안드로이드 시간 조작 취약점과 보안 문제

조건희 [1], 이연준 [2]
[1] 한양대학교 소프트웨어융합대학 컴퓨터학부 학부생
[2] 한양대학교 컴퓨터공학과 바이오인공지능 융합전공 교수
no1gun2@hanyang.ac.kr, yeonjoonlee@hanyang.ac.kr

Android Time Manipulation Vulnerabilities and Security Issues

Gun-Hee Cho[1], Yeonjoon Lee[2]
[1]College of Computing School of Computer Science, Hanyang University ERICA
[2] MAJOR IN BIO ARTIFICIAL INTELLIGENCE , DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

요 약

스마트폰에서의 시간 조작이 JVM 과 앱에 미치는 과정을 분석하여 세 가지 유형의 공격을 식별했다. 구글플레이 스토어의 990 개의 앱을 대상으로 앱의 취약성 분포를 정량화했으며 앱 개발자들에게 권장사항을 제공하고 있다.

I 서론

본 연구는 스마트폰 시간 조작의 취약성이 앱 보안에 미치는 영향에 대해 논의하고 있다. 스마트폰 설정의 변화가 자바 가상머신과 앱에 영향을 미치는 과정을 분석하여 사용자 활동 조작, 앱 기능 중단, 증거 조작이라는 세 가지 유형의 공격을 식별하였다. 구글 플레이 스토어의 990 개 앱에 대한 취약성 분포를 정량화하고 빠른 해결을 위한 대책을 제안하였으며 앱 개발자들에게 권장 사항을 제공하고 있다.

II 스마트폰 시간 조작의 취약점

1. 시간 조작이 앱에 영향을 미치는 과정

스마트폰에서 시간을 설정하는 것은 운영 체제에서 제공하는 기능 중 하나이며, 이는 스마트폰 전체에 영향을 준다.[1] 따라서 설정에서 시간을 바꾸면 JVM 도 영향을 받는다. 조작된 시간이 앱에 영향을 미치는 과정은 다음과 같다.

① 스마트폰 설정에서 [연도, 날짜, 요일, 시간]을 사용자가 임의로 조작.
② 조작된 시간은 즉각적으로 시스템(Java Virtual Machine)에 반영됨.
③ 앱에서는 m_code 를 통해 시스템 시간을 불러옴.
④ 연산, 매소드 호출 인자, 함수 반환 등 다양한 과정에서 조작된 시간이 사용됨.
⑤ 해당 함수가 실행될 때 개발자가 의도하지 않았던 데이터가 노출되거나 보안상의 문제를 야기함.

2. 공격 종류

i. 사용자 활동 유도:
(Tmap, 삼성 캘린더, 삼성 헬스, 네이버 지도 등) 시간을 조작함으로써 길 안내, 날씨, 캘린더와 같은 앱에 영향을 줄 수 있고 다른 경로로 안내, 다른 옷차림 유도, 일정 착각 등의 피해를 유발한다.

ii. 앱 기능 방해:
(삼성 인터넷, 토스, 요기요, 한국장학재단 출근부 등) 시간 데이터에 의존하여 엑세스 제어를 시행하는 앱은 시간 데이터가 손상되면 정상적인 앱 기능에 방해를 받는다. 만약 사용자가 악의적으로 시간을 조작하면 플레이 시간을 기준으로 보상을 주는 게임의 취약점을 악용할 수 있고, 인터넷 검색 히스토리를 조작할 수 있으며 개발자가 의도하지 않았던 앱 페이지로 접근이 가능하다.

iii. 증거 조작:
(갤러리, 전화, 메시지 등) 데이터 조사 과정에서 가장 중요한 타임라인 작성은 기기의 시간을 직접 조작함으로써 정상 경로를 통해 메타 데이터 값을 조작할 수 있다. 이렇게 조작된 메타 데이터들은 조사를 지연시키거나 혼선을 일으킨다[2].

III 앱 분석 및 대응 방안

Google Play 스토어 상위에 있는 990 개의 앱을 대상으로 테스트를 진행했으며 apktool 을 이용해 apk 파일을 smali 언어로 변환한 뒤 파이썬으로 정적 분석을 하여 취약한 포인트를 찾아냈다. 시간 조작 과정을 살펴보면 OS 의 시간 값을 참조하기 위해 안드로이드 프로그램은 m_code 를 활용한다.

m_code

m_code 는 시스템의 값을 프로그램에서 읽을 수 있도록 시간 값을 리턴해주는 함수이며 그 종류는 아래와 같다.

["currentTimeMillis()", "getTime()", "getDate()", "getDay()", "getHours()", "getMinutes()",

"getMonth()", "getSeconds()",
"getTimezoneOffset()", "getFullYear()",
"getYear()", "getUTCDate()", "getUTCDay()",
"getUTCHours()", "getUTCMilliseconds()",
"getUTCMinutes()", "getUTCMonth()",
"getUTCSeconds()"]

Case 분류

m_code를 호출한 뒤 해당 시간 값을 할당받은 변수를 추적하여 각 앱당 아래의 Case가 얼마나 존재하는지 확인했다.

