https://doi.org/10.5392/JKCA.2022.22.02.071

보안이 강화된 특수목적용 웹서버 설계 및 구축 제안

Proposal for Designing and Building a Special Purpose Web Server with Enhanced Security

홍성락, 조인준
배재대학교 사이버보안학과

Seong-Rak Hong(ennp@kakao.com), In-June Jo(injune@pcu.ac.kr)

요약

현재 웹 서버의 보안을 위해 관제와 모의 해킹을 한다고 해도 계속해서 취약점이 발생하고 해킹을 당하는 것이 현실이다. 해당 문제를 해결하기 위해 L4와 L5 사이에서 소켓을 사용해 모든 웹 통신을 제어할 수 있는 보안 웹 서버를 개발했다. 그리고 HTTP 응답을 줄 때 매번 파일과 헤더를 합치는 행위를 미리 합쳐놓는 방식을 제안했다. 그 결과 보안과 속도를 둘 다 향상할 수 있었다. 따라서 본 논문에서는 관제와 모의 해킹을 해도 취약점이 발생하는 이유와 그것에 대한 해결방안과 더 나아가 DB까지 보안을 유지할 수 있는 보안 웹 서버 개발 방식을 제안하였다.

■ 중심어 : | Socket | TCP/IP | WEB | HTTP | Linux | Multiple Fork |

Abstract

Currently, even if control and mock hacking are performed for the security of web servers, vulnerabilities continue to occur and be hacked. To solve this problem, we have developed a secure web server that can control all web communication using sockets between L4 and L5. And when giving HTTP responses, we proposed a method of combining files and headers in advance. As a result, both security and speed could be improved. Therefore, in this paper, we proposed the reason why vulnerabilities occur even if control and mock hacking occur, a solution to it, and a security web server development method that can maintain security up to DB.

■ keyword : | Socket | TCP/IP | WEB | HTTP | Linux | Multiple fork |

I. 서 론

현재 지속적인 보안관제에도 불구하고 웹 서버에게 취약점이 발생하여 해킹을 당한다. 해킹을 당하는 원초적인 이유는 웹 서버를 직접 개발하지 않았기 때문이라고 생각한다. 따라서 관제를 한다고 해도, 웹 방화벽만 우회한다면 웹 서버에 취약점을 이용해서 해킹을 할 수 있다.

웹 서버는 크게 Front-end와 Back-end로 구분할 수 있다. Front-end의 유명한 프로그램은 Apache 서버[1], Back-end의 유명한 프로그램은 PHP 서버[2]라고 할 수 있다. 이들 프로그램을 사용해서 웹서버를 구축하고 보안 취약점을 점검하고자 모의 해킹 서비스를 받는다고 가정을 해보자. 알려진 모의 해킹 결과에서

접수일자 : 2021년 10월 25일
수정일자 : 2021년 11월 23일
심사완료일 : 2021년 11월 23일
교신저자 : 조인준, e-mail : injune@pcu.ac.kr

얻어진 취약점 분석 결과 다음과 같은 문제점이 이들 서버에 내제되어 있다는 점을 들 수 있다. 첫째, 이들 웹 서버가 “conf” 값과 백 엔드 소스 코드에 대해서만 점검이 진행된다는 점을 들 수 있다. 둘째, 추가적인 프로그램의 취약점 분석 부담이 수반되어야 한다. 즉, 백 엔드 소스 코드에서 취약점이 발생하는 경우 왜 발생하는지를 추적하기 위해 백 엔드 프로그램의 정적분석과 동적 분석을 하여 백 엔드 프로그램의 소스 코드 수정이 필요하다. 하지만, Apache와 PHP 서버가 방대하기 때문에 손쉽게 소스 코드를 수정하기가 힘들다는 점을 들 수 있다.

본 논문에서는 이에 대한 새로운 해결책으로 소켓을 사용해서 웹 서버 개발하는 방안을 제안하였다. 소켓은 L4와 L5를 이어주는 함수이다. 즉, 처음부터 보안을 최우선으로 두고 L4와 L5 사이에서 웹 서버를 개발하는 것이다. 그렇게 되면 웹 서버에서 입출력하는 거의 모든 것을 제어할 수 있게 된다. 그렇기 때문에 완벽한 보안에 가까운 웹 서버를 개발할 수 있다. 본 논문에서 제안한 소켓 웹 서버를 설계 및 구축하는 과정에서 중점적으로 고려한 사항 및 특성을 개략적으로 요약하면 다음과 같다.

Apache와 PHP 서버 프로그램은 범용성이 강하기 때문에 많은 것을 메모리에 올려야 한다. 하지만 본 논문에서 제안한 소켓으로 간단하게 웹 서버를 개발할 수 있다. 즉, 범용성이 아닌 각 홈페이지에 맞는 맞춤형 소켓 웹 서버를 만들면 보안과 속도를 같이 제고시킬 수 있다는 장점을 지닌다.

