

מטלת מנחה 15

31.10.2022

הדר תפארת, 205492507

שאלה 1

בחרתי לממש את הפרוטוקול stateless, כלומר באופן בו כל בקשה של לקוח פותחת חיבור חדש לשרת והחיבור נסגר לאחר הטיפול בבקשה שנשלחה (קבלת תשובה מהשרת או סיום שליחת הבקשה, כתלות בבקשה עצמה).

הערה: הוספתי בקוד השרת ובקוד הלקוח הדפסות למסך אשר נועדו לסייע בהמחשת התהליך המתרחש. הודעות אלו מופיעות מיד לאחר comment אחיד המצוין: "for ease of operation tracking", כך שניתן להסין מהקוד בקלות במידה ויש צורך בכך.

חלק 1

חלק זה כולל שרת אשר מנהל תקשורת עם לקוחות ושמירת קבצים המועברים ברשת לאחר הצפנה על פי הפרוטוקול הנתון. השרת כתוב בשפת python.

לכל בקשה אפשרית מהשרת על פי הפרוטוקול, הגדרתי מחלקה אשר יורשת ממחלקה בסיסית של בקשה, ובאותו האופן הגדרתי מחלקה לכל תשובה אפשרית של השרת על פי הפרוטוקול אשר יורשת ממחלקה בסיסית של תשובה.

השרת ממתין לבקשות חיבור, ובעת קבלת בקשה – פותח socket חדש עם port פנוי, מדפיס למסך הודעה בדבר קבלת בקשת חיבור חדשה ומעביר את הטיפול בבקשה לפונקציה service_client על גבי thread חדש.

הפונקציה service_client אחראית לטיפול בבקשה מקבלתה ועד סגירת ההתקשרות עם הלקוח. ראשית הפונקציה מקבלת אובייקט database האמון על שמירת המידע של השרת בזיכרון הנדיף ובקובץ sqlite.

בנוסף הפונקציה יוצרת אובייקט Protocol אשר אחראי על בדיקת תקינותו של הקלט המגיע מהמשתמש ווידוא היישום של השדות השונים כפי שמוגדרים בפרוטוקול.

לאחר מכן הפונקציה קוראת מהsocket מספיק בתים ליצירת header של בקשה תקינה. 23 בתים, לפי:

בקשה לשרת

Request	שדה	גודל	משמעות
נתות (Header)	Client ID	16 בתים (128 ביט)	מזהה ייחודי עבור כל לקוח
	Version	בית	מספר גרסת לקוח
	Code	2 בתים	קוד בקשה
	Payload size	4 בתים	גודל תוכן הבקשה

במידה ונקלטה פנייה עם קוד בקשה מוכר על פי הפרוטוקול – נוצר אובייקט מהמחלקה לטיפול בבקשה זו, אחרת – נוצר אובייקט מהמחלקה לטיפול בבקשה שאינה נתמכת, אשר לפי הפרוטוקול מסיימת את ההתקשרות ללא שליחת תשובה.

כל המחלקות לטיפול בבקשות מכילות מתודה בשם process אשר מקבלת את הsocket של התקשורת עם הלקוח ומחזירה אובייקט של המחלקה המתאימה לטיפול בתשובת השרת ללקוח, בהתאם לבקשה הספציפית שהתקבלה. לאחר שליחת תשובת השרת response.execute – מבוצעת סגירה של התקשורת עם הלקוח והthread אשר טיפל בבקשה נסגר.

פרוטוקול:

ההתאמה וההכנות של המידע המועבר בשרת ביחס לפרוטוקול נבדקות על ידי אובייקט של המחלקה Protocol מתוך הקובץ protocol.py.

המחלקה Protocol מכילה הגדרות לגודל ולפורמט המצוין בעבור השדות השונים בפרוטוקול הנתון, בנוסף לפונקציות אשר בודקות את תקינות שדה מסוים ביחס לפרוטוקול.

