21-1-R04

웹 응용 프로그램의 보안 취약성 분석

(A Vulnerability Analysis of Web Application Programs)

김영민* · 안준선**

한국항공대학교 항공전자정보통신공학부* · 정보통신공학전공**

ymkim44@kau.ac.kr* · jsahn@kau.ac.kr**

요 약

현재 대부분의 웹 사이트 구축은 웹 응용 프로그램(web application program)이 적절한 웹 페이지를 생성하여 전송하는 형태인 동적 웹페이지(dynamic web pages)를 사용하고 있다. 이러한 웹 환경에서 전통적인 웹 서버 자체에 대한 공격 외에도 웹 응용 프로그램이 가지고 있는 취약점을 대상으로 한 공격이 증가하고 있다. 본 논문에서는 웹 사이트에 대한 공격을 사전에 방지하기 위하여 웹 응용 프로그램에 존재하는 취약점을 미리 찾아내기 위하여 요약 해석에 기반 한 프로그램 내의 안전한 자료 흐름(secure information flow) 분석을 제시하였다. 개발한 요약 해석은 문자열 자료값들의 멱집합(powerset)을 근사하는 요약된 자료 공간 내에서 프로그램을 수행함으로써, PHP 프로그램의 특징인 복잡한 가변 변수(variable variables)와 문자열을 첨자(index)로 사용하는 배열 등을 적절히 처리할 수 있다. 주어진 분석을 통하여 프로그램 내의 민감한 부분에서 사용되는 값에 외부의 입력이 영향을 미칠 수 있는지 여부와 사용되는 문자열 값의 종류를 알아낼 수 있다.

1. 서 론

최근의 웹 사이트 개발에 있어서는 웹 응용 프로그램이 현재 상황이나 입력에 맞는 웹 페이지를 생성하여 전송하는 동적인 웹페이지가 대부분을 차지하고 있다. JSP, ASP, PHP, Servlet 등의 웹 응용 프로그램은 웹 페이지에 접근하는 사용자의 입력을 받아 데이터베이스에 저장된 최신의 자료로부터 요구에 맞는 웹 페이지를 생성하여 클라이언트에게 전송한다. 따라서 웹 어플리케이션은 웹이라는 열린 통로를 통하여 수행이 시작되고 내부의 데이터에 접근하는 그 수행 특성으로 인하여 불가피하게 불특정 다수의 공격에 노출된다. 이러한 환경에서 최근 웹 서버에 대한 공격은 전통적인 웹 서버 자체에 대한 공격보다는 웹 응용 프로그램이 가지고 있는 취약점을 대상으로 공격이 이루어지고 있다. 즉, 바이러스 방지(anti-virus) 프로그램이나 네트워크 방화벽 같은 현재의 기술들은 컴퓨터 시스템과 네트워크 수준에서 높은 수준의 보안을 제공하고 있는데 반해 웹 응용 프로그램의 수준에서는 그렇지 못한 이유로, 웹 응용 프로그램의 열린 통로는 점점 공격의 대상이 되어 가고 있다[1][2].

현재 웹 응용 프로그램의 보안을 위해서는 다음과 같은 다양한 접근법이 시도되고 있다. Scott 와 Sharp는 응용 프로그램에게 유효하지 않고 악의적인 입력을 필터링 하는 게이트웨이(gateway)의 사용을 제안하였고[3][4], 이러한 방식은 AppSheild[5]와 InterDo[6]과 같이 계층 7에서 응용 프로그램 데이터에 대한 검사를 수행하는 웹 응용 프로그램 게이트웨이 방식의 상용제품에 적용되었다. 그러나 이러한 게이트웨이 방식은 사용자의 세밀한 환경설정이 요구되며, 보안상의 안정성이 이러한 환경설정에 전적으로 의존되어 진다. 또한 게이트웨이는 웹 응용 프로그램의 전단부에서 해당 웹 응용 프로그램에 대한 모든 HTTP 요청을 검사하게 되므로 웹사이트의 성능이 저하되는 단점을 가지고 있다. 또 다른 방식으로, 가상의 공격을 통하여 웹 사이트의 보안 문제점을 찾아내는 방식이 있다. 이 방식은 일반적으로 블랙박스 방법(black-boxed method)을 사용하여 가상의 침투를 시도하는 방법으로 Huanget은 웹 응용 프로그램의 취약성을 식별해 내기 위해 블랙박스 검사(black-boxed testing)를 제공하는 웹 응용 프로그램 보안 평가 구조를 디자인하였다[7]. 이 방식 또한 현재 상업적으로 사용되는 방법이기는 하나, 검사가 근본적으로 보안 취약성의 부재를 증명해 주지는 못하는 한계점을 가진다.

