

בית הספר "אהל שם"

תיק פרויקט מאת זיו כספי:

**The Drone Project**

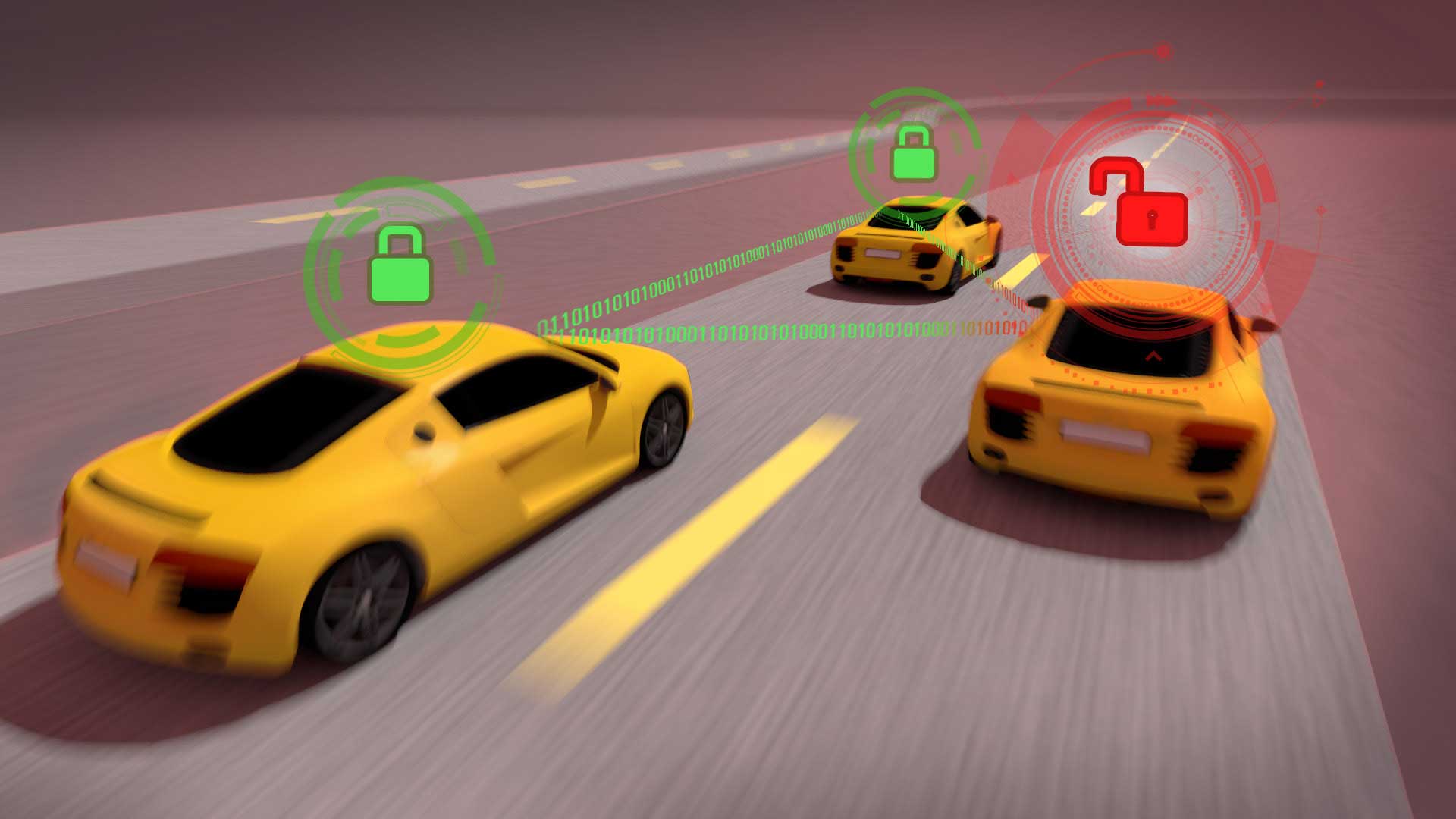
212733133

המנחה תומר טלגם

מגמת סייבר 14.6.2020

*מבוא*

שליטה והגנה על העולם הפיזי באמצעות העולם הדיגיטלי



מכוניות ללא נהג, מטוסים ללא טייס, שערים ללא שומר ואפילו מפעלים ללא פועלים.

בעולם כיום יותר ויותר חלקים קריטיים מחיינו עוברים לידי המכונות, ומתופעלים על ידי תוכנה במקום על ידי אדם.

מגמה זו מגדירה שימוש חדש לאינטרנט ולמחשוב – The Internet of Things. מחשבים הם כבר מזמן לא קופסה כבדה שיושבת בחדר העבודה, וכתוצאה מכך חפצים פיזיים יומיומיים נהפכים לחכמים ומקבלים גישה לרשת האינטרנט בצורה כזו או אחרת. חפצים פשוטים כבר אינם פשוטים ובתים שלמים מנוהלים על ידי מחשבים אלו- מהדלקת האורות והשמעת מוזיקה עד לפתיחה ונעילה של דלת הבית הראשית.

התופעה הזו בשלביה הראשונים והיא כבר עכשיו מקלה על חיי האדם באופן משמעותי מאוד- מה שמצביע על כך שבקרוב עוד ועוד חלקים **פיזיים** מעולמנו ינוהלו על ידי תוכנה, ואיך לא, יחוברו לאינטרנט.

תופעה זו מביאה ביחד עם הנוחות ממד חדש עבור התקפות סייבר. כיום כאשר תוקף מצליח לחדור בהצלחה לרשת, ההשלכות הן כבר לא הדלפת מידע וריגול בעזרת מצלמת הרשת- אלא ניצול לרעה של מכשירים **פיזיים** ברחבי הרשת, דבר שיכול לגרום לפגיעה בפעולות קריטיות של חיינו ואף לפגיעה מוחשית בחיי אדם.

שיבוש רמזורים, סטיית מכוניות, הפסקת המכונות במפעל או פגיעה בהן, שינוי במזג האוויר, שריפה של הטוסטר ואף פגיעה במכשור רפואי (לדוגמא קוצב לב) – כל אלו נשמעים כמו כוחות על, אך כיום אלו הם הדברים שתוקף מיומן יכול להצליח לבצע במכשירי ה- IoT.

כאמור, מגמת ה- IoT מביאה עמה סכנות חדשות ובעלות השלכות חמורות יותר מאשר בעבר ולכן בתקופה כזו חשובה עוד יותר ההתגוננות מפני תוקפים ופושעי סייבר ברשתות עם מכשירי IoT .

המגמה שבעקבותיה התחלתי להסתקרן ולחקור את הנושא היא מגמת המכוניות האוטונומיות. אני עוקב אחרי הנושא כבר זמן רב והוא מסקרן אותי מאוד- הרעיון של מכונית אשר נוסעת בכוחות עצמה בזמן שהנוסע יושב מאחורה נתפס לי כלקוח מסרט מדע בדיוני. אני עוקב אחרי הנושא ובעקבות הלימוד במגמת הסייבר התחלתי להתעניין עוד יותר בחלק הטכני ולחקור יותר לעומק. התרחיש האידיאלי בעיני חברות הרכבים האלו הוא עולם בו כל המכוניות הן אוטונומיות ובתקשורת מתמדת אחת עם השנייה- עולם בלי רמזורים ותמרורים בו מכוניות ידעו לבד היכן נמצאות שאר המכוניות על הכביש ולהתכוונן בהתאם. הכל אוטומטי. הכל מחובר.