בחרתי ליצור אובייקט של מחלקה זו בעבור כל בקשה חדשה. כך ניתן יהיה להרחיב את התכנית בקלות יחסית ולאפשר תמיכה במספר פרוטוקולים שונים בהתאם לאובייקט המחלקה שמוגדר.

תקשורת:

משום שהפונקציה הסטנדרטית `socket.recv(n)` מקבלת `socket` עד `n` בתים, אך עלולה לחזור גם עם פחות מ-`n` בתים אם זו הכמות שהמתינה ב-`socket` בעת ביצוע הקריאה – יתכן מצב בו שימוש בפונקציה זו עלול לגרום לשרת לחשוב שהודעה המגיעה מלקוח אינה תקינה, רק משום שההודעה נשלחה בחלקים קטנים יותר ממה שהשרת ציפה. בכדי למנוע בעיה זו – הגדרתי את המתודה `receive_exact_bytes` בקובץ `connection_utility.py`.

המתודה הנ"ל מבצעת קריאות בלולאה עד שמתקבלת בדיוק כמות הבתים המצוינת בקריאה לפונקציה. כך ניתן לבצע שאילתה לגדלים הידועים מראש בפרוטוקול ולתמוך בכל לקוח אשר פועל לפי הפרוטוקול – בין אם הוא שולח את כל הבקשה בפעם אחת או בית בית.

ניהול זיכרון השרת:

המידע השמור בשרת מחולק לשני גורמים – טבלאות `sqlite` המוגדרות בקובץ `database.py` בתוך אובייקט של המחלקה `DB`, וזיכרון נדיף אשר מוגדר בקובץ `client_info.py` בתוך אובייקט של המחלקה `ClientInfo`. את המחלקה `DB`, אשר מטפלת בניהול ושמירת מידע לקובץ בטבלאות `sql`, כתבתי באופן המאפשר תמיכה בתכנית השרת גם ללא שימוש בזיכרון הנדיף (ולמעשה עדיין ניתן לממש זאת כך על ידי החלפת האובייקט הנשלח לכל בקשה חדשה המתקבלת אצל השרת מאובייקט של המחלקה `ClientInfo` לאובייקט המחלקה `DB`) מחלקה זו מכילה בנאי אשר יוצר את הטבלאות המצוינות בפרוטוקול בקובץ עם שם נתון, במידה והטבלאות אינן קיימות. מתודה לייצוא המידע השמור בטבלאות (לשם טעינתו לזיכרון הנדיף) ומתודות בעבור כל פעולה של הוצאה או הכנסת מידע למאגר הנתונים המתוארת בפרוטוקול, כאשר כל שאילתה הכוללת מידע אשר מסופק על ידי הלקוח או גורם חיצוני למאגר מבוצעת על ידי שאילתה פרמטרית, על מנת להגן מפני הזרקות של פקודות `SQL` נוספות כמשתנים בשאילתה המקורית.

המחלקה `ClientInfo`, אשר מטפלת בניהול המידע בזיכרון הנדיף, מקבלת כארגומנט את שם קובץ ה-`sql` ממנו יטען ואליו ישמר המידע בהמשך. המחלקה יוצרת אובייקט של המחלקה `DB` ומתקשרת דרכו עם קובץ ה-`sql`.

המחלקה מכילה מתודה בעבור כל שאילתה ואינטראקציה עם מאגר הלקוחות המוגדרות בפרוטוקול ושמירה של מידע חדש לקובץ מאגר הנתונים המנוהל על ידי `DB` לאחר עדכון המידע בזיכרון הנדיף.

משום שאובייקט אחד של המחלקה `ClientInfo` אחראי לתחזוקת כל המידע הצבור בשרת, המחלקה מתוכננת לתמיכה ב-`multithreading`:

שאילתות אשר מבקשות מידע מהמאגר ללא עדכון המידע וביצוע שינוי בו יכולות להתקיים במקביל.

