WebOS 환경에서의 악성 애플리케이션 자동화 분석1)

박재유*, 차상길**

*한국과학기술원 소프트웨어대학원, **한국과학기술원 정보보호대학원

Automated Analysis of Malicious Applications in WebOS

Jae You Park*, Sang Kil Cha**

*Software Graduate Program, KAIST.
**Graduate School of Information Security, KAIST.

요 약

WebOS는 스마트 가전, IoT 등 다양한 제품에서 운영체제로 널리 사용되고 있다. WebOS에서는, 대부분의 모바일 운영체제가 그러하듯, 단 하나의 악성 앱 만으로도 전체 시스템이 장악될 수 있다. 그럼에도 불구하고 WebOS의 앱 보안에 관한 체계적인 연구는 여전히 미미한 실정이다. 이에 본 연구에서는 WebOS 앱에 대한 정적 및 동적 분석을 수행하여 악성 앱을 자동으로 판단할 수 있는 방법론 및 도구를 제안한다. 이를 위해 우리는 메모리 포렌식 기반의 악성코드 분석 프레임워크를 적용하였으며, 사용자 실험을 통해 제안한 기술이 랜섬웨어 등의 주요 악성코드를 신속하게 탐지하는 데 도움을 준다는 것을 확인하였다. 우리는 본 논문에서 제안하는 도구가 WebOS의 앱스토어 관리 측면에서 높은 업무 효율을 보장할 것으로 기대한다.

I. 서론

사물인터넷 시대의 도래로 생활 곳곳의 모든 장비가 네트워크에 연결되어 온라인으로 동작하고 있다. 대부분의 가전제품에도 모바일 운영체제가 설치되어 있는데, 특히 WebOS는 [1] 스마트 TV나 디지털 사이니지 제품에서 널리 사용되고 있다.

모바일 운영체제의 발전과 더불어 그에 대한 보안 위협은 나날이 증가하고 있다. 특히 PC 환경에서 활개를 치던 악성코드가 이제는 그 영역을 모바일 쪽으로도 확장하고 있는데, 실제로 다양한 스마트폰용 악성 앱이 확산되는가 하면, IoT 장비만을 집중적으로 노리는 Mirai 봇넷이 등장하기도 하였다. 특히 스마트 가전은 사용자의 사생활 속에서 밀접하게 이용되므로 개인정보 노출 등의 피해 발생 가능성이 매우 높다.

1) 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다(UD160066BD).

만약 악의적 해커가 위장된 악성 앱을 개발하여 앱스토어에 등록한다면, 사용자는 이를 무의식적으로 설치하여 막대한 피해를 입을 수 있다. 이를 방지하기 위해 안드로이드 앱의 보안성 검토에 관한 연구가 다수 진행되었으며 [2][3], Apple의 경우에도 앱 감사(auditing)를 통해 불필요하게 권한 상승을 시도하거나, 불법적인 메모리 조작이 발생할 경우 스토어 등록을 거부하는 정책을 채택하고 있다 [4].

다른 모바일 운영체제와는 달리, WebOS 앱에 대한 체계적인 연구는 그 활용도 및 파급효과 대비 여전히 미미한 수준이다. 이에 본 연구에서는 WebOS 앱의 감사과정을 자동화할 수 있는 새로운 방법론 및 도구를 제안한다.

II. 관련 연구

본 장에서는 다른 모바일 플랫폼에서의 악성 앱에 대한 분석도구 동향을 살펴보고, WebOS 앱 분석에 대한 선행연구의 한계점을 살펴본다.

2.1 기존의 모바일 악성 앱 분석도구 동향

2017 BlackHat USA에서는 안드로이드 악성 앱을 감사(audit)하는 CuckooDroid 도구가 공개되었다 [5]. 이 프로젝트는 Windows PC 환경의 악성코드를 분석하는 오픈소스 Cuckoo Sandbox를 [6] 안드로이드 에뮬레이터에서 호환되도록 개조한 것으로, 정적 분석과 동적 분석을 모두 지원한다. 하지만 본 연구와는 달리 메모리 분석은 불가능하다는 단점이 있으며, 안드로이드의 Dalvik API 후킹 기반으로 구현되어 WebOS 환경으로의 직접 적용은 불가능하다.

