웹 서버 전용 에이전트를 이용한 실시간 웹 서버 침입탐지에 관한 연구

진홍태* · 박종서**

요 약

인터넷 사용이 보편화됨에 따라 기존의 방화벽만으로는 탐지가 불가능한 웹 서버의 취약점을 이용한 공격이 나날이 증가하고 있다. 그 중에서도 특히 웹 어플리케이션의 프로그래밍 오류를 이용한 침입이 공격 수단의 대부분을 차지하고 있다. 본 논문에서는 웹 어플리케이션의 취약점을 분석한 후 취약점 발생 부분에 대해 웹 서버 전용으로 로그 분석을 해 주는 실시간 에이전트를 도입하였다. 실시간 에이전트는 공격 패턴을 비교·분석한 후 프로세스 분석기를 통한 결정(decision) 과정을 통해 침입으로 판단되면 해당 접속 프로세스(pid)를 제거한 후 공격 아이피를 차단함으로서 침입을 탐지하는 모델을 제시한다.

A Study on Real-Time Web-Server Intrusion Detection using Web-Server Agent

Hong Tae Jin* · Jong Sou Park**

ABSTRACT

As Internet and Internet users are rapidly increasing and getting popularized in the world, the existing firewall has limitations to detect attacks which exploit vulnerability of web server. And these attacks are increasing. Most of all, intrusions using web application's programming error are occupying for the most part. In this paper, we introduced real-time web-server agent which analyze web-server based log and detect web-based attacks after the analysis of the web-application's vulnerability. We propose the method using real-time agent which remove process ID(pid) and block out attacker's IP if it detects the intrusion through the decision stage after judging attack types and patterns.

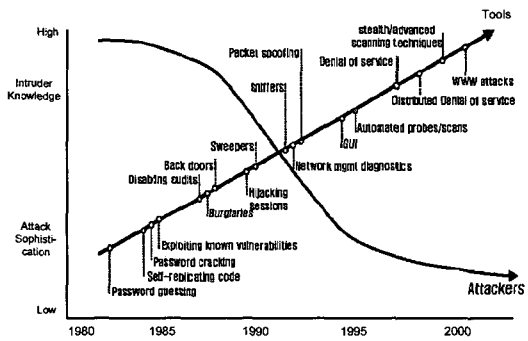
Key words : Web-Application, Intrusion, Vulnerability

• 본 연구는 인터넷 정보검색(IRC) 지역 연구센터(RRC)와 대학 IT 연구센터 육성지원사업의 연구결과로 수행되었음

* 한국항공대학교 대학원 컴퓨터공학과
** 한국항공대학교 컴퓨터공학과

1. 서 론

인터넷이 급속히 확산되고 보편화됨에 따라 인터넷을 이용하는 공격의 유형도 변화하고 있다. 전통적인 공격 (traditional attack) 방식은 주로 OS와 네트워크 서비스에 존재하는 취약점(vulnerability)을 목표로 했고 또한 버퍼 오버플로우와 같은 공격을 위해서는 어셈블러로 직접 공격 프로그램을 작성해야 했었다. 그러나 방화벽(firewall)의 도입이 보편화됨에 따라 이런 방식의 공격이 거의 불가능해 졌고 또한 OS 및 네트워크 서비스 개발업체들의 꾸준한 패치로 인해 전통적인 공격 방식으로는 침입이 거의 불가능하게 되었다. 그렇지만 80번 포트를 통해 침입하는 웹 공격의 경우에는 웹 서비스를 위해 공식적으로 허가된 합법적인 트래픽이므로 기존의 방화벽으로는 탐지할 수 없다. 또한 침입탐지시스템(IDS) 우회 기법을 통한 웹 서비스 공격이 고도화, 다양화, 대중화되어 감에 따라 IDS만으로는 그러한 웹 서버 공격 탐지가 사실상 불가능하다.