**“The key with autonomous is the whole ecosystem. One of the keys to having truly fully autonomous is vehicles talking to each other.” – מארי בארה מנכ"לית General Motors**

רשת מכוניות בקנה מידה עולמי כזו היא תרחיש כבר לא כל כך עתידני ומומחים קובעים שבעוד מספר שנים לא יהיו על הכביש יותר מכוניות רגילות. ככל שלמדתי יותר במסגרת מגמת הסייבר גדלה ההתעניינות הטכנית שלי ויחד איתה גדלו כמות הבעיות שקפצו- מי יתאם בין כל המכוניות האלו? מי יאכוף את אופן המימוש? והחשובה מכולן- מי יבטיח שמכוניות אלו מוגנות מפני תוקפים זדוניים?

הבעיה הזו מסקרנת אותי מאוד והיא אחת הסיבות המשמעותיות לכך שבחרתי בנושא זה עבור פרויקט הגמר שלי.

עבור הפרויקט שלי החלטתי לתקוף בעיה זאת באמצעות ניסוי עצמי- לדמות את סיטואציית הרכבים המקושרים ולחקור את האמצעים שתוקף יכול להשתמש בהן כדי לפגוע בהן, תוך כדי הגנה נגד התקיפות השונות שגיליתי.

באמצעות יצירת הבעיה בעצמי מהשלב הנמוך ביותר שניתן תוך כדי ניסיון לפתור את הבעיות שנמצאות קיוויתי להבין את הבעיה לעומק ולהבין במה כרוך יצירת עולם בו חפצי היום יום שלנו חכמים ומחוברים, ולענות על השאלה- האם מתי שהוא נהיה בטוחים באמת?

לצורך כך בניתי בעזרת:

1. RaspberryPi: מחשב זעיר בעל יכולת שליטה ברכיבים חשמליים- מריץ Linux ובעל יכולת תקשור דרך רשת האינטרנט.
2. מספר מנועים ורכיבים אלקטרוניים.
3. מתאם Wi-Fi שמאפשר לRaspberryPi לתקשר עם מחשב הלקוח באמצעות Wi-Fi..
4. שרת Python אשר מקבלת פקודות מהאינטרנט וגורם למכונית לבצע אותן.

מכונית קטנה אשר ניתנת לשליטה מרחוק דרך האינטרנט, כדי לבדוק כמה מכוניות אלו בטוחות ואיך ניתן להגן עליהן.

התחלתי כאשר על המכונית אין שום הגנה והצלחתי לשלוט בה בתור תוקף, וכך התקדמתי עם מעגל של מציאה וניצול של תקיפה – והגנה נגדה.

האופן שבו הגנתי נגד התקיפות שמצאתי הוא דרך מחשבה עצמאית- כלומר הגנתי נגד כל תקיפה באופן ספציפי תוך כדי הפעלת מחשבה ומציאת פתרון בעצמי, במקום להשתמש בפתרונות ידועים וכלליים שמקובלים בתעשייה.

כתוצאה מכך, לב הפרויקט הוא **פרוטוקול פקודות מאובטח** אשר מאפשר למכונית קבלת פקודות שליטה מרחוק רק ממי שאמור לשלוט בה- באמצעות פתרון שונה מהמקובל בתעשייה.

*ארכיטקטורה*

הפרויקט שלי הינו פרוטוקול פקודות מאובטח בין המכונית לבין מחשב של לקוח בעל אישור לשימוש במכונית, דרך Wi-Fi ורשת האינטרנט. מטרת הפרוטוקול הוא הגנה מפני תוקף אשר חדר לרשת הפנימית, ונמצא במצב של Man in the Middle- כלומר התוקף הצליח למקם עצמו בין המכונית לבין מחשב המכונית, כך שהוא רואה ושולט בכל המידע שעובר ביניהם דרך האינטרנט. הפרוטוקול מניח שעל מנת לשלוט במכונית יש צורך באמצעי הזדהות כלשהו, אחרת כל לקוח ברשת יכול לשלוט בה באופן חופשי.

**הסיכונים אשר מהווה תוקף במצב MitM:**

1. התוקף רואה את כל מה שנשלח בין 2 הצדדים ויכול להשתמש במידע עבור קבלת שליטה במכונית. לדוגמא- צפייה בסיסמא שנשלחת על מנת לאמת את זהות המשתמש.
2. התוקף שולט במידע שעובר ויכול לשנות את תוכנו. לדוגמא- שינוי הפקודות שהמשתמש המאומת שולח, לפקודות זדוניות משלו.
3. התוקף רואה את תוכן הפקודות שהמשתמש שולח, ויכול כך לעקוב אחרי המשתמש- חדירה לפרטיות המשתמש.

פרוטוקול הפקודות שפיתחתי שואף להגן נגד סיכונים אלו, ולצמצם ככל שניתן את הסיכון שמהווה תוקף במצב MitM.

הפרוטוקול שואף להגיע למצב שבו- המכונית תוכל לוודא שמשתמש מחזיק באמצעי הזדהות חוקי, מבלי לשלוח אותו באופן מפורש על גבי האינטרנט.

פתרון זה מומש ללא שימוש בפרוטוקולי הצפנה ממומשים מראש וללא שימוש בהצפנה קונבנציונלית.

