301 – App Sleuth

Team Information

Team Name : LuckyVicky

Team Member : Eungchang Lee, Hyun Yi, Juho Heo, Dongkyu Lee

Email Address : dfc_luckyvicky@googlegroups.com

Instructions

Description A spy from the enemy has been captured, and a mobile device has been obtained. Although the lock screen was successfully bypassed using the spy's biometric authentication, the contents of a specific app used by the spy couldn't be accessed. This app appears to be used for communicating with the spy's headquarters and receiving instructions for secret missions. Analyze the app to decrypt the encrypted data and obtain information about the secret missions.

Target	Hash (MD5)
spy_s_mobile.ad1	0CDA2CDBF69E54EF9B03F1B04B25F6EA

Questions

1. What is the content of the first mission? (100 points)
2. What is the content of the second mission? (100 points)
3. Write an analysis of the app used by the spy and impersonate the spy to obtain information about additional secret missions. (100 points)
   A. The analysis of the app must be included.
   B. Be cautious, as reckless communication may expose your identity and result in being blocked by the enemy.

Teams must:

- Develop and document the step-by-step approach used to solve this problem to allow another examiner to replicate team actions and results.
- Specify all tools used in deriving the conclusion(s).

Tools used:

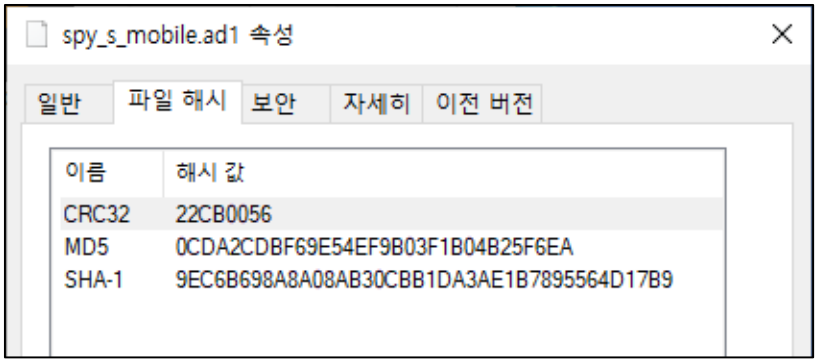
Name:	JEB PRO	Publisher:	PNF Software, Inc
Version:	5.5.0.202311022109		
URL:	https://www.pnfsoftware.com		

Name:	HashTab	Publisher:	Implbits Software
Version:	6.0.0		
URL:	https://implbits.com		

Name:	Sublime Text 4	Publisher:	Sublime Text
Version:	Build 4180		
URL:	https://www.sublimetext.com		

Name:	FTK Imager	Publisher:	Exterro
Version:	4.7.0.31		
URL:	https://www.exterro.com		

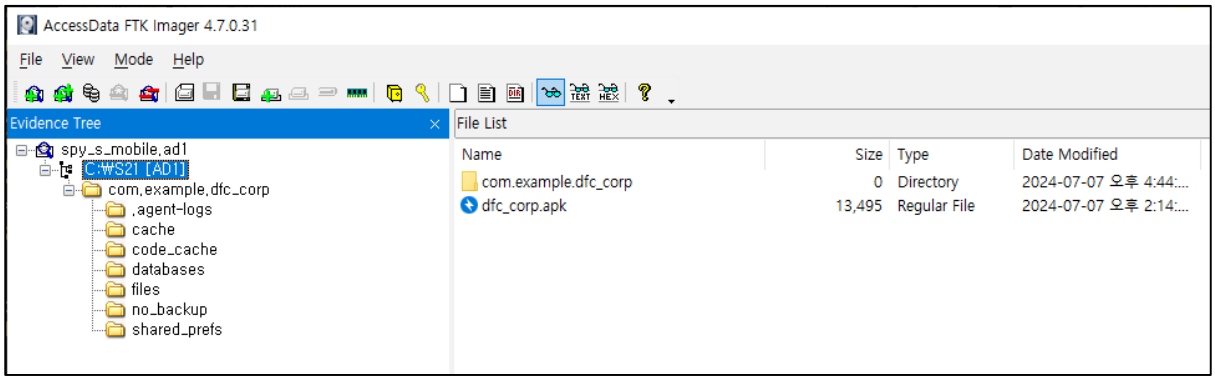
Step-by-step methodology:



