

דו"ח מחקר – פרויקט מבוא לאבטחת המרחב המקוון

מבוא:

אבטחת המרחב המקוון תלולה באופן ישיר בחזוקם ביעילותם של מנגנוני האימות. אימות מבודס סיסמאות נותר האמצעי המרכזי להגנה על פרטיות המשתמשים בראשת. הגישה הבסיסית לאימות מתבססת על הזרנת שם משתמש וסיסמה, המהווים את הוכחה לבועלות על החשבון. סיסמאות אלה מאבטחות מידע ברמות חשיבות שונות, החל מרשות חברותיות ועד מערכות פיננסיות לדוגמא בנקים. لكن, שמירה על שלמות הסיסמאות ומונעת גישה בלתי מורשית לאתרים הוא חלק גדול ומשמעותי באבטחת המרחב המקוון.

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

בפרויקט זה נתמקד בשני סוגי התקיפות העיקריים: 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, ונמדד אוטם אותם בצורה כמותית – כמו זמן עד לפריצה מספר נסיניות נדרשים. בנוסף, נבדוק איך חזק הסיסמה משפיע על מהירות הפריצה לחשבון.

סוגי התקיפה – הסבר:

Brute Force – ניסוי של כל אוסף הסיסמאות האפשריים על משתמש מסוים יחיד מקורה

Password Spraying – התקיפה בה מנוטים 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 נעשו ע"י אלגוריתם שמחציא את הקלט לכמה בלוקים ועושה פעולות על סיביות על כל בלוק באופן חוזר עד שmagiu 256 ביטים. מתרבע במהירות מה שנណן לתוכף יכולות ניסוי סיסמאות רבים גם כן. מקורה

Argon2 – אלגוריתם שנועד להקשوت על התקיפות force-force-brue מוסיף מלח אקריאי לסיסמא ואז עושה hash האלגוריתם זהה עובד באיטיות בכוונה ומשתמש בהרבה זיכרון מחשב בעט חישוב hash

bcrypt – אלגוריתם שנועד להקשות על התקיפות force-force-brue מוסיף מלח אקריאי לסיסמא ואז עושה hash האלגוריתם זהה עובד באיטיות בכוונה כדי להאט התקיפות בשונה מאלגוריתם Aragon2 הוא לא משתמש גם במשאבי זיכרון גדולים מאד.

:מנגנוני הגנה - הסבר

Rate Limit – מנגן הגנה בו המערכת מזהה ניסוי כניסה רב מדי' ממוקור מסוים -
ואז חוסמת לו את האפשרות לכניתה למערכת לזמן מסוים

נעילת חשבון – מנגן הגנה בו המערכת מזהה ניסוי כניסה למשתמש מסוים מספר רב של פעמים המערכת תחסום את כניסה המשתמש הספציפי זהה למערכת לזמן מסוים

TOTP - קוד חד פעמי שימושו כל כמה שני' והוא שלב נוסף בזיהוי

CAPTCHA - בודק שהמשתמש הוא אדם ע"י יצירת שדה קלט ורק תשובה נכונה מאפשרת המשיך

Pepper - הוספת סוד כללי קבוע לסיסמה בדעתה ב"י

תצורה ניסوية מיתולוגיה:

הניסיונות בוצעו על שלוש שיטות שונות לאחסון סיסמות

.GROUP_SEED:(username:password:salt)SHA-256: bcrypt: הגדרנו את העלות להיות 12 לפי דרישות המטלה.

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

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

חלוקת הקבצים עברו כל שיטה גיבוב אפשרית להשוות בין תוצאות התקيفות תחת אותן תנאים עומס, כאשר ההבדל היחיד הוא אלגוריתם הגיבוב שבו נעשה שימוש.

חלשה: "בדרך כלל אם לא מגבלים משתמשים בבחירה הסיסמה, הם נוטים לבחור מילים או ביטויים נפוצים, חלקים ממשם או תארכים ידועים." שמננו מילים נפוצות וסיסמות של מספרים בלבד מצביע בינם: סיסמה שמורכבת משלוב של מילים נפוצות ומספרים.

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

לכל משתמש נשמרו הנתונים הבאים: שם משתמש (username), סיסמה לאחר גיבוב (hashed password) (hash_mode), קטגורית חזק הסיסמה (strength_category), שיטת הגיבוב (hash_mode) על כל אחת משלוטות אחסון הסיסמות נבדקו חמש רמות הגנה שונות: ללא הגנות- baseline הגבלת קצב ניסיונות - Rate Limiting Account Lockout CAPTCHA token . מבוסס לכל משתמש TOTP ואימונות דו-שלבי . בנוסף הוספנו בדיקה בה מוסיפים לכל הסיסמות ערך סוד גלובלי לפני הגיבוב הוא נשמר בנפרד מהדעתה ב"י - הוספת pepper.

