Regular paper http://dx.doi.org/10.13067/JKIECS.2014.9.9.1011

웹 취약점 분석을 위한 프락시 시스템의 설계 및 구현

김광현*

Implementation and Design of Proxy System for Web vulnerability Analysis

Gwang-Hyun Kim*

요 약

웹 사이트를 통한 정보제공이 활성화 되면서 웹 애플리케이션의 취약점을 이용한 웹 해킹 시도가 증가하고 있다. 웹 애플리케이션의 보안을 강화하려면 먼저 웹 애플리케이션의 취약점을 찾아 제거할 필요가 있다. 본 논문은 웹 애플리케이션에 대한 기존의 취약점 해결 방법을 분석하고 보다 발전된 취약점 해결방안을 제시하고자 한다. 웹 애플리케이션 취약점 분석을 통해 현존하는 웹 취약점을 제거한 웹 보안 상태의 안정성을 점검하고 기존 방법의 적합성을 평가하였다. 또한 기존 취약점 해결방안의 미비점을 보완한 방법으로 웹 프락시(Proxy) 시스템을 통한 취약점 분석 툴을 구현하고 최적화 방안을 제시하였다.

ABSTRACT

Because of the proliferation of web services through web site, web hacking attempts are increasing using vulnerabilities of the web application. In order to improve the security of web applications, we have to find vulnerabilities in web applications and then have to remove. This paper addresses a vulnerability in a web application on existing problems and analyze and propose solutions to the vulnerability. This paper have checked the stability of existing web security solutions and evaluated its suitability through analysis of vulnerability. Also, we have implemented the vulnerability analysis tools for web Proxy system and proposed methods to optimize for resolution of web vulnerabilities.

키워드

Web Hacking, Vulnerability, Web Proxy, Web Application
웹해킹, 취약점, 웹 프락시, 웹 응용

Ⅰ. 서 론

인터넷의 비약적인 발전과 동시에 월드와이드웹(World Wide Web)이 활성화되기 시작하면서 부터 웹 사이트에 대한 해킹 시도가 급증하고 있다. 특히 2000년 이후 발견된 웹 애플리케이션의 취약점 누적 개수가 그 이전에 비해 기하급수적으로 증가하고 있다. 이는 웹을 통한 해킹 가능성이 점점 높아지고 있다는 의미이다. 과거에는 네트워크나 시스템을 통한 해킹시도가 주류를 이루었으나 최근 들어 점점 웹을 통한 해킹시도가 많아지고 있다[1]. IBM X-Force Trend and Risk Report에 따르면 2009년 기준으로 전체 취약점 가운데 웹 취약점이 차지하는 비중이 절반가량에 이른다고 한다[2]. 국내 실정도 이와 크게 다르지 않아서 한국인터넷진흥원 인터넷침해대응센터의 침해사고 통계에 따르면 최근 5년 사이에 웹을 통

* 교신저자(corresponding author) : 광주대학교 정보통신학과(ghkim@gwangju.ac.kr)
접수일자 : 2014. 07. 07 심사(수정)일자 : 2014. 08. 21 게재확정일자 : 2014. 09. 19

한 침해사고 발생 건수가 그 이전에 비해 비약적으로 증가하고 있다[3]. 본 논문의 연구 목적은 지속적으로 증가하고 있는 웹을 통한 해킹 사고를 방지하기 위해 웹 애플리케이션의 취약점을 찾아내고 분석하여 제거하는데 있다.