[그림 1] md5 해시 값 확인

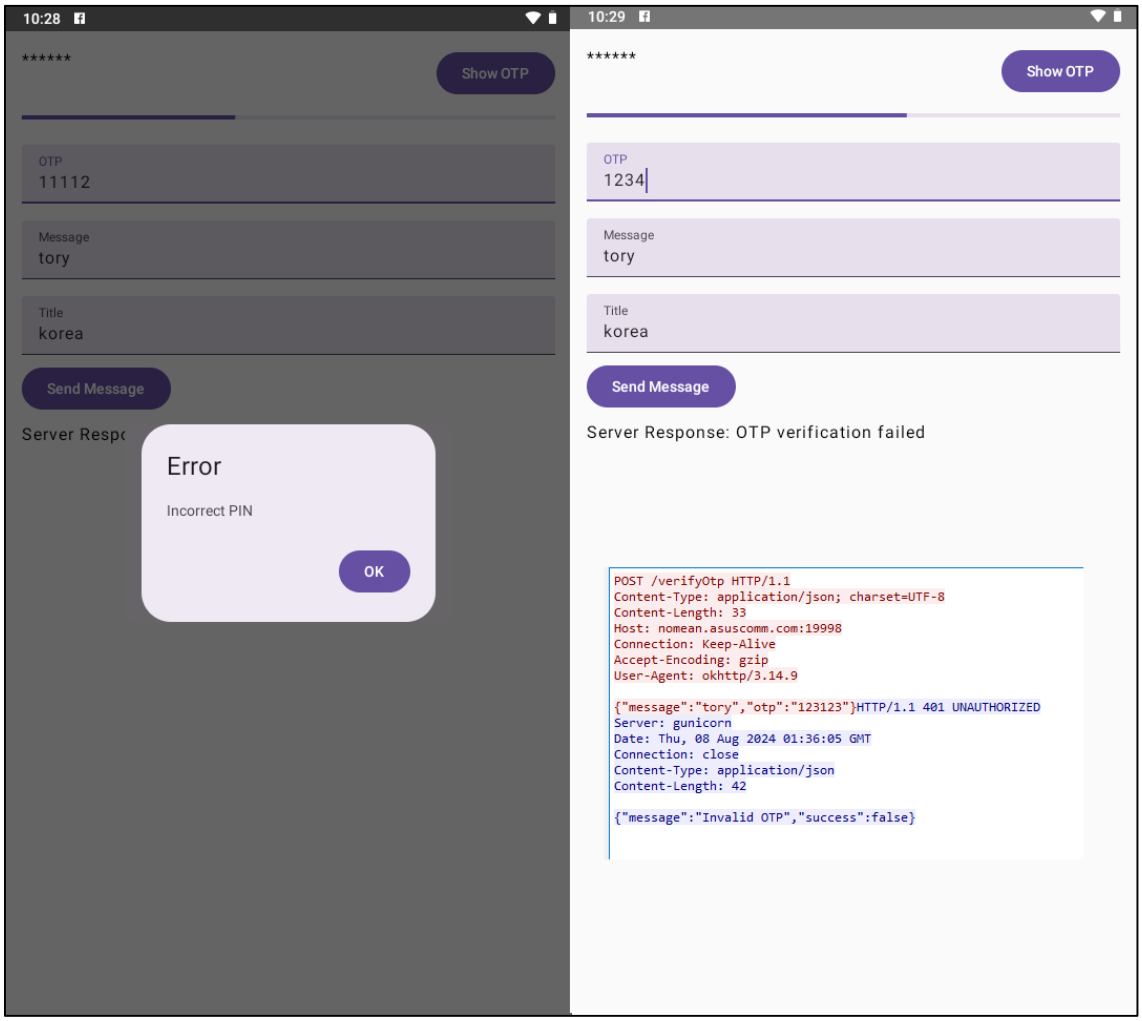
주어진 파일들의 MD5 해시 값이 일치하는 것을 확인하였습니다.

1. What is the content of the first mission? (100 points)



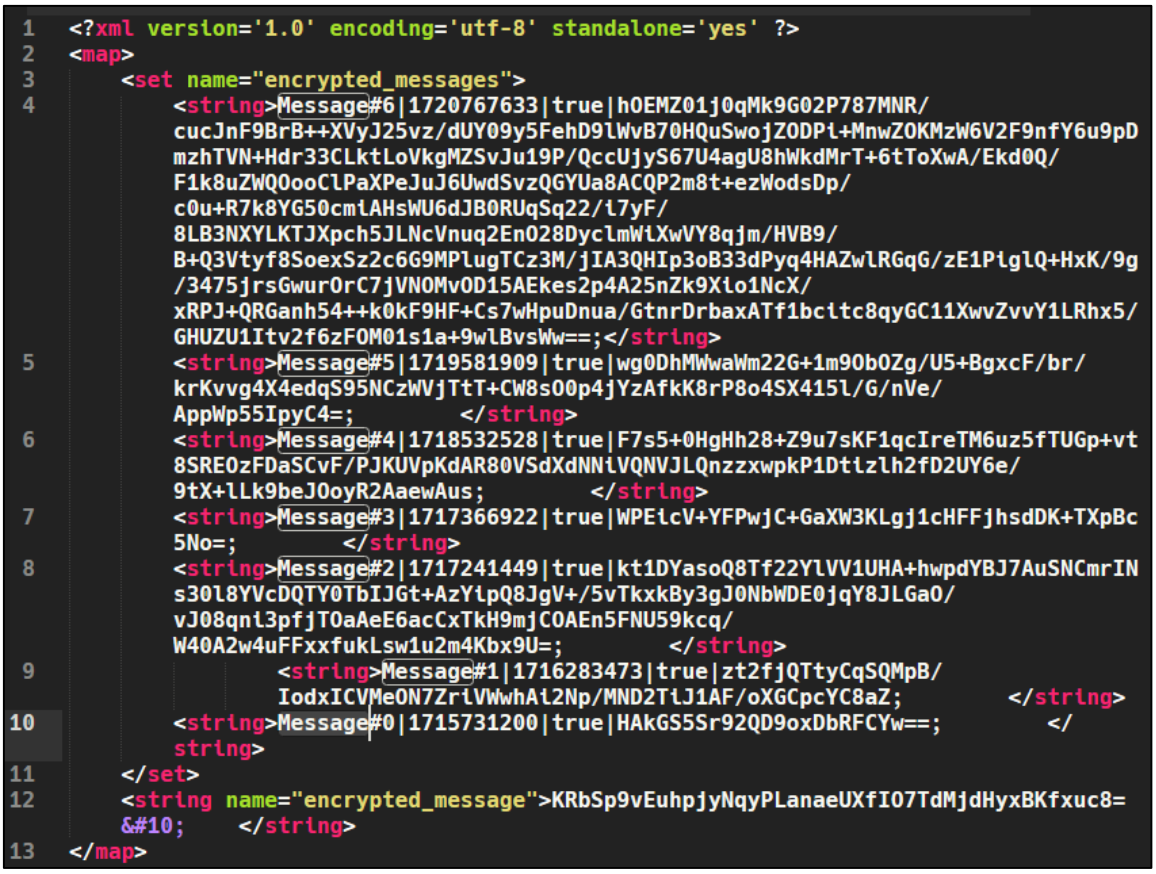
[그림 2] ad1 확인

제공된 AD1 파일을 FTK Imager로 분석한 결과 APK 파일, shared_prefs, files 등 특정 애플리케이션과 관련된 모든 데이터가 포함되어 있었습니다. 데이터를 추출한 후 애플리케이션 실행 시 아래와 같은 화면을 확인할 수 있었습니다.