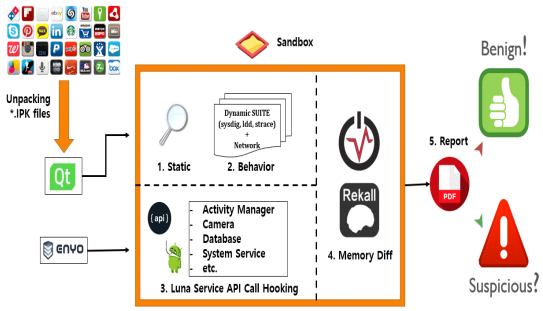
2.2 WebOS 악성 앱에 대한 연구

2017 Hack in Paris에서는 WebOS가 탑재된 스마트 TV를 표적으로 하여, SDK를 통해 악성 앱을 제작하고 이를 앱스토어에 등록함으로써 광범위하게 유통시키는 방식의 공격 시나리오가 제시된 바 있으며, 현재까지도 유효한 공격으로 알려져 있다 [7][8]. 본 연구에서는 실험을 통해 파일 시스템에 접근하여 암호화를 수행하는 랜섬웨어, 네트워크 패킷을 다량으로 송신하는 DoS 공격 등이 실제로 WebOS 위에서 동작 가능함을 직접 확인하였다.

WebOS 관련 다수의 제품을 유통하는 'L 전자'에서는 자사의 WebOS 앱스토어에 등록된 앱에 대한 보안성 검증 방안을 논문으로 발표한 바 있다 [9]. 해당 연구에서는 웹 브라우저 상에서 작동하는 HTML 또는 자바스크립트 코드의 정적 정보를 추출하여 패턴인식 및 머신러닝을 통해 악성 유무를 점검한다. 그러나 이러한 방식은 오직 정적(Static) 정보만을 활용하기 때문에 난독화된 프로그램에 대해서는 동작하기 어려운 한계가 존재한다. 반면, 본 연구에서는 정적 및 동적 분석을 모두 활용하기 때문에 보다 높은 정확도를 기대할 수 있다.

III. 제안 기법

본 논문에서는 최초의 자동화된 WebOS 앱 감사 기법을 제안한다. 제안하는 기법은 정적 분석과 동적 분석을 모두 포함하며, WebOS의 Luna API Hooking, 메모리 포렌식 등을 통합적으로 적용하여 정확도를 높였다.



[그림 1] 제안 기법의 구조

[그림 1]은 제안 기법의 전체적인 흐름도이며, 각 단계별 세부 분석방법을 아래의 절에서 순서대로 각각 소개한다.

3.1 정적 분석

정적 분석 단계에서는 애플리케이션이 배포되는 ipk 형식의 패키지 파일을 대상으로 분석한다. WebOS에서 구동되는 앱은 EnyoJS 프레임워크를 사용하여 제작된 웹 앱과, QT/QML C++을 이용하여 작성하고 컴파일한 ELF 바이너리 형식의 네이티브 앱으로 구분된다. 앱의 형식에 따라 향후 동적 분석 방법이 달라지므로 이 단계에서 언팩을 수행하고 타입을 판단한다. 타입 판별 이후에는 각 타입별로 존재하는 심볼 정보를 추출하고, Yara 룰 기반의 패턴 매칭 및 해쉬비교를 진행한다. 이 단계에서 악성코드로 판별하기 어려운 경우에는 다음의 동적 분석 단계로 넘어가게 되며, 최종 리포트에는 정적 및 동적 분석의 정보가 모두 포함된다.

3.2 동적 행위 분석

동적 분석 과정에서는 실제로 해당 앱을 실행하는 과정이 포함된다. 단, 매번 실제 환경에서 임의의 코드를 실행하면, 악성코드에 의해 테스트베드 자체가 감염될 위험이 있다. 따라서 가상화 솔루션인 VirtualBox를 사용하여 가상머신(WebOS-Ports 기반)을 구동하고 스냅샷 복원 지점을 만든다. 이후 호스트와 VM 사이에 연결을 수립하여 안전한 통신을 맺고(이를

ARES-Tunneling이라 한다), 앱을 VM 내부로 유입시켜 실행한 뒤 시스템콜을 추적함으로써 프로세스 호출이나 네트워크 연결 등 다양한 동적 행위 변화를 기록하는 방식으로 진행한다. 기록된 데이터들은 호스트 시스템으로 전송되어 데이터베이스에 적재된다. 특히 앱 실행 전후의 상태변화를 정상 스냅샷과 비교하여 해당 앱에 의해 어떤 행위가 발생했는지를 선별하여 보여주는 기능을 지원한다.