웹 취약점을 제거하기 위해서는 먼저 웹에 존재하는 취약점 유형, 취약점을 이용한 해킹 방법, 취약점을 해결하기 위한 보안 방법 등 웹 해킹에 대한 전반적인 이해가 필요하다. 또한 취약점을 쉽게 찾아낼 수 있는 방법을 알아야 한다. 이에 본 논문은 웹에 존재하는 갖가지 취약점에 대한 개념을 이해하고, 취약점을 쉽게 찾아내기 위한 기술적 방법으로 웹 프락시(Proxy)를 구현 및 최적화하여 웹 취약점을 해결하기 위한 방안으로 제시하고자 한다. 먼저 웹에 존재하는 취약점에 대한 개념을 정리하고 이를 해결하기 위한 기존의 해결방안 및 대응 도구에 대해 알아본다. 다음으로 기존의 취약점 대응 방법을 연구하여 기존 해결방안의 부족함을 보완할 수 있는 발전된 취약점 대응 방안을 연구한다. 마지막으로 웹 프락시 시스템을 구현하고 최적화 하여 웹 취약점을 쉽게 찾아 낼 수 있는 기술적 방안을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 주로 발생하는 웹 취약점 유형을 설명함으로서 웹 애플리케이션의 보안 유형에 대해 알아보고 기존의 취약점 진단방법 및 대응방안에 대해 알아본다. 3장에서는 2장에서 제시한 기존의 대응 방안을 기반으로 보다 발전된 취약점 대응방안에 대해 연구한다. 4장에서는 본 논문의 결론을 제시한다.

II. 웹 애플리케이션의 보안과 웹 취약점 분석

최근 들어 기존의 C/S 프로그램들이 많은 부분 웹으로 이동하여 웹 애플리케이션이 점점 복잡해짐에 따라 그만큼 보안문제가 확대되고 있다. 특히 과거에는 해킹이 서버나 네트워크 차원에서 이루어지던 것이 웹 쪽으로 점점 이동하고 있는 추세다. 이는 서버나 네트워크 같은 기존의 전통적인 보안문제가 오랜 기간 동안 많은 부분 해결되었기 때문이라고 볼 수 있다. 더불어 웹 애플리케이션의 수가 기하급수적으로 늘어나 보안 이슈가 웹 쪽으로 이동했다고도 볼 수 있다[4]. 한마디로 해커들이 뚫기 어려운 서버나 네트워크 보다는 상대적으로 취약한 웹 쪽으로 몰린다는 의미로 보면 되겠다.

과거 정적인 페이지를 통하여 단순한 정보를 제공하던 시절에는 웹 서버가 해킹에 노출되어도 큰 피해가 없었지만 지금은 사정이 다르다. 웹상에서 수많은 정보들이 가공되어 처리되므로 일련의 중요하고도 의미 있는 정보들 혹은 대단히 사적인 정보들이 유출될 염려가 있다. 최근 들어 개인정보유출이나 금융권 해킹 문제가 사회적 이슈로 자주 등장하는 것도 이와 무관하지 않다[5].

2.1. 웹 프락시의 개념

웹 서버가 클라이언트로부터 어떤 인터넷 페이지에 대한 요청을 받으면 일단 그 주소를 이전에 방문한 적이 있는지 검사한다. 만약 방문한 적이 있다면 지난번에 내용을 저장한 캐시 디렉토리에서 그 정보를 보여주고 캐시 디렉토리에 해당 내용이 없거나 처음 요청하는 페이지라면 URL의 서버로부터 다시 가지고와서 보여준다. 프락시는 캐시 디렉토리와 비슷한 역할을 한다고 보면 된다. 다만 임시 저장 공간이 디렉토리가 아니라 서버가 된다는 사실만 다를 뿐이다[6].

이렇게 프락시 서버의 기능에는 캐시 기능이 있다. 캐시를 사용하면 네트워크의 트래픽을 줄이고, 데이터의 전송 시간을 향상 시킨다. 실제 전송되는 콘텐츠가 프락시 서버를 통해서 이동하므로 내용을 필터링하여 웹 방화벽 기능을 담당하기도 한다. 인터넷 동시 접속자가 많을 때, 음란사이트 등 유해 사이트를 차단할 때, 내부 사용자 IP주소를 사설 IP주소로 설정하여 보안을 강화할 때, 해커 등 외부의 침입을 방지하고자 할 때 유용하다. 따라서 인터넷을 사용할 때 보안이나 규제가 필요한 기업이나 학교 등에서 많이 도입하고 있다.