(그림 1) 최근 해킹 동향

이와 같은 이유로 오늘날의 공격 방식은 (그림 1)에서 알 수 있듯이 주로 80번 포트를 이용하여 웹 어플리케이션 내부로의 침입을 시도하는 웹 해킹으로 그 형태가 변화되고 있는데, 그 중에서도 특히 프로그래밍 오류를 이용한 공격은 서버를 다운시켜 서비스를 못하게 하는 DOS 공격이나 상대방의 정보를 빼내는 sniffing 공격처럼 가벼운 수준이 아니라 해당 서버의 모든 자료뿐만 아니라 심지어는 관리자 권한 획득을 통해 해당 웹 서버 전부를 장악할 수 있는 가장 위험성이 높고 경계해야 할 공격이다[1-4]. 따라서 본 논문에서는 프로그래밍 오류를 이용한 웹 공격이 발생시 웹 전용 에이전트가 공격에 사용되는 필수적인 절차나 유형을 추출하여 침입을 탐지하고 방어하는 메커니즘을 제시한다. 본 논문의 구성은 다음과 같다. 2장에서는 웹 어플리케이션의 동작 및 취약점을 살펴본 후, 3장에서 관련 연구로서 웹 해킹 및 공격자들의 침입 패턴에 관하여 설명할 것이다. 그리고 4장에서는 제안 모델의 구조 및 특징을 살펴본 후 사례 연구를 통해 이 시스템의 효율성을 검증한 후 결론을 맺는다.

2. 웹 어플리케이션의 취약점 분석

전자상거래와 같은 일반적인 웹 어플리케이션 동작 과정을 살펴보면 다음과 같다. 일반적으로 클라이언트에서 웹 서버로 요청(request)을 보내게 되면 웹 서버는 요청받은 쿼리를 웹 어플리케이션(C/C++, Perl, JSP, ASP, etc …)을 통해 해석하게 되고 DB서버로의 커넥션을 시도하여 그 결과를 클라이언트로 응답(reply)하게 된다. 그러나 그 과정에서 다음과 같은 몇 가지 취약점들이 발생할 수 있다. 현재 웹상에서 발생할 수 있는 취약점들을 살펴보면 다음과 같이 크게 4가지로 나눌 수 있다[5-7]

- **웹 서버 설정상의 오류** : 웹 서버 설정을 올바르게 하지 않음으로서 클라이언트 스크립트 명령을 수행할 수 있고 또한 쿠키 값 설정 실수를 통해서 쿠키 정보를 빼낼

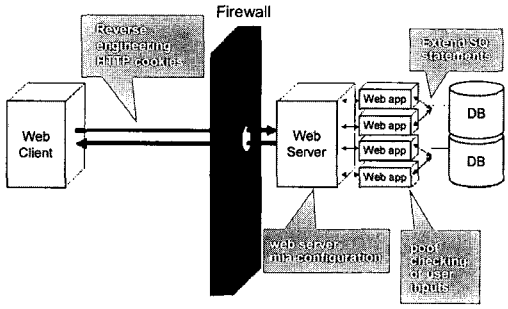
수도 있다. 그리고 마이크로소프트 IIS서버의 경우에는 URL 인코딩 버그와 같은 취약점이 발생할 수 있다.

- **웹 어플리케이션에서의 프로그래밍 오류 :** 클라이언트로부터 오는 값에 대한 검증 부분을 고려하지 않았기 때문에 발생할 수 있는 취약성이다. 이것은 현재 발생하고 있는 웹 해킹 공격수단의 대부분을 차지하고 있고 또한 웹 어플리케이션의 대부분이 데이터베이스와 연계하여 구동되고 있기 때문에 그 피해가 심각하다 할 수 있다.