שאילתות המבקשות להוסיף משתמש למאגר (רישום) או להוסיף קובץ לשרת (ובכך משנות את המידע הרלוונטי במאגר לכלל המשתמשים) יכולות להתבצע אחת בכל זמן נתון, על ידי `reentrant locks` – `clients_lock` בעבור הגישה למאגר המשתמשים `files_lock` בעבור הגישה למאגר הקבצים.

בנוסף, הגישה בזיכרון לכל אובייקט המציין משתמש מוגבלת ל-`thread` יחיד בעבור כל משתמש בכל זמן נתון, על ידי מנעול `client_lock`, בכדי למנוע מצב בו תקשורת של אותו לקוח עם השרת בשתי מופעים שונים באותו הזמן משבשת את אמינות מאגר הנתונים.

טיפול בבקשות לקוח:

כאמור, כל בקשה חוקית לפי הפרוטוקול מנוהלת על ידי מחלקה מתאימה אשר יורשת ממחלקה כללית של בקשה.

המידע הנקלט מהsocket מתורגם לצורה המתאימה באמצעות שימוש בstruct, כאשר הגדלים והסוגים של השדות השונים נקראים מאובייקט המחלקה Protocol, אשר אחראית על היצמדות לפרוטוקול התקשורת.

כל קלט המגיע מהלקוח נבדל בעבור תקינותו באמצעות מתודה מתאימה של protocol, ולאחר מכן המידע מאומת\נשמר במאגר הנתונים של השרת (זיכרון נדיף וקובץ טבלאות sql) באמצעות מתודה מתאימה של אובייקט המחלקה ClientInfo.

אם במהלך הפענוח של הבקשה נתקלים בשגיאה אשר לא תואמת לאחת מהתשובות המצוינות בפרוטוקול – הודעה מתאימה מודפסת למסך השרת והתקשורת עם הלקוח נסגרת. אחרת – הבקשה מפורשת בהתאם ותשובה מתאימה מוחזרת ללקוח לפני סגירת התקשורת עם הלקוח.

- בבקשה להרשמה של לקוח, במידה ולא רשום כבר לקוח עם אותו שם משתמש – מוגרל uuid בעבור הלקוח החדש. למרות שהסיכוי לכך קטן מאוד – רישום הלקוח החדש מבוצע בלולאה עד אשר מוגרל uuid אשר אינו מופיע במאגר המידע בעבור לקוח קיים.
- בבקשה לשמירת קובץ, מכיוון שבחירתי לאפשר ללקוחות לשלוח מסלול של קובץ (וכך לאפשר ללקוח לשמור שני קבצים בעלי שם זהה אך מתיקיות שונות), ביצעתי הגרלה של שם רנדומלי באורך המקסימלי האפשרי לפי הפרוטוקול (פחות תו אחד בעבור null המצוין כדרישה לשמירת השם במאגר הנתונים), לכן יתכן בסיכוי נמוך מאוד שהשם שיוגרל בעבור קובץ חדש כבר נמצא בשימוש על ידי קובץ אחר. בכדי לטפל במצב זה, ראשית מבוצעת בדיקה אם הקובץ המבוקש כבר קיים בעבור הלקוח במאגר.
- במידה והלקוח כבר שמר את הקובץ בשרת – מוחזר השם בו נשמר הקובץ והבקשה הנוכחית דורסת קובץ זה אחרת, מבוצעת הגרלה של שם חדש בעבור הקובץ, עד אשר מוגרל שם שאינו משויך עדיין לקובץ אחר בשרת והבקשה מבצעת רישום של הקובץ בצירוף השם שהוגרל למאגר המידע של השרת.
- במידה ואירעה שגיאה במהלך קליטת, פיענוח או כתיבת הקובץ שנשלח מהלקוח, או במידה והגיעה הודעה מהלקוח שהקובץ אינו מכיל את ערך checksum המצופה – הקובץ נמחק פיזית מהשרת (במידה והספיק להישמר בו) וממאגרי הזיכרון של השרת.

