

**עבודת גמר 5 יח"ל**

**נושא העבודה :ממשק משתמש לשליטה על מחשבים ברשת המקומית**

**שם תלמיד :** מקסים לכוביצקי

**ת.ז תלמיד :** 213476112

**שם בית ספר ועיר :** קריית החינוך ע"ש עמוס דה-שליט, רחובות

**שם המנחה :** ערן בינט

**מועד הגשה :** 28.3.2020

תוכן עניינים

[1. מבוא 3](#_Toc509432052)

[2. תיאוריה 4](#_Toc509432053)

[3. תוצר סופי 7](#_Toc509432054)

[4. תהליך כתיבת הפרויקט 13](#_Toc509432055)

[5. מרכיבי פתרון 15](#_Toc509432056)

[6. תסריטי בדיקה 20](#_Toc509432057)

[7. רפלקציה 21](#_Toc509432058)

[8. הוראות התקנה ותפעול 23](#_Toc509432059)

[9. ביבליוגרפיה 23](#_Toc509432060)

[10. נספחים 24](#_Toc509432061)

1. מבוא
   1. נושא העבודה

נושא הפרויקט אותו בחרתי ליישם הוא Remote Control Shell. במסגרת זו המערכת תעסוק בשליטה במחשבים הנמצאים באותה סביבת LAN, ובבקרת הפעילות המתבצעת במחשבים אלה.

התוצר הסופי ייוצג על ידי ממשק דמוי cmd המוצע על ידי Windows, אשר יאפשר למשתמש לראות את המחשבים השונים בסביבת העבודה ואת פרטיהם, ולבצע מספר פקודות מרחוק שלרוב ניתנות לביצוע רק פיזית על המחשב עם הרשאות מנהל. בנוסף, למערכת יהיה זיכרון שיתופי שמור משלה שבו יוכלו המחשבים לשמור ולחלוק קבצים – קריאה וכתיבה. כתיבת הפקודות בקונסולה תיושם מעל interpreter של שפה עילית שיגדיר תחביר שפה עילית של הפקודות

* 1. מטרות מרכזיות

המטרות המרכזיות של העבודה הן:

* קבלה וניתוח של מידע על מחשבים ברשת
* שיתוף קבצים בין מחשבים ללא שליחה ישירה
* שליטה על תהליכים במחשבים מרוחקים
* קריאה\כתיבה של קבצים במחשבים מרוחקים
* יישום נוח וקל גם למשתמשים לא מנוסים
  1. רציונל

המוטיבציה שלי לפיתוח הפרויקט נבעה מרצוני ליצור כלי ידידותי למשתמש וקל לשימוש שיאפשר שליטה על מחשבים בסביבת העבודה ובכך יעזור להאיץ את קצב העבודה ויקל את התקשורת בין המחשבים. עבור קבוצות של עובדים במשרד שמפתחים פרויקט כלשהו, חוסר תיאום יכול להיות אתגר מעכב ובמיוחד אם הוא לא מתאפשר בזמן אמת. הפרויקט שלי מקל את העניין על ידי יצירת רשת שדרכה ניתן לראות על מה כל אחד עובד (מבחינת קבצים) וגם יכול לעזור למחשב בצרה, לדוגמה כאשר תהליך קרס ותקע את המחשב, יכול משתמש אחר להרוג את התהליך מרחוק ולמנוע מחיקה או הריסה של קבצים. אני מדמיין את עצמי בעתיד עובד בקבוצה ואפליקציה שכזו כן יכולה לעזור ולבסס את התקשורת בין חברי הקבוצה ובכך להקל רבות בנייה של פרויקט. ואיפה הזווית האישית?

באמצעות הפרויקט אני מקווה להגיע להבנה עמוקה יותר בנושא מערכות הפעלה ושליטה מרחוק, אך גם בנושא התקשורת לאור השימוש הרב בפרוטוקולי תקשורת בין המחשבים שהשירות שלי דורש. בנוסף ,אני מקווה שאוכל לשפר את יכולות התכנות שלי והעיסוק בתחום המחשבים.

* 1. קישור לחומר הנלמד

הפרויקט עוסק במגוון רחב מהתחומים שנלמדו במסגרת שיעורי הסייבר. תחום מרכזי אשר מתבטא בפרויקט הוא תחום מערכות ההפעלה. בשיעורים אנו לומדים לעומק על הדרכים בהן מתפקדות מערכות ההפעלה, אך בגלל שלא כל החומר שאני זקוק לו בפרויקט נלמד בכיתה, נדרשה למידה עצמית בבית שבעזרתה הגעתי להבנה עמוקה יותר של נושאי הפרויקט. גם תחום התקשורת משחק חלק מרכזי בפרויקט שכן התקשורת בין המחשבים הרי מתבצעת בעזרת פרוטוקולים רשתיים כגון TCP ו-RPC, בשיעורים למדנו כיצד מתפקדת התקשורת בין מחשבים ולכן לנסות להבין את הדרך שבה עובדים הכלים שבהם נעשה שימוש בפרויקט היה אתגר לא קש כמו שיכל להיות.

ומה לגבי תחום התקשורת?

1. תיאוריה
   1. תיאוריה

הפרויקט עוסק בתחום המקשר בין התקשורת למערכות ההפעלה, בעיקר Windows המאפשרת שימוש במנגנון שנותן לשלוט מרחוק במחשבים המחוברים (WMI) Windows Management Instrumentation . WMI הוא שירות שמערכת ההפעלה Windows מספקת החל מגרסה Windows 2000 המאפשר מגוון של אפשרויות לשליטה על מחשב מרחוק בעת היווצרות התנאים המתאימים. לדוגמא, בעזרת הפקודה המתאימה ניתן לקבל את הרשימה של כל התהליכים הרצים על המחשב המרוחק ואף לסיים את פעולתם. מהם התנאים המתאימים?