- **SQL 쿼리문 삽입(SQL Injection) :** 일반적으로 대부분의 웹 사이트들은 다이내믹한 웹 페이지를 구성하기 위해 php, asp, jsp 등의 언어를 이용하여 웹 페이지를 구성하고 있다. 이런 모든 언어를 사용하는 거의 모든 웹 페이지에서는 데이터베이스를 연동하여 여러 가지 정보를 저장하거나, 읽어오는 동작을 수행하게 된다. 하지만 데이터베이스와의 연동에 있어서 쿼리를 작성해야만 원하는 정보를 얻을 수가 있는데, 이때 URL의 쿼리에 대한 입력값(', ", or, etc …) 검증 모듈을 고려하지 않아 발생할 수 있는 취약점이다.

- **Reverse-engineering을 통한 HTTP 쿠키값 획득 :** Sniffing과 Spoofing을 통해 패킷 정보를 수집하거나 변조하는 것을 예로 들수 있다. 이 방법은 실제로 웹 서버에 대한 공격을 하는 것이 아니라 단지 클라이언트와 서버간의 자료(password, cookie, etc …)수집을 통해서 원하는 정보 획득이 목표이다.

과거에는 웹으로 할 수 있는 일이 많지 않았기 때문에 웹의 보안이 크게 문제시 되지 않았고, 지금도 다른 부분 보다는 훨씬 그 위험도가 낮다고 할 수 있다. 일반적으로 웹 서비스만 제공하는 서버의 경우는 해킹의 위험도가 다른 서버보다 낮지만, 위와 같은 취약점들이 발생할 수 있으므로 그 위험성을 간과할 수 없다. 또한 최근에 많은 해킹 시도가 웹 서비스를 이용한 공격에서부터 시작되고 있기 때문에 웹의 보안이 어느 때 보다도 중요하다.

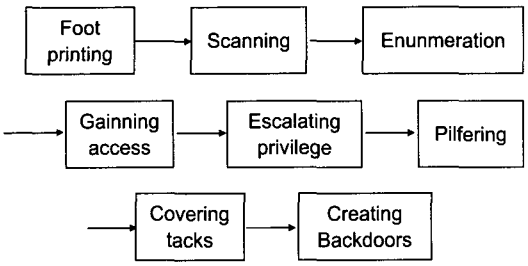


(그림 2) 웹 취약 포인트

3. 웹 해킹 및 침입 패턴 연구

웹은 앞에서 언급한 바와 같이 80번 포트로써 언제나 열려있다. 웹 서비스는 인터넷을 통하여 열려있어야만, 그 역할을 할 수 있기 때문이다. 초기의 해킹은 ID와 패스워드를 알아내어, 텔넷과 같은 서비스를 통하여 접속한 후, 서버의 특정한 취약성을 이용하여 ROOT(서버관리자)로 접근한다. 즉, 그 서버의 계정이 있어야만 특정한 서버에서의 취약성을 이용할 수 있는 것이다. 하지만, 그 계정을 얻어 내는 것은 그리 쉬운 일은 아니었다. 반면, 웹은 항상 실행되고 있고, 그 코드들은 서버에서 실행(html은 제외)되고 있다. 즉, 웹 서비스를 이용하기 위해서는 그 서버에 접근 할 수 있는 계정을 가지고 있다고 볼 수가 있는 것이다. 그렇기 때문에 웹에 취약성이 생기면 그 취약성을 이용하여 바로 서버에서 자신이 원하는 일을 시킬 수 있게 되는 것이다. 물론,

오직 웹으로만 자신이 원하는 일을 하는 것은 힘들다. 웹이 열려있는 만큼, 그만큼 권한이 약하고, 할 수 있는 일이 적기 때문이다. 그렇지만 웹 해킹의 진정한 의미는, 자신이 서버의 원하는 일을 시킬 수 있고, 그것이 또 다른 치명적인 해킹으로 이어지게 된다는 점이다. 한편, McClure et al.는 Hacking Exposed라는 책에서 일반적으로 해커가 시스템에 침입했을 때 취할 수 있는 공격의 일반적인 단계를 다음 (그림 3)과 같이 정의하였다[8]



