ESCAPE: 자동 익스플로잇 생성 PHP 개체 주입 취약점

박순녀
카이스트

김대준*
카이스트

수만 자나
컬럼비아 대학교

손수엘
카이스트

추상적인

PHP 개체 주입(POI) 취약점은 취약한 PHP 애플리케이션에 존재하는 클래스 메서드의 원격 코드 실행을 허용하는 보안에 중요한 버그입니다. 이 취약점을 악용하려면 주입 개체를 형성하기 위해 정교한 속성 지향 프로그래밍이 필요한 경우가 많습니다 . 기존 의 기성 도구는 익스플로잇 개체 의 존재를 확인하지 않고 잠재적인 POI 취약점을 식별하는 데만 중점을 둡니다 . 이를 위해 POI 취약점에 대한 최초의 AEG(Automatic Exploit Generation) 도구인 FUGIO를 제안한다. FUGIO는 대략적인 정적 및 동적 프로그램 분석 을 수행하여 악용 개체 의 청사진 역할을 하는 가제트 체인 목록을 생성 합니다. 그런 다음 FUGIO는 이러한 식별된 체인을 사용하여 퍼징 캠페인을 실행하고 익스플로잇 개체를 생성합니다. FUGIO 는 오탐지가 없는 알려진 POI 취약점을 포함하는 30개 애플리케이션에서 68개의 익스플로잇 개체를 생성 했습니다. FUGIO 는 또한 5개의 익스플로잇과 함께 이전에 보고되지 않은 2개의 POI 취약점을 발견 하여 기능 익스플로잇 생성에 대한 효율성을 입증했습니다 .

1. 소개

PHP 개체 삽입(POI) 취약점은 보안에 중요한 PHP 응용 프로그램 버그 [46] 로, 교차 사이트 스크립팅, SQL 삽입, 임의 파일 삭제 를 비롯한 다양한 공격을 가능하게 합니다. PHP는 문자열을 PHP 개체로 역직렬화 하여 런타임 개체 관리를 용이하게 하는 기능을 지원합니다. POI 취약점을 악용할 때 공격자는 이 기능을 표적으로 삼습니다. 공격자는 위조된 문자열을 역직렬화 함수 호출에 전달 하여 임의의 PHP 개체를 대상 응용 프로그램 범위에 주입합니다.

이 공격의 독특한 측면은 공격자가 PHP 개체의 속성과 구조를 변경하여 일련의 사용자 정의 함수 또는 클래스 메서드를 호출하여 다양한 유형의 웹 공격을 허용해야 한다는 것입니다. 이 익스플로잇 개체를 구성하는 데 사용되는 기술을 속성 지향 프로그래밍(POP)이라고 합니다.

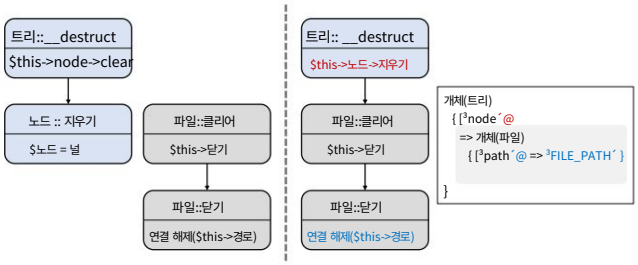


그림 1: POP의 상위 수준 개요

그림 1 은 공격자 가 임의 파일 삭제를 위해 POP를 수행 하는 개략적인 개요를 보여줍니다 . 그림 의 왼쪽 은 대상 애플리케이션이 구현하는 두 개의 합법적인 호출 스택을 보여줍니다. 오른쪽 은 공격자가 대상 애플리케이션이 구현하지 않는 POP를 사용하여 구성하는 새로운 스택 추적을 보여줍니다. 오른쪽에 클래스 개체를 삽입한 후 대상 응용 프로그램은 이 새로운 스택 추적을 실행하여 공격자가 선택한 파일을 삭제합니다. 공격자는 node 속성에 File 클래스 개체가 있는 개체를 주입하여 두 개의 스택 추적을 연결 하고 path 속성 을 변경 하여 파일 경로 를 제공합니다. 이 새로운 스택 추적 을 POP 체인이라고 하며 일련의 사용자 정의 기존 클래스 메서드 또는 함수를 호출합니다.

