

암호화 기반 안티-포렌식 보안 메신저 분석 방법론

로컬 크리덴셜을 이용한 복호화 기법

소 속	서울경찰청, 고려대학교
발표자	손 지 훈
일 시	2023-07-30 (일)
설 명	BoB 특강

발표 목차

- 보안 메신저 소개
- 로컬 크리덴셜을 이용한 복호화 기법
- Windows 기반 보안 메신저 실습
- Android 기반 보안 메신저 분석 방법론

Keywords



보안 메신저

로컬 기기

자동 로그인

안티 포렌식

리버스 엔지니어링

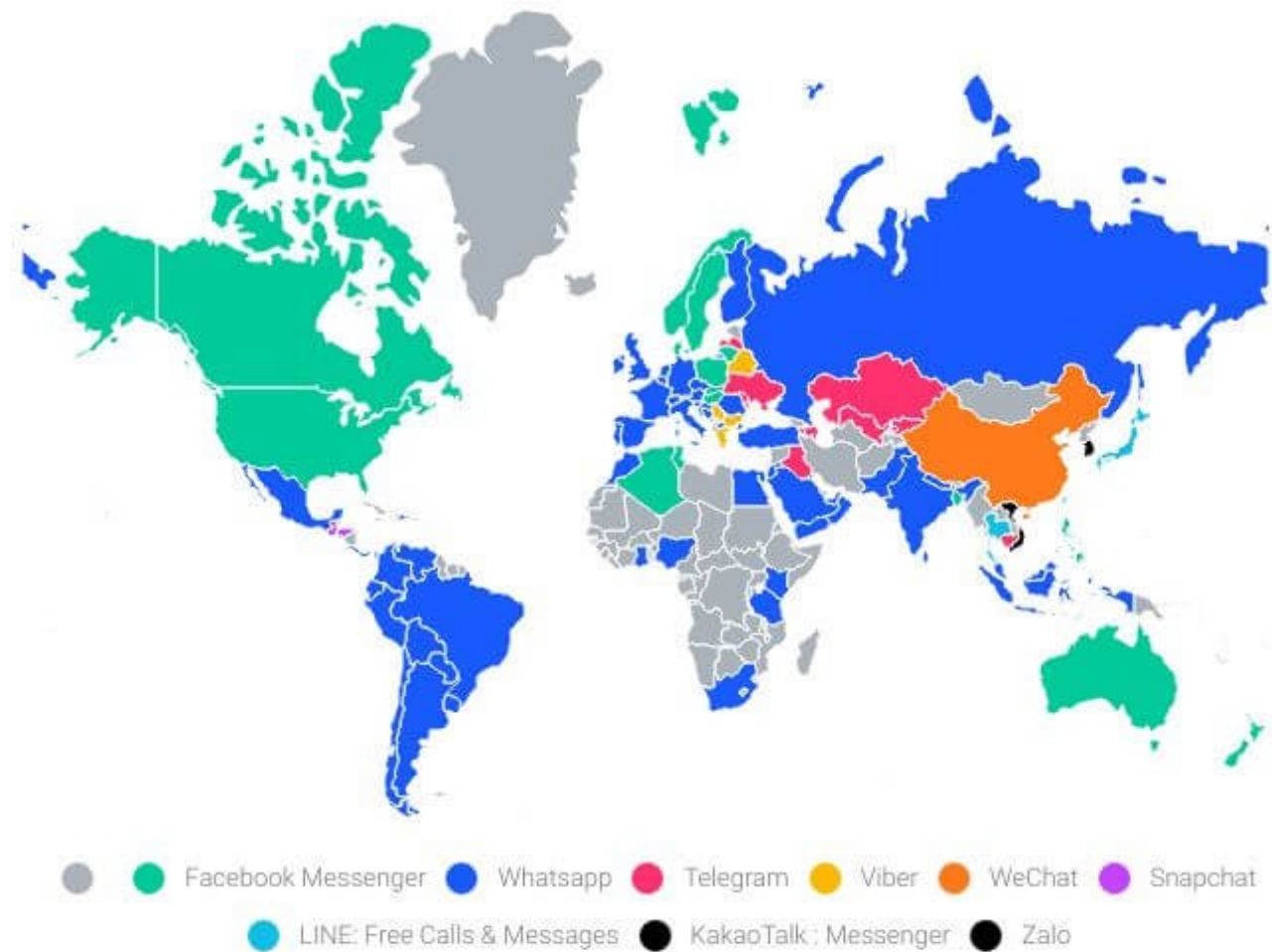
정적/동적 분석

보안 메신저 소개

보안 메신저 소개

보안 메신저

- 2022년 세계 보안 메신저 순위 (월간 활성 유저)

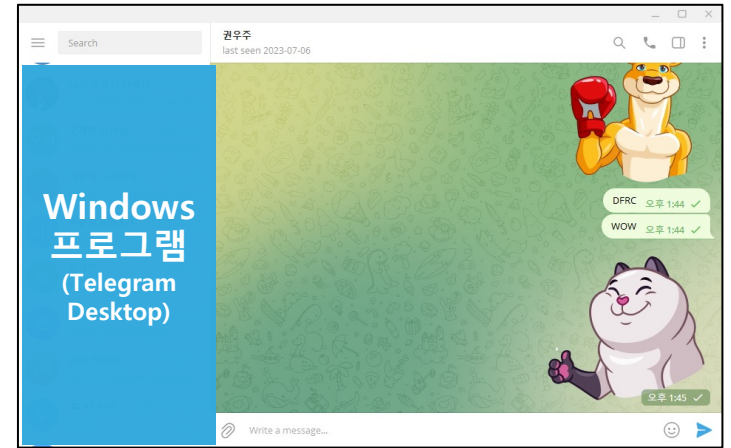
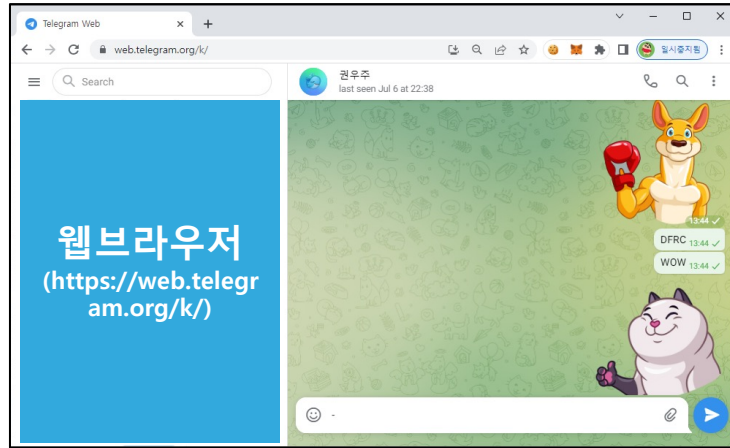
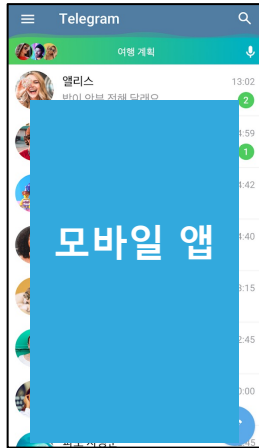


순위
1위: Whatsapp (러시아, 남미, 유럽)
2위: Facebook Messenger (북미, 호주, 유럽)
3위: WeChat (중국)
기타: Telegram, Line, KakaoTalk 등

보안 메신저 소개

보안 메신저

■ 보안 메신저의 다양한 형태



■ 사용자 데이터의 저장 위치

■ 로컬 기기 (Local) ★

- 로컬 기기에 저장된 데이터베이스 수집 및 해석, 데이터베이스 암호화 알고리즘 분석

■ 원격 서버 (Server)

- 데이터 수집을 위한 Open API 및 Internal API 분석, 메시징 프로토콜 및 웹 소켓 분석

보안 메신저 소개

[참고]보안 메신저 분석 결과: Windows

메신저명	데이터 위치	데이터 획득 방법	사용자 데이터	첨부 파일
Wire	<div>Android 기반 메신저에서는 로컬 기기에 데이터를 저장하는 경우가 더 많다</div>			○
Telegram X				○
Naver BAND				○
Facebook Messe				○
Wickr				○
Threema				○
Kakao Story				○
Instagram				○
Telegram Desk				○
TikTok				○