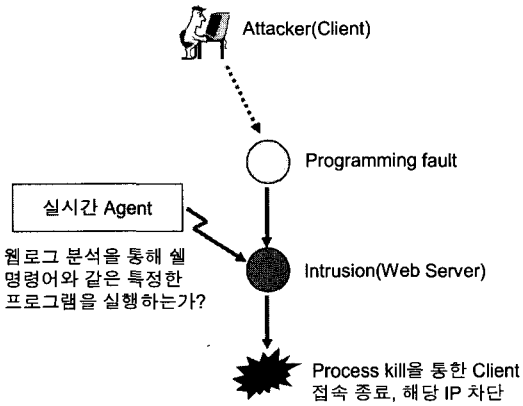
(그림 3) 공격의 일반적인 단계

본 논문에서는 이러한 공격의 일반적인 단계를 바탕으로 웹 서비스의 취약점을 분석한 후 웹 공격의 대부분을 차지하고 있는 프로그래밍 오류를 이용한 웹 공격에 대하여 웹 서버 전용의 에이전트를 도입함으로써 침입을 탐지하고 웹 서버를 보호하고자 한다.

4. 제안 모델

4.1 등장 배경

일반적으로 어플리케이션에서 기존의 자동화 툴을 통한 소스코드 오류 검출 방법이나 패치를 통하여 알려진 프로그래밍 오류를 어느 정도 보완할 수는 있지만 근본적으로 프로그램 개발자들이 설계 단계에서부터 보안에 대한 인식이 없이 기존의 개발툴이나 습관에 의존하여 웹 어플리케이션을 제작하기 때문에 취약점이 존재하지 않는 완벽한 웹 어플리케이션을 기대하기는 어려운 현실이다[9]. 일단 공격자가 프로그래밍상의 취약점이 존재하는 웹 서버에 침입했을 경우에 취할 수 있는 행동으로는 시스템 명령어를 통한 대상 서버의 정보수집, 파일 전송 프로그램(wget, tftp)을 이용한 악의적인 코드전송, 리버스 텔넷을 위한 Netcat 프로그램을 사용하는 등의 일정한 행동 패턴들을 취하게 된다.



(그림 4) 침입 탐지 메커니즘

이러한 행동 패턴들을 바탕으로 (그림 4)와 같이 본 논문에서는 웹 서버 전용 에이전트들이 웹 로그 마이닝을 통해 기존의 분석된 공격 패턴들과 비교하고, 로그와 프로세스를 분석한 후 결정 과정을 거친 후 침입 여부를 판단해 본다. 그리고 만일 침입으로 판단이 되면 공격자와 연결을 담당하고 있는 접속 프로세스(pid)를 찾아내어 프로세스 kill을 통해 그 프로세스를 강제로 종료하고 공격자의 재접속 시도를 방지하기 위해 해당 아이피를 실시간으로 차단하는 침입 탐지 방식을 제안한다.

4.2 데이터 전처리

일반적으로 프로그래밍 오류를 이용한 웹 서

버 공격은 대부분이 아래 <표 1>과 같이 URL의 패스와 쿼리 정보를 조작하는 행위이므로 기본적인 침입 탐지 접근 방법은 C. Kruegel이 제안한 대부분의 웹 서버들에 나타나는 로그의 HTTP request를 통해서 분석한다[10, 11]. 더욱 구체적으로는 GET request들에 초점을 맞추었다. 웹 에이전트가 추출하는 데이터는 GET request들의 헤더 데이터와 POST/HEAD request들은 고려하지 않는다. 또한 침입 탐지를 위한 입력은 <표 2>와 같이 여러 상태 코드들 중 접속 성공을 나타내는 200번 코드로 구성된 GET request로부터 추출되어진 URL의 집합 $U=\{u_i, u_2, \cdots, u_m\}$로 구성된다.

