http://dx.doi.org/10.3745/KIPSTA.2011.18A.5.193

PHP 파일 삽입 취약성 검사를 위한 정적 분석기의 구현

안 준 선[†] · 임 성 채[††]

요 약

인터넷 상의 웹 응용 프로그램은 불특정 다수의 사용자가 접근할 수 있기 때문에 보안상의 위험이 가중된다. 특히, 응용 프로그램의 소스 코드에 보안 취약성이 있을 경우에는 침입 탐지 시스템과 같은 시스템 수준의 방어가 어렵기 때문에 이를 미리 제거하는 것이 중요하다. 본 논문에서는 웹 응용 프로그램의 대표적인 소스 코드 취약성인 PHP 파일 삽입 취약성을 자동으로 검출할 수 있는 정적 분석기의 구현에 대해 다룬다. 본 연구에서는 의미 기반의 정적 분석을 사용하여 소스 코드의 취약성을 미리 자동으로 검출하고 수정하도록 함으로써, 기존의 침입 테스트 기법이나 응용 프로그램 방화벽 사용과 다르게 보안 취약성을 안전하게 제거하면서 추가적인 실행 시간 부하를 피하고자 하였다. 이를 위하여 의미 기반 분석 방법인 요약 해석 방법론을 적용했으며, PHP 삽입 취약성에 최적화된 요약 분석 공간을 설계하여 사용함으로써 PHP의 특성인 복잡한 문자열 기반 자료 흐름을 효과적으로 처리하면서 목적으로 하는 취약성을 효과적으로 검출할 수 있었다. 프로그램의 취약성 분석 결과는 Java GUI 도구를 통해 확인할 수 있으며, 분석된 취약성 지점에서의 메모리 상태 및 계산 정보도 같은 도구를 사용해 확인할 수 있다. 구현된 분석기의 취약성 검출의 정확성과 실행 속도를 검증하기 위하여 공개된 PHP 프로그램을 사용하여 성능 실험을 수행하였으며, 이를 통해 구현된 분석기의 실용성을 확인하였다.

키워드 : PHP 파일 삽입 취약성, 정적 분석, 요약 해석, 보안 취약성

Implementation of a Static Analyzer for Detecting the PHP File Inclusion Vulnerabilities

Joonseon Ahn[†] · Seong Chae Lim[††]

ABSTRACT

Since web applications are accessed by anonymous users via web, more security risks are imposed on those applications. In particular, because security vulnerabilities caused by insecure source codes cannot be properly handled by the system-level security system such as the intrusion detection system, it is necessary to eliminate such problems in advance. In this paper, to enhance the security of web applications, we develop a static analyzer for detecting the well-known security vulnerability of PHP file inclusion vulnerability. Using a semantic based static analysis, our vulnerability analyzer guarantees the soundness of the vulnerability detection and imposes no runtime overhead, differently from the other approaches such as the penetration test method and the application firewall method. For this end, our analyzer adopts abstract interpretation framework and uses an abstract analysis domain designed for the detection of the target vulnerability in PHP programs. Thus, our analyzer can efficiently analyze complicated data-flow relations in PHP programs caused by extensive usage of string data. The analysis results can be browsed using a JAVA GUI tool and the memory states and variable values at vulnerable program points can also be checked. To show the correctness and practicability of our analyzer, we analyzed the source codes of open PHP applications using the analyzer. Our experimental results show that our analyzer has practical performance in analysis capability and execution time.

Keywords : PHP File Inclusion Vulnerability, Static Analysis, Abstract Interpretation, Security Vulnerability

1. 서 론

최근 인터넷에 기반한 컴퓨터 시스템의 증가로 웹 응용 프로그램의 사용이 확산되고 있다. PHP, JSP, ASP, Servlet 등으로 작성된 웹 응용 프로그램은 웹이라는 열린 통로로 데이터를 입력 받아 수행되므로 불특정 다수의 공격에 쉽게 노출될 수 있다[1]. 이런 이유로 전통적인 웹 서버나 운영체

† 정 회 원 : 한국항공대학교 항공전자및정보통신공학부 교수
†† 종신회원 : 동덕여자대학교 컴퓨터학과 부교수
논문접수 : 2011년 3월 21일
수 정 일 : 1차 2011년 7월 12일, 2차 2011년 7월 28일
심사완료 : 2011년 7월 28일

제에 대한 직접 공격보다는, 보다 손쉬운 웹 응용 프로그램의 보안 취약성을 이용하는 컴퓨터 시스템 공격이 증가하는 추세이다[2,3].

Gartener의 2006년 보고서에 따르면 소스 코드의 보안 취약성이 시스템의 보안 문제를 심각하게 야기하며, 이로 인한 기업 손실이 매우 크기 때문에 소스 코드의 보안 취약성에 대한 적극적인 대응이 매우 시급한 것으로 보고되었다[2]. 이런 소스 코드 취약성에 효과적으로 대처하기 위하여 취약성의 패턴과 이에 대한 공격 및 대응 방법에 대한 연구가 활발히 이루어지고 있으며, 미국 등 선진국에서는 여러 보안 취약성 사례를 정리하여 데이터베이스화하고, 이에 대한 대처 방법을 보급하는 정부 및 민간 차원의 활동도 수행되고 있다. 대표적인 소스 코드 취약성 관련 데이터베이스로 미국의 CWE(Common Weakness Enumeration)와 CVE(Common Vulnerabilities and Exposures) 등이 있으며 대표적인 소프트웨어 취약성을 선정한 결과로 OWASP Top 10과 CWE/SANS TOP 25 등이 발표되어 널리 참조되고 있다[4,5,6,7].

본 연구에서는 대표적인 웹 보안 취약성의 하나인 PHP 파일 삽입 보안 취약성을 해결하기 위한 방법에 대하여 다룬다. PHP 파일 삽입 보안 취약성은 PHP 프로그램의 파일 삽입 연산이 외부의 부적절한 입력을 허용함으로써 보안상 위협이 되는 프로그램을 수행하게 되는 취약성을 말한다. CWE/SANS Top 25에서도 그 중요성이 인식되어 단일 언어를 위한 취약성으로는 유일하게 포함되고 있다[7,8]. 현재 공개 소프트웨어로 제공되는 많은 PHP 프로그램들이 이러한 취약성을 가지고 있는 것으로 보고되고 있어 이에 따른 보안 침해의 소지가 존재하는 상황이다[4, 5].

본 논문에서는 PHP 웹 응용 프로그램의 PHP 파일 삽입 보안 취약성을 사전에 자동으로 검출할 수 있는 정적 분석기에 대해 설명한다. 구현된 분석기는 프로그램 수행 전에 사용자의 추가 설정 없이 프로그램 내의 취약성을 자동으로 발견 할 수 있어서, 웹 응용 프로그램 방화벽이나 침투 테스트(penetration test)의 방법에 비해 취약성 검출의 안전성이 높고 프로그램 실행시의 부담이 없다는 장점을 가진다. 구현된 정적 분석기를 사용하여 개발한 PHP 프로그램을 웹상에 공개하기 전에 보안 취약성을 미리 검출하여 수정할 수 있다.

구현된 취약성 분석기는 대표적인 의미 기반 분석 방법인 요약 해석(abstract interpretation) 방법론[15]에 기반하여 개발되었다. 요약 해석 기법을 사용함으로써 PHP의 특징인 문자열 첨자나 변수값 변수(variable variables) 등의 PHP의 확장된 문자열 자료값 사용에 따른 복잡한 자료 흐름을 효과적으로 분석할 수 있다. 또한 분석 목적에 맞게 설계된 요약 공간(abstract domain)을 사용함으로써 적절한 부담 내에서 분석에 요구되는 실용적인 정확성을 얻을 수 있었다. 분석의 결과는 Java GUI 도구로 볼 수 있으며, 이 도구를 통해 검출된 PHP 소스 코드의 보안 취약성 지점은 물론 특정 프로그램 지점에서의 변수들의 값 및 계산된 값을 확인할 수 있어 프로그램의 수정에 유용하다. 구현된 분석기의 성능을 알아보기 위하여 공개된 웹 소프트웨어를 대상으로 취약성 검출 실험을 수행했으며 이를 통해 보안 취약성 검출의 정확성과 실행 속도에서 실용성을 확인할 수 있었다.

본 논문의 전체적인 구성은 다음과 같다. 2장에서는 다루고자 하는 보안 취약성의 내용 및 기존 연구에 대해 설명한다. 3장에서는 분석기 개발의 주요 사항인 요약 공간의 설계와 취약성 분석의 과정을 설명하며, 아울러 실행 화면의 예를 보인다. 4장에서는 개발된 취약성 분석기의 성능 실험 결과를 제시하며, 마지막으로 5장에서 결론을 맺는다.