בכל תרחיש, ניסיונות ההתחברות נרשמו לקובץ לוג בפורמט Lines JSON. כל רשומת לוג כוללת את השדות: חותמת זמן, GROUP_SEED, שם משתמש, שיטת גיבוב, סוג הגנה, תוצאת הניסוי, latency, CPU.

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

תקיפות מסוג Brute Force בוצעו כבדיקה כל אפשרויות הקומבינטוריקה (ע"י שימוש בספרייה iterator python) על משתמש יחיד. קודם בדיקת סיסמאות באורך 6 תוך כדי הרחבה הCharset 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 התקיפות Brute Force תוכננה כך שברגע שסיסמה של משתמש מסוים מתגללה, ההתקפה עוברת למשתמש הבא. גישה זו נועדה לדמות התנהלות של תוקף ריאלי, שמנצל את מגבלת הניסיונות על פני חשבונות שונים ולא מתמקד רק בחשבון אחד.

עבור כל שלבי של שיטת אחסון סיסמאות ומנגנון הגנה, הורץ בקצב התקיפת Password Spraying ולחדריה התקipa Brute Force.

לכל הריצה נוצר קובץ לוג נפרד, לאחר סיום כל הרצות, קובצי הלוג שימשו לחישוב מדדים כמוותים, כגון מספר ניסיונות עד הצלחה, מספר החשבונות שנפרצו וזמן הריצה, וכן לניתוח איכוטי של הייעילות של מנגמוני ההגנה ואלגוריתמי הגיבוב.

חלוקתנו את הרצאה ל 2 שלבים עיקריים:
שלב ראשון - הרצינו את קוד ההתקפה עם שלושת שיטות הגיבוב (SHA256, bcrypt, Argon2id) ללא שום הגנות.
שלב שני - הריצה שנייה עם כל מנגנון הגנה בנפרד.

בנינו 3 קובצי הריצה:

HashTable_generator.py: קובץ ייצור הסיסמאות בטבלאות ויצירת קובץ json למשתמשים.

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

defence.py: קובץ ההגנות.
ההריצה מתבצעת בattack והוא קוראת לקובץ של ההגנות.

** ב כדי להריץ ולקיים נתונים תואמים לניטונים שלנו צריך להריץ את קובץ hashTable_generator - יותר קובץ json של המשתמשים. ואחריו להריץ את defence.py - מרים את כל השילובים attack.

תוצאות:

סיכום password spraying ללא הגנת: 50000 נסיעות תקיפה:

מנגנון גיבוב	הגנה	מספר נסיעות	זמן הצלחות מס	זמן ביצוע כולל (ש')	ניסיעות לשניה	זמן עד הצלחה שנייה
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 הרצת brute force ללא הגנה:

משתמש 1 : 123456

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

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

משתמש 5 :

הרצה brute force:

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

163.152	2.08	340	totp	
13.058	26.03	340	none	Argon2id
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 salt+SHA-256 ולשיטת גיבוב Argon2id הגנת rate limiting מאטת את זמן התקיפה באופן משמעותי, פי ≈ 4,734 ביחס לשיטת גיבוב של SHA-256 ובריגון פי ≈ 2.5, הסיבה שבשיטת bcrypt לא רואים את ההאטה על התקוף בשיטה זו היא שcmcות סיסמאות בשניה קטן מ 10 ואנחנו הגדכנו את ההאטה להיות מקסימום 10 סיסמאות בשניה.

רואים שהשיטות captcha וtotp מחזיראותו כמות של זמן כי הם עושים פחות או יותר את אותה פעולה. ניתן לראות שהגנת lock הינה אפקטיבית ביחס לbcrypt, משום ששיטת התקיפה brute force מנסה הרבה סיסמאות על משתמש יחיד ולכן המשתמש נחסם מהר, והתקוף לא יכול להיכנס לחשבון משום שהחישבו נגעל, וזה חוסם את רוב הניסיונות של התקוף.

