

## דוח מחקר – פרויקט מבוא לאבטחת המרחב המקוון

### מבוא:

אבטחת המרחב המקוון תלויה באופן ישיר בחוזקם ביעילותם של מנגנוני האימות. אימות מבוסס סיסמאות נותר האמצעי המרכזי להגנה על פרטיות המשתמשים ברשת. הגישה הבסיסית לאימות מתבססת על הזנת שם משתמש וסיסמה, המהווים את ההוכחה לבעלות על החשבון. סיסמאות אלה מאבטחות מידע ברמות חשיבות שונות, החל מרשתות חברתיות ועד מערכות פיננסיות לדוגמה בנקים. לכן, שמירה על שלמות הסיסמאות ומניעת גישה בלתי מורשית לאתרים הוא חלק גדול ומשמעותי באבטחת המרחב המקוון.

עם זאת, דווקא בגלל החשיבות שלהן, מערכות אימות מהוות יעד קבוע לתקיפות. תוקפים מנסים לנצל חולשות באופן שבו סיסמאות נשמרות או לבחור סיסמאות חלשות, ומשתמשים בכלים אוטומטיים כדי לנסות לנחש סיסמאות בשיטות של ניסוי וטעייה.

בפרויקט זה נתמקד בשני סוגי תקיפות עיקריים: Password Spraying ו-Brute Force. נבדוק כיצד שיטות שונות לאחסון סיסמאות ומנגנוני הגנה משפיעות על קצב ההצלחה של התקיפות האלה. נבחן את ההבדלים בין שיטות האחסון וההגנה, ונמדוד אותם בצורה כמותית - כמו זמן עד לפריצה מספר נסיונות נדרשים. בנוסף, נבדוק איך חוזק הסיסמה משפיע על מהירות הפריצה לחשבון.

### סוגי התקיפה – הסבר:

**Brute Force** – ניסוי של כל אוספי הסיסמאות האפשריים על משתמש מסוים יחיד מקור  
**Password Spraying** – תקיפה בה מנסים מספר סיסמאות נפוצות על פני משתמשים רבים.

### מנגנוני אחסון סיסמאות – הסבר:

**SHA-256** - פונקציית גיבוב שמקבלת קלט (לא משנה מה אורכו). היא מעבירה אותו לערך גיבוב באורך 256 ביטים. ההמרה של הקלט לערך גיבוב נעשה ע"י אלגוריתם שמפצל את הקלט לכמה בלוקים ועושה פעולות על סיביות על כל בלוק באופן חזרתי עד שמגיע 256 ביטים. מתבצע במהירות מה שנותן לתוקף יכולות ניסוי סיסמאות רבים במהירות גם כן. מקור

**Argon2** – אלגוריתם שנועד להקשות על תקיפות brue-force מוסיף מלח אקראי לסיסמא ואז עושה hash האלגוריתם הזה עובד באיטיות בכוונה ומשתמש בהרבה זיכרון מחשב בעט חישוב hash

**bcrypt** – אלגוריתם שנועד להקשות על תקיפות brue-force מוסיף מלח אקראי לסיסמא ואז עושה hash האלגוריתם הזה עובד באיטיות בכוונה כדי להאט תקיפות בשונה מאלגוריתם Argon2 הוא לא משתמש גם במשאבי זיכרון גדולים מאד.

### מנגנוני הגנה – הסבר:

**Rate Limit** מנגנון הגנה בו המערכת מזהה ניסוי כניסה רב מידי ממקור מסוים -  
ואז חוסמת לו את האפשרות לכניסה למערכת לזמן מסוים

**נעילת חשבון** – מנגנון הגנה בו המערכת מזהה ניסוי כניסה למשתמש מסוים מספר רב של פעמים המערכת תחסום את כניסת המשתמש הספציפי הזה למערכת לזמן מסוים  
**TOTP** -קוד חד פעמי שמשתנה כל כמה שניו והוא שלב נוסף בזהוי  
**CAPTCHA** - בודק שהמשתמש הוא אדם ע"י יצירת שדה קלט ורק תשובה נכונה מאפשרת להמשיך  
**Pepper** - הוספת סוד כללי קבוע לסיסמא בדאטה בייס

## תצורה ניסויית מיתולוגיה:

הניסויים בוצעו על שלוש שיטות שונות לאחסון סיסמאות

SHA-256: הוספנו salt ייחודי לכל משתמש שהוא username: GROUP\_SEED.

bcrypt: הגדרנו את העלות להיות 12 לפי דרישות המטלה.