2.2. 쿠키정보와 취약점 분석

HTTP는 웹 서버와 클라이언트 간의 연결을 지속적으로 유지하지 않고 페이지를 로딩하고 난 후 접속을 끊게 되는데 이 때 서버의 클라이언트간의 접속을

유지하거나 사용자 편의를 제공하기 위해 쿠키를 활용한다. 예를 들어 어떤 사용자가 아이디와 패스워드를 입력하여 특정 페이지에 로그인 한 경우 이 페이지가 전부 로딩되면 서버와의 통신이 중단되고 사용자가 새로고침 하거나 특정 링크를 클릭한 경우 다시 서버와의 통신을 재개하게 되는데, 웹 서버 입장에서 보면 이 사용자가 이전 로그인 한 사용자인지 아닌지 구분해야 할 필요가 있게 된다. 이 때 웹 서버는 클라이언트 PC에 저장된 사용자의 쿠키정보를 확인하여 로그인 된 사용자인지 아닌지를 구별한다[6]. 쿠키는 이렇게 사용자의 PC에 저장되기 때문에 얼마든지 조작이 가능하다. 따라서 최근에는 로그인 시 쿠키만 사용하는 경우는 드물고 세션을 같이 사용하여 서버에도 별도의 세션 정보를 저장하여 쿠키 조작에 대한 위험을 방지하고 있다.

클라이언트에 저장할 수 있는 쿠키의 개수는 최대 300개이며 쿠키 하나의 최대 크기는 4KByte를 넘을 수 없다. 또한 한 개의 서버나 도메인 당 최고 20개의 쿠키를 보낼 수 있다. 쿠키 파일은 웹 서핑을 하는 동안 수도 없이 생겨서 사용자의 PC에 쌓이게 된다.

HTTP 프로토콜은 기본적으로 암호화되지 않는 평문을 사용하기 때문에 패킷 스니핑 툴(WireShark, Expert 등)을 사용하여 얼마든지 열어 볼 수 있고 헤더를 조작하는 방식으로 쿠키를 변경할 수 있다. 웹 프락시(Proxy)를 사용해서 헤더를 조작하거나 Cooxie Toolbar나 FireCookie 등을 사용하여 보다 쉽게 쿠키를 조작할 수 있다.

2.3. 크로스 사이트 스크립팅 취약점 분석

크로스 사이트 스크립팅(XSS : cross-site scripting)은 웹 서비스 상의 취약점 중 하나로, 권한이 없는 사용자가 특정 웹 페이지에 클라이언트 사이드 스크립트를 삽입하여 다른 사용자가 이를 실행하게끔 허용하는 취약점을 말한다[6-7]. 일반적으로 여러 사용자가 공유하여 이용하는 게시판이 XSS 공격 대상이 되기 쉽다.

이 취약점은 사용자로부터 입력 받은 값을 웹 애플리케이션이 검사하지 않고 그대로 사용할 경우 나타난다. 주로 사용자의 쿠키나 세션 정보를 탈취하기 위하여 사용되며, 특수 문자나 예약어, 스크립트를 나타내는 ', " , > , < , % , $ 등의 문자를 이용한다. 공격 대상 웹사이트에 삽입한 스크립트를 이용하여 다른 웹사이트로 접근하는 것도 가능하기 때문에 크로스 사이트 스크립팅이라고 한다.

XSS는 < > 과 같은 태그를 허용하면서 문제가 발생하게 되는 것이다. 예를 들어 태그가 허용된 게시판에 <script>alert('docuement.cookie')</script> 와 같이 작성하면 해당 게시물을 확인할 때 쿠키 정보가 메시지 창으로 뜬다. 이런 원리를 통해 여러 가지 공격이 가능하게 된다. 단순하게 로그인한 사용자의 IP 정보를 수집하거나 쿠키정보를 확인 할 수 있으며 더 나아가 악성코드를 이용하여 사용자의 개인정보를 수집하거나 사용자 컴퓨터를 무력화 할 수도 있다.

XSS 문제를 해결하기 위해서는 < > 와 같은 태그를 사용하지 못하게 해야 한다. 태그 문자를 유니코드 &lt, &gt 등으로 치환하면 된다.