(הריצנו את אלגוריתם pepper על שיטת הגיבוב salt+SHA-256+salt וקיבliśmy תוצאות כמעט זהות להריצה של SHA-256+salt ללא pepper, ולכן הסקנו שההגנה לא מגנה על פריצת סיסמה אלה מגנה על הדטההבייס מפני התקיפות נתונים ולכן לא יהיה שינוי משמעותי על סיסמאות בשאר שיטות הגיבוב.)

טבלת נתונים של שיטת salt+SHA-256+salt עם ובלי pepper נראה שהזמן נרחב בפריצה דומימית

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

ניתוח ודינן:

מסקנה על שני אלגורתמי התקיפה:

בין שני אלגורתמי 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 עליוני. ואילו spray מנוטים סיסמות נפוצות שהסיסמים לפרוץ את המערכת אותם גדולים יותר. ניתן לראות זאת בתוצאות של ההרצאות: לדוגמה brute בדיק 340 ניסיונות לפרוץ סיסמה המוגדרת כקלה ובספרי אחריו 340 ניסיונות הוא מצא 4 סיסמות!

כפי שהובא לעיל הסיסמות חולקו בקטגוריות בהן- מספרים וסיסמות נפוצות מאוד- חלשים שלוב מספרים ואותיות בלבד, שלוב עם תווים מיוחדים- קשה. מפהת מגבלות הזמן ומספר הסיסמות בינוין וחזקות לא היו נמצאות לפחות מ-1,000,000,000,000 ניסיונות הקומבינטוריקה שנגיעה אליה ב-1,000,000 עדין תהיה סיסמה הנחשבת כחלשה והוא מספרים בלבד באורך 6 (10^{18}). (כפי שניתן לראות בהרצה על משתמש עם סיסמא של מספרים 11111111 באורך 7 קיימים 72 סוגים (מספרים, אותיות גדולות וקטנות ותווים מיוחדים) כך שכך שארך הסיסמה גדלה כך גדלים גם הניסיונות פי 72 עבור כל TWO נספחים!
זמן משוער לפיצוח סיסמא בינוין סיסמא abc123: brute יקח 2.25 שניות לbcrypt יקח 273 שעות לשונו argon יקח כ-56.5 שעות.
זמן משוער לפיצוח סיסמא חזקה: בכל שלושת שיטות הגיבוב יקח יותר מ-100 שנה.

ניתן לראות כי אלגוריתם sha-256 לא ממש חומר שום התקיפה ולא ממלץ למטרה זו. המהירות בה ניתן להכניס סיסמות מאפשרת לתוקף לנסיבות סיסמות רבות ללא בעיה. לעומת זאת bcrypt וargon2 יותר טובים למטרת הגנה על סיסמות במקומן 1,000,000 ניסיונות התוקף נבלם ל-12,846 | 109,427 ניסיונות. אלגוריתמי bcrypt ו-argon2 הופכים כל ניסיון התחרבות ליקיר מבחינה חישובית. למשל, bcrypt, לשון, 50k ניסיונות מצטמצמים ל-300,18 בלבד אך זמן הריצה מגיעה לשעות. שיטת הגיבוב האיטית ביותר (עם הנתונים שהוגדרו בפרק) היא bcrypt.

מנגן lock התגלה לגרום המשמעותי ביותר, כאשר הוא חוסם את מתקפת brute. וכן bcrypt הקטין את מספר הסיסמות שנמצאו.SHA256: brute חום 49,990 מיותר k 50 מנייסיונות התקיפה. bcrypt הוריד את מס הצלחות התקיפה ביותר מחצי. Argon2: brute חום 49,990 מיותר k 50 מנייסיונות התקיפה. bcrypt הוריד את מס הצלחות התקיפה ביותר מחצי גם. ואילו bcrypt הגנה זאת לא הצלחה לבולם את התוקף מפני שמספר הניסיונות לשניה הוא קטן מ-10 (חסמנו 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 נעלית החשבון לא הייתה מוגבלת בזמן אלה בבקשת גישה מהמערכת. בכך שילוב זה הופך את המתפקידות למעט בלתי אפשריות.

דין בתוקף הניתן ומגבלותיו

- מספר משתמשים קטן: 30 משתמשים בלבד, ואין מיצג סביבת תקיפה אמיתית.
- כל תקיפה בסיסיים: שימוש ב-product.itertools.
- גישה למשתמש אחד בכל פעם לא מייצגת שרת אמיתית בו ניתן לגשת לכמה חשבונות ועליהם לנסות לתוקוף בו זמן.
- זמן ניסוי מוגבל: כל תצורה נבדקה עד 2 שעות או 50 נסיניות, בעוד שתוקף אמיתי עשוי לפעול לאחר זמן רב יותר.