[그림 3] 애플리케이션 실행 화면

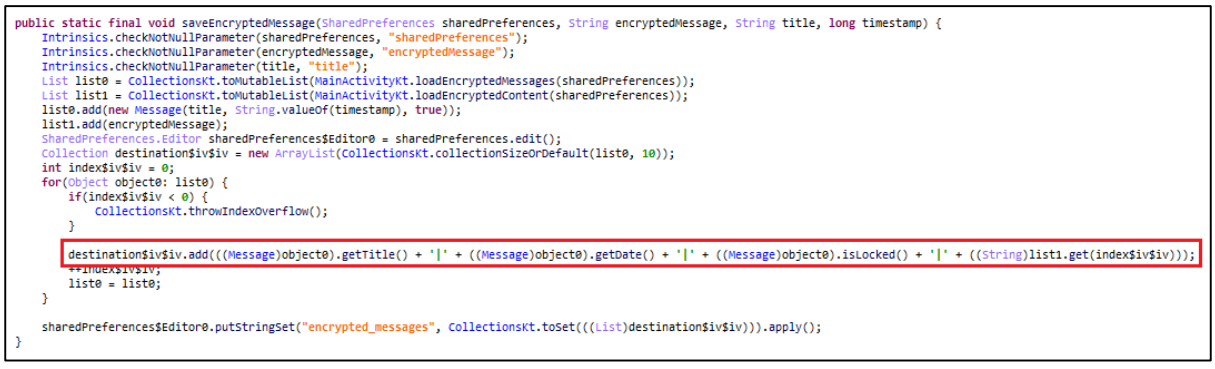
해당 애플리케이션은 OTP(일회용 비밀번호) 코드를 서버에 전송하여 검증한 후, 성공적으로 인증되면 메시지를 암호화하여 sharedPreferences.xml에 저장하는 기능을 제공합니다. OTP는 10초마다 갱신되며 사용자가 ShowOTP 버튼을 통해 PIN을 올바르게 입력하면 화면 좌측 상단의 별표(*)로 가려진 OTP 코드가 정상적으로 표시됩니다.



```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <set name="encrypted_messages">
        <string>Message#6|1720767633|true|hOEMZ01j0qMk9G02P787MNR/
        cucJnF9BrB++XVyJ25vz/dUY09y5FehD9lWvB70HQuSwojZODPi+MnwZOKMzW6V2F9nfY6u9pD
        mzhTVN+Hdr33CLktLoVkgMZSvJu19P/QccUjyS67U4agU8hWkdMrT+6tToXwA/Ekd0Q/
        F1k8uZWQOooClPaXPeJuJ6UwdSvzQGYUa8ACQP2m8t+ezWodsDp/
        c0u+R7k8YG50cmiAHsWU6dJB0RUqSq22/i7yF/
        8LB3NXYLKTJXpch5JLNcVnuq2EnO28DyclmWlXwVY8qjm/HVB9/
        B+Q3Vtyf8SoexSz2c6G9MPlugTCz3M/jIA3QHIp3oB33dPyq4HAZwlRGqG/zE1PiglQ+HxK/9g
        /3475jrsGwurOrC7jVNOMvOD15AEkes2p4A25nZk9Xio1NcX/
        xRPJ+QRGanh54++k0kF9HF+Cs7wHpuDnua/GtnrDrbaxATf1bcltc8qyGC11XwvZvvY1LRhx5/
        GHUZU1Itv2f6zFOM01s1a+9wlBvsWw==;</string>
        <string>Message#5|1719581909|true|wg0DhMWwaWm22G+1m9ObOZg/U5+BgxcF/br/
        krKvvg4X4edqS95NCzWVjTtT+CW8sO0p4jYzAfkK8rP8o4SX415l/G/nVe/
        AppWp55IpyC4=;        </string>
        <string>Message#4|1718532528|true|F7s5+0HgHh28+Z9u7sKF1qcIreTM6uz5fTUGp+vt
        8SREOzFDaSCvF/PJKUVpKdAR80VSdXdNNiVQNVJLQnzzxwpkP1Dtizlh2fD2UY6e/
        9tX+lLk9beJOoyR2AaewAus;        </string>
        <string>Message#3|1717366922|true|WPEicV+YFPwjC+GaXW3KLgj1cHFFjhsdDK+TXpBc
        5No=;        </string>
        <string>Message#2|1717241449|true|kt1DYasoQ8Tf22YlVV1UHA+hwpdYBJ7AuSNCmrIN
        s30l8YVcDQTY0TbIJGt+AzYipQ8JgV+/5vTkxkBy3gJ0NbWDE0jqY8JLGaO/
        vJ08qni3pfjTOaAeE6acCxTkH9mjCOAEn5FNU59kcq/
        W40A2w4uFFxxfukLsw1u2m4Kbx9U=;        </string>
        <string>Message#1|1716283473|true|zt2fjQTtyCqSQMpB/
        IodxICVMeON7ZrlVWwhAi2Np/MND2TiJ1AF/oXGCpcYC8aZ;        </string>
        <string>Message#0|1715731200|true|HAkGS5Sr92QD9oxDbRFCYw==;        </
        string>
    </set>
    <string name="encrypted_message">KRbSp9vEuhpjyNqyPLanaeUXfIO7TdMjdHyxBKfxuc8=
    &#10;    </string>
</map>
```