정제되지 않은 사용자 입력 이 있는 역직렬화 호출의 존재는 단지 잠재적인 위협을 제기하는 반면, 실제 익스플로잇 개체의 존재는 심각한 보안 위협을 제기합니다. 그러나 기존 웹 스캐닝 도구 [32, 50, 58] 는 단지 사용자 입력 을 역직렬화하는 잠재적인 POI 취약점 만 보고합니다. 불행히도 POP를 통해 익스플로잇 개체를 생성 하려면 상당한 인력과 전문 지식이 필요합니다.

그러나 이러한 잠재적인 POI 취약점 의 악용 가능성을 확인하기 위한 실질적인 대안은 없습니다 .

기여. 우리는 POI 취약점을 찾고 식별된 취약점에 대한 익스플로잇을 생성하도록 설계된 최초의 AEG(Automatic Exploit Generation) 도구인 FUGIO를 제안합니다.

AEG for POI 취약점은 두 가지 문제를 해결해야 합니다 .

*두 저자는 논문에 동등하게 기여했습니다.

이러한 가제트의 가용성을 고려하면서
발신자-호출자 관계; 2) 식별된 각 사슬에 대해,
적절하게 설정하여 주입 대상을 형성하지 않아야 합니다.
개체 계층뿐 아니라 여러 속성도 포함하므로
이 POP 체인이 지정하는 실행 흐름을 따릅니다.
우리는 정적 및 동적 분석을 통해 지원되는 피드백 기반 타겟 퍼징
을 사용하여 POI 버그에 대한 AEG를 처리하기로 결정했습니다 .
특히 FUGIO는 다음 두 가지 문제를 모두 해결합니다. 1) FUGIO는 분석
을 통해 모든 POP 체인을 식별합니다.
동적으로 생성된 클래스와 정적으로 정의된 클래스
tar get POI 취약성 을 트리거할 때 사용할 수 없는 가젯을 분류하고 제거
합니다. 2) FUGIO는 익스플로잇 개체를 생성합니다.
다음을 수행하여 식별된 사슬의 실행을 따르는
부풀려진 PHP 애플리케이션에 대한 피드백 기반 퍼징.
대상 PHP 애플리케이션의 소스 코드가 주어지면 FUGIO
를 통해 모든 클래스 및 기능 정보를 수집하여 시작합니다.
사용 가능한 모든 가젯을 식별하는 정적 분석. 또한 수집
동적으로 생성된 클래스와 함수는 POI가
취약점은 동적 분석을 통해 트리거되므로
FUGIO는 포괄적인 POP 체인 세트를 고려합니다.
두 가지 분석을 기반으로 FUGIO는 실현 가능성을 식별합니다.
POP 체인 및 테스트 중인 프로그램(PUT) 생성
모든 가제트를 구현합니다. PUT을 구성할 때 취약점이 있는 실
행 환경을 모방합니다.
발동. 그런 다음 FUGIO는 피드백 기반 퍼징을 수행합니다.
각 체인에 대한 PUT 캠페인을 통해 익스플로잇 생성
사물. 퍼징 프로세스는 실행 피드백을 활용합니다.
가제트에 더 깊이 도달하는 유망한 입력의 우선 순위를 지정할 때
주어진 체인에서 속성 값을 변경합니다. FU GIO가 잠재적인 익스플로잇을
찾으면 공격 페이로드를
주입된 페이로드가 싱크 기능에 나타나는지 확인하기 위해 익스플로잇 개체
의 속성을 확인하여
생성된 페이로드를 악용합니다.
우리는 각각 30개의 PHP 애플리케이션에서 FUGIO를 평가했습니다.
적어도 하나의 알려진 POI 취약점이 있습니다. 이들로부터
응용 프로그램에서 FUGIO는 68개의 악용 가능한 체인과 그
거짓 긍정이 없는 실제 익스플로잇. 또한 FU GIO를 두 가지 PHP
애플리케이션의 최신 버전인 WordPress에 적용했습니다.
WooCommerce [3] 및 Concrete5와 함께. FUGIO는 두 가지를 보고했습니다.
구체적인 익스플로잇으로 이전에 보고되지 않은 POI 취약점을 발견하여 그
효능을 입증했습니다.
요약하면, 우리 는 일련의 정적 및 동적 프로그램을 적용하는 POI 취
약성을 위한 새로운 AEG 도구를 제안합니다.
분석 및 피드백 기반 퍼징. 우리는 그것을 보여줍니다
제안된 기술은 익스플로잇 생성에 효과적입니다.
제로 포지티브(false positive)로 악명 높은 요구 사항
노동 집약적인 엔지니어링 노력.