제안한 소켓 웹 서버는 C언어로 개발을 했다. 다양한 언어를 동일한 환경에서 비교를 한 결과, C언어가 가장 빠르기 때문에 선택했다. 그리고 메모리를 적게 사용하면서 완벽한 보안을 위해 완벽한 필터링이 되는 구조로 개발했다. 그 구조를 간단하게 설명하자면, 서버에 들어온 “Path”와 “Query” 문자열 “index.php?id=pw” 을 하나씩 아스키코드값으로 변경 후 더한 값으로 해당 요청을 처리했다. 그리고 문자열 비교를 통해 한 번 더 확인하는 구조이다. 하지만 본 논문에서 제안한 방식은 모든 웹 서버 파일(즉, CSS, JS, PNG 등)들에 대해서도 “Path”와 “Query” 문자열 처리를 해야 한다. 따라서 처리하는 소스 코드를 모두 생성해야 했다. 작은 크기의 웹 서버라면 가능하겠지만 중간 크기의 웹 서버부터는 그 수가 너무 많아 수작업으로 이를 할 수는 없다. 그래서 보안 웹 서버 개발을 도와주는 자동화 프로그램을 개발했다.

소켓으로 개발을 하다 보니, 많은 것을 제어할 수 있었다. 동적 분석을 통해 Apache와 PHP 서버에서는 응답 헤더를 요청 시마다 파일과 응답 헤더를 붙여서 나가는 것을 확인했다. 본 논문에서는 해당 모든 파일 앞에 헤더를 미리 붙이는 방식을 개발했다. 따라서 웹 서버가 요청 값을 처리하고 응답을 보낼 때 기존 웹 서버들은 그때서야 헤더를 붙이지만, 본 논문에서의 방식은 미리 응답 헤더가 붙어있기 때문에 요청한 파일을 바로 응답하면 된다. 이 방식으로 시간을 조금 더 줄일 수 있었다. 응답 헤더를 미리 붙이는 행위는 자동화 프로그램으로 수행되도록 하였다.

지금까지는 웹 서버에 대한 보안에 관해서 이야기를 했다. 하지만 웹 서버에는 DB 서버도 연동된다. No-SQL과 SQL 사용을 둘 다 해보니, 두 방식 중에 선택할 수는 없었다. No-SQL이 필요한 곳에서는 No-SQL을 사용하는 것이 더 빠르고, SQL이 필요한 곳에서는 SQL을 사용해야 한다는 것이 본 논문의 결론이다. 하지만 이대로 DB 서버를 운영하면, SQL인 경우 SQL-Injection, No-SQL인 경우 새로운 취약점이 발생할 수 있다. 따라서 본 논문에서는 웹서버를 2중으로 보안을 유지했듯이 DB 서버도 2중으로 보안을 유지하는 방식을 제안하였다. 취지는 좋지만 모든 DB쿼리와 데이터를 미리 프로그램에 넣어둘 수는 없다. 따라서 본 논문에서는 기존 No-SQL과 SQL에서 사용하는 방식이 아닌, 내부에서 사용하는 특정키를 이용하는 방식과 패스워드의 해시값을 질의하는 방식 두 가지를 제안했다. 물론 이 부분은 각 웹 서버의 DB 구조에 따라 많이 변경될 수 있다. 본 논문에서는 이미 웹 서버에서 이중 보안을 적용했고, DB에서는 이중 보안 방안을 제안했다. 즉, 본 논문에서 제안한 방식으로 웹 프로그램을 개발했다면 속도와 보안 모두를 제고시킬 수 있다.

결론적으로 본 논문에서 제안한 소켓 웹서버를 대상으로 보안관제와 모의 해킹을 시도해도 웹 서버에 취약점이 발생하는 원초적인 요인들을 첫째, 소켓을 사용해서 직접 웹 서버를 제작하고, 둘째, DB에 직접적인 접

근이 불가능하도록 구현하는 방식으로 제거하였기 때문에 보안 및 속도까지 향상할 수 있는 새로운 방안이라 할 수 있다.

II. 범용 웹서버 기본동작 분석

1. 범용 웹 서버 프로그램(Apache 및 PHP) 분석

1.1 Apache 프로그램

Cent 운영체제 기준, Apache를 실행(즉, service httpd start)하면 [그림 1]처럼 생성된 것을 확인할 수 있다. 여기에서 GET 헤더가 들어올 때 Apache가 어떤 처리를 하고 어떤 함수들을 호출하는지를 분석했다. 분석은 해당 명령어 "ltrace -tt -p PID"로 가능하다. PID에는 프로세스 아이디를 입력하면 된다. 하지만 [그림 1]에서 본 것처럼 자식 프로세스가 여러 개 실행되어있어 어떤 것을 분석할지 모를 것이다. 그래서 사진 기준 trace에 123997을 먼저 실행해두고 계속 GET 요청을 했었다. 나오지 않으면 다른 자식 프로세스들에 접근한 후 계속 GET 요청을 했었다. 이 과정을 반복하다 보면 GET 요청을 했을 때 처리하는 것을 [그림 2]처럼 확인할 수 있다.