보안 메신저 소개

[참고]보안 메신저 분석 결과: Windows

- 메신저 분석 난이도
 - Level 1: 옆집 초등학생도 가능
 - Level 2: 약간의 지식만으로 가능
 - Level 3: 적당한 분석 능력이 필요
 - Level 4: 도.. 망.. 쳐..
- 실습 안내
 - 와이어 (Wire)의 IndexedDB로부터 복호화 키를 획득
 - 암호화된 캐시 파일 복호화

*Android 플랫폼에서는 로컬 기기에 데이터를 저장하는 경우가 더 많음

분석 난이도	데이터 저장 위치	
	로컬 기기	원격 서버
Level 1 (Nothing)	-	데이터 내보내기 (페이스북 등)
Level 2 (Easy)	 시그널 (Signal)	 카카오토리
Level 3 (Normal)	 와이어 (Wire) 	 네이버 밴드 페이스북 메신저 인스타그램 텔레그램 데스크톱
Level 4 (Hard)	 위커 (Wickr)	 틱톡 (TikTok)

로컬 크리덴셜을 이용한 복호화 기법

로컬 크리덴셜을 이용한 복호화 기법

자동 로그인

■ 메신저 동작 원리



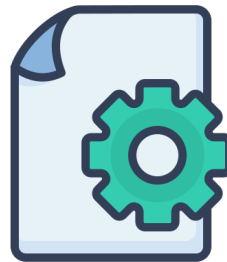
사용자



메신저 애플리케이션



데이터베이스



설정파일



캐시파일

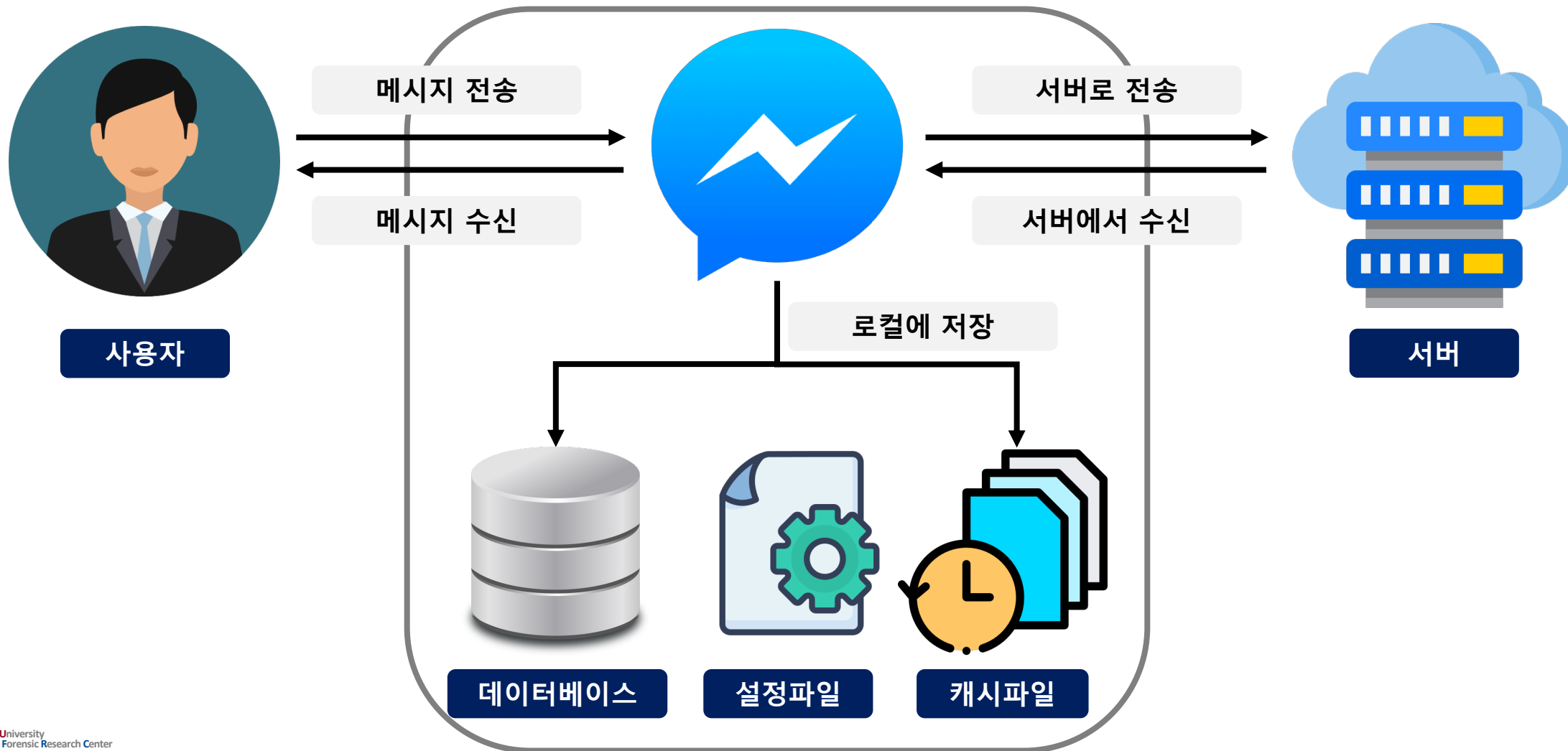


서버

로컬 크리덴셜을 이용한 복호화 기법

자동 로그인

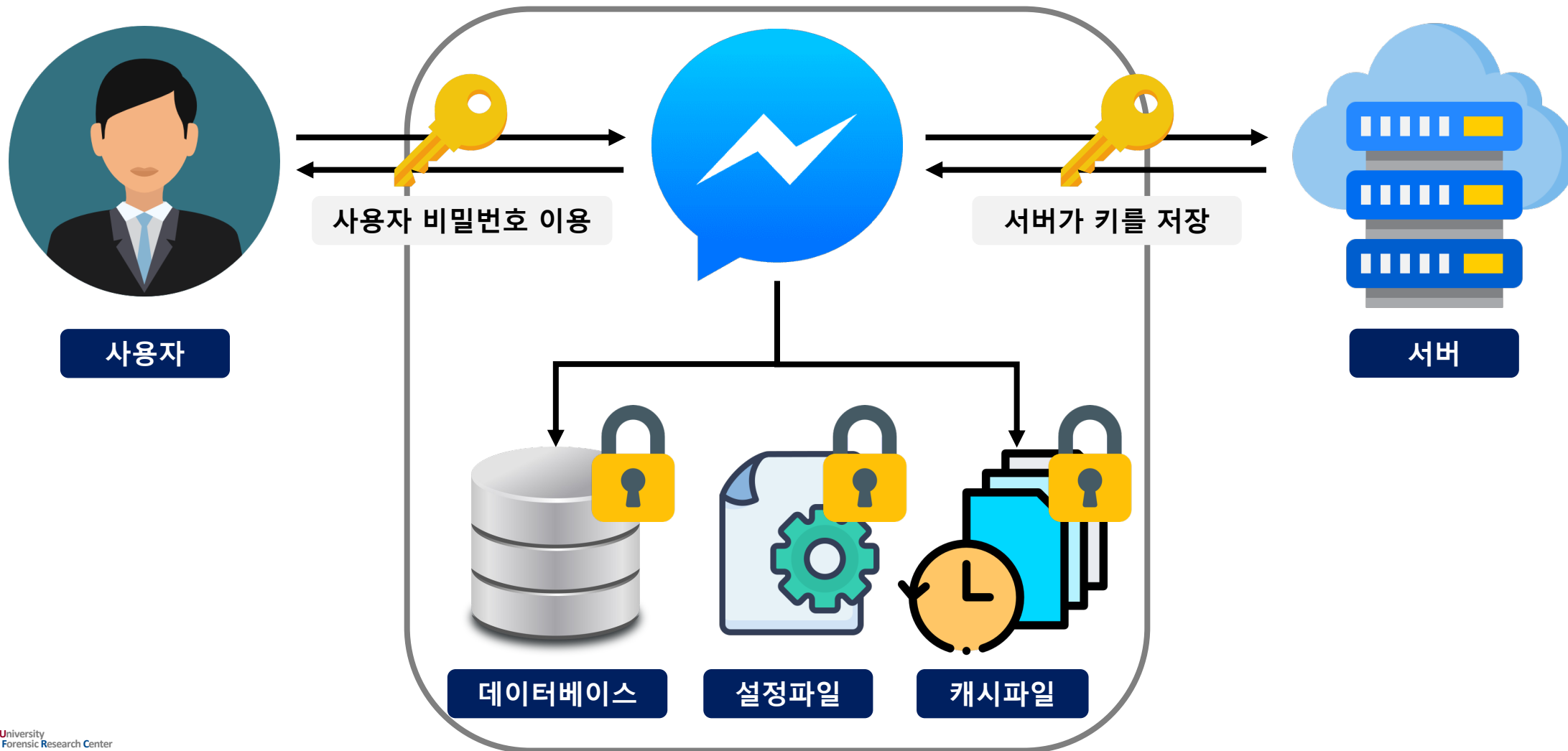
■ 메신저 동작 원리 (메시지 전송)



