כללי

אבני היסוד באבטחה - שלמות, זמינות, סודיות.

Accountability -לנתר כל פעולה במערכת ע"י לוגים כך שנוכל לאתר מי ביצע כל פעולה באופן ייחודי. (עוזר לנו בהתאוששות, מציאת באגים, מציאת גורם אחראי)

סוגי התקפות – פאסיבי(האזנה), אקטיבי(הזרקה), איש מבחוץ(ללא כל גישה למערכת), איש מבפנים(בעל גישה חלקית\מלאה למערכת)

דרכי תקיפה - עקיפת אימות(ניחוש סיסמא), הנדסה חברתית, שימוש בטעות אנוש, חורים בתוכנה, שימוש בכלים זדוניים, הסקת נתונים ממידע אליו יש לנו גישה.

שלבי ההגנה - מניעה, זיהוי, התאוששות.

מהם הסכנות - גישה למידע רגיש, הפרעה לפעולת המערכת, שינוי נתונים רגישים (מצב חשבון בנק, ציונים וכו')

עקרונות – לשלוט ברמה ממוקדת ככל האפשר, לא לתת הרשאות יותר ממה שצריך, למדר את המערכת כך שהכח יפוזר, כלום לא מותר אלא אם הגדרנו, מנגנון מתן הרשאות, מה לעשות במקרה של חפיפה?

AAA – זיהוי המשתמש – אימות וזיהוי של ישות במערכת, הרשאה – הענקת אישור (הרשאה) לישות במערכת לגשת למשאב וניטור – סקירה ובדיקה עצמאית של רשומות המערכת לזיהוי פריצות באבטחה ולהמלצה על שינויים במדיניות.

## הצפנות

**אסימטרי** – מפתח להצפנה ומפתח שונה לפענוח

\*מפתח גדול יותר ישפר האבטחה אך יעלה לנו בזמן ריצה.

**סימטרי** – אותו מפתח גם להצפנה וגם לפענוח לכולם:

**Stream** – נעבור על הטקסט לפי Byte אחד אחד ונצפין כל אחד מהם בנפרד ונשרשר באמצעות XOR למה שכבר הצפנו. (מהיר יותר וצורך פחות כתיבה של קוד מאשר block)

**Block** – נשבור את הטקסט לקוביות בגודל קבוע ונצפין כל בלוק באמצעות מפתח (ניתן להשתמש באותו מפתח).

**הערות:** בלוק גדול יותר נותן אבטחה טובה יותר אך מאט את האלגוריתם.

## דרישות ממפתח ציבורי

**1.**ייצור קל של זוגות

**2.**חישוב קל של פענוח והצפנה של הודעה באמצעות המפתח הציבורי והפרטי.

**3.**שלא יהיה פשוט לחשב את המפתח השני במידה ויש לנו את המפתח הראשון

**4.**לא יעיל אם כל מפתח יכול לשמש כל תפקיד./

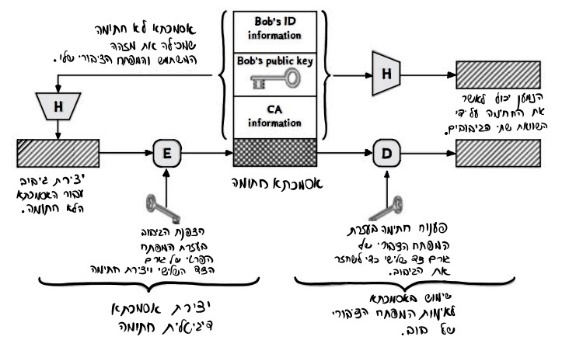
**5.**קושי לפענח את ההודעה עם המפתח הציבורי ללא הפרטי

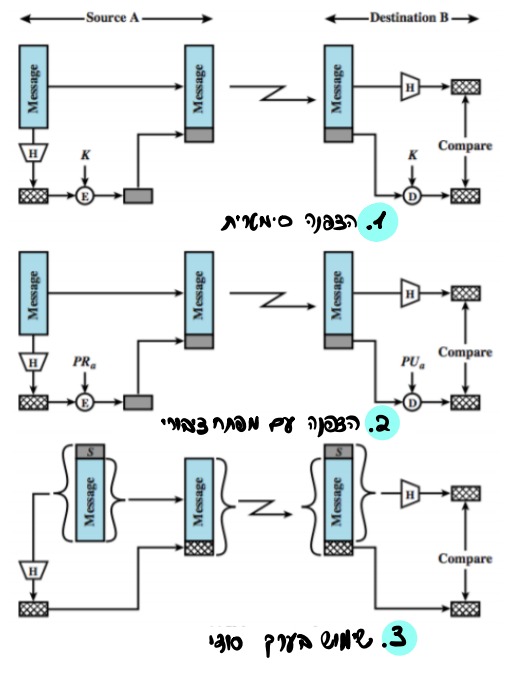
אלגוריתמי הצפנה סימטריים -DES, AES, TripleDES

א סימטריים – RSA, Diffie-Helman, DigitalSig .