3.3 Luna API Hooking

일반적으로 WebOS 앱은 하드웨어 등의 하위 계층에 직접적으로 접근할 수 없다. 때문에 프로그래머의 편의를 위해 OS의 여러 가지 서비스와 구성요소를 사용하기 위한 API를 제공하고 있다. WebOS는 'Luna API'를 제공하고 있으며, 앱이 API 메서드 함수를 호출하면 Luna-Bus를 통해 서비스 데몬 프로세스에 메시지를 전달한다. 메시지에는 해당 앱이 요청하고 응답받는 다양한 정보가 포함된다. 이 과정을 관찰하면 어떤 종류의 API가 작동되는지 감지할 수 있다. 앞서 언급한 CuckooDroid [5] 은 Droidmon [10] 을 활용하여 Dalvik API를 관찰하는데, 이에 착안하여 우리는 WebOS의 Luna API 함수를 후킹 하는 방법을 개발하였다. 단, 카메라, 리모컨 등 일부 API의 경우 가상 환경에서 작동하지 않는 한계가 있으므로, 지원 불가한 기능에 대해서는 실제 장비를 사용해야만 정확한 결과를 얻을 수 있다.

3.4 Memory 비교 분석

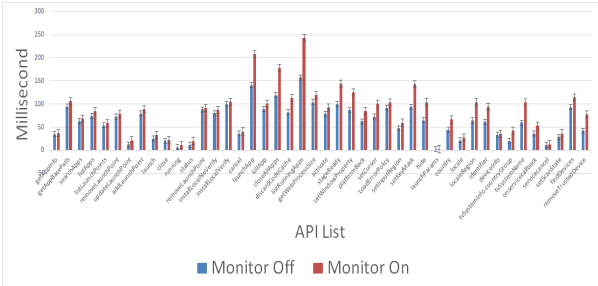
악성코드 분석을 위한 메모리 포렌식은 바이너리 코드에서 알 수 없는 동적 정보나 암호화 비밀키 등을 메모리 분석으로 해결하는 방안이다 [11]. 그러나 메모리 분석 도구인 Volatility [12]는 대부분 Windows 운영체제에 최적화되어 있으므로, WebOS 프로파일은 지원되지 않는다. 본 연구의 대상인 Open WebOS는 리눅스 커널 3.14을 기반으로 하므로, 해당 환경에 부합하는 커널 모듈을 입수하여 직접 빌드한 후 프로파일을 생성했다. 프로파일을 구성요소는 커널 구조가 정의된 vtypes (module.dwarf) 파일과, 디버그 심볼 정보를 담은 System.map 이다. 이를 압축하여 플러그인으로 저장해두면 대상 시스템을 인식하게 된다. 그 결과 Volatility로 WebOS의 메모리 덤프를 분석할 수 있었으며, 악성코드 실행 전후의 메모리 비교 분석을 통해 행위를 파악할 수 있었다.

IV. 적용결과

본 논문에서 제안한 기법을 체계적으로 수행하기 위해 약 1,000 줄의 파이선 프로그램을 제작하였고, 해당 도구의 성능 점검을 위해서 우리는 macOS 환경에서 실험을 진행하였다. 또한 구글 폼을 활용하여 온라인으로 사용성 설문을 진행하였다.

4.1 Overhead 테스트

[그림 2]은 후킹을 실시할 때 전체 44종의 API에서 발생하는 성능 저하의 정도를 요약한다. 파란 막대는 후킹을 사용하지 않은 호출시간이며, 빨간 막대는 후킹 시 지연된 시간이 포함된 값이다. 결과적으로 평균 20ms의 부하가 발생하는 것으로 파악되었으며, 대체적으로 네이티브 앱보다 웹 앱에서 저하가 뚜렷하게 나타났다. 이는 웹앱 관리자(WAM)가 앱 관리 시스템(SAM)을 경유하는 과정에서 지연이 중첩되기 때문인 것으로 보인다.



[그림 2] API Hook Overhead (100회 평균, ms)

하지만 본 기술의 목적은 앱 감사자가 효율적으로 악성 앱을 판별할 수 있도록 도움을 주는 것이며, 앱이 최초 등록될 때에만 사용되므로 20ms의 오버헤드는 큰 문제가 되지 않는다고 판단된다. 또한 다음 절에서 이어지는 사용성 조사를 통해, 우리는 도구의 실익이 어느 정도인지 측정할 수 있었다.