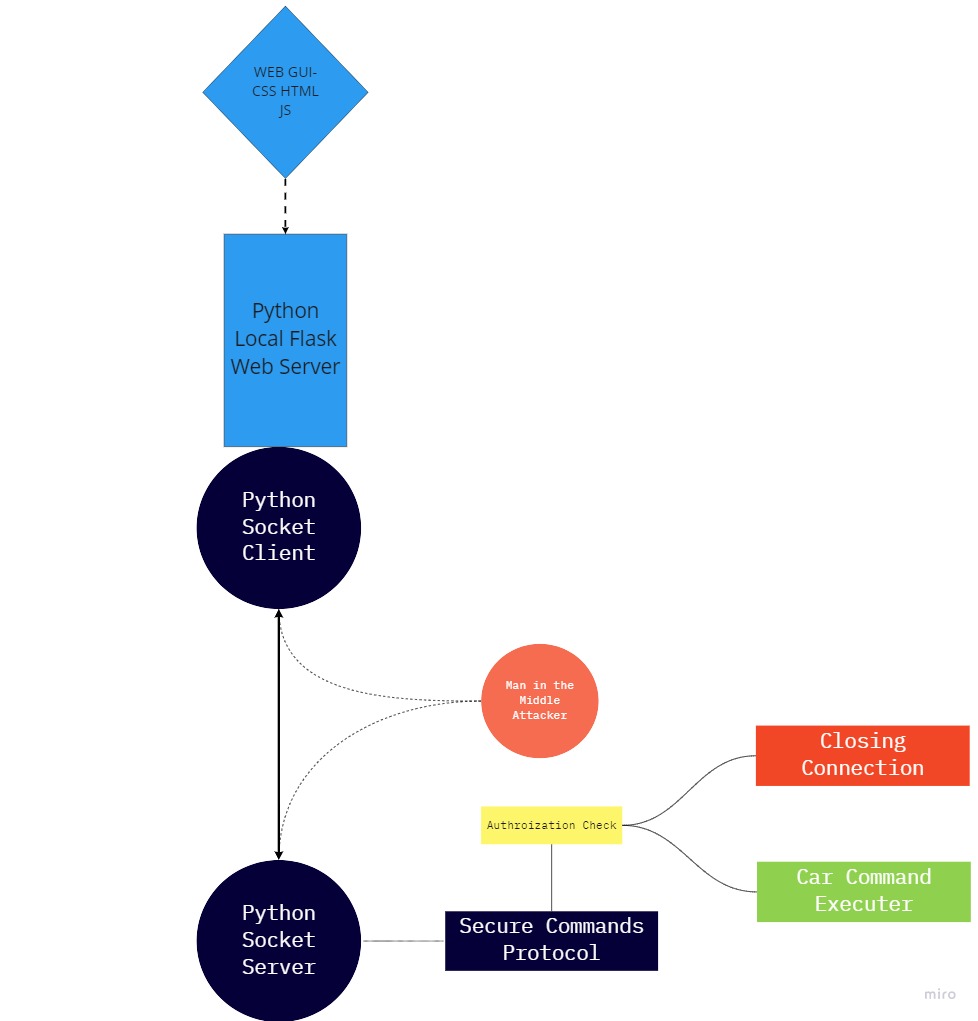
ממשק השליטה מרחוק נבנה כמספר חלקים,

**בצד השרת**:

1. שרת Python אשר מאזין לפקודות דרך האינטרנט, לפי הפרוטוקול ( אשר אודותיו אפרט בהמשך).
2. חתיכה לוגית אשר מוודאת את תוקפן של הפקודות ומאשרת אותן, לביצוע של המכונית במקרה חוקי ולניתוק של החיבור במקרה אי חוקי.
3. תוכנית אשר מחוברת אל מנועי המכונית ומבצעת את הפקודות שנשלחות אליה מחתיכה מספר 2 במידה וזאת מחליטה שהינן חוקיות.

**בצד הלקוח:**

1. אתר Web המשמש כממשק משתמש גרפי, כתוב בעזרת HTML, CSS, JS.
2. שרת Python ממומש באמצעות ספריית Flask, שרץ באופן מקומי ומשמש כגשר בין החיבור לשרת המכונית לבין ממשק המשתמש הכתוב בטכנולוגיות Web.
3. תוכנית Python אשר מתקשרת עם המכונית ומטפלת בחלק הלוגי של הלקוח, מקבלת הוראות משרת הFlask שמקבל הוראות מהמשתמש באתר.



החלקים שבצד הלקוח רצים באופן מקומי, והגישה היחידה אליהן עבור תוקף היא גישה ישירה אל המחשב. אותו המקרה תקף גם לחלקים אשר נמצאים בצד השרת(המכונית).

הגנה מפני מקרים כאלו- של תוקף בעל גישה ישירה אל המחשב, היא בור ללא תחתית ונושא המטופל באופן תמידי ובאופן שלא יגמר לעולם על ידי מומחי אבטחה, אנטי-וירוסים ומערכות ההפעלה עצמן. כלומר- Out of scope עבור פרויקט צנוע זה.

לכן, השכבה שבה בחרתי להתמקד עם ההגנות שלי היא שכבת הרשת, כלומר המידע שעובר בין נקודות הקצה השונות על גבי רשת האינטרנט. מידע זה הוא הרבה יותר נגיש וממד התקיפה עליו הוא גדול יותר ונגיש עבור תוקפים עם מעט מאוד עבודה. באופן ספציפי מתגוננים בפני תוקף בעל עמדת Man in the Middle- עמדה זו היא במספר רמות גבוהה יותר מאשר תוקף הנמצא ברשת, ומקנה לתוקף שליטה מוחלטת במידע שעובר בין 2 נקודות קצה. למרות זאת- זו עמדה נפוצה מאוד ותוקף מיומן יכול להגיע אליה בחלק גדול מהמקרים.

פרוטוקול הפקודות המאובטח שפיתחתי שואף כאמור להגן מפני תוקף בעמדה כזו, וכעת אפרט אודותיו.

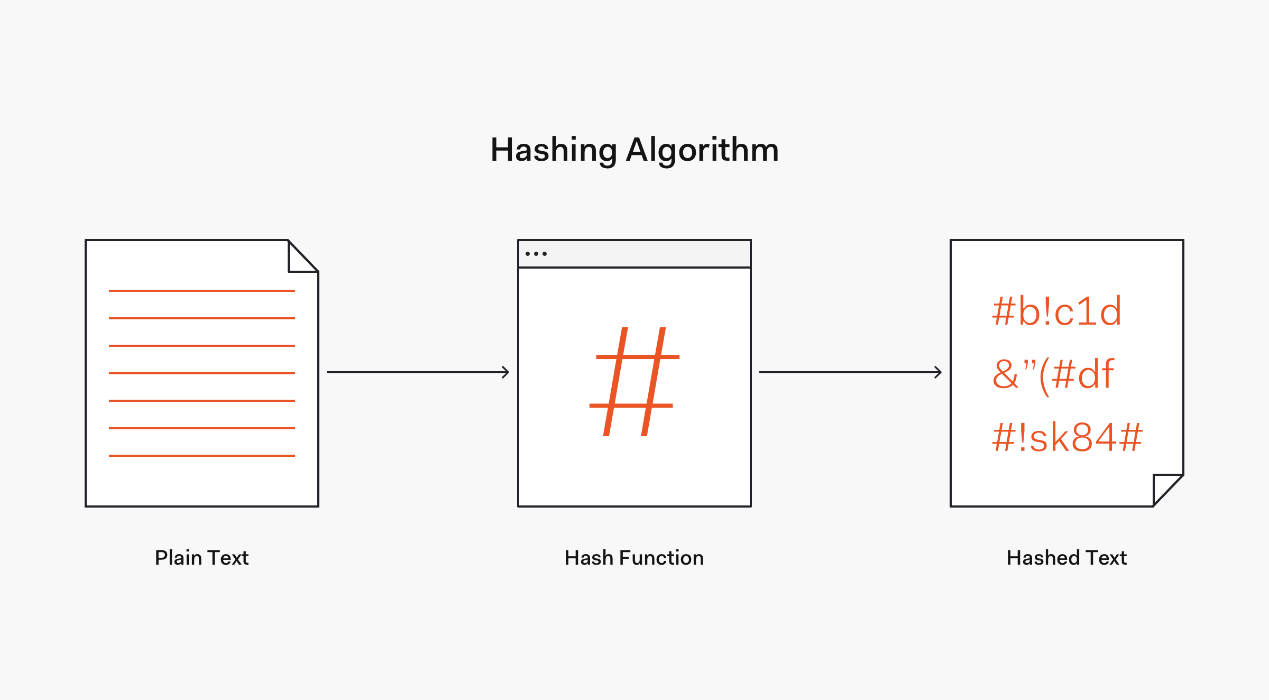
Secure Commands Protocol

חשוב לציין פעם נוספת- פרוטוקול זה פותח משיקולים לימודיים. הפרוטוקול איננו משתמש באמצעי הגנה מקובלים ועונה על הבעיות שעלו במהלך תהליך המחקר באמצעות מחשבה עצמאית. קיימים פרוטוקולי רשת (כגון SSL וTLS) אשר עונים על אותן הדרישות באמצעות הצפנה סימטרית ואי סימטרית, אך פרוטוקול זה איננו מבוסס עליהן או משתמש בהן.