〈표 1〉 공격에 따른 패스와 쿼리 정보

공격 유형	로그 기록 정보
remote file include	"GET /Hakbu/data1_8.jsp?dir=http://www.kof.co.kr&cmd=ls -al HTTP/1.1" 200 22016
SQL injection	"GET /servlet/JMBoard?tablename=hk_notice' or ''='&mode=view&boardpage=1&searchword=&searchscope=&category=&no=245 HTTP/1.1" 200 50581
directory listing	"GET /%3f%/index.jsp HTTP/1.1" 200 45102
password file listing	"GET /.../../../../../etc/passwd HTTP/1.1" 200 331

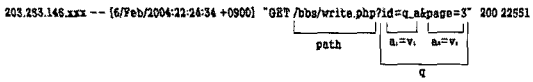
URL U_i는 원하는 리소스($path_i$), 부가적인 패스 정보 요소($pinfo_i$), 그리고 부가적인 쿼리 스트링(q)에서 존재하는 패스의 구성으로 나타낸다. 쿼리 스트링은 참조된 리소스에서 파라미터를 넘겨주는데 사용되며 문자 "?"에 의해 구분된다. 쿼리 스트링은 일치하는 값들을 갖는 파라미터들(또는 속성들)의 정렬된 리스트로 구성된다.

$q=(a_1, v_1), (a_2, v_2), \cdots, (a_n, v_n)$, 즉 $a_i \in A$이고, v_i는 문자 값들이다. 집합 S_q는 쿼리 q의 속성들의 부분집합 $\{a_j, \cdots, a_k\}$로 정의된다.

〈표 2〉 HTTP/1.1 상태코드

상태코드	설 명	상태코드	설 명
100	Continue	404	Not Found
101	Switching Protocols	405	Method Not Allowed
200	OK	406	Not Acceptable
201	Created	407	Proxy Authentication Require
202	Accepted	408	Request Time-out
203	Non-Authoritative Information	409	Conflict
204	No Content	410	Gone
205	Reset Content	411	Length Required
206	Partial Content	412	Precondition Failed
300	Multiple Choices	413	Request Entity Too Large
301	Moved Permanently	414	Request-URI Too Large
302	Moved Temporarily	415	Unsupported Media Type
303	See Other	500	Internal Server Error
304	Not Modified	501	Not Implemented
305	Use Proxy	502	Bad Gateway
400	Bad Request	503	Service Unavailable
401	Unauthorized	504	Gateway Time-out
402	Payment Required	505	HTTP Version not supported
403	Forbidden		

(그림 5)는 웹 서버에 나타난 로그 기록의 예를 보여준다. 이 예에서 쿼리 q는 $S_q=\{a_1, a_2\}$이므로 $u_i=\{Path, S_q\}$이다.



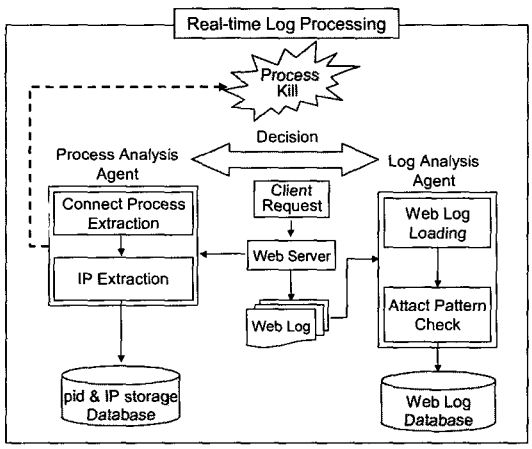
(그림 5) 웹 로그 정보

여기서 정의한 모델은 각각의 프로그램이 사

용한 쿼리와 파라미터 그리고 그 값들에 초점을 맞춘다. 프로그래밍 오류를 이용한 공격 기법들 대부분이 쿼리를 조작하는 공격이므로 쿼리와 패스 정보를 하나의 쌍으로 묶어 데이터 모델을 정의할 필요가 있다. 이러한 로그 분석 모델과 이를 처리하는 웹 에이전트를 바탕으로 다음 장에서는 침입을 탐지하는 메커니즘을 설명하고 사례 연구를 통하여 그것을 검증한 후 평가하고 결론을 맺고자 한다.