로컬 크리덴셜을 이용한 복호화 기법

자동 로그인

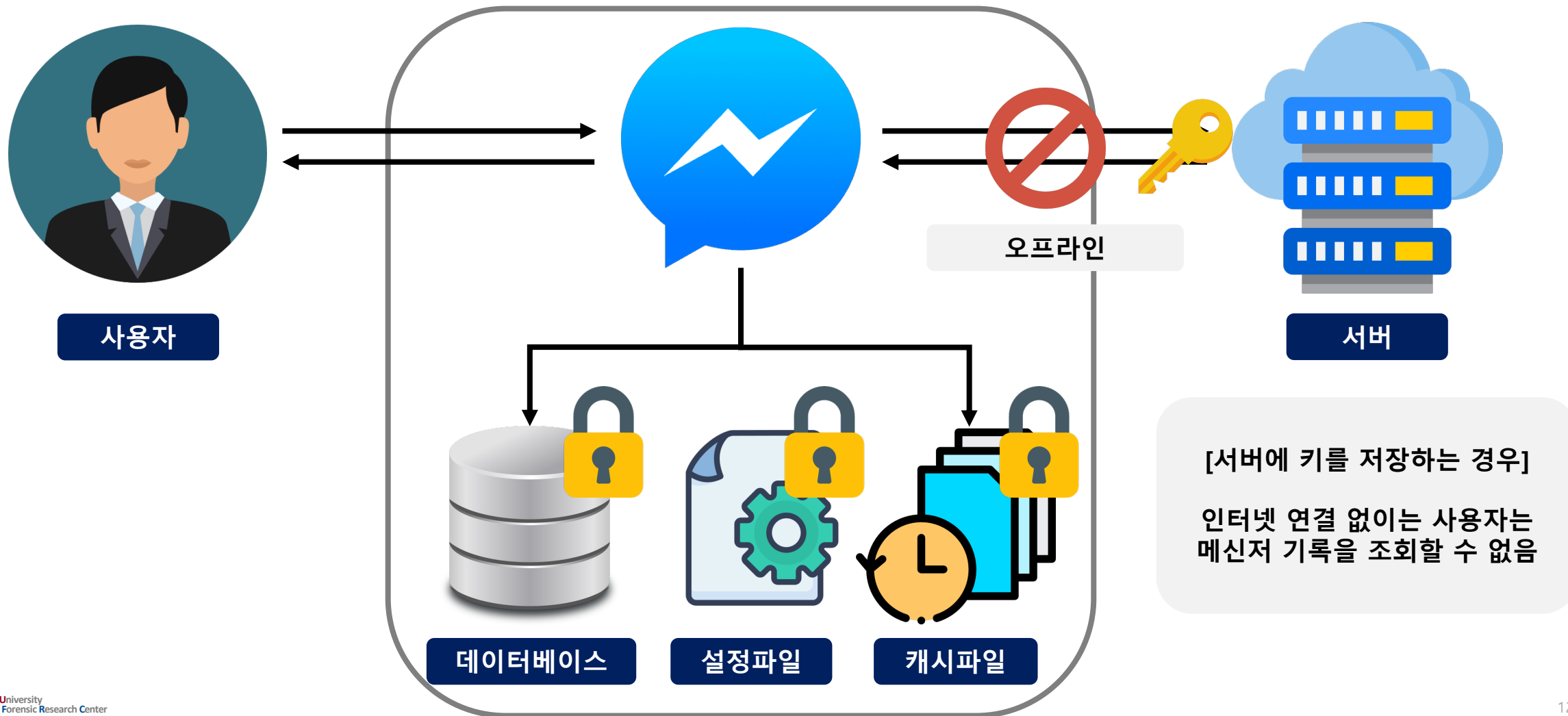
- 메신저 동작 원리 (암호화 적용)



로컬 크리덴셜을 이용한 복호화 기법

자동 로그인

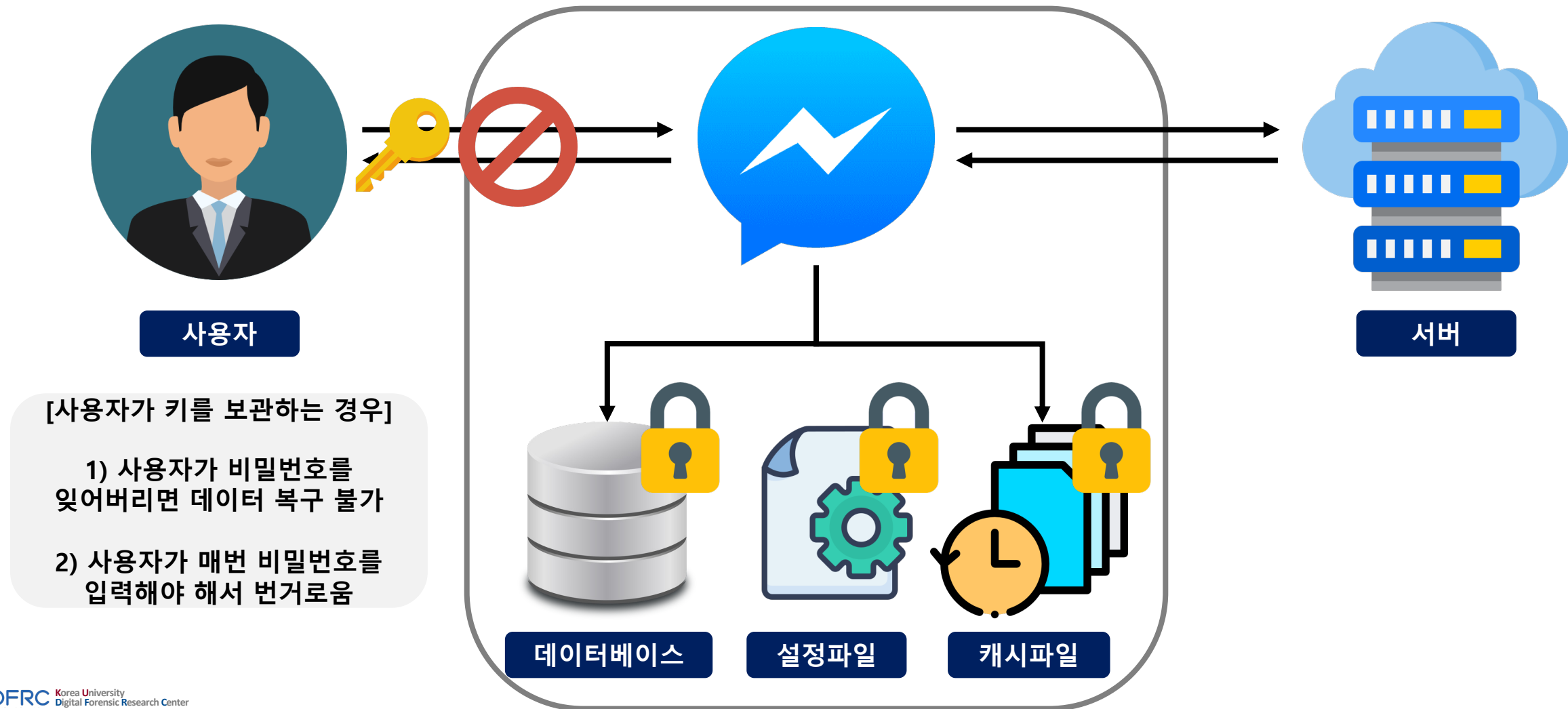
- 메신저 동작 원리 (암호화 적용: 오프라인 상황)



로컬 크리덴셜을 이용한 복호화 기법

자동 로그인

- 메신저 동작 원리 (암호화 적용: 사용자 비밀번호 기반)



로컬 크리덴셜을 이용한 복호화 기법

자동 로그인



우리 메신저가 **보안 메신저**로 보이기 위해서는 안티-포렌식 기법을 적용해야만 해. 사용자의 대화 내용, 이미지, 동영상, 문서 등 전부 **암호화**해서 저장해야지!