[그림 4] AD1에서 추출한 sharedPreferences.xml

AD1 파일을 FTK Imager로 추가 분석하여 스파이가 기존에 받은 암호를 발견할 수 있었습니다.



```
public static final void saveEncryptedMessage(SharedPreferences sharedPreferences, String encryptedMessage, String title, long timestamp) {
    Intrinsics.checkNotNullParameter(sharedPreferences, "sharedPreferences");
    Intrinsics.checkNotNullParameter(encryptedMessage, "encryptedMessage");
    Intrinsics.checkNotNullParameter(title, "title");
    List list0 = CollectionsKt.toMutableList(MainActivityKt.loadEncryptedMessages(sharedPreferences));
    List list1 = CollectionsKt.toMutableList(MainActivityKt.loadEncryptedContent(sharedPreferences));
    list0.add(new Message(title, String.valueOf(timestamp), true));
    list1.add(encryptedMessage);
    SharedPreferences.Editor sharedPreferences$Editor0 = sharedPreferences.edit();
    Collection destination$iv$iv = new ArrayList(CollectionsKt.collectionSizeOrDefault(list0, 10));
    int index$iv$iv = 0;
    for(Object object0: list0) {
        if(index$iv$iv < 0) {
            CollectionsKt.throwIndexOverflow();
        }

        destination$iv$iv.add(((Message)object0).getTitle() + '|' + ((Message)object0).getDate() + '|' + ((Message)object0).isLocked() + '|' + ((String)list1.get(index$iv$iv)));
        ++index$iv$iv;
        list0 = list0;
    }

    sharedPreferences$Editor0.putStringSet("encrypted_messages", CollectionsKt.toSet(((List)destination$iv$iv))).apply();
}
```

[그림 5] saveEncryptedMessage 함수

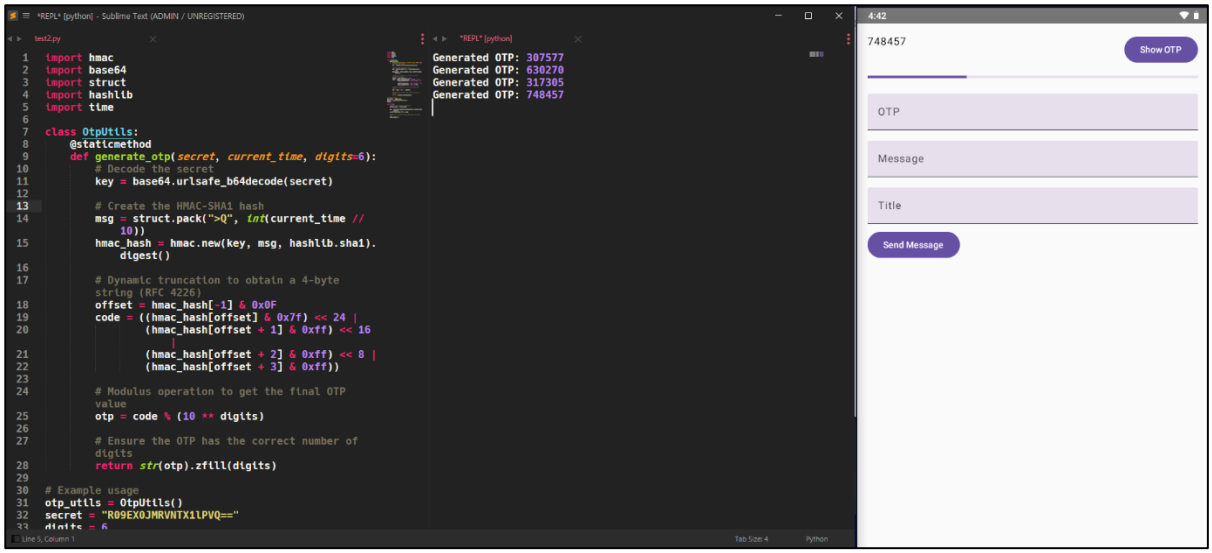
암호화 된 메시지는 총 7개이며 애플리케이션 분석을 통해 제목, 시간, 잠금 여부, 메시지 순서대로 저장되었음을 확인하였습니다.

```
public final void invoke() {
    String s = MainActivityKt.MainScreen$lambda$40(this.$selectedEncryptedMessage$delegate);
    if(s != null) {
        MutableState mutableState0 = this.$serverResponse$delegate;
        try {
            String s1 = MainActivityKt.MainScreen$lambda$55(this.$decryptOtpInput$delegate);
            byte[] arr_b = AesUtils.INSTANCE.sha256(s1);
            String s2 = AesUtils.INSTANCE.decrypt(s, arr_b);
            MainActivityKt.MainScreen$lambda$20(this.$decryptedMessage$delegate, s2);
            MainActivityKt.MainScreen$lambda$47(this.$showDecryptedMessageDialog$delegate, true);
            MainActivityKt.MainScreen$lambda$35(this.$showOtpDialog$delegate, false);
        }
        catch(Exception e) {
            MainActivityKt.MainScreen$lambda$17(mutableState0, "Invalid OTP or decryption error");
        }

        return;
    }
```

[그림 6] 복호화 관련 함수

복호화 과정은 AES 알고리즘을 통해 이루어지며 메시지가 저장될 당시의 OTP 코드가 복호화 키로 사용됩니다.