**בעיות במפתח ציבורי:** איך נוכל להבטיח כי מפתח ציבורי של מישהו אכן הגיע ממנו ולא מגורם המתחזה אליו? יהיה לנו צד שלישי אשר כולנו בוטחים בו אשר ייתן "אסמכתה" כי המפתח אכן הגיע ממי שהוא טוען שהוא הגיע.

כעת מפתח ציבורי יהיה "מזהה משתמש + מפתח ציבורי + חתימה של הגורם השלישי".

****

אימות לוודא מקור ותוכן, לא חייב להצפין את ההודעה כדי לבצע אימות. (**3 שיטות** - מפתח סימטרי, מפתח ציבורי, ערך סודי)  


## MAC – Message authentication code

שלמות הודעה - משתמש במפתח סודי על מנת לייצר בלוק קטן של מידע המייצג את ההודעה, בלוק זה נשלח ביחד עם המידע ולאחר מכן ישמש לאימות המידע. ניתן גם להוסיף מספר סידורי להודעה ובכך לוודא שהודעות הגיעו בסדר נכון ולחסום מתקפות ריפליי.

One way Hash - אלטרנטיבה ל-MAC, נעביר את ההודעה דרך פונקציית HASH ונייצר בלוק של מידע מיצג, השינוי המהותי לעומת ה-MAC זה שפה אין לנו שימוש במפתח סודי על מנת ליצר את הבלוק.

## דרישות מפונקציית HASH

**1.**ניתן להריץ על כל קלט (לא משנה האורך)

**2.**הפלט הוא בגודל קבוע 🡨 כמה שהפלט יותר קצר כך יהיה יותר התנגשויות ופחות אופציות, למשל אם הפלט הוא באורך 3 ויש לנו בכל תו 256 אופציות זה סך הכל אפשרויות.

**3.**קל לחשב את הפלט עבור כל קלט

4.שהפונקציה לא תחזיר את הערך שלה (בלתי אפשרי לגלות את ההודעה מפונקציית הגיבוב)

**5.**weak collision- לכל בלוק X יהיה קשה למצוא X≠**Y** כזה שהגיבוב שלהם יהיה זהה-לתוקף יהיה קשה למצוא Y כזה

**6.** strong collision- מראש קשה למצוא שתי הודעות שההאש שלהן זהה.

**שתי גישות של Hash**

**Cryptanalysis** – שיטה המסתמכת על טבע האלג' וידע מוקדם על מאפייני הטקסט ע"מ לנצל חורים ולהסיק מסקנות לגבי הטקסט או לגבי המפתח.

**Brute Force** – ריצה על כל האפשרויות

## DS – חתימה דיגיטאלית

**\***נותן לנו אפשרות לאימות זהות המקור והבטחת שלמות המסמך הדיגיטלי.

**\***השולח יחתום עם המפתח הפרטי וכך נוכל לפענח רק עם הציבורי שלו – נבטיח את זהות השולח. אם צד שלישי ינסה לשנות את תוכן ההודעה פונקציית האימות תחזיר ערך שונה ונדחה את ההודעה.

**\***ההודעה אינה בהכרח מוצפנת אך ניתן להצפין גם אותה.

**\***חתימה דיגיטלית אינה מספיקה על מנת למנוע התכחשות – על מנת למנוע התכחשות נצטרך להוסיף צד שלישי אליו נשלח את פרטי ההודעה כולל החתימה והוא יוסיף חתימה משלו לאחר אימות השולח.

Digital Envelope- מבטיח כי השולח הוא מי שהוא טוען שהוא ושההודעה תגיע אל הנמען הרצוי בלבד.