정적 분석에 기반 한 웹 응용 프로그램 분석 방법은 프로그램 내의 자료 흐름을 분석하여 이를 기반으로 외부의 입력에 의한 공격이 시스템의 안전성을 침해할 수 있는지 검사하는 방법을 사용한다. 이러한 정적 분석 방법은 실행시간의 성능에 부담이 되지 않으면서 주어진 공격 방법에 대한 안전성을 보장할 수 있는 장점을 가진다. Minamide는 자바 문자열 분석[8]에 관한 연구를 기반으로 PHP 프로그램 내에서 사용되는 문자열의 형태를 문맥 자유 문법(context-free grammar)으로 근사하여 분석하는 정적 분석기를 개발하였다[9]. 또한 Yichen Xie과 Alex Aieken은 문자열을 문자열들의 순서화된 접합(concatenation)으로 기본 블록(basic block) 요약(block summary) 을 기반으로 블록 내(intrablock), 프로시저 내(intra-procedural), 프로시저 간(interprocedural)의 3단계로 분석을 하는 방법을 제안하였다[10].

본 논문에서는 웹 보안 취약성과 밀접한 관련을 가진 문자열 데이터의 형태를 근사할 수 있는 요약 공간(abstract domain)을 기반으로 프로그램 내의 각각의 수행문에서 사용되는 데이터의 형태와 메모리의 상태를 안전하게 알아낼 수 있는 정적 분석기를 설계하였다. 설계된 분석기는 [10]과 비교하여 모든 부분식의 문자열 값을 분석할 수 있으며 요약 공간의 복잡도를 적절히 조절함으로써 성능에 대한 요구 사항 내에서 최대한 정확한(less false positive) 분석을 안전하게(no false negative) 수행할 수 있는 특징을 가진다. 또한 PHP의 특징인 가변 변수를 적절히 처리할 수 있는 요약 수행 규칙을 정의하였다.

본 논문의 전체적인 구성은 다음과 같다. 2장에서는 배경 연구 내용으로서 본 논문의 분석기가 대상으로 하고 있는 웹 응용 프로그램 보안 취약성을 소개한다. 3장에서는 본 논문의 보안 취약성 정적 분석기의 설계를 위한 기본 아이디어를 설명하고, 분석을 위한 추상 문법(abstract syntax), 요약 공간(abstract domain)과 의미 규칙을 기술한다. 마지막으로 4장에서 결론을 맺는다.

2. 배경 연구

2.1. 웹 응용 프로그램 보안 취약성

OWASP(The Open Web Application Security Project)에서는 웹 응용 프로그램에서의 심각한 10대 취약성 발표하였다. 여기에는 버퍼 넘침(buffer overflow)와 같은 프로그램의 문제점에서부터 시작하여 서비스 방해 공격과 같은 시스템 차원의 취약점까지 다양한 웹 응용의 취약성을 포함하고 있다. 이러한 취약점 중, 본 논문에서는 XSS 공격(cross-site scripting)과 SQL 삽입 공격(SQL injection)을 주된 분석 대상으로 한다.

2.1.1. SQL 삽입공격(SQL injection)

SQL 삽입공격은 사용자의 입력 값으로 들어오는 신뢰할 수 없는 값들이 별다른 검증 없이 SQL 명령문을 생성하는데 사용되어질 때 발생할 수 있다. 즉, 데이터베이스에 대한 질의(query)들이 동적으로 생성되는 상황에서, 사용자의 입력 값이 데이터베이스에서는 의미 있는 명령문으로 취급되지만, 웹 응용 프로그램에서는 단지 입력 값이 단순한 문자열로 취급되는 상황으로 인하여 발생한다.

(그림 1)은 웹 응용 프로그램에서 볼 수 있는 SQL 삽입공격 취약성의 예이다. (그림 1)의 코드는 사용자로 하여금 입력 값을 받아 데이타베이스에 대한 질의를 생성한다. 이때, 악의적인 사용자가 key값으로 아래와 같은 값을 입력으로 주면

```
' ; DROP TABLE ( ' contents
```

```
$key=$_GET['key'];
$query="SELECT * FROM page
WHERE key='" . $key . " '";
mysql_query($query);
```

(그림 1) SQL 삽입공격 취약성의 예

(그림 2)와 같은 SQL문을 $query값으로 생성하게 된다.

```
SELECT * FROM page
WHERE key=' ';
DROP TABLE('contents');
```

(그림 2) 생성된 SQL 명령문

(그림 2)의 SQL 명령문이 실행이 되면 contents 테이블은 삭제가 되며, 이는 프로그래머가 의도하지 않은 데이터베이스에 대한 동작이다. 위와 같은 기법의 웹 응용 프로그램에 대한 SQL 삽입공격 취약성은 실제로 많은 웹 사이트에서 발견되고 있다.

