Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.24, NO.6, Dec. 2014 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2014.24.6.1215

안드로이드 로깅 시스템을 이용한 DDoS 공격 애플리케이션 탐지 기법*

최 슬 기,[1†] 홍 민,[2] 곽 진[3‡]
[1]순천향대학교 정보보호학과 정보보호응용및보증연구실, [2]순천향대학교 컴퓨터소프트웨어공학과,
[3]순천향대학교 정보보호학과

DDoS Attack Application Detection Method with Android Logging System*

Seul-Ki Choi,[1†] Min Hong,[2] Jin Kwak[3‡]
[1]ISAA Lab., Department of Information Security Engineering,Soonchunhyang University,
[2]Department of Computer Software Engineering, Soonchunhyang University,
[3]Department of Information Security Engineering, Soonchunhyang University

요 약

현재까지는 스마트폰에 저장된 사용자의 개인정보를 유출시키고, 유출된 개인정보를 악용하기 위한 악성 애플리케이션을 탐지하고, 이러한 악성 애플리케이션으로부터 사용자의 데이터를 보호하기 위한 다양한 연구가 진행되었다. 하지만, 최근에는 스마트폰을 공격 대상이 아닌 DDoS와 같은 2차적인 공격을 수행하기 위한 새로운 공격 도구로 사용하기 위한 악성 애플리케이션이 유포되고 있다. 따라서 본 논문에서는 안드로이드 로깅 시스템을 이용하여 단말기 내부에 설치된 DDoS 공격 애플리케이션을 탐지하는 기법에 대하여 제안한다.

ABSTRACT

Various research was done to protect user's private data from malicious application which expose user's private data and abuse exposed data. However, a new type of malicious application were appeared. And these malicious applications use a smart phone as a new tools to perform secondary attack. Therefore, in this paper, we propose a method to detect the DDoS attack application installed inside the mobile device using the Android logging system.

Keywords: Android, DDoS, Detection, Log analysis

I. 서 론

최근 다양한 기능과 높은 성능을 지닌 스마트폰과 태블릿 같은 모바일 기기가 사용자들에게 널리 보급되면서 일상생활에서의 편리함이 증가하고 있다. 모바일 단말기의 연산 및 처리 능력의 향상과 저장 용량의 증가로 인해 다양한 서비스를 제공하는 애플리케이션들이 개발되고 사용자들에게 널리 전파되었다. 이러한 애플리케이션들이 제공하는 서비스의 유형으로는 소셜 네트워크 서비스와 웹 서버를 이용한 게임과 같이 사용자의 흥미를 유발시키는 것부터 간단한 문서 편집과 사용자의 일정관리와 같이 사용자의 생산성을 증가시켜주기 위한 서비스, 그리고 모바일 뱅킹과 같은 금

접수일(2014년 10월 8일), 수정일(2014년 10월 22일), 게재확정일(2014년 10월 22일)
* 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업[13-912-06-003, 안드로이드 신규 취약점 탐지를 위한 모바일 소프트웨어 보안 테스팅 도구 개발]과 순천향대학교의 지원을 받아 수행된 연구임
† 주저자, skchoi@sch.ac.kr
‡ 교신저자, jkwak@sch.ac.kr(Corresponding author)

융관련 서비스까지 다양한 형태로 존재하고 있다. 반면, 모바일 단말기로 인한 사용자의 편리성이 증가된 만큼, 이를 악용하기 위한 악성 애플리케이션들도 나타나게 되었다. 이러한 악성 애플리케이션들은 최초에는 모바일 단말기의 베터리를 소모시키거나 배경화면을 변경시키는 등의 단순한 장난을 치기 위한 용도였다. 하지만 점차 악성행위의 목적이 금전적인 이득을 취하기 위함 바뀌게 되면서 모바일 단말기에 저장된 사용자의 개인정보를 유출시키고, 유출된 정보를 악용하기 위한 악성 애플리케이션들이 잇따라 출현하게 되었다. 또한, 최근에는 모바일 단말기를 단순한 공격대상으로 삼는 것뿐만 아니라 다른 공격대상을 공격하기 위한 새로운 도구로써 활용하기 위한 기능이 포함된 악성 애플리케이션이 확산되고 있다. 특히, DoS와 DDoS 공격에 대한 탐지 및 방어 솔루션을 제공하는 PROLEXIC 업체에서는 앞으로 다가올 새로운 DDoS 공격의 근원지로 스마트폰과 같은 모바일 단말기를 지목하였으며, 이에 대한 다양한 연구가 필요함을 주장하고 있다[1].

