[학부생논문]Dirtycow 취약점 기반 vold 권한탈취를 이용한 안드로이드 사용자 데이터 초기화

나성훈○ 석호식
강원대학교 컴퓨터정보통신공학과
nsh1189@gmail.com hsseok@kangwon.ac.kr

Android system user-data wiping through Dirtycow vulnerability based vold privilege escalation

Na SungHun Seok Ho-Sik
Department of Computer and Communications Engineering
Kangwon National University

요 약

Dirtycow 취약점(CVE-2016-5195)은 리눅스 커널의 메모리 서브 시스템이 읽기 전용 메모리 매핑의 COW (Copy-on-write) 장애를 처리하는 방식에서 발견된 취약점으로, 사용자의 권한에 관계 없이 해당 파일에 대한 read 권한이 있다면 해당 파일 수정을 가능하게 한다. 본 논문에서는 Dirtycow 취약점을 활용한 안드로이드 시스템 공격이 가능하다는 점을 보이기 위하여, Dirtycow 취약점을 이용하여 사용자 데이터 블록의 기록 권한을 가진 데몬을 인젝션한 후 모든 데이터를 삭제하는 악성 코드를 구현하고 성능을 확인하였다. 구현 결과 커널의 자원 공유 처리 방식의 허점을 통해 사용자 권한 제약을 극복하면 공격자가 자유롭게 시스템상의 데이터를 삭제할 수 있음을 확인하였다.

1. 서 론

최근 스마트폰의 보급률이 높아지면서 사용자들은 민감한 개인 정보들을 스마트폰에 저장하기 시작했다. 대부분의 사람들이 사용하는 스마트폰의 운영체제는 안드로이드(Android) 와 iOS로 구분된다. 이 중 안드로이드 운영체제는 Linux 커널에 기반한 시스템이기 때문에 linux kernel에 존재하는 취약점이 모두 악용될 수 있다는 문제가 있다.

본 논문에서는 linux 커널에서 발견된 Dirtycow 취약점을 이용하여 안드로이드 시스템을 공격할 수 있는 방법을 소개한다. Dirtycow 취약점(CVE-2016-5195)은 리눅스 커널의 메모리 서브 시스템이 읽기 전용 메모리 매핑의 COW (Copy-on-write) 장애를 처리하는 과정에 존재하는 취약점으로, 낮은 권한의 일반 사용자가 Dirtycow 취약점을 사용하면 최초 권한에 관계 없이 읽기 전용 메모리시스템에 대한 쓰기 권한을 획득할 수 있다[1]. 단, 취약점을 통해 수정한 파일(읽기전용)은 실제 저장소에 기록되는 것이 아닌, 로드된 메모리 정보만 바뀌는 것이므로 재부팅이나 저장소를 리마운팅할 경우 이 취약점으로 인해 수정된 변경사항은 사라지게 된다. Dirtycow 취약점은 Linux kernel 2.6.22 이전부터 존재한 취약점이고 초기 안드로이드 시스템은 커널 2.6.25에 기반하여 구현되었기 때문에 이전 안드로이드 시스템에도 Dirtycow 취약점은 존재하고 있다. 따라서 Dirtycow 취약점을 이용하여 일부 안드로이드 디바이스의 루트 권한을 탈취할 수 있다는 사실이 이미 보고되었다[1].

Dirtycow 취약점을 다양한 방식으로 활용하여 안드로이드 시스템을 공격할 수 있지만 우리는 특히 사용자 데이터 파티션 접근이 가능한 vold (volume daemon)를 활용한 공격 방식에 주목하였으며, Dirtycow 취약점을 활용하여 vold의 권한을 탈취하고 사용자 데이터를 삭제할 수 있는 악성 코드를 제작하고 성능을 확인하였다.

