ממ"ן 16 (פרוייקט)

הקורס: מבוא לאבטחת המרחב המקוון - 20940

חומר הלימוד למטלה : ספר הקורס ומדריך הלמידה

מספר השאלות: 2 **משקל המטלה:** 25 נקודות

סמסטר: 2024א **מועד אחרון להגשה:** 15.2.2024

ניתן להגיש בזוגות

שאלה 1 (75%)

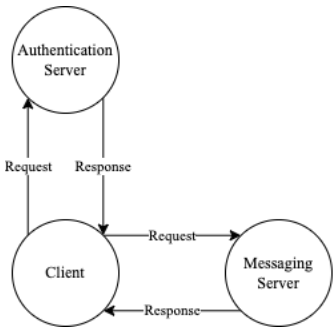
בתרגיל זה תממשו מערכת העברת מסרים המבוססת על פרוטוקול Kerberos. המערכת תכלול שרת אימות, וקוד שרת-לקוח המשתמשים במפתח משותף על מנת להעביר הודעות. הקוד ייכתב בשפת Python או C++ או Java.

חשוב!

קראו היטב את כל המטלה לפני תחילת העבודה. וודאו שאתם מבינים היטב את פרוטוקול התקשורת ואת המבנה של תוכנת השרת והלקוח.

ארכיטקטורה

ארכיטקטורת התוכנה מבוססת על שרת-לקוח. הלקוח יוצר קשר ביוזמתו עם שרת האימות, מאמת את זהותו, ומקבל מפתח סימטרי לתקשורת עם שרת ההודעות. לאחר מכן, הלקוח מעביר לשרת ההודעות את המפתח הסימטרי, אחריו יוכל לשלוח הודעות מוצפנות ומאומתות לשרת ההודעות. תפקיד שרת ההודעות הוא לקבל הודעות מלקוחות ולהדפיס אותן למסך.



שרת האימות

תפקיד שרת האימות הוא לנהל את רשימת המכשירים הרשומים לשירות ולאפשר להם להחליף ביניהם הודעות מסוגים שונים.

א. השרת יתמוך בריבוי משתמשים ע״י חוטים (threads).
ב. גרסת הפרוטוקול היא **24** (גרסה זו מופיעה בהודעות התקשורת).

פורט

השרת יקרא את מספר הפורט מתוך קובץ טקסט בצורה הבאה:

- שם הקובץ: port.info
- מיקום הקובץ: באותה תיקיה של קבצי הקוד של השרת
- תוכן הקובץ: מספר פורט
לדוגמא:
1234

נתונים

השרת ישמור את נתוני הלקוחות והשרתים הרשומים לשירות בזיכרון (RAM) ובקבצים.

מידע על הלקוחות ישמר בקובץ בשם clients. מבנה כל שורה בקובץ:

ID:Name:PasswordHash:LastSeen

כאשר:

שם	סוג	הערות
ID	16 בתים (128 ביט)	מזהה ייחודי עבור כל לקוח. אינדקס
Name	מחרוזת (255 תוים)	מחרוזת ASCII המייצגת שם משתמש. **כולל תו מסיים!** (null terminated)
PasswordHash	32 בתים	תמצית SHA-256 של סיסמת הלקוח. **מהווה מפתח סימטרי ארוך טווח עבור הלקוח**
LastSeen	תאריך ושעה	הזמן בו התקבלה בקשה אחרונה מלקוח

מידע על השרתים ישמר בקובץ בשם servers. מבנה כל שורה בקובץ:

ID:Name:AESKey

כאשר:

שם	סוג	הערות
ID	16 בתים (128 ביט)	מזהה ייחודי עבור כל שרת. אינדקס
Name	מחרוזת (255 תוים)	מחרוזת ASCII המייצגת שם משתמש. **כולל תו מסיים!** (null terminated)
AESKey	32 בתים	מפתח סימטרי ארוך טווח עבור השרת

במקרה והשרת נפל, תהיה לו אפשרות לעלות מחדש ולטעון את רשימת הלקוחות/שרתים הרשומים מהקבצים. לקוחות רשומים יוכלו להמשיך לשלוח בקשות מבלי לבצע רישום מחדש.