하지만, 모바일 단말기를 이용한 DDoS 공격에 대한 연구가 많이 진행되고 있지 않다. 따라서 우리는 이에 대한 기초 연구로써 모바일 단말기 내에서 설치된 DDoS 애플리케이션을 탐지해내는 기법을 연구하고자 한다. 현재까지, 모바일 단말기 내에 설치된 애플리케이션들 중 악성 애플리케이션을 탐지하기 위한 기법으로는 애플리케이션의 소스코드를 분석하는 정적 분석 기법과 애플리케이션의 행위를 분석하는 동적 분석 기법들을 사용하고 있다[2,3]. 하지만, 이러한 분석 기법들은 악성 애플리케이션에 난독화 기술을 적용함으로써 정적 분석을 방해 혹은 불가능하게 할 수 있으며, 모바일 단말기뿐만 아니라 별도로 분석하기 위한 가상 서버 등과 같이 외부 자원이 필요하기 때문에 모바일 단말기 자체만으로는 단말기 내에 설치된 애플리케이션들 중 악성 애플리케이션을 탐지하기 어렵다는 단점이 존재한다.

한편, 스마트폰 운영체제 중에서 가장 많이 사용되고 있는 안드로이드 운영체제에는 해당 단말기의 하부 시스템부터 애플리케이션으로 인해 발생한 이벤트에 대한 정보를 로그의 형태로 기록하는 안드로이드 로깅 시스템이 기본적으로 탑재되어있다. 이러한 로그 데이터는 커널 상의 로그 버퍼에 기록되기 사용자에 의한 수정이 불가능하기 때문에 로그 데이터에 대한 신뢰도가 높다[4].

따라서 본 논문에서는 안드로이드 단말기에서 발생한 일련의 행위들을 기록하는 안드로이드 로깅 시스템을 이용하여 단말기에 설치된 DDoS 공격 애플리케이션을 탐지하는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 안드로이드 로깅 시스템의 구성 요소와 로그 버퍼을 분석하고, 3장에서는 제안 기법으로 DDoS 공격

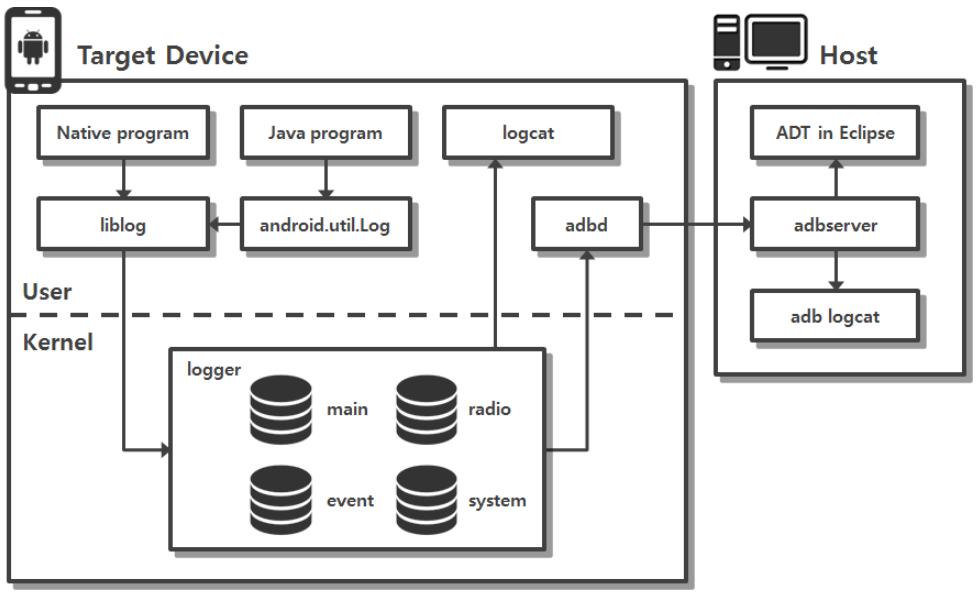


Fig. 1. Overview of Android logging system

애플리케이션에 의해 생성되는 로그들의 특징을 분석하고, 이에 따른 DDoS 공격 애플리케이션 탐지룰을 개발한다. 마지막으로 4장을 결론으로 끝을 맺는다.

II. 관련 연구

2.1 안드로이드 로깅 시스템

안드로이드 로깅 시스템은 시스템 디버그 출력을 수집하고 보고하는 메커니즘을 제공한다. 다양한 애플리케이션과 시스템의 일부분으로부터 수집된 로그들은 안드로이드 시스템의 logcat 명령에 의해서 필터링 및 출력될 수 있다. 본 절에서는 안드로이드 로깅 시스템의 구성 요소, 로그 버퍼 및 로그 포맷과 특징에 대해서 분석한다. Fig.1은 안드로이드 로깅 시스템의 전체적인 구조를 나타내고 있으며, Table 1은 안드로이드 로깅 시스템에 의해 생성된 로그의 예시를 나타내고 있다.

2.1.1 구성 요소

안드로이드 로깅 시스템은 애플리케이션부터 시스템 구성요소까지 시스템의 전반적인 정보를 로깅할 수 있는 기능을 가지고 있다[4]. 안드로이드 로깅 시스템은 아래와 같이 4가지 구성 요소로 이루어진다[4].