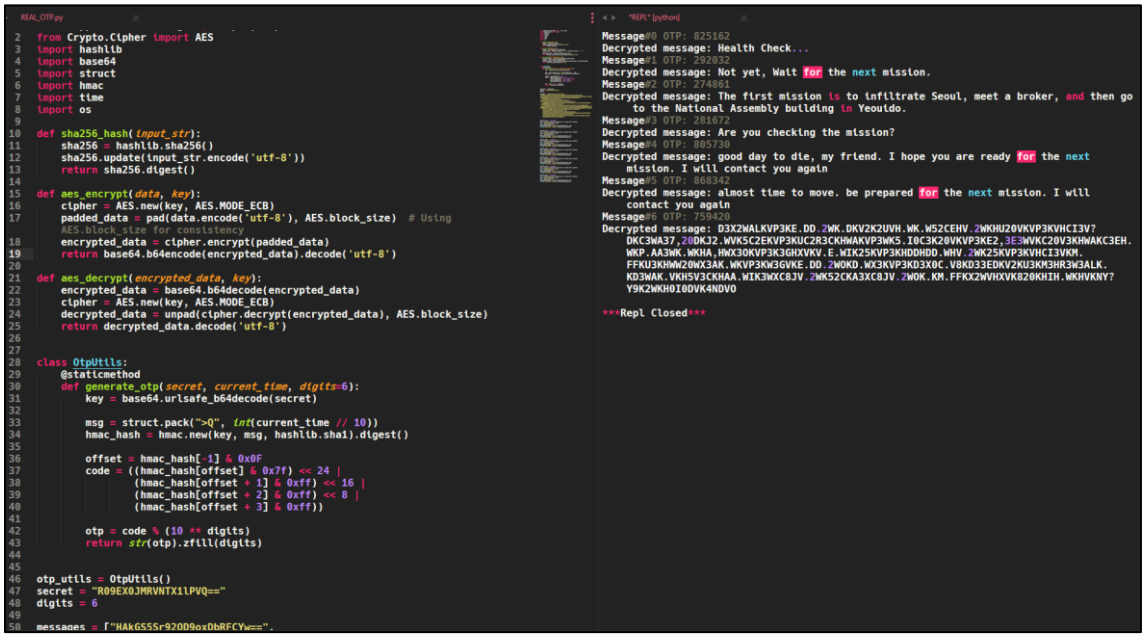
[그림 7] OTP 생성 알고리즘 검증

소스 코드를 통해 생성된 OTP 코드가 예상한 대로 동작하는지 확인하기 위해 ShowOTP 기능을 Frida 후킹으로 우회하여 OTP 검증을 성공적으로 하였습니다.

Frida 후킹 소스코드

```
Java.perform(function () {
    var HashUtils = Java.use('com.example.dfc_corp.utils.HashUtils');
    HashUtils.hash.implementation = function (input) {
        console.log("Hooked HashUtils.hash with input: " + input);
        var fixedHash = "3dcef420185a26c829576282199cbe449bce4d3cb8347575829d639be1426c20";
        console.log("Returning fixed hash: " + fixedHash);
        return fixedHash;
    };
    var MainActivityKt = Java.use('com.example.dfc_corp.MainActivityKt');
    MainActivityKt.MainScreen$lambda$69.implementation = function (mutableState, newCount) {
        console.log("Tapped! Tap count: " + newCount);
        return this.MainScreen$lambda$69(mutableState, newCount);
    };
});
```

[표 1] 후킹 소스코드



[그림 8] OTP 생성 알고리즘 검증

sharedPreferences.xml에 저장된 암호문에는 Unix 타임스탬프가 포함되어 있어, 이를 기반으로 OTP를 생성할 수 있습니다. 생성된 OTP 코드는 SHA-256 해시 과정을 거쳐 AES 복호화 키로 사용됩니다. 이를 통해 소스 코드 작업을 통해 7개의 메시지를 성공적으로 복호화하였으며 첫 번째 미션의 내용을 확인할 수 있었습니다.

답: The first mission is to infiltrate Seoul, meet a broker, and then go to the National Assembly building in Yeouido.

2. What is the content of the second mission? (100 points)

두 번째 미션
D3X2WALKVP3KE.DD.2WK.DKV2K2UVH.WK.W52CEHV.2WKHU20VKVP3KVH CI3V?DKC3WA37,20DKJ2.WVK5C2EKVP3KUC2R3CKHWAKVP3WK5.I0C3K20V KVP3KE2,3E3WVKC20V3KHWAKC3EH.WKP.AA3WK.WKHA,HWX3OKVP3K3G HXVKV.E.WIK25KVP3KHDDHDD.WHV.2WK25KVP3KVHCI3VKM.FFKU3KHWW 20WX3AK.WKVP3KW3GVKE.DD.2WOKD.WX3KVP3KD3X0C.V8KD33EDKV2KU 3KM3HR3W3ALK.KD3WAK.VKH5V3CKHAA.WIK3WXC8JV.2WK52CKA3XC8JV. 2WOK.KM.FFKX2WVHXVK820KHIH.WKHVKNY?Y9K2WKH0I0DVK4NDVO

[표 2] 미션 암호문

두 번째 미션의 경우, 알 수 없는 방법으로 암호화되어 있습니다. 해당 암호문을 풀기 위해 다양한 가정들을 세워 시도하였고 알파벳 빈도수에 따라 단순 치환되는 구조라는 것을 파악하였습니다.