אופן פעולת שרת האימות

1. קורא את הפורט מתוך הקובץ port.info. (אם הקובץ לא קיים, להוציא אזהרה ולעבוד על פורט ברירת מחדל 1256. לא להגיע לנפילה עם Traceback במידה והקובץ לא זמין.)
2. השרת בודק את קובץ המכשירים, אם כבר קיים, וטוען נתוני מכשירים שנרשמו בהפעלות קודמות.
3. ממתין לבקשות מלקוחות בלולאה אין סופית.
4. בעת קבלת בקשה מפענח את הבקשה בהתאם לפרוטוקול:

א. בקשה לרישום: במידה ושם הלקוח/שרת המבוקש כבר קיים, שרת האימות יחזיר שגיאה. אחרת, השרת ייצר UUID חדש עבור הלקוח/שרת, ישמור את הנתונים בזיכרון ובקובץ ויחזיר תשובת הצלחה.

ב. בקשת לרשימת שרתי הודעות: השרת יחזיר את רשימת השרתים לפי הפרוטוקול.

ג. בקשה למפתח: שרת האימות ייצור מפתח AES, יצפין אותו בעזרת המפתח של הלקוח וישלח בתגובה יחד עם Ticket שמיועד לשרת ההודעות.

לקוח

תוכנת הלקוח תדע לתקשר מול שרת האימות ושרת ההודעות:

(1) תדע להירשם לשרת האימות (במידה ולא רשום מהפעלה קודמת).
(2) תדע לתקשר עם שרת ההודעות.

א. הלקוח יפעל על פי סדר פעולות קבוע, כך שניתן להפעילו במצב Batch mode.
ב. גרסת הלקוח תהיה **24**.

קובץ פרטי לקוח

שם ומזהה ייחודי[1]: הלקוח ישמור ויקרא את השם והמזהה הייחודי שלו מתוך קובץ טקסט בצורה הבאה:

- שם הקובץ: me.info
- מיקום הקובץ: בתיקיה של קובץ ההרצה/סקריפט
- תוכן הקובץ:

שורה ראשונה: כתובת IP + נקודתיים + מספר פורט
שורה שניה: שם הלקוח (מחרוזת עד 100 תווים)
שורה שלישית: מזהה ייחודי בייצוג ASCII כאשר כל שני תווים מייצגים ערך hex בעל 8 סיביות.
לדוגמא:

```
127.0.0.1:1234
Michael Jackson
64f3f63985f04beb81a0e43321880182
```

שרת ההודעות

השרת מקבל הודעות מוצפנות מלקוחות ומדפיס אותן למסך (stdout). מערכת זו מדמה שרת הדפסה אמיתי מבוסס Kerberos.

א. השרת יתמוך בריבוי משתמשים ע"י חוטים (threads).
ב. גרסת השרת תהיה **24** (גרסה זו מופיעה בהודעות תקשורת מטעם השרת).

[1] בתרגיל זה נעשה שימוש במזהה ייחודי גלובלי (UUID). לקריאה נוספת:
https://en.wikipedia.org/wiki/Universally_unique_identifier

פורט

השרת יקרא את מספר הפורט מתוך קובץ טקסט בצורה הבאה:

- שם הקובץ: port.info
- מיקום הקובץ: באותה תיקיה של קבצי הקוד של השרת
- תוכן הקובץ: מספר פורט
  לדוגמא:
  1234

קובץ פרטי שרת הודעות

שם ומזהה ייחודי: שרת ההודעות ישמור ויקרא את השם והמזהה הייחודי שלו מתוך קובץ טקסט בצורה הבאה:

- שם הקובץ: msg.info
- מיקום הקובץ: בתיקיה של קובץ ההרצה/סקריפט
- תוכן הקובץ:
  שורה ראשונה: מספר פורט
  שורה שניה: שם השרת (מחרוזת עד 100 תווים)
  שורה שלישית: מזהה ייחודי בייצוג ASCII כאשר כל שני תווים מייצגים ערך hex בעל 8 סיביות.
  שורה רביעית: מפתח סימטרי ארוך טווח (משותף עם שרת האימות) שנוצר ברישום הראשונה של התוכנית בפורמט בסיס 64.
  לדוגמא:

```
1234
Printer 20
64f3f63985f04beb81a0e43321880182
MIGdMA0GCSqGSIb3DQEBA…
```

אופן פעולת שרת ההודעות

1. קורא את פרטי השרת מתוך הקובץ msg.info.
2. ממתין לבקשות מלקוחות בלולאה אין סופית.
3. בעת קבלת בקשה מפענח את הבקשה בהתאם לפרוטוקול:
   א. קבלת מפתח: השרת מקבל Ticket, מפענח אותו עם המפתח הסימטרי ארוך הטווח שלו ושומר את המפתח עבור הלקוח.
   ב. הדפסת הודעה: השרת מפענח את ההודעה עם המפתח הסימטרי ומדפיס את ההודעה.

שגיאה מצד השרת

בכל מקרה של שגיאה הלקוח ידפיס למסך הודעה: "server responded with an error".

פעולות אפשריות

בקשת רישום

1. במידה והקובץ me.info לא קיים, הלקוח יקלוט שם משתמש וישלח בקשת רישום לשרת האימות.
2. הלקוח ישמור בקובץ בשם **me.info** את השם והמזהה הייחודי שיקבל מהשרת.
3. במידה והקובץ כן קיים, הלקוח יקרא את הנתונים מהקובץ להתחברות חוזרת.
שימו לב! במידה והקובץ כבר קיים הלקוח לא יירשם שנית.

בקשת רשימת שרתי הודעות

הלקוח ישלח בקשת רשימת שרתי הודעות לשרת האימות. יפענח את התשובה וידפיס למסך את שמות השרתים.

קבלת מפתח AES לשרת הודעות

לאחר שהלקוח מבקש מפתח AES לשרת הודעות ספציפי, הוא מקבל מפתח מוצפן ו-Ticket. הלקוח פותח את המפתח בעזרת התמצית של הסיסמה שלו ושומר את מפתח ה-AES ואת ה-Ticket לשימוש עתידי עם שרת ההודעות.

שליחת הודעה לשרת ההודעות

הלקוח קולט הודעה לשליחה, מצפין אותה בעזרת מפתח ה-AES, ושולח אותה לשרת ההדפסה יחד עם ה-Ticket.

פרוטוקול התקשורת

כללי

- הפרוטוקול הוא בינארי וממומש מעל TCP.
- כל השדות המספריים חייבים להיות עם ערכים גדולים מאפס (**unsigned**) ומיוצגים כ- **little endian**.
- פרוטוקול זה תומך **בבקשות** לשרת **ותשובות** ללקוח. בקשות או תשובות יכולות להכיל **"הודעה"**.
- הודעה עוברת בין לקוחות לשרתי הדפסה.

זכרו! הפרוטוקול מחייב ולא ניתן לעשות בו שינויים. כפועל יוצא, כל שרת ולקוח הממשים את הפרוטוקול (ללא תלות בשפת התכנות) יכולים לעבוד אחד מול השני.

רישום למערכת

1. כל לקוח שמתחבר בפעם הראשונה נרשם בשירות מול שרת האימות עם שם (מחרוזת באורך מקסימלי של 255 בתים) וסיסמה.
2. שרת האימות שומר את תמצית הסיסמה ומחזיר ללקוח מזהה ייחודי שנוצר עבורו או שגיאה אם השם כבר קיים בבסיס הנתונים.

