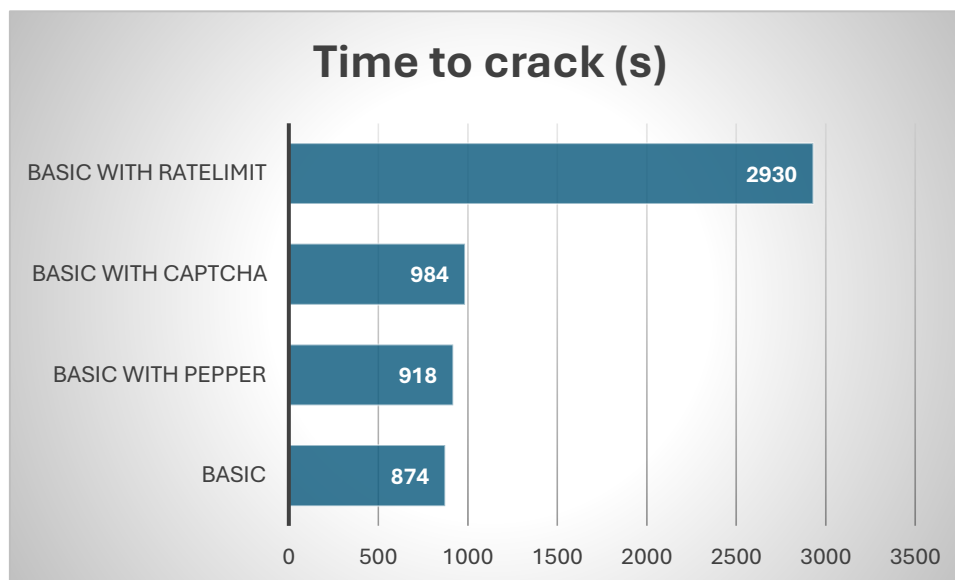
לכן אנחנו מצפים שזמן הפיצוח יהיה ארוך יותר כתוצאה מתקורה בבקשות הרשת שמיועדות למעבר אתגרי CAPTCHA.

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



נעיר, שעל אף שההבדל בזמן הפיצוח איננו ענק, הגדרות ניסוי זה דרשו אתגר CAPTCHA כל 20 ניסיונות. ככל שמספר זה יקטן, תוקף פוטנציאלי יצטרך לעבור יותר אתגרים לעיתים יותר תכופות. בנוסף לכך, במציאות אפשר לצפות שההבדל יהיה גדול יותר מאחר ואורך הזמן שלוקח לעבור אתגר מהצורה שמופיע בתמונה למטה ארוך יותר ממעקף בקשת ה-HTTPS שהתוקף ביצע בניסוי שלנו –

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



#### המלצות:

מהניסויים שלנו עולה כי סדר העדיפויות בעת בחירת ההגנות שכוסו עד כה בשרת, בנוסף להגנות שיכולות למנוע לחלוטין את ההצלחה של מתקפות Brute force, כמו הגנת Account Lockout והגנת TOTP שנסקור מיד, הן:

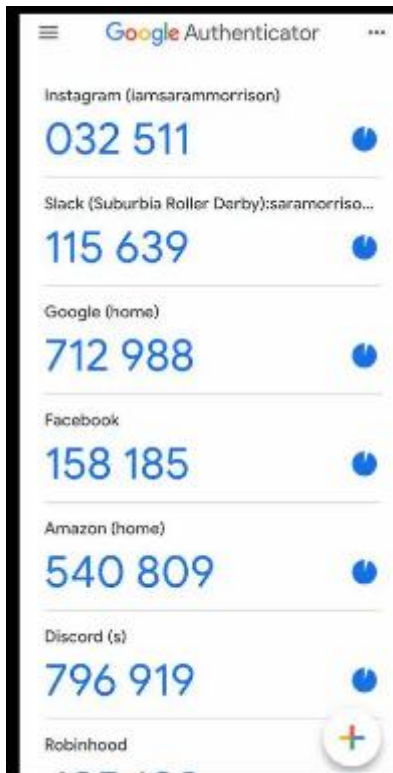
1. הגנת Rate limit, בשאיפה למספר ניסיונות קטן ככל האפשר עד להפעלת ההגבלה.
2. הגנת CAPTCHA

## 3. שילוב של Pepper בעת הגיבוב במסד הנתונים

הפעלה של מנגנון TOTP

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

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



מנגנון TOTP יכול לחסום מתקפות Bruteforce לחלוטין, כמו שמייד נראה, מאחר והסיכוי של התוקף לנחש את הסיסמה החד פעמית הוא אפסי. גם אם התוקף הצליח לגלות באמצעות Bruteforce את הסיסמה של המשתמש, הוא נדרש לאמצעי זיהוי נוסף – כאשר מאוד לא סביר שהתוקף מסוגל לפצח את הסיסמה החד פעמית שנדרשת בתוך 30 שניות (ואחרת, גם אם יפצח אותה – היא כבר תתחלף מאז).

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

מאחר וכתוקפים, כאמור, איננו מסוגלים לחשב את ה-TOTP, בעת זיהוי של הסיסמה במתקפת Bruteforce נעצור ונדווח: "הסיסמה זוהתה, 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 התוקף לנחש את הסיסמה של המשתמש, בגלל השימוש ב-TOTP, התקיפה נכשלה והתוקף 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 סיסמאות נכונה -**

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

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

המלצות: חשוב מאוד כמשתמשים להשתמש בסיסמאות חזקות שמכילות מספר גדול ככל שיותר של תווים, אותיות קטנות וגדולות וסימנים מיוחדים.

## שיקולים אתיים

1. **הגבלת היקף הניסוי**: הניסוי הוגבל במכוון 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 שמקורם ממשתמשים אמיתיים. כל קלטי הניסוי נוצרו באופן מלאכותי, בשיטות שהוסברו בדו"ח לצורכי ההדגמה.
3. **מניעת שימוש לרעה**: תוצרי הניסוי והנתונים לא מיועדים לשימוש בעולם האמיתי ולא יפורסמו או מפורסמים באופן פומבי מעבר להקשרי בדיקת המטלה.

## הצהרה אתית:

ניסוי זה בוצע במסגרת סביבת מעבדה מבוקרת וסגורה, שהוקמה למטרות לימוד בלבד. כל המערכות, המשתמשים, הסיסמאות והנתונים שמעורבים בניסוי הם סינתטיים ויוצרו על ידינו, ואינם שייכים למשתמשים אמיתיים או מערכות אמיתיות. במהלך הניסוי לא התבצעה תקיפה של שום מערכת חיצונית, שירות ציבור או תשתית שלא בבעלותנו, ולא נגרמה פגיעה בפרטיות של מידע אמיתי. כל הפעולות בוצעו על מנת לנתח את מנגנוני 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 argon2id, ויקיפדיה - <https://en.wikipedia.org/wiki/Argon2>
4. הדגמה (יותר פשוטה מהדוח) של דימוי Bruteforce בפייתון - <https://youtu.be/HHOzhtrJJg0>

## נספחים

כחלק מקובץ ה-ZIP שהוגש, ביחד עם דו"ח זה, הגשנו:

1. סרטון קצר שמתאר את כלי העבודה שלנו ומתאר איך הרצנו 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](https://github.com/idocalm/20940-project) (<https://github.com/idocalm/20940-project>) – בהתאם לשיקולים האתיים, הגישה ל-repo תיסגר לאחר סיום הקורס

רשימת חמשת אלפים הסיסמאות הכי נפוצות שבה השתמשנו לצורך ביצוע מתקפות ה-Password Spray לקוחות [מהאתר הבא](#)

## סוף