그런데, 서버에 **암호화 키**를 저장하면 **오프라인 사용이 불가능**해. 그렇다고 사용자가 매번 **비밀번호를 입력**하도록 하면 **사용자가 너무 불편**해질 것 같아...

로컬 크리덴셜을 이용한 복호화 기법

자동 로그인

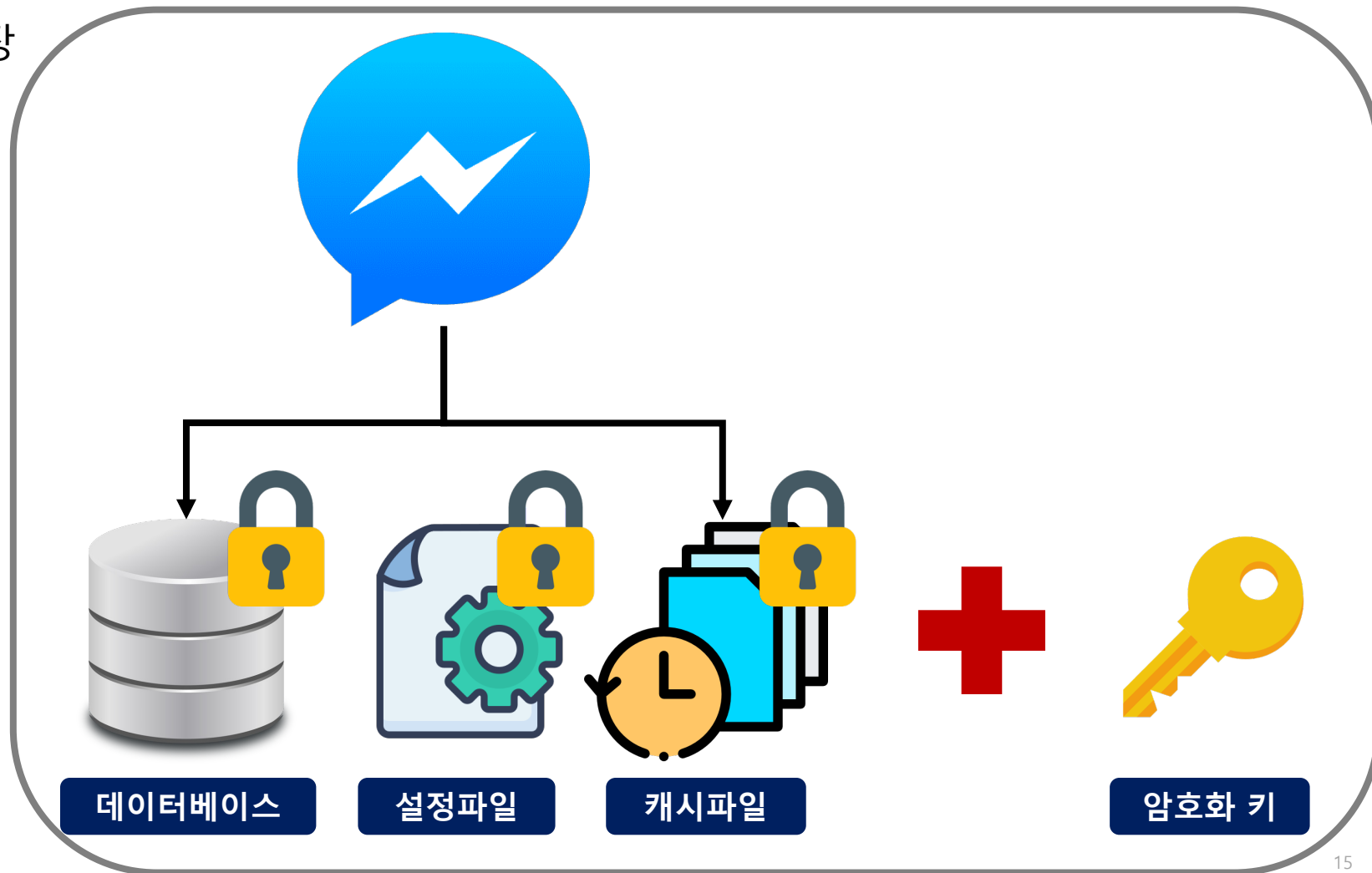
■ 로컬 기기 내부에 암호화 키를 저장

■ 암호화 키를 별도의 파일로 저장

- 암호화 키를 '대놓고' 저장하면
논란이 생길 수 있음
예) Signal Desktop

■ 비교적 덜 중요한 자산의 암호화를 풀고, 거기에 암호화 키를 저장

- 예) SharedPreferences



로컬 크리덴셜을 이용한 복호화 기법

자동 로그인

- 암호화가 적용된 보안 메시저의 자동 로그인 과정
 - '로컬 크리덴셜'이 있는지 확인
 - 있다면 로컬 크리덴셜을 이용해서 복호화 키 생성
 - 복호화 키를 이용해서 암호화된 사용자 데이터 복호화
 - '로컬 크리덴셜'이 없는 경우
 - 암호화된 데이터베이스 복호화 불가능
(일반적으로 종단간 암호화가 적용되어 있어, 서버도 키를 가지고 있지 않음)
- 사용자 회원 가입
 - 회원 가입 과정에서 '로컬 크리덴셜' 생성하여 기기 내부에 저장

Windows 기반 보안 메신저 서비스

Wire의 암호화된 첨부파일 복호화 실습

Windows 기반 보안 메신저 서비스

와이어 (Wire)

■ 메신저 특징

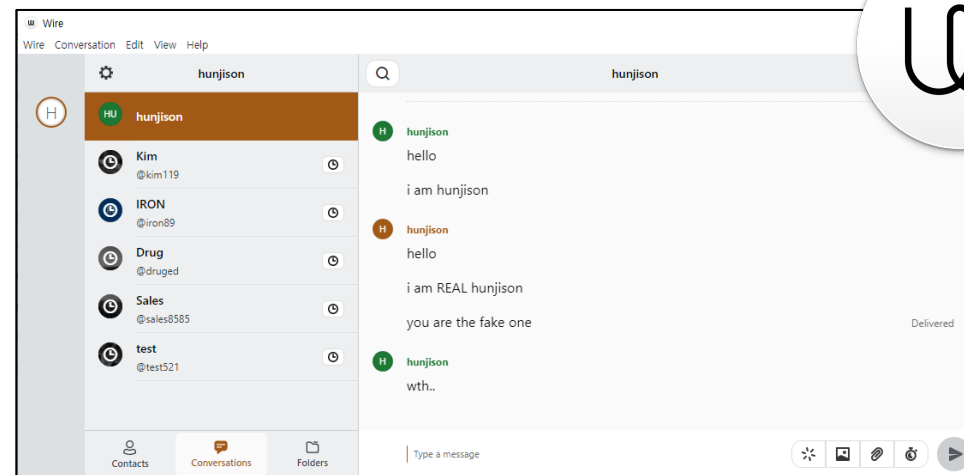
- 다양한 플랫폼 호환성 지원
 - Mobile (Android/iOS) 및 Desktop (**Windows**/macOS/Linux)

■ 와이어 (Wire) 아티팩트

- 대화내역 저장에 LevelDB를 이용
- 첨부파일은 별도의 경로에 캐시로 저장
 - 캐시는 암호화되지만, LevelDB 에서 키 발견 가능

LevelDB 경로: %UserProfile%\AppData\Roaming\Wire\IndexedDB\

첨부파일 경로: %UserProfile%\AppData\Roaming\Wire\Service Worker\CacheStorage



dfrc > AppData > Roaming > Wire > IndexedDB >		
이름	수정된 날짜	유형
000003.log	2023-07-13 오후 3:49	텍스트
CURRENT	2023-07-13 오후 3:22	파일
LOCK	2023-07-13 오후 3:22	파일
LOG	2023-07-13 오후 3:23	파일
LOG.old	2023-07-13 오후 3:22	OLD 파일
MANIFEST-000001	2023-07-13 오후 3:22	파일