- ❒ 로그 메시지 저장을 위한 커널 드라이버와 커널 버퍼
- ❒ 로그 메시지에 대한 접근과 로그 엔트리 생성을 위한 C, C++, Java 클래스
- ❒ 로그 메시지를 확인하기 위한 독립적인 실행 프로그램 (logcat)
- ❒ 호스트 기기로부터 로그를 확인하고 필터링하기 위한 기능

Fig.1과 같이 안드로이드 단말기의 커널 레벨에는 4개의 로그 버퍼를 관리하고 있는 로거라는 커널 드라이브가 존재하고 있으며, 유저 레벨의 애플리케이션은 로그 버퍼에 직접 접근이 불가능하다. 따라서 안드로이드 시스템은 liblog라는 디바이스 노드를 읽고 쓸 수 있는 네이티브 라이브러리를 제공함으로써 애플리케이션이 로그 버퍼에 접근하는 것을 가능하게 한다. 애플리케이션은 C로 작성된 네이티브 애플리케이션과 자바로 개발되어 Dalvik 가상머신 상에서 동작하는 자바 애플리케이션으로 나눌 수 있다. 네이티브 애플리케이션은 liblog를 직접 사용하며, 자바 애플리케이션들은 android.util.log와 같은 클래스들을 통해서 간접적으로 liblog를 사용한다.

로그 버퍼의 추출은 네이티브 프로그램인 logcat을 이용하는 방법과 ADBD(Android Debug Bridge Daemon)를 이용하는 방법이 있다. 네이티브 프로그

Table 1. The Log example generated by Android logging system

04-30 10:50:48.175 D/dalvikvm(5717): GC_CONCURRENT freed 667K, 4% free 17698K/18400K, paused 2ms+1ms, total 16ms
04-30 10:50:48.175 D/dalvikvm(5717): WAIT_FOR_CONCURRENT_GC blocked 4ms
04-30 10:50:48.125 D/dalvikvm(5717): GC_CONCURRENT freed 262K, 3% free 17856K/18296K, paused 3ms+1ms, total 15ms
04-30 10:50:48.115 D/PicasaSyncManager(14418): reject MetadataSyncTask (**********) because auto sync is off
04-30 10:50:48.115 D/dalvikvm(857): GC_FOR_ALLOC freed 6562K, 54% free 25434K/54788K, paused 20ms, total 25ms
04-30 10:50:48.105 D/PicasaSyncManager(14418): reject MetadataSyncTask (**********) because auto sync is off
04-30 10:50:48.095 W/ContextImpl(14468): Implicit intents with startService are not safe: Intent { act=com.nhn.nni.intent.REGISTER flg=0x20 (has extras) }
android.content.ContextWrapper.startService:494 com.nhn.npush.d.a:271
com.nhn.npush.NPushBaseIntentService.a:139
04-30 10:50:48.085 D/dalvikvm(31372): GC_CONCURRENT freed 9299K, 24% free 56274K/73092K, paused 13ms+7ms, total 65ms
04-30 10:50:48.075 V/GCMBaseIntentService(14468): Intent service name: GCMIntentService-DynamicSenderIds-3
04-30 10:50:48.075 D/MessagePoller(1499): try start poller service since android.intent.action.ACTION_POWER_CONNECTED received.

램인 logcat은 안드로이드 단말기 내부에서만 실행이 가능한 방법이다. 로그 버퍼를 호스트로 추출하는 방법은 안드로이드 개발사인 구글에서 제공하는 ADB를 이용하여 호스트에서 안드로이드 단말기로 접속한 후에 logcat을 실행하여 로그 버퍼를 추출할 수 있다.

2.1.2 로그 버퍼

안드로이드 로깅 시스템이 생성하는 로그 메시지는 커널 내에 4개의 로그 버퍼에 저장되며 이들은 각각 시스템의 다른 부분의 로그 메시지를 생성하고 저장한다. 안드로이드 로깅 시스템의 로그 버퍼는 아래와 같이 나뉘어져 있다.

- ❒ main : 메인 애플리케이션의 로그
- ❒ events : 시스템 이벤트 정보를 위한 로그
- ❒ radio : 통신망 접속에 관련된 로그
- ❒ system : 시스템 하위 레벨의 메시지와 디버깅을 위한 로그

2.1.3 로그 수집

안드로이드 로깅 시스템에 의해 생성된 로그들을 수집하기 위해서는 커널 내의 로그 버퍼에 접근하여야 하며, 생성된 로그들은 파일시스템 내의 /dev/log에 위치하고 있다. 하지만 /dev/log에 대한 접근 권한이 설정되어있기 때문에 로그 버퍼에 직접적으로 접근하는 것은 어렵다. 따라서 안드로이드 로깅 시스템의 logcat 명령어를 이용하여 로그 버퍼에 저장된 로그 내용을 수집할 수 있다.

- ❒ main log : logcat -v time -b main -d
- ❒ events log : logcat -v time -b events -d
- ❒ radio log : logcat -v time -b radio -d
- ❒ system log : logcat -v time -b system -d

위 명령어를 이용하여 수집한 로그들을 시간 순으로 정렬하여 하나의 파일에 기록한 뒤 로그들의 특징을 분석한다.