**1.**כותבים הודעה

**2.**מצפינים את ההודעה באמצעות איזשהו מפתח סימטרי חד פעמי שאנו מייצרים session key

**3.**מצפינים את המפתח K על ידי המפתח הציבורי של היעד

**4.**שולחים את ההודעה המוצפנת ואת המפתח K

**5.**הנמען יפענח את המפתח K באמצעות המפתח הפרטי שלו **6.**יפענח את ההודעה באמצעות המפתח K המפוענח

## User authentication

אימות משתמש נגזר מ-3 דברים:

**1.** מה יש לך? (תעודה מזהה, כרטיס אשראי, וכו') – עלול להיאבד

**2.**מה אתה יודע? (סיסמא) – עלול לשכוח

**3.**מי אתה? (טביעת אצבע, רשתית וכו') – לא תמיד אמין (false positive etc.)

ניתן להשתמש באחד או שילוב של המאפיינים הללו.

## כרטיס חכם smart-token))

שלושת הקטגוריות אליהן מסווגים פרוטוקולי אימות זיהוי המשמשים לכרטיס:

**1.**סטטים – המשתמש מאמת עצמו לכרטיס ואז הכרטיס מאמת את המשתמש אל מול המחשב.

**2.**יוצר סיסמאות דינאמי – הכרטיס מייצר סיסמא ייחודית כל הזמן מחדש.

**3.**אתגר תגובה – מערכת המחשב מייצרת אתגר, כמו מחרוזת אקראית של מספרים. הכרטיס החכם יוצר תגובה על בסיס האתגר.

## הצפנת מידע מאוחסן

**אמצעים** – רכיב פיזי היושב בין השרת לבין האחסון, תוכנה אשר מצפינה נתונים על המחשב.

**גישות**:

**1.**Back and appliance – הצפנה באמצעות מתווך בין הלקוח לבין ה-storage.

**2.**Library based type encryption – תהליך הצפנה בו אוכל לשחזר עם דרייברים דומים.

**3.**Background laptop & PC data – מצפינים את כל המערכת. תלוי ברגישות ובניהול סיכונים חישוב של זמן ועלות.

## נקודות חולשה של סיסמא:

**1.**האקר עלול להשיג גישה לקובץ הסיסמאות (גם אם מוצפן – התוקף יכול לבדוק סיסמאות מוכרות)

**2.**מתקפה מוכוונת כלפי משתמש ספציפי, מנחש סיסמאות עד שהסיסמא מצליחה

**3.**מתקפת סיסמאות פופולאריות – מנסה סיסמאות נפוצות על הרבה משתמשים שונים.

**4.**התוקף עלול להשיג מידע באופן עקיף על משתמש מסוים על מנת להסיק מהי הסיסמא שלו (לדוגמא תאריך לידה + שם ילד וכו')

**5.**חטיפת עמדת עבודה – מחשב שנשאר פתוח ללא פיקוח.

**6.**המשתמש עלול להשאיר פתק אשר רשום עליו את הסיסמא

**7.**משתמשים נותנים לאחרים את הסיסמא שלהם

**8.**שימוש באותה סיסמא בהמון אתרים שונים

**9.**ציטוט ברשת לסיסמא.

Salt – הוספת "רעש" על מנת להקשות על זיהוי, סיסמאות זהות יראו שונה במאגר(לא נוכל לחפש סיסמא מוכרת במאגר ולשייך לה משתמש), מקשה על הפענוח – צריך להריץ סיסמא + כל salt אפשרי במקום רק סיסמא, אם סיסמא נפרצה במאגר אחד היא לא תהיה שימושית במאגר אחר.

אורך הSalt זניח כי לא טורם לקושי הפענוח לתקוף.

SetUserID – או גם SetGroupID הגברת הרשאות זמנית, suid נותן הרשאות של הבעלים וgid נותן הרשאות של הקבוצה.

Least Privilege – נרצה להעניק למשתמש מינימום הרשאות, רק מה שהכרחי לתפקידו הוא יורשה לעשות.

SuperUser – מעניק את כל ההרשאות של מי שמריץ את התוכנית.

Fine grained specifications – נרצה להיות מסוגלים להגדיר הרשאה למשאב ספציפי (קובץ, רשומה) נרצה לדייק.

Separation of duty – הפרדת סמכויות כדי למנוע מצב שלאדם אחד יהיה את כל השליטה בידו (במקרה שפרצו למשתמש מסוים, לא כל המערכת תיחשף).

פריווילגיות פחותות – לתת כמה שפחות הרשאות כדי למנוע ממשתמשים להגיע לאן שלא צריכים ולהכביד פחות על המערכת, המערכת צריכה להיות יעילה.