2 배경

2.1 PHP 객체 주입

PHP 개체 주입(POI) 취약점은 원격 코드 exe를 수반하는 PHP 애플리
케이션의 보안에 중요한 버그를 나타냅니다.

```
1 <?php
2 종 로거 {
#         public function __destruct () { // 매직 메소드"
4             if ($this - > logtype === $this - >log -       임시 ") {
5                 > clear () ;
6             } 다른 {
7                 $this - >log - > 저장 () ;
8 } } }
9 클래스 스트림 {
          공개 함수 지우기 () {
10 $this -> 닫기 () ;
12          }
13          공개 함수 닫기 () {
14              $this - > 핸들 - > 닫기 () ;
15 } }
16 클래스 TempFile 확장 스트림 {
          공개 함수 저장 () {
17 $tmpfile = 임시남 ("/tmp ", "                                XYZ_ ") ;
19              $data = file_get_contents ($this -> 파일명 ) ;
20              file_put_contents ( $tmpfile $data ) ; // 싱크대
21          }
22          공개 함수 닫기 () {
23              연결 해제 ($this -> 파일 이름 ) ; // 싱크대
24 } }
25 $data = 직렬화 해제 ( $_COOKIE ['데이터 ']) ; // POI 버그
```

목록 1: POI 취약점 악용

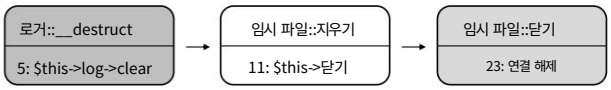
컷 [46]. 공격자는 다음을 악용하는 입력 문자열을 주입합니다.
이 취약점과 취약한 애플리케이션은
이 삽입된 문자열을 역직렬화할 때 PHP 개체입니다. 공격자 는 POP를
통해 이 PHP 개체를 프로그래밍하여
파일 삭제 /생성, 사이트 간 스크립팅, 원격 코드 실행 및

기타 악의적인 행동.

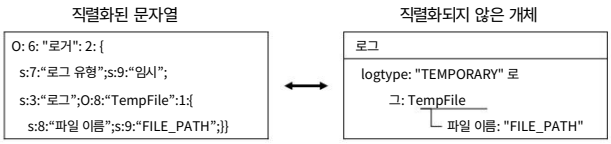
직렬화 해제. PHP는 다음과 같은 두 가지 내장 함수를 지원합니다.
개발자는 객체 유형 데이터를 인코딩 및 디코딩하는 데 자주 사용합니다.
직렬화() 및 직렬화 해제(). 이 메서드는 주어진 객체를 문자열로 직렬화하고

주어진 문자열을 각각 객체에 넣습니다. 직렬화 해제도
PHP 아카이브에서 파일 작업이 수행될 때 발생
(PHAR) 파일(예: file_exists 및 is_file)
파일 형식은 메타 데이터를 직렬화된 형식으로 저장합니다. 따라서
공격자는 런타임에
직렬화 해제 호출 에 입력 문자열을 입력 하거나 공격자가 업로드한
PHAR 파일에서 파일 작업을 트리거 합니다. 을 위한
후자의 경우 위조된 PHAR 파일이 이미
UFU(Unrestricted File Upload) 취약점 을 악용하여 대상 웹 서버에
업로드되었습니다 [40]. 적
따라서 이 업로드된 파일에서 PHP 파일 작업을 호출하려고 시도합니다.
메타 데이터를 직렬화 해제하는 PHAR 파일.