대화내역

<< Roaming > Wire > Service Worker > CacheStorage > 37e27b3		
이름	수정된 날짜	유형
index-dir	2023-07-18 오후 9:06	파일 폴더
0ff9c001a7c50d7f_0	2023-07-18 오후 5:19	파일
2aa93fcb7eda92bb_0	2023-07-18 오후 5:16	파일
93a867a423a0d442_0	2023-07-18 오후 6:38	파일
2450c651522b8875_0	2023-07-18 오후 5:16	파일
7480b9fdb484b8b5_0	2023-07-18 오후 6:30	파일
a4cc3adddf962e3e_0	2023-07-18 오후 5:19	파일
a7703b7c8f262953_0	2023-07-18 오후 5:16	파일
def40213f5a99cd7_0	2023-07-18 오후 9:05	파일
index	2023-07-18 오후 5:16	파일

첨부파일

Windows 기반 보안 메신저 서비스

와이어 (Wire)

■ 분석 결과

- LevelDB 내에서 특정 구조 파싱
- 기기 접속 정보, 사용자 계정 정보 확인
- 대화방 정보, 대화 내역 확인
- 첨부파일 (캐시) 복호화 가능

■ 도구 개발

- 수집 및 복호화 Python 스크립트를 개발
- 수집 목적에 맞게 데이터베이스로 구성
- GUI 도구로 통합

기기 접속 정보

device	id	time	latitude	longitude	model
desktop	e7a41ea9445ba67d	2021-12-15 11:57:44.988Z			Windows 10 64-bit 10
phone	6d7af92a1daa8115	2021-08-05T11:54:17.132Z	37.5355	126.9766	sueun의 iPhone
desktop	7bca3f9a31b3599b	2021-11-17T13:03:29.390Z	37.6034	127.0065	Windows 10 64-bit 10
desktop	bb2e73476874830a	2021-12-15T15:55:44.962Z			Windows 10 64-bit 10
desktop	bd4f9e4ad8d007dc	2021-10-11T09:09:19.465Z	37.6034	127.0065	Windows 10 64-bit 10
desktop	ccb91da9994ba52a	2021-10-11T08:30:06.089Z	37.6034	127.0065	Windows 10 64-bit 10
desktop	f005241c36b7ca97	2021-11-09T07:21:48.737Z	37.6034	127.0065	Windows 10 64-bit 10

대화 내역

acc1aea1-7273-4b4f...	2021-1	message-add	test11
acc1aea1-7273-4b4f...	2021-11-10T07:13:54.634Z	conversation.message-add	test22
acc1aea1-7273-4b4f...	2021-11-10T07:13:55.764Z	conversation.message-add	test33
acc1aea1-7273-4b4f...	2021-11-10T07:13:57.508Z	conversation.message-add	test 44
acc1aea1-7273-4b4f...	2021-11-10T07:13:59.220Z	conversation.message-add	test 55

첨부파일 복호화

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	F8	0A	D4	0D	B7	A0	9E	8B	0B	93	55	7E	31	D1	51	B6	ø.Ö. · ž<."U~iŕog
00000010	AE	CE	78	7C	49	EE	EA	B6	EC	26	57	FD	F9	AA	77	1B	0ix Iiêgi&Wý
00000020	33	1B	06	15	3D	4B	80	15	82	4A	07	5A	6C	CA	0B	69	3...=K€,J.Z
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	00	00	48	ÿ0ÿà..JFIF...
00000010	00	48	00	00	FF	E1	00	58	45	78	69	66	00	00	4D	4D	.H..ÿá.XExif..MM
00000020	00	2A	00	00	00	08	00	02	01	12	00	03	00	00	00	01	.*.....

Windows 기반 보안 메신저 서비스

[실습] 와이어 (Wire) 첨부파일 복호화

■ 도구 다운로드

- Wire 데스크톱: <https://wire.com/en/download/>
- LevelDB 해석 도구: https://github.com/cclgroup/ltd/ccl_chrome_indexeddb

■ 주요 경로

- LevelDB 경로: %UserProfile%\AppData\Roaming\Wire\IndexedDB\
- 첨부파일 경로: %UserProfile%\AppData\Roaming\Wire\Service Worker\CacheStorage

■ 실습 과정

- LevelDB 내에서 첨부파일 이름, 암호화 키 (otr_key) 발견
- Cyberchef 이용하여 복호화 수행

Windows 기반 보안 메신저 서비스

[실습] 와이어 (Wire) 첨부파일 복호화

- Step1. 암호화된 캐시 파일 분석하기
 - Chromium "Simple Cache" 구조 (#####_0 파일)

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	30	5C	72	A7	1B	6D	FB	FC	05	00	00	00	54	01	00	00	0\r\x5C.m\u0072....T...
00000010	64	7C	89	5A	00	00	00	00	68	74	74	70	73	3A	2F	2F	d Z....https://
00000020	70	72	6F	64	2D	6E	67	69	6E	7A	2D	68	74	74	70	73	prod-nginz-https
00000030	2E	77	69	72	65	2E	63	6F	6D	2F	76	32	2F	61	73	73	.wire.com/v2/ass
00000040	65	74	73	2F	77	69	72	65	2E	63	6F	6D	2F	33	2D	35	ets/wire.com/3-5
00000050	2D	61	62	38	34	39	37	33	38	2D	38	35	34	33	2D	34	-ab849738-8543-4

Offset	Field Name	Size(bytes)	Description / Value
0x00 - 0x08	Magic	8	Simple Cache Magic Number
0x03 - 0x0A	URL Length	4	URL의 길이 / 0x154
0x0B - 0x0C	URL Data	-	URL 데이터 / https://..

Stream Start Offset: Header Offset(0x18) + URL Length

Windows 기반 보안 메신저 서비스

[실습] 와이어 (Wire) 첨부파일 복호화

- Step1. 암호화된 캐시 파일 분석하기
 - Chromium "Simple Cache" 구조 (#####_0 파일)

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00008C50	56	08	97	C3	F8	DA	95	FC	2E	CC	A5	AC	D8	41	0D	97	V.-ÃøÚ•ü.Ï¥-øA.-
00008C60	45	6F	FA	F4	01	00	00	00	3A	A3	0D	3A	F0	8A	00	00	Eouô.....:£.:ðŠ..
00008C70	00	00	00	00	0A	C6	02	0A	03	47	45	54	12	2B	0A	06Æ...GET.+..
00008C80	41	63	63	65	70	74	12	21	61	70	70	6C	69	63	61	74	Accept.!applicat
00008C90	69	6F	6E	2F	6A	73	6F	6E	2C	20	74	65	78	74	2F	70	ion/json, text/p
00008CA0	6C	61	69	6E	2C	20	2A	2F	2A	12	34	0A	09	73	65	63	lain, */*.4..sec

Offset	Field Name	Size(bytes)	Description / Value
0x00 - 0x08	Stream	-	캐시 Stream
0x03 - 0x0A	EOF Signature	8	Stream의 끝을 의미 / 0xD841..
0x0B - 0x0C	Response	-	응답에 대한 요청을 저장하는 캐시

Stream End Offset : EOF Signature Offset - 1

Windows 기반 보안 메신저 서비스

[실습] 와이어 (Wire) 첨부파일 복호화

■ Step2. IndexedDB 출력

- Github 'ccl_chrome_indexeddb' 다운로드
- 'dump_indexeddb_details.py' 소스코드 수정

```
for obj_store_name in db.object_store_names:
    obj_store = db[obj_store_name]
    print(f"\tobject_store_id={obj_store.object_store_id};
    try:
        one_record = next(obj_store.iterate_records())
    except StopIteration:
        one_record = None
    if one_record is not None:
        print("\tExample record:")
        print(f"\tkey: {one_record.key}")
        print(f"\tvalue: {one_record.value}")
    else:
        print("\tNo records")
    print()
print()
```

```
for obj_store_name in db.object_store_names:
    obj_store = db[obj_store_name]
    print(f"\tobject_store_id={obj_store.object_store_id};
    try:
        for message in obj_store.iterate_records():
            print(message.value)
    except StopIteration:
        one_record = None
    print()
print()
```