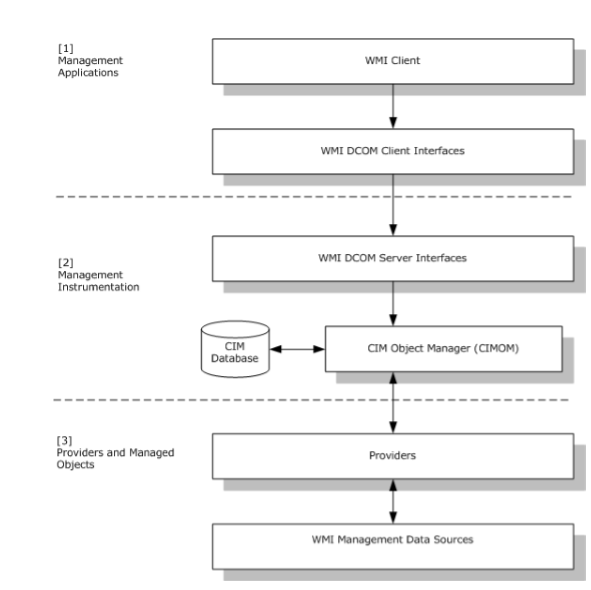
WMI מהווה ממשק מידע עבור שפות תכנות כדי לגשת, לקרוא ולשנות מידע של מחשב אישי או שרת גם בצורה מקומית וגם בצורה מרוחקת. הוא מתפקד כתאום של Web Based Enterprise Management (WBEM)שבדומה ל WMI, מבוסס על Common Information Model (CIM). WMI מהווה חלק מה -DCOM (Distributed Component Object Model) – מבנה שמאפשר לתוכנות על מחשב אחד לרוץ על מחשב אחר.

הפרוטוקול שמנהל את WMI בנוי בארכיטקטורה תלת שלבית:

1. שלב הלקוח – שלב זה הוא המקום שממנו נשלחת הפקודה הראשונה, השאילתה שתלווה אותנו לאורך כל הדרך להשגת המידע. הבקשה נשלחת מהלקוח (לדוגמא cmd עם wmic) אל ממשקי ה-WMI DCOM תחילה אל הממשק של הלקוח וישר לאחר מכן לצד הממשק DCOM של השרת. ממשקים אלה הם היחידים שמנוהלים דרך הרשת בזמן שכל שאר המעברים בין רכיבים קורים כבר בתוך המחשב וברמות ה – WMI האחרות במחשב. לא ברור.

2. שלב ה -CIM - משם עוברת הבקשה ישר ל – CIMOM, שאחראי על בנייה וניהול אובייקטים מסוג CIM. לאחר בניית אובייקט ניגש ה – CIMOM למסד הנתונים של CIM ושולף את המידע לפי השאילתה. דורש קצת יותר הרחבה על המושגים

3. שלב ה – Managed Objects – בשלב זה פונה ה – CIMOM אל ה – ??Providers ואל ה – WMI Management Data Sources ומכין את ה – Management Objects להחזרה. לאחר הבנייה, האובייקטים מוחזרים דרך CIMOM אל ממשקי ה-DCOM וישר ללקוח.



הפרוטוקולים של DCOM רצים מעל RPC, או בשמו המלא – Remote Procedure Call שמבסס את האפשרות למחשב אחד לקרוא לפעולה על מחשב אחר כאילו היא הייתה מקומית במקורה. פרוטוקול ה-RPC מבצע את ההתחקות הדרושה לביצוע הפקודה בזמן שה-DCOM בעצמו אחראי על ההעברה הנכונה של המידע.

סקירה טובה (למעט המושגים הדורשים הבהרה נוספת), מה שיכול לעזור זה דוגמא ספציפית, אם אתה יכול לקחת פקודה פשוטה ב REMOTE לשרטט אותה בין השלבים זה יכול היות יותר מובן.

* 1. מוצרים קיימים

<https://www.techinline.com> – FixMe.IT קישור למוצר דומה: FixMe.IT

FixMe.IT הוא כלי המאפשר שיתוף של קבצים בין מחשבים, תקשורת צ'אט, שיתוף מסך והקלטה של המסך בעת החיבור. הפעלת המערכת מתבצעת בעזרת אימות קוד באינטרנט על ידי המשתמש שרוצה להשתלט, ה-Expert, אשר שולח בקשה למשתמש המרוחק שיוריד את ה- Client Side וייצור חדר עם ID ספציפי, אשר ישמש את ה-Expert כדי להתחבר ל-Client. למטרת אבטחה, האפליקציה משתמשת בפרוטוקול Remote Desktop proprietary המועבר מעל SSL או TLS שמשתמש בהצפנת – 256 bit AES encryption.

המערכת הזאת אכן מספקת מגוון רחב של אפשרויות אך יש בה מספר בעיות נוחות. ראשית, העובדה שהתקשורת מתבצעת דרך אתר באינטרנט כבר עלולה להקשות בתהליך החיבור בין המחשבים, שהרי בעיות בדפדפנים וסיבוך בהרשאת הגישות של חומת האש מסבך את כל התהליך. יתר על כן, המערכת לא נותנת נתוני אבחון על המחשב ללא בדיקה ידנית של המערכת – זו בעיה שעלולה למנוע את החיבור בין המחשבים שכן בעת היווצרות בעיה יכול להיות שנתונים אלה יכלו לעזור.