[그림 2] 작업은 "\r\n" 기준으로 구분하는 것이다. 따라서 "test_1\r\ntest_2\r\n\r\n"이라는 헤더가 들어왔을 때 "test_1\r\n" 과 "test_2\r\n" 과 "\r\n"처럼 나눠진다는 것을 확인했다.

```
root      123996      1  1 08:03
apache    123997 123996  0 08:03
apache    123998 123996  0 08:03
apache    123999 123996  0 08:03
apache    124000 123996  0 08:03
apache    124001 123996  0 08:03
apache    124002 123996  0 08:03
```

그림 1. ps -ef 결과

```
memcpy(0x559b059b2948, "Host: 192.168.127.3\r\n", 21) =
strlen("Host: 192.168.127.3")  = 19
```

그림 2. Apache 프로그램 ltrace 결과 V1

그리고 [그림 3]에서는 strchr 함수를 이용해서 strchr 함수의 1번째 매개변수가 0번째 매개변수에 존재하는지 비교했다. 이 분석을 통해 Apache는 이런 방식으로 서버의 결과를 던져준다는 것을 알게 되었다. 하지만 요청을 받고 처리하고 응답을 보내는 것까지의 함수 실행이 너무 많은 것도 확인했다.

```
strlen("/")                              = 1
strlen("var/www/html/index.html")
apr_palloc(0x559b059b1098, 26, 0,
memcpy(0x559b059b2f60, "/\0", 2) =
apr_array_push(0x559b059b2ee8, 0x7
strchr("var/www/html/index.html",
memcpy(0x559b059b2f61, "var\0", 4)
strchr("www/html/index.html", '/')
memcpy(0x559b059b2f64, "/www\0", 5
strcmp("/var/www/", "/var/www/") =
```

그림 3. Apache 프로그램 ltrace 결과 V3

1.2 PHP 프로그램

Apache와 연동해서 사용하는 PHP 기준이며, 해당 테스트에 사용한 테스트 PHP 소스 코드는 'echo "test"' 이다.

일단 읽어오는 부분은 Apache와 동일하기 때문에 다를 건 없었다. 이제 웹 서버에서 처리가 끝났다면 응답을 줘야 하는데, [그림 4]에서 응답 헤더를 준비하고 있는 것을 확인했다. 또한 ?strcasecmp 함수를 사용해서 대소문자 상관없이 문자를 비교하고 있다. 하지만 대부분 사용자는 이 사실에 대해서 인지하고 있지 않을 거로 생각한다.

```
strcasecmp("Host", "Content-type") = 5
strcasecmp("Host", "Content-length") = 5
strcasecmp("Host", "Authorization") = 7
strcasecmp("Host", "Proxy-Authorization") = -8
strcasecmp("Host", "Proxy") = -8
strlen("Host")    = 4
```

그림 4. PHP 프로그램 ltrace 결과

1.3 분석 결론

분석 결과 함수를 호출하고, 확인하고, 처리하는 것까지 기본적으로 실행해야 하는 것이 많다. 즉, 간단한 HTTP 헤더와 html 언어를 보내는 데 시간이 오래 소요된다는 것이다. 그래서 Apache나 PHP에서 최적화

를 해서 서버의 성능을 올리려고 한다면, 설정값을 최적화에 가깝게 설정하거나 PHP의 경우 최적화에 가까운 코드를 작성하는 것이 최선일 것이다. 하지만 위에서 봤듯이 기본적으로 해당 프로그램이 수행하는 기본 함수 및 소스 코드를 직접 수정하지 않는 이상 정밀한 최적화 작업은 힘들 것이다.

2. 범용 웹 서버 제작 언어별 속도 분석

범용 웹서버는 다양한 언어로 개발되고 있다. 따라서 본 논문에서는 대표적인 4개의 언어(C, C++, Python, Java)를 가지고 "test" 문자열을 출력하는 프로그램을 테스트 선상에 올려서 비교했다.

이제 명령어 실행 시간 비교해 보겠다. 정확한 결과 산출을 위해 같은 조건에서 10번을 실행하여 [그림 5-그림 8]과 같은 결과를 얻었다. 이들 각 언어에 대한 평균값은 다음과 같다.

각 10번 시도 결과, C 언어와 C++ 언어는 평균 3ms, Python은 24ms, Java는 평균 95ms 나왔다. 이 중에서 C언어는 다른 언어들과는 다르게 10번 시도의 결과가 균일하다. 이 하나의 실험으로 판단 내릴 수는 없지만, 어느 정도의 안정성을 가지고 있고, 속도가 빠르다는 것을 증명했다.