Open&close policies – סגורה: הגישה מותרת רק למי שיש הרשאות. פתוחה: הגישה מותרת לכולם חוץ ממי שאסור לו. (למשל חסומים)

מודל BLP – מודל לבקרת גישה

1.No read up – ss property - לא ניתן לקרוא מסמך הדורש הרשאה גבוה יותר.

2. No write down --\* property לא ניתן לכתוב מסמך ולתת לו הרשאה נמוכה יותר.

3. ds property - זה פשוט אומר שזה שיש לך סיווג כלשהו לא בהכרח תהיה לך הרשאה לאותו משאב עם סיווג כזה, אבל אם יש לך הרשאה היא חייבת להופיע במטריצה

גישות

Access control - מניעת גישה לא מאושרת למשאב כלשהו, נרצה למנוע ממשתמשים לא מאושרים לגשת לקבצים וכמו כן נרצה למנוע ממשתמשים אשר יש להם הרשאות מסוימות לגשת אך ורק לקבצים להם אנו רוצים לאפשר גישה.

**מודל גישות:** ייצוג של ההגנה, אכיפת גישה, אפשרות להגדיר את מרחב ההגנה.

Access matrix - מטריצת גישות הבנויה כך שבעמודות נשים את המשאבים ובשורות נשים את המשתמשים. בתוך המטריצה נשים השראות כגון RWE וכ'ו.

**חיסרון** במטריצה גישה -– הרבה משתמשים והרבה אובייקטים יצרו הרבה תאים ריקים בהם לא יהיו הרשאות(מטריצה דלילה).

Access list – רשימת גישה, רשימה של הרשאות המוצמדות לקובץ, בצורה הבאה:

File A 🡪 (Alice, RW->Bob, R)

Capability list - רשימה של משתמשים ולכל משתמש מה מותר לו בצורה הבאה:

Alice 🡪 (File A, RW->File B, R)

## DAC – Discrete access control

## בקרת גישה מבחינה – מדיניות שמבוססת על שיקול דעת, ההרשאות ניתנות בהתאם למשתמש. כל משתמש יכול להעניק את ההרשאות שלו למשתמשים אחרים. נייצג זאת בAccess Matrix או ACL.

## MAC – Mandatory access control

בקרת גישה מחייבת - ניתן תווית לכל משאב ("קריטי, רגיש, וכו') לפי הסיווג של אותו אדם הוא יוכל לגשת למשאב באותה רמת סיווג. הרשאה גורפת – למי שיש הרשאה מסווגת יש לו גישה למשאב מסווג וכו'.

RBAC- דומה לAccess List ומטריצת הגישות רק שכאן נגדיר קבוצות משתמשים אשר יקבלו הרשאות מסוימות ולאחר מכן נשייך משתמשים לקבוצות.

3 סוגים: היררכי (יש חפיפה), אילוצים (אין חפיפה) ושילוב (יתרון – חלוקת תפקידים מוגדרת, חסרון – אין גמישות).

3 סוגי אילוצים: **1.** Mutually exclusive roles תפקידים שהמשתמש יכול להחזיק רק בתפקיד אחד. המטרה למנוע תפקידים חופפים וכן מרחב סמכויות רחב מידי. הרשאה תהיה שייכת רק לתפקיד אחד.

**2**.cardinality עוצמת הקבוצה – הגדרת מספר מקסימאלי של משתמשים שיכולים להחזיק בתפקיד נתון (למשל ראש פרויקט יחיד) נועד להפחית סיכונים.

**3.**Prerequisite משתמש יכול לקבל תפקיד רק אם כבר בעל תפקיד ספציפי אחר(התקדמות היררכית).

הגבלות – סטטית: הגבלה מתחילת המערכת עד לשינוי הבא דינמית: הגבלה פר הרצה, כל הרצה ההרשאות משתנות.

Protection domain - *תחומי הגנה, הגישה נועדה להיות יותר גמישה ולאפשר למעשה להגדיר קבוצות של אובייקטים עם הרשאות עליהם ואת סוג ההרשאות. לא מגדירים עבור יוזר ספציפי.*

מודל בלום – סדרה של K פונקציות HASH בלתי תלויות כך שכל אחת ממפה סיסמא לערך מסוים בטבלה, כל סיסמא חדשה במאגר עוברת חישוב בסדרת הפונקציות ומדליקה ביטים בהתאם לפלט הפונקציות, אם כל הביטים דלוקים אנו שוללים את הסיסמא, אם כמות הפונקציות גדולה יחסית לגודל הווקטור נחזיר הרבה false positive.