2. 연구 배경

2.1 PHP 파일 삽입 취약성

PHP 파일 삽입 취약성은 외부 입력으로부터 생성된 값이 PHP 프로그램 내 파일 삽입을 위한 include, include_once, require, require_once 문이나 코드 수행을 위한 system 문의 인자값으로 사용될 때, 발생할 수 있는 보안 취약성을 말한다[7,8]. 웹 응용 프로그램은 임의의 외부 데이터를 사용함으로 이러한 취약성이 존재할 경우 위험한 파일의 삽입이나 의도하지 않은 프로그램의 수행이 발생할 수 있다. 아래는 이런 취약성의 전형적인 예이다. 아래 코드가 파일명 test.php로 저장되고 코드가 수행되는 호스트 이름을 foo.com이라고 하자.

```
$dir = $_GET['module_name'];
...
include($dir . "/function.php");
```

위의 PHP 코드는 변수 module_name의 값을 웹으로부터 입력 받아 이를 아래의 include 문의 파일 경로값 생성에 사용하고 있다. 즉, 입력 받은 문자열 $dir과 "/function.php"란 문자열을 합쳐 삽입 할 파일의 저장 경로를 계산한다. 이때 만일 공격자가 악의적인 코드 파일을 http://malicious.com/function.php에 설치한 후, http://foo.com/test.php?module_name=http://malicious.com란 URL로 foo.com 사이트에 접근한다면 위 코드는 foo.com 서버 안에서 "http://malicious.com/function.php"를 동적으로 삽입하여 수행하게 되어 foo.com 측의 보안성(security) 침해를 야기할 수 있게 된다.

PHP 삽입 공격은 프로그램 내 정보 흐름의 보안성과 관련이 있다. 즉, 안전하지 않은 외부 입력이 프로그램 삽입이나 시스템 명령어 수행과 같은 민감한 작업의 인자값에 영향을 미칠 때 프로그램은 취약성을 가지게 된다. 따라서 본 연구에서는 외부에서 입력된 값이 어떻게 전달되어 사용되는지를 세밀히 분석함으로써 취약성의 존재를 알아내고자 한다.

2.2 요약 해석 기법

요약 해석(abstract interpretation)이란 프로그램 실행 시에 변수들이 가질 수 있는 값을 격자(lattice) 구조의 유한한 요약값들로 근사하여 표현한 후, 이 격자 구조의 자료값 공간에서 프로그램을 미리 수행해 봄으로써 프로그램의 실제 수행 결과를 예측할 수 있는 대표적인 의미 기반 정적 분석 기법으로[18,19,] 1970년도에 기본적인 방법론이 제시된 이래 다양한 프로그램 분석에 대한 구현 방법론으로 사용되어 왔다. 프로그램의 변수들은 외부 입력이나 실제 코드 수행 상황에 따라 매우 다양한 값을 가질 수 있고 경우에 따라서는 그 수가 무한할 수 있으므로, 요약 해석 방법에서는 발생 가능한 무한한 변수 값들을 유한한 공간 내의 포괄적 의미를 부여한 요약 값들로 근사하여 나타낸다. 그리고, 이런 요약 공간의 범위 내에서 프로그램을 수행시켜 봄으로써, 수행 중 발생 가능한 프로그램의 상태 정보를 미리 알아 낼 수 있다[19].

프로그램을 미리 수행해 본다는 개념에서 유사한 기법으로는 프로그램 테스팅 기법이 있지만, 이 기법은 무한한 실제 입력값 조합 중 일부의 경우에 대해서만 분석이 이루어지는 반면, 요약 해석 방법론은 일종의 정적 분석기법으로서 적절한 요약을 통하여 프로그램 수행 전에 프로그램 수행시의 모든 상황을 고려할 수 있다는(soundness) 장점이 있다. 또한 자료 흐름 분석과 같은 일반적인 정적 분석 방법과 비교하여 요약해석 방법은 프로그램의 의미를 참조하여 실제 수행하는 방식에 기반하여 분석이 이루어지므로 포인터나 PHP의 변수값 변수 및 파일명이 변수에 저장되어 있는 파일 삽입 등을 효과적으로 처리할 수 있는 장점을 가진다.

다음은 일반적인 자료 흐름 분석으로 분석이 어려운 PHP의 변수 값 변수의 예이다.

```
$x = "abc"
$$x = 123;
```

PHP 프로그램에서 두 번째 문장의 실행 시 $x의 값이 "abc"이므로 두 번째 문장은 변수 $abc에 정수 123을 저장하게 된다. 이러한 프로그램을 분석할 경우에 두 번째 수행문이 변수 $abc의 값에 영향을 준다는 사실은 $x의 수행시의 값을 파악해야 알 수 있으므로 일반적인 자료 흐름 분석으로는 분석이 어려운 내용이 된다. 이에 대하여 요약 해석은 적절히 제한된 데이터 공간 내에서 실제 프로그램을 수행하는 방식으로 프로그램을 분석하므로, 프로그램 수행시의 계산되는 값을 적절히 요약하여 유지함으로써 이러한 경우를 필요한 정밀도 내에서 분석해 낼 수 있다.

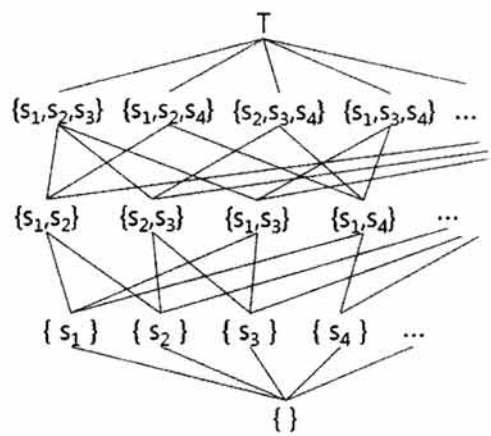
이러한 요약해석 방법론은 대부분의 프로그램 의미론에 적용 가능하며, 기존의 집합기반 분석, 타입 기반 분석 등 다양한 프로그램 분석방법을 포괄하는 방법론으로 사용될 수 있는 프로그램 분석 방법론으로 제시되고 있다[22].

요약 해석에 사용되는 요약 공간 개념을 살펴보기 위해 (그림 1)의 예를 사용한다. (그림 1(a))는 요약 해석을 적용할 PHP 코드의 예이고, (그림 1(b))는 프로그램 변수의 요약 공간 예로서 최대 3개의 원소까지 표현이 가능한 스트링 멱잡합 공간 $P^3(String)$이다. 요약 공간은 최대 3개 값을 가질 수 있는 문자열 집합과 최상위 원소(T)를 가지며, 이들 간의 상하관계는 실선으로 표시된다.

요약 공간에서 선으로 연결된 상하관계는 상위의 원소가 더 넓은 범위로 포괄하여 근사함을 나타낸다. 예를 들어 요약값 {"v1", "v2"}는 이 두 값을 모두 가진다는 뜻이 아니라 아무 값도 가지지 않는 경우, "v1"만 가질 경우, "v2"만 가질 경우, "v1"과 "v2"를 모두 가질 수 있는 경우를 모두 포괄한다. 정적 분석은 정확한 실제 상황을 분석할 수는 없으므로 실제 수행 상황을 포괄하는 안전한(sound) 분석 결과를 생성하게 되며 이때 안전성의 의미는 실제 수행시의 값을 모두 포함한다는 뜻을 가진다. 예를 들어, 실제 수행 중에 변수 $x가 한 지점에서 "v1" 값만을 가질 경우 $x가 가질 수 있는 값을 {"v1"}으로 분석하는 것이 {"v1", "v2"}로 분석하는 것 보다 정밀한 분석이 되나, 두 분석 결과는 모두 안전한 분석이 된다.

```
1:  $x = $input();
2:  if ($x>0)
3:     $start = "a";
4:  else
5:     $start = "b";
6:  $y = $start;
7:  while (strlen($y)<9)
8:    $y = $y."c";
```

(a)



(b)

(그림 1) $P^3(String)$: 최대 3개의 원소를 표현하는 문자열 멱집합 공간의 사용

(그림 1(b))의 공집합 { }는 수행 중 가질 수 있는 값이 없음을 의미하고, 네 가지 이상의 값을 가질 수 있을 때는 임의의 값을 가질 수 있음을 나타내는 T로 표시한다. 이때 집합이 가질 수 있는 최대의 원소 개수 3은 필요성에 따라 적절히 증가할 수 있다. 어떤 변수 $x가 T로 분석되었음은 $x가 임의의 값을 가질 수 있음을 의미하며, T를 사용함으로써 요약공간의 크기를 한정하면서도 모든 가능한 값을 포함한다는 면에서 해석의 안전성을 확보할 수 있다. 만약 변수 $x가 프로그램의 한 위치에서 "a", "b", "c", "d"의 네 가지 값을 가질 수 있다면 분석결과는 이러한 모든 결과를 포괄하도록 임의의 값을 가질 수 있음을 나타내는 T로 표현되게 된다. 이와 같이 무한한 경우를 포괄하는 적절한 요약값을 사용함으로써 실행시의 무한한 수행상황을 유한한 공간에서 나타낼 수 있다. 이를 통하여 무한한 경우를 일일히 동적 분석이나 테스팅을 통하여 수행해 보지 않고 유한한 시간 내에 프로그램의 성질을 분석할 수 있게 된다.