III. 제안 기법

본 논문에서 제안하는 기법은 안드로이드 로깅 시스템을 이용하여 모바일 단말기에 설치되고 동작하는 애플리케이션들 중 DDoS 공격에 사용되고 있는 애플리케이션을 탐지하는 기법이다. 스마트폰 상에서 DDoS 공격을 발생시키는 애플리케이션들이 실행되었을 경우, 안드로이드 로깅 시스템에 의해 수집 및 기록되는 로그들의 특징을 분석하고, 분석된 특징을 바탕으로 DDoS 공격 애플리케이션을 탐지하기 위한 기법을 제안한다.

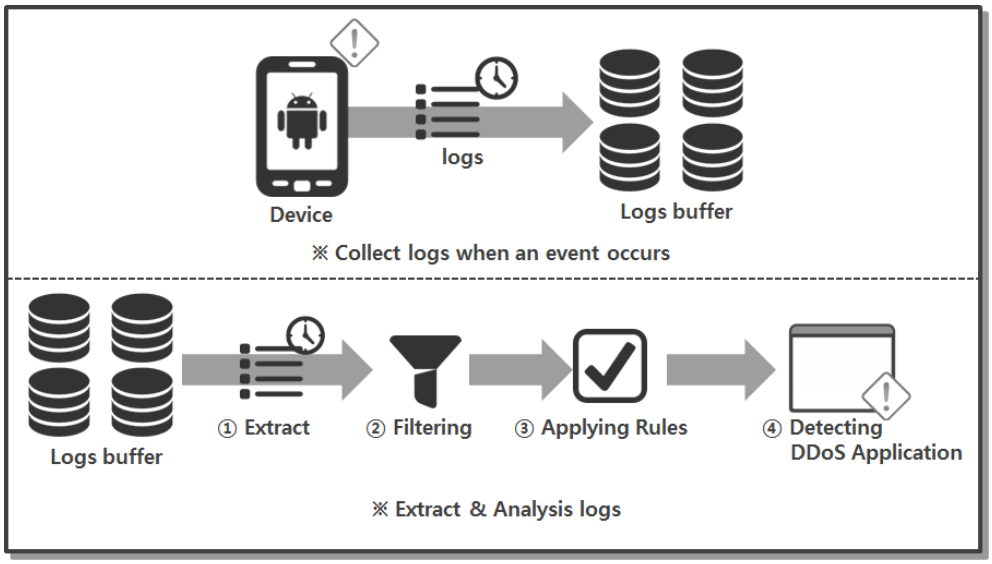


Fig. 2. Overview of proposed method

3.1 DoS 및 DDoS 공격 기능 소스코드 분석

네트워크 스트레스 테스트 도구는 웹 서버 관리자가 서버가 감당해낼 수 있는 네트워크 트래픽 용량을 측정하기 위한 도구이다. 이러한 도구들은 일정한 시간동안 테스트 대상인 웹 서버에 지속적으로 서버 접속에 관련된 네트워크 트래픽을 유발시키는 기능을 가지고 있다. 하지만 네트워크 스트레스 테스트 도구들의 네트워크 트래픽 유발 기능은 웹 서버가 감당해낼 수 있는 네트워크 트래픽 용량을 측정한다는 본래의 목적과는 다르게 테스트 도구의 DoS 기능을 DDoS 공격으로 사용하기 위한 악성 애플리케이션에 탑재되어 구현되어질 수 있다.

따라서 본 절에서는 네트워크 스트레스 테스트 도구에 구현된 DoS 공격 기능과 실제로 DDoS 공격을 수행하기 위한 목적으로 구현된 악성 애플리케이션의 DDoS 공격 기능을 분석하여, 네트워크 스트레스 테스트 도구와 DDoS 공격 애플리케이션이 동일한 메커니즘으로 DDoS 공격을 수행하는지 분석한다.

Table 2는 네트워크 스트레스 도구의 DoS 기능을 수행하는 핵심 소스코드를 나타내고 있으며, 이어서 Table 3은 DDoS 공격을 수행하는 악성 애플리케이션의 핵심 소스코드를 나타내고 있다.

DDoS 공격을 수행하는 악성 애플리케이션의 핵심 소스코드를 네트워크 스트레스 테스트 도구와 비교 분석한 결과, 네트워크 스트레스 테스트 도구의 UDP flooding 공격 기능과 동일한 메커니즘으로 구성되어 동작한다는 것을 알 수 있다.