חלק 2

חלק זה כולל לקוח אשר מנהל תקשורת עם השרת בהתאם לפרוטוקול המצוין במטלה. תכנית הלקוח כתובה בשפה C++.

לכל בקשה לשרת ותשובה מהשרת אשר נתמכת על פי הפרוטוקול כתבתי מחלקה מתאימה, כאשר מחלקות הבקשות יורשות ממחלקת בקשה בסיסית ומחלקות התשובות יורשות ממחלקת תשובה בסיסית.

המחלקות הנ"ל אחראיות על שליחה וקבלה של מידע מהשרת בהתאם לסוג הפניה. כל בקשה / תשובה מכילה struct אחד בעבור header של הפניה ואחד בעבור payload, כאשר מבנים אלו מוגדרים בתוך #pragma pack על מנת לסמן לקומפיילר של visual studio שאין להוסיף padding למבנה.

כל בקשה עוברת בדיקת נכונות ותאימות לפרוטוקול של כל שדה, לפני השליחה לשרת, וכך גם מתבצע בעבור כל תשובה, לאחר הקבלה מהשרת.

את פעולת הלקוח מגדירה מחלקה בשם Client אשר מכילה מתודות לכל אחת מהפעולות המותרות לשימוש לפי הפרוטוקול.

כל מתודה כזו מבצעת חיבור לשרת, שולחת את הבקשה ומצפה לקבל תשובה מתאימה מהשרת במידה וכך מצופה לפי הפרוטוקול לפני סגירת התקשורת, כך שהתקשורת היא חסרת מצב כפי שבחרתי להגדיר אותה.

בניתי את הלקוח באופן זה על מנת לתמוך בתוספות אופציונליות לתכנית (למשל, ניתן בקלות, על ידי קריאה נוספת למתודה `Client.sendFile(filepath)` לבצע בקשה לשמירת קובץ נוסף בשרת עם אותו מפתח הצפנה שהתקבל בעבור הבקשה הקודמת)

תכנית הלקוח עצמה מופעלת מתוך קובץ `main` אשר קורא מהקובץ `transfer.info` את פרטי ההתקשרות ומריץ כל בקשה ברצף הבקשות המצוינות בפרוטוקול עד 3 פעמים חוזרות במקרה של כישלון.

פרוטוקול:

ההתאמה ונכונות המידע אשר עובר בלקוח ומועבר לשרת ביחס לפרוטוקול מאומתת על ידי המחלקה `Protocol`. המחלקה מכילה הגדרות לגדלי השדות השונים המוגדרים בפרוטוקול, לצד מתודות לבדיקת התאימות של התוכן בעבור כל שדה. כך ניתן להתאים את התכנית לפרוטוקול שונה במקצת על ידי שינוי הגדלים הספציפיים במחלקה זו.

תקשורת:

בדומה לשרת, הגדרתי פונקציות עזר בכדי לסייע בתקשורת עם השרת. הפונקציות במחלקה `ConnectionUtility` מסתמכות על המימוש של `Boost::asio`, ומוסיפות הודעה רלוונטית במקרה של כשל בתקשורת. בנוסף, המתודה `receiveBytes` פועלת בדומה למתודה שהוגדרה בשרת על מנת לקלוט כמות מסוימת של בתים בדיוק, כך מתאפשרת היכולת של הלקוח להסתמך על קליטה נכונה של תשובת השרת ללא תלות באופן השליחה הספציפי שלו (תשובה במלואה או בית בית).

הצפנה:

לשם ההצפנה השתמשתי ב `cryptopp wrappers` אשר סופקו באתר הקורס.