2.4. SQL 인젝션 취약점 분석

인젝션(Injection)은 사용자로부터 입력을 받는 부분에 해커가 작성한 특수한 문자 등을 삽입하는 것을 의미한다. 이러한 인젝션 공격은 데이터베이스나 XML, 텍스트 등 데이터를 전송할 때 제공하는 입력 인터페이스(텍스트박스, 셀렉트박스, 히든타입, 체크박스, 라디오 버튼 등)를 통해 서버에 정보를 전달하는 모든 경로를 따라 삽입된다. 이중 특별히 사용자가 데이터베이스 시스템에게 요청하는 SQL 쿼리문에 특수한 문자를 첨가하여 중요 정보를 얻거나 수정하는 공격을 SQL 인젝션이라고 한다. 실제 SQL 인젝션은 모든 인젝션 공격 중 가장 빈번하게 일어나는 형태 중 하나이다.

SQL 인젝션 공격을 하기 위해서는 데이터가 어떤 과정을 거쳐 저장과 조회가 되는지 알아야 할 필요가 있다. 우선 회원가입 하는 과정을 통해 아이디와 패스워드가 데이터베이스에 저장되고 로그인시 저장된 데이터를 내부적으로 SQL 쿼리를 사용해서 조회하게 된다. 다음은 asp로 작성된 로그인 처리 프로세스의 일부이다.

```
<%
userid = Request.Form("memberid")
pwd = Request.Form("pass1")
```

```
Set objRs =
Server.CreateObject("ADODB.Recordset")
strSQL = "SELECT * FROM member WHERE
bId = '" & userid & "' AND bPass= '"& pwd &
"'"
objRs.Open strSQL, objConn
Set Rs = objConn.Execute(strSQL)
.....
%>
```

사용자로부터 memberid와 pass1 변수값 GET이나 POST를 통해 받아들여 내부적으로 SQL 쿼리문을 통해 아이디와 패스워드가 일치하는 사용자를 검사한다. 로그인 양식에 아이디를 TEST라고 입력하고 패스워드를 ' or '1' = '1 라고 입력했다고 가정하고 SQL 인젝션이 어떤식으로 이루어지는지 분석해보도록 하겠다. 입력받은 문자열을 SQL 쿼리 부분에 그대로 대입해보면 결과를 쉽게 알 수 있다. 우선 userid 변수에는 TEST 가 저장 되고 pwd 변수에는 ' or '1' = '1 이 저장 될 것이다. 이와 같이 SQL 인젝션 공격을 통해 완성된 최종 쿼리는 다음과 같다.

```
SELECT * FROM member WHERE bId =
'TEST' AND bPass= '' OR '1' = '1'
```

원래 목적은 입력받은 아이디와 패스워드를 member 테이블에 있는지 없는지 검사하여 회원 등록 여부를 확인하는 것이지만 이렇게 SQL 인젝션이 적용돼 변경된 쿼리문을 보면 앞부분의 쿼리문에 상관없이 뒷부분의 OR '1' = '1' 이라는 문장에 의하여 항상 참이게 된다. 이렇듯 앞부분에 어떤 아이디를 써넣어도 항상 참이므로 인증을 통과하게 되는 것이다.

SQL 인젝션 취약점에서 작은따옴표가 가장 많이 사용되고 대표적인 특수문자이기 때문에 SQL 인젝션 공격을 막는 가장 일반적인 접근 방식은 사용자가 입력한 입력 값에서 작은따옴표를 제거하는 것이다.

물론 사용자 입력 데이터가 숫자가 일 경우에는 쿼리문에 작은따옴표가 일반적으로 포함되지 않는다. 따라서 공격자는 데이터 내용을 파괴할 수 있고 작은따옴표를 입력할 필요 없이 임의의 SQL을 입력할 수 있다는 문제가 남아 있다. 따라서 " ' \ / ; % @ * & () $ ^ ~ ? 등의 다양한 특수문자나 예외 문자 등을 필터링 해야 한다.