3.2 DDoS 공격 애플리케이션의 로그 특징 분석

본 절에서는 DDoS 공격 애플리케이션이 DDoS 공격을 수행하게 될 때, 안드로이드 로깅 시스템에

Table 2. Source code for DoS attack in network stress test tools

```
// TCP flooding attack source code
while(true) {
.....(skip).....
   try {
      Socket localSocket = new Socket();
      localSocket.connect(new InetSocketAddress(this.ip, this.port), this.timeout);
      OutputStreamWriter localOutputStreamWriter = new OutputStreamWriter(localSocket.getOutputStream());
      localOutputStreamWriter.write(this.message);

// UDP flooding attack source code
while(true) {
.....(skip).....
   try {
      DatagramSocket localDatagramSocket = new DatagramSocket();
      InetAddress localInetAddress = InetAddress.getByName(this.ip);
      new byte[1024];
      byte[] arrayOfByte = this.message.getBytes();
      localDatagramSocket.send(new DatagramPacket(arrayOfByte, arrayOfByte.length, localInetAddress, this.port));
```

Table 3. Source code for DDoS attack in malicious application

```
// UDP flooding attack source code
System.out.println("ATTASCK " + this.a + ":" + this.b + "(" + this.d + "/" + this.c + ")");
   while (true) {
      DatagramSocket localDatagramSocket;
      DatagramPacket localDatagramPacket;
.....(skip).....
      localDatagramPacket = new DatagramPacket(arrayOfByte, arrayOfByte.length, InetAddress.getByName(this.a), this.b);
.....(skip).....
   localDatagramSocket.send(localDatagramPacket);
```

의해서 기록되는 로그들의 특징을 분석한다.

DDoS 공격을 발생시키는 악성 애플리케이션은 일정 시간 동안에 3.1절에서 분석한 결과와 같이 네트워크 스트레스 테스트 도구에 구현된 TCP 및 UDP flooding 공격 소스코드와 같은 네트워크 트래픽을 발생시키는 작업을 반복적으로 수행하게 된다. 또한 DDoS 공격을 효과적으로 발생시키기 위해서 모바일 단말기 사용자가 단말기의 성능 저하를 인지하지 못하는 수준에서 최대한으로 네트워크 트래픽을 발생하도록 구현을 한다. 이로 인해 DDoS 공격을 유발시키는 기능이 구현된 애플리케이션은 네트워크 트래픽 관련 오브젝트가 다수 생성된다. 이처럼 애플리케이션 내에 오브젝트들이 지속적으로 생성되면 애플리케이션이 할당 받은 힙 메모리 영역이 서서히 초과상태에 이르게 되는데 시스템 내부에서는 애플리케이션의 메모리 확보를 위해서 쓰레기 수집(Garbage Collection)이라는 작업을 수행하게 된다. 즉 DDoS 공격 애플리케이션이 실행되어 네트워크 트래픽을 증가시키는 과정을 통해 많은 양의 오브젝트가 지속적으로 생성될수록 시스템 내부에서 쓰레기 수집 작업 또한 지속적으로 발생한다.

따라서 DDoS 공격 기능이 탑재된 애플리케이션이 실행될 경우 안드로이드 로깅 시스템에 의해 쓰레기 수집에 관련된 로그들이 일정 시간 동안에 많이 기록되는 것을 확인할 수 있다.

Table 4는 악성 애플리케이션이 DDoS 공격을 수행하였을 때 안드로이드 로깅 시스템에 수집 및 기록된 쓰레기 수집 관련 로그들이다.

Table 4에서 나타난 로그들의 Text 영역을 분석하면 GC_CONCURRENT 형식의 기록이 짧은 시간 동안에 주기적으로 남는 것을 확인할 수 있다. GC_CONCURRENT 메시지는 애플리케이션의 메모리 힙 영역이 거의 가득 차게 되어, 쓰레기 수집 작업을 수행하는 GC(Garbage Collection)가 수행 되었을 때 남기게 되는 디버그 클래스의 로그이다. GC_CONCURRENT 메시지의 형식은 아래 Table 5와 같다.

DDoS 공격 애플리케이션과 정상적인 애플리케이션이 남기는 쓰레기 수집 관련 로그들의 양을 비교 분석하기 위해 실제 모바일 단말기에 여러 종류의 정상 애플리케이션을 설치하고 장시간 사용하면서 지속적으로 로그를 수집 및 기록하여 분석하였다. 로그 수집의 환경은 아래 Table 6과 같다.

Table 4. The generated logs when performing DDoS attacks

```
04-30 06:28:43.944 D/dalvikvm( 1670): GC_CONCURRENT freed 318K, 2% free 18154K/18508K, paused
1ms+2ms, total 11ms
04-30 06:28:44.224 D/dalvikvm( 1670): GC_CONCURRENT freed 429K, 4% free 18117K/18688K, paused
2ms+2ms, total 19ms
04-30 06:28:44.504 D/dalvikvm( 1670): GC_CONCURRENT freed 406K, 3% free 18151K/18688K, paused
3ms+2ms, total 17ms
04-30 06:28:44.764 D/dalvikvm( 1670): GC_CONCURRENT freed 369K, 3% free 18193K/18688K, paused
3ms+5ms, total 23ms
.....(skip).....
04-30 06:28:14.544 D/dalvikvm( 1544): GC_CONCURRENT freed 9K, 2% free 20407K/20672K, paused
2ms+1ms, total 12ms
04-30 06:28:14.604 I/System.out( 1544): Creating connection
04-30 06:28:14.674 I/System.out( 1544): Creating connection
04-30 06:28:14.724 I/System.out( 1544): Creating connection
.....(skip).....
04-30 06:28:16.244 D/dalvikvm( 1544): GC_CONCURRENT freed 316K, 2% free 21208K/21556K, paused
1ms+2ms, total 22ms
04-30 06:28:16.284 I/System.out( 1544): Creating connection
04-30 06:28:16.344 I/System.out( 1544): Creating connection
.....(skip).....
04-30 06:28:18.554 D/dalvikvm( 1544): GC_CONCURRENT freed 436K, 3% free 22144K/22612K, paused
3ms+5ms, total 37ms
04-30 06:28:18.574 I/System.out( 1544): Creating connection
04-30 06:28:18.634 I/System.out( 1544): Creating connection
```