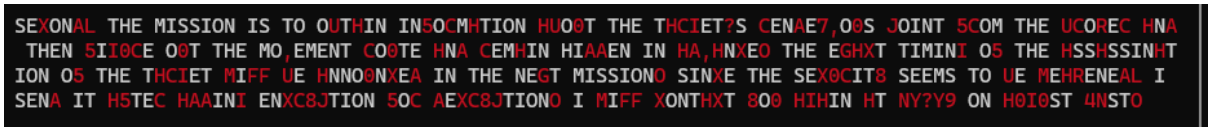
암호문에서 제일 많이 등장하는 알파벳이 'K'임을 확인하였고 이를 보편적인 영어 문장에서 가장 많이 사용되는 알파벳 'E'로 치환하는 방식으로 접근하였습니다. 분석을 진행한 결과, 'K'가 띄어쓰기를 나타낸다는 결론에 도달하였습니다. 이와 같은 방식으로 치환 테이블을 점진적으로 구축하며 암호문을 해독해 나갔습니다.

```
D3X2WAL VP3 E.DD.2W .D V2 2UVH.W .W52CEHV.2W HU20V VP3 VHCI3V?D C3WA37,20D J2.WV 5C2E VP3 UC2R3C HWA
 VP3W 5.I0C3 20V VP3 E2,3E3WV C20V3 HWA C3EH.W P.AA3W .W HA,HWX3O VP3 3GHXV V.E.WI 25 VP3 HDDHDD.WHV
.2W 25 VP3 VHCI3V M.FF U3 HWW20WX3A .W VP3 W3GV E.DD.2WO D.WX3 VP3 D3X0C.V8 D33ED V2 U3 M3HR3W3AL .
D3WA .V H5V3C HAA.WI 3WXC8JV.2W 52C A3XC8JV.2WO . M.FF X2WVHXV 820 HIH.W HV NY?Y9 2W H0I0DV 4NDVO
```

[그림 9] 치환 결과

K = "띄어쓰기"로 가정하여 암호문을 치환했을 시, [그림 9]와 같이 출력됩니다. 여기서 VP3이라는 단어가 자주 등장하는데 총 10번 정도 사용되며 이는 The라고 치환을 할 수 있었습니다.

또한, 첫 번째 미션에서 확인할 수 있듯이 "The first mission is to…"가 등장함에 따라 E.DD.2W는 mission으로 치환할 수 있습니다.



[그림 10] 치환 결과

총 9개의 문자를 1:1로 치환하였고 [그림 10]과 같은 문장을 확인할 수 있습니다.

암호 치환 소스 코드

```
from colorama import Fore, Style, init

init(autoreset=True)

def replace_pattern(input_string, replacements):
    result = []
    for char in input_string:
        if char in replacements:
            result.append(replacements[char])
        else:
            result.append(Fore.RED + char + Style.RESET_ALL)
    return ''.join(result)

input_string =
"D3X2WALKVP3KE.DD.2WK.DKV2K2UVH.WK.W52CEHV.2WKHU20VKVP3KVHCI3V?DKC3WA37,20DKJ2.WVK5C2EKVP3KUC2R3CKH
WAKVP3WK5.I0C3K20VKVP3KE2,3E3WVKC20V3KHWAKC3EH.WKP.AA3WK.WKHA,HWX3OKVP3K3GHXVKV.E.WIK25KVP3KHDDHDD.
WHV.2WK25KVP3KVHCI3VKM.FFKU3KHWW20WX3AK.WKVP3KW3GVKE.DD.2WOKD.WX3KVP3KD3X0C.V8KD33EDKV2KU3KM3HR3
W3ALK.KD3WAK.VKH5V3CKHAA.WIK3WXC8JV.2WK52CKA3XC8JV.2WOK.KM.FFKX2WVHXVK820KHIH.WKHVKNY?Y9K2WKH0I0DVK
4NDVO"

# 치환 규칙
replacements = {
    # 가장 많이 보이는 알파벳 K = E로 가정하였으나, 띄어쓰기일 가능성이 있으며 이에 따라 VP3은 THE로 해석됩니다.
    'K': ' ',
    'V': 'T',
    'P': 'H',
    '3': 'E',

    # "E.DD.2W"이 반복되는 것으로 보아 "MISSION"일 가능성이 높습니다.
    'E': 'M',
    '.': 'I',
    'D': 'S',
    '2': 'O',
    'W': 'N',

    # 또한, 2번째 미션임으로 SECOND를 앞부분에서 찾아 치환합니다.
    # 기존 메시지5번에서 I will contact you again 문자열이 있으므로 비슷한 패턴을 추가합니다.
    'X': 'C',
    'A': 'D',
    'H': 'A',
    'M': 'W',
```

```
    'F': 'L',

    # IN5OCMATION -> INFORMATION
    # AUO0T -> ABOUT
    # AIAIN -> AGAIN
    '5': 'F',
    'C': 'R',
    'U': 'B',
    '0': 'U',
    'I': 'G',

    # O는 마침표일 가능성이 있습니다.
    # SECURIT8 -> SECURITY
    # ENCRYJTION -> ENCRYPTION
    'O': '.',
    '8': 'Y',
    'J': 'P',

    # BRORER -> BROKER
    # MO,EMENT -> MOVEMENT
    # NEGT -> NEXT
    # SECONDL, WEAKENEDL -> 부적절한 L은 ,(콤마)로 대체
    # RENDE7VOUS -> RENDEZVOUS
    'R': 'K',
    ',': 'V',
    'G': 'X',
    'L': ',',
    '7': 'Z',

    # TARGET?S -> TARGET'S
    # NY?Y9
    # 4N -> 21, 4N -> 31
    'N': '1',
    '?': "'",
}
output_string = replace_pattern(input_string, replacements)
print(output_string)
```