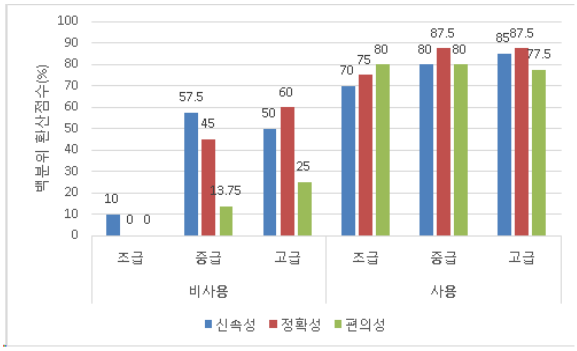
4.2 사용성 설문 연구

본 논문에서 제안한 도구가 실제 유관 업무에 적용되었을 때 기존의 수작업 방식보다 얼마나 효율적인지를 평가하기 위해 필드 연구를 수행하였다. [표 1]은 피험자의 특징이며, 각자의 배경지식, 학력 및 관련 업무 경력을 토대로 세 등급으로 구분하여 참여시켰다.

[표 1] 피험자 개요

전문성	초급	중급	고급
인원수	2	8	4
합계	14		

우리는 이들을 실험군과 대조군으로 나누었는데, 각 그룹에는 등급별 인원이 정확히 절반으로 포함되도록 구성하였다. 실험군에는 3개의 악성(랜섬웨어, 임의 다운로드 시도, 쉘 호출) 및 3개의 정상 앱(메모장, 테트리스, 계산기)과 더불어 우리의 자동화 도구를 제공하였으며, 대조군에는 악성 및 정상 앱 만을 제공하였다. 각자 분석을 수행하도록 한 후 객관식 및 주관식 문항을 제시하여 답변을 수집하였다. 또한, 문제풀이에 소요된 시간을 측정하여 두 집단 간의 신속성을 비교함으로써 도구 사용이 주는 유익을 종합적으로 판단하였다.



[그림 3] 도구의 사용성 설문조사 결과

[그림 3]은 집계된 수치를 백분위로 환산한 결과이며, 이에 따르면 실험군이 대조군보다 평균 78%로 더 신속하게 분석할 수 있었고, 특히 초급 분석가에게 큰 도움이 되었다. 도구 사용 그룹이 시스템에 대해 긍정적으로 인식한 가장 큰 요인은 편의성 측면(10점 만점에 8점)이며, 정확성도 48% 더 우수하게 평가되었다.

V. 결론

본 논문에서는 WebOS 앱 보안감사와 관련하여 메모리 비교분석 기반의 새로운 자동화 분석 기법을 제안하였다. 정적 및 동적 분석을 통합 적용하였으며, API 후킹 기법을 활용한 최초의 WebOS 앱 감사시스템을 구현할 수 있었다. 또한 사용자 연구를 통해 제안된 시스템이 분석가의 업무 효율을 향상하는 것을 확인할 수 있었다. 후속 연구로는 누적된 분석 데이터에 머신러닝 기반 분류 기법을 적용하여 탐지 성능을 향상하는 방안을 제안한다.

[참고문헌]

[1] WebOS-Ports : http://webos-ports.org

[2] Zhang, Yuan, et al. "Vetting undesirable behaviors in android apps with permission use analysis." In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. ACM, 2013

[3] Lu, Long, et al. "CHEX: statically vetting android apps for component hijacking vulnerabilities." In Proceedings of the 2012 ACM Conference on Computer & Communications Security. ACM, 2012

[4] Apple inc. “iOS Security White Paper”, 2017

[5] CuckooDroid : https://goo.gl/7FJ1NQ

[6] Guarnieri, Claudio, et al. "The cuckoo sandbox.", 2012

[7] Lee, JongHo, et al. "Are you watching TV now? Is it real?: Hacking of smart TV", Hack in Paris, 2017

[8] 이종호, 김승주, “Open webOS 기반 스마트 기기에 대한 해킹 시나리오 분석”, 한국정보보호학회 동계 학술발표논문집, 2016

[9] Belenko, Vyacheslav, et al. "Security Verification of the Programs in WebOS app Store.", 2014

[10] Droidmon. : https://github.com/idanr1986/droidmon

[11] Amari, Kristine. "Techniques and tools for recovering and analyzing data from volatile memory." SANS Institute, 2009

[12] Volatility : https://goo.gl/TymJ21