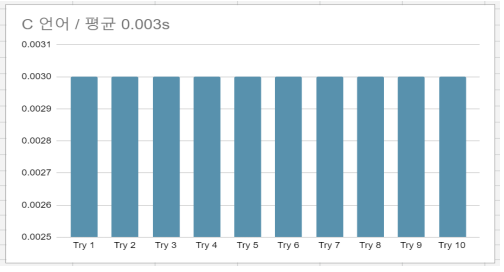


그림 5. C 언어 결과

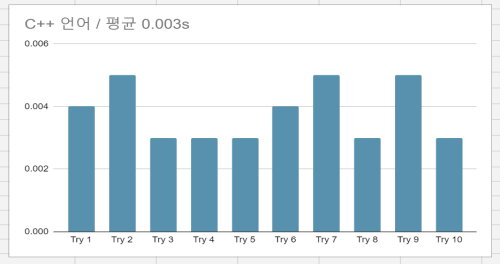


그림 6. C++ 언어 결과

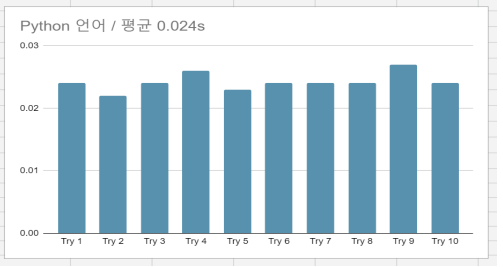


그림 7. Python 언어 결과

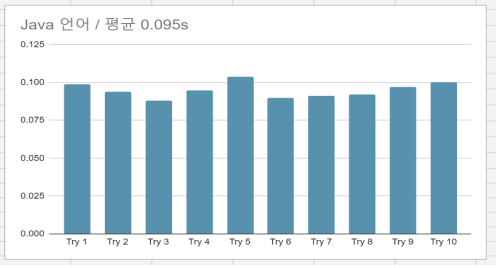


그림 8. Java 언어 결과

3. 범용 웹 서버와 데이터베이스(Mysql) 보안 취약점 분석

웹서버와 데이터베이스의 취약점이 발생하는 이유는 바로 프로그램 소스 코드가 취약하기 때문이다. 즉, 프로그램의 특정 코드에서 무언가를 할 수 있기 때문에 발생하는 것이다. 보통 [그림 9]와 [그림 10] 같은 방식으로 공격을 진행하는데, 이 간단한 코드 뒤에는 해커들의 수많은 노력이 숨어있다. 이것을 발견하기 위해 수많은 동적 분석과 정적분석을 오가며 분석을 했을 것이다. 여기서 알 수 있는 것은 프로그램의 소스 코드가 공개되어있다는 것이다. 따라서 보안을 중요시한다면, 소스 코드를 공개해서는 안 된다. 더 나아가 다른 범용 웹서버의 에러페이지를 통해 서버가 다른 범용 웹 서버를 사용하고 있다고 속이는 것도 하나의 방법이다.

```
Content-Type: %{(#kjy='multipart/form-data').
(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?
(#_memberAccess=#dm):((#container=#context
['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.getInstance
(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().clear()).
(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).(#cmd='whoami').
(#iswin=(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=(#iswin?
{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new java.lang.ProcessBuilder(#cmds)).
(#p.redirectErrorStream(true)).(#process=#p.start()).(#ros=
(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).
(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).(#ros.flush())}
```

그림 9. CVE-2017-5638 Apache Struts2 취약점 공격 데이터[3]

```
select * from information_schema.tables ₩
procedure analyse((select*from(select 1)x),1);
```

그림 10. CVE-2015-4870 MySQL DoS 취약점 공격 쿼리[4]

Ⅲ. 새로운 특수 목적용 보안 웹서버 설계 및 구현 제안

1. HTTP 통신 처리 구조 보안 설계 및 구현

각 소켓 함수를 통해 기본적인 ESTABLISHED를 맺은 후 read와 write 영역에 대해서 설계를 진행했다. 일단 [그림 11]에서는 read 함수를 통해서 값이 들어오면 공백을 기준으로 문자열을 자른다. 이 코드에 목적은 메소드를 변수에 저장하려는 것이다. 따라서 strcut_arr[0]에는 HTTP 메소드가 저장된다.

[그림 12]에서는 strcmp 함수로 메소드를 확인했다. 일단 strlen 함수를 사용해서 문자열 길이를 구한 다음 문자열 길이 만큼 반복하면서 문자들을 전부 아스키코드를 변환하고 그것을 더했다. 이제 모든 요청은 아스키코드의 합산 값으로 구분된다. 예를 들어 /index.html 이 요청이 왔다면, 해당 문자열은 1066이라는 숫자로 구분된다.

```
read ( cs, buff_rcv, 1024);
strcut0 = strtok(buff_rcv, " ");

for( strcut_i = 0 ; strcut_i <= 1 ; strcut_i++ ){

   strcpy(strcut_arr[strcut_i],strcut0);

   strcut0 = strtok(NULL, " ");

}
```

그림 11. 소켓 서버 구조 V1

```
if ( strcmp(strcut_arr[0],"GET") == 0 ){

   status_code = 0;

   strcut2_len = strlen(strcut_arr[1]);

   for( strcut2_for = 0 ; strcut2_for <= strcut2_len ; strcut2_for++ ){

      strcut2_add += strcut_arr[1][strcut2_for];

   }
```

그림 12. 소켓 서버 구조 V2

그리고 [그림 13]에서는 switch를 통해 정해진 case에 들어가면, strcmp 로 한 번 더 확인했다. 즉, 이중으로 확인하는 코드를 개발한 것이다. 만약 해당 코드를 뚫기 위해서 동일한 아스키코드 합이 나왔다고 하더라도, strcmp 함수로 인해 차단이 확실하게 되는 것이다.

```
switch(strcut2_add){
   case 47:
   {
      if ( strcmp(strcut_arr[1], "/") == 0 ){

         FILE * f = fopen("index.html", "r");
```