Table 5. Format of GC_CONCURRENT

Text	Description
freed 318K	Amount of memory freed
2% free	Memory after freeing %
18154K/18508K	Actual memory in heap
paused 1ms+2ms	Time taken for GC
total 11ms	Total paused time for memory freeing

Table 6. The environment of logs collecting

Device Versions		Hours of use
Android Versions	Kernel Versions	
4.4.2	3.4.0	1 hours
4.4.2	3.1.10	
4.1.2	3.0.31	

실험 환경의 분석 대상 단말기로부터 수집한 로그들 중 GC_CONCURRENT 관련 로그의 비율을 분석한 결과, 정상적으로 동작하는 애플리케이션만 약 1시간가량 사용했을 경우에는 메모리 부족현상으로 인해 시스템이 GC를 호출하여 메모리를 해제하는 작업에 관련된 로그가 전체 로그 중에서 약 7.59% 차지하고 있다는 결과를 얻어낼 수 있었다. 이어서, 1시간가량 정상 애플리케이션들을 사용한 이후에 DDoS 공격 애플리케이션을 약 1분간 실행하고 난 뒤 GC_CONCURRENT 관련 로그 비율의 변화를 살펴본 결과, 메모리 해제 관련 로그의 비율이 1분 사이에 약 12.74%로 크게 증가한 것을 확인할 수 있었다.

Table 7은 DDoS 공격 애플리케이션의 실행 전 및 실행 후에 안드로이드 로깅 시스템에 기록된 로그 중 GC_CONCURRENT 관련된 로그의 비율을 나타낸 표이다.

즉, 악성 애플리케이션의 DDoS 공격이 실행되면 안드로이드 단말기 내 메모리 해제작업이 급격하게 증가하게 되고 이로 인해, 쓰레기 수집 작업을 수행하는 GC에 의해 GC_CONCURRENT 관련 로그가 짧은 시간동안에 다수 발생한다는 결과를 얻을 수 있었다.

Table 7. The ratio of memory free messages

Case	Run time	Ratio
Running only normal applications	1 hour	7.59%
After running DDoS application	1 minute	12.74%

물론, DDoS 공격 기능이 탑재되지 않은 정상 애플리케이션에 의해서 메모리 해제 작업이 발생하여 GC_CONCURRENT 관련 로그가 발생할 수 있다. 하지만 이러한 경우에는 일시적으로 발생하는 현상일 뿐, DDoS 공격 애플리케이션과 같이 지속적으로 발생하지 않는다는 것을 실험 결과를 통해 알아낼 수 있었다.

3.3 DDoS 공격 애플리케이션 탐지 룰

DDoS 공격 애플리케이션은 오브젝트를 자주 생성하여 GC 호출로 인한 GC_CONCURRENT 관련 로그가 지속적으로 쌓인다는 특징을 갖고 있다. 따라서 안드로이드 로깅 시스템에 의해 기록되는 수많은 로그들 중 GC_CONCURRENT 관련 로그들을 수집하고 지속적으로 발생되는지 여부를 판단하는 룰을 정의함으로써 안드로이드 단말기 내에서 DDoS 공격을 수행하는 악성 애플리케이션을 탐지한다.

본 논문에서의 DDoS 애플리케이션 탐지룰은 GC_CONCURRENT 로그 메시지를 최대 10초 간격으로 지속적으로 생성하면서, 그 상태를 1분간 유지하고 있는 PID를 DDoS 애플리케이션으로 탐지한다.

Table 8. Pseudo code for DDoS attack application detection

```
01  Select GC_CONCURRENT logs
02
03  List ← Time list in analysis PID's log
04
05  Start_Time ← List[0]
06  i ← 0
07
08  While(until List is empty) :
09
10     if(List[i+1] - List[i] 〈 10sec):
11        if(List[i+1] - Start_Time 〉 1min):
12           Alert PID to user
13        else:
14           i ← i+1
15     else:
16        Start_Time ← List[i]
```

다음 Table 8은 GC_CONCURRENT 관련 로그 메시지의 지속시간을 이용하여 DDoS 공격을 발생시키는 애플리케이션을 탐지하는 룰의 의사코드이다.

List에는 분석 대상 PID의 로그에 삽입된 시간들을 담고 있는 리스트이다. Start_Time은 GC관련 로그 메시지 지속상태를 측정하기 위한 기준을 기준 역할을 수행하기 위한 변수이다.