Windows 기반 보안 메신저 서비스

[실습] 와이어 (Wire) 첨부파일 복호화

■ Step3. 출력된 IndexedDB에서 암호화 키 확인하기

■ 주고 받은 대화 내용 검색해보기

```
{'conversation': '381bdb72-3908-4869-bec4-f06976f62b85', 'from': '8b241de3-8c32-4203-aa99-01186f249980',  
'from_client_id': <Undefined>, 'id': '697cd48f-b242-4e58-b31e-562ca2370c28', 'qualified_conversation':  
{'domain': 'wire.com', 'id': '381bdb72-3908-4869-bec4-f06976f62b85'}, 'qualified_from': <Undefined>, 'status':  
1, 'time': '2023-07-29T06:52:08.724Z', 'data': {'content': 'HELLO', 'mentions': []}, 'expects_read_confirmation': False, 'legal_hold_status': 1}, 'type': 'conversation.message-add'}
```



■ 'otr_key' 검색해보기

• 파일 이름, 크기 등 확인

```
{'conversation': '381bdb72-3908-4869-bec4-f06976f62b85', 'from': '8b241de3-8c32-4203-aa99-01186f249980',  
'from_client_id': <Undefined>, 'id': '2b730c7d-0354-4690-a012-14d37294a98f', 'qualified_conversation':  
{'domain': 'wire.com', 'id': '381bdb72-3908-4869-bec4-f06976f62b85'}, 'qualified_from': <Undefined>, 'status':  
2, 'time': '2023-07-29T11:47:10.207Z', 'data': {'content_length': 1838833, 'content_type': 'image/jpeg',  
'info': {'name': None, 'height': 2775, 'width': 4031, 'tag': 'medium'}, 'expects_read_confirmation': False,  
'legal_hold_status': 0, 'domain': 'wire.com', 'key': '3-5-bf100854-ad80-45b2-8113-0e58cf916da6', 'otr_key':  
(197, 251, 220, 82, 32, 3, 146, 39, 233, 3, 195, 60, 182, 157, 251, 99, 96, 104, 132, 189, 202, 204, 153, 241,  
56, 89, 174, 247, 18, 75, 91, 19), 'sha256': (229, 140, 137, 160, 95, 143, 171, 227, 69, 208, 60, 230, 135, 32,  
179, 114, 28, 61, 121, 185, 64, 101, 17, 23, 227, 250, 165, 12, 37, 59, 29, 151), 'status': 'uploaded',  
'token': ''}, 'type': 'conversation.asset-add', 'category': 128, 'primary_key': 5}
```


Windows 기반 보안 메신저 서비스

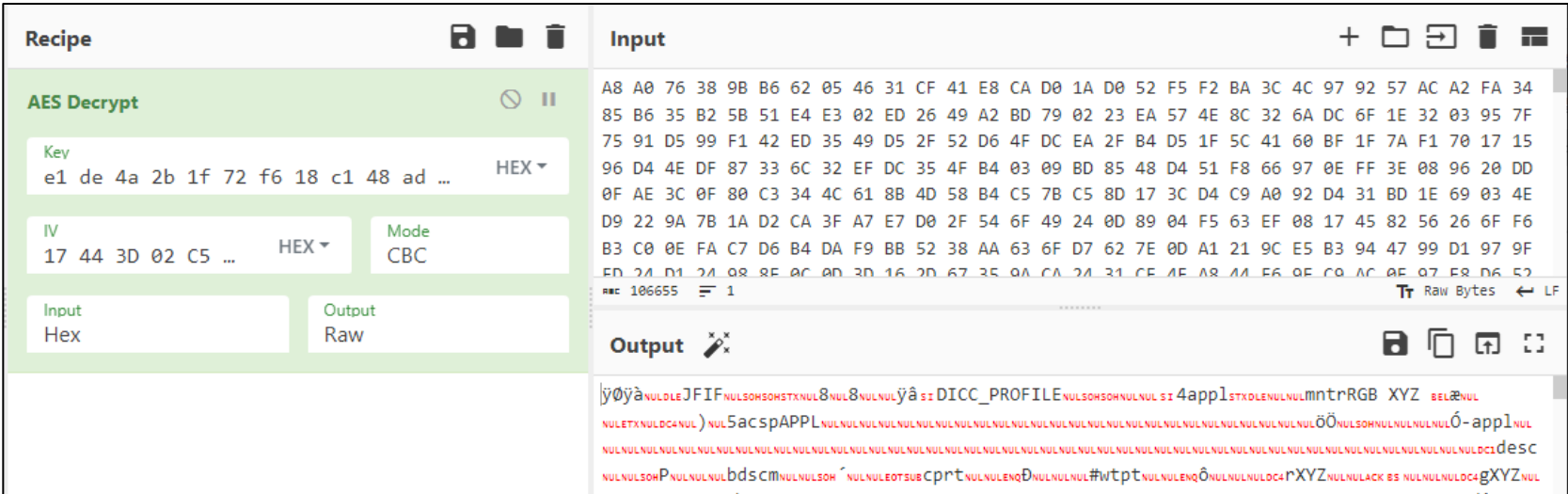
[실습] 와이어 (Wire) 첨부파일 복호화

- Step4. 암호화된 캐시 복호화 (CyberChef)

- Ciphertext, IV 확인하기

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	17	44	3D	02	C5	E0	D4	4B	EF	C9	D9	1A	EF	11	F7	6E	.D=.ÅàÔKïÉÛ.ï.÷n
00000010	A8	A0	76	38	9B	B6	62	05	46	31	CF	41	E8	CA	D0	1A	" v8>Jb.F1İAèÊĐ.
00000020	D0	52	F5	F2	BA	3C	4C	97	92	57	AC	A2	FA	34	85	B6	ĐRöð°<L-’W-çú4...J
00000030	35	B2	5B	51	E4	E3	02	ED	26	49	A2	BD	79	02	23	EA	5²[Qăă.í&Iç%y.#ê

- 'otr_key' 변환 (decimal to int)
- 복호화



Android 기반 보안 메신저 분석 방법론

정적/동적 분석 방법론 및 과제 설명

Android 기반 보안 메신저 서비스

안드로이드 앱 분석

■ 정적 분석

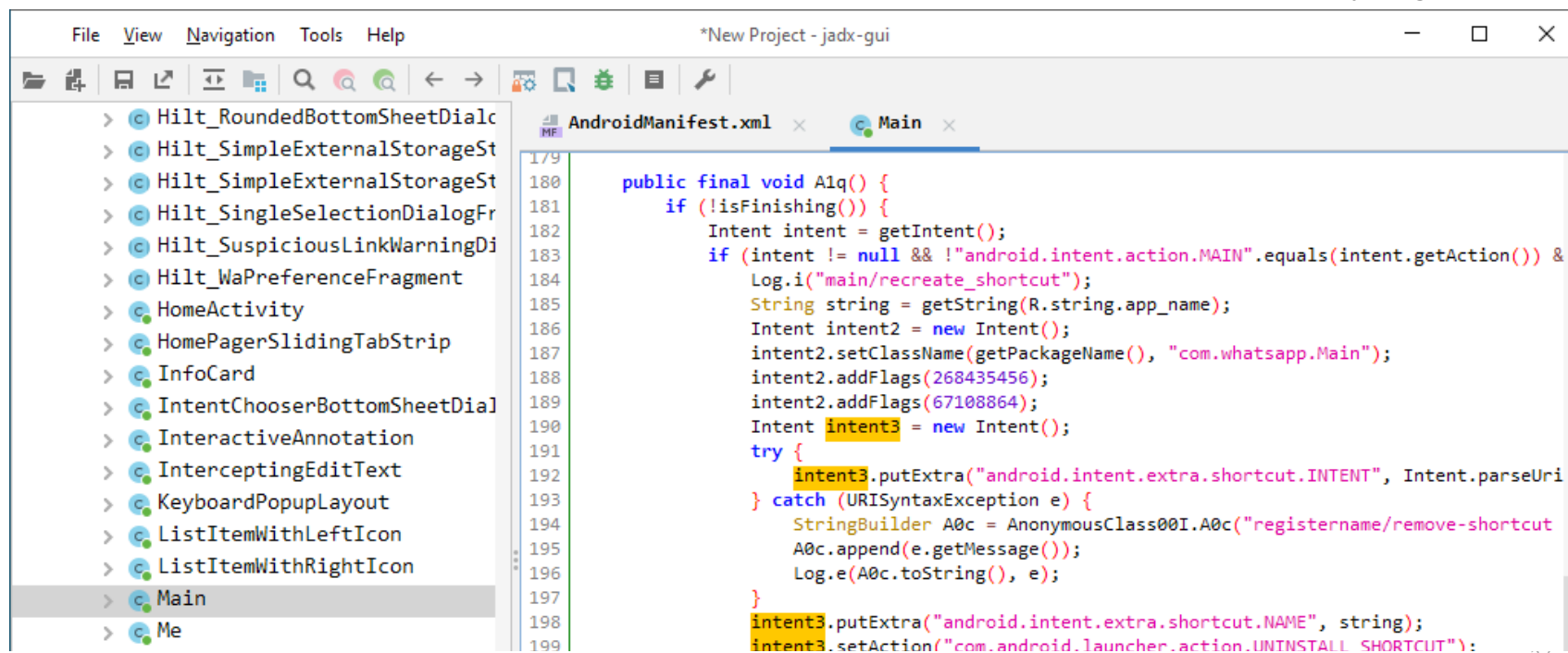
■ 애플리케이션 컴파일 및 디컴파일 과정

- 컴파일 과정: .java (.kt) → .class → .dex (Android APK)
- 디컴파일 과정: .dex → .smali → .java