그림 13. 소켓 서버 구조 V3

2. HTTP 응답 속도 최적화 설계 및 구현

기존 웹 서버 프로그램 중 Apache에서는 stat 함수를 통해 index.html 파일 유/무를 확인하고, index.html 파일을 읽어오는 것을 확인할 수 있다. 또한 read 함수로 목적지 주소를 한 번 더 읽어 와서 해당 주소로 wrivev 함수를 통해 여러 개의 배열에 저장된 값들을 전부 가져와 합쳐서 전송하는 것을 확인했다. 따라서 본 논문에서는 웹 서버를 실행하기 전에 HTML, JS, CSS, JPG, PNG 등 모든 콘텐츠에 헤더를 붙이면 좋을 것 같다는 생각이 들었다. 또한 파일 확장자에 맞는 Content type을 넣었다. [그림 16]에서 보는 index.html 은 우리가 아는 html 파일이 아니다. HTTP 응답 헤더가 합쳐진 파일이다. 그 결과, 웹 서버 프로그램을 메모리에 올리기 전에 미리 파일에 맞는 헤더를 붙여 놓으면, 웹 서버는 요청이 들어와서 응답을 줘야 할 때 해당 파일만 주면 된다. 기존에 Apache 프로그램처럼 writev 함수를 사용해서 매번 붙일 필요가 없다는 것이다.

3. 데이터베이스 (NO-SQL, SQL) 통신 처리 구조 보안 설계

웹서버를 운영하면서 데이터베이스는 절 때 유출되면 안 되는 것으로 우리는 인지하고 있다. 여기서 데이터베이스는 어떤 개념으로 운영해야 하는지 고민이 많았지만 다시 한번 확인해보니 우리가 C 소켓 보안 웹서버 프로그램을 만든 것에 답이 나와 있었다. 본 논문

에서 C 소켓 보안 웹 서버를 만든 이유가 바로 기존 프로그램들이 유명하거나 모든 HTTP 통신을 제어하지 못해서였다. 가장 확실한 이유는 유명하기 때문이라고 이야기할 수 있다.

데이터베이스 종류에는 No-SQL과 SQL이 존재한다. No-SQL은 간단하게 말해서 텍스트에서 구분자를 통해 데이터를 가져온다고 생각하면 된다. 흔히 SIEM 같은 장비에서 SYSLOG를 파싱할 때와 비슷하다고 보면 된다. SQL은 현재 많이 사용되고 있는 데이터베이스 프로그래밍 언어다. 하지만 이런 언어들을 사용해서 DB를 구축한다면 약간의 리스크는 감수해야 한다.

일단 MYSQL 프로그램[5]에서 로그인할 때 아이디와 패스워드를 질의하는 쿼리문(select id, pw from shere id="user0" and pw="passwod0";)의 결과는 정해진 올바른 값만 나오게 된다. 하지만 여러분들도 알고 있듯이 SQL 취약점을 이용한 쿼리를 질의하면 DB의 테이블들을 해킹할 수 있다.

[그림 14]은 SQL 취약점을 이용한 쿼리 (select id,pw from shere id="user0" and pw='p'or'1'='1'--;)를 질의했을 때 결과이다. 즉, 해당 취약점을 이용해서 정보를 탈취할 수 있는 것이다. 오히려 이런 것 때문에라도 No-SQL이 훨씬 좋아 보였던 것은 사실이다.

```
+-------+---------+
| id    | pw      |
+-------+---------+
| user0 | NULL    |
| user0 | passwd0 |
+-------+---------+
2 rows in set (0.00 sec)
```

그림 14. 취약한 SQL 쿼리 질의 결과

하지만 조금 더 공부해본 결과 SQL과 No-SQL의 장단점이 있었고, 누가 더 좋다 나쁘다가 아닌 A 환경에서는 SQL 사용이 올바르며, B 환경에서는 No-SQL 사용이 올바르다는 것이다.

이미 본 논문에서는 C 소켓 보안 웹 서버를 개발했다. 그래서 HTTP 요청부터 처리 그리고 응답까지 전부 제어할 수 있다. 따라서 개발만 잘하면, DB 에러 또는 취약점을 이용한 쿼리의 응답을 노출 시키지 않을 수가 있다는 것이다.

따라서 C 소켓 서버에서 질의할 때 SQL 쿼리를 질의하는 것이 아니라, 내부에서 사용하는 특정키를 질의하는 것이다. 특정키는 FLAG가 될 수도 있고, 아니면 패스워드의 해시값이 될 수도 있다. 따라서 이때 SQL을 사용하고 있다면, [그림 15]에 있는 DB 관리 프로그램 또는 함수에서 해당 특정키를 해석해서 SQL 질의를 하는 것이다.