Argon2id: זכרון MB64 ז"א KB65,536, זמן 1, איטרציית זמן 1

עבור כל שיטה נוצר קובץ משתמשים ייעודי מסוג גייסון, שכלל כ-30 משתמשים כאשר לכל משתמש הוגדרה סיסמה ברמת חוזק שונה: חלשה, בינונית או חזקה

חילוק הקבצים עבור כל שיטה גיבוב מאפשרת להשוות בין תוצאות התקיפות תחת אותם תנאי עומס, כאשר ההבדל היחיד הוא אלגוריתם הגיבוב שבו נעשה שימוש.

**חלשה:** "בדרך כלל אם לא מגבילים משתמשים בבחירת הסיסמה, הם נוטים לבחור מילים או ביטויים נפוצים, חלקים משמם או תאריכים ידועים." שמנו מילים נפוצות וסיסמאות של מספרים בלבד [מקור](#)

**בינוני:** סיסמה שמורכבת משילוב של מילים נפוצות ומספרים.

**חזק:** סיסמה ארוכה, ללא רצף הגיוני המורכבת מכל התווים האפשריים.

**לכל משתמש נשמרו הנתונים הבאים:**

שם משתמש (username), סיסמה לאחר גיבוב (hashed password), קטגוריית חוזק הסיסמה (strength\_category), שיטת הגיבוב (hash\_mode)

על כל אחת משיטות אחסון הסיסמאות נבדקו חמש רמות הגנה שונות:  
ללא הגנות - baseline

הגבלת קצב ניסיונות - Rate Limiting

נעילת חשבון - Account Lockout

token . מבוסס CAPTCHA

. מבוסס לכל משתמש TOTP ואימות דו-שלבי

בנוסף הוספנו בדיקה בה מוסיפים לכל הסיסמאות ערך סוד גלובלי לפני הגיבוב הוא נשמר בנפרד מהדאטה בייס - הוספת pepper.

בכל תרחיש, ניסיונות ההתחברות נרשמו לקבצי לוג בפורמט JSON Lines. כל רשומת לוג כללה את השדות: חותמת זמן, GROUP\_SEED, שם משתמש, שיטת גיבוב, סוג הגנה, תוצאת הניסוי, latency, CPU.

תקיפות מסוג Password Spraying בוצעו באמצעות רשימת סיסמאות נפוצות שהופעלה על כלל המשתמשים. את רשימת הסיסמאות הנפוצות מצאנו באתר הבא: [מקור](#)

תקיפות מסוג Brute Force בוצעו כבדיקת כל אפשרויות הקומבינטוריקה (ע"י שימוש בספריית python iterator) על משתמש יחיד. קודם בדיקת סיסמאות באורך 6 תוך כדי הרחבה charset האפשרי בכל שלב ואז מעבר לסיסמאות ארוכות יותר.

כל התקיפות בוצעו תחת מגבלות קבועות של זמן ריצה ומספר ניסיונות תקיפה בהתאם לדרישות המטלה. הריצה נעצרת מיד כאשר אחת מהמגבלות נחצית

לוגיקת תקיפת Brute Force תוכננה כך שברגע שסיסמה של משתמש מסוים מתגלה, ההתקפה עוברת למשתמש הבא. גישה זו נועדה לדמות התנהגות של תוקף ריאלי, שמנצל את מגבלת הניסיונות על פני כמה חשבונות שונים ולא מתמקד רק בחשבון אחד. עבור כל שילוב של שיטת אחסון סיסמאות ומנגנון הגנה, הורצו ברצף תקיפת Password Spraying ולאחריה תקיפת Brute Force. לכל הרצה נוצר קובץ לוג נפרד, לאחר סיום כל ההרצות, קובצי הלוג שימשו לחישוב מדדים כמותיים, כגון מספר ניסיונות עד הצלחה, מספר החשבונות שנפרצו וזמני הריצה, וכן לניתוח איכותי של היעילות של מנגנוני ההגנה ואלגוריתמי הגיבוב.