[표 3] 치환 소스코드

```
SECOND, THE MISSION IS TO OBTAIN INFORMATION ABOUT THE TARGET'S RENDEZVOUS POINT FROM THE BROKER AND
 THEN FIGURE OUT THE MOVEMENT ROUTE AND REMAIN HIDDEN IN ADVANCE. THE EXACT TIMING OF THE ASSASSINAT
ION OF THE TARGET WILL BE ANNOUNCED IN THE NEXT MISSION. SINCE THE SECURITY SEEMS TO BE WEAKENED, I
SEND IT AFTER ADDING ENCRYPTION FOR DECRYPTION. I WILL CONTACT YOU AGAIN AT 1Y'Y9 ON AUGUST 41ST.
```

[그림 11] 치환 결과

치환 테이블을 구축해 나아갔지만 최종 시간적인 부분은 파악할 수 없었고 세 번째 미션을 파악하는 과정에서 정확한 시간을 알 수 있게 되었습니다.

답: SECOND, THE MISSION IS TO OBTAIN INFORMATION ABOUT THE TARGET'S RENDEZVOUS POINT FROM THE BROKER AND THEN FIGURE OUT THE MOVEMENT ROUTE AND REMAIN HIDDEN IN ADVANCE. THE EXACT TIMING OF THE ASSASSINATION OF THE TARGET WILL BE ANNOUNCED IN THE NEXT MISSION. SINCE THE SECURITY SEEMS TO BE WEAKENED, I SEND IT AFTER ADDING ENCRYPTION FOR DECRYPTION. I WILL CONTACT YOU AGAIN AT 13'30 ON AUGUST 21ST.

3. Write an analysis of the app used by the spy and impersonate the spy to obtain information about additional secret missions. (100 points)

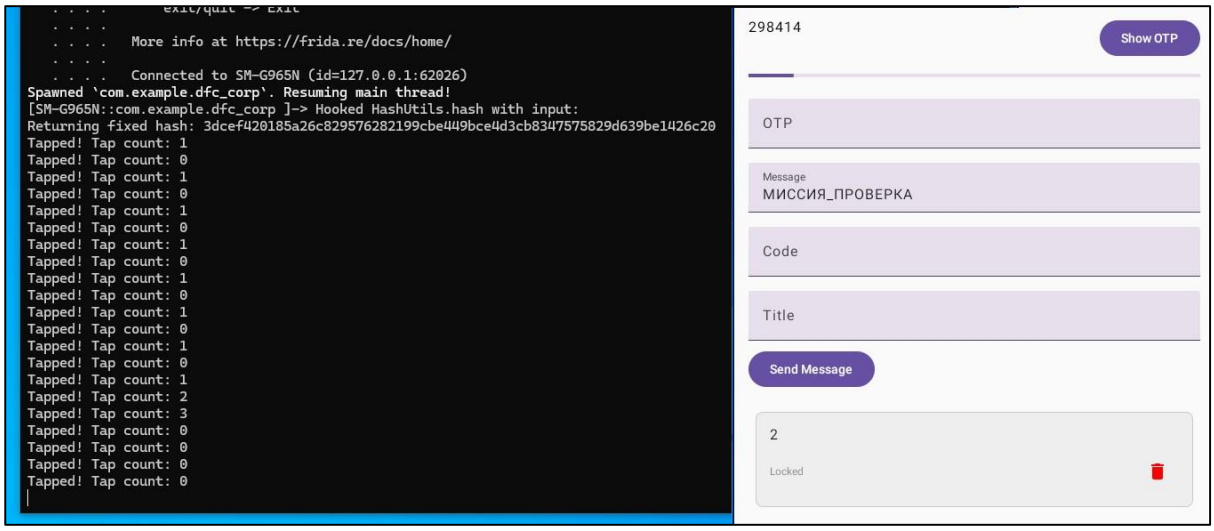
```
public final void invoke-k-4lQ0M(long it) {
    int v1 = MainActivityKt.MainScreen$lambda$68(this.$tapCount$delegate);
    MainActivityKt.MainScreen$lambda$69(this.$tapCount$delegate, v1 + 1);
    if(MainActivityKt.MainScreen$lambda$68(this.$tapCount$delegate) == 3) {
        if(Intrinsics.areEqual(MainActivityKt.MainScreen$lambda$28(this.$messageToSend$delegate), "МИССИЯ_ПРОВЕРКА")) {
            MainActivityKt.MainScreen$lambda$29(this.$messageToSend$delegate, "");
            MainActivityKt.MainScreen$lambda$50(this.$showCodeInput$delegate, false);
        }
        else {
            MainActivityKt.MainScreen$lambda$29(this.$messageToSend$delegate, "МИССИЯ_ПРОВЕРКА");
            MainActivityKt.MainScreen$lambda$50(this.$showCodeInput$delegate, true);
        }

        MainActivityKt.MainScreen$lambda$69(this.$tapCount$delegate, 0);
    }

    this.$resetTapCount.invoke();
}
```

[그림 13] 숨겨진 기능

추가 분석을 통해 애플리케이션 화면에서 연속으로 3번 터치했을 경우 새로운 입력 칸이 나타난다는 것을 확인할 수 있었습니다.