위 의사코드의 12번째 줄 "Alert PID to user" 코드는 GC 로그 메시지가 10초 간격으로 1분 이상 지속된 경우 수행되는 코드이며, 사용자에게 해당 PID를 가진 애플리케이션이 DDoS 발생 애플리케이션임을 알리는 역할을 수행한다.

IV. 효율성 분석

안드로이드 로깅 시스템은 안드로이드 단말기의 시스템 내부에서 발생하는 이벤트부터 애플리케이션의 행위 및 디버그 관련 메시지까지 다양한 정보들을 로그의 형태로 기록하고 관리하는 시스템이다. 따라서 안드로이드 단말기에서 발생하는 행동을 분석하는 동적 분석을 직접 실시하지 않더라도, 안드로이드 로깅 시스템에 기록된 로그들을 수집 및 분석함으로써 단말기에서 발생한 다양한 행위들을 분석할 수 있다. 또한, 안드로이드 로깅 시스템은 커널 레벨에서 동작하기 때문에 안드로이드 로깅 시스템에 의해 기록된 로그들은 사용자 레벨에서 위조 및 변조를 할 수 없기 때문에 로그에 대한 신뢰도가 높다고 할 수 있다.

따라서 본 논문에서 제안한 기법은 안드로이드 단말기의 행위에 대한 정보를 담고 있으며, 그 정보에 대한 신뢰도가 높은 안드로이드 로그를 분석함으로써 직접적으로 동적 및 정적 분석을 수행하지 않더라도 안드로이드 단말기 내에서 발생하는 DDoS 공격을 탐지할 수 있으며, DDoS 공격을 실질적으로 수행하고 있는 악성 애플리케이션까지 탐지해 낼 수 있다는 특징을 갖고 있다.

또한 안드로이드 로깅 시스템은 안드로이드 단말기에 기본적으로 내장된 시스템이기 때문에 DDoS 공격 애플리케이션을 탐지하기 위해 안드로이드 단말기 내부에 별도의 애플리케이션을 설치하거나, 단말기를 수정하는 행위들이 불필요하다. 따라서 안드로이드 로깅 시스템으로부터 로그들을 추출하여 DDoS 탐지룰만 적용함으로써 DDoS 공격 애플리케이션을 탐지할 수 있기 때문에 다양한 사용자들에게 쉽게 이용될 수 있을 것이라고 판단된다.

V. 결 론

스마트폰과 같은 모바일 단말기를 공격 대상으로 삼는 악성 애플리케이션들은 점차 다양해지고 있으며 그 행위들 또한 점차 발전되고 있다. 초기에는 모바일 단말기의 배터리를 소모시키는 등의 단순한 악성 행위부터 모바일 단말기 내에 저장된 사용자의 개인정보를 수집하고, 수집된 정보를 이용하여 금전적인 피해를 입히는 등의 행위까지 이르렀다. 또한, 최근에는 스마트폰을 단순한 공격 대상으로 삼는 것뿐만 아니라 2차 공격 대상을 공격하기 위한 새로운 수단으로 활용하기 위한 악성 애플리케이션도 발생하게 되었다. 특히, 스마트폰과 같은 모바일 단말기를 DDoS 공격의 도구로 활용할 수 있다는 가능성이 높아지고 있으며, 이에 대한 연구와 대응이 필요해졌다. 따라서 본 논문에서는 다양한 스마트폰 운영체제 중 높은 점유율을 차지하고 있는 안드로이드 운영체제를 탑재한 단말기에서 발생하는 DDoS 공격을 탐지하기 위한 기법을 제안하였다.

본 논문의 제안기법의 안드로이드 단말기의 행위를 분석해낼 수 있는 정보를 담고 있는 안드로이드 로깅 시스템을 이용하여 안드로이드 단말기 내에 설치된 DDoS 공격 애플리케이션을 탐지할 수 있는 기법을 제안하였으며, 그에 대한 효율성을 분석하였다. 하지만 현재 DDoS 공격 기능을 탑재한 악성 애플리케이션이 많이 발견되고 있지 않기 때문에 제안기법의 한계점이 존재할 것으로 판단된다. 하지만 본 논문이 모바일 환경에서의 DDoS 공격을 탐지하기 위한 연구의 초석이 될 것으로 기대한다.

References

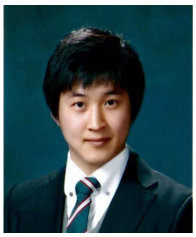
[1] PROLEXIC, "Prolexic quarterly global ddoS attack report Q1 2014," Apr. 2014.

[2] Seung-hwan Ju, Hee-suk Seo and Jin Kwak, "Study on analysis methodology for android applications," Journal of Internet Technology, vol. 14, no. 5, pp. 851-857, Sep. 2013.

[3] Seung-hwan Ju, Hee-suk Seo and Jin Kwak, "Study on analysis application smartwork," Lecture Notes in Electrical Engineering, vol. 214, pp. 31-37, Dec. 2012.