גיבוב LM – חסרונות

**1.**הגבלה ל14 תווים לסיסמא

**2.**בפחות מ14 תווים, מרפדים בNull-ים

**3.** 7 ועוד 7 תווים מבטל את ה14 והופך ל8

**4.**הכל באותיות גדולות וזה מקטין אפשרויות לסיסמא

## הגנה מפני brute force

**1.**לא לאפשר למשתמש להכניס סיסמאות נפוצות, קצרות, פשוטות לניחוש.

**2.**להריץ מנגנוני פריצה על ידי המערכת אשר תנסה למצוא סיסמאות רגישות ולהתריע למשתמש – מודל מרקוב מבוסס מודל הסתברות המייצר סיסמאות קלות או בלום פילטר (בזמן הרשמה).(**proactive password checking**)

Password cracker-מילון גדול עם סיסמאות "רעות". משתמש בוחר סיסמא-בדיקה אל מול המאגר ה"רע".

2 בעיות – יעילות דורשת מקום רב. זמן חיפוש יקר.

# מתקפות

התקפת סיסמאות - מקוון: יתרון - מתבצע בזמן אמת, לא צריך לנחש salt. ניתן להאזין ולפענח סיסמאות.

חיסרון - לוקח זמן לקבלת תשובה מהשרת, איטי מאוד.

לא מקוון: יתרון – ניתן לרוץ על כל הסיסמאות ולעשות ברוט פורס על הקובץ. חיסרון – קיים salt שמקשה על הפענוח.

Denials of service -DOS - התקפות על מנת למנוע שירות, למשל העמסת שרת כך שלא יוכל להיענות לבקשות לקוח.

Replay – הקלטה של הודעות על ידי ציטוט ושליחה חוזרת (לא חייב לדעת מה התוכן).

התקפת מילון – נפתח מילון גדול של סיסמאות נפוצות וננסה כל סיסמא.

התקפת קשת בענן - לחשב מראש את כל הסיסמאות האפשריות עם ה saltובסוף מבצעים התאמות. חוסך זמן אך תופס מקום – לא אפשרי כאשר יש המון אפשרויות.

# DB

נרצה להגביל גישות על יצירה, הכנסה, מחיקה, עדכון, קריאה, כתיבה של מידע ברמות שונות – מאגר נתונים, טבלאות ספציפיות, עמודות ושורות. כמו כן נרצה להגביל constrains.קשה לחסום מבלי לתקוע את המערכת - צריכה להיות גמישה, יש להבין את השפה בצורה מקסימלית על מנת להתגונן, לרוב יש DBA שלא תמיד יש לו מושג באבטחה בבסיסי נתונים, ריבוי פלטפורמות – קשה להתמודד מערכות הפעלה שונות.

סוגי DB – יש לנו DB מסוג 1.סטטיסטי טהור ומי שיש לו הרשאה אז יכול לגשת לבסיס נתונים כולו. 2.שילוב עם סטטיסטי כך שנוכל להגביל משתמשים לראות חלקים רצויים בלבד.

\*שאילתות נוטות להחזיר ערכים באותו סדר, לכן נרצה לערבב את התוצאה לפני שמחזירים אותה על מנת למנוע הסקת מסקנות.

הסקת מסקנות DB - הסקת מידע שלא היינו אמורים להיחשף אליו מתוך מידע אליו יש לנו גישה.

ניתן להסיק מסקנות על ידי צפייה בMeta data, למשל ניתן לראות תלויות והקשרים ולהסיק מידע מתלויות אלו.

התמודדות: **בזמן תכנון** - ניתן לשנות את מבנה הטבלאות ולהסיר תלויות בכך שנפרק למספר טבלאות במקום אחת, מה שיאפשר לנו לשלוט בגישות באופן יותר מקרוסקופי. קשה והלוגיקה לא תמיד נוחה והגיונית. **בזמן ריצה** – על ידי ניטור של שאילתות ננסה לזהות הסקת מידע רגיש ולדחות את השאילתה או לשנות אותה.

הסקה מStatisticDB:שני דרכים-

1**.** Positive compromise - יכולים להסיק שערך מסוים קשור למישהו מסוים **2.** Negative compromise -יכולים להסיק שערך מסוים לא קשור למישהו מסוים**.**

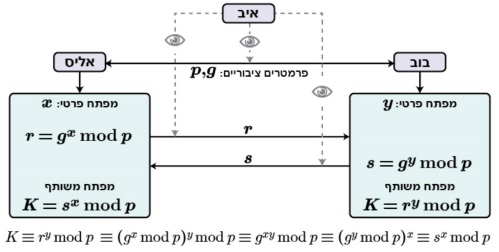
**איך מתגברים? 1.**הגבלת שאילתות מסוימות שנחשוב שיחשפו מידע רגיש. **2.** הפרעה במידע (דרך SDB מתווך עם הפרעות או פלט ישיר עם הפרעות).