פרטי הפרוטוקול

בקשות

מבנה בקשה מהלקוח לשרת. השרת יפענח את התוכן (payload) לפי קוד הבקשה.

בקשה לשרת

Request	שדה	גודל	משמעות
כותרת (Header)	Client ID	16 בתים (128 ביט)	מזהה ייחודי עבור כל לקוח
	Version	בית	מספר גירסת לקוח
	Code	2 בתים	קוד בקשה
	Payload size	4 בתים	גודל תוכן הבקשה
תוכן (payload)	payload	משתנה	תוכן הבקשה. משתנה בהתאם לבקשה

תוכן (payload)

התוכן משתנה בהתאם לבקשה. לכל בקשה מבנה שונה.

קוד בקשה 1025 – רישום לקוח

שדה	גודל	משמעות
Name	255 בתים	מחרוזת ASCII המייצגת שם משתמש. **כולל תו מסיים!** (null terminated)
Password	255 בתים	מחרוזת ASCII המייצגת סיסמה. **כולל תו מסיים!** (null terminated)

שימו לב: השרת יתעלם מהשדה Client ID

קוד בקשה 1027 – רישום שרת

שדה	גודל	משמעות
Name	255 בתים	מחרוזת ASCII המייצגת שם משתמש. **כולל תו מסיים!** (null terminated)
מפתח סימטרי	32 בתים	מפתח AES סימטרי לשרת ההדפסה (מיועד לפענוח Ticket-ים)

קוד בקשה 1026 – בקשת רשימת שרתי הודעות

שדה payload לא קיים. שדה Payload size=0.

קוד בקשה 1027 – בקשת מפתח סימטרי

שדה	גודל	משמעות
Client ID	16 בתים	מזהה ייחודי עבור כל לקוח
Server ID	16 בתים	מזהה ייחודי עבור כל שרת הדפסה
Nonce	8 בתים	ערך אקראי שהלקוח יוצר

תשובות משרת האימות

Response	שדה	גודל	משמעות
כותרת (Header)	Version	בית	מספר גירסת שרת
	Code	2 בתים	קוד התשובה
	Payload size	4 בתים	גודל תוכן התשובה
תוכן (payload)	payload	משתנה	תוכן התשובה. משתנה בהתאם לתשובה

קוד תשובה 1600 – רישום הצליח

שדה	גודל	משמעות
Client ID	16 בתים	מזהה ייחודי של לקוח/שרת

קוד תשובה 1601 – רישום נכשל

קוד תשובה 1602 – רשימת שרתי הודעות

שדה	גודל	משמעות
Server ID	16 בתים	מזהה ייחודי של שרת
Server Name	255 בתים	מחרוזת ASCII המייצגת שם משתמש. **כולל תו מסיים!** (null terminated)

חשוב: הרשימה עשויה לכלול שרתים רבים. הם יופיעו אחד אחרי השני וניתן לחשב את מספרם ע"י הנוסחה :

Payload Size / (16+255)

קוד תשובה 1603 – שליחת מפתח סימטרי מוצפן

שדה	גודל	משמעות
Client ID	16 בתים	מזהה ייחודי של לקוח
מפתח סימטרי מוצפן		מפתח AES מוצפן ללקוח
Ticket		מוצפן

מבנה המפתח המוצפן :

שדה	גודל	משמעות
IV	16 בתים	
AES key	32 בתים	מפתח הצפנה עבור הלקוח והשרת. **מוצפן** באמצעות המפתח הסימטרי של הלקוח

מבנה ה-Ticket :

שדה	גודל	משמעות
Version	בית	מספר גירסת שרת
Client ID	16 בתים	מזהה ייחודי עבור כל לקוח
Server ID	16 בתים	מזהה ייחודי עבור כל שרת
Timestamp		
Ticket IV	16 בתים	
AES key	32 בתים	מפתח הצפנה עבור הלקוח והשרת. **מוצפן** באמצעות המפתח הסימטרי של השרת
Expiration time	4 בתים	Timestamp, זמן תום התוקף של ה-Ticket. **מוצפן** באמצעות המפתח הסימטרי של השרת