יש לציין כי במחלקה `AESWrapper`, המתודה `encrypt` מחזירה את המערך הנתון לאחר הצפנה כ `string`. כאשר מעבירים למתודה זו קובץ גדול יחסית (מעל 550mb) בסביבת עבודה של 32bit – המתודה קורסת. אך במערכת המבוקשת, בסביבת עבודה של 64bit אין לתכנית בעיה לטעון לזיכרון, להצפין ולהעביר קבצים גם בגודל המקסימלי המתאפשר לפי הפרוטוקול (קצת מעל 4gb).

פונקציות עזר:

הגדרתי מחלקת עזר בשם `Utility` אשר תומכת בניהול המערכים המופיעים בתכנית.

במיוחד נעשה שימוש במתודה `stringToChar` אשר מעתיקה תווים מאובייקט `string` לתוך מערך תווים נתון, ובמידה והמערך גדול מה `string` – מאפסת את שאר התווים במערך. המתודה הנ"ל שימושית בעיקר משום שאת המידע הפנימי של הלקוח נוח לשמור כאובייקט `string`, שכן ניהול הזיכרון שלו פשוט יותר ופחות בעייתי, אך שליחת המידע לפי הפרוטוקול (ובעיקר כ `payload` | `header`) מתנהל כמערך של תווים, ופשוט להצגה בתור `structs`, כך ששיטה למעבר מאופן הייצוג הפנימי של התכנית לאופן הייצוג המצופה בתקשורת, תוך איפוס המקומות אשר אינם בשימוש פעיל מקלה על מימוש תכנית הלקוח.

הלקוח, בהתאם לדרישה המצוינת במטלה – פועל בצורה קבועה מראש.

ראשית התכנית טוענת מהקובץ `transfer.info` את כתובת ה IP והפורט עליו ממתין השרת, שם המשתמש של הלקוח ושם הקובץ אותו הלקוח ינסה לשלוח באופן מוצפן לשרת.

בפעילות התכנית של הלקוח מול השרת:

- 1) הלקוח מנסה לבצע הרשמה לשרת – אם קיים בתקיה של הלקוח קובץ בשם `me.info` אשר מכיל, לפי הפרוטוקול, את שם המשתמש כפי שידוע ללקוח (מהמידע שנלקח מהקובץ `transfer.info` בתחילת ההרצה) בשורה הראשונה 32ו תווים אשר ניתנים להמרה מ `hex` ל 16 בתים אשר יצינו את `client id` בתקשורת מול השרת, המידע נשמר אצל הלקוח ומדלגים על הצורך בתקשורת מול השרת בכדי להירשם, אחרת נשלחת בקשה להרשמה אצל השרת דרך אובייקט המחלקה

- RequestRegistration, ובמידה ומתקבלת תשובה התואמת לאובייקט המחלקה ResponseRegistrationSuccess, מעדכנים את תוכן client id של הלקוח ושומרים את שם המשתמש והid בהתאם לפרוטוקול בקובץ me.info.
- (2) הלקוח שולח בקשה להחלפת מפתחות הצפנה על ידי שימוש במחלקה RequestSendKey, אשר כחלק מפעילותה יוצרת צמד מפתחות RSA פרטי ופומבי, כותבת את המפתח הפרטי בהתאם לפרוטוקול לקובץ me.info ושולחת את המפתח הפומבי לשרת. לאחר מכן הלקוח ממתיין לקבלת תשובה המתאימה למחלקה ResponseKeyExchange, מפענח את מפתח האes שנשלח מהשרת באמצעות מפתח הפרטי שלו ושומר את מפתח האes המפוענח בזיכרון לשימוש בפניה הבאה.
- (3) הלקוח שולח בקשה לשליחת קובץ מוצפן על ידי אובייקט מהמחלקה RequestSendFile, אשר מחשבת את ערך הcrc בעבור הקובץ שצוין בהגדרת הלקוח, טוענת את הקובץ המצוין לזיכרון, מצפינה אותו באמצעות מפתח האes שהתקבל בבקשה 2 ושולחת את הקובץ המוצפן לשרת. לאחר מכן התכנית מחכה לקבלת הודעת קבלה מהשרת התואמת את המחלקה ResponseFileReceived. במידה וערך הcrc שהתקבל מהשרת בעבור הקובץ תואם את הערך שחושב בבקשה 3 – הלקוח שולח בקשה לאימות ערך הcrc על ידי אובייקט מהמחלקה RequestAcceptCRC וממתיין לקבלת תשובה התואמת למחלקה ResponseFileVerified, לאחר קבלתה הלקוח מסיים את פעולתו.
- במידה והתקשורת או אחת המתודות נכשלת במהלך ריצת הלקוח - הבקשה שכשלה מתבצעת עד 3 פעמים חוזרות, לאחר הדפסת הודעת כשל גנרית למסך, ואם הבקשה נכשלת בפעם הרביעית – מודפסת למסך הודעת כשל מפורטת והלקוח מסיים את פעולתו בכישלון.
 - במידה והשרת מחזיר ערך crc שונה מהערך שחושב אצל הלקוח בעבור הקובץ – מבוצעת שליחה חוזרת של הקובץ עד 3 פעמים, ואם מתקבלת תשובה של ערך crc שונה בפעם הרביעית – נשלחת בקשת RequestRejectCRC, אחריה הלקוח מסיים את פעולתו.