2.1.2. XSS 공격(Cross-Site Scripting)

XSS 공격은 사용자의 입력으로부터 온 데이터가 악의적인 스크립트의 수행에 사용되어 HTTP 쿠키와 같은 정보가 유출되는 결과로 이어지는 것을 말한다.

```
$key=$_GET['key'];
echo "<a href=\"result_key
.php?keywds=$key\">";
```

(그림 3) XSS에 대한 취약성 예

(그림 3)은 어느 웹페이지에서나 볼 수 있을 법한 XSS 취약성의 예제이다. $key 변수에 대한 값은 HTTP 요청으로부터 오며, 그 값은 HTML의 출력을 생성하는데 사용된다. 이때 XSS 공격 취약성을 이용하여 URL에 대한 공격을 시도하면 아래와 같은 공격의 형태가 될 것이다.

http://www.reserch.com/find_key.php?key=><script>malicious_sript();</script>

위와 같은 HTTP 요청을 공격 대상자가

```
<a href="result_key.php?
keywds=>
<script>malicious_script();</script>
```

(그림 4) 생성된 HTML 출력문

수행하게 될 경우 (그림 3)의 PHP 코드는 (그림 4)와 같은 HTML문을 공격 대상자의 웹 브라우저에서 수행하게 되며, 이러한 과정에서 공격자가 임의로 작성할 수 있는 위험한 스크립트가 공격 대상자의 권한으로 수행될 수 있게 된다.

XSS의 공격 기법을 사용하면, 공격 대상자의 웹 브라우저는 공격자에 의해 준비된 악성 스크립트가 신뢰할 수 있는 사이트로부터 전송되었다고 생각하기 때문에, 악성 스크립트는 해당 사이트에서 사용하는 브라우저 쿠키, 세션 토큰 혹은 다른 민감한 정보에 접근할 수 있다. 이 스크립트는 심지어 HTML 페이지의 내용을 조작하여 변경할 수도 있다.

SQL 삽입공격 이나 XSS 공격은 모두 정보 흐름의 안전성과 관련이 있다. 즉 안전하지 않은 외부로부터의 입력 정보가 스크립트의 수행이나 SQL 명령문의 수행과 같은 민감한 작업에 사용되었을 때 웹 응용 프로그램은 취약성을 노출하게 된다. 따라서 이러한 정보의 흐름을 분석함으로써, 취약성이 존재할 소지가 있는지를 분석할 수 있다. 이를 위하여 본 논문에서는 정적 분석 방법인 안전한 자료 흐름 분석 방법을 사용하여 프로그램 내의 정보의 흐름을 분석하고자 하였다.

3. PHP 보안취약성 정적 분석기의 설계

3.1. PHP 프로그램의 보안 취약성의 검출

본 논문에서는 PHP 프로그램에서 SQL 삽입공격과 XSS 공격 보안 취약성을 검출하는 분석기를 설계하였다. 2절의 예제를 보면 안전에 민감한 작업을 위한 입력으로 외부의 검증되지 않은 입력이 취약점 제거 과정을 거치지 않고 웹 페이지의 출력과 같은 민감한 작업에 사용되는 경우에 보안 취약성이 발생할 수 있음을 알 수 있다. PHP는 모든 초기화 되지 않은 변수의 값은 웹을 통한 외부 입력에 의하여 설정될 수 있으므로, 이러한 입력은 악의적인 값을 가질 수 있음을 가정하여야 한다. 외부의 악의적인 입력은 직접적으로 또는 함수 인자, 함수 호출 결과, 지정문을 통하여 간접적으로 민감한 자료값에 영향을 줄 수 있으므로, 프로그램 내의 자료의 흐름에 대한 분석이 필요하다.

그런데, PHP 언어는 일반적인 프로그래밍언어와는 다르게 자료 흐름 분석에 있어 다음과 같은 어려운 점을 가진다.

- 가변 변수(variable variables) : PHP는 문자열 자료값이 변수 명으로 사용될 수 있어 변수의 문자열 값이 자료 흐름에 영향을 미친다.
- 배열의 첨자(index)가 문자열 데이터를 자유롭게 가질 수 있으므로, 배열에 의한 자료 흐름 분석을 위해서는 첨자로 사용되는 문자열 값에 대한 정확한 분석이 필요하다.
- 변수의 타입이 정해져 있지 않아 다양한 타입의 값을 동적으로 가질 수 있다. 따라서 문자열과 숫자가 상호 호환되어 사용되는 경우가 빈번하며, 이러한 경우에 대한 고려가 분석의 정확도를 높일 수 있다.