**פונקציית Hash:**

אחת ממטרותיו של פרוטוקול הפקודות היא הימנעות מהעברת אמצעי הזדהות כגון סיסמא באופן מפורש דרך האינטרנט, על מנת שתוקף לא יוכל לקרוא את אלו ולהשתמש בהם בעצמו. לצורך כך בחרתי להשתמש בפונקציה קריפטוגרפית מסוג Hash.

מה היא פונקציית Hash? פונקציית Hash הינה פונקציה מתמטית חד כיוונית וחד ערכית. הפונקציה מקבלת כקלט ערך כלשהו באורך משתנה, ומחזירה כפלט ערך אחר בעל אורך קבוע. לקלט ולפלט אין שום מרכיבים משותפים ואי אפשר בשום דרך מתמטית )שידועה לנו(, לשחזר את ערך הקלט באמצעות ערך הפלט.



על מנת להסביר טוב יותר את אופייה של פונקציית הHash אשתמש בדוגמא פשוטה: שארית.

נסמן את פעולת השארית כ %. אם נחלק את 8 ב 7 ונבדוק מה היא השארית נקבל: . מה יקרה אם נחלק את המספר 15 ב7 ונבדוק מה היא השארית?

נקבל . ניתן לראות שתופעה זו ממשיכה עד אין-סוף: ,

וכן הלאה.

כאשר מקבלים את התוצאה 1 בלתי אפשרי לדעת מה היה הקלט המקורי ולשחזר את הפלט אליו.

ההבדל בין הדוגמא הפשוטה מאוד הזו לבין פונקציית Hash אמיתית היא חד הערכיות. בניגוד לחילוק ושארית שאותה התוצאה חוזרת עבור מספר גדול מאוד של ערכים (כל 7 בדוגמא זו), הפלט שמייצרת פונקציית הHash איננו חוזר על עצמו(ככל שניתן במסגרת המגבלות הפיזיות של כמות המספרים בפלט)- כלומר עבור קלט מסוים **ערך הפלט שהתקבל תואם אך ורק לאותו הקלט המקורי.**

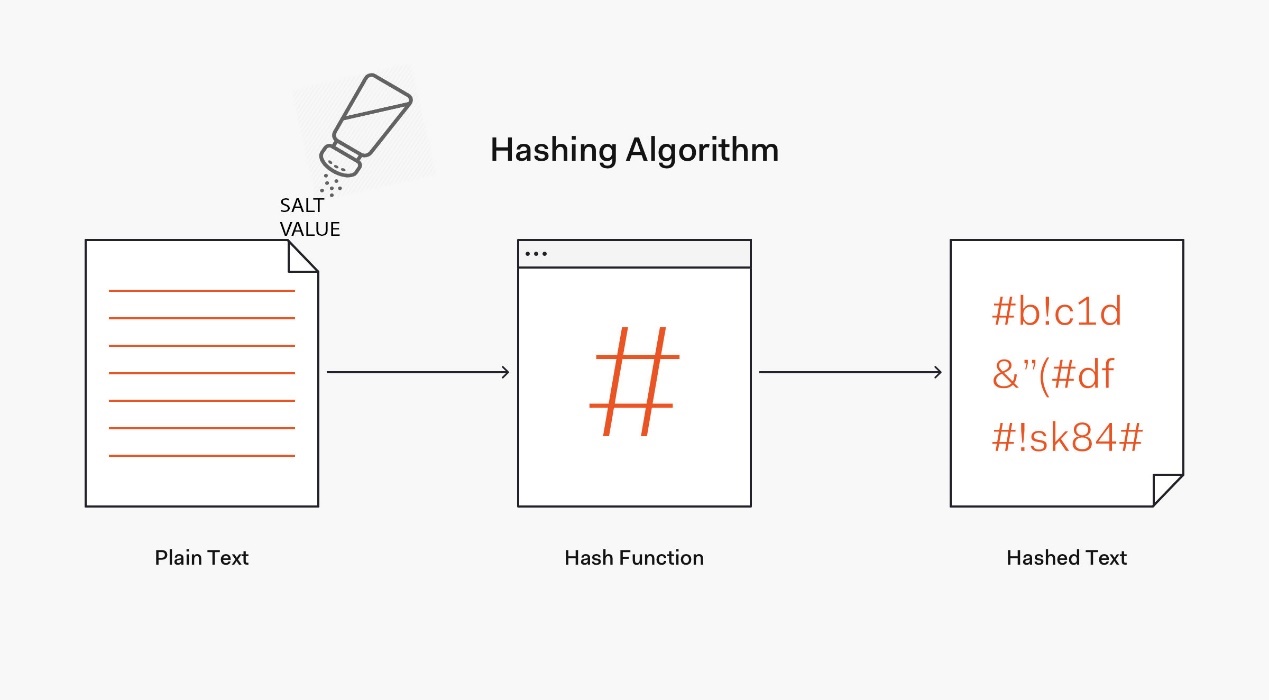
בעקבות תכונות חזקות מאוד אלו של פונקציות מסוג זה, בחרתי להשתמש בהן כבסיס עבור האבטחה בפרוטוקול הפקודות.

הרעיון בשימוש בפונקציית Hash היא שאפשר באמצעותה להוכיח ידיעה של ערך מסוים מבלי לשלוח אותו או לשמור אותו באופן מפורש. לדוגמא, ניתן לשלוח את הפלט כתוצאה משימוש ב Hash על הסיסמא אל השרת, השרת יבצע את אותו החישוב על הסיסמא שהוא מכיר וישווה בין 2 הפלטים. פלט זהה – שולח ההודעה מכיר את הסיסמא.

הבעיה במבנה כזה היא שברגע שנשלח פלט הHash של הסיסמא על גבי האינטרנט פעם אחת, תוקף יוכל לשמור אותו ולהשתמש בו בעצמו על מנת לאמת את זהותו כמשתמש חוקי.