III. 웹 프락시(Proxy) 최적화 구현

웹 공격은 URL, Header, MIME 등 웹 프로토콜 구조의 각 부분의 취약성을 이용하여 이루어지고 각각의 특성에 맞도록 공격도 특화되어 있다. 웹 해킹은 시스템을 파괴하려는 불법적인 공격도 있겠지만 직접적인 공격이 아니더라도 중요한 정보를 탈취하려는 시도도 많이 이루어지고 있다. 따라서 웹에서 전송하는 민감한 정보들을 외부로 노출시키지 않기 위하여 웹 서버의 응답 메시지에 대하여 적절한 보안을 유지해야 할 필요가 있다. 이를 위해서는 웹 방화벽은 패킷의 내용을 열어 보는 형태의 프락시(Proxy) 방식으로 구현되는 경우가 많다.

이 장에서는 웹 사이트 취약점 스캔에 사용되는 파로스(Paros)의 내부 메커니즘을 분석하고 새로운 기능을 추가해 보도록 하겠다. 파로스는 공개용 웹프락시(web proxy) 툴로 잘 알려져 있고 기능 또한 막강하다. 하지만 2006년 이후로 더 이상 업그레이드되지 않아 최근 보안 이슈에 대한 대응이 미흡한 단점이 있다. 이에 본 장에서는 파로스 소스를 직접 이클립스를 통해 컴파일 해보고 소스코드를 분석해 보고자 한다. 더 나아가 프락시 시스템의 메커니즘을 이해한 후 소스코드에 덧붙여 새로운 기능을 추가하고 취약점을 쉽게 찾아 낼 수 있도록 최적화 하는데 목표를 두었다.

3.1. 웹 프락시 컴파일 환경

컴파일을 위한 실험 환경은 다음과 같다. 운영체제는 Windows 2008 Server R2 64bit와 Windows 7을 사용했고 JAVA 컴파일러는 J2SDK 1.6.0_25 버전을 사용하였다. 또한 이클립스(Eclipse)는 3.6.2 버전을 사용했으며, 웹 프락시로는 파로스의 마지막 버전인 paros-3.1.13을 사용하였다.

컴파일 방법은 이클립스에서 새로운 프로젝트를 생성한 다음 파로스 소스를 import 하는 방식을 사용한다. http://sourceforge.net/projects/paros/files/Paros/에서 소스를 내려 받아서 이클립스로 import 한후

Ant를 이용해서 컴파일 한다. Ant는 일종의 자바 컴파일 자동화 도구이다. C언어의 make와 비슷한 역할을 한다. Ant 뷰 영역에서 build.xml 라는 빌드파일을 선택하여 빌드 하면 된다.

3.2. 웹 프락시 최적화 실험

쓰레기 파일의 대표적인 예로 백업 파일을 들 수 있다. 백업 파일은 일반적으로 .bak, .old 등의 확장자 형태로 저장된다는 사실을 악용하여 해당 확장자를 가진 파일이 존재하는지 스캔하여 취약점을 발견하는 방법을 다루고자 한다. 웹 서버에 개발 시 사용했던 불필요한 쓰레기 파일을 놔두게 되어 공격자에게 노출된다면 php, jsp, asp 등의 서버 사이드 스크립트의 소스가 그대로 노출될 우려가 있다.

파로스는 기본적으로 .old, .bak, .inc 확장자를 가진 파일을 쓰레기 파일로 간주한다. 여기에 .tar, .tgz, .tmp를 추가해 보도록 하겠다. 파로스의 소스 디렉토리에서 org.parosproxy.paros.core.plugin 패키지 안의 TestObsoleteFile.java 파일을 열어 그림 1과 같이 세 개의 확장자를 추가하면 된다.

```
public class TestObsoleteFile extends AbstractAppPlugin {

    private final static String[] staticSuffixList = {
            ".old",
            ".bak",
            ".inc",
            ".tar",
            ".tgz",
            ".tmp"
```

그림 1. 특정 확장자 패턴 추가
Fig. 1 Adding specific extension pattern

3.2.1 사설 IP 찾기(Private IP disclosure)

사설 IP 찾기에 대한 설정은 org.parosproxy.paros.core.plugin 패키지 안에 있다. TestInfoPrivateAddressDisclosure.java 파일에 보면 patternPrivateIP 라는 이름의 변수에 각각의 사설 IP 패턴이 정규 표현식으로 정의돼 있다.

```
public class TestInfoPrivateAddressDisclosure extends
AbstractAppPlugin {


    // check for private IP list
    public static final Pattern patternPrivateIP =
Pattern.compile("(10\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}|172\\.\
\d{2,2}\\.\\d{1,3}\\.\\d{1,3}|192\\.168\\.\\d{1,3}\\.\
\d{1,3})", PATTERN_PARAM);
```