- Case 1: m_code 호출 후 move로 할당 받는 경우(할당만 받고 쓰이지 않음)
- Case 2: m_code 연산 값이 다른 값으로 덮어쓰이는 경우
- Case 3: m_code 연산 값이 다른 값에 연산되는 경우
- Case 4: m_code 연산 값이 다른 매소드의 호출 인자로 쓰이는 경우
- Case 5: m_code 연산 값이 해당 함수의 리턴 값인 경우
- Case 6: m_code 연산 값이 필드 값에 직접 영향을 주는 경우

이 중 Case 4, Case 5, Case 6이 보안상에 문제를 일으키는 요소이며 해당 취약점의 **possible point of attack**이 된다. 특히 Case 6은 시간 값을 필드 전역 변수에 직접 할당하는 것으로 검증 과정을 거치지 않기에 가장 큰 취약점이 된다.

Case 조사

앱당 평균적으로 31개의 Case 6 취약점이 있으며 OS의 시간을 할당받고 전혀 사용하지 않는 불필요한 Case 1도 앱당 평균 8개나 됨을 알 수 있다.

	1	2	3	4	5	6
count	990.000000	990.000000	990.000000	990.000000	990.000000	990.000000
mean	12.446465	170.970707	363.672727	296.325253	188.303030	52.263636
std	18.988226	211.301456	468.698389	405.409370	220.624129	65.988219
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2.000000	28.000000	60.000000	44.250000	30.000000	11.000000
50%	8.000000	103.500000	228.000000	185.000000	127.000000	31.000000
75%	18.000000	239.750000	506.500000	415.500000	272.750000	69.000000
max	393.000000	1920.000000	5571.000000	6416.000000	2830.000000	696.000000

그림 1

권고사항

시간 조작 취약점을 완화하기 위해서는 몇 가지 고려 사항이 존재한다. 온라인 앱은 서버와 시간 동기화 주기를 짧게 설정하여 피해를 완화할 수 있지만 서버 비용이 증가할 수 있다. 오프라인 앱은 서버가 없기 때문에 기기 자체에서 이를 해결해야 한다. 이러한 점을 고려했을 때 설정에서 [자동 시간 설정] 항목이 해제되어 있는 경우, 현재 기기의 시간 값이 마지막으로 활동한 시간보다 과거로 갔을 때 등 특정 상황을 이용해 앱을 제어하면 된다. 아래는 앱에서 시간 조작을 방어하는 예시 코드이다.

```
boolean autoTime = Settings.Global.getInt(getContentResolver(),
        Settings.Global.AUTO_TIME, def: 1) == 1;
boolean autoTimeZone = Settings.Global.getInt(getContentResolver(),
        Settings.Global.AUTO_TIME_ZONE, def: 1) == 1;
if (autoTime && autoTimeZone) {
    return true;
SimpleDateFormat dateFormat = new SimpleDateFormat(
        pattern: "yyyy-MM-dd HH:mm:ss", Locale.getDefault());
String current = getSharedPreferences( name: "MY_PREFS_NAME", MODE_PRIVATE)
        .getString( key: "CURRENT_TIME", defValue: "");
String last = getSharedPreferences( name: "MY_PREFS_NAME", MODE_PRIVATE)
        .getString( key: "LAST_TIME", defValue: "");
Date currentTime = dateFormat.parse(current);
Date lastTime = dateFormat.parse(last);
if(currentTime.getTime() >= lastTime.getTime()) {
    isValid = true;
```

그림 2

최근에는 블록체인, 5G, 블루투스 등과 같은 다양한 기술이 시간 조작에 의한 보안 취약점을 방지하기 위해 개발되었지만[3][4], 즉시 도입되지 않았고 오프라인 환경에서 실행되는 앱들은 보안 기술의 범위 외에 존재하여 여전히 위험이 남아있다. 하지만 위와 같은 코드를 사용하면 즉각적 패치가 가능하고 온라인 환경뿐 아니라 오프라인 환경에서도 사용자 활동을 보다 상세히 제어할 수 있다.

IV 결론

해당 연구는 구글 플레이 스토어에서 상위 990개의 앱을 대상으로 시간 조작 취약점을 조사했으며 앱당 평균 350개의 case 4,5,6이 있었다. 정적 분석을 통해 찾은 취약점으로 실제 앱에서 동적 분석을 진행했고 삼성 기본 앱, 토스, 네이버 지도, 요기요 등의 앱에서 취약점이 실존함을 밝혔다. 해당 취약점의 빠른 패치를 위해 앱 자체에서 시간 조작을 탐지하여 사용자의 활동 범위를 제어하는 방어 코드를 추가하여 피해를 완화할 것을 제안하고 있다.

"본 연구는 2023년 과학기술정통신부 및 정보통신기획평가원의 SW중심대학지원사업의 연구결과로 수행되었음"(2018-0-00192)"

V 참고문헌

1. Young-Min Yoon, Myeong-Seon Gil, Jin-Woo Jung, Hyun-Kyo Lim, "Android time settings vulnerabilities and security issues.", In 2020 International Conference on Electronics, Information, and Communication (ICEIC), Jeju Island-South Korea, 2020, pp. 1-4.
2. 장명진, 이상훈, 최지훈, 이상균, "메타데이터 조작이 스마트폰 디지털 증거의 타당성에 미치는 영향", 한국경찰학회보, 19권, 4호, pp.47-72, 2020
3. 이영희, "스마트폰 시간 조작 방지를 위한 5G 통신 기술 활용 연구", 정보보호학회 논문지, 26권, 3호, pp.200-215, 2021.
4. 박민수, "블루투스를 활용한 스마트폰 시간 조작 탐지 및 방지 기술 연구", 한국통신학회 논문지, 44권, 5호, pp.832-846, 2020.