논문의 구성은 다음과 같다. 2장에서는 Dirtycow 취약점을 활용한 대표적인 공격 사례들을 소개한다. 3장에서는 사용자 데이터 공격을 위해 활용 가능한 데몬을 소개한 후 Dirtycow 취약점을 활용하여 데몬의 권한을 탈취하는 방법을 소개한다. 4장에서는 연구내용을 요약하고 추후 연구 방향을 제시한다

2. 관련 연구

실제 공격 방법을 소개하기 전에 Dirtycow 취약점을 이용한 기존의 공격 사례를 소개한다.

DirtySanta: Dirtycow 취약점을 활용하여 LG V20의 루트

[1] http://blog.alyac.co.kr/860

권한을 획득하는 공격이다. LG 전자의 개발진이 개발 편의성을 위해 최고 권한을 부여한 /system/bin/atd 바이너리를 인젝션(injection, 해당 바이너리의 내용을 원하는 기능의 바이너리로 덮어씌워 수정한다는 의미)하여 잠금 해제된 엔지니어 부트로더와 DM-Verify가 해제된 부트이미지(boot.img)를 플래싱한다. 이 과정을 통해 안드로이드 OS의 부트이미지 시그네쳐 검사, DM-Verify 시스템 무결성 검사를 모두 통과할 수 있기에, root 바이너리 삽입진행이 가능해지게 된다[3,7].

Viki Root: vDSO (virtual dynamic shared object)를 탈취하여 레이스 컨디션을 발생시킨 후 shellcode를 동작시킨다. 커널단에서 SELinux를 무력화시킨 후 root 바이너리를 삽입한다[4].

Universal Backup TA: Sony 기기의 /sbin/tad_static을 인젝션한 후 기기 고유의 DRM 정보, 보증 정보 등을 담은 TA 파티션을 덤프하여 백업한다[5].

Patch/init: 초기 부팅 과정에서 /init 데몬은 SELinux 컨텍스트가 정의된 SEPolicy를 로딩하며 따라서 /init 데몬(u:r:init:s0)은 전체 SELinux 체계를 수정할 수 있는 권한이 있다. 이런 내용을 활용하여 /init 접근 권한이 있는 기기에서, /init에서 항시 실행되는 함수인 bootchart_sample()이 새로운 sepolicy 를 로드하도록 함수의 내용을 바꿈으로써 SELinux 권한 체계를 원하는대로 변조시킨다[6].

3. Android 권한 분석 및 탈취

3장에서는 안드로이드 시스템의 기본 SELinux 체계[8]를 분석하고 AOSP 소스 내에 명시된 컨텍스트들의 권한을 분석하여 원하는 시스템 공격 방식에 활용할 수 있는 바이너리를 선정하는 과정을 설명한다. 그리고 바이너리 선정 후 저장소 관리 데몬의 권한을 탈취하는 과정과 사용자 데이터 저장공간 초기화 방법을 소개한다.

3.1. 권한 분석

안드로이드 내의 SELinux 권한은 각 데몬의 필요에 따라 한정적으로 제공된다. 이를테면 카메라 처리를 관장하는 cameraserver 는 root 권한으로 실행되지만, SELinux 가 한정하는 범위 내인 /data/misc/camera 폴더에서만 파일을 생성하고 삭제할 수 있다. 또한 장치 드라이버나 시스템 파일 접근에도 제한이 존재하는데, cameraserver 는 SELinux 컨텍스트 가 video_device, ion_device 인 경우와 컨텍스트 내에 명시된 서비스 등에만 한정적인 접근이 가능하다. 안드로이드에서 기본적으로 지정된 컨텍스트와 규칙은 AndroidSRC/system/sepolicy 내에 정의되어 있다.

우리는 **사용자 데이터 초기화**를 목표로 하므로 많은 데몬들 중 userdata 파티션에 접근하고 block 읽기/쓰기 권한을 가진 데몬이 필요하다. 이 과정에서 도움이 되는 것이 안드로이드의 FDE 지원이다.