4.3 제안 모델의 구조

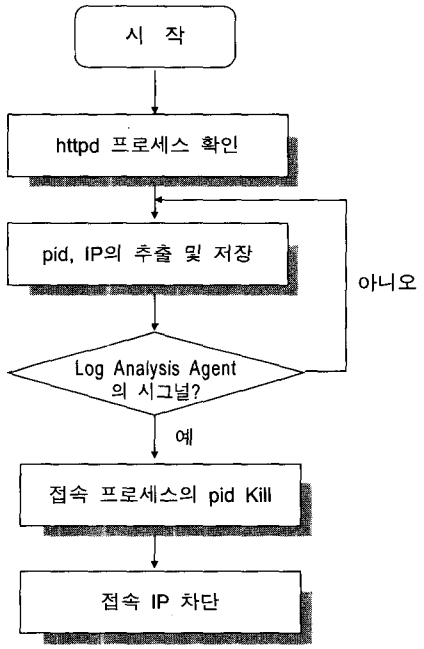
(그림 6)은 웹 서버의 접속 프로세스를 관리하는 에이전트와 로그를 분석하는 에이전트를 통해 침입 탐지가 어떻게 이루어지는지에 대한 전체적인 구조를 보여 주고 있다. 각각의 에이전트는 웹 서버와 라우터 또는 IDS나 방화벽 사이에 탑재되어 실시간으로 웹 서버로 들어오는 모든 커넥션을 수집하고 분석하여 체계적인 방법으로 데이터베이스화 한다. 특히 전처리 되어진 정상적인 커넥션에서의 패스와 쿼리 정보를 바탕으로 패스를 조작하거나 쿼리를 조작하는 공격에 대해 특정 쉘 명령어의 사용이나 행동 패턴들을 감시한다. 각 에이전트의 역할을 살펴보면 다음과 같다.



(그림 6) 웹 에이전트를 통한 침입탐지

- **Process Analysis Agent** : 클라이언트 로부터 접속 요청이 발생할 때마다 실시간으로 모든 접속 프로세스(pid)와 IP를 검출한다. Log Analysis Agent를 통해 공격 패턴과 일치하는 결과를 얻게 되면 즉시 해당 접속 프로세스를 제거하여 공격자와의 접속을 해제한 후 재접속 방지를 위해 IP를 차단한다. (그림 7)은 Process Analysis Agent의 전체 수행 흐름도이다.

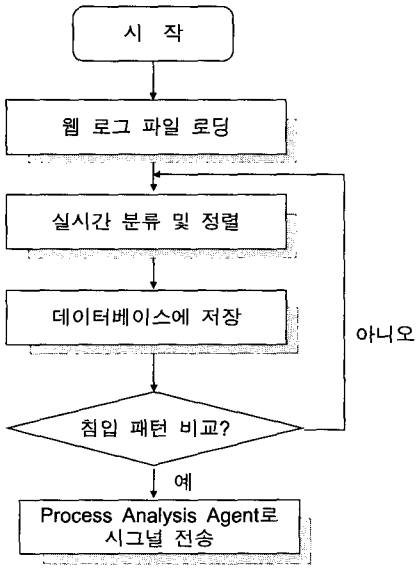
- **Log Analysis Agent** : 웹 서버 로그를 실시간으로 분류(classification)하고 정렬(order)한다. 침입 패턴들을 비교하기 위해 각각의 URL에서 특정 문자열를 검색하는 Token Finder와 같은 방법들을 사용한다. 침입 패턴을 추출하면 Process Analysis Agent로 그 결과를 전송한다. (그림 8)은 이의 수행 흐름을 보여준다.