이제 (그림 1(a))의 코드를 요약해석에 따라 수행하는 과정을 살펴본다. 라인 1에서는 임의의 문자열이 입력될 수 있기에 라인 1의 수행 직후 변수 $x의 변수값 요약은 [$x: T]로 표현된다. 이어서 if 문이 조건 충족되어 라인 3이 수행된다면 변수 $x와 $start의 메모리 상태는 [$x:T, $start:{a}]이 되며, 그렇지 않고 라인 5가 수행된다면 [$x:T, $start:{b}]의 상태를 갖는다. 따라서 이 if 문 직후의 메모리 상태는 이 두 경우를 포함하는 [$x:T, $start:{a, b}]로 표현한다. 이렇게 발생 가능한 경우들을 합치는 것을 조인(join)이라고 하며 이를 통해 모든 수행 가능성을 고려할 수 있다. 또, 라인 6에서는 변수 $y가 변수 $start의 값으로 치환되므로 수행 후의 상태는 [$x:T, $start:{a, b}, $y:{a, b}]가 된다. 이어지는 while 문의 경우 반복 수행에 의해 변수 $y의 값이 ["ac", "bc", "acc", "bcc", "accc", "bccc", ...]등으로 늘어나 가능한 변수 값의 종류가 한도를 초과하여 T로 근사되며, while문이 종료한 후의 변수들의 값은 [$x: T, $start:{a, b}, $y:T]로 분석된다. 이와 같이 요약 해석 방법은 프로그램 수행 전에 프로그램 실제 수행의 모든 상황을 요약된 값으로 포괄하여 나타내고, 이를 실제 수행하는 것과 비슷한 방법으로 분석함으로써 모든 가능한 입력에 대하여 동적 분석을 수행하는 효과를 얻을 수 있다.

2.3 기존 연구

앞서 살펴본 PHP 파일 삽입 취약성 처리에 대해서는 다양한 연구들이 있었다. CWE(Common Weakness Enumeration)에 의하면 미리 정해진 파일만 삽입되도록 프로그램을 작성하거나, 환경 설정을 통하여 정해진 디렉터리 외에는 접근이 차단되도록 설정하는 방법이 제시되고 있다[8]. 그러나 이러한 방법은 프로그램 작성자나 시스템 관리자가 해당 취약성에 대한 대응을 적절하게 하지 못할 경우 취약성을 배제할 수 없는 약점을 가진다.

한편 Scott와 Sharp는 시스템에 설치되는 각각의 웹 응용 프로그램에 대하여 유효하지 않은 악의적인 입력을 필터링하는 응용 프로그램 방화벽(application firewall)의 사용을 제안하였고[9,10], 이러한 방식은 웹 응용 프로그램에 전달되는 자료값에 대한 검사를 수행하는 AppSheild와 InterDo등의 상용제품에 적용되었다[11, 12]. 그러나 이러한 응용 프로그램 방화벽 방식은 보안상의 안전을 사용자 또는 관리자의 검사 규칙 설정에 크게 의존한다. 또한 웹 응용 프로그램의 실행 전단부에서 사용자 HTTP 메시지에 저장된 자료값을 검사해야 하므로 웹사이트의 성능이 저하되는 문제가 있다.

이와 다른 방식으로, 가상의 공격을 통하여 웹 사이트의 보안 문제점을 찾아내는 침투 테스트(penetration test) 방법이 제안되어, Huanget등은 웹 응용 프로그램의 취약성을 식별해 내기 위해 블랙박스 검사(black-boxed testing)를 제공하는 웹 응용 프로그램 보안 평가 방법을 설계하였다[13]. 이 방식은 현재 상업적으로 사용되는 방법이기는 하지만 이런 검사를 통해서는 보안 취약성 부재를 안정적으로 증명하지는 못하는 한계가 있다.

이와는 달리 본 연구에서는 의미 기반 정적 분석 방법을 사용한다. 의미 기반 정적 분석 방법은 프로그램의 수행 의미에 기반하여 프로그램 수행 전에 추가적인 사용자의 설정 없이 자동으로 프로그램 내의 취약성을 찾아주기 때문에, 위의 기존 접근 방법들과 다르게 취약성 검출의 안전성을 보장하고 프로그램 실행시의 성능 부담이 없다는 장점을 가진다.

정적 분석을 사용한 취약성 분석 방법으로 fortify와 같은 상용 도구가 개발되었으나[14], 이러한 상용도구는 의미 기반이 아닌 일반적인 자료흐름 분석을 사용하고 있으며[15], 기존에 발견된 취약점 발생 패턴을 DB화 하여 취약성을 검출하기 때문에 검사의 안전성을 보장할 수 없고 변수값 변수, 함수 포인터, 고차 함수 등으로 인한 복잡한 자료 흐름의 분석이 어려운 단점이 있다. 따라서 요약 해석과 같은 의미 기반의 분석 방법을 사용하여 취약성을 검출하고자 하는 방법에 대한 연구가 진행되어 왔으며, 그 결과 SQL 삽입 공격 취약성과 XSS 취약성에 대한 연구 결과[16,17]가 발표된 바 있다. 본 연구에서는 요약 해석 기법을 사용한 PHP 삽입 공격의 정적 분석을 대상 취약성으로 설정하여, 이 취약성에 최적화된 정적 분석기를 구현하고자 한다.

3. 정적 분석기 구현

본 장에서는 앞서 설명된 요약 해석 기법을 사용한 보안 취약성 정적 분석기의 구현과 관련된 주요 내용을 기술한다.

3.1 분석기를 위한 요약 공간 설계

분석기에 사용될 요약 해석 기법은 다음과 같은 장점을 가진다[18,19]. 첫째로 실제 프로그램을 실행하는 것과 거의 유사한 방식으로 자료값 계산 및 전달 과정을 추적하기에, 기존의 자료 흐름 분석[15]에서 다루기 어려운 함수 포인터나 문자열 첨자를 사용한 변수, 고차 함수, 변수값 변수와 같은 프로그램 구조에 효과적으로 대처할 수 있다. 둘째로

실제 발생 가능한 모든 수행 값을 요약하여 사용하므로 안정적(sound) 분석이 가능하다. 마지막으로 해석에 사용될 요약 공간을 적절히 조정함으로써 분석에 소요되는 계산과 분석의 정밀도 간의 중요도를 적절히 조정할 수 있다.

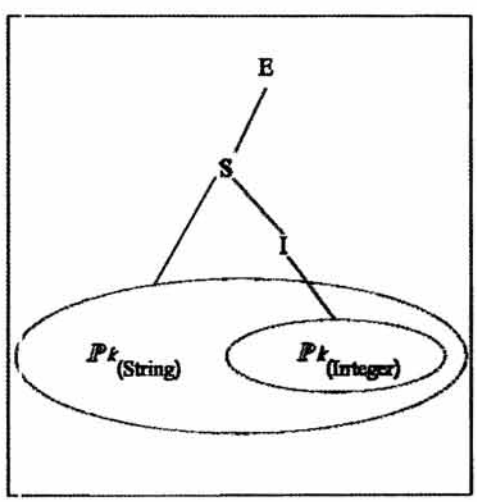
이를 고려할 때 분석 대상인 PHP 언어는 변수값 변수와 같은 문자열 값의 복잡한 사용으로 인하여 일반적인 자료흐름 분석 기법으로는 정확한 분석이 어렵기 때문에 요약 해석과 같은 의미 기반 분석이 적합한 것으로 판단된다. 요약 해석 기반의 정적 분석기의 구현을 위해서는 분석 목적에 적합한 요약 공간의 설계가 가장 중요한 문제가 되며 본 연구를 위해서는 다음과 같은 점이 고려되어야 한다.