לעומת הכלי שמוזכר לעיל, לפרויקט שלי יש היבט יותר Low Level. בפרויקט שלי הדגש הוא על ניתוח המידע שנאסף ממחשבים השונים ולא בהכרח התממשקות ישירה ומלאה למחשב המטרה. בנוסף, העבודה שבפרויקט שלי נעשה שימוש בשפה עילית והממשק משתמש יותר בסיסי מאפשר למשתמש הגיע להבנה יותר עמוקה של הכלי ושל הדרך בא האפליקציה מתפקדת. השפה העילית מתפקדת בצורה דו כיוונית: מצד אחד היא נוחה לשימוש בגלל שלא נדרש syntax מסובך כדי להגיע לתוצאות ומצד שני ניתן לראות דרך הפקודה ישירות את כל הפרמטרים (הרי אנו מספקים אותם).

|  |  |  |
| --- | --- | --- |
| קריטריון | הפרויקט שלי | Fixme.IT |
| גישה ישירה לנתוני מחשב | **V** | **X** |
| שליטה קלה בתהליכים | **V** | **V** |
| שיתוף קבצים בזיכרון משותף | **V** | **X** |
| שיתוף מסך ושליטה שותפת במקלדת ועכבר | **X** | **V** |

ומה? אין אף פרמטר בו ל FIXIT יש יתרון? מה לגבי גישה מכל מקום ללא צורך בהתקנת האפליקציה ובטח יש עוד יתרונות...

1. תוצר סופי
   1. תיאור הפרויקט

בעת הרצת האפליקציה, יפתח חלון לכתיבת פקודות. בחלון יופיע ה-directory בו יושב קובץ ה-exe של האפליקציה שמראה מאיפה אפליקציה רצה למה זה חשוב? ותינתן האפשרות לכתוב פקודות בחלון. ההתחברות לשרת מרכזי מתבצעת בעת פתיחת האפליקציה, המשתמש קורא את מידע ההתחברות מפקטות שמפורסמות על ידי השרת, פותח socket ומתחבר.

לאחר כתיבת הפקודה ושליחתה תתקבל כפלט ההודעה המתאימה עבור הפקודה מתחת לשורת הפקודה. הפקודות ייכתבו בשפה עילית וגרפית אשר תאפשר התמודדות עם שגיאות כתיב או שמות דומים לשמות המקוריים של הפקודה – זוהי עבודתו של ה-Interpreter של שפת הממשק. ניתן גם להשתמש בכפתורים ולא לכתוב פקודות כדי לממש את רוב הפקודות של הפרויקט

* ניתן לקבל מידע על המערכת המקומית וגם על כל המחשבים המחוברים, כגון תהליכים שרצים כרגע, מידע פיזי וגם ביצוע פעולות במחשב כגון סיום תהליכים.
* ניתן יהיה לקבל מידע מכמה מחשבים ביחד ולבצע את אותן הפעולות עליהם בו זמנית, לדוגמא? מה לגבי איחוד נתונים?
* שמירת קבצים בכונן וירטואלי משותף וגישה אליהם ממחשב מרוחק
* זהו? יש עוד.

יש לפרט את כל הפקודות הנתמכות ומה הם עושות.

* 1. אלגוריתמים עיקריים

1) התחברות לקוח לשרת

הלקוח שולח לשרת בקשה להתחברות דרך socket, מאיפה הוא יודע IP PORT? יש לפרט את מנגנון ה SCAPY UDP... השרת מקבל את הבקשה וממשיך להודעה הבאה. בזמן הזה בלקוח נפתח חלון LOGIN ובו צריך הלקוח להכניס את פרטי ההתחברות: כינוי וסיסמה. הסיסמה מוגדרת בהתחלת ריצת הסרבר על ידי האדם המריץ. מה הכוונה מראש? איפה? נשמרת ב DB? ניתן לשנות? כשהלקוח מסיים לכתוב את הפרטים, שולח לשרת והשרת בודק ומחזיר תשובה. אם הפרטים נכונים השרת מודיע ללקוח שהוא יכול להתחיל לשלוח פקודות. לאחר מכן השרת יוצר אובייקט מסוג Client ומכניס לתוכו את כל נתוני ההתחברות שהוא צריך על מנת לשלוח ללקוח הודעות בעתיד. לאחר ההתחברות פותח השרת עוד socket עם הלקוח ובו מועברות הודעות כלליות כגון הודעות על משתמשים אחרים והודעות מהשרת. בשלב זה הלקוח מוכן לתקשורת בינו לבין השרת ויכול להתחיל לשלוח פקודות. כיצד PYTHON C# משתלב? IRON PYTHON? PROCESS?

2) אלגוריתם הגרפיקה של הלקוח

לאחר שהלקוח מחובר נפתח לו מסך פקודות שבו ניתן לכתוב פקודות ומתחתיהן תיבת טקסט בה נכתבות תגובות השרת. בנוסף, יכול הלקוח ללחוץ על כפתור ה -File Manager שמאפשר לו גישה לזיכרון המשותף של כל הלקוחות עם השרת. אם הלקוח כותב פקודה ושולח אותה לשרת, הפקודה נשלחת דרך ה -Socket , מנותחת על ידי השרת ולאחר מען השרת שולח תשובה. אם יבחר לפתוח את מנהל הקבצים, תחילה תרוץ פקודה ברקע שתישלח הודעה לשרת לבצע סריקה על כל המשתמשים המחוברים ובהתאם לתשובה ייפתח החלון עם רשימת כל המחשבים וכל התיקיות המשותפות בהם. אם יבחר המשתמש בקובץ או תיקיה מקומיים, תרוץ פקודה פשוטה של פתיחה שמכילה בתוכה התחלת תהליך עם ה – path של הקובץ. אם נבחר קובץ לא מקומי, תישלח פקודה לשרת להעתיק את הקובץ על מחשב הלקוח לתיקיה שרומזת על המקור של הקובץ וממנה תיקרא שוב הפעולה שפותחת קבצים.