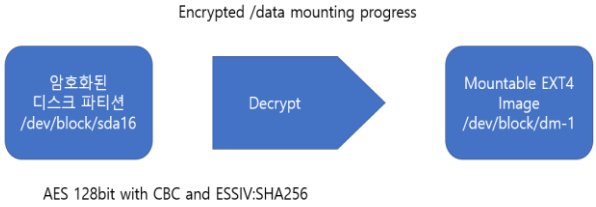

그림 1. FDE를 이용한 암호화된 데이터 파티션 마운트 과정

안드로이드에서는 FDE(Full Disk Encryption, [9])를 지원하는데, 그 과정에서 그림 1과 같이 userdata 파티션의 암호를 해독한 후 mount 하는 과정이 필수적이다. /data 파티션은 R/W 로 마운팅되어야 하기에, FDE 진행을 위하여 userdata 파티션 읽기/쓰기 권한은 필수적이므로 이 데몬의 컨텍스트에는 userdata 읽기/쓰기 권한이 포함되어 있을 것이다. 해당 데몬을 찾아 SELinux context 를 확인해본다면 정확한 정보를 알 수 있게 된다.

```
151  # talk to keymaster
152  allow vold tee_device:chr_file rw_file_perms;
153
154  # Access userdata block device.
155  allow vold userdata_block_device:blk_file rw_file_perms;
156
157  # Access metadata block device used for encryption meta-data.
158  allow vold metadata_block_device:blk_file rw_file_perms;
```

그림 2. AndroidSRC/system/sepolicy/vold.te

저장소 암호화를 포함하여 대부분의 저장소 관련 일을 담당하는 데몬인 vold 의 SELinux context 에는 그림 2와 같이 userdata_block_device 를 읽기/쓰기 할 수 있는 권한이 포함되어 있다. 이 데몬은 userdata 파티션에 대한 읽기/쓰기 권한을 가지고 있으므로, 이를 인젝션 한다면 사용자 데이터를 초기화시킬 수 있을 것이다.

3.2. 권한 탈취

본 논문의 서론에서 설명하였듯이, Dirtycow 취약점은 파일의 read 권한이 있어야 해당 파일을 수정 가능하다.

3.1절에서 찾은 /system/bin/vold는 특수 context 를 가진 파일로, 기본적인 u:r:shell:s0 권한으로는 접근이 불가능하다.

vold 는 현재 context 로는 읽기 불가능하므로 다른 context 로 권한 변경이 필요하다.

```
18
19  # run-as switches to the app UID/GID.
20  allow runas self:capability { setuid setgid };
21
22  # run-as switches to the app security context.
23  # read /seapp_contexts and /data/security/seapp_contexts
24  security_access_policy(runas)
25  selinux_check_context(runas) # validate context
26  allow runas self:process setcurrent;
27  allow runas non_system_app_set:process dyntransition; # setcon
28
```

그림 3. AndroidSRC/system/sepolicy/runas.te

권한 변경을 위하여 context 변경(setcon)이 가능한 바이너리를 인젝션한다. AOSP 소스의 sepolicy 에는 그림 3과 같이 run-as 컨텍스트에서 App 수준의 context 변경이 가능하다고 명시되어 있다. Dirtycow 취약점을 활용하여 run-as를 공격자가 원하는 권한으로 setcon 진행하게끔 변조시킨다면, 이를 통해 원하는 App 수준으로의 권한 변경이 가능해진다.

/system/bin/vold는 App 수준의 권한인 u:r:untrusted_app:s0 으로 변경 시 접근 (read) 가능했으며, 삼성 갤럭시의 경우에는 권한 변경 없이 기본 셸 권한(u:r:shell:s0) 으로도 접근 가능했다.

3.3. 공격 데몬 제작 및 결과

3.3에서는 실질적으로 userdata 파티션을 0으로 채우는 (zero-fill) 코드의 제작 과정을 설명한다.