**על מנת לפתור בעיה זו נשתמש בSalt:**

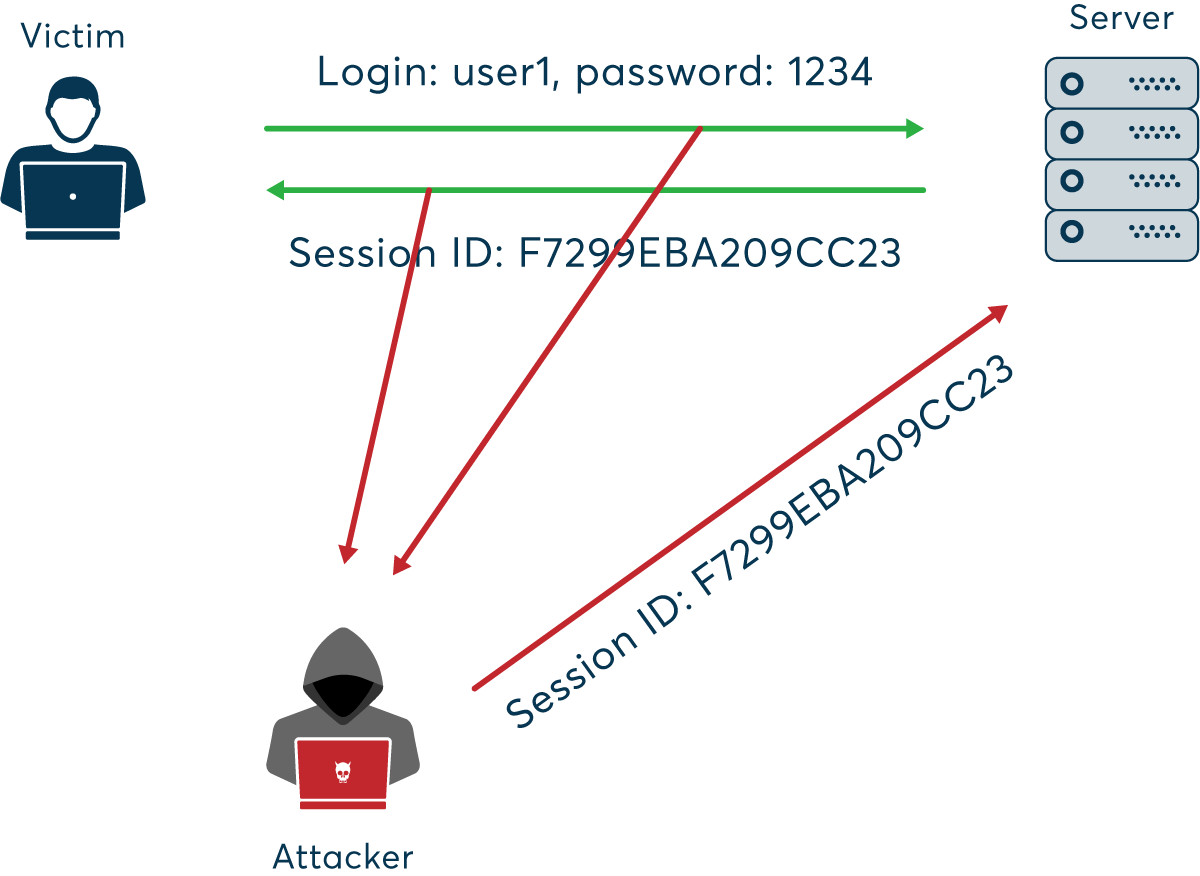
Salt הוא ערך כל שהוא אשר משמש כדי "לתבל" את קלט פונקציית ה Hash. את הSALT מוסיפים לקלט לפני שמחשבים את הHASH, פעולה זו גורמת לכך שפלט הHash איננו קבוע, ומשתנה מכל שליחה לשליחה. במקרה כזה השרת גם כן יחבר את הסיסמא שהוא מכיר עם ערך הSALT ויחשב את הHash של ערך זה בהתאם, וישווה כמו מקודם.

כעת- תוקף איננו יכול לשמור בצד את פלט הHash אשר נשלח על גבי האינטרנט ולהשתמש בו לאחר מכן בעצמו, מכיוון שאותו פלט תקף בדיוק לפעם אחת, ולאחר מכן מוחלף ערך הSALT ואיתו משתנה הפלט של פונקציית הHash הדרוש כדי לאשר את זהותו של המשתמש.

המצב הקיים הוא כזה- לקוח מסוגל להוכיח למכונית שברשותו הסיסמא- מבלי לחשוף אותה לתקוף שמאזין על הרשת. מעולה! מה חסר? לא נגענו במה קורה לאחר והשרת מכיר בלקוח כלקוח מאושר. האם הפקודות נשלחות כמו שהן? זה פוגע בפרטיות, מה מאפיין לקוח? כתובת הIP? TCP Session? קיימות מספר תקיפות על מאפיינים אלו שמאפשרות לתוקף להעביר מידע בשמו של משתמש אחר לדוגמא IP Spoofing ו TCP Hijacking.

ביצעתי בפועל את התקיפות הללו על המכונית והיא בחלט פגיעה אליהן!

דוגמא אחת היא מתקפה על פרוטוקול TCP. הרעיון במתקפה זו הוא לחכות שמשתמש לגיטימי יזדהה מול המכונית ויגרום לה לסמוך עליו, ואז לרכוב על סשן ה TCP שלו מולה ולשלוח פקודות זדוניות בשמו. שוב- הפקודות נשלחת כמו שהן ולכן תוקף שמכיר את אופן ההרכבה שלהן יכול לשלוח פקודות בעצמו לאחר תקיפה שהושלמה בהצלחה. חיבור TCP מוגדר באמצעות מספר רנדומלי שבוחר השרת והוא הSequence Number. כל פקטת TCP ממוספרת באמצעות מספר התחלתי זה (הוא גדל בערך קבוע עם כל פקטה), ולפי המספר פקטות שונות משויכות לחיבור מסוים. לכן, תוקף אשר יכול לצפות בפקטות הנשלחות בין המכונית אל הלקוח. בעזרת המידע הזה תוקף יכול ליצור פקטה משלו המכילה את כתובת הIP של הלקוח בשדה המוען ואת מספר הSEQ (פלוס הערך הקבוע) בתור המספור, וכך תגיע הפקטה אל השרת כפקטה רגילה שנשלחה בידי הלקוח המאומת. במקרה זה, המכונית תתייחס אל הפקודה כפקודה חוקית שנשלחה על ידי לקוח מאומת **ותבצע אותה**.