חילקנו את ההרצה ל 2 שלבים עיקריים:  
שלב ראשון - הרצנו את קוד ההתקפה עם שלושת שיטות הגיבוב (SHA256, bcrypt, Argon2id) ללא שום הגנות.  
שלב שני - הרצה שניה עם כל מנגנון הגנה בנפרד.

בנינו 3 קבצי הרצה:

**HashTable\_generator.py**: קובץ יצירת הסמאות בטבלאות ויצירת קבצי json למשתמשים.

**attack.py**: קובץ התקיפה של 2 האלגוריתמים Password Spraying ו-Brute Force יוצר json לכל סוגי התקיפה.

**defence.py**: קובץ ההגנות.  
ההרצה מתבצעת בattack והיא קוראת לקובץ של ההגנות.

**\*\*** בכדי להריץ ולקבל נתונים תואמים לנתונים שלנו צריך להריץ את קובץ **HashTable\_generator** יותר קבצי json של המשתמשים. ואחריו להריץ את - מריץ את כל השילובים **attack**.

## תוצאות:

**סיכום password spraying ללא הגנות: 50000 ניסיונות תקיפה:**

מנגנון גיבוב	הגנה	סך ניסיונות	מס הצלחות	זמן ביצוע כולל (ש')	ניסיונות לשניה	זמן עד הצלחה שניה
SHA-256+salt	none	50,000	9	2.7	18,518.52	0.011
	rate	50,000	9	5019.1	9.96	9.04
	lock	50,000	4	3.0	16,666.67	0.011
	captcha	50,000	9	4.8	10,416.67	0.011
	totp	50,000	9	4.8	10,416.67	0.016
bcrypt	none	18,300	6	7208	2.54	35.365
	rate	17970	6	7204.8	2.49	38.207
	lock	18,090	6	7201.1	2.51	40.164
	captcha	18,390	6	7211.3	2.55	35.612
	totp	18,397	6	7210.7	2.55	35.842
Argon2id	none	50000	9	4065.1	12.301	6.152
	rate	50000	9	6994.5	7.149	11.743
	lock	50000	4	24.1	2074.68	5.999
	captcha	27,000	6	7355.1	4.08	317.651
	totp	30,180	6	7375.2	4.09	6.29

**סיכום הרצה brute force ללא הגנה:**

**משתמש 1: 123456**

<u>מנגנון גיבוב</u>	<u>סך ניסיונות</u>	<u>תוצאה</u>	<u>ממוצע Letency</u>	<u>Cpu ממוצע</u>	<u>זיכרון ממוצע</u>	<u>זמן ביצוע כולל</u>
SHA-256+salt	123,457	הצלחה	ms 0.06	99.61%	MB 18.29	7.803 ש
bcrypt	18,121	כישלון	ms 397.35	99.41%	MB 7.91	7200.37 ש
Argon2id	59,109	כישלון	ms 128.75	99.87%	MB 71.84	7610.29 ש

<u>מנגנון גיבוב</u>	<u>סך ניסיונות</u>	<u>תוצאה</u>	<u>ממוצע Letency</u>	<u>Cpu ממוצע</u>	<u>זיכרון ממוצע</u>
SHA-256+salt	1,000,000	כישלון	ms 0.07	99.64%	MB 12.29
bcrypt	12,846	כישלון	ms 560.49	99.76%	MB 7.51
Argon2id	109,427	כישלון	ms 65.80	99.72%	MB 71.92

משתמש 5: 111111

הרצת brute force:

<u>מנגנון גיבוב</u>	<u>הגנה</u>	<u>סך ניסיונות</u>	<u>ניסיונות לשניה</u>	<u>זמן עד הצלחה</u>
SHA-256+salt	none	340	48,571	0.007
	Rate limiting	340	10.26	33.133
	lock	50,000	X	Account locked
	captcha	340	10,625	0.032
	totp	340	10.26	0.025
bcrypt	none	340	4.2	80.881
	Rate limiting	340	4.19	81.074
	lock	50,000	X	Account locked
	captcha	340	2.09	162.109