Perturbation (הפרעה)– מכניסים הפרעה לנתונים, כך שזה לא פוגע בנתונים למי שמורשה להשתמש בהם, אך מונע מתוקפים להסיק מהם מסקנות, נותן מענה לכל השאילתות והתשובות יהיו מקורבות למידע האמיתי.  
**חיסרון:** קשה לבצע הפרעה על בסיס נתונים קטן, צריך לדאוג לאיזון בהפרעה כי אם טווח השגיאה קטן אנחנו קרובים לערכים האמיתיים ואם הוא גדול אז אנחנו רחוקים מהערכים האמיתיים.  
 Data swapping: שינוי ערכים בעמודות מסוימות במידה מספקת בשביל שמי שאין לו גישה לא יוכל להסיק דברים.

הערכת התפלגות: מוסיפים ערכים חדשים ששומרים על ההתפלגות אך הם לא יהיו הערכים האמיתיים ולא יהיה ניתן להסיק מידע. שינוי בתוצאה: משתמשים באותו DB אבל מכניסים רעשים לתשובה.

הרשאות –  **Cascade**  – הרשאות בשרשרת – אם שללתי מ-B כל מי שנגזר ממנו נשלל גם. **Restrict**- מגביל – אם שללתי מ-B, המשך החולייה מחובר חזרה ללא B.

**מדיניות מרכזית** מס קטן של משתמשים יכול להעניק ולשלול הרשאות **מדיניות מבוססת בעלות** מי שיצר יכול להעביר הרשאות לאחרים (בנוסף למנהלים) **מדיניות מבוזרת** הבעלים יכול להעביר את יכולת ההענקה וההרשאה הלאה.



Kerberos (קרברוס) - מנגנון הפצת מפתחות באופן בטוח על ידי צד שלישי, נותן לנו אפשרות לנהל במקום מרוכז הרשאות לשירותים שונים עם משתמשים שונים.

\*במידה ויש לנו מספר תחומי שליטה, נצטרך ששרתי ה- Kerberos יחלקו מפתחות משותפים.

שרת ה-Kerberos – מורכב מ6 חלקים:

**1.**המשתמש נכנס למערכת ומבקש שירות/שרת.

**2.**השרת AS(auth server) מזהה את ההרשאות שיש לו במאגר הנתונים ואם מזהה משתמש ידוע הוא מייצר TGT(Ticket Granting Ticket) שהוא אישור לקבל כרטיס לשרת האמיתי, הוא מעביר למשתמש אותו עם Session Key .

**3.**עם הכרטיס הולך לTGS(server) שהוא נותן את הכרטיסים לשרתים האמיתיים ולכל שירות צריך לבקש כרטיס חדש.

**4.**בגלל שהוא יודע שהמשתמש מגיע עם אישור מהשרת הראשון הוא עושה וידוא ונותן לו את הכרטיס שבאמצעותו יוכל לקבל שירות מהשרת שאליו ניגש.

**5.**המשתמש שולח לשרת את הכרטיס שקיבל עם שם המשתמש.

**6.**השרת מאמת את הכרטיס שהמשתמש קיבל ואם מאשר הוא מחזיק למשתמש אישור לפעולה שביקש, אם יש צורך באימות הדדי שולח אימות למשתמש.

RSA אלגוריתם:

מפתח ציבורי-KU{e,n}, פרטי- KR{d,n} .

הצפנה: C=M^e mod n, פענוח: M= C^d mod n  
פענוח נוסף : (M^e)^d mod n

1.בחר שני מספרים ראשוניים P וQ   
2.חשב את N=p\*q , 3. חשב o(n) =(p-1)(q-1)

4.בחר e ראשוני שיהיה פחות מ o (n)

5.בחר d כך ש de mod o(n) שווה 1 ו-d<o(n)

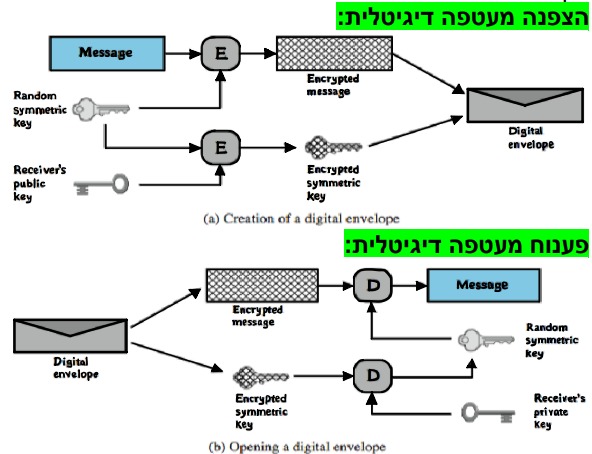
6.חשב C^d mod n ותקבל פענוח .