3) חיבור ל – WMI וביצוע פקודה

לאחר ניתוח פקודה תקינה מתבצע חיבור ראשוני למחשב המטרה בעזרת ManagementScope. לאחר מכן, נשלחים הפרמטרים לפונקציה שקובעת איזו פעולת WMI צריך השרת להפעיל. הפונקציה נעזרת ב -Dictionary בין מחרוזות שמות הפקודות לביןdelegates של הפקודות בעצמן כאשר המחרוזת משמשת כמפתח וה -delegate כערך – הערך הראשון במערך הפרמטרים משמש כמזהה שם פקודה וכך מועברים הפרמטרים המתאימים לפעולה עצמה. בסוף כל פעולה מוחזרת מחרוזת שמעידה על הצלחת או כישלון הפעולה שמוחזרות ללקוח.

מקבל פקודה מהלקוח

מפצל לפי רווחים

מוציא את הערך הראשון

לא תקין

תקין

האם שם הפקודה תקין?

לוקח את שם המשתמש

האם שם המשתמש הוא שם מחשב או וכינוי

מחפש את הפקודה הכי דומה לפי אורך ותווים מתאימים

כינוי

שם מחשב

שולח חזרה למשתמש

מחפש את שם המחשב לפי הכינוי ברשימת המשתמשים

דוחף לתוך מחסנית הפרמטרים

הפלט מוצג למשתמש בקונסולה

לוקח את כל שאר הפרמטרים לפי פיצול הגרש שנמצא לפני כל פרמטר

ממיר את המחסנית למערך

מחזירת את המערך

4) Interpreter – מנתח פקודות

על כל פקודה שמקבל השרת מופעל תהליך הניתוח שבו הפקודה מפורקת לפי רווחים וסימנים ונדחפת לתוך מחסנית בסדר של מילות המפתח שמופיעות בפקודה. נעשות בדיקות על תקינות שם המחשב (כולל תרגום בין שם משתמש לשם מחשב) ושם הפקודה. אם שני הפרמטרים תקינים, המחסנית מומרת למערך באותו הגודל ומוחזרת.

אם שם הפקודה אינו תקין מוחזרת מחרוזת שמעידה על אי-תקינות שם הפקודה ותופעל פעולה שמחפשת את הפקודה שאליה יכול היה הלקוח להתכוון על ידי השוואת אורכי פקודה ותווים, ומספר תווים מתאימים לפקודות המובנות במערכת. במקרה של אי תקינות שם המחשב יישלח גם כן חיווי מתאים

* 1. דרישות ומגבלות מערכת

חסר – מה האילוצים המקובלים ומה הדרישות הרלוונטיות בהפעלת המערכת

* 1. ממשקים למערכות חיצוניות

אין למערכת ממשק עם מערכות חיצוניות.

* 1. התייחסות לנושא אבטחה

אבטחת הכניסה מתבצעת על ידי סיסמא שמתבקש כל client להקליד בעת כניסתו למערכת. לשרת יש את הזכות לקבוע סיסמא כשהוא עולה וסיסמא זו היא המאמתת את כניסתם של המשתמשים. כשהשרת עולה, ניתנת האפשרות של לאדם המריץ את הקוד לבחור בסיסמה שתהיה תקפה כל עוד הסשן העכשווי של השרת רץ. כשהמשתמש פותח את חלון הכניסה, עליו להקליד את השם שהוא בוחר למשתמש ואת הסיסמא, והשרת מוודא שהסיסמא מתאימה לזו השמורה: עבור התאמה יפתח ממשק הפקודות, ובהעדרה – חיווי על טעות וניסיון חוזר. לא ברור מנגנון אימות הסיסמא ובחירתה / שינויה ע"י המשתמש

האם יש הצפנה בפרויקט? האם נדרש לדעתך?

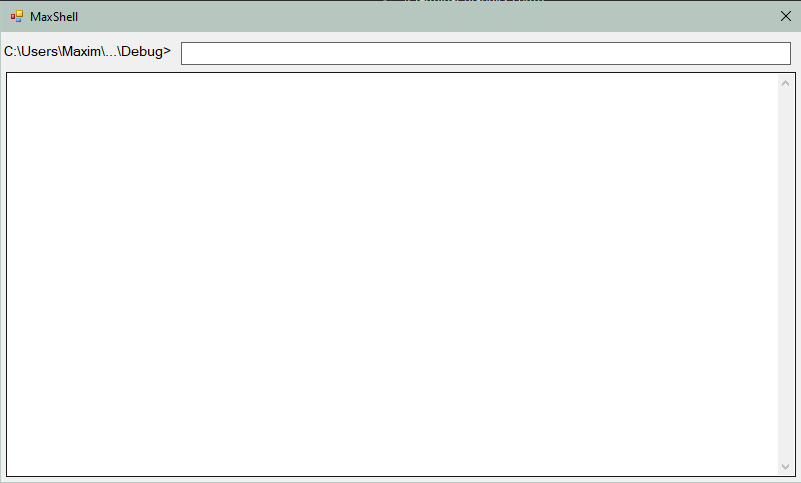
* 1. ממשק משתמש

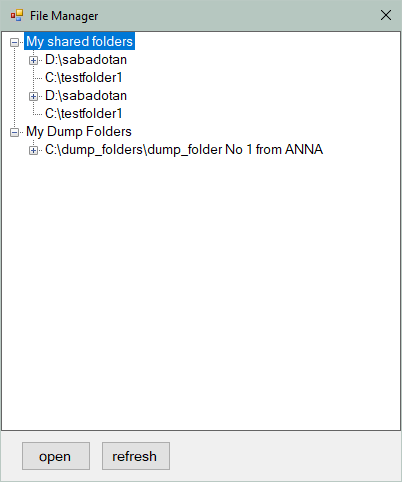
שים לב שהרבה יותר טוב לראות חלון מלא בפרטים ולא חלון ריק