שאלה 2

במסמך question 2 – findings summary.pdf צירפתי פירוט של החולשות המרכזיות שמצאתי בפרוטוקול, חולשות אשר עדיין קיימות גם במימוש שלי.

חולשות אלו כוללות בין היתר:

- חולשה מפני תקיפת man in the middle, כאשר גורם שלישי מתחזה לשרת בעבור הלקוח וללקוח בעבור השרת ומשבש את התקשורת התקינה ביניהם.
- חולשה של ניצול לא מבוקר של משאבי השרת, שכן לפי הפרוטוקול אין הגבלה על כמות המשתמשים שניתן ליצור או על כמות וגודל הקבצים שכל לקוח יכול לשלוח.
- חולשה מפני גישה לקבצים של הלקוחות השמורים בשרת בעקבות תקיפה ישירה של המערכת אשר מריצה את תכנית השרת (שכן הקבצים שמורים בשרת לאחר פיענוח ההצפנה).

במסמך שכתבתי לא תיארתי חולשות נוספות אשר עלולות להיגרם על ידי מימושים שונים של הפרוטוקול, כגון בעיות של path traversal בעת שמירת הקבצים שנשלחים מהלקוח בשרת, או buffer overflow בעת שמירת המידע הרלוונטי לריצת התכנית אצל הלקוח.

זאת משום שבעיות אלו אינן מושרשות בפרוטוקול עצמו אלא תלויות במימוש ספציפי ודאגתי לטפל בהן במימוש שעשיתי לשרת והלקוח על גבי הפרוטוקול בשאלה 1.

קובץ זה נכתב באנגלית לשם הנוחות, אך אוכל לשלוח גרסה בעברית במידה ויש צורך בכך.

דוגמת הרצה

שרת נותן שירות לשני לקוחות בו זמנית, כולל הרשמה ושליחת קובץ.