파티션을 0으로 채우기 위해서는 해당 파티션의 읽기/쓰기 권한이 필요한데, vold에는 스마트폰의 eMMC 메모리에 대한 전체 접근 권한은 존재하지 않고 특정 파티션에 대한 읽기/쓰기 권한만이 존재한다. 따라서 userdata 파티션을 먼저 찾아야만 사용자데이터 초기화라는 원하는 기능을 구현할 수 있다. 각 스마트폰 제조사마다 파티션 위치가 다르기에, mount 명령줄의 마운트 포인트인 /data 부분을 파싱하여 파티션의 위치를 알아낸다.

Zero-fill 진행 시, 파티션의 헤더부분을 먼저 지워버리면, OS 에서 파티션 인식 불가로 즉시 커널 패닉이 발생한다. Zero-fill 진행이 완료될 때까지 시스템이 유지되어야 하므로 커널 패닉에 대한 해결책으로 파티션 헤더가 끝난 이후인 첫 40MB 이후부터 zero-fill 을 진행하였다.

공격 바이너리는 vold 대신 실행되며 1회 100MB씩 파티션을 0으로 채우도록 구현하였으며 의도한 기능이 안정적으로 실행됨을 확인하였다.

공격 프로그램으로 제작된 바이너리와 필요 파일을 포함한 애플리케이션 형태로 구현하였으며 해당 애플리케이션을 실행하면 공격 작업이 자동으로 수행되도록 제작하였다. 구현 프로그램은 예상과 같이 userdata 파티션을 0으로 채웠으며, OS는 data 파티션 손상으로 부트 불가 상태에 빠지게 된다.

4. 결 론

본 논문에서는 Dirtycow 취약점을 활용하여 안드로이드 기기를 공격할 수 있는 한 가지 가능성을 보고하였다. Android 보안패치 레벨이 2016-12-01 이전인 경우 안드로이드 2.1부터 7.0에 해당하는 대부분의 시스템에 Dirtycow 공격을 적용할 수 있다.

공격 프로그램의 성능 평가를 위하여 삼성 갤럭시 S7 (Android 6.0.1), S6 (Android 6.0.1), LG G5 (Android 7.0), G4 (Android 6.0.1) 과 같은 보급률이 높은 최신 기종들에서 테스트를 진행하였는데, 상기 기종 모두에서 공격 애플리케이션이 완벽히 동작하였다.

안드로이드의 특성상 애플리케이션의 decompile 후 re-compile이 쉽게 가능하기 때문에, 이와 같은 악성 프로그램을 정상 프로그램과 쉽게 합칠 수 있다. 공식 Google Play 스토어 이외 인터넷에 배포되는 수많은 단독 애플리케이션 파일(.apk) 은 대부분 출처가 불분명한데, 상용 프로그램의 무료 사용 가능성 및 보안에 대한 인식 부족으로 많은 사용자들이 의심 없이 선택하고 있다. 만약, 정상 애플리케이션을 위장한 악성 코드로 본 논문에서 제작한 Dirtycow 기반의 공격을 시행하면 파티션 자체를 0으로 채우는 로우레벨 포맷과 유사한 결과가 발생하므로 데이터 복구가 불가능해진다. 본 연구팀은 차후 리눅스 커널의 취약점을 안드로이드 시스템 공격에 활용할 수 있는 방법을 계속해서 연구하여 스마트폰의 보안 취약점을 개선시킬 수 있도록 노력할 것이다.

감사의 글

이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(과제번호 : NRF-2016R1D1A1B03931615)

References

[1] https://Dirtycow.ninja/ https://github.com/Dirtycow/Dirtycow.github.io/wiki/PoCs

[2] https://source.android.com/security/bulletin/2016-12-01?hl=ko

[3] https://forum.xda-developers.com/v20/development/ls997vs995h910-dirtysanta-bootloader-t3519410

[4] https://github.com/hyln9/VIKIROOT

[5] https://github.com/EnJens/backupTA

[6] https://github.com/matteoserva/Dirtycow-arm32

[7] https://source.android.com/security/verifiedboot/

[8] https://source.android.com/security/selinux/

[9] https://source.android.com/security/encryption/full-disk