1) חלון התחברות ראשוני

בחלון מוצגת שורה ובה ניתן להקליד את השם שהמשתמש יבחר. עם הקלדת הסיסמא יוכל המשתמש ללחוץ על הכפתור אישור או ללחוץ enter על המקלדת על מנת להיכנס. אם המשתמש הקיש סיסמא נכונה, השרת יאשר אותו ויפתח חלון ממשק פקודות, אחרת תוקפץ הודעה על הטעות והוא יוכל לנסות שוב. חסר צילום מסך

2) חלון קונסולה ראשי

לאחר ההתחברות נפתח חלון ממשק הפקודות ובו שורה שבה ניתן להקליד ושדה text חסום לכתיבה שבו יוצג מידע מהשרת. המשתמש יכול להקליד פקודות, לשלוח באמצעות הכפתור enter ולקבל חיווי תואם מהשרת. בנוסף, המשתמש יכול לפתוח את חלון סייר הקבצים באמצעות הכפתור בתחתית המסך או על ידי הפקודה file manager. רשימת פקודות זמינות ניתן להשיג על ידי הפקודה help. אין בצילום המסך את כפתור ה FILE



3) חלון סייר הקבצים

לאחר שחלון זה נפתח מוצג עץ שמתפצל למספר ענפים לפי מספר המשתמשים המחוברים. תחת כל משתמש מוצגת רשימה של התיקיות המשותפות של אותו המשתמש כאשר הענף הראשון מציג את התיקיות של המשתמש המקומי ומתחתיו שאר התיקיות של המשתמשים האחרים. בתחתית הרשימה ניתן למצוא את ענף ה-dump folders. בענף זה נמצאים כל הקבצים שהועתקו ממחשבים אחרים. כאשר לכל תיקיה יש שם שמראה מאיזה משתמש\מחשב היא באה. פתיחה של תיקיה או קובץ ניתן לבצע באמצעות הכפתור open לאחר שסימנו את המטרה.

במקרה שננסה לפתוח קובץ על מחשב מרוחק, הוא תחילה יועתק לתוך dump folder ולאחר מכן יפתח אוטומטית. בנוסף, נמצא גם כפתור refresh שתפקידו לרענן את המידע של החלון. הוא מריץ את פקודות בניית החלון מחדש ועל ידי כך בודק שינויים ומכניס אותם.

איפה כל הגרפיקה?

* 1. תרחישים עיקריים

1) התחברות לקוח לשרת

|  |  |
| --- | --- |
| **שרת** | **לקוח** |
| עולה, מפרסם את הכתובת בעזרת scapi UDP?? שלו ומתחיל להאזין |  |
|  | קורא את המידע מהפקטות ומתחבר לשרת. |
| מקבל את ההתחברות ושולח הודעה על אישור login |  |
|  | מקליד את השם ואת הסיסמא ושולח לשרת |
| מאמת ושולח ללקוח הודעה שהוא יכול להתחיל לשלוח פקודות |  |
| מוסיף את השרת לתור הלקוחות |  |

יש כנ"ל לתמוך בקביעת סיסמא, שינוי סיסמא וכו

חסרים עוד תרחישים, שים לב שהתרחיש שציינת בנושא תרגום פקודה הינו אלגוריתם ולא תרחיש. תרחיש הוא כזה שיש לו קלט ופלי של לקוח. הקשת פקודה, העתקת קובץ ועוד. תחשוב ותוסיף.

1. תהליך כתיבת הפרויקט
   1. תהליך הפרויקט

היה לך פרויקט קודם ?!

כתיבת הפרויקט התחילה מניסיון לכתוב את ממשק הפקודות באופן מקומי ולאחר מכן התפצל לשרת לקוח כאשר רוב הלוגיקה נמצאת בשרת והגרפיקה אצל השרת. בתחילת הדרך עבדתי במעבדה שבה התנאים היו מושלמים לעבודה, אך עם המעבר ללמידה מקוונת, נדרשתי לבנות את הסביבה בבית, וזה יצר קושי גדול. למה? לקח מעבר משבוע והרבה תקשורת עם המורה לסייבר, טכנאי בית הספר וחברים שעזרו לי במשך כל הדרך ותמכו בי עד שהצלחתי בסופו של דבר להקים את הסביבה.

בתחילת הדרך היה קושי התמודדות עם git אך למדתי להשתמש גם בכלי זה והוא אפילו עזר לי רבות והוציא אותי מבעיות ותקלות שקרו במהלך הדרך. עבדתי במעבדה כמה שיכולתי ועל כל הדברים שיכולתי לעשות לפני השינוי במצב ולאחר מכן נאלצתי להתמודד עם האתגר. לאחר פתירת הבעיה חזרתי לתכנות הרגיל והרגשתי הקלה מהחזרה לחלק היותר מעשי של הפרויקט – כתיבת קוד ופיתוח.

כמובן שגם נאלצתי להעמיק בתיאוריה מאחורי הקוד כדי להבין איך הקוד מתפקד וגם כדי לפתור באגים ובעיות בדרך. החקירה של הנושא הייתה מעניינת כי גיליתי כמה שהנושא של WMI משמעותי, כמה הוא חשוב למערכת המחשב, כמה דברים מושפעים ממנו ומכמה גורמים הוא תלוי בעצמו – זה עזר לי להבין הרבה יותר טוב את הממדים של המכונה המורכבת שמולה אני יושב.

לאורך כתיבת הפרויקט שמתי לב איך יכולות הכתיבה שלי משתפרות, איך אני שם לב מתי להשתמש באיזה ביטוי, איך לכתוב אלגוריתמים בצורה יותר יעילה ואפילו שיניתי הרגלים ישנים לכתיבה. אני מאמין שעם כל פרויקט היכולות כתיבה שלי השתפרו, משתפרות וישתפרו וזה נכון במיוחד עבור פרויקט גדול ומשמעותי זה.