(그림 7) Process Analysis Agent의 전체 동작 흐름도

4.4 제안 모델의 장점 및 단점

본 논문에서 제안한 에이전트 기반 모델은 침입이 이루어진 후에 로그 분석을 통해 실시간으로 판단하고 대처하는 방식이므로 공격자의 패턴 유형만 제대로 분류되고 클러스터링(clustering)되어진다면 즉각적인 의사결정을 통한 해당 프로세스 제거가 가능하기 때문에 침입 탐지율이 높은 웹 서버 보안 구현이 가능하다. 또한 웹 서버로 접속하는 모든 접속 프로세스를 실시간으로 모니터링 하고 있기 때문에 프로세스 분석을 통해 DOS와 같이 웹 서버의 자원을 고갈시킴으로서 웹 서버의 서비스를 하지 못하게 하는 공격들을 차단할 수도 있다. 그렇지만 대용량 웹 서버의 경우에는 보통 초당 몇 백 메가바이트(MegaByte)의 로그가 기록되므로 실시간으로 로그 기록들을 분석하기 위해서는 고성능의 하드웨어 사양을 요구하고 공격 패턴들의 분류와 클러스터링을 위한 실시간 모니터링이 필요하므로 이에 따른 프로세스에 과부하 문제가 발생할 수 있다.

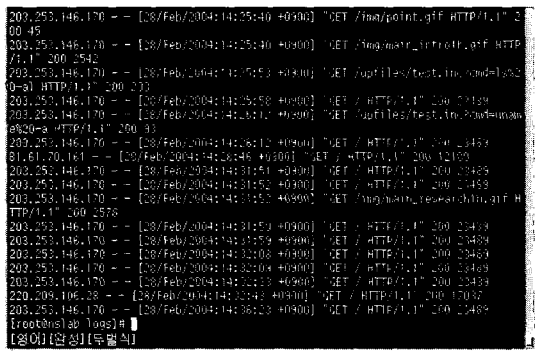


(그림 8) Log Analysis Agent의 전체 동작 흐름도

5. 사례 연구

여기서는 프로그래밍 오류의 하나인 php파일 업로드 버그를 통해 제안 모델의 효율성을 검증하고자 한다. php 파일 업로드 버그는 php 언어를 사용하는 일반적인 파일 업로드 가능한 게시판에서 확장자를 체크하지 않기 때문에 클라이언트 측에서 임의의 코드를 실행시킬 수 있는 버그이다. (그림 9)는 임의의 공격자가 php파일 업로드 버그가 있는 웹 서버에 침입한 후 정보 수집을 위해 몇몇 시스템 명령어를 실행했을 때의 웹 서버 로그 파일에 나타난 기록과 그 접속 프로세스의 pid를 포착한 것이다[12]. 이 그림에서 알 수 있듯이 php파일 업로드 버그의 경우 일단 침입이 성공하게 되면 공격자들의 궁극적인 목표인 웹 상에서 nobody권한으로 쉘 명령어의 실행이 가능해진다. 또한 취약성이 발견된 커널인 경우는 공개된 익스플로잇 코드를 통해 관리자 권한 획득도 가능하다. 그렇지만 에이전트들이 로그 분석을 통해 공격자들의 정확한 행동 패턴을 분류해준다면 그림에서와 같이 pid 추출을 통해 실시간으로 공격 IP를 차단하는 것이 가능하다.

(그림 9a) 접속 프로세스의 pid 정보



(그림 9b) 쉘 명령어 탐지

(그림 9a)는 Process Analysis Agent에서 접속 프로세스의 pid를 추출하는 것을 보여준다. 유닉스 계열에서는 현재 System에서 돌아가는 모든 Process에 의해서 Open된 파일들에 대한 정보를 보여주는 lsof(List Open Files)을 이용하여 구현이 가능하며 kill 명령어를 사용하여 해당 프로세스의 pid 제거가 가능하다. (그림 9b)는 Log Analysis Agent가 정상적인 커넥션 중에서 쿼리 정보를 분석하여 쉘 명령어를 탐지한 화면을 보여준다. 실시간으로 특정 쉘 명령어의 사용을 탐지하기 위해서는 정상적인 패스와 쿼리의 정보를 분류하고 정렬하는 기능이 필요하며 이는 고성능의 하드웨어를 사용함으로써 구현 가능하다.