그림 15. 데이터베이스 보안 개념도

즉, 이렇게 된다면, 어떤 사용자라 할지라도 DB 쿼리문에 직접적으로 접근이 불가능하다. 그 이유는 해당 서버가 어떤 DB를 사용하고 있는지 모르기 때문이다. 따라서 보안성이 향상 됐다고 이야기 할 수 있다. 윗글과 공통점을 찾아보자면 계속해서 이중으로 확인하는 구조를 만들고 있다는 것이다. 처음에 모든 URL에 대해서도 이중으로 확인하는 구조를 만들었었고, 지금 데이터베이스 질의 부분에서도 사용자의 요청에 대해 바로 SQL 질의하는 것이 아니라 한 번 더 거쳐서 내부 프로그램이나 함수가 대신 질의하는 것이다.

이렇게 되면 정규식이 필요 없어진다. 아니다, 아예 할 필요가 없다. 특정키가 들어왔을 때 DB 관리 프로그램이나 함수에서 해당키가 있는지 질의해서 확인할 것이다. 이때 질의하고 나오는 결과도 한 번 더 확인해서 관리하면 완벽에 가까운 보안이 아닐까 싶다. 더 나아가 C 소켓 보안 웹서버에서도 한 번 더 응답 값을 거르거나 확인했다. 확실히 기존에 범용성이 좋은 프로그램을 사용할 때는 이런 퍼포먼스를 기대하지 못했지만, 직접 전부 개발을 하니, 이런 세세한 부분까지도 제어를 할 수 있어서 완벽한 보안에 한 발짝 더 다가갈 수 있는 것 같다.

따라서 No-SQL과 SQL 둘 중에 누가 더 좋은지 구분하지 말고, 각자 환경에 맞는 DB 구조를 사용하는 대신, 그 질의하고 응답할 때를 윗글처럼 한 번 더 확인하는 구조를 만들어버리면 DB도 자유롭게 사용하면서 보

안은 보안대로 신경 쓸 수 있게 되는 것이다.

4. 보안 웹 서버 개발 자동화 프로그램

본 논문에서는 모든 URL에 대해서 아스키코드 합을 구하고, 그것에 맞는 문자열 비교를 if 문으로 한 번 더 했다고 이야기했었다. 만약 사이트 규모가 거대하다면, 그 모든 URL에 대해서 직접 개발하기가 힘들고 시간이 오래 걸릴 것이다. 또한 다른 문자열인데 아스키코드 합의 값이 같은 경우, case 안에 하나의 if 문이 아닌 else if를 추가해줘야 한다. 이것을 사람이 한다면 인력 낭비일 것이다. 따라서 본 논문에서는 그것을 자동으로 개발해주는 보안 웹서버 개발 자동화 프로그램을 소개했다.

Apache, PHP 그리고 이번 논문에서 제안한 방안으로 개발한 해당 프로그램에서 C로 작성된 소켓 서버를 3개의 구성으로 구분했다. 처음부터 switch 문 전까지 “SERVER TOP”, switch 문을 부분을 “SERVER MIDDLE” 그리고 switch 문이 끝나고 마지막까지를 “SERVER BOTTOM”이라고 구분했다. 이렇게 나눴을 때 계산해서 소스 코드를 채워 넣어야 하는 핵심 부분은 바로 SERVER MIDDLE이라는 것이다. 또한 서버에 올라가는 것은 C언어로 작성하되, 자동화 프로그램은 성능이 어느 정도만 나와 주면 되기 때문에, 강력한 기능을 자랑하는 Python을 [그림 16]처럼 사용했다.

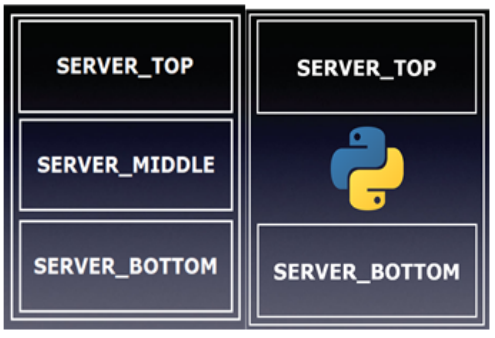


그림 16. 보안 웹 서버 구조 및 파이썬 활용

프로그램 구조를 간단하게 설명하자면, 먼저 파이썬에 os.walk 함수를 사용해서 경로, 폴더, 파일을 읽어온다. 마지막으로 중복 아스키코드 합의 값이 나왔을 때 case 안에서 if 문으로 한 번 더 구분을 해줘야 한다. 처음에는 if 문을 기본으로 진행했지만, 추후 else if를 넣어야 할 때 구조상 else if가 먼저 들어가서 if가 들어가는 상황이 만들어졌었다. 따라서 모든 if 문들을 처음에 else if로 설정하고, 추후 replace 함수를 사용해서 if 문으로 변경하는 동시에 카운트도 올리고, 프로그램에서 파악한 else if가 하나 더 있다면, 그 else if 문은 if 문으로 변경하지 않는다. 이러한 방식으로 자동화를 구현했다.