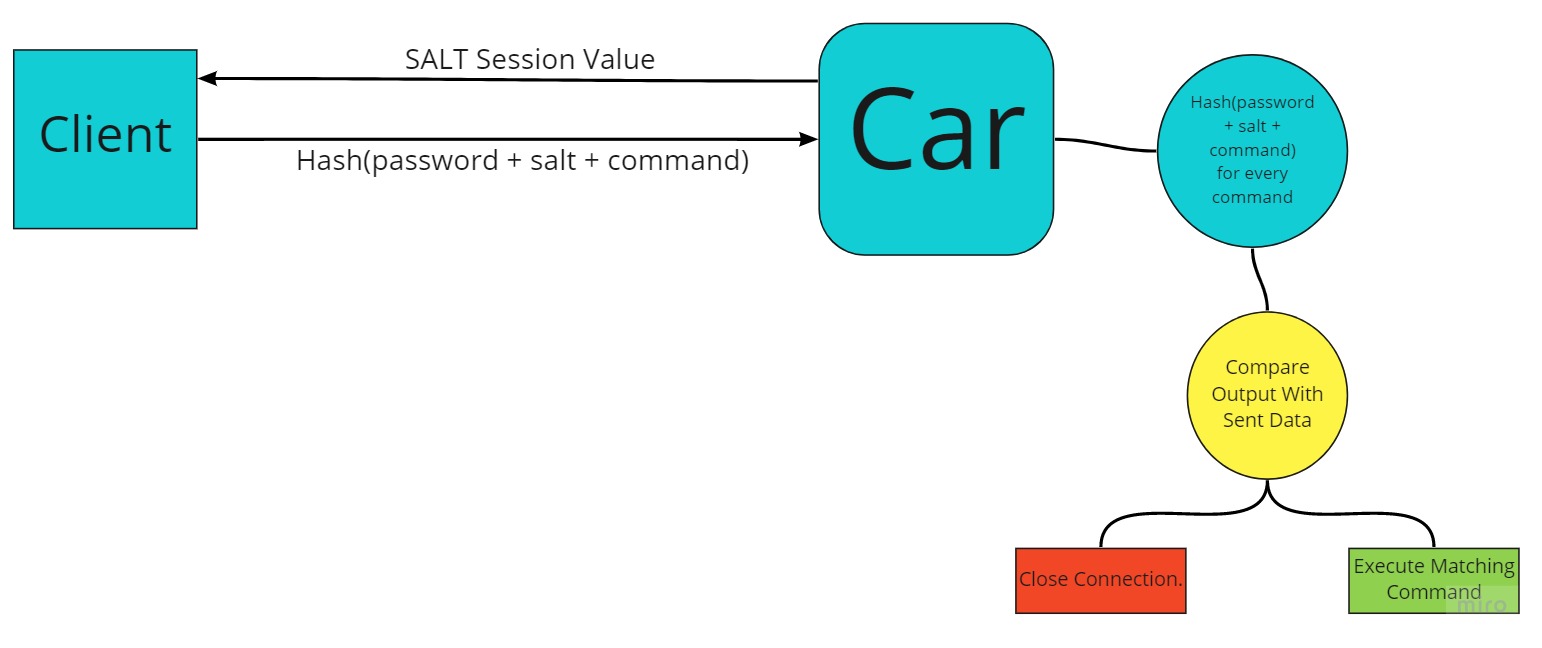
איך נתגונן? הפתרון עליו החלטתי הוא גישה אשר איננה סומכת על שום לקוח. המערכת מניחה הנחה בסיסית- הרשת נפרצה ויש לקחת כל פקטה וכל פקודה בערבון מוגבל. ברוח גישה זו הפתרון המוצע הוא פתרון שלפיו יש להוכיח בכל שליחת פקודה שהינך משתמש חוקי אשר מכיר את הסיסמא – אמצעי ההזדהות.

לצורך כך, במקום לשלוח את Hash(password + salt) בתחילת החיבור כדי לאמת את זהותך, ולאחר מכן לשלוח פקודות באופן גלוי- יש לשלוח עבור כל פקודה ופקודה את פלט פונקציית הHash עבור הקלט: הסיסמא, ערך הSALT והפקודה הרצויה.

*המכונית שתקבל את פלט הפונקציה המתוארת מעלה, תחשב גם היא את אותו החישוב עבור כל אחת מהפקודות שהיא מכירה ויודעת לבצע. המכונית תשווה בין הפקטה שנשלחה אליה לבין פלט חישוביה עבור כל אחת מהפקודות בחיפוש אחר פלט זהה. אם פלט החישוב של המכונית עבור פקודה מסוימת זהה למידע הנשלח כלומר- המשתמש ששלח את הפקודה מכיר את הסיסמא ופקודה זו היא הפקודה שביקש לבצע. אם אף אחד מהפלטים אינו תואם-*

1. *המשתמש אינו מכיר את הסיסמא*
2. *המשתמש השתמש בSALT לא נכון*
3. *המשתמש ביקש פקודה לא חוקית שאיננה מוכרת על ידי המכונית*

*במצב כזה החיבור בין המכונית לבין הלקוח הספציפי ינותק והמכונית תאזין לחיבור חדש.*



*כאמור ערך הHash הינו חד ערכי וחד כיווני ולכן:*

1. *תוקף איננו יכול לשחזר את הסיסמא באמצעות הפלט הנשלח*
2. *תוקף איננו יכול לשנות את המידע שנשלח או לרכוב על החיבור כדי לבצע פקודות משלו, מבלי לדעת מה הסיסמא*
3. *תוקף איננו יכול לשמור את פלט ה* Hash *שמתקבל לשימוש חוזר מאוחר יותר- כי ערך הSALT דואג שהם אינם יחזרו על עצמם*
4. *התוקף אינו יכול לעקוב אחרי המשתמש החוקי ולצפות בפקודות שזה מבקש מהמכונית לבצע*

*השגנו את כל המטרות שלנו! כעת, הפרוטוקול שלנו מגן ברמה שרצינו על התקשורת שבין המכונית למשתמש מפני תוקפים במצב Man in the Middle.*

אז זהו, שלא. קיימת בעיה נוספת שלא התגוננו בפניה והיא בעיית ה- Brute Force.

**Brute Force:**

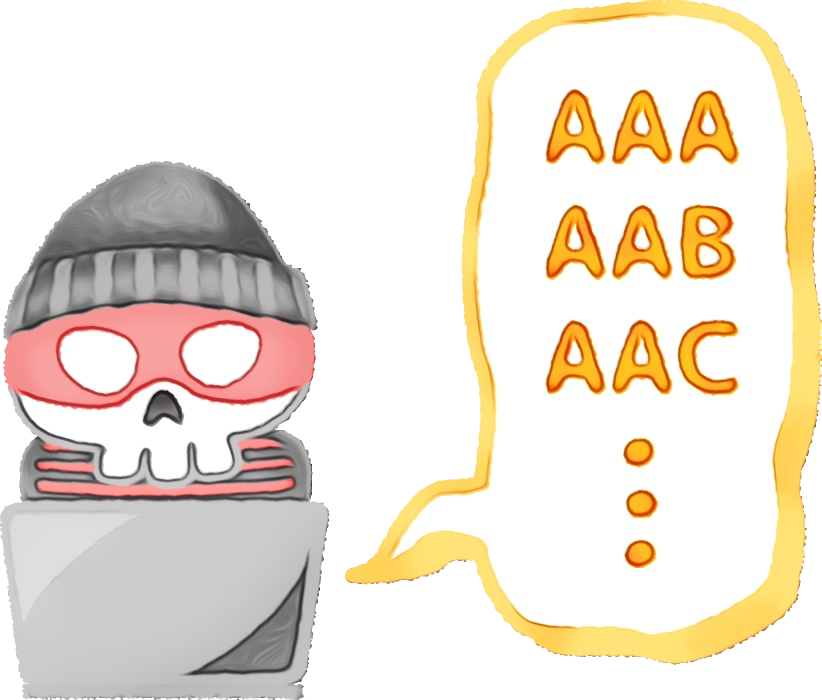
כאמור, המידע שאנו שולחים על גבי הרשת נגיש לגמרי לתוקף הוא הפלט של:

נניח שהתוקף מכיר את 2 המרכיבים salt ו command. המשתנה היחיד שאינו מכיר הוא הסיסמא. תוקף יכול לנסות ולחשב את אותה הפונקציה, בעזרת 2 המרכיבים שברשותו- עבור כל סיסמא אפשרית בעולם. בשלב כל-שהוא יגיע התוקף אל הסיסמא הנכונה ופלט הHash שיקבל עבור הקלט של הסיסמא ו2 המרכיבים יהיה זהה לזה שנשלח על ידי המשתמש הלגיטימי ובכך **לשחזר את הסיסמא**.