그림 2. 사설 IP 정규 표현식 패턴
Fig. 2 Private IP regular expression pattern

3.2.2 IIS 기본 인덱스 파일 찾기

IIS 기본 인덱스 파일 찾기는 사전 정의된 패턴을 루트(/) 디렉토리에 전송하여 응답코드가 200번인 경우(정상적인 응답코드가 왔을 경우)는 취약점이 존재한다고 판단한다.

기본 인덱스 파일 패턴이 입력되 있는 파일은 TestDefaultFileIIS.java 이다. 이 안에 보면 [그림 3] 처럼 addTest()라는 메소드 안에 기본 인덱스 파일 패턴이 입력되 있다. addTest() 메소드는 두 개의 인자를 입력받는데 첫 번째 인자는 디렉토리를 두 번째 인자는 파일명을 나타낸다. 파일이 여러 개인 경우 콤마(,)로 구분 할 수 있다. addTest() 메소드에 대한 정의는 AbstractDefaultFilePlugin.java 파일에 있다.

```
addTest("/","iisstart.asp,postinfo.html,_vti_inf.html");
        addTest("msadc","msadcs.dll");

        addTest("_vti_bin", "fpcount.exe,shtml.dll");
        addTest("_vti_bin/_vti_adm", "admin.dll");
        addTest("_vti_bin/_vti_aut", "author.dll");
        addTest("_cti_pvt",
"access.cnf,botinfs.cnf,bots.cnf,deptodoc.btr,dectodep.btr,lin
kinfo.cnf,service.cnf,services.cnf,svcacl.cnf,uniqperm.cnf,wri
teto.cnf");
```

그림 3. 기본 인덱스 패턴
Fig. 3 Basic index pattern

만일 기본 인덱스 파일로 default.htm이나 default,html, index.html, index.html 등을 추가로 찾고자 한다면 **addTest("/", "default, html, default. html, index. htm, index.html");** 처럼 addTest() 메소드의 인자로 추가하면 된다.

3.2.3 SQL 인젝션(SQL Injection Fingerprinting) 최적화

SQL 인젝션의 검사는 기본적으로 특수문자 작은따옴표(') 등을 삽입하여 웹 서버에 전달한 후 서버가 에러를 발생시키는지 확인하는 형태이다. 만일 SQL 인젝션 코드를 삽입했는데도 서버가 에러 메시지를 출력하지 않고 정상적으로 200번을 응답한다면 SQL 인젝션 취약점이 있는 것으로 간주한다.

SQL 인젝션과 관련된 소스는 TestInjectionSQLFingerprint.java 이다. [그림 4]는 TestInjectionSQLFingerprint.java 파일 내용의 일부로서 파로스에서 지원하는 기본 패턴 네 가지를 확인할 수 있다.

```
    private static final Pattern patternErrorODBC1 =
Pattern.compile("Microsoft OLE DB Provider for ODBC
Drivers.*error", PATTERN_PARAM);
    private static final Pattern patternErrorODBC2 =
Pattern.compile("ODBC.*Drivers.*error", PATTERN_PARAM);
    private static final Pattern patternErrorGeneric =
Pattern.compile("JDBC|ODBC|not a valid MySQL|SQL",
PATTERN_PARAM);
    private static final Pattern patternErrorODBCMSSQL =
Pattern.compile("ODBC SQL Server Driver", PATTERN_PARAM);
```

그림 4. SQL 인젝션 에러 메시지 패턴
Fig. 4 SQL injection error message pattern

동작원리를 간단히 설명하자면 파로스는 SQL 인젝션 공격 코드인 특수문자를 삽입하여 웹서버에 전달하여 응답을 살펴본다. 웹 서버가 보내온 응답에 [그림 4]에 포함된 에러메시지를 포함하지 않은 정상적인 페이지를 보내오면 해당 사이트는 SQL 인젝션 취약점에 노출 돼있는 것으로 본다. 하지만 위의 네 가지 에러 유형만으로는 취약점을 모두 선별할 수 없으므로 다른 취약점 패턴을 추가하는 실험을 진행하였다. TestInjectionSQLFingerprint.java 파일의 네 가지 패턴 아래에 차례로 변수 이름을 달리해서 추가하면 된다. 에러메시지는 단순하게 **Invalid query** 라고 하였다. 아래 내용과 같이 새로운 변수에 추가하면 된다.