속성 지향 프로그래밍. POI를 악용할 때
취약점, 공격자는 신중하게 주입 개체를 만듭니다.
기존 클래스의 체인을 호출하기 위해 속성 값 선택
방법 또는 기능. 이 체인은 시작되는 호출 스택을 반영합니다.
항목 메소드에서 시작하고 다음을 호출하는 메소드로 끝납니다.
공격 페이로드가 있는 보안에 민감한 기능. 이 체인
POP 체인이라고 하며 이 POP 체인의 각 방법은 다음과 같습니다.
가제트라고 합니다.
임의의 객체를 주입한다고 해서 항상 코드가 실행되는 것은 아
닙니다. 실행하려면 PHP
취약한 애플리케이션을 실행하는 인터프리터는 다음을 호출해야 합니다.



(a) 링크 해제를 유발하는 POP 체인



(b) 2a 에 대한 익스플로잇 개체

그림 2: 목록 1 의 POI에 대한 POP의 예

POP 체인의 항목 가젯. 이를 달성하기 위해 공격자 는 취약한 응용 프로그램에 정의된 마법의 방법을 사용합니다.
런타임에 특정 조건이 충족되면 다양한 유형의 매직 메서드가 자동으로 호출 됩니다. 예를 들어 __destruct 메서드는 이 매직 메서드를 구현 하는 클래스 정의에서 객체를 삭제할 때 호출됩니다 .

공격자는 또한 주입된 개체의 속성을 조정하여 이전 가젯의 본문 내에서 다음 가젯을 호출해야 합니다. 다음 가젯은 1) 이전 가젯을 포함하는 클래스의 멤버 메소드, 2) 이전 가젯의 호출 문에서 이름이 사용된 다른 클래스의 멤버 메소드 또는 3) 다음을 포함하는 사용자 정의 함수여야 합니다. 소유자 클래스가 없습니다.

목록 1 은 25 행 에 POI 취약점이 있는 PHP 스니펫을 보여줍니다 . 공격자는 "데이터" 쿠키 를 통해 임의의 개체를 삽입할 수 있습니다. 그림 2a 는 이 취약점을 악용하기 위해 공격자가 고안한 POP 체인을 보여줍니다 . 이 체인은 Logger 개체의 __destruct에서 시작하여 23 행 에서 보안에 중요한 파일 삭제가 발생하는 TempFile 개체의 닫기 멤버 메서드로 끝납니다 . 이 체인 을 기반 으로 공격자 는 그림 의 왼쪽에 직렬화된 문자열을 구현합니다. 2b; 이 문자열은 값이 "TEMPORARY"로 설정된 logtype 속성과 값이 TempFile 개체 로 설정된 log 속성을 포함 하는 Logger 개체 로 역직렬화 됩니다. 이 객체는 파일이름 속성을 가지고 있으며 그 값은 그림 2b 의 오른쪽 과 같이 삭제할 파일 경로로 설정되어 있습니다.

이 객체가 런타임에 소멸되면 해당 __destruct 메서드가 자동으로 호출 됩니다. logtype 속성 값이 "TEMPORARY"이므로 __destruct 는 TempFile의 clear 를 호출 하며 , 이는 순차적으로 TempFile 의 닫기 를 호출하고 filename 에 구현된 대상 파일 이름과의 연결 을 해제 합니다.

따라서 공격자는 앞서 언급한 입력 문자열을 주입하여 임의의 파일을 삭제할 수 있다. 공격자 는 Logger 의 log 속성에 TempFile 개체를 할당 하여 Line 5 의 호출 사이트를 TempFile의 clear 멤버 메서드인 두 번째 가젯에 연결 합니다 . 이 제어 흐름은 응용 프로그램 개발자가 의도한 것이 아니라 공격자가 도입한 것입니다. 공격자 는 새로운 코드를 도입하지 않고 대신 개체의 계층 구조를 재구성하고 적절한 속성 값을 설정하여 기존 가젯을 재사용합니다.