■ 정적 분석 도구

- JEB (유료, 추천)
- jadx-gui (많이 사용됨)
- apktool
- Android Studio
 - 오픈 소스인 경우

jadx-gui 사용화면



The screenshot displays the jadx-gui application window titled '*New Project - jadx-gui'. The interface includes a menu bar (File, View, Navigation, Tools, Help) and a toolbar. On the left, a tree view lists various classes, with 'Main' selected. The right pane shows the decompiled Java code for the 'Main' class, starting with a 'public final void A1q()' method. The code includes logic for handling intents, specifically checking for 'android.intent.action.MAIN' and setting a class name to 'com.whatsapp.Main'. It also shows the creation of an intent and the addition of flags and extras. The code is color-coded for readability, with comments in Korean.

```
179
180 public final void A1q() {
181     if (!isFinishing()) {
182         Intent intent = getIntent();
183         if (intent != null && !"android.intent.action.MAIN".equals(intent.getAction()) &
184             Log.i("main/recreate_shortcut");
185         String string = getString(R.string.app_name);
186         Intent intent2 = new Intent();
187         intent2.setClassName(getPackageName(), "com.whatsapp.Main");
188         intent2.addFlags(268435456);
189         intent2.addFlags(67108864);
190         Intent intent3 = new Intent();
191         try {
192             intent3.putExtra("android.intent.extra.shortcut.INTENT", Intent.parseUri
193         } catch (URISyntaxException e) {
194             StringBuilder A0c = AnonymousClass00I.A0c("registername/remove-shortcut
195             A0c.append(e.getMessage());
196             Log.e(A0c.toString(), e);
197         }
198         intent3.putExtra("android.intent.extra.shortcut.NAME", string);
199         intent3.setAction("com.android.launcher.action.UNINSTALL_SHORTCUT");
```

Android 기반 보안 메신저 서비스

안드로이드 앱 분석

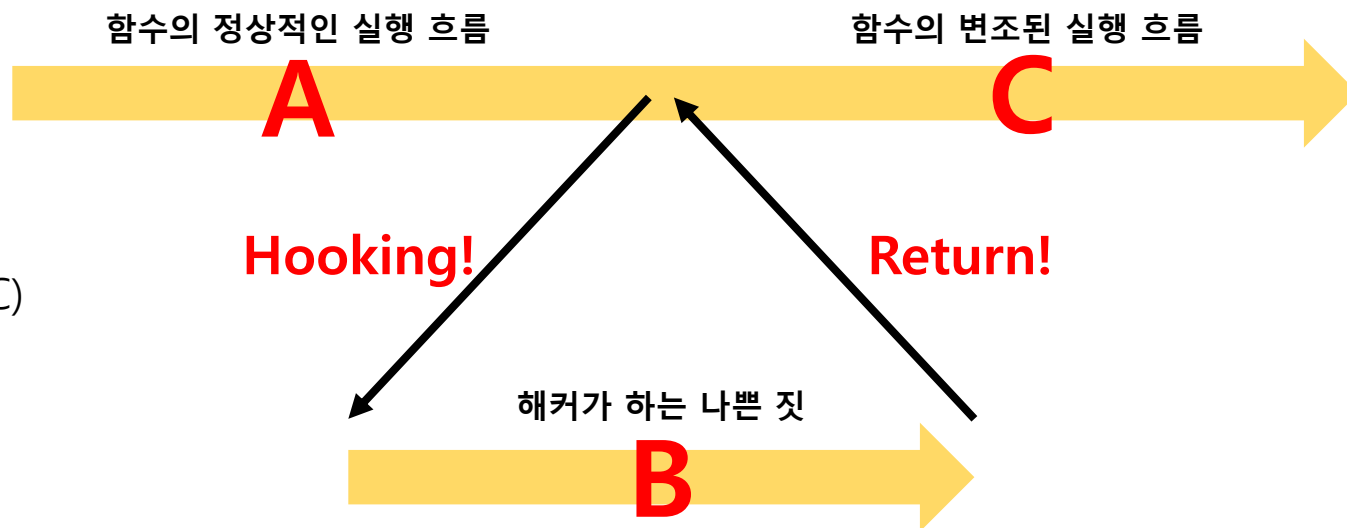
■ 동적 분석

■ 후킹 (Hooking)

- 함수의 실행 흐름을 잠시 가져와 (A)
어떠한 나쁜 짓을 수행하고 (B)
원래의 실행 흐름인 것처럼 되돌려 놓는 일 (C)

■ Frida

- 동적 분석 도구, 멀티-플랫폼 지원
- 안드로이드 애플리케이션 보안 모듈 우회 등에 자주 사용됨
- 비교적 쉽고, 간단하게 후킹 스크립트를 짤 수 있음
- 기회가 있다면 꼭 한 번 써보세요!