* 1. אתגרים ואופציות שונות למימוש
  + Interpreter – הדרך שבה הוגדר מנתח הפקודות הייתה בהתחלה גנרית והיה קשה להגיע לתוצאה מספקת ולרעיון שיאפשר את רמת הנוחות המקסימאלית עבור המשתמשים. לכן, התייעצתי הרבה עם אנשים שונים כדי להגיע להחלטה על הדרך הכי נורמלית שבה תוצג פקודה. לבסוף הוחלט הרעיון של פיצול הטקסט בין מילות מפתח וזו לדעתי אחת הדרכים הכי יעילות ונוחות לניתוח טקסט גנרי שלא לפי תווים מוגדרים מראש.

דרך אחרת למימוש הייתה פיצול הפקודה לפי רווחים ובדיקה איטרטיבית על כל חלק בנפרד – במקרה של פרמטר עם רווח ניתן היה לקבוע מלכתחילה שצריך להוסיף גרש לפני ואחרי פרמטר וכך לחבר כך. טכניקה זה אמנם יכולה לעבוד אך דורשת מהמשתמש דיוק בפקודות.

* + WMI – ההבנה איך עובד הכלי דרשה זמן, התרגלות ואפילו שכתוב של פקודות בדרכים שונות על מנת להבין בדיוק איך לעבוד אתו. ישנם הרבה פרמטרים שWMI- תלוי בהם, הרבה התעסקות במחשב כדי לגרום לו לעבוד. לא היה ניתן לגרום לכלי לעבוד בין מחשבים ללא התעסקות במערכות וכלים ניהוליים של מערכת ההפעלה וכמו כן בכמות לא מבוטלת של הגדרות.

ישנם כלים זמינים אחרים דמויי WMIכגון Spiceworksוניתן גם להוציא מידע באמצעות ממשק הפקודות ולא דרך System.Management. ניתן היה להוציא את המידע ישירות מממשק הפקודות ולעשות עליו regex אך אני רציתי לנסות לעבוד עם הקוד ולחסוך לעצמי את ה-string manipulation שגם ככה יש ממנו כבר מספיק בפרויקט.

מה לגבי הגרפיקה?

1. מרכיבי פתרון
   1. תיחום הפרויקט

* תקשורת – תקשורת שרת לקוח מתקיימת באמצעות TCP, פרוטוקולים של WMI מתבצעים בעזרת DCOM (RPC)
* מערכת הפעלה – Windows 10
* תצוגה – Windows Forms
* בסיס נתונים לשמירת הסיסמאות?
* שימוש במחלקות ותיעוד בקוד.
  1. סביבת העבודה (טכנולוגיה)

**קוד המערכת** נכתב ב-C# בגלל ש-,WMI השירות המרכזי עליו בנוי הפרויקט שלי, נוח לשימוש והכי מפורט ומובן בשפה זו.

הגרפיקה גם נכתבה ב-C# WinForms כדי לא להסתבך עם שילוב בין קודים בשפות שונות העדפתי לעבוד דווקא עם. WinForms בנוסף, למדנו WinForms בכיתה ולכן הרגשתי שליטה וקירבה גדולה יותר לסביבה זו.

**סביבת פיתוח**: Visual Studio לשפת C#

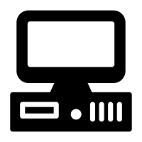
האלמנט היחיד שלא נכתב ב-C# הוא ה - scapi של הפקטות שמעבירות את נתוני ההתחברות לשרת וזה נכתב ב -python .

סביבת פיתוח: PyCharm

* 1. מבט טופולוגי

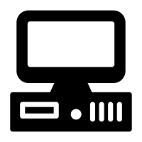
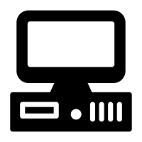
המערכת בנויה ממעגל של מחשבים כאשר בכל פעם, אחד מהמחשבים משמש כשרת בזמן שהשאר משמשים כלקוחות. הכול מתבצע בשיטת peer to peer ולכן כל המשתמשים באותה רמה. DB

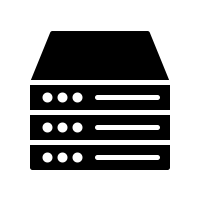
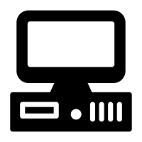
user



user

user





server

user

* 1. מבני נתונים
* תור משתמשים – עבור כל משתמש חדש שמתחבר נוצר אובייקט מסוג Client ובו נשמרים הנתונים אודות הלקוח: Socket, כינוי ושם מחשב. לכל פעם שנדרשת סריקה של כל המשתמשים עבור חיפוש של משתמש, פעולה שאמורה להתבצע על כולם או הודעת תפוצה לכל המשתמשים המחוברים.
* מילון פקודות – לאחר סיום פירוש פקודה, במקרה שנמצאה כתקינה יישלחו הפרמטרים להוצאת הפקודה מהמילון. במילון מחרוזות שמות הפקודות משמשות כמפתחות בעוד ש – delegates של הפקודות משמשים כערכים. כשהמערך נשלח לפעולה, הערך הראשון הוא שם הפקודה ולכן משווה עם המפתח ובהתאם מוצא הערך לאחר ביצוע הפקודה עם הפרמטרים הנוספים.
  1. מבט מודולרי

להלן החלקים העיקריים בפרויקט:

שרת:

* + Client – אובייקט שמייצג כל משתמש שמחובר למערכת, פרט לשרת עצמו. עבור כל משתמש שמתחבר לשרת נוצר אובייקט חדש ונוסף לתור הלקוחות.
  + Server – מחלקה שבה מנוהל כל הפירוש של הפקודות שמגיעות לשרת ושליחתם לביצוע על ידי ה-WMI.
  + Program – המחלקה הראשית שבה מנוהלת ההתחברות וניהול ה-sockets של כל המשתמשים המחוברים וכמו כן העברת הנתונים לניתוח על ידי השרת.
  + WmiFuncs – מחלקה שבה נמצאות כל הפעולות WMI של השרת: יצירת כל האובייקטים טרם ההתחברות, ההתחברות עצמה וביצוע הפעולות השונות.