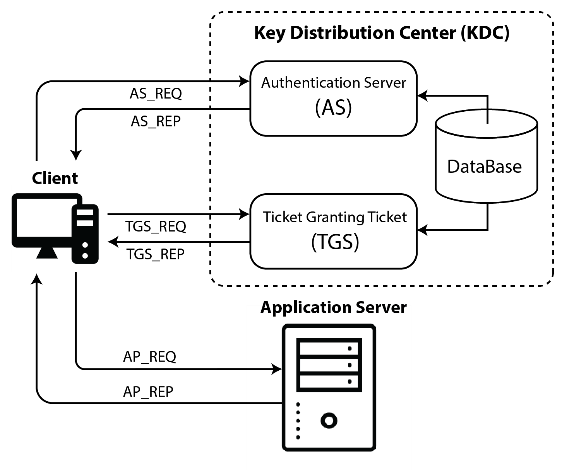
Cross site scripting (XSS)

התקפה נגד גולש אינטרנט המנוצלת ע"י פגיעות ביישומי אינטרנט ומאפשרת לתוקף להזריק סקריפט זדוני שמטרתו לרוץ בדפדפנים של משתמשי מערכת אחרים (נפוץ בעיקר בשפות מבוססות סקריפט).

Injection (command, SQL, code)

התקפות שמנצלות חולשה בתוכנה, נובעת מטיפול לא נכון או לא מקיף בקלט אפשרי, שיכול בטעות/כוונה להשפיע על זרימת התכנית. במקרה של פקודה הקלט מאפשר החדרת פקודה. בשאילתא – החדרה ל DB.

  
**קרברוס-ציור:**

****

**des -** ה64 ביט מתחלקים ל2 קבוצות 32 הביטים הימניים עוברים כקלט לפונקציה ואז זה עושה XOR עם ה32 הביטים השמאליים וגם ה32 הביטים הימנים הולכים כמו שהם לצד שמאל וככה זה חוזר על עצמו 16 פעמים הפונקציות נגזרות מהמפתח ובפענוח עושים את התהליך הזה מהסוף להתחלה.  
**פונקציה בתוך des –** 1.מקבלים ki (מפתח עם 48ביט שנגזר מהמפתח המקורי) ו32ביט של קלט(חצי מהקלט)  
2.הקלט נכנס לתוך ריפלקטור שמעבה אותו ל48ביט  
3.המפתח והקלט נכנסים לתוך קסור  
4.ה48ביט נכנסים לתוך 8 sbox שהופכים את ה6ביט שקיבלו ל4ביט  
5.ה32 ביט המוצפנים עוברים פרמוטציה ומוחזרים לנו חזרה להמשך הdes  
**aes –** תחליף לdes,צופן בלוקים סימטרי, משתמש בבלוקים של 128ביט ומפתחות של 128/192/256 ביט.

**Unix** – 12 ביטים של הגנה, 9 שמחולקים לבעלים, קבוצה ואחרים. 3 ביטים נוספים של setuid/gid, stickybit (לקובץ – המערכת תשמור את הקובץ אחרי כל ריצה, ספרייה – רק הבעלים יכול לשנות את שם הקובץ, להעביר את הקובץ ולמחוק אותו.).  
בגרסה המעודכנת של יוניקס יש תמיכה בacl שמאפשרת לתת הרשאות למשתמשים בצורה פרטנית או לתת הרשאות מסוימות לכמה קבוצות ספציפיות.  
**nonce -** מחרוזת רנדומלית כמו אתגר תגובה לשרת וכך זה יבטיח שתוקף לא יוכל להתחזות לאחד מהשרתים, לא מוגן מפני תקיפה של קובץ סיסמאות.  
**נוסחת אפיון:** נוסחה לוגית שמגדירה אוסף רשומות  
**הגבלת שאילתות:  
הגבלת גודל:** k>1 ,   
\*הגבול העליון של N-k מבטיח שלמשתמש לא תהיה גישה לסטטיסטיקות על שאילתות אם פחות מK רשומות.

