**חלק יבש**

א)נרצה לשנות את הזמן בין פניות של הפונקציה poll לשרת , לשנות את המשתנה הגלובלי בתחילת התכנית לא יעזור, אבל אם נוסיף קטע קוד שלוקח הרבה זמן, יותר מכמה שהתכנית כבר מחכה, אז הוא יגדיל את כמות הזמן בין בקשות של התכנית, לכן נבנה קטע קוד שפשוט ישן לתקופה שהיא יותר גדולה מהמשתנה הגלובאלי, ונדרוס בתחילת הפונקציה קפיצה אליו, את הקטע עצמו נוסיף לקובץ הריצה ואז כאשר יקראו לפונקציה, היא תחכה בהתחלה שלה יותר זמן ממה שהיא אמורה לחכות בין הבקשות, ולכן התדירות פניה שלה לשרת תשתנה.

ב) כלומר נרצה לעשות הוק בהתחלה של הפונקציה ולהפוך את x להיפוך שלו ואז להמשיך את פעולת התוכנית כאילו הכנסנו את ההיפוך של x, אבל אנחנו יודעים שזה בעייתי להכניס בתחילת הפונקציה הזו קוד, ולכן את ההוק נרצה לעשות בסוף הפונקציה,מה שנרצה לעשות זה להפוך את x ואז להפעיל עליו שוב את הפונקציה, מכיוון שאנחנו לא יודעים כמה מקום יש לנו להוק נרצה פשוט להפוך את x ואז לקרוא שוב לפונקציה, לכן מה שנרצה לעשות זה לקחת את כתובת החזרה של הפונקציה, להקטין אותה כך שהיא תחזור שורה אחורה, כלומר למקום שבו קראו לפונקציה, ולפני זה להפוך את x ולוודא שאנחנו עושים את זה רק פעם אחת, כלומר שבקריאה החדשה לפונקציה לא נכנס שוב להוק, בעזרת רגיסטר שהפונקציה לא משתמשת בו או על המחסנית. את כל הקוד הזה נצטרך להכניס איפשהו, ולקפוץ אליו עם הדריסה בסוף הפונקציה, נוכל להכניס אותו בsection נוסף בקובץ הריצה שאליו ההוק יקפוץ, וכמובן שנוסיף גם את הקוד שאנחנו דורסים לשם.

ג)הפונקציה sendf ככל הנראה מחברת את ההודעה למחרוזת אחת ואז שולחת אותה לשרת, כדי לשלוח אותה לשרת היא תצטרך להשתמש בפונקציית ספריה לשליחה על גבי סוקט לשרת(ככל הנראה send מwinsocket ) לכן מה שנרצה לעשות זה לעשות הוקינג לפונקציה הזו כך שלפני שהיא שולחת לשרת היא מצפינה את ההודעה, במקרה והפונקציה משתמשת בהוט פטצינג, נוכל להשתמש באפשרות הזו כדי להזריק פונקציה משלנו שנכתוב על dll (כמו שעשינו בסדנה), או שנוכל להשתמש בIAT ונחליף את הכתובת של הפונקציה לפונקציה שמצפינה את ההודעה ואז קוראת לפונקציית הספריה המקורית.

ד) נרצה בעצם לגרום לפונקציה connect לחכות עד שתופעל הפונקציה parse לפני שהיא מתחילה את התכנית, זה דומה למה שאנחנו עושים כשאנחנו רוצים לדבג ומחכים לדיבאגר ולכן נשתמש באותה השיטה, connect תחכה לדיבאגר ופארס יפעיל דיבאגר על connect כשהוא מתחיל (הוא לא באמת ידבאג אותו, הוא יכבה אותו ישר) וכך שניהם יתחילו לפעול במקביל. נשתמש בפונקציות של DebugActiveProcess וDebugActiveProcessStop כדי להפעיל את הדיבאגר ולכבות אותו מparse ובconnect נחכה לדיבאגר כמו שלמדנו בתרגול, כך ברגע שהדיבאגר יופסק שני הפונקציות יחזרו להתחלה, את הקוד הזה נכניס לשתי פונקציות, אחת לconnect ואחת לparse שנוסיף לקובת הריצה ונדרוס בתחילת connect וparse קפיצה אליהן(נוסיף להן גם את השורה שאנו דורסים).

ה)הפונקציה בודקת במהלך ריצתה את שלמותה ולכן אם לפני ריצתה נכניס את הhook שלנו ואז נדרוס חזרה את השורה היא לא תזהה את השינוי, מה שנרצה לעשות זה לגרום לפונקציה לפני חזרתה להדפיס את הערך חזרה שלה, נעשה את זה על ידי כך שנשנה את כתובת החזרה מהפונקציה לכתובת משלנו ונשמור את הכתובת שמחקנו, ואז בסוף הפונקציה היא תחזור לפונקציה שלנו, בה נדפיס את ערך החזרה(כנראה בeax ) ואז נחזור לכתובת חזרה המקורית ששמרנו, את שתי קטעי הקוד האלה נוכל גם להוסיף לקובץ הריצה(קטע אחד שדורס חזרה את הקפיצה ומשנה את כתובת החזרה ושומר אותה, וקטע אחר שמדפיס את ערך החזרה וחוזר)

ו)נכתוב קטע קוד שמכפיל את eax ונרצה שהפונקציה תפעיל אותו לפני שהיא מחזירה בכל פעם(כי הערך חזרה יהה בeax ) לכן עבור א בתחילת הפונקציה נוסיף קטע קוד שמשנה את כתובת החזרה לכתובת של הפונקציה הזאת ושומר את ערך החזרה הקודם גם במחסנית, ואז בסוף הפונקציה היא תקפוץ לקטע שלנו שמכפיל את eax ומשם נוסיף קפיצה לערך החזרה האמיתי ששמרנו על המחסנית, כך עבור אופציה א בכל פעם שהפונקציה תפעל היא תחזיר ערך כפול. עבור ב, נרצה לעשות אותו הדבר, אבל נרצה שזה יקרה רק בקריאה הראשונה לפונקציה, בשביל לבדוק אם זה הקריאה הראשונה, נשמור על המחסנית גם ערך מסוים שיסמן לנו שהתבצעה קריאה הפונקציה ונשים אותו במחסנית בכל פעם שאנחנו דוחפים לשם ערך חזרה בפונקציה הראשונה, ואז בפונקציה השנייה לפני שאנו מכפילים, נבדוק שאין מעלינו במחסנית את הערך הזה יותר מפעם אחת וככה נדע שזוהי הקריאה הראשונה של הפונקציה.