본 논문에서는 이러함 점을 고려하여, 프로그램 내에서 사용되는 문자열 자료의

종류를 정확히 분석하고자 하였다. 제시된 요약 해석 방법은 상수 전달 분석에서 사용되는 방법과 비슷한 방법으로 프로그램 내의 문자열 자료의 흐름을 분석하며, 문자열 값을 주소 공간으로 사용하여 프로그램 내의 자료 흐름을 분석하므로 위에서 제기한 문제를 적절히 극복할 수 있다.

3.2. PHP 추상 문법

(그림 5)는 PHP 프로그램의 주요 형태를 나타내는 추상 문법이다.

```
string ::= static_string | dynamic_string;
e ::= integer | boolean | string | constant | e = e | e + e | e >= e | ~e
   | variable | variable[e] | $variable | string{e}
statement ::= e | echo e* | for e* e* e* statement*
        | while e statement* | if e+ statement* ( else statement* )?
        | { statement* } ;
P ::= statement*;
```

(그림5) PHP에 대한 추상 문법(abstract syntax)

PHP의 자료형에서 문자열은 정적문자열(static string) 또는 동적 문자열(dynamic string)로 나눌 수 있는데 이는 작은따옴표(single quotes, ‘ ’), 와 큰따옴표(double quotes, “ ”)로 구분된다.

```
$name = “Guido” ;
echo “hi, $name\n” ;
echo “hi, $name’ ;
```

(그림 6) 정적 문자열(static string)과 동적 문자열(dynamic string)의 예

동적 문자열 내의 변수는 해당 값이 추출된다. 따라서 위 (그림 6)의 실행결과는 아래와 같다.

hi, Guido
hi, $name

```
$foo = 'bar';
$$foo = 'baz'; //$bar = 'baz';
```

(그림 7) 가변변수(Variable Variables)의 예

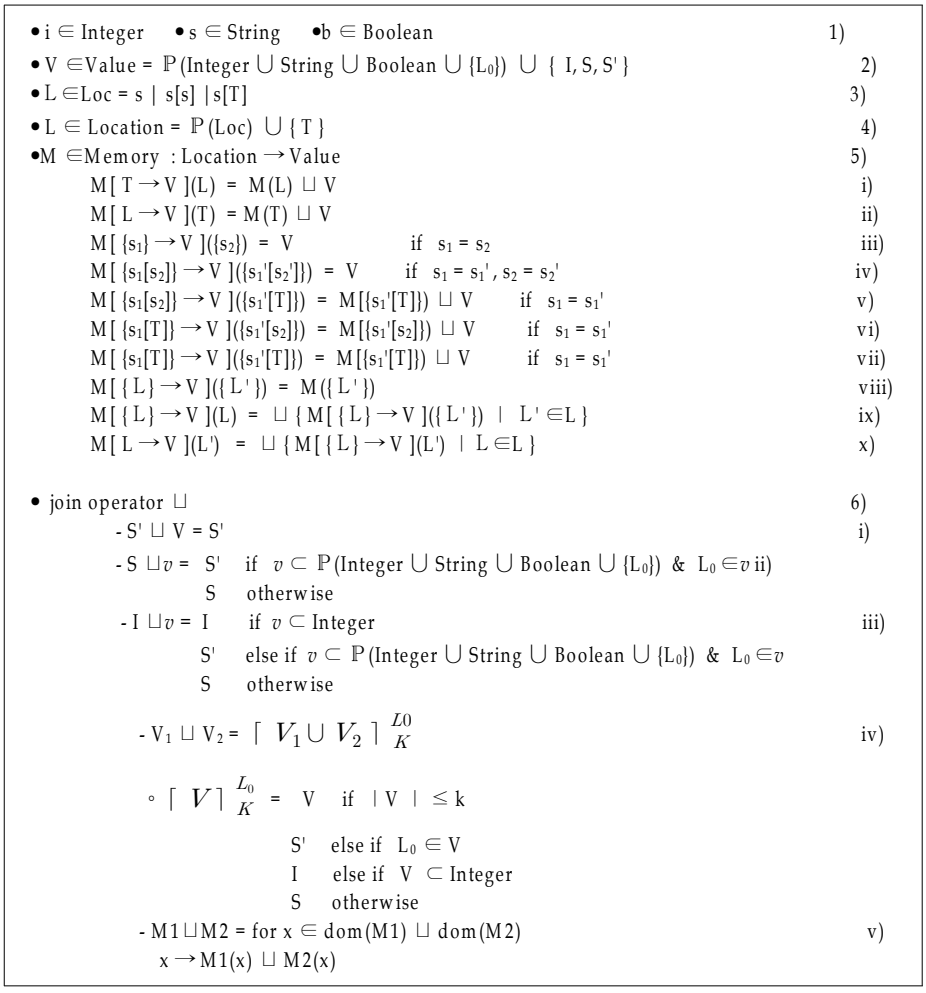
(그림 7)은 가변변수의 예로서 가변 변수는 문자열 변수값을 취해서 변수명으로 사용한다. *variable*[*e?*]는 스트링을 첨자로 사용하는 배열을 나타내며 이러한 문자열 첨바 배열이나 가변 변수에 의할 자료 흐름를 적절히 분석하기 위해서는 각 변수가 가질 수 있는 문자열 값을 분석할 수 있어야 한다.