163.152	2.08	340	totp	
13.058	26.03	340	none	<b>Argon2id</b>
33.574	10.12	340	Rate limiting	
Account locked	X	50,000	lock	
26.701	12.73	340	captcha	
26.571	12.79	340	totp	

ניתן לראות כי לשיטת גיבוב SHA-256+salt ולשיטת גיבוב Argon2id. הגנת rate limiting מאטה את זמן התקיפה באופן משמעותי, פי  $\approx 4,734$  בשיטת גיבוב של SHA-256 ובארגון פי  $\approx 2.5$ , הסיבה שבשיטת bcrypt לא רואים את ההאטה על התוקף בשיטה זו היא שכמות סיסמאות בשניה קטן מ-10 ואנחנו הגדרנו את ההאטה להיות מקסימום 10 סיסמאות בשניה.

רואים שcaptcha וtotp מחזיר אותו כמות של זמן כי הם עושים פחות או יותר את אותה פעולה. ניתן לראות שהגנת lock הכי אפקטיבית בbrute force, משום ששיטת התקיפה brute force מנסה הרבה סיסמאות על משתמש יחיד ולכן המשתמש נחסם מהר, והתוקף לא יכול להיכנס לחשבון משום שהחשבון ננעל, וזה חוסם את רוב הניסיונות של התוקף.

(הרצנו את אלגוריתם pepper על שיטת גיבוב SHA-256+salt וקיבלנו תוצאות כמעט זהות להרצה של SHA-256+salt ללא pepper, ולכן הסקנו שההגנה לא מגנה על פריצת סיסמה אלה מגנה על הדאטהבייס מפני תקיפת נתונים ולכן לא יהיה שינוי משמעותי על סיסמאות בשאר שיטות הגיבוב.)

טבלת נתונים של שיטת SHA-256+salt עם ובלי pepper נראה שהזמנים לפריצה דומים

הגנה	הרצה ללא pepper	הרצה עם pepper
none	0.007	0.09
rate	33.133	33.167
lock	Account locked	Account locked
captcha	0.032	0.128
totp	0.028	0.048

## ניתוח ודין:

### מסקנה על שני אלגוריתמי התקיפה:

מבין שני אלגוריתמי התקיפה אלגוריתם spray עדיף לתוקף על brute, brute הוא ינסה את כל הצירופים האפשריים עד שיגיע לסיסמה נכונה, וככל שהסיסמה מורכבת מתווים שונים או ארוכה יותר כך גדלים האפשרויות ויקח לתוקף יותר זמן, שגדל באופן מעריכי. ואילו ב spray מנסים סיסמאות נפוצות שהסיכויים לפרוץ את המערכת איתם גדולים יותר. ניתן לראות זאת בתוצאות של ההרצות: לקח brute בדיוק 340 ניסיונות לפרוץ סיסמא המוגדרת כקלה ובספריי אחרי 340 ניסיונות הוא מצא 4 סיסמאות!

כפי שהובא לעיל הסיסמאות חולקו בקטגוריות בהן- מספרים וסיסמאות נפוצות מאוד- חלש שילוב מספרים ואותיות בינוני, שילוב עם תווים מיוחדים- קשה. מפאת מגבלות הזמן ומס הנסיונות סיסמאות בינוניות וחזקות לא היו נמצאות בפחות מ-1,000,000 ניסיונות הקומבינטוריקה שנגיע אליה ב-1,000,000 עדיין תהיה סיסמא הנחשבת כחלשה והיא מספרים בלבד באורך  $6^{10}$ . (כפי שניתן לראות בהרצה על משתמש עם סיסמא של מספרים 111111 באורך 7) קיימים 72 סוגי תווים (מספרים, אותיות גדולות וקטנות ותווים מיוחדים) כך שכל שאורך הסיסמה גדלה כך גדלים גם הנסיונות פי 72 עבור כל תו נוסף! זמן משוער לפיצוח סיסמא בינונית סיסמא abc123: sha יקח 2.25 שניות לbcrypt יקח 273 שעות וargon יקח 56.5 שעות. זמן משוער לפיצוח סיסמא חזקה: בכל שלושת שיטות הגיבוב ייקח יותר מ-100 שנה.