첫째로 외부에서 전달되는 입력을 사용하여 계산한 값과 프로그램 내부의 정적인 자료값을 사용하여 계산된 값을 구별할 수 있어야 한다. PHP파일 삽입 취약성은 외부의 불특정 입력이 여러 경로를 거쳐 전달되어 삽입 취약성과 관련한 include, include_once, require, require_once, system 등의 인자로 사용될 때 발생할 수 있다. 따라서 PHP 삽입 취약성을 분석하기 위해서는 해당 인자 값이 외부에서 전달된 값을 사용하여 생성된 값인지 여부를 알아내야 하므로, 외부의 값에 대한 구별된 표현이 분석 공간 내에 포함되어야 한다.

둘째로 문자열 값에 대한 적절한 분석이 필요하다. PHP 프로그램은 문자열을 배열의 첨자로 사용할 수 있으며 변수의 값을 변수명으로 사용하는 변수값 변수(variable variables)를 사용하기 때문에, 변수들이 어떤 문자열 값을 갖는지에 대한 정보가 있어야 자료의 흐름에 대한 정확한 분석이 가능하다. 또한 PHP의 특징인 파일 삽입으로 인한 프로그램 수행의 흐름을 파악하기 위해서는 include와 같은 삽입문의 인가값인 파일명을 알아내야 하므로 문자열에 대한 분석이 필수적이다.

위 첫 번째 요구사항에 대하여 외부의 입력이 포함될 수 있을 경우 임의의 값이 전달될 수 있고 위험한 공격이 발생할 수 있으므로 이를 나타내는 특별한 최상위 값을 설정하는 것이 필요하며, 두 번째 요구사항에 의하여 문자열 변수에 대한 값분석이 이루어지도록 문자열의 멱집합을 분석공간 설계에 포함시키는 것이 필수적이다.

(그림 2)는 설계된 격자 구조의 요약 공간을 보이고 있다. 외부에서 전달되는 입력을 포함할 수 있는 경우 공격 문자열을 포함한 임의의 문자열을 가질 수 있으므로 이를 표현하는 최상위 원소 E를 사용하였다. E 하위의 값들은 외부 입력의 영향을 받지 않는 내부에서 계산된 값을 나타내며 변수가 가질 수 있는 가능한 내부 계산값들을 표현하기 위하여 원소의 갯수를 k로 제한한 문자열값의 멱집합 공간을 사용하였다. PHP에서는 숫자와 일반 문자열이 혼용되므로 집합의 원소는 숫자나 문자열이 모두 가능하며, 숫자값만을 가지는 경우를 따로 나타내기 위하여 문자열 멱집합의 부분집합인 숫자들의 멱집합을 세분화하여 포함시켰다. 또한 프로그램 분석을 유한시간 내에 종료하기 위하여 내부에서 생성된 값들의 집합의 크기가 미리 정한 정확도 척도 k를 초과할 경우 이러한 집합들을 포괄적으로 나타내기 위한 요약값인 S와 I를 포함시켰다. 집합의 원소가 모두 숫자이면 임의의 숫자를 가질 수 있음을 나타내는 기호인 I로 표현되며, 집합에 숫자가 아닌 문자열이 포함되어 있으면 임의의 문자열 값을 가질 수 있음을 나타내는 S로 근사하여 표현된다. 이러한 근사를 통하여 분석의 정밀도를 잃어버릴 수 있지만, 적절한 k 를 설정함으로써 효율적이면서도 적정한 정밀도와 분석 속도를 얻어낼 수 있다. 이러한 k 값은 목적으로 하는 분석의 목표와 소요시간에 따라서 필요로 하는 정밀도를 얻을 수 있고 적절한 수행시간을 가지도록 실험들을 통하여 적절히 설정하게 된다. PHP 프로그램 내에서 하나의 변수가 한 프로그램 지점에서 가지는 문자열 값의 범위는 대부분 제한적이며 본 분석의 근본적인 목적은 외부의 위험한



•요약 공간 값의 의미

$C(E)$ = String^{ext} (외부입력값 포함 가능)
$C(S)$ = String^{in} (내부 생성 문자열 만을 가짐)
$C(I)$ = Integer (내부 생성 숫자값 만을 가짐)
$C(\{s_1,\ldots,s_n\}) = \{s_1,\ldots,s_n\}$ (실제 가질 수 있는 값을 나열함)

•조인(join) 연산

$E \sqcup v = v \sqcup E = E$
$S \sqcup v = v \sqcup S = S$
$I \sqcup I = I$
$I \sqcup \{s_1,s_2,\ldots,s_n\} = \{s_1,s_2,\ldots,s_n\} \sqcup I = I$ (s는 모두 숫자)
$I \sqcup \{s_1,s_2,\ldots,s_n\} = \{s_1,s_2,\ldots,s_n\} \sqcup I = S$ (숫자 아닌, s가 존재)
$\{s_1,s_2,\ldots,s_n\} \sqcup \{p_1,p_2,\ldots,p_m\} = \lceil \{s_1,s_2,\ldots,s_n\} \cup \{p_1,p_2,\ldots,p_m\} \rceil^k$

•집합크기 제한 연산자

$\lceil \{s_1,\ldots,s_n\} \rceil^k = \{s_1,\ldots,s_n\}$ n≤k일 경우
$\lceil \{s_1,\ldots,s_n\} \rceil^k = I$ n>k이고, s 는 모두 숫자
$\lceil \{s_1,\ldots,s_n\} \rceil^k = S$ 그 외의 경우

(그림 2) 설계된 요약공간과 연산 규칙

입력을 분석해 내는데 있으므로 10 이내의 비교적 작은 k 값을 사용하더라도 적절한 취약성 분석을 수행할 수 있을 것으로 판단된다.

격자 구조인 요약 공간의 설계를 위해서는 요약 공간 원소들 간의 최소 상한값(least upper bound)을 계산하는 조인 연산을 의미에 맞게 정의하여야 한다. 요약 해석에 기반한 분석에서 조인(⊔) 연산은 두 원소를 포괄하는 의미의 값을 생성한다. 즉 예를 들어 어떤 한 경로의 수행으로 $x가 가질 수 있는 값이 v_1으로 나타내어 지고, 다른 경로의 수행으로 $x가 가질 수 있는 값이 v_2로 나타내어 진다면, 두 경로가 모두 가능할 경우 $x가 가질 수 있는 값은 두 값을 포괄하는 $v_1 \sqcup v_2$로 표현된다.

(그림 2)의 오른쪽 부분은 설계된 요약 공간의 원소들간의 조인 연산을 정의하고 있다. 각각의 경우에 대한 결과 정의는 상위의 경우가 우선순위를 갖는다. E값과의 임의의 값과의 조인 연산은 E가 모든 다른 값을 포괄하므로 E를 결과로 생성하며, 그 외의 경우에 S가 피연산자로 있을 경우에는 결과는 S가 된다. I와 집합표현과의 조인 연산은 집합 표현이 숫자만을 포함할 경우 I가 되며 숫자가 아닌 문자열이 포함되어 있을 경우에는 I는 숫자만을 나타내므로 상위의 요약값인 S가 결과가 된다. 집합표현간의 조인 연산은 두 집합의 합집합을 결과로 생성하며 합집합의 결과가 집합의 원소수의 제한을 초과할 경우에는 숫자가 아닌 원소의 포함 여부에 따라 S또는 I로 근사한 값을 결과로 생성하게 된다.

3.2 구현된 정적 분석의 단계

구현된 PHP 삽입 취약성 분석기는 크게 세 단계에 걸쳐 취약성 검출 작업을 수행한다. 첫 단계인 전단부(frontend)에서는 입력된 PHP 프로그램의 어휘 분석과 구문 분석을 수행하여 추상 구문 트리(Abstract Syntax Tree: AST)를 생성하고, 이에 대한 전처리를 통하여 분석을 용이하게 해주는 과정을 거친다. 두 번째 단계는 값 분석 단계로서 요약 해석이 수행되는 단계라 할 수 있다. 마지막은 취약성 검출 단계로서 값 분석의 결과를 사용하여 PHP 삽입 취약성이 검사되고 그 결과가 보고되는 단계이다. 이들 단계의 구체적 내용은 다음과 같다.

3.2.1 전단부

전단부는 추상 구문 트리의 생성과 분석을 위한 전처리 단계로 이루어진다. 입력 받은 프로그램은 어휘 분석과 구문 분석 과정을 거쳐 추상 구문 트리로 변환된다. 생성된 추상 구문 트리에 대하여 분석을 위한 전처리로서, 사용자 정의 함수 및 파일 테이블을 생성하고 변수 범위 처리 작업을 수행한다.