בקשות לשרת ההודעות

קוד בקשה 1028 – שליחת מפתח סימטרי לשרת הודעות

שדה	גודל	משמעות
Authenticator		
Ticket		

מבנה ה-Authenticator :

שדה	גודל	משמעות
Version	בית	מספר גירסת שרת. **מוצפן** באמצעות המפתח הסימטרי של המקבל
Client ID	16 בתים	מזהה ייחודי עבור כל לקוח. **מוצפן** באמצעות המפתח הסימטרי של המקבל
Server ID	16 בתים	מזהה ייחודי עבור כל שרת. **מוצפן** באמצעות המפתח הסימטרי של המקבל
Timestamp	4 בתים	**מוצפן** באמצעות המפתח הסימטרי של המקבל

קוד בקשה 1029 – שליחת הודעה

שדה	גודל	משמעות
Message Size	4 בתים	גודל ההודעה (לאחר הצפנה)
IV	16 בתים	
Message Content	משתנה	תוכן ההודעה. **מוצפן** תחת מפתח סימטרי שנוצר ע״י שרת האימות.

קוד תשובה 1609 – שגיאה כללית בשרת שלא טופלה באחד המקרים הקודמים.

קוד תשובות משרת ההודעות

קוד תשובה 1604 – מאשר קבלת מפתח סימטרי

קוד תשובה 1605 – מאשר קבלת הודעה, תודה

קוד תשובה 1609 – שגיאה כללית בשרת שלא טופלה באחד המקרים הקודמים.

הצפנה

פרוטוקול התקשורת משתמש בהצפנה סימטרית על מנת להעביר מפתחות והודעות.

השתמשו ב-AES-CBC.

אורך המפתח **256 ביט**. נדרש לייצר IV אקראי בכל הצפנה.

שימוש כזה ב- IV לא בטוח אם משתמשים באותו מפתח בכל פעם, אך לצורך הממן הוא מספק.

דגשים לפיתוח

1. מומלץ לעבוד עם מערכת לניהול קוד (כדוגמת גיט[2])
2. עבדו באופן מודולרי ובדקו את עצמכם כל הזמן
   א. זהו את המחלקות והפונקציות החשובות
   ב. **בצד השרת:**
   כיתבו קוד לטיפול בבקשה אחת. הוסיפו תמיכה בריבוי לקוחות בשלב מאוחר יותר
   ג. **בצד הלקוח:**
   ממשו את הרכיבים הגדולים באופן בלתי תלוי בחלקים אחרים של המערכת (תקשורת, הצפנה, פרוטוקול וכו').
3. ממשו קוד לבדיקה כבר בשלבים מוקדמים של הפרויקט
   א. **בצד השרת:**
   השתמשו בהדפסות למסך או בכתיבה ללוג כדי לעקוב אחרי התקשורת. תוכלו גם לטעון את המודול לתוך ה- interpreter ולעבוד באופן דינמי.
   ב. **בצד הלקוח:**
   כיתבו פונקציות קטנות שבודקות חלקים נפרדים של המערכת. השתמשו בפונקציות הללו תוך כדי כתיבת הקוד עצמו.
4. כתיבת הקוד
   א. ממשו את התוכנה לפי עקרונות תכנות מונחה עצמים
   ב. שימו לב לייצוג ערכים בזיכרון כ- little-endian או big-endian
   ג. הקפידו על תיעוד של הקוד (comments)
   ד. תנו שמות משמעותיים למשתנים, פונקציות ומחלקות. המנעו ממספרי קסם!

[2] https://www.atlassian.com/git/tutorials/what-is-version-control

ה. הודעה יכולה להיות גדולה מאוד (בגודל דינמי). חשבו על הדרך הנכונה ביותר לקבל ולשלוח כמות מידע גדולה.
ו. **אבטחת מידע** – חשבו לאורך כל הדרך על כתיבת קוד בטוח לפי העקרונות שלמדתם:
- האם בדקתם את הקלט?
- איך נעשה שימוש בזיכרון דינמי?
- האם מתבצעת המרת טיפוסים (casting) וכו'..