ניתן לראות כי אלגוריתם sha-256 לא ממש חוסם שום תקיפה ולא ממולץ למטרה זו. המהירות בה ניתן להכניס סיסמאות מאפשרת לתוקף לנסות סיסמאות רבות ללא בעיה. לעומת זאת bcrypt וargon2id יותר טובים למטרת הגנה על סיסמאות במקום 1,000,000 ניסיונות התוקף נבלם ל 12,846 ו 109,427 ניסיונות. אלגוריתמים bcrypt ו-Argon2id הופכים כל ניסיון התחברות ליקר מבחינה חישובית. ב-bcrypt, למשל, 50k ניסיונות מצטמצמים ל-18,300 בלבד אך זמן הריצה מגיע לשעתיים. שיטת הגיבוב האיטית ביותר (עם הנתונים שהוגדרו בפרוייקט) היא bcrypt.

מנגנון lock התגלה כגורם המשמעותי ביותר, כאשר הוא חסם את מתקפת brute. וכן ב spray הקטין את מספר הסיסמאות שנמצאו. ב SHA256: brute חסם 49,990 מתוך 50k מניסיונות התקיפה. ב spray הוריד את מס הצלחות התקיפה ביותר מחצי. ב Argon2id: brute חסם 49,990 מתוך 50k מניסיונות התקיפה. ב spray הוריד את מס הצלחות התקיפה ביותר מחצי גם. ואילו ב bcrypt הגנה זאת לא הצליחה לבלום את התוקף מפני שמספר הנסיונות לשניה הוא קטן מ-10 (חסמו רק אם מס ניסיונות לשניה עולה יותר מ-10) ולכן ההגנה לא פועלת.

המסקנה המרכזית היא ששילוב של bcrypt עם lock מספק את רמת ההגנה האפקטיבית ביותר

לאלגוריתם brute force: בעוד שאם היינו מערכת אמיתית נעילת החשבון לא הייתה מוגבלת בזמן אלה בבקשת גישה מהמערכת. בכך שילוב זה הופך את המתקפות לכמעט בלתי אפשריות.

### דיון בתוקף הניסוי ומגבלותיו

- מספר משתמשים קטן: 30 משתמשים בלבד, ואינו מייצג סביבת תקיפה אמיתית.
- כלי תקיפה בסיסיים: שימוש ב-`itertools.product` ללא אופטימיזציה
- גישה למשתמש אחד בכל פעם לא מייצגת שרת אמיתי בו ניתן לגשת לכמה חשבונות ועליהם לנסות לתקוף בו זמנית.
- זמן ניסוי מוגבל: כל תצורה נבדקה עד 2 שעות או 50k נסיונות, בעוד שתוקף אמיתי עשוי לפעול לאורך זמן רב יותר.

### מקורות אי-דיוק אפשריים

- שונות חישובית: ביצועי `bcrypt` ו-`Argon2id` מושפעים משימוש רב ב-CPU ותחרות תהליכים.
- דיוק זמנים מוגבל: ערך `Time: 0.0s` ב-`SHA-256` עשוי לנבוע מעיגול מדידה.
- רעש סביבתי: הרצות חוזרות על אותו מעבד השפיעו על זמני החישוב. וכן הרצות על מחשבים שונים עם גודל מעבדים שונה.

### נסיונות כושלים:

בשלב כתיבת הדוח מצאנו כי הנתונים לא הסתדרו: הרצה על 50,000 נסיונות הוציאה רק 7000 או פחות נסיונות תקיפה בשעתיים באלגוריתם `Argon2id`. לאחר חקירה עמוקה הבנו את הבעיה. בכל נסיון תקיפה התבצע פתיחת הלוג כתיבה וסגירה מה שלקח משאבים רבים וזמן CPU כך שהוריד `Argon2id` ו-`bcrypt` היו צריכים לריב על זמן מעבד וחיכו ל-`context switch` עם `io`. - מה שגרם להורדה משמעותית של מספר הנסיונות. לכן פתרנו את הבעיה בכך שהשתמשנו ב-`buffer` שכתב לקובץ רק כל 500 נסיונות. ובכך צמצמנו זמן מעבד והגענו לנתונים אמניים.