3.3. Abstract Domain (Value Domain) 정의

본 논문에서는 PHP로 작성된 웹 응용프로그램을 요약 해석으로 분석하기 위해서 (그림 8)과 같은 자료값 공간을 정의하였다.

1) 의 i, s, b은 각각 PHP의 정수, 문자열, 논리값(boolean)의 원소를 나타낸다.

2) 는 래티스(lattice) 공간의 값으로 PHP에서 사용되어지는 모든 기본 데이터 값을 표현하고 있다. 값(Value)은 집합으로 표현되어지는데 특정 프로그램 부분에서 표현식의 계산 결과가 가질 수 있는 가능한 모든 값들을 포함하기 위함이다. 이때 L0는 초기값으로서 모든 변수는 처음에 해당 변수명의 초기값을 갖는다. PHP는 변수의 선언 없이 바로 사용이 가능하고, 이는 HTTP 요청으로부터 그 값을 가져오므로 만약 민감한 작업을 위한 입력값에 초기값(L0)이 영향을 미치는 값이 사용된다면 이는 보안 취약성이 있을 수 있음을 의미한다. 프로그램 수행 중에 집합은 무한한 값을 가질 수 있는데 이를 유한하게 근사하기 위해 I, S, S'값을 사용한다. I는 어떠한 정수값도 될 수 있음을 나타내고 이는 정



(그림 8) 요약 해석을 위한 정보 공간 정의

수 집합의 상위 집합(superset)이다. S는 어떠한 문자열도 될 수 있음을 나타내고 초기값(L0)을 포함하지 않는 다른 문자열 집합과 정수 집합의 상위 집합이 된다. 초기값을 포함하지 않으므로 S는 정적 분석 시 안전한 값으로 분류되어 진다. S'는 모든 값의 상위 집합으로 초기값(L0)을 포함한 어떠한 문자열이 될 수 있음을 나타낸다. 일반적으로 PHP에서 정수는 문자열의 특수한 경우로 간주되므로 문자열 집합은 정수 집합의 상위집합이 된다. 요약 공간을 가지고 안전한 자료 흐름 분석을 수행하고 수행 후 결과에서 민감한 자료 값이 L0를 포함한 문자열 집합이나 S'을 가지면 이를 보안상 취약한 부분으로 분석한다.

3) 의 L는 Loc의 원소로 메모리 내의 위치를 표현하는 위치(Location) 공간을 만드는데 사용되어진다. s 와 s[s]는 프로그램의 변수(예, $x) 이름이나 배열(예, $x[$key]) 이름과 첨자(index)의 문자열을 나타낸다. s[T]는 정적 분석 시간에 배열의 첨자를 결정할 수 없는 경우를 나타낸다. 즉, PHP 프로그램 내에서 $array[$key]가 있을 경우, $key의 값을 정적 분석 시간에 결정할

수 없다면 이는 s[T]의 형태로 표현된다.

4) 의 L은 메모리 내에서 위치를 표현하는 위치(Location)의 원소로써 Loc의 멱집합(powerset)과 정적 분석 시간에 위치를 결정할 수 없음을 나타내는 T로 표현되어진다.

5) 는 특정 시점의 메모리 상태를 나타내며 주어진 위치(Location)에 대해 해당위치의 값(Value)를 반환한다. 위치에 대한 값 반환 규칙은 i)~x)에서와 같이 나타낼 수 있으며 위로부터 우선순위를 갖는다. 불확실한 메모리 위치(T, s[T] 등)의 갱신에 대하여 관련 가능한 위치의 값은 갱신 이전의 값과 불확실한 위치의 값의 조인으로 얻어진다.