6. 결론 및 향후 연구

본 논문에서는 웹 로그 마이닝을 통해 실시간으로 웹 서버의 침입을 탐지하는 모델을 제안하였다. 기존의 웹 소스코드 오류 검출 자동화 툴이나 방화벽만으로는 80번 포트를 통해 실시간으로 이루어지는 웹 관련 공격에 노출되어 있으므로 이를 해결하기 위해 기존의 웹 어플리케이션의 취약점을 분석하였다. 또한 데이터 전처리를 통한 Agent의 도입으로 프로그래밍 오류를 이용한 웹 공격을 탐지하고 웹 서버를 보호하기 위해 웹 관련 로그를 분석한 후 실시간으로 판단하고, 해당 접속 프로세스를 제거하여 공격을 차단할 수 있는 메커니즘을 제안하였다. 사례 연구를 통하여 공격자의 패턴 유형만 제대로 분류되고 클러스터링 되어진다면 즉각적인 의사결정을 통한 해당 프로세스 제거가 가능하기 때문에 침입 탐지율이 높은 웹 서버 보안이 가능함을 살펴보았다. 본 논문에서 제안한 방법을 확대해서 향후에는 프로그래밍 상의 오류를 이용한 침입 탐지뿐 아니라 SQL Injection, XSS(Cross Site Script), Cookie sniffing or spoofing 등의 공격 탐지를 위한 연구가 필요하고 좀더 효율적으로 공격자들의 행동 패턴을 탐지하기 위해 분석된 공격자들의 행동 패턴들을 분류할 수 있는 분류 모델과 기존의 침입탐지 시스템과 결합하여 침입 탐지율을 높이고 더욱 웹 서버의 성능을 향상 시킬 수 있는 방법에 관한 연구가 필요하다.

참 고 문 헌

[1] http://www.stgsecurity.com, STG 시큐리티(주), "웹 해킹기술".

[2] http://www.krcert.or.kr, "인터넷침해사고대응지원센터", 인터넷 통계.

[3] http://www.securitytracker.com, Security-Tracker Statistics, April 2001 - March 2002.

[4] http://www.cert.org, CERT/CC Incident and Vulnerability Trends, May 2003.

[5] http://www.ezhack.net, 황순일, "웹 어플리케이션 해킹".

[6] "Web Application Security Secrets & Solutions", 사이버 출판사, 2002.

[7] "리눅스 웹 서버와 실전 웹 해킹", 세화출판사, 2003.

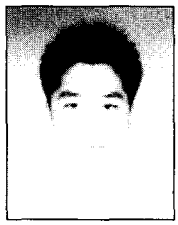
[8] Joel Scambray, Stuart Mcclure, and George Kurtz. Hacking Exposed : Network Security Secrets & Solutions/2nd ed, McGraw-Hill, 2001.

[9] "소스코드를 이용한 웹 응용 취약점 분석에 관한 연구", CISC 2003, pp.458-462, 추계학술대회.

[10] C. Kruegel and G. Vigna, "Anomaly Detection of Web-based Attacks", ACM CCS, 2003.

[11] R. Fielding et al. Hypertext Transfer Protocol - HTTP/1.1. RFC 2616, June 1999.

[12] http://www.null2root.org, "Hacker group [NULL@ROOT]", 해킹강좌.



진 홍 태

2003년 한국항공대학교 항공통신 정보공학과(공학사)
2003년~현재 한국항공대학교 컴퓨터공학과 네트워크 보안연구실 석사과정

박 종 서

1983년 한국항공대학교 항공통신 공학과(공학사)
1986년 North Carolina State Univ. 대학원 졸업(공학석사)
1994년 Pennsylvania State Univ. 대학원 졸업(공학박사)
1996년~현재 한국항공대학교 컴퓨터공학과 부교수