제시된 공격은 return-to-libc [44], 반환 지향 프로그래밍 [52], 점프 지향 프로그래밍 [9] 과 같은 코드 재사용 공격 기술과 유사합니다 . 그러나 POP에서 코드 실행을 위한 기본 블록이 클래스 멤버 메서드라는 점에서 다릅니다 .

3 동기와 도전

우리는 POI 취약점을 찾고 이 취약점을 악용하는 주입 개체를 생성 하기 위한 AEG 접근 방식을 제안합니다 . Burp [50] 및 Acunetix [32] 와 같은 기존의 기성 침투 테스트 도구는 사용자 입력을 역직렬화하고 잠재적인 POI 취약성 을 보고 하는 내장 호출 사이트를 식별하는 데만 집중 합니다. 악용 개체의 존재는 가능한 오탐을 제거할 수 있습니다. 그러나 이 작업에는 상당한 수작업과 전문 지식이 필요한 POP가 필요합니다.

Dahseet al. [13] 은 유망한 POP 체인을 보고 하는 정적 분석을 수행하여 이 한계를 해결했습니다 . 그러나 가양성을 제거하기 위해 이 정적 접근 방식은 여전히 유망한 POP 체인 중 하나를 악용할 적절한 입력을 찾는 도달 가능성 분석 문제를 해결 해야 합니다.

또한, 유망한 체인의 수가 증가함에 따라 각 체인을 검증하고 익스플로잇을 생성하는 것이 힘든 작업이 됩니다. 예를 들어 Contao CMS에는 최소 26,180개의 체인이 있으므로 이러한 체인을 수동으로 확인하고 5개의 악용 가능한 체인에 대해 작동하는 익스플로잇을 생성하는 것은 불가능합니다.

POI 취약성에 대해 AEG를 수행하려면 다음 목표를 달성해야 합니다. 1) POI 취약성을 찾습니다. 2) 식별 된 취약점 을 트리거할 때 사용 가능한 가젯의 POP 체인을 식별 합니다. 3) 식별된 것들 중에서 악용 가능한 POP 체인에 대한 입력 개체를 생성하여 취약점의 악용 가능성을 보고합니다.

앞서 언급한 정적 및 펜 테스트 도구 [13, 32, 50, 58] 가 사용자 입력을 역직렬화 함수 호출로 전달하는 잠재적 POI 취약점을 이미 찾을 수 있다는 점을 고려하여 잠재적 POI 버그 감지의 기여를 강조하지 않습니다 . 오히려 우리의 핵심 기여는 두 번째 및 세 번째 목표 를 해결하는 데 필요한 익스플로잇 개체 를 생성하는 데 있습니다. 아래에서 각 목표의 기술적 과제를 설명합니다.

POP 체인 식별. AEG 도구는 사용자 입력이 역직렬화되는 위치에서 로드 가능한 모든 클래스의 가젯을 고려하는 모든 POP 체인을 식별해야 합니다.

(C1-1) PHP의 동적 특성으로 인해 로드 가능한 클래스와 해당 가젯 을 식별하기 어렵습니다 . 자동 로드 기능을 사용하면 개발자가 지정한 로드 콜백 을 호출하여 기존 클래스를 로드할 수 있습니다 . 또한 많은 PHP CMS 응용 프로그램은 동적으로 생성된 PHP 클래스에 크게 의존합니다.

따라서 정적으로 정의된 클래스만 고려하면 POP에 사용할 수 있는 가젯을 고려하지 않고 거짓 부정을 생성합니다.