מקורות אידיאק אפשריים

- שונות חישובית: ביצועי bcrypt ו-Argon2id מושפעים משימוש רב ב-CPU ותחרות תהליכיים.
- דיק זמינים מוגבל: ערך 0.0s Time ב-SHA-256.
- רעש סביבתי: הרצות חזירות על אותו מעבד השפיעו על זמן החישוב. וכן הרצות על מחשבים שונים עם גודל מעבדים שונה.

נסיניות כשלים:

בשלב כתיבת הדוח מצאנו כי הנתונים לא הסתדרו: הריצה על 50,000 נסיניות הוצאה רק 7000 או פחות נסיניות תקיפה בשעתים באlgorthm Argon2id. לאחר חקירה عمוקה הבנו את הבעיה. בכל נסיכון תקיפה התבצע פתיחה הלוג כתיבה וסירה מה שלקוח משאים רבים וזמן CPU כך שהוריד Argon2id bcrypt היו צריכים לרכיב על זמן מעבד וחיכו לswitch context עם זו. - מה שגרם להורדה משמעותית של מספר הנסיניות. לכן פתרנו את הבעיה בכך שהשתמשנו בז'רף שכתב לקובץ רק כל 500 נסיניות. ובכך צמצמנו זמן מעבד והגענו לנתונים אמינים.

כשראינו את הנתון של מס' נסיניות לשניה בclock הוא יהיה גדול משמעותית מנסיניות ללא הגנות. בהתחלה המידע היה לא מובן אך הבנו שבגלל שהוא חוסם נסיניות רבים ולא עבר את הבדיקה הסיסמה יש לו יותר זמן לבצע יותר נסיניות. הערך letancy הממוצע של argon הוא 65.8 ואילו בדיקה אם משתמש חסום לא לוקח כ"כ הרבה.

סיכום אטיים

עבדנו על הפרויקט בסביבה לימודית, יצרנו דאטא ב-YAML בעצמנו (users1.json, users2.json, users3.json) הכוללים משתמשים וסיסמות שהומצאו על ידיינו, ללא כל שימוש נתונים אמיתיים או גישה למערכות פעילות. קוד התקיפה (attack.py) פועל על בסיס נתונים מקומי בלבד ולא חיבור לרשת, לא נעשה ניסיון התקיפה על מקומות חיצוניים, תוך עמידה בעקרונות האתיקה המחקרית וקידום אבטחת המרחב המקומי.

مسקנות והמלצות עתידיות

مسקנות מרכזיות

הפרויקט הוכח לנו כי שילוב של אלגוריתם גיבוב איטי (Argon2id) יחד עם מנגןן נעילת חשבון מספק הגנה אפקטיבית במיוחד מפני התקיפות. התקיפה נחסמה לחלווטן! בנוסף הגנת `rate` בلم את התקוף אך לא חסם אותו לחלווטן.

תקיפות Password Spraying נבלמו חלקית, עם צמצום ההצלחות במעט חצי. בנוסף, הוסבר כי אורך הסיסמה הוא גורם קרייטי: הוספה זו אחד (מ-6 ל-7) מגדילה את עמידות הסיסמה פי 72.

לключи 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` מאפשרת בדיקה והשוואה עצמאית של מנגןנו הגנה.
- הגדרת משתנים סופיים (`global_time_limit`, `max_attempts_per_pass`) למניעת ריצות אינסופיות.
- לוגים מפורטים בפורמט Lines JSON אפשרו ניתוח מדויק של זמני תגובה, עומסים וצריכת משאביים.

המלצות לעתיד

להרחבת המחקר מומלץ:

- לבצע בדיקות בקנה מידה גדול באמצעות GPU וענן (AWS/GCP) עם אלפי משתמשים.
- לשלב מנגןנו הגנה מבוססי למידת מכונה לזרחי סוגים רבים יותר של סיסמאות.
- לאכוף מדיניות סיסמאות חמירה (אורך 12+) שמספר חוסר הזמן הגבלנו ל-10-6 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 הייעיל ביותר תחת הנתונים שהוגדרו בפרויקט. שינוי של ברירת המחדל של `bcrypt` ו-`argon` ישנו את תוכניות המחקר וכן שינוי של נתוני `lock` וה`rate` ישנו את תוכניות המחקר גם כן.

מקורות

מקורות תיאורתיים

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

נספחים:

מצורף בתיקייה קבצי לוגים.