함수 및 파일 테이블은 함수간 분석과 파일간 분석을 위한 기본 정보를 저장한다. 함수 및 파일테이블에는 프로그램 내에서 사용자에 의하여 정의된 함수와 파일에 대한 이름과 추상 구문 트리 위치를 저장한다. 이를 통하여 함수 호출의 분석 시 라이브러리 함수와 사용자 정의 함수를 구별할 수 있도록 해준다. 또한 각각의 파일에 대하여 각 파일의 시작 전 상태와 수행 후 상태를 접근할 수 있도록 함으로써 include 문과 같은 파일 삽입 문장의 수행 결과를 분석할 수 있도록 해준다. 값 분석 단계에서 파일 삽입과 관련된 문장을 분석 시에, 해당 파일의 수행 후 상태가 파일 삽입 문장의 결과 상태로 사용되고 파일 삽입 문장의 수행 전 상태가 해당 파일의 수행 전 상태로 갱신된다. 또한 한 파일의 수행 후 상태가 갱신 되면 해당 파일에 대한 모든 삽입 문장의 수행 후 상태가 갱신된다. 또한 이와 같은 파일간의 삽입 관계를 고려한 분석은, 파일 삽입 취약성 분석의 정확성을 높이는데 필수적인 요소이다.

변수 범위 처리 작업에서는 프로그램의 전역 변수와 지역 변수의 이름이 동일한 경우를 처리하기 위하여, PHP 언어의 변수 범위 규칙에 따라, 충돌하는 지역 변수에 대한 적절한 변수 이름 변경(renaming) 작업을 수행한다.

3.2.2 값 분석 단계

값 분석 단계에서는 3.1의 요약 공간 상에서 프로그램을 수행해 봄으로써, 소스 코드의 모든 부분(수식, 변수, 함수 호출, 지정문, if 문 등)에서 어떤 값이 계산되고 계산 전후에 변수들이 어떤 값을 갖는지 분석된다.

〈표 1〉 분석 정보 표현을 위한 공간 정의

State : (SymbolTable, Memory)
SymbolTable : Variable → Location
Memory : Location → Value
preState : Node → State
postState : Node → (Value, State)

<표 1>은 값 분석 단계에서 사용되는 자료 구조의 타입을 보여주고 있다. Value는 3.1절에서 설명한 요약 도메인의 원소로서 실제 계산될 수 있는 문자열 값을 근사하여 나타내는 요약 공간의 값이다. 프로그램 수행 중의 상태는 State 로 나타내며 심볼테이블(SymbolTable)과 메모리상태(Memory)로 이루어진다. SymbolTable은 각 변수에 대한 메모리 주소를 저장하며, 이를 통해 변수들 사이의 이명 관계(aliasing)를 저장한다. Memory는 각각의 메모리 주소에 대하여 해당 주소에 저장된 요약 공간의 값(Value)을 저장한다. preState와 postState는 프로그램의 각각의 부분을 나타내는 추상 구문 트리의 노드(Node)들에 대하여 해당 노드가 나타내는 프로그램 코드 수행 전후의 상태를 저장한다. preState는 각 노드에 대하여 해당 코드 수행 전의 상태(State)를 저장하며, postState는 해당 코드 수행으로 인하여 계산된 값과 해당 코드 수행 후의 상태(State)를 저장한다. preState와 postState는 프로그램 내의 모든 노드에 대하여 계산되어 해쉬 테이블에 저장된다. 값분석 단계가 끝나면 이 해쉬 테이블을 사용하여, 추상 구문 트리의 각각의 노드

가 나타내는 프로그램 코드의 수행 전후의 상태와 계산되는 값을 알 수 있다.

추상 구문 트리의 각각의 노드에 대하여 수행 전 상태와 수행 후 상태를 계산하기 위한 값분석 단계는 일반적인 요약해석 분석의 방식대로 방정식 생성 단계와 값 계산 단계의 두 단계로 이루어진다.

방정식 생성단계에서는 프로그램 내 모든 노드 들 간의 수행 전 상태와 수행 후 상태들 간의 관계식을 도출하는 단계이다. 예를 들어 식 e_i의 수행 전 메모리 상태를 $Pre(e_i)$, 수행 후 메모리 상태를 $Post(e_i)$, 수행한 결과를 $V(e_i)$라고 표시한다고 하고 (위 표 1에서 PostState(e)는 (V(e), Post(e))가 된다.), 프로그램 수행문 노드 e0의 형태가 $x=e1+e2의 지정문 형태일 경우에 다음과 같은 관계식이 도출될 수 있다.

Pre(e1) = Pre(e) // e 수행 전 메모리 상태에서 e1이 수행

Pre(e2) = Post(e1) // e1 수행 후 상태에서 e2가 수행됨

Post(e) = Post(e2)[$x->V(e1)+V(e2)] // e의 수행 후 상태는 e2 수행 후 상태에서 변수 $x의 값을 e1의 값과 e2의 값을 더한 값으로 갱신한 상태임.

V(e) = V(e1)+V(e2) // 지정된 값이 지정문의 값이 됨.

이와 같은 방식으로 추상 구문 트리의 모든 노드에 대하여 해당 식과 부분식과의 관계식을 도출함으로써 방정식이 완성되며 이러한 방정식의 해가 프로그램의 분석 결과가 된다.

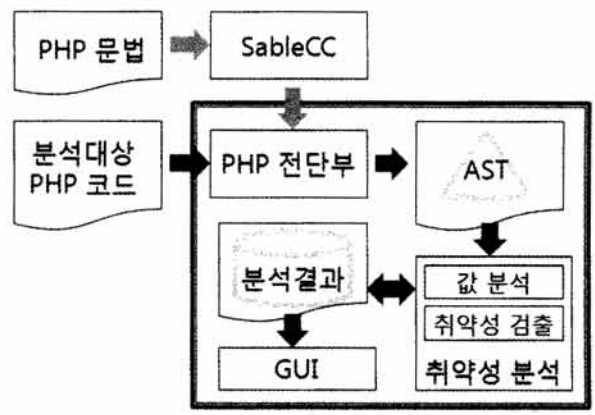
값 계산 단계에서는 방정식을 해를 구하는 단계로서 대부분의 분석 시간은 값 계산 단계에서 소요된다. 방정식의 해를 계산하기 위하여 일단 모든 방정식의 구하고자 하는 상태값들을 요약 공간의 최하위 값으로 초기화 한 후에 고정점(fixed point)에 도달할 때까지 반복하여 위 방정식의 식들을 적용하는 고정점 계산 방법을 사용한다. 본 연구의 분석기에는 일반적인 고정점 계산 알고리즘인 워크리스트 알고리즘(Worklist Algorithm)[18, 23],이 사용되었으며, 본 연구의 프로그램 분석을 위한 워크리스트 알고리즘은 프로그램 내의 모든 계산식의 개수를 e, 요약 공간의 최하위 원소로부터 최상위 원소까지의 증가 체인(ascending chain)의 최대 길이를 h 프로그램 내의 변수의 개수를 v라고 할 때 O(ehv)의 최악의 시간 복잡도를 갖는다.

3.2.3 취약성 검출 단계

취약성 검출 단계에서는 값 분석 단계에서 생성된 정보를 사용하여 프로그램 내에 존재하는 PHP 삽입 취약성을 찾는 작업을 수행한다. 취약성 검출 단계에서는 추상 구문 트리를 순회(traversal) 하면서, PHP 삽입 취약성과 관련된 include, require, include_once, require_once, system 문의 인자값이 어떤 값을 갖는지 postState 해쉬 테이블을 사용하여 검사한다. 만약 해당 실인자의 값이 요약 공간의 E일 경우 이는 해당 문장 수행의 인자값으로 외부에서 전달된 값이 사용될 수 있음을 나타내는 것이므로 취약성이 존재하는 것으로 판정하게 되고 해당 정보를 저장하여 사용자에게 보고한다.

3.3 분석기 시스템 구조

(그림 3)은 구현 분석기의 시스템 구성도이다. PHP 언어에 대한 전단부를 구현하기 위하여 프로그램 정적 분석기 생성 도구인 SableCC를 사용하였다[21]. SableCC는 PHP 문법을 입력으로 받아 PHP 구문 분석기와 어휘 분석기를 자동으로 생성하며 생성된 구문 분석기 및 어휘 분석기는 본 연구의 정적 분석기의 전단부에 포함된다.