(C1-2) 로드 가능한 가제트를 연결하기 위한 순진한 알고리즘은 악용 가능성 을 평가할 때 검사할 엄청난 수의 POP 체인을 생성합니다 . 예를 들어 목록 1에서 __destruct

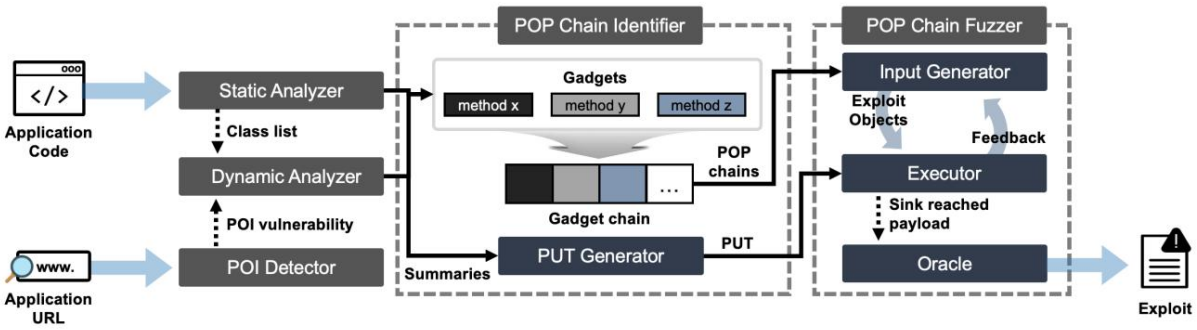


그림 3: FUGIO 아키텍처: POI 취약성에 대한 AEG의 워크플로 개요

clear 및 save 를 호출하는 두 개의 호출 사이트가 있습니다. 이 __destruct 항목 가젯 에서 공격자는 $ this- >log 를 조작 하여 clear 또는 save 라는 이름의 다른 가젯에 연결할 수 있습니다 . 따라서 가제트 내의 호출 수가 증가함에 따라 주어진 가제트에서 연결할 수 있는 후보의 수가 증가합니다. 이것은 특히 POP 체인의 수가 길이가 증가함에 따라 기하급수적으로 증가 하기 때문에 긴 체인을 열거할 때 문제가 됩니다 .

익스플로잇 생성. AEG 도구는 중단 없이 식별된 체인의 익스플로잇 실행을 가능하게 하는 여러 속성 값으로 익스플로잇 개체를 생성해야 합니다.

이것은 애플리케이션이 목표 명령문에 도달할 수 있도록 하는 적절한 입력을 생성하는 어려운 문제를 제기합니다 .

본문에 여러 조건을 포함하는 가제트를 고려하십시오. 다음 가젯의 호출문이 이러한 조건을 전달하는 것과 관련된 경우 도구는 이러한 조건 을 전달하기 위해 적절한 속성 값을 생성해야 합니다. 예를 들어 목록 1에서 Logger::__destruct 의 다음 가젯 이 TempFile ::clear인 경우 주입할 개체 의 로그 유형은 "TEMPORARY"여야 하고 로그 는 TempFile 클래스 개체 여야 합니다. 또한 도구는 속성 값을 삽입하여 마지막 가제트에서 보안에 민감한 싱크의 실제 매개변수에 공급할 공격 페이로드도 제공해야 합니다.

익스플로잇 생성을 위해 피드백 기반 퍼징을 선택합니다.
기호 실행은 의심할 여지 없이 익스플로잇 개체를 체계적으로 생성하는 데 적용할 수 있습니다 [4, 11, 67]. 그러나 이 접근 방식을 사용하려면 기호 측면에서 PHP 내장 함수의 의미 체계를 모델링해야 하므로 많은 엔지니어링 노력이 필요합니다. 벤치마크의 30개 애플리케이션이 평가에서 식별된 약 30만 개의 가젯에서 460개의 내장 PHP 기능을 사용 하는 것을 관찰했습니다 . 또한 PHP 인터프리터가 발전함에 따라 기호 실행을 위해 더 많은 내장 함수를 지원해야 합니다. 따라서 우리 는 위음성의 위험 을 수반하는 더 일반적인 퍼징 접근 방식을 선택합니다 .

퍼징. 대상 PHP 애플리케이션을 퍼징하는 것은 독특한 기술적 문제를 야기합니다.