〔4〕 Ilyoung Hong and Sangjin Lee, "Research on efficient live evidence analysis system based on user activity using android logging system," Journal of The Korea Institute of information Security & Cryptology, vol. 22, no. 1, pp. 67-80, Feb. 2012.
〔5〕 W. Enck, D. Octeau, P. McDaniel and S. Chauduri, "A study of android security," In Proc. of the 20th USENIX Security Symposium, Aug. 2011.
〔6〕 Jun-Sub Kim, Jae-Byoung Yoon, Kyung-Gi Kim, Min-Soo Kim, Jin Kwak and Bong Gyou Lee, "Vulnerability analysis method on android application," Proceeding of CISC-W'12, pp. 161-164, Dec. 2012.
〔7〕 Yujong Jang and Jin Kwak, "Android application verification method using security testbed," Proceeding of CISC-W'12, pp. 161-164, Dec. 2012.
〔8〕 Google, "Android logging system," http://developer.android.com/tools/debugging/debugging-log.html
〔9〕 Google, "Android debug bridge," http://developer.android.com/tools/help/adb.html
〔10〕 Google, "Application fundamentals," http://developer.android.com/guide/components/fundamentals.html

〈저자소개〉



최 슬 기 (Seul-Ki Choi) 학생회원
2013년 2월: 순천향대학교 정보보호학과(공학사)
2013년 2월~현재: 순천향대학교 정보보호학과 석사과정
〈관심분야〉 스마트폰 보안, 리눅스 시스템 보안, 응용시스템보안, 디지털 포렌식

홍 민 (Min Hong) 종신회원
1995년 2월: 순천향대학교 학사
2001년 5월: University of Colorado at Boulder 석사
2001년 8월~2001년: Teaching Assistant, University of Colorado at Denver and Health Science center
2002년 8월~2005년: Research Assistant, University of Colorado at Denver and Health Science center
2006년 3월~현재: 순천향대학교 컴퓨터소프트웨어공학과 교수
2008년 1월~현재: 한국 지식정보기술 학회(이사)
2008년 3월~ 현재: 한국 인터넷 정보학회 기획이사
2009년 1월~현재: 한국 멀티미디어학회 논문지 편집위원
2010년 3월~2011년: 컴퓨터 소프트웨어 공학과 학과장, 순천향대학교
2011년 2월~현재: 한국정보처리 학회 (상임 이사(국제))
2013년 2월: Visiting Scholar, Computer Graphics Lab, University of Colorado Denver
2014년 3월~현재: 컴퓨터 소프트웨어 공학과 학과장, 순천향대학교
〈관심분야〉 컴퓨터그래픽스, 컴퓨터게임, 바이오 인포메틱스, u-Healthcare 시스템, 이미지 프로세싱

곽 진 (Jin Kwak) 종신회원
2000년 8월: 성균관대학교 학사
2003년 2월: 성균관대학교 석사
2006년 2월: 성균관대학교 박사
2006년 4월~2006년 11월: 일본 큐슈대학교 방문연구원
2006년 4월~2006년 11월: 일본 큐슈시스템정보기술연구소 특별연구원
2006년~2007년 2월: 정보통신부 정보보호기획단 개인정보보호팀 통신사무관
2007년 3월~현재: 순천향대학교 정보보호학과 교수
2009년 1월~2009년 12월: 순천향대학교 공과대학 교학부장
2009년 1월~2010년 12월: 순천향대학교 정보보호학과 학과장
2010년 1월~2012년 12월: 순천향대학교 SCH BIT 창업보육센터장
2011년 2월~2012년 12월: 순천향대학교 중소기업산학협력센터 센터장
2007년 1월~2009년 12월: 정보통신산업진흥원 주간기술동향 집필위원
2008년 1월~2010년 12월: 한국정보보호학회 논문지편집위원
2008년 1월~현재: 한국정보보호학회 이사
2008년 4월~현재: 한국인터넷정보학회 논문지편집위원
2008년 12월~현재: 정보통신산업진흥원 기술평가위원
2009년 5월~현재: TTA 표준화로드맵 기술표준기획전담반 위원
2010년 3월~현재: 조달청 기술평가위원
2010년 5월~2010년 7월: 교육과학기술부 국가기술수준평가 위원
2011년 1월~현재: 한국정보기술융합학회 이사
2011년 1월~현재: 한국정보처리학회 이사
2011년 1월~현재: JIPS 논문지 편집위원
2011년 1월~현재: 지식경제부 지식경제기술혁신평가단 위원
2012년 ~현재: 한국암호포럼 운영위원
2012년 ~현재: 한국방송통신전파진흥원 평가위원
2013년 ~현재: 교육부 정책자문위원
2013년 ~현재: 금융보안연구원 보안기술 자문위원
2013년 ~현재: 국가정보원 암호검증위원
2013년 ~현재: 금융감독원 인증방법평가위원
〈관심분야〉 자동차 보안, 암호프로토콜, 응용시스템보안, 개인정보보호, 정보보호제품평가, 클라우드 컴퓨팅 보안