פרק הזמן שייקח לתוקף לנסות כל סיסמא אפשרית הוא ארוך- ארוך במידה אין סופית. עבור סיסמא בעלת 8 תווים עם 4 אותיות 2 מספרים ו2 סימנים מיוחדים: מספר האפשרויות לסיסמא הוא **457,163,239,653,376**. זה אמנם המון ניסיונות ופרק הזמן שייקח לשחזר את הסיסמא הוא לא פרקטי כתקיפה, אך רוב המקרים הנפוצים שונים מכך.

בני אדם הם יצורים צפויים ומונעים מהרגל. קיימים ברחבי האינטרנט מילוני ענק שמכילים מיליוני סיסמאות נפוצות. תוכנות חכמות יודעות היום לנסות את כל האופציות האלו בשילוב עם פרטים אישיים שונים שמסופק עליכם- רוב הסיכויים שהסיסמא שלכם אינה חזקה כמו שאתם חושבים.

בעיה! לא דיברנו על מהיכן מגיע ערך הSALT והאם תוקף יודע אותו. אם כן, תוקף בסבירות יחסית גבוהה יוכל לשחזר את סיסמת המשתמש ולהשתמש בה על מנת לקבל גישה אל המכונית. לכן, **ברור שערך הSALT אינו יכול להישלח באינטרנט בשום צורה או דרך.**



**Locally Generated SALT Number**

אז ראינו שערך הSALT איננו יכול להישלח על גבי האינטרנט. המסקנה הבלתי נמנעת היא כזו- 2 צדדי הקצה (המשתמש והמכונית) צריכים בכל רגע נתון לדעת מה הוא ערך הSALT אשר בשימוש כרגע, מבלי להעביר אותו באופן מפורש על גבי האינטרנט.

בשביל להשיג מטרה זו ניסיתי מספר כיוונים. הפתרון כאן איננו ברור מראש וקיים מספר אינסופי של פתרונות אפשריים, בחרתי באחד האידיאלי ביותר בעייני.

המובן מאליו הוא זמן- 2 הצדדים יודעים מהו הזמן הנוכחי ויוכלו להשתמש בו (או בשינוי שלו בהשפעת איזו שהיא פונקציה מתמטית) כדי לקבוע את ערך ה SALT. בפתרון הזמן יש מספר בעיות:

1. קשה מאוד לסנכרן זמן מדויק בין 2 מחשבים וזה לא אמין
2. אם נבחר פרק זמן לא קטן מספיק (בעקבות בעיות סנכרון לדוגמא) יכול להיווצר מצב של שרשור פקודות, בו תוקף יוכל לשלוח שוב את המידע שנשלח על ידי המשתמש לפני שמתחלף ערך ה SALT וכך לגרום למכונית לבצע פקודות משלו.

הפתרון שהגעתי אליו כדי לייצר מצב זה- בו 2 הצדדים יודעים בכל רגע מה הוא הSALT מבלי שתוקף יוכל לדעת זו הוא באמצעות **ניצול האופן בו המחשב מחשב מספר רנדומלי.**

רנדומליות אמיתית איננה קיימת במחשב- כל דבר אשר קיים במחשב הוא תוצאה ישירה של חישובים מתמטיים ופעולות בסיסיות, המחשב פיזית איננו מסוגל לבחור מספר באופן רנדומלי אמיתי. במקום זאת הוא משתמש בפונקציות שנקראות **Pseudo Random Generator** או בקיצור **PRG**.

פונקציות אלו מקבלות ערך התחלתי שנקרא Seed ומבצעות עליו חישובים מתמטיים סטטיסטיים. הרעיון הוא שאם ערך הSEED משתנה מספיק מהר בעקבות החישובים הסטטיסטיים יתקבל רצף מספרים כביכול רנדומליים. ערך ה Seed הוא המפתח לפתרון זה. באופן ברירת המחדל הSEED נקבע באמצעות השעון של המחשב- מספר שמשתנה בקצב גבוה מאוד ולכן מתאים מאוד ליצירת מספרים רנדומליים. למרות זאת, קיימת האופציה לקבוע את ערך הSEED באופן ידני. אני נחשפתי לאופציה זו במסגרת תרגיל אשר דרש אתחול רנדומלי של ערכים. כותבי התרגיל קבעו ערך SEED שרירותי וטענו שכך יהיה ניתן להשוות בין תוצאות התרגיל ולוודא שהצלחנו בו.

במסגרת חיפוש הפתרון נזכרתי בדבר זה וזה גרם לי לחשוב- מה זה אומר שאפשר לוודא את תוצאות התרגיל? זה אומר שתוצאות ההגרלה הרנדומלית יהיו זהות בכל המחשבים שיריצו את התרגיל בתנאי שערך ה SEED יהיה זהה?

כן! חקרתי לעומק את הנושא והגעתי למסקנה כזו- הגרלת המספרים מבוססת על הSEED וחוץ ממנו הינה מתמטית לחלוטין. לכן, עבור אותו ערך SEED יבוצעו אותם חישובים מתמטיים על אותם המספרים וכתוצאה מכך **בדיוק אותו רצף של מספרים.**

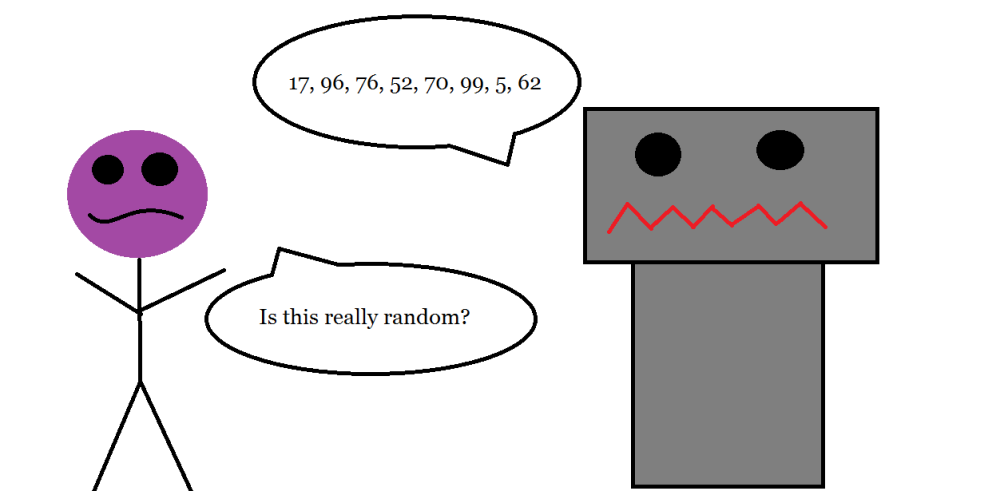
זה אומר שאם גם המכונית וגם הלקוח יאתחלו את ערך ה SEED לאותו הערך, עבור כל הגרלת מספר "רנדומלי" שניהם יקבלו את אותה התוצאה! לשם כך, הוספתי "סוד" נוסף. ערך נוסף שמשמש להזדהות בנוסף לסיסמא, אשר אליו לתוקף אין גישה.