Android 기반 보안 메신저 서비스

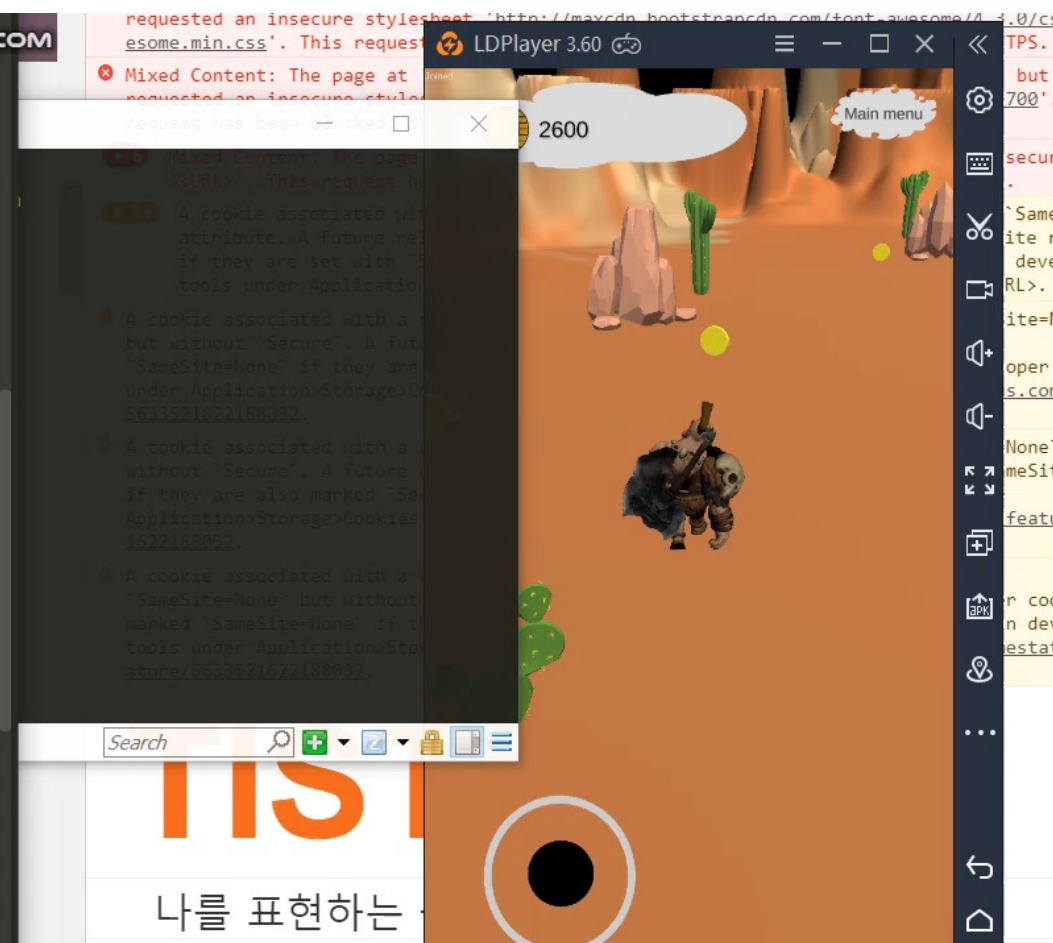
안드로이드 앱 분석

■ 동적 분석

■ Frida를 이용한 후킹

```
58     args[1] = ptr(1000);
59     },
60     onLeave : function(retval) {
61     };
62 }
63 });
64
65 var target_address2 = il2cpp.add(plus_gold);
66 var my_plus_gold = new NativeFunction(target_address2, "void", ["pointer"]);
67 send("public void plus_gold() @ "+ target_address2.toString());
68 Interceptor.attach(target_address2,{
69     onEnter : function(args){
70         send("onEnter! : plus_gold");
71         save = args[0];
72     },
73     onLeave : function(retval) {
74         for(var i = 0; i<100; i++){
75             setTimeout(my_plus_gold,1000,save);
76         }
77     }
78 });
79 }
80
81 Java.perform(function() {
82     send("Starting Script...");
83     check();
84     hook();
85 });
86
87
88 process = frida.get_usb_device().attach(PACKAGE_NAME)
89 script = process.create_script(jscode)
```

해킹캠프 23회, "Unity 게임, 어디까지 털어봤니?"



Android 기반 보안 메신저 서비스

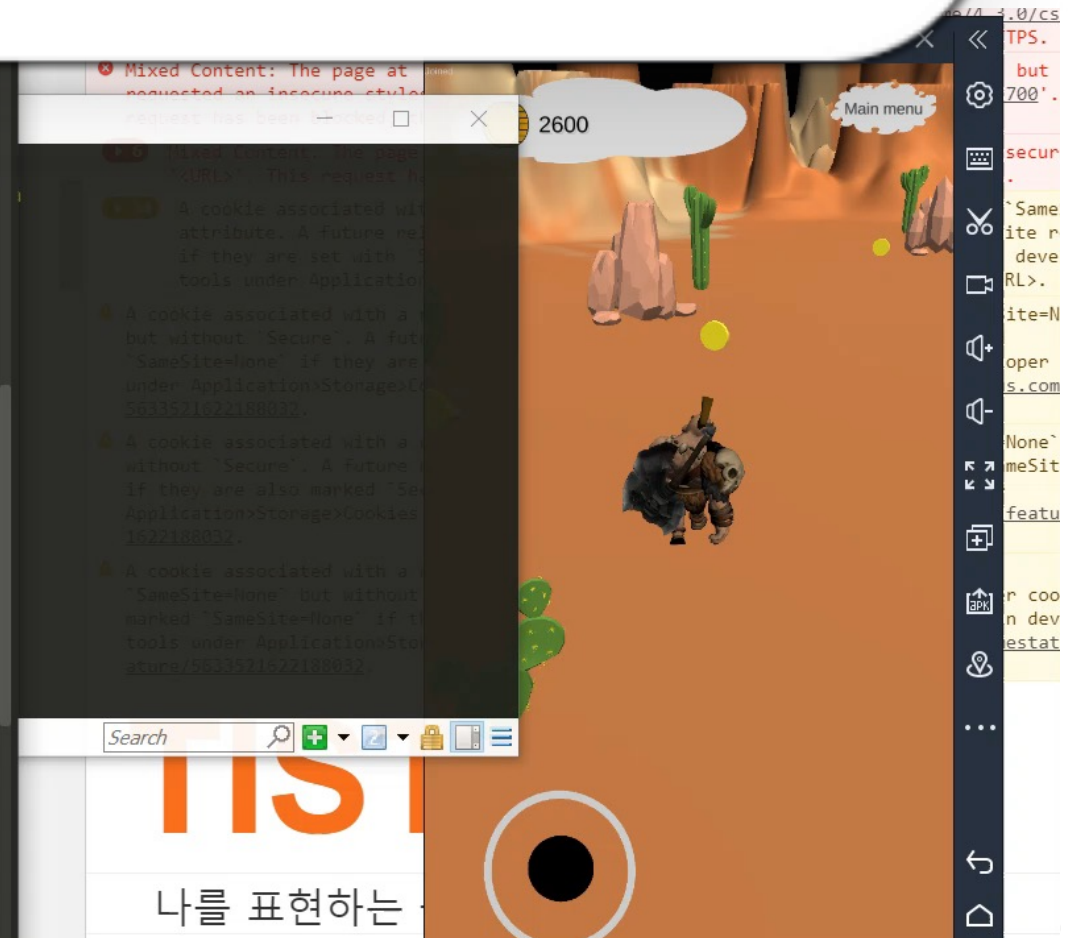
안드로이드 앱 분석

동적 분석

- Frida를 이용한 후킹

`console.log(return_value);`

```
58     args[1] = ptr(1000);
59 },
60 onLeave : function(retval) {
61     ;
62 }
63 });
64
65 var target_address2 = il2cpp.add(plus_gold);
66 var my_plus_gold = new NativeFunction(target_address2, "void", ["pointer"]);
67 send("public void plus_gold() @ "+ target_address2.toString());
68 Interceptor.attach(target_address2,{
69     onEnter : function(args){
70         send("onEnter! : plus_gold");
71         save = args[0];
72     },
73     onLeave : function(retval) {
74         for(var i = 0; i<100; i++){
75             setTimeout(my_plus_gold,1000,save);
76         }
77     }
78 });
79 }
80
81 Java.perform(function() {
82     send("Starting Script...");
83     check();
84     hook();
85 });
86
87
88 process = frida.get_usb_device().attach(PACKAGE_NAME)
89 script = process.create_script(jscode)
```



Android 기반 보안 메신저 서비스

안드로이드 기반 메신저 분석 방법론

■ 정적 분석

■ 함수 이름 기반 검색

- 예) openOrCreateDatabase(), password = 2nd param
- 예) AES 관련 함수들: SecretKeySpec(), IvParameterSpec(), cipher.init()

■ 키워드 기반 검색

- 로컬 기기에 저장된 수상한 파일 목록 리스팅 → 소스 코드 내에서 검색

■ 동적 분석

■ 정적 분석만으로 무언가 확실하지 않을 때 매우 유용

- 특히 난독화된 APK 혹은 JNI (Java Native Interface) 혹은 Native Library (.so) 분석할 때 효과적

■ 의심되는 함수 후킹 → 인자 값, 반환 값 확인

- 정적 분석에서의 분석 과정 흐름을 찾을 수 있음

Android 기반 보안 메신저 서비스

[과제] 쓰리마 (Threema) 로컬 크리덴셜을 이용한 복호화

■ Threema의 데이터베이스 암호화 방식 분석

■ 과제 내용

- Threema의 데이터베이스 암호화 키 생성 및 로컬 크리덴셜 저장 방식을 분석할 것
- 반드시 'threema-android'의 소스코드 정적 분석을 통해 원리를 파악하고 소스코드 캡처와 함께 설명할 것

■ 과제 수행 방법

- Github 'threema-android' 다운로드 후, Android Studio 이용하여 소스코드 분석
- 절대 남의 과제를 배껴서 내지 말 것

■ [힌트] 아래 사진의 'databaseKey' 부터 분석을 시작

- 하단 논문 참고해도 좋음

/app/src/main/java/ch/threema/storage/DatabaseServiceNew.java

```
final int databaseVersion;
try (SQLiteDatabase database = SQLiteDatabase.openOrCreateDatabase(oldDatabaseFile.getAbsolutePath(), databaseKey, null,
    if (database.isOpen()) {
        databaseVersion = database.getVersion();
        logger.info("Original database version: {}", databaseVersion);

        database.rawQuerySQL(
            "PRAGMA key = '" + databaseKey + "';" +
            "PRAGMA cipher_page_size = 1024;" +
```


경청해주셔서 감사합니다

Q & A