Ⅳ. 검증 및 비교 검토

1. 속도 및 보안 향상 검증 및 비교 분석 정리

Apache, PHP 그리고 본 논문에서 제안한 방안으로 개발한 C 소켓 웹서버 이 3가지를 비교해 보겠다. [그림 17]과 같이 Apache는 331줄에 3,015ms (54.502613 - 54.201109 = 0.301504), PHP는 579줄에 4920ms (22.268253 - 21.776196), C언어는 39줄에 408ms (32.244849 - 32.203986) 라는 결과가 나왔다. 똑같이 웹서버 처리부터 로그 파일 저장까지 하는데 성능이 더 좋게 나왔다.

본 논문에서 제안하는 방안을 기반으로 디렉터리 리스팅 공격에 적용을 시켜 비교해봤다.

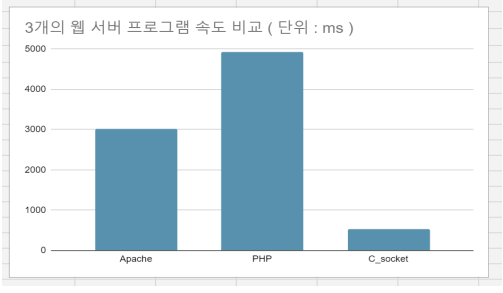


그림 17. 3개의 웹 서버 프로그램 속도 비교

일단 본 논문에서 제시한 방안으로 개발할 경우 디렉터리 공격을 수행했을 때 그 행위를 직접 소스코드로 확인하고 제어할 수 있다. [그림 18] 처럼 “/../../../eta/passed“ 라는 요청이 들어왔다면 해당 요청을 그대로 해석하는 것이 아닌, [그림 19]와 [그림

20] 처럼 1485라는 아스키코드 합산 값으로 변경하고 해당 값을 가지고 Switch 문을 통해 문자열을 한 번 더 검증한다. 왜냐하면, 문자열이 달라도 아스키코드 합산 값이 같을 수도 있기 때문이다. 그래서 이중 필터 방식을 전부 통과하지 못한다면 바로 404를 응답한다.

또한 소스코드에서 모든 요청에 대한 모든 응답 값이 각각 설정되어 있기 때문에 설정되지 않은 응답 값이 전달될 수는 없다. 따라서 함수 자체의 취약점이 발견됐다 하더라도 응답 값이 정해져 있기 때문에 서버에 대한 어떠한 값이나 데이터도 알아내지 못한다.

```
[root@localhost ~]# curl http://192.168.12
7.128/../../../etc/passwd
<html><head><title>404 Not Found</title></
head><body><center><h1>404 Not Found</h1><
/center><hr><center>nginx</center></body><
/html>[root@localhost ~]#
```

그림 18. 디렉토리 리스팅 공격 시도 및 결과

```
1485 /../../../etc/passwd
```

그림 19. 디렉토리 리스팅 공격
요청에 대한 아스키코드 합산 값

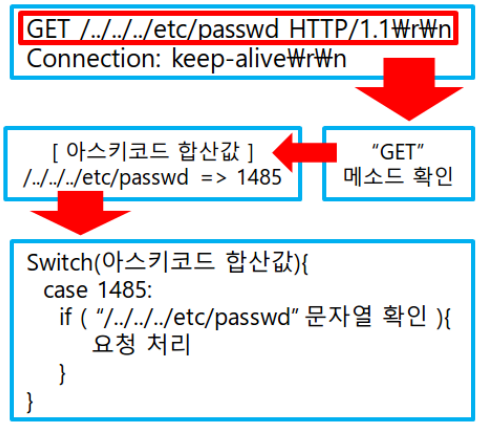


그림 20. 웹 요청 데이터에 대한 처리 프로세스

그에 반해 Apache에서는 Conf 파일에서 Options Indexed FollowSymLinks를 Options None으로 설정하라는 말 이외에는 내부 흐름을 사용자들이 파악하기 힘들다. 물론 Cong 파일 수정 시 공격에 대한 보안이 이루어진다. 하지만 내부가 어떻게 처리되고 더 나아가 직접 제어할 수 있는지에 대한 유/무는 보안 웹서버 제작에서 큰 차이다.

따라서 본 논문에서 제시한 방식은 한눈에 볼 수 있고, 어떤 요청 데이터를 처리하고 분석하고 응답하는 것까지 직접 제어가 가능하여서 보안 웹서버 개발에 있어 훨씬 효과적이다.

[표 1]처럼 범용 웹 서버 개발 방식은 하나의 목적을 두고 개발한 프로그램이 아녀서 보안이나 속도가 떨어질 수밖에 없다. 하지만 본 논문이 제안하는 방식은 각 사이트에 맞게 맞춤형으로 필요한 것만 메모리에 올리고, 모든 웹 통신 데이터를 직접 처리하기 때문에 속도와 보안 측면에서 범용 웹 서버보다 강력하다.