ז. בדקו שגיאות בכל בקשה ותשובה בשרת ובלקוח!

5. **לפני ההגשה**

א. בדקו שהפרויקט מתקמפל ורץ בצורה תקינה ללא קריסות או תלויות בספריות שונות (למעט הספריות הנדרשות לתרגיל)
ב. מומלץ לייצר תיקיה חדשה ולהעתיק לשם את הקבצים המיועדים לשליחה. לייצר פרוייקט VS חדש, לקמפל ולהריץ
ג. העבודה תבדק על מ"ה חלונות עם Visual Studio Community 2022 עם גרסת 17 ++C

המלצות לקוד פייתון

1. השתמשו בפייתון **גירסה 3**
2. עשו שימוש בספריות פייתון הסטנדרטיות בלבד (פרט לספריית ההצפנה)!
3. תוכלו להעזר בספריה **struct** על מנת לעבוד עם נתוני התקשורת בנוחות
4. השרת יפעל עם חבילת הצפנה PyCryptodome, ופרט לכך עם חבילות סטנדרטיות הכלולות במפרש.

המלצות לקוד ++C

1. ממשו את הקוד בשפת ++C תואמת גרסה 11 ומעלה (לדוגמא פונקציות מסוג למדה, שימוש ב- auto וכו'..), בעזרת Visual Studio 2022.
2. עשו שימוש בספריות STL.
3. השתמשו בצד הלקוח בספריה **Crypto++**[3] **(ראו דוגמת קוד באתר הקורס)**
4. למימוש התקשורת עשו שימוש ב- winsock או בספריית boost

הגשה

פייתון

1. עליכם להגיש רק את קבצי הקוד (כלומר קבצי py.).
שימו לב! על התוכנית להטען ולרוץ בצורה תקינה (ללא צורך בתוספות קבצים וללא קריסות).
2. יש לכלול פונקציה ראשית בשם **main**. פונקציה זו תהיה הפונקציה הראשית של תוכנית השרת והיא תעבוד לפי אופן פעולת השרת המפורט לעיל.
טיפ:
תוכלו להשתמש במנגנון הבא כדי לאפשר עבודה אינטראקטיבית **וגם** הרצה של הקוד:

```
if __name__ == "__main__":
```

[3] https://www.cryptopp.com/

C++

1. עליכם להגיש רק את קבצי הקוד (כלומר קבצי h. ו- cpp.).
   שימו לב! על התוכנית לרוץ בצורה תקינה (ללא צורך בתוספות קבצים, ללא קריסות)
2. עבודתכם תיבדק במערכת הפעלה חלונות, באמצעות Visual Studio ולכן מומלץ לעבוד עם סביבה זו.

וידאו עם דוגמת ריצה

עליכם להקליט וידאו ממסך המחשב, בו אתם פותחים שני חלונות cmd במקביל ומריצים את המערכת שפיתחתם. יש להפעיל קודם את השרת, לאחר מכן גם את לקוח, לעבור את התהליך של רישום לקוח והחלפת מפתחות כאשר ההודעות המתאימות מופיעות בשני החלונות במקביל, והעברת קובץ נתונים בינארי בגודל של כ-100 KB מהלקוח לשרת. בוידאו צריך להיות פרט מזהה הכולל את השם או תעודת הזהות שלכם, והוא צריך להימשך 2-5 דקות.

שאלה 2 (25%)

עליכם לממשל התקפת מילון נגד ההודעה הראשונה של שרת האימות בפרוטוקול. יש להגיש מסמך המסביר את ההתקפה והצעה לתיקון.

הגשה

מסמך word או pdf.

את כלל קבצי המערכת יש לארוז לקובץ zip.