לקוח:

* + Program – מחלקה שבה מתבצע ניהול ה – sockets של המשתמש וגם כן ניתוח פקודות מסוימות טרם שליחתם לשרת כגון file manager, disconnect וכו'.
  + Login\_Windows – מחלקה האחראית על בניית חלון ההתחברות ועל אימות הנתונים ששולח המשתמש על מנת להתחבר.
  + Shell – המחלקה האחראית על בניית חלון הקונסולה הראשי וכל הפונקציות שלו. היא גם אחראי על הצגת המחרוזות המתקבלות מהשרת בצורה מסודרת בתוך תיבת הטקסט של הקונסולה.
  1. פירוט מודלים עיקריים

: Server

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Input/output | Function | Class |
| Starts the app, sets up the server sockets and starts listening | Input: None  Output: None | Main | Program |
| Runs on a thread when a client connects, Maintains the server-side socket for the client, sends and receives messages and passed received messages for interpretation before sending a reaction | Input: Socket s  Output: None | KeepInTact |
| If a message is longer than 4096 bytes, this function makes sure every part is sent by dividing and sending the messages in 4096 bytes maximum till the end of the message | Input: string fullstring, Socket handler  Output: None | SendPackets |
| Constructs a client object with the given parameters and add the needed info to the Client queue and Socket dictionary | Input: Socket cl, string nickname, string machname  Output: None | Client (ctr) | Client |
| Upon adding a new client to the data structures, this function checks that the info is correct and creates a new instance for the client | Input: Socket, socket, string mach, string nick  Output: None | CheckAndAdd |
| Receives the message from the socket, sends for interpretation and then to execution of the command. It returns the output from the result of the executed command | Input: None  Output: None | SetCommands | Server |
| Starts the app, sets up the server sockets and starts listening | Input: string command, Socket s  Output: string | CommandOutput |
| Receives the full command from the server, checks for errors, divides it by keywords and transforms it into an array of parameters ready for execution | Input: string command  Output: string[] | Interpreter |
| If a command name sent by the client is not matching any of the server's commands, this function searches the most similar one | Input: string input, string[] commands  Output: string | MostSimilar |
| Gets the paths and network paths of all shared folders on all the registered clients and builds a dictionary | Input: None  Output: None | SetPaths | WmiFuncs |
| Builds and connects the management scope before an action is invoked | Input: string target  Output: None | RemoteConScope |
| Returns the ip of the server by searching up all the ip's of the computer and choosing the one associated with the socket | Input: None  Output: string | GetIp |
| Returns the free space on each of the machine specified by the parameter | Input: string target  Output: string | FreeSpace |
| Returns the list of all running process on the machine specified in the parameter | Input: string target  Output: string | ShowProcess |
| Performs the "terminate" method for a process on a target machine as specified by the parameters | Input: string target, string targetprocess  Output: string | KillProcess |
| Performs the "create" method to start a process on a target machine as specified by the parameters | Input: string target, string procname  Output: string | RemoteProcess |
| Checks whether a folder exists and creates if not, performs the "create" method on a folder to set it as shared on a target as specified by the parameters | Input: string target, string sharefolder  Output: None | ShareFolder |
| Lists all the files inside a folder on a directory specified in the parameters | Input: string target, string path  Output: None | ListFiles |  |
| Lists all the shared folders of all connected clients and server. | Input: None  Output: string | ShowFolders |
| Copies a file from a source path to the destination path, the socket is passed n order to get the source machine to which the file should be copied | Input: Socket src, string target, string srcpath  Output: None | CopyFile |

Client :

|  |  |  |  |
| --- | --- | --- | --- |
| Sends the nickname for a press of the "choose" button and closes the login window | Input: None  Output: None | SendNick | Login\_Window |
| Sends the nickname for a "enter" button press and closes the login window | Input: None  Output: None | SendNick\_2 |
| Presents the output in the shell textbox in an organized way and takes care of command memory of the console | Input: None  Output: None | OutputProcedure | Shell |
| Recursive presentation of the files and directories in each path given as parameter, the TreeNode parameter is the one from which all new child nodes diverge for the current directory | Input: string dir, TreeNode node  Output: None | ProcessDirectory | File Manager |
| For remote folder, this function sends a message to list all files in a remote folder | Input: string data  Output: string | AddInsides |
| The function which decides how to open a file, if it's local – starts the process right away, if it's remote – copy and then start process | Input: None  Output: None | OpenFile |
| Restarts the window, including the rebuild of all the data structures | Input: None  Output: None | Refresh |
| This function decides how to build each node, if it's a new computer, it creates a new node, and if it's a folder inside a folder it, builds the tree accordingly | Input: string[] direcs  Output: None | BuildTree |
| Shows all the dump folders on the computer (the place to which files get copied) | Input: None  Output: None | ShowDumps |

1. תסריטי בדיקה
   1. דגשים בבדיקה

* חיבור והתנתקות של לקוחות מהשרת ללא באגים
* ניתוק לקוח בעת סגירת שרת ללא הקרסה של הלקוח וחיווי מתאים
* טיפול במספר משתמשים
* הפעלת מספר פקודות WMI במקביל
  1. תסריטי בדיקה עיקריים
* חיבור 2 משתמשים בעלי אותו שם משני מחשבים שונים
* שליחת 2 פקודות בו זמנית מ-2 משתמשים שונים
* סגירת השרת בצורה ידנית באופן פתאומי (לפני, בזמן או אחרי פקודה שמתבצעת)
* בדיקת פקודות ספציפיות על פי מפרט התקשורת
* ויש עוד

1. רפלקציה
   1. לוח זמנים מוערך לניהול הפרויקט:

|  |  |
| --- | --- |
| ינואר | אלגוריתמים ופקודות בין מחשבים + הצגת קבצים בתיקיות מרוחקות |
| פברואר | כונן משותף לשמירת קבצים, אינטרפרטר, ניהול משתמשים |
| מרץ | גרפיקה ונגישות |
| אפריל | תיקון באגים |

לוח הזמנים צריך להיות אמיתי.