שני הצדדים מאתחלים את ערך הSEED עם אותו סוד. המכונית מגרילה מספר רנדומלי בכל פעם שהיא מקבלת חיבור חדש מלקוח- מספר זה הינו ה SALT עבור אותו החיבור. המכונית שומרת את מספר הפעמים שהיא הגרילה מספר- מספר החיבורים שקיבלה. כאשר לקוח חדש מתחבר אל המכונית, המכונית שולחת אליו בחזרה את מספר הפעמים שהגרילה מספר.

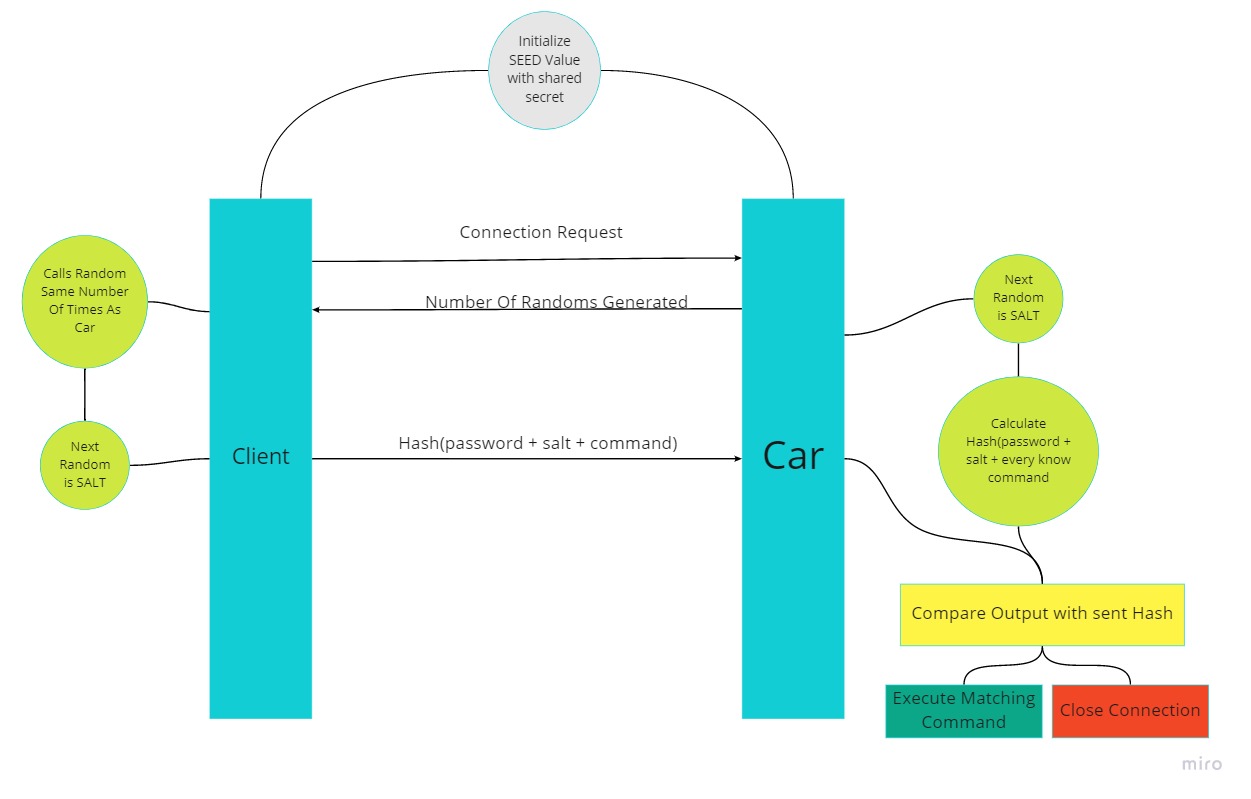
הלקוח שמקבל את המספר הזה, מגריל גם הוא מספרים כמספר הפעמים שהגרילה המכונית- על מנת להשוות אליה. לאחר שגם הלקוח וגם המכונית הגרילו את אותו מספר של פעמים מספרים רנדומליים, מבוססים על אותו SEED ראשוני, המספר הבא שיגרילו יהיה בדיוק אותו הדבר! במספר זה שניהם משתמשים בתור ה SALT של החיבור, ואותו רק משנים בערך קבוע עם כל פקודה נשלחת.

לתוקף יש גישה אל מספר הפעמים שהגרילה המכונית מספר, אך ללא ערך הSEED הראשוני איננו יכול לשחזר את אותו המספר ולכן הוא אינו יודע מה הוא הSALT , בעוד שגם המשתמש וגם המכונית יודעים זאת שניהם.

**כך השגתי חישוב מקומי של ערך הSALT.**



פיו. זה היה מעט ארוך. כעת אחרי שדיברנו לעומק על כל החלקים בפרוטוקול אפשר לחבר הכל ביחד ☺.



נעבור שוב על מבנה הפרוטוקול ככולל- אשר מתואר בתרשים למעלה.

1. 2 הצדדים מאתחלים את ערך הSEED עם המספר הסודי ששניהם יודעים
2. לקוח יוזם חיבור אל המכונית
3. המכונית שולחת בחזרה את כמות הפעמים שהגרילה מספר רנדומלי
4. הלקוח מגריל גם הוא מספרים רנדומליים כמספר הפעמים שנשלח מהמכונית, אל מנת להשוות אליה.
5. 2 הצדדים מגרילים מספר- ומקבלים מספר זהה. מספר זה משמש כערך הSALT ההתחלתי.
6. מעכשיו הלקוח יכול לשלוח פקודות אשר מכילות את פלט הפונקציה HASH עבור Hash(password + salt + command)
7. המכונית מחשבת את אותו הדבר עבור כל הפקודות שברשותה, ומשווה בין חישוביה לבין המידע שהתקבל
8. במידה וקיבלה התאמה, הפקודה עוברת לביצוע המכונית. במידה ולא החיבור מנותק
9. 2 הצדדים מוסיפים 1 לערך הSALT על מנת לדאוג שפלט הHASH איננו חוזר על עצמו ואיננו שמיש יותר מפעם אחת.

תמונה שמכילה מזון, שלט, ציור

התיאור נוצר באופן אוטומטי