**מתקפת עוקב:** בגלל הגבלת גודל השאילתה אנחנו נחלק אותה לכמה חלקים

**שליטה על חפיפה בין שאילתות:** ניתן להגביל חפיפה בין שאילתות ולגלות רצפי שאילתות שיחדיו עלולות להוביל להסקה.

**חציצה:** חלוקת המידע לקבוצות מוגדרות מראש שלא ניתן לחלק אותן.

**מניעת גישה לשאילתות:** חסימת שאילתות מסוימות שידליפו מידע לא רצוי.

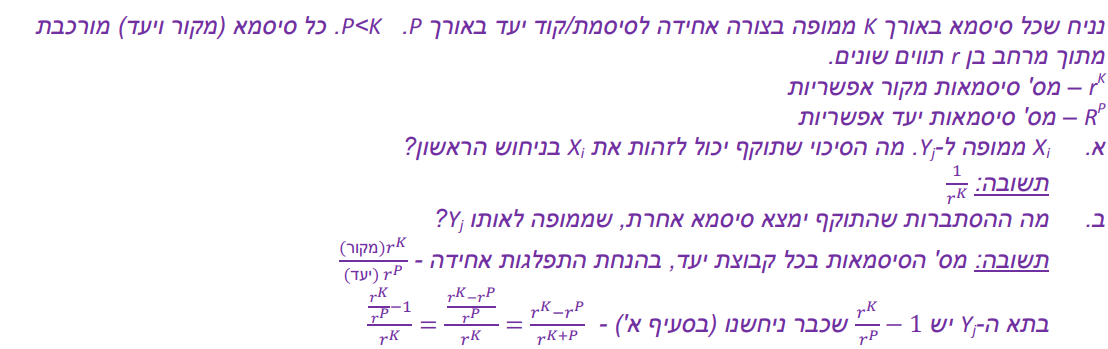
**דוגמא כלשהי למשהו:**כמה קשרים בין משתמשים להרשאות נצטרך בdac לעומת rbac – N תפקידים, לתפקיד K יש Uk משתמשים בתפקיד ולתפקיד k יש Pk הרשאות לתפקיד

**אמצעי זיהוי ביומטרי:**  
**פאסיבי – רשתית**: באמצעות אינפרה אדום, **קשתית**:תמונה של מבנה הקשתית (נחשב להכי בטוח),**טביעת אצבע**:תבנית של הטביעה,**תווי פנים:**מבני פנים או חום.  
**אקטיבי – קול** **וחתימה**: ניתנים לשינוי לכן דרושות יותר דגימות מהמשתמש כדי לקבל התאמה טובה.

**תהליך זיהוי ביומטרי:**

**הרשמה:**קידוד של מזהה ייחודי(תהליך שדורש התקן מיוחד ואמין),**אימות:**ע"פ המזהה (שם משתמש וסיסמא או PIN) והנתון הביומטרי בודקים אם יש ביניהם התאמה,**זיהוי:**מוודא כי הנתון הביומטרי מופיע במאגר המשתמשים.

**דוגמא לחישוב זמן פיצוח סיסמא:**  
1.נתונה סיסמא באורך 4 מתוך אלףבת בן 26תווים,אם מנסים סיסמא אחת לשנייה כמה זמן יקח לגלות סיסמא?  
(26בחזקת4) חלקי 2 – החילוק ב2 הוא בגלל חישוב זמן ממוצע.  
2.אם מקבלים אינדיקציה לאחר כל תו לא חוקי זה יקח 13 כפול 4 שניות, 13 זה עבור 26 חלקי 2 נסיונות בממוצע לתו.



**דיפי-הלמן**-יצירת מפתח משותף בין 2 אנשים שאין ביניהם קשר  
**לגבי בלום פילטר**- ככל שווקטור הביטים גדול יותר נמנע יותר התנגשו יותר וככל שיהיו יותר פונקציות k ואז נקבל פחות שגיאות.  
**בעיית זרימת אינפורמציה-** לאליס יש הרשאות rw לקובץ a וb ולבוב יש רק הרשאת קריאה לקובץ b אליס יכולה לקרוא משהו מקובץ a ולכתוב אותו לb וכרגע בוב קורא משהו שבעיקרון אין לו הרשאת גישה אליו במודל blp אין את הבעיה הזאת בגלל 2 כללי האכיפה