이제부터 **Invalid query** 라는 메시지를 만나면 파로스는 해당 웹 서버가 취약하다고 판단할 것이다.

```
  - private static final Pattern patternErrorODBC
= Pattern.compile("Microsoft OLE DB Provider for
ODBC Drivers.*error", PATTERN_PARAM);
  - private static final Pattern patternErrorODBC2
= Pattern.compile("ODBC.*Drivers.*error",
PATTERN_PARAM);
  - private static final Pattern
patternErrorODBC3 = Pattern.compile("Invalid
query", PATTERN_PARAM); // Adding part
  - private static final Pattern patternErrorGeneri
= Pattern.compile("JDBC|ODBC|not a valid
MySQL|SQL", PATTERN_PARAM);
- private static final Pattern
patternErrorODBCMSSQL =
Pattern.compile("ODBC SQL Server Driver",
PATTERN_PARAM);
```

3.2.4 개인정보 유출방지 구현

개인정보유출은 최근 들어 가장 빈번히 발생하고 있는 보안 이슈중 하나다. 개인정보유출은 단순히 서버의 파괴에 그치지 않고 유출된 정보를 다른 범죄에 악용할 소지가 있으므로 중요한 문제다. 하지만 파로스는 오래전에 업데이트가 중단되었기 때문에 최근에 이슈가 된 개인정보 문제를 상대적으로 소홀히 다루고 있다. 이에 따라 본 논문에서는 파로스에 개인정보유출 중에서도 가장 흔하게 발생하는 주민등록번호 유출에 관한 패턴을 새로 만들어 최적화 해보고자 한다.

우선 주민등록번호를 탐지하기 위해서 특정 패턴을 전송할 필요는 없고, 돌아오는 응답 메시지만 검사하면 된다는 사실을 알아야 한다. 앞서 살펴본 사설 IP 찾기 기능과 유사하다고 보면 된다. 사설 IP 정규표현식 패턴 대신 주민등록번호 정규표현식 패턴을 사용하면 된다.

아래 예제와 같이 주민등록번호에 대한 정규표현식 패턴을 정의할 수 있다. 앞 여섯 자리 중 맨 앞 두 자리는 아무 숫자나 와도 되고 그 다음은 월을 나타내기 때문에 01~09 또는 11~12가 와야 되며, 마지막 두 자리는 날짜 이므로 01~09나 10~29 또는 30~31이 오면 된다. 뒷 7자리는 간단하다. 맨 앞 첫 자는 성별을 나타내고 남자는 1,3 여자는 2,4로 표현되며 그 뒤로는 6자리 임의의 숫자가 올 수 있다.

```
Front 6 digit :
\\d{2}(0[1-9]|1[0-2])(0[1-9]|[12][0-9]|3[01])
Back 7digit : [1-4]\\d{6}
```

TestInfoPrivateAddressDisclosure.java 파일을 복사하여 TestInfoJumin.java 라고 만든 후 내용 중 pat

ternPrivateIP 부분을 위의 정규표현식 패턴으로 수정하면 된다. 최종 수정된 내용은 다음과 같다.

```
public static final Pattern patternJumin =
Pattern.compile("(\\d{2}(0[1-9]|1[0-2])(0[1-9]|[12][0-9]|3[
01]))[\\-|\\s]+([1-4]\\d{6})", PATTERN_PARAM);
```

이제부터 웹사이트에서 오고가는 주민번호 패턴을 감지하여 파로스가 개인정보에 관한 취약점을 보고하게 될 것이다.