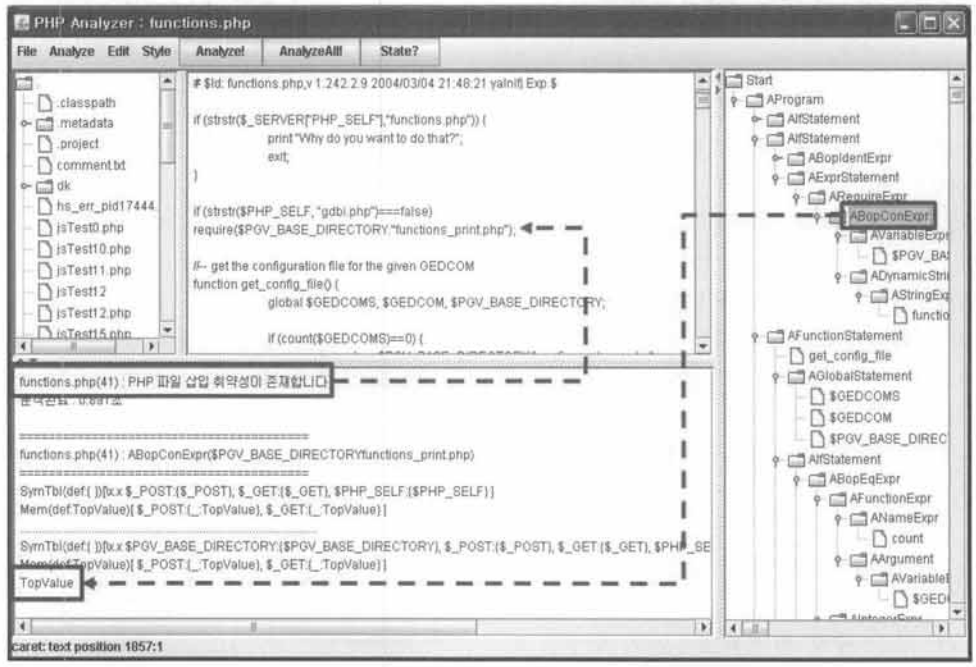


(그림 3) 분석기의 시스템 구성도

입력된 분석 대상 PHP 프로그램은 전단부를 거쳐 문법 구조를 나타내는 추상 구문 트리로 변환된다. 생성된 추상 구문트리에 대한 취약성 분석 과정은 두 단계로 이루어진다. 값분석 과정을 통하여 프로그램 내의 각각의 부분이 가질 수 있는 값과 각 프로그램 문장 수행 전후의 상태가 분석되어 해쉬 테이블 구조에 각각의 추상 구문 트리 노드에 대한 값이 저장되며, 취약성 추출 단계에서 이 결과를 바탕으로 PHP 삽입 취약성이 해당 프로그램에 존재하는지 각각 관련 구문을 중심으로 조사된다. 분석된 결과는 GUI를 통하여 사용자에게 보고되며, 사용자는 GUI를 통하여 취약성의 존재 여부와 프로그램 내 각각의 문장의 수행 전후의 메모리 상태를 살펴볼 수 있다.

3.4 실행 화면

(그림 4)는 구현된 분석기를 사용하여 공개 소프트웨어인 PHPGEDVIEW의 소스 파일 중 하나인 functions.php에 대한 취약성 분석을 수행한 화면이다. 화면의 구성은 다음과 같다. 좌측 상단에는 관련된 소스 파일들이 저장된 디렉토리 정보가 보여지며, 이 화면을 사용하여 취약성 분석을 수행할 파일을 선택하거나, 이미 분석이 수행된 파일에 대하여 취약성 정보를 열람할 수 있다. 상단의 중앙은 현재 분



(그림 4) GUI 도구를 통한 결과 분석 화면

석된 프로그램의 소스 코드를 보이며, 우측의 트리 구조는 해당 PHP 프로그램의 추상 구문 트리이다. 화면 하단에는 취약성 검출 결과와 취약성과 관련한 메모리 상태, 분석 소요 시간 등의 정보가 출력된다.

프로그램 파일을 선택한 후 상단의 분석(Analyze) 버튼을 누르면 해당 파일에 대한 분석이 수행되어 취약성 검출 결과가 출력된다. 또는 AnalyzeAll 버튼을 사용하여 해당 파일의 디렉토리에 속한 모든 PHP 파일에 대한 분석을 수행할 수도 있다. 상단 가운데 창의 소스 코드 중 일부를 마우스로 선택하고 State 버튼을 누르면 해당 선택된 프로그램 부분에서 계산되는 요약 공간의 값과 수행 전후의 상태 정보를 열람할 수 있다. 이러한 계산값 및 상태 정보의 열람은 추상 구문 트리의 노드를 클릭하는 방법의 사용도 가능하다.

(그림 4)의 분석 결과 창은 function.php 파일 41번째 줄의 require 문에 PHP 삽입 취약성이 있는 것으로 보고하고 있다. 이와 관련하여 오른쪽의 추상 구문 트리에서 require 문의 인자값에 해당하는 부분을 클릭해 보면 분석 결과 창에 인자값 계산 전후의 상태와 계산되는 값이 표시되게 된다. 현재 오른쪽 추상 구문 트리의 실인자 부분을 클릭하여 해당 인자값 계산 전후의 심볼 테이블과 메모리 상태 및 계산된 결과가 출력되어 있다. 정보 창 제일 아랫줄을 보면 인자값의 계산 결과가 최상위 값(TopValue)인 E인 것으로 출력되어 require 문의 인자로 외부 입력으로부터 생성된 값이 사용되었음을 확인할 수 있다.

4. 성능 평가

구현된 PHP 보안 취약성 정적 분석기의 실용성을 평가하기 위하여 실제로 사용되는 PHP 소프트웨어에 대한 파일 삽입 취약성 분석을 수행하였다. 분석 대상 프로그램은 CVE[5]에서 파일 삽입 취약성이 있는 것으로 보고된 프로그램 중 소스가 공개되어 있고 널리 사용되는 공개 프로그램을 선택하여 사용하였다.

분석 대상 응용 프로그램 중 PHPGEDVIEW는 웹 상에서 가계도를 보여주고 편집할 수 있는 공개 소프트웨어로서 약 130개의 PHP 파일로 구성되어 있다. 또한 Mantis는 웹을 기반으로 하는 무료 버그 추적 시스템 소프트웨어이며, PhpDig는 MySQL을 기반으로 구현된 검색 엔진 프로그램이다. 구현된 분석기는 Intel Core2 Quad 2.34GHz Q8200 CPU와 4G 메모리를 장착한 컴퓨터에서 수행되었다. 사용된 운영체제는 Windows XP이며 자바 가상기계의 최대 메모리는 1GB로 설정하였다. 또한 테스트 프로그램 수행 시 요약 공간의 멱집합의 최대 원소 개수인 k는 10으로 설정하였다.

다음은 테스트 프로그램 중 PHPGEDVIEW 프로그램과 관련하여 CVE 보고되고 있는 파일 삽입 취약점 내용이다.

3개의 파일에서 파일 삽입 취약점이 있음을 보고하고 있으며, 이와 관련된 내용을 보고하고 있는 사이트들을 참조하고 있다. 위의 PHPGEDVIEW 프로그램을 본 취약성 분석기를 사용하여 분석한 결과는 (그림 6)과 같다.

CVE에서 보고된 3개의 파일에 대한 취약성 보고 외에 추가적인 많은 취약성을 검출하고 있는 것을 알 수 있다. CVE에서 보고된 authentication_index.php 파일의 취약성에 대하여 분석 결과와 원인이 일치하는지를 확인하기 위하여, 취약성이 있는 것으로 보고되고 있는 39라인의 require문의 인자식(argument expression)을 클릭한 결과는 (그림 7)과 같다. 아래의 상태 화면은 require문의 인자인 $PGV_BASE_DIRECTORY."authenticate.php" 식의 계산 전의 상