תדפיס שרת:

```
Run - DefensiveProgrammingMmn15
main
C:\Users\Tiferet\PycharmProjects\DefensiveProgrammingMmn15\venv\Scripts\python.exe C:\Users\Tiferet\PycharmProjects\DefensiveProgrammingMmn15\main.py
Invalid port number read from file
failed to read a legitimate port for the server from: port.info , will use default port 1234
Server is listening for clients
connection established with: 127.0.0.1:57496
received a registration request from client
Michael Moore
sent a registration success response to client
communication ended successfully with: 127.0.0.1:57496
connection established with: 127.0.0.1:57497
connection established with: 127.0.0.1:57498
received a registration request from client
Michael Jackson
sent a registration success response to client
communication ended successfully with: 127.0.0.1:57498
received a key exchange request from client
Michael Moore
connection established with: 127.0.0.1:57499
sent a key exchange response to client
communication ended successfully with: 127.0.0.1:57497
connection established with: 127.0.0.1:57500
received a key exchange request from client
Michael Jackson
sent a key exchange response to client
communication ended successfully with: 127.0.0.1:57499
connection established with: 127.0.0.1:57501
received a send file request from client
attempt to receive and decipher a file from client
received a send file request from client
attempt to receive and decipher a file from client
received and decipher a file from client, calculating crc value for file
received and decipher a file from client, calculating crc value for file
sent a file received response to client
communication ended successfully with: 127.0.0.1:57501
connection established with: 127.0.0.1:57587
received a crc correct request from client
sent a crc verified response to client
communication ended successfully with: 127.0.0.1:57587
sent a file received response to client
communication ended successfully with: 127.0.0.1:57500
connection established with: 127.0.0.1:57592
received a crc correct request from client
sent a crc verified response to client
communication ended successfully with: 127.0.0.1:57592
```

- ניתן לראות כי השרת התחיל את פעולתו עם קובץ port לא תקין ולכן השתמש בערך הדיפולטיבי.
- ההדפסה כוללת גם הצגת שם המשתמש הפעיל בבקשה, הדפסה זו לא קיימת בגרסה הסופית.

תדפיס לקוח 1 (עם הדפסים תואמים לגרסת הלקוח הסופית):

```
Microsoft Visual Studio Debug
sent a registration request to server
received a registration success response from server
sent a key exchange request to server
received a key exchange response from server
sent a send file request to server
received a file received response from server - checksum match
sent an accept crc request to server
received a file verified response from server
C:\Users\Tiferet\source\repos\DefensiveProgrammingMmn15\x64\Debug\Client.exe (process 17040) exited with code 0.
```

תדפיס לקוח 2 (עם הדפסים מגרסה ישנה יותר של הלקוח):

```
Microsoft Visual Studio Debu X + v
request: client id= version= code=1100 payload size=255
response: version= code=2100 payload size=16
request: client id=:0&^_hLd-q-xm+TJ]M*f@ version= code=1101 payload size=415
response: version= code=2102 payload size=144
entered response key exchange
in response key exchange, received encrypted key
in response key exchange, decrypted the key
attempting to read file, size: 1353405353
read file, size: 1353405353
content sent to server
response: version= code=2103 payload size=279
request: client id=:0&^_hLd-q-xm+TJ]M*f@ version= code=1104 payload size=271

C:\Users\Tiferet\Desktop\uni\sem7\Defensive programming\mmn15\backup 30.10.22\client\DefensiveProgrammingMmn15\x64\Debug
\Client.exe (process 26660) exited with code 0.
```

פרטי הקובץ הנשלח:

gamerresources_4_1.streamdb	21/03/2020 16:01	STREAMDB File	1,321,685 KB
-----------------------------	------------------	---------------	--------------

תוכן תיקיית הקבצים השמורים בשרת בסיום הפעולה:

<< Local Disk (C:) > Users > Tiferet > PycharmProjects > DefensiveProgrammingMmn15 > client_files

Name	Date modified	Type	Size
FJy5gs7PqDaAJ2HwnchC24kdxnQkpwAx...	31/10/2022 16:13	File	1,321,685 KB
IHyikrYHhAzHU3r5Q5BEdRAKivEI2rJGIUsr...	31/10/2022 16:13	File	1,321,685 KB

IHyikrYHhAzHU3r5Q5BEdRAKivEI2rJGIUsr...A5MYK89JQtmThdUaW6clwz6JY7uPi3UULA3xPsLmKHWE5a7phYZHwBXKcfHmL9QDKvBd15FiNcaFPh8Nbuv6jhujhtPZTbERvJE8fL4VYgVnvK9FUOn4WCACopoMwkErSL3SRmJ5ZcbtbfI7Mok4U11glx3A47PKzFE992m1I2H6qY8C9RjYyH7LV41poBhOdTGIPKGB
Type: File
Size: 1.25 GB
Date modified: 31/10/2022 16:13

- ניתן לראות כי הקבצים נשמרים בצד הלקוח עם שם ייחודי אשר מוגרל רנדומלית ללא תלות בשם המקורי של הקובץ.