Ⅳ. 결 론

최근 들어 일반인들도 간단한 해킹 툴을 사용할 수 있게 되어 예기치 않은 악영향들이 기하급수적으로 늘어나고 있다. 이에 따라 해킹을 방어 하는 보안에 대한 관심이 한층 증가되고 있는 실정이다. 더불어 사내 인트라넷, 인터넷뱅킹, 전자정부 등 일상생활과 밀접한 여러 가지 서비스들이 인터넷에 의존하게 됨에 따라 더욱 보안의 중요성은 더욱 부각되고 있다. 본 논문에서는 다양한 웹 공격과 함께 대응 방안에 대해서 다루었다. 웹 공격이 어떤 형태로 이루어지는지 알아보았고 각각의 공격 패턴에 따른 기존의 대응 방안을 제시하였다.

웹 보안은 단순히 패킷의 패턴만을 감지하여 방어하기 힘들고 콘텐츠 영역까지 확인해서 불법적인 코드를 필터링 할 필요가 있는데 이 때 사용할 수 있는 유용한 도구로 웹 프락시(Proxy)를 들 수 있다. 이에 본 논문에서는 웹 프락시 중에서도 강력한 성능을 가지면서도 오픈소스로 제공되는 파로스(Paros)의 사용방법과 소스코드를 수정 및 컴파일하여 여러 가지 웹 공격의 패턴과 최적화된 대응 방법에 대한 실험을 실시하였다.

향후 연구로 구현된 웹 프락시 시스템에 예상되는 공격패턴을 모듈 형태로 쉽게 추가할 수 있는 메커니즘이 필요하다. 웹 공격의 형태가 점점 다양화 되고 최신 보안이슈와 관련된 취약점 공격이 많아짐에 따라 방어하는데 시간적 여유가 없다는 점에 착안한다면 실제 예상되는 공격패턴을 쉽고 빠르게 추가할 필요가 있다. 이를 통하여 빠르고 강력한 취약점 검출 엔진을 구축하여 웹 프락시 시스템의 성능을 높일 수 있다.

감사의 글

본 논문은 2014년도 광주대학교 대학 연구비의 지원을 받아 수행되었음.

References

[1] I.-Y. Lee, J.-I. Cho, K.-H. Cho, and J.-S. Moon, “A Method for SQL Injection Attack Detection using the Removal of SQL Query Attribute Values,” *J. of the Korea Institute of information Security & Cryptology*, vol. 18, no. 5, 2008, pp. 135-147.

[2] S.-J. Park and J.-H. Park, "Current Status and Analysis of Domestic Security Monitoring Systems," *J. of the Korea Institute of Electronic Communication Sciences*, vol. 9, no. 2, 2014, pp. 261-266.

[3] M. O'Neill, P. Hallam-Baker, and S. M. Cann, *Web Services Security*. New York : McGraw-Hill, 2003.

[4] S. Garfinke, *Web Security, Privacy and Commerce,* 2nd Edition, Sebastopol, CA : O'Reilly Media, 2002.

[5] C. Kaufman, M. Spiciner, and R. Perlman, *Network Security Private Communication in a PUBLIC World, 2nd Edition,* Englewood Cliffs, NJ : Prentice Hall, 2002.

[6] D.-Y. Kim, "Vulnerability Analysis for Industrial Control System Cyber Security," *J. of the Korea Institute of Electronic Communication Sciences*, vol. 9, no. 1, 2014, pp. 137-142.

[7] D.-K. Kang, M.-Y. Hyun, and C.-S. Kim, "Cyber trap : Unknown Attack Detection System based on Virtual Honeynet," *J. of the Korea Institute of Electronic Communication Sciences*, vol. 8, no. 6, 2013, pp. 863-871.

저자 소개

김광현(Gwang-Hyun Kim)

1989년 2월 광운대학교 전자계산학과 학사
1991년 2월 광운대학교 전자계산학과 석사
1997년 2월 광운대학교 전자계산학과 박사
2001년 8월~2002년 7월 Pennsylvania State Univ. Post-Doc
1997년 3월~현재 광주대학교 정보통신학과 교수
※ 관심분야 : 차세대인터넷, 인터넷 QoS, 네트워크 보안, 센서네트워크