כשראינו את הנתון של מס' נסיונות לשניה ב-`lock` הוא היה גדול משמעותית מנסיונות ללא הגנות. בהתחלה המידע היה לא מובן אך הבנו שבגלל שהוא חוסם נסיונות רבים ולא עובר את הגיבוב של הסיסמא יש לו יותר זמן לבצע יותר נסיונות. `latency` הממוצע של `argon` הוא 65.8 ואילו בדיקה אם משתמש חסום לא לוקח כ"כ הרבה.

### שיקולים אתיים

עבדנו על הפרויקט בסביבה לימודית, יצרנו דאטה בייס בעצמנו (`users1.json`, `users2.json`, `users3.json`) הכוללים משתמשים וסיסמאות שהומצאו על ידינו, ללא כל שימוש בנתונים אמיתיים או גישה למערכות פעילות. קוד התקיפה (`attack.py`) פועל על בסיס נתונים מקומי בלבד וללא חיבור לרשת, לא נעשה ניסיון תקיפה על מקומות חיצוניים, תוך עמידה בעקרונות האתיקה המחקרית וקידום אבטחת המרחב המקוון.



## **מסקנות והמלצות עתידיות**

### **מסקנות מרכזיות**

הפרויקט הוכיח לנו כי שילוב של אלגוריתם גיבוב איטי (Argon2id) יחד עם מנגנון נעילת חשבון מספק הגנה אפקטיבית במיוחד מפני תקיפות. התקיפה נחסמה לחלוטין! בנוסף הגנת rate בלם את התוקף אך לא חסם אותו לחלוטין. תקיפות Password Spraying נבלמו חלקית, עם צמצום ההצלחות בכמעט חצי. בנוסף, הוסבר כי אורך הסיסמה הוא גורם קריטי: הוספת תו אחד (מ-6 ל-7) מגדילה את עמידות הסיסמה פי 72.

### **לקחי פיתוח מהפרויקט**

במהלך הפיתוח הופקו מספר תובנות הנדסיות חשובות:

- חלוקת ההרצה לשלבים, קודם ריצה על מספרים בלבד שזה יותר נפוץ בסיסמאות ואחכ שילוב של מספרים עם אותיות וכן הלאה.
- כמו כן חלוקת ההרצה לקבצים, הפרדה בין attack.py ל-defence.py אפשרה בדיקה והשוואה עצמאית של מנגנוני הגנה.
- הגדרת משתנים סופיים (global\_time\_limit, max\_attempts\_per\_pass) למניעת ריצות אינסופיות.
- לוגים מפורטים בפורמט JSON Lines אפשרו ניתוח מדויק של זמני תגובה, עומסים וצריכת משאבים.

### **המלצות לעתיד**

להרחבת המחקר מומלץ:

- לבצע בדיקות בקנה מידה גדול באמצעות GPU וענן (AWS/GCP) עם אלפי משתמשים.
  - לשלב מנגנוני הגנה מבוססי למידת מכונה לזיהוי סוגים רבים יותר של סיסמאות.
  - לאכוף מדיניות סיסמאות מחמירה (אורך +12) שמפאת חוסר הזמן הגבלנו ל-6-10 תווים.
- לסיכום, הפרויקט מספק בסיס מוצק לתכנון מערכות אימות מאובטחות, כאשר שילוב של Argon2id, נעילת חשבון וסיסמאות באורך 10 תווים ומעלה מהווה את רף ההגנה היעיל ביותר תחת הנתונים שהוגדרו בפרויקט. שינוי של ברירת המחדל של bcrypt ל-argon ישנו את תוצאות המחקר וכן שינוי של נתוני lock והrate ישנו את תוצאות המחקר גם כן.

## **מקורות**

### **מקורות תיאורטיים**

1. [Brute Force Attack - Wikipedia](#)
  2. [SHA-2 Algorithm - Wikipedia](#)
  3. <https://www.argon2.com>
  4. <https://www.openwall.com/bcrypt>
  5. [Most Common Passwords - Geektime Israel 20](#)
- כלי פיתוח ומקורות קוד:**
6. <https://docs.python.org/3/library/itertools.html#itertools.product>
  7. <https://github.com/pyauth/pyotp>
  8. [passlib Hashing Library - Documentation](#)

## **נספחים:**

מצורף בתיקייה קבצי לוגים.