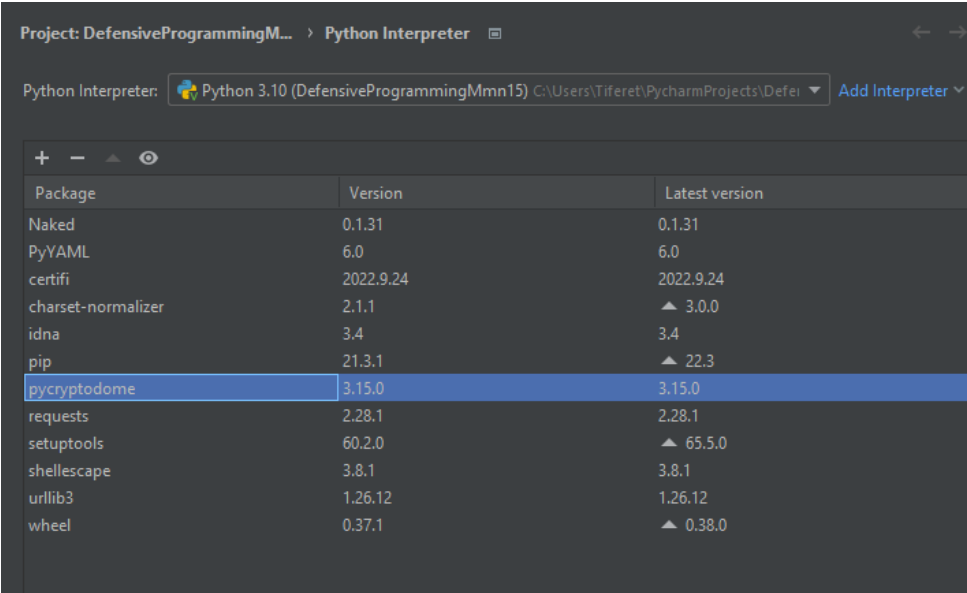
הרצת הלקוח כאשר השרת אינו פעיל:

```
Microsoft Visual Studio Debu X + v
server responded with an error
server responded with an error
server responded with an error
exception: failed to establish a connection to the server
exception: connect: No connection could be made because the target machine actively refused it [system:10061 at C:\Program Files\boost\boost_1_80_0\boost\asio\detail\win_iocp_socket_service.hpp:632:5 in function 'connect']

C:\Users\Tiferet\source\repos\DefensiveProgrammingMmn15\x64\Debug\Client.exe (process 13168) exited with code 1.
```

נספח:

לשם השימוש במחלקות ההצפנה בצד השרת, השתמשתי בחבילה pycryptodome, המסומנת בתמונה הבאה:



The screenshot shows the 'Python Interpreter' window for a project named 'DefensiveProgrammingM...'. It lists installed packages and their versions. The 'pycryptodome' package is highlighted in blue.

Package	Version	Latest version
Naked	0.1.31	0.1.31
PyYAML	6.0	6.0
certifi	2022.9.24	2022.9.24
charset-normalizer	2.1.1	▲ 3.0.0
idna	3.4	3.4
pip	21.3.1	▲ 22.3
pycryptodome	3.15.0	3.15.0
requests	2.28.1	2.28.1
setuptools	60.2.0	▲ 65.5.0
shellescape	3.8.1	3.8.1
urllib3	1.26.12	1.26.12
wheel	0.37.1	▲ 0.38.0

ואת תוכנת הלקוח בניתי לסביבת עבודה של debug, 64bit, ונעזרתי בboost וcryptopp, בהתאם לדרישות המטלה.