6) 은 값 공간 V와 메모리 공간 M의 래티스 공간에 대한 조인(join) 관계를 정의하고 있다. 정의되는 값 공간은 무한한 크기를 가지나, 값 공간 내의 원소의 개수가 미리 정해진 한도(k)를 넘을 경우에 S', S 또는 I로 표현함으로써 공간을 유한하게 근사시킬 수 있다. 메모리 공간의 경우 정의구역인 위치(Location)를 유한하게 근사시킬 수 있으므로, 역시 유한한 공간이 된다. i)~iv)는 i)부터 우선순위를 갖는다.

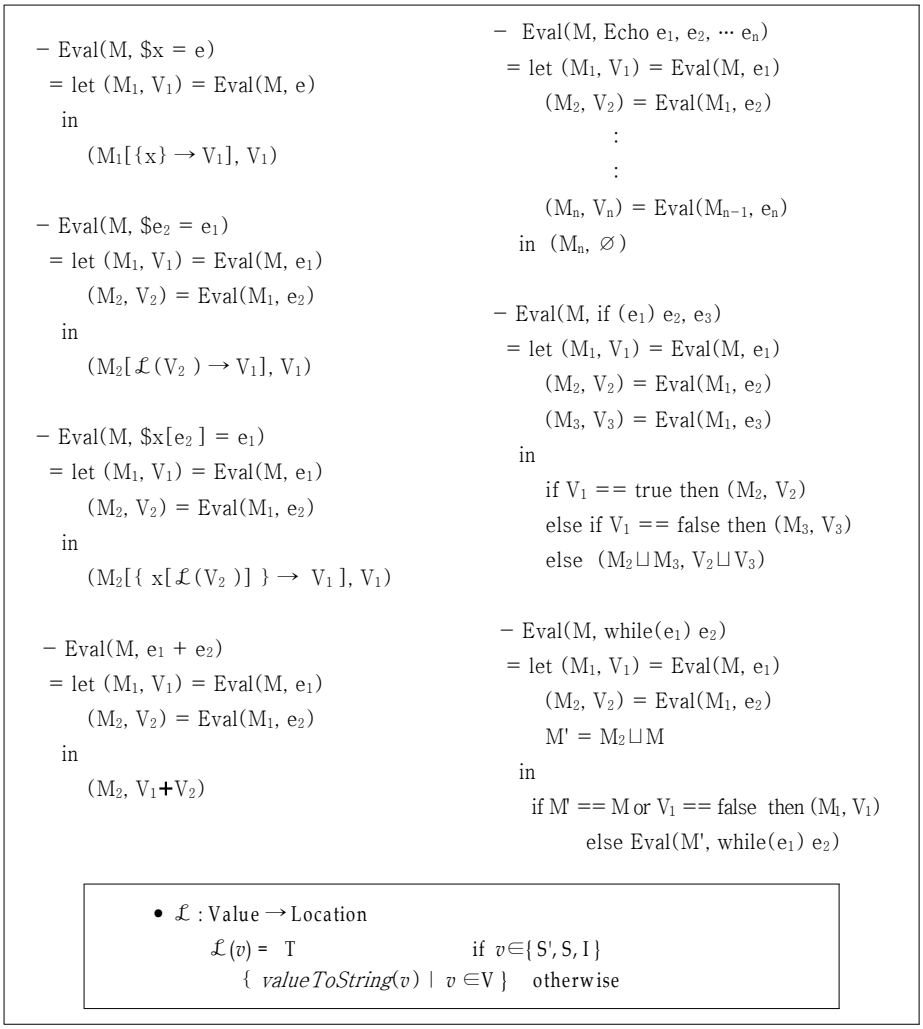
i)의 경우 S'는 L0과 임의의 문자열을 모두 포함하므로 다른 값과 조인하면 S'이 된다. ii)의 경우 S는 문자열의 최상위 원소(top)와의 조인 관계를 나타내는 것으로써 L0를 포함한 값 집합과의 조인 대해서는 S'를 결과로 생성한다. 그 외의 경우에는 S가 된다. iii)의 I의 경우 정수의 최상위 원소를 나타내므로 정수 집합과의 조인은 역시 I가 되며, ii)에서와 마찬가지로 L0를 포함한 값 집합과의 조인 대해서는 S'를 결과로 생성한다. 그 외의 경우에 대해서는 S를 조인 결과로 생성한다. iv)는 S', S, I를 포함하지 않는 그 외의 값 집합들(V1, V2)에 대한 조인 연산으로 두 값 집합을 합집합한 집합의 원소의 수가 일정 한도(k)를 넘지 않을 경우 그 두 집합의 합집합을 결과로 생성하게 된다. 만약 합집합의 원소의 수가 일정 한도(k)를 넘는 경우 L0를 원소로 포함하고 있으면 S'을 조인 결과로 생성하며, 합집합의 결과가 정수의 부분 집합이면 I를 결과로 생성하게 된다. 합집합의 결과가 일정 한도(k)를 넘고 정수의 부분집합이 아니면서 L0를 원소로 갖지 않는 경우에는 S를 조인 연산의 결과로 생성하게 된다. v)에서 두 메모리 상태의 조인은 같은 메모리 위치에 대한 각각의 메모리의 저장된 값을 조인한 결과를 돌려주는 메모리 상태를 결과로 갖는다.