[CVE-2004-0030] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-0030

Description

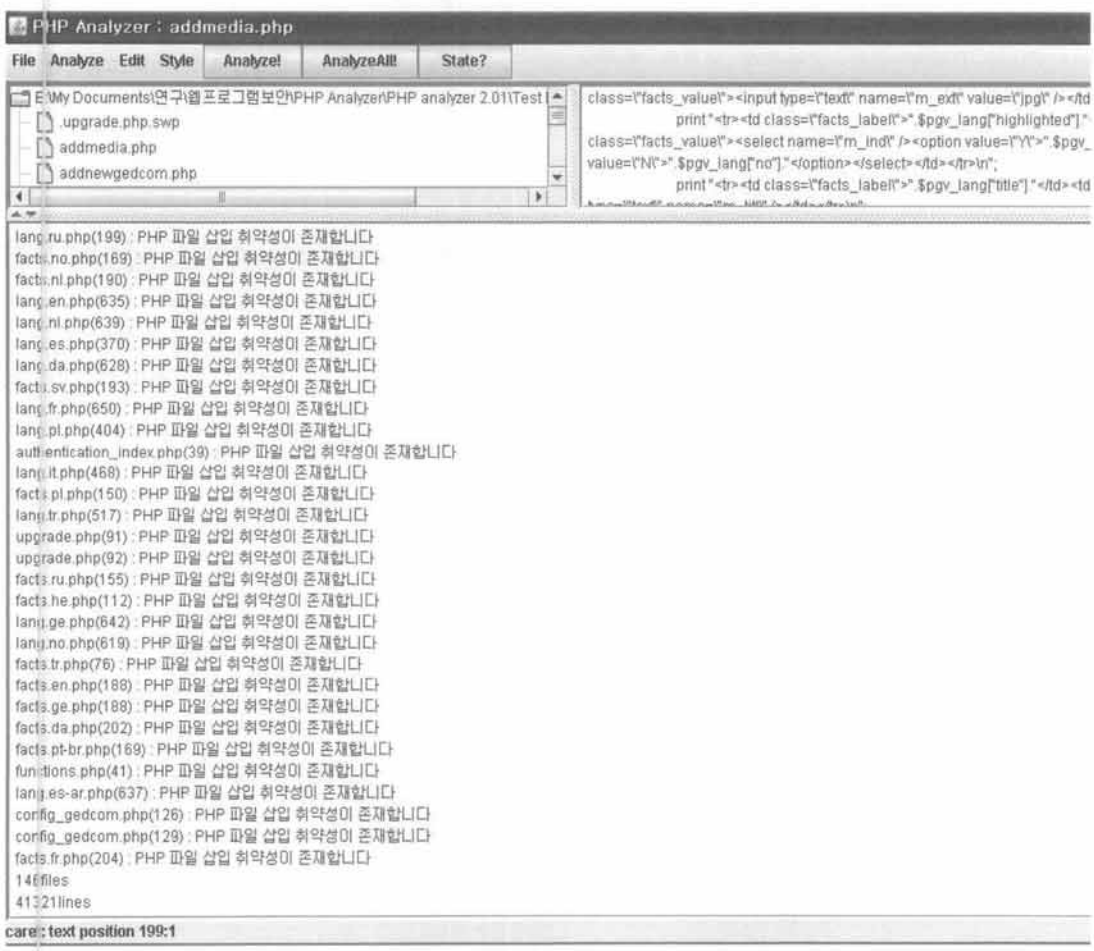
PHP remote file inclusion vulnerability in (1) functions.php, (2) authentication_index.php, and (3) config_gedcom.php for PHPGEDVIEW 2.61 allows remote attackers to execute arbitrary PHP code by modifying the PGV_BASE_DIRECTORY parameter to reference a URL on a remote web server that contains the code.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- BUGTRAQ:20040106 Vuln in PHPGEDVIEW 2.61 Multi-Problem
- URL:http://marc.theaimsgroup.com/?l=bugtraq&m=107340840209453&w=2
- BID:9368
- URL:http://www.securityfocus.com/bid/9368
- OSVDB:3343
- URL:http://www.osvdb.org/3343
- SECTRACK:1008632
- URL:http://www.securitytracker.com/id?1008632
- SECUNIA:10565
- URL:http://secunia.com/advisories/10565
- XF:phpgedview-pgvbasedirectory-file-include(14159)
- URL:http://xforce.iss.net/xforce/xfdb/14159

(그림 5) PHPGEDVIEW 프로그램의 파일 삽입 취약성 보고 화면



(그림 6) PHPGEDVIEW 프로그램의 분석 실행 화면

태와 계산 후의 상태 및 결과값을 보여주고 있으며, 이때 결과값이 TopValue로 계산되어 외부의 위험한 입력이 영향을 줄 수 있으므로 취약성이 존재함을 보여주고 있다. 이는 CVE의 보고 결과와 일치한다.

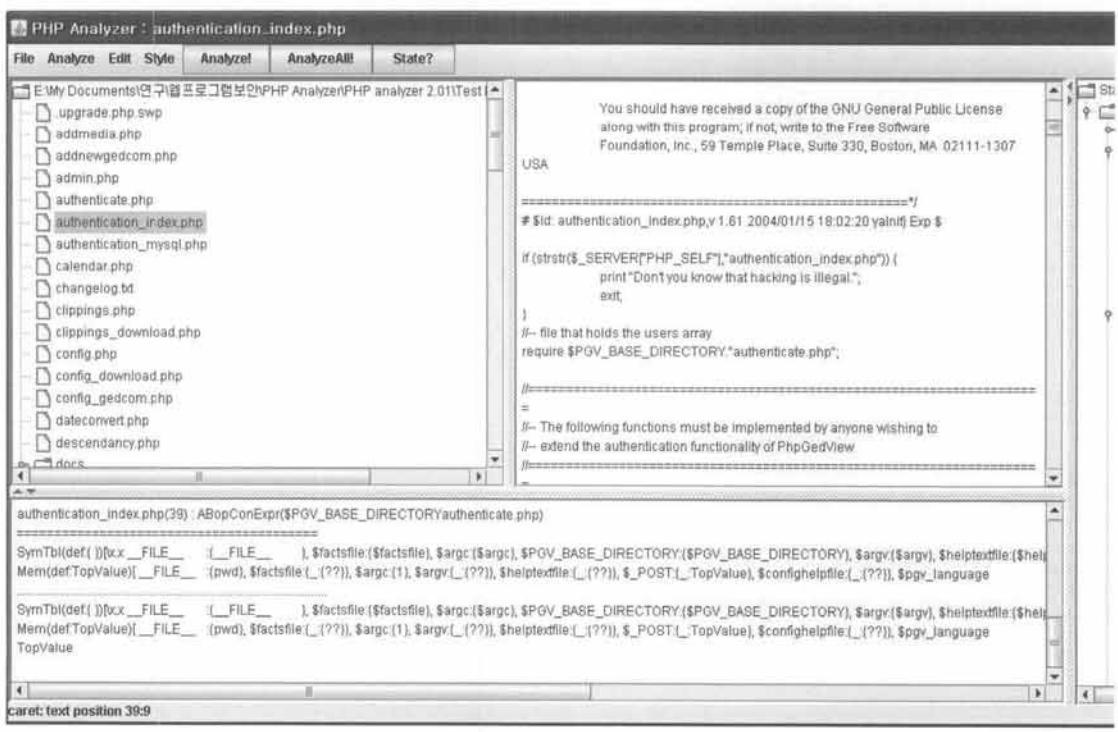
CVE에서 보고된 3개 파일의 취약성 외에 추가로 발견된 facts.*.php와 lang.*.php 파일들의 취약성도 실제 파일 삽입 취약성임이 확인되었다. 단, upgrade.php에서 발견된 2개의 취약성은 실제 파일 삽입 취약성이 아닌 잘못된 보고(false positive)로 확인되었다. upgrade.php에서 취약성이 있는 것으로 보고된 부분은 다음과 같다.

```
require ("config.php");
        ...
require($PGV_BASE_DIRECTORY .
$pgv_language["english"]);
if ( ... ) require $PGV_BASE_DIRECTORY .
$pgv_language[$LANGUAGE];
```

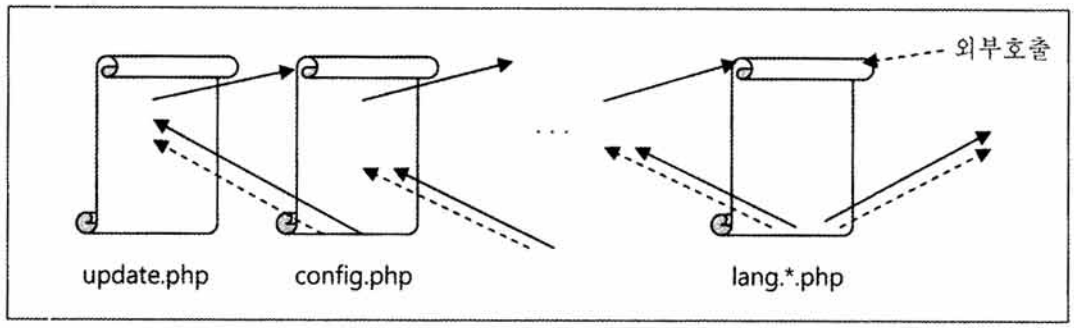
config.php 파일을 삽입하여 $PGV_BASE_DIRECTORY를 설정한 후에 해당 디렉토리의 파일을 삽입하는 부분인데 이 때 $PGV_BASE_DIRECTORY가 외부의 위험한 값을 가질 수 있기 때문에 프로그램에 취약성이 있는 것으로 보고하고 있다. 그런데, config.php 파일의 내용을 조사해 보면 여러 경로를 거쳐 lang.*.php 파일들을 수행함으로써 $PGV_BASE_DIRECTORY 값을 설정하게 되므로 실제로 upgrade.php 파일에는 취약성이 없는 것으로 보고되고 있다. 그런데, lang.*.php 파일들은 config.php를 통해 삽입되어 수행할 경우에는 문제가 없으나 다른 경로로 수행되는 경우에는 $PGV_BASE_DIRECTORY가 위험한 값을 가질 수 있는 것으로 확인되었다. 그 결과 lang.*.php의 수행을 통하여 $PGV_BASE_DIRECTORY가 위험한 값을 가질 수 있는 것으로 분석하게 되고, 따라서 update.php 파일도 취약성이 있는 것으로 판단하게 된다. (그림 8)은 이러한 상황을 그림으로 나타낸 것이다. 실선은 $PGV_BASE_DIRECTORY가 취약한 값을 갖지 않는 호출 경로와 분석된 상태를 나타내고 점선은 취약한 값을 가질 수 있는 경로와 분석된 상태를 나타낸다. 실제 lang.*.php는 외부에서 호출 될 때에만(오른쪽 부분) 취약성을 가지나, 분석에서는 이 결과가 합쳐져서 전달되므로 update.php에 취약성이 있는 것으로 분석됨을 볼 수 있다.