[그림 14] code 입력 칸 활성화

code 입력 칸을 활성화 했을 경우, OTP와 마찬가지로 6글자로 고정되어 더 이상의 입력이 불가능합니다. 또한, 메시지는 МИССИЯ_ПРОВЕРКА(미션_체크)로 변경되고 해당 문구를 포함해야만 code를 검증하고 있습니다.

```
{'code': '595931', 'message': 'МИССИЯ_ПРОВЕРКА', 'otp': '990636'}
Request was successful
Response: {'message': '[Mission Code Incorret] There is no matching mission code.
    Please check the mission code.', 'success': True}
```

[그림 15] 통신 과정

메시지(미션_체크)는 고정된 값이며 OTP는 현재 시간에 생성된 코드를 사용합니다. 마지막으로 code는 두 번째 미션 내용을 바탕으로 주어진 시간을 OTP 알고리즘을 사용해 변환한 후 입력해야 미션을 완료할 수 있습니다.

[그림 16] 연락 시간

두 번째 미션을 해독하는 과정에서 Y, 9, 4는 정확히 치환하지 못해 시간을 찾는 데 어려움이 있었습니다. 그러나 1:1 매칭을 통해 치환이 이루어지므로 이미 치환된 값은 중복되지 않습니다. 이를 바탕으로 가능한 범위를 최대한 축소하여 code 입력 시도 횟수를 최소화할 수 있습니다.

문자	범위
Y	0, 2, 3, 4, 5
9	0, 2, 3, 4, 5, 6, 7, 8, 9
4	2, 3

[표 4] 치환 범위

4의 경우 21st 31st 임으로 2개로 특정됩니다. 또한, 21st 일 경우 Y, 9는 2를 사용할 수 없습니다. 이러한 방법을 통해 시도 횟수를 최소화하여 코드를 입력하면 추가 미션을 받을 수 있습니다. 마지막으로 OTP는 10초마다 갱신되므로 13:30:00부터 13:30:50까지의 범위가 포함되어야 합니다. 이 범위에서의 경우의 수는 총 336개이며 21일과 31일 각각 168번의 입력 시도가 필요합니다.

추가 미션 확인 소스코드

```
import time
import hmac
import json
import base64
import random
import struct
import hashlib
import requests
import itertools
from datetime import datetime

class OtpUtils:
    @staticmethod
    def generate_otp(secret, current_time, digits=6):
        key = base64.urlsafe_b64decode(secret)

        msg = struct.pack(">Q", int(current_time // 10))
        hmac_hash = hmac.new(key, msg, hashlib.sha1).digest()
```

```
        offset = hmac_hash[-1] & 0x0F
        code = ((hmac_hash[offset] & 0x7f) << 24 |
                (hmac_hash[offset + 1] & 0xff) << 16 |
                (hmac_hash[offset + 2] & 0xff) << 8 |
                (hmac_hash[offset + 3] & 0xff))

        otp = code % (10 ** digits)

        return str(otp).zfill(digits)


def send_otp_request(url, message, code, otp):
    headers = {
        "Content-Type": "application/json; charset=UTF-8",
        "Host": "nomean.asuscomm.com:19998",
        "Connection": "Keep-Alive",
        "Accept-Encoding": "gzip",
        "User-Agent": "okhttp/3.14.9"
    }

    payload = {
        "code": code,
        "message": message,
        "otp": otp
    }

    response = requests.post(url, headers=headers, data=json.dumps(payload))

    print(payload)

    if response.status_code == 200:
        print("Request was successful")
        print("Response:", response.json())
    else:
        print("Failed to send request")
        print("Status code:", response.status_code)
        print("Response:", response.text)



otp_utils = OtpUtils()
secret = "R09EX0JMRVNTX1lPVQ=="

url = "http://nomean.asuscomm.com:19998/verifyOtp"
message = "₩u041c₩u0418₩u0421₩u0421₩u0418₩u042f_₩u041f₩u0420₩u041e₩u0412₩u0415₩u0420₩u041a₩u0410"

possible_1 = [2, 3]
possible_2 = [0, 2, 3, 4, 5]
possible_3 = [0, 2, 3, 4, 5, 6, 7, 8, 9]
possible_4 = [0, 1, 2, 3, 4, 5]

combinations = list(itertools.product(possible_1, possible_2, possible_3, possible_4))

valid_combinations = [comb for comb in combinations if len(set(comb[:3])) == 3]
processing = 0
```

```
for comb in valid_combinations:
    print(f"[{processing}/{len(valid_combinations)}]...")
    processing += 1

    formatted_date = f"2024.08.{comb[0]}1 - 1{comb[1]}:{comb[1]}{comb[2]}:{comb[3]}0"
    print(f"Formatted date: {formatted_date}")

    dt_object = datetime.strptime(formatted_date, "%Y.%m.%d - %H:%M:%S")

    unix_time = int(dt_object.timestamp())

    code = otp_utils.generate_otp(secret, unix_time)
    print(f"code: {code}")

    current_time = time.time()
    otp = otp_utils.generate_otp(secret, current_time)
    print(f"otp: {otp}₩n")

    send_otp_request(url, message, code, otp)
    randnum = random.randint(3,7)
    time.sleep(randnum)
```

[표 4] 추가 미션 확인 소스코드

```
[44/336]...
Formatted date: 2024.08.21 - 13:30:20
code: 410905
otp: 597095

{'code': '410905', 'message': 'МИССИЯ_ПРОВЕРКА', 'otp': '597095'}
Request was successful
Response: {'message': '[Mission Code Corret] This is the final mission. Alice,
    a DFC corporate leader, is the target at an event on October 1, 2024. Make
    it look like a car accident.', 'success': True}
```

[그림 17] 추가 미션 확인

답: [Mission Code Corret] This is the final mission. Alice, a DFC corporate leader, is the target at an event on October 1, 2024. Make it look like a car accident.