* 1. אתגרים ותרומה אישית

תהליך הכתיבה היה מעניין, מאתגר, וגרס הרבה שעות לא רק של תכנות אלא גם של חקירה באינטרנט על דרכים לטיוב הקוד ושיפורו. חקירה וחשיבה לפני שאני ניגש לתכנת עזרו לי להאיץ את קצב הכתיבה שלי ובכך גם את קצב ביצוע המטלות.

לאורך כל הפרויקט, כל בעיה שהייתי צריך לטפל וכל feature שתכננתי להוסיף נכתב בקובץ טקסט מסודר לפי אבני דרך ורמות קושי של ביצוע ולכן לא היה מצב ובו ישבתי מול הקוד בחוסר ידיעה על הצעד הבא. כך גם הרגשתי את הסיפוק במחיקת סעיפים שביצעתי אך גם הארגון של המטלות ובחירה במה להתעסק מתי לפי התחושה שלי והיכולת שלי לתכנת ולחשוב באותו הרגע. בבוקר הייתי משתדל לפתור את הבעיות המסובכות בעוד שבערב הייתי מתעסק עם נושאים יותר משניים וקלים לביצוע.

קביעת deadlines, והרצון שלי לעמוד בהם היוו את הגורם העיקרי שלי למוטיבציה להתקדמות בפרויקט. העמידה בזמנים אמנם טיפה מלחיצה, אך מאפשרת תכנון נכון ומושכל של הזמן וגם מעלה את רמת המוטיבציה כלפי המטלות שאותן ברצוננו לבצע. היו זמנים קשים שבהם ישבתי ימים על בעיה שלא הייתה בעיית קוד אפילו אלא בעיה סביבתית שבגללה לא יכולתי להמשיך לעבוד על הפרויקט אך עם הכוח לעבוד, רצון, זמן ועזרה מאחרים הבנתי שגם עבור בעיות שלאחר ימים רבים הפתרון כבר מתחיל להידמות לחלום באספמיה, הוא ניתן למימוש כל עוד אתה יודע שהדבר אפשרי – זה מה שקובע אם אתה מסוגל או לא.

* 1. תובנות

כתיבת הפרויקט העשירה אותי רבות בתחום התכנות והסייבר. התהליך עזר לי לרכוש ניסיון בתכנות, ובנוסף עזר לי להכיר יותר לעומק את המכונה עליה אני מבלה כל הרבה שעות בשבוע. הייתי צריך להתמודד עם אתגרים לא פשוטים במהלך הדרך ולהתגבר עליהם ולהמשיך הלאה שכן זוהי מהות הפרויקט בפני עצמה. העבודה האישית עזרה לי לפתח ביטחון ביכולות שלי והוכיחה לי שיש ביכולתי לבצע כל מטלה שאני נדרש לה אם רק אני מאמין בעצמי ולא מפסיק להתמיד עד סיומה.

בתחילת הדרך, היסקתי שהפרויקט לא יהווה אתגר גדול, כי חשבתי שניתן למצוא את כל שורות הקוד שאני צריך באינטרנט אך מהר מאוד התברר שאני טועה ובגדול. כאשר החלטתי ללכת על כיוון זה, הייתי בטוח שאגיע לתוצר סופי עם 70% קוד מועתק מהאינטרנט בעוד שבפועל המספר לא עובר את ה-10%. כעת, אני שמח שטעיתי ושכך הוכחתי לעצמי שהפרויקט הזה הוא לא בדיחה וגם עזר לי להבין, גם אם בקטנה את המשמעות האמיתית של עשיית פרויקט, בייחוד בתחום המחשבים אך גם בכללי בפרויקטים שאעשה בעתיד. קצת ישיר מדי...תשכתב.

1. הוראות התקנה ותפעול
   1. תצורה ודרישות קדם

* על מערכת ההפעלה בכל המחשבים להיות Windows ואותה גרסה (עדיף 10)
* על כל המחשבים צריך להיות מותקן python 3
* על כל המחשבים להיות באותה רשת מקומית – DNS, WORKGROUP...
* לכל המחשבים צריך להיות משתמש עם הרשאות מנהל עם אותו שם משתמש וסיסמא
* על כל מחשב צריך להוריד את חסימת WMI מה-Firewall
* אפשור שליטה מרחוק על המחשב ב-DCOM CONFIG וב- WMI MGMT
* אפשור שיתוף תיקיות ברשת בהגדרות המחשב
  1. התקנה

תחילה יש להפעיל את השרת, להקליד סיסמא רצויה, לחכות כמספר שניות עד שהשרת יעלה סופית ויתחיל להפיץ את פקטות נתוני ההתחברות ולאחר מכן ניתן יהיה לחבר כל מחשב ברשת המקומית לשרת כרצוננו.

1. ביבליוגרפיה

במהלך כתיבת הפרויקט הסתמכתי על מספר מקורות מידע:

* דוגמה למוצר דומה – https://www.techinline.com
* הסבר על פעולת המוצר - https://en.wikipedia.org/wiki/Techinline
* הסבר להתקנת WMI - [https://docs.microsoft.com/en-us/window](https://docs.microsoft.com/en-us/window%20)  s/win32/wmisdk/connecting-to-wmi-remotely-starting-with-vista

1. נספחים

אין נספחים