표 1. 속도 및 보안 향상 검증 및 비교 분석 정리 표

비교요소 \ 프로그램		기존 범용 웹 서버	본 논문 제안 웹 서버
속도	응답	응답시 마다 헤더와 데이터를 붙여서 응답(느림)	서버 실행 전 미리 헤더와 데이터를 붙여 둠(빠름)
	최적화	범용성을 위해 다양한 기능 추가(느림)	각 사이트에 맞춤형으로 웹 서버 제작(빠름)
보안	응답	서버 내부의 특정 데이터가 응답될 수 있음(불안전)	모든 요청에 대한 모든 응답이 미리 설정되어 있음(안전)
	취약점	설정값을 변경 및 패치 프로그램 실행 (불안전)	직접 작성한 소스코드에 의한 검증(안전)
	노출	오픈소스로 인해 시큐어 코딩을 해도 프로그램 자체의 취약점으로 공격 가능(불안전)	공개되지 않았고, 사이트별로 다른 구조의 소켓 웹 서버(안전)
	웹 방화벽	지속적인 업데이트 및 사용 권장(불안전)	소스코드가 웹 방화벽을 대신함(안전)

V. 결론

본 논문에서 제안한 웹 서버 프로그램은 들어오는 요청 데이터를 아스키코드로 변환하고, 합을 구한 뒤 해당 관련 코드에 스위치로 접근 후 다시 한 번 문자열을 확인하다. 더 나아가 해당 문자열에 대한 응답 값은 미리 정해져 있다. 또한 기타 규모가 큰 프로그램들과 다르게 간단한 구조로 이루어져 있기 때문에 취약점 발생 시 분석과 보완이 간단하다. 즉, 웹 요청, 처리, 응답 과정을 하나의 코드 파일로 확인할 수 있다. 이로써 취약점에 안전하다고 할 수 있다.

기존 웹 서버 프로그램들은 HTTP/1.1 기준 HTTP

응답 헤더를 일일이 붙여서 응답했다. 이것을 보완하기 위해 해당 프로그램에서는 일일이 헤더를 보낼 필요 없이 서버 실행 전, 확장자에 맞는 Content-type을 미리 파일에 붙여넣어 둔다. 즉, 웹 요청이 들어왔을 때 헤더를 붙일 필요 없이, 파일 하나만 보내면 된다는 것이다.

더 나아가 해당 개념과 자동화 프로그램이 하나로 된 프로그램을 현재 보안 시장에 도입된다면, 일반 사용자들도 아주 쉽고 성능이 빠르고 보안에 강력한 웹 서버 프로그램을 운영할 수 있고, 기업에서는 "../"(디렉터리 리스팅) ,"%3C"(XSS) 구문 등 일반적인 공격 문구에 관제 인력을 낭비하지 않고, 특정 문구가 들어왔을 때 서버 프로그램 상태를 확인하는 것처럼 훨씬 고도화된 관제를 할 것이다.

완벽해 보이는 이 논문에도 한계점이 존재한다. 모든 요청 URL을 처리하는 코드에서 if 문이 아닌 Switch로 처리하는 것은 정말 좋은 아이디어라고 생각한다. 분명 처리 속도가 if 문과 비교했을 때 큰 폭으로 향상된 것을 확인했었다. 또한 Switch 문이 일정 개수 이상이면 컴파일러가 자동으로 점프 테이블을 생성한다고 한다. 따라서 더 이상의 최적화 방법은 없을 것 같다. 하지만 이 부분에 대해서는 몇 년 정도 더 연구해보면, 훨씬 더 빠르게 최적화가 가능하리라 생각한다.

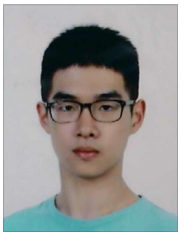
* 본 논문은 홍성락의 석사 학위논문을 바탕으로 작성되었다.

참 고 문 헌

[1] https://httpd.apache.org
[2] https://www.php.net
[3] https:/www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5638
[4] https:/www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-4870
[5] https://www.mysql.com
[6] 권희웅, 정형진, 곽후근, 김영종, 정규식, “멀티코어 시스템에서 흐름 수준 병렬처리에 기반한 리눅스 TCP/IP 스택의 성능 개선,” 정보처리학회논문지, Vol.16, No.2, pp.113-124, 2009.
[7] 김지훈, 최주호, “윈도우즈 소켓 기반 분산처리 기술을 이용한 항공기 엔진마운트의 최적설계,” 論文集, Vol.38, pp.29-38. 2000.
[8] 강동조, 박현주, “TCP/IP 소켓통신에서 대용량 스트링 데이터의 전송 속도를 높이기 위한 송수신 모델 설계 및 구현,” 한국정보통신학회논문지, Vol.17, No.4, pp. 885-892, 2013.

저 자 소 개

홍 성 락(Seong-Rak Hong) 준회원



- 2020년 2월 : 배재대학교 사이버보안과 졸업
- 2020년 3월 : 배재대학교 사이버보안학과 석사과정

〈관심분야〉 : 정보보호, 보안 SW개발, 네트워크 보안

조 인 준(In-June Jo) 정회원

- 1982년 2월 : 전남대학교 계산통계학과 졸업
- 1985년 2월 : 전남대학교 전자계산학과 석사
- 1999년 2월 : 아주대학교 컴퓨터공학과 박사
- 1983년 ~ 1993년 : 한국전자통신연구원 선임연구원
- 1994년 ~ 현재 : 배재대학교 사이버보안학과 교수

〈관심분야〉 : 정보보호, 컴퓨터네트워크보안, 전산조직응용