3.4 실행(interpretation) 규칙

요약 해석 방법은 실제 프로그램을 실행하는 것과 비슷한 과정을 통해서 분석을 수행한다. 따라서 앞에서 정의한 요약 공간을 가지고 PHP 문법에 대한 의미 정의가 필요하다. PHP 문법에 대한 요약된 의미를 정의하기 위한 실행 규칙 중 대표적인 것을 실행기반 의미구조(operational semantics)의 형태로 기술하면 (그림 9)와 같다.

Eval 규칙은 Eval : Expr × Memory → V × Memory의 타입을 가지며, 특정 표현식(exression)과 그 표현식이 수행되기 직전의 메모리 상태를 인자로 받아 그 식에 대한 계산 값과 계산 후의 메모리 상태를 생성한다.

$x = e은 대입(assign)문으로 대입문 수행전의 메모리 상태에서 e을 계산하여 생성된 V1값과 변수 x의 이름을 문자열로 하는 위치에 대한 V1의 값을 할당한 새로운 메모리 M1을 생성한다.



(그림 9) PHP 문법에 대한 실행 규칙

$e2 = e1은 가변 변수(Variable Vairables)에 대한 연산으로 위의 대입문에서 e2에 대한 계산이 추가된 것이다. 즉, e2에 대한 연산이 먼저 수행되고 e2 연산시 생성된 메모리 M2에 그 결과값(V2)을 위치로 하는 메모리 위치에 V1의 값을 할당한 새로운 메모리를 조인하여 결과 메모리를 생성한다. ℒ은 Value → Location을 가지며 값을 받아 S', S, I의 원소이면 T를 반환하며 그 외의 경우에는 그 값을 문자열로 캐스팅한 문자열의 집합을 반환한다.

$x[e2] = e1은 배열 표현식에 대한 연산으로 첨자(index)에 대한 연산이 추가된다. 즉, 첨자에 대한(e2) 연산 결과값(V2)을 위치 공간 값으로 계산하고 배열 이름 x와 위에서 계산된 첨자 위치를 가지고 표현한 메모리 위치에 대해 V1값을 할당한 새로운 메모리 M2를 생성한다.

e1+e2는 기본 연산(primitive operation)을 나타내는데, 먼저 e1을 계산하고 e2를 계산하므로 e1의 실행 후 메모리 상태 M1이 e2의 실행 전 메모리 상태가 되며, 기본

연산은 메모리 상태를 변화시키지 않으므로 e2 실행 후의 메모리 상태가 전체 식의 실행 후 메모리 상태가 되고 전체 식의 수행결과는 V1과 V2에 해당 연산자의 의미를 적용하여 생성된 값으로 한다.

Echo e1, e2, …, en은 각각의 부분식 e1, e2, …, en들을 순서대로 계산하므로 이전 표현식 계산 후 메모리 상태가 다음 표현식의 수행 전 메모리 상태가 되며 echo문은 단지 출력을 수행하므로 전체식의 계산 결과는 없다.

if문은 e1의 계산 결과에 따라 e2 또는 e3를 계산하므로 e1의 실행 후의 메모리 상태 M1이 e2와 e3의 실행 전 메모리 상태가 되며, e1을 계산한 결과가 true 또는 false인 경우에 따라 (M2, V2) 또는 (M3, V3)를 결과로 생성한다. 이때 e1의 계산 결과를 결정 할 수 없는 경우, 요약된 의미는 모든 가능한 경우를 포함하여야 하므로 e2와 e3의 계산 후의 메모리 상태 M2, M3와 계산 후의 값 V2, V3를 각각 조인 연산함으로써 안전(sound)한 결과를 생성한다.