이러한 잘못된 보고의 근본적인 원인은 본 분석이 문맥을 고려하지 않는 분석(context insensitive analysis) [14]을 사용하기 때문이다. 문맥을 고려한 분석을 사용할 경우 분석의 정확도를 높여 이러한 문제를 해결할 가능성은 있으나, 함수 호출의 경로와 파일 삽입 지점에 따라 함수 및 파일에 대한 분석을 모두 개별적으로 수행하여야 하므로 분석 시간 및 소요 메모리가 급격하게 증가할 가능성이 높고 값 분석의 최악의 복잡도도 지수적으로 증가한다[14]. 따라서 요약 해석에 기반한 정적 분석은 일반적으로는 문맥을 고려하지 않는 분석을 대부분 사용하고 있으며, 문맥을 고려한 분석은 제한적으로 사용된다.

<표 2>는 선택된 모든 예제 프로그램들에 대하여 성능 테스트를 수행한 결과이다. 38개 파일에서 58개의 PHP삽입 취약점이 보고되었다. 취약점의 검출 내용을 조사한 결과



(그림 7) authentication_index.php의 취약 부분의 수행 전후 상태



(그림 8) lang.*.php 파일의 취약성이 update.php의 분석 결과에 영향을 주는 경로

〈표 2〉 분석기를 사용한 보안 취약성 분석 및 수행 시간

소프트웨어	파일수 (총 라인수)	분석시간 (sec)	취약점 발견 파일	취약점 보고 개수	허위 탐지 개수
PHPGEDVIEW (Ver. 2.65)	146 (41,321)	16.4	functions.php	1개	0
			authentication_index.php	1개	0
			config_gedcom.php	2개	0
			lang.*.php (14 files)	14개	0
			facts.*.php (15 files)	15개	0
			upgrade.php	2개	2개
Mantis (Ver. 1.0.0rc2)	424 (97,628)	27.39	core.php	12개	0
			bug_actiongroup_page.php	1개	0
			bug_sponsorship_list_view_inc.php	2개	0
			my_view_page.php	2개	0
PhpDig (Ver. 1.8.9-rc1)	64 (14,802)	28.3	config.php	6개	0

CVE 보안 사이트에서 존재하는 것으로 보고되었던 PHP 삽입 취약성이 모두 검출된 것으로 확인되었으며, 전체 58개의 취약점 보고 중 잘못된 보고(false positive)는 PHPGEDVIEW에서 2개만이 발견되어 실용적인 정밀도를 가지고 있음을 알 수 있었다. 검출 시간에 있어서도, 2000라인 당 1초 정도의 분석 시간을 사용하여 실용적인 검사 속도를 가지고 있는 것으로 판단된다.

5. 결 론

본 논문에서는 웹 응용 프로그램 개발에 있어 문제가 될 수 있는 PHP 파일 삽입 취약성을 자동 검출할 수 있는 보안 취약성 분석기를 구현하였다. 구현된 분석기는 요약 해석에 기반한 정적 분석 방법을 사용하여, 사용자에 의한 수행 환경 설정이 필요 없고, 취약성 분석에 있어 안전성이 제공되기 때문에 소스 코드의 보안 취약성 포인트가 누락되지 않는 장점을 가진다. 또한 분석 공간을 설계함에 있어 PHP 언어와 분석 대상 취약성의 특성을 고려하여 수행 시간과 검출 성능을 높이고자 하였다. 또한 구현된 분석기는 자바 GUI 도구를 사용하여 검출된 취약성과 관련 분석 정보를 제공함으로써 프로그래머의 코드 수정이 용이하도록 하였다. 구현된 분석기를 PHP로 작성된 공개 소프트웨어에 적용하여 실험해 본 결과 실용적인 검출 능력과 분석 속도를 가짐을 알 수 있었다.

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, http://www.cgisecurity.com/owasp/html/guide.html, Sep., 2002.

[2] Gartner, "Now is the time for security at Application Level", 2006, 12.

[3] D. Turner, S. Entwisle, "Symantec Internet Security Threat Report Vol.IX - Trends for July 05-December 05," Symantec, March, 2006.
[4] Common Weakness Enumeration, cwe.mitre.org.
[5] Common Vulnerabilities and Exposures, mitre.cve.org.
[6] OWASP Top Ten Project, http://www.owasp.org /index.php/Category:OWASP_Top_Ten_Project.
[7] SANS: CWE/SANS TOP 25 Most Dangerous Software Errors, Http link: http://www.sans.org/top25-software-errors.
[8] CWE-98: Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion'), Http link: http://cwe.mitre.org/data/definitions/98.html
[9] Scott, D., Sharp, R., "Developing Secure Web Applications," IEEE internet Computing, Vol.6, No.6, pp.38-45, Nov., 2002.
[10] Scott, D., Sharp, R. "Abstracting Application-Level Web Security," Proc. 11th Int'l Conf. World Wide Web (WWW2002), pp.396-407, May 17-22, 2002.
[11] Sanctum Inc. "AppShield 4.0 Whitepaper," http://www.sanctuminc.com, 2002.
[12] Kavado, Inc. "InterDo Version 3.0," Kavado Whitepaper, 2003.
[13] 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 (WWW2003), pp.148-159, May 21-25, 2003.
[14] Fortify Software, Http link: http://fortify.com
[15] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers: Principles, Techniques and Tools, Addison Wesley.
[16] Wassermann, Zhendong Su., "Sound and Precise Analysis of Web Applications for Injection Vulnerabilities," In Proceedings of PLDI 2007, pp.32-41, San Diego, CA, June 10-13, 2007
[17] Gary Wassermann and Zhendong Su, "Static Detection of Cross-Site Scripting Vulnerabilities," In Proceedings of ICSE 2008, Leipzig, Germany, May 10-18, 2008.
[18] Flemming Nielson, Hanne Riis Nielson, Chris Hankin, "Principles of Program Analysis." Springer, 452pp, 2005.
[19] Patric Cousot, Radia Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, pp.238-252, LA, California, USA, 1977.
[20] PHP: Hypertext Preprocessor, Http link: http://www.php.net
[21] Gagnon, E. M., Hendren, L. J., "SableCC, an Object-Oriented Compiler Framework," Proc. 1998 Conf. Technology of Object-Oriented Languages and Systems (TOOLS-98), pp. 140-154, Santa Barbara, California, USA, Aug. 3-7, 1998.
[22] Patrick Cousot, "Abstract Interpretation Based Formal Methods and Future Challenges", In Informatics, 10 Years Back - 10 Years Ahead, R. Wilhelm (Ed.), Lecture Notes in Computer Science 2000, pp.138-156, 2001.
[23] Joonsen Ahn, "Differential Evaluation of Fixpoints of Non-distributive Functions", IEICE Transactions on Information and Systems, Vol.E-86-D, No.12, pp.2710-2721, Dec., 2003.

안 준 선

e-mail : jsahn@kau.ac.kr
1992년 서울대학교 계산통계학과(이학사)
1994년 KAIST 전산학과 (공학석사)
2000년 KAIST 전자전산학과 (공학박사(전산학))
2000년~2001년 KAIST 프로그램분석 시스템연구단 연구원
2001년~현 재 한국항공대학교 항공전자및정보통신공학부 교수
관심분야 : 프로그램 분석, 프로그램 보안, 유비쿼터스 컴퓨팅 등

임 성 채

e-mail : sclim@dongduk.ac.kr
1992년 서울대학교 컴퓨터공학과(학사)
1994년 KAIST 전산학과(석사)
2003년 KAIST 전산학과(박사)
2000년~2005년 코리아 와이즈넛 수석연구원 및 이사
2005년~현 재 동덕여자대학교 컴퓨터학과 부교수
관심분야 : 이동객체 색인, 멀티미디어 스토리지, Flash-memory SSD 등