while(e1) e2는 정적 분석에서의 모든 가능한 경우를 포함하기 위해 조건식 e1의 계산 결과로 생성된 메모리 상태 M1에서 e2를 수행하여 M2를 생성하며, 반복문(loop)의 순환에 의해 생기는 접점에서의 메모리 상태 M'은 M과 M2의 조인 연산의 결과로 나타낼 수 있다. 이때, 접점에서의 메모리 공간의 분석 결과가 더 이상 증가하지 않거나, 또는 e1의 계산 결과가 거짓(false)인 경우에는 전체 식의 계산 후 메모리 상태는 M1이 되며 계산 결과 값은 V1로 나타낼 수 있고, 그렇지 않은 경우에는 새로운 메모리 상태 M'에 대해 재귀적으로 정의되어 진다.

4. 결 론

웹 응용 프로그램 내의 취약성을 검출하기 위해서는 프로그램 내의 자료의 흐름을 파악하여 외부의 검증되지 않은 입력이 프로그램 내에서 어떻게 사용되는지를 분석하여야 한다. 이를 위하여 요약 해석에 기반한 안전한 자료 흐름 분석기를 설계하였다. 개발한 요약 해석은 문자열의 종류를 근사하는 요약된 자료 공간 내에서 프로그램을 실행하는 형식으로 분석을 수행하므로 PHP 프로그램의 복잡한 이명과 변수에 저장된 문자열 값이 변수명으로 사용되는 경우 등을 적절히 고려하면서 프로그램 내의 자료의 흐름과 사용되는 자료의 형태를 정확하게 분석할 수 있다.

향후 연구로는 프로시저 간 분석에 대한 설계가 필요하며, 위의 설계를 바탕으로 분석기를 실제적으로 구현하여 다양한 PHP 프로그램에 적용하면서 보안 취약성 검출의 효과를 분석하는 작업이 필요하다.

참 고 문 헌

[1] Curphey, M., Endler, D., Hau, W., Taylor, S., Smith, T., Russell, A., McKenna, G., Parke, R., McLaughlin, K., Tranter, N., Klien, A., Groves, D., By-Gad, I., Huseby, S., Eizner, M., McNamara, R. “A Guide to Building Secure Web Applications.” The Open Web Application Security Project, v.1.1.1, Sep 2002.

[2] D. Turner, S. Entwisle, “Symantec Internet Security Threat Report Vol.Ⅸ - Trends for July 05-December 05”, Symantec,

March 2006.

[3] Scott, D., Sharp, R. “Abstracting Application-Level Web Security.”, *Proc. 11th Int'l Conf. World Wide Web (WWW2002)*, pages 396-407, May 17-22, 2002.

[4] Scott, D., Sharp, R. “Developing Secure Web Applications.” *IEEE nternet Computing*, 6(6), 38-45, Nov 2002.

[5] Sanctum Inc. “AppShield 4.0 Whitepaper.” http://www.sanctuminc. com, 2002.

[6] Kavado, Inc. “InterDo Version 3.0” Kavado Whitepaper, 2003.

[7] Huang, Y. W., Huang, S. K., Lin, T. P., Tsai, C. H. “Web Application Security Assessment by Fault Injection and Behavior Monitoring.” *In Proc. 12th International World Wide Web Conference (WWW 2003)*, 148-159, May 21-25, 2003.

[8] Aske Simon Christensen, Anders Møller, and Michael I. Schwartzbach , “Precise Analysis of String Expressions”,14th *International Static Analysis Symposium (SAS '03)*, June. 2003.

[9] Minamide, Y. “Static Approximation of Dynamically Generated Web Pages”. *14th International World Wide Web Conference(WWW 2005)* May. 2005.

[10] Yichen, X., Alex, A. “Static Detection of Security Vulnerabilities in Scripting Languages”. *15th USENIX Security Symposium*, 2006.

김 영 민

2006년 2월 : 한국항공대학교 정보통신공학과 졸업

2006년 3월 ~ 현재 : 한국항공대학교 정보통신공학과 석사과정

관심분야 : 정보공학, 통신공학, 전산 공학

안 준 선

1992년 2월 : 서울대학교 계산통계학과 학사

1994년 2월 : KAIST 전산학과 석사

2000년 8월 : KAIST 전자전산학과 박사

2000년 9월~2001년 8월 : KAIST 프로그램분석시스템연구단 (ROPAS) 연구원

2001년 9월~현재 : 한국항공대학교 항공전자 및 정보통신 공학부 조교수

관심분야 : 프로그램분석, 유비쿼터스 프로그래밍 환경, 정보검색, 프로그램보안, 병렬화컴파일러, 함수형 언어 등