(C2-1) 대용량 PHP 응용 프로그램에서 Stateless 퍼징을 수행할 때 높은 처리량을 설정하기가 어렵습니다.
대상 PHP 애플리케이션의 크기가 클 때 실행 생성된 각 입력이 있는 응용 프로그램은 느린 프로세스로 인해 퍼징 성능이 저하됩니다. 벤치마크에서 30개의 PHP 애플리케이션은 하나의 객체 입력을 실행하는 데 평균 0.6초가 소요되었으며 , 이는 최첨단 fuzzer [10] 보다 훨씬 느립니다 . 대상 응용 프로그램 이 POI 취약점과 관련이 없는 모듈도 실행 하기 때문 입니다. 또한 이러한 모듈은 내부 프로그램 상태를 변경하거나 많은 수의 퍼징 시도로 인해 대상 응용 프로그램을 실행할 수 없도록 만드는 부작용을 일으킬 수 있습니다 . 마지막으로 각 퍼징 시도 후 애플리케이션을 초기 상태 로 복원하는 데 시간이 많이 걸립니다 .

(C2-2) 퍼징을 통해 여러 속성 값을 가진 익스플로잇 개체를 고안하는 것은 간단하지 않습니다 . 방대한 양 의 기존 퍼징 문헌은 퍼징 입력을 바이트 스트림으로 모델링합니다. 그러나 POI 취약점에 대한 익스플로잇 개체를 생성하려면 속성 지향 프로그래밍을 수행해야 하며, 이를 위해서는 여러 속성을 변경 하고 POP 체인을 실행할 수 있도록 개체 계층 구조를 형성해야 합니다. 익스플로잇 개체에 필요한 속성을 어떻게 식별할 수 있습니까?

돌연변이를 위해 어떤 속성을 선택해야 합니까? 선택한 속성 값을 어떻게 변경하고 생성 해야 합니까? 퍼즈 테스팅 을 통해 익스플로잇을 생성하려면 이러한 질문을 해결해야 합니다 .

4 개요

대상 PHP 애플리케이션에서 POI 취약점을 탐지 하고 익스플로잇 객체를 생성하여 탐지된 취약점 의 익스플로잇 가능성을 확인 하는 시스템인 FUGIO를 제안한다 .

앞서 언급한 두 가지 문제를 해결하기 위해 FUGIO는 정적 및 동적 프로그램 분석을 모두 수행하여 정적으로 선언되고 동적으로 생성된 모든 가젯(C1-1)을 통합합니다 . 그런 다음 FUGIO는 절차 간 오염 분석(불필요한 POP 체인 제거 목적)을 수행하고 모든 유망한 체인을 열거하는 깊이 제한 너비 우선 검색 을 수행합니다(C1-2).

FUGIO는 피드백 기반 퍼징을 수행하여 악용을 생성하여 두 번째 문제를 해결합니다. 퍼즈 테스트 의 높은 처리량을 달성하기 위해 FUGIO는 실행 환경을 시뮬레이션하는 테스트 중인 프로그램(PUT)을 합성합니다.

POI 취약점이 트리거되는 곳(C2-1). FUGIO 는 테스트 중인 POP 체인 (C2-2)에서 더 깊은 가젯에 도달하는 측면에서 더 유망한 입력 개체를 고안하기 위해 퍼징 중 조건식에 나타나는 분기 적용 범위, 런타임 참조 오류 및 힌트를 활용합니다 .

FUGIO는 POI 감지기, 정적 분석기, 동적 분석기, POP 체인 식별자 및 POP 체인 fuzzer의 다섯 가지 구성 요소로 구성됩니다. 이러한 구성 요소는 함께 작동하여 POI 취약성을 찾고 악용 개체를 생성합니다. 그림 3 은 FUGIO의 작업 흐름을 보여줍니다. 대상 PHP 응용 프로그램 소스 코드와 URL을 입력으로 사용합니다. POI 탐지기는 POI 취약점을 탐지하기 위해 웹 사이트를 크롤링하기 시작하고 탐지된 취약점을 동적 분석기로 전달합니다. 정적 분석기와 동적 분석기는 각각 의 취약점이 발견되면 주어진 소스 코드와 실행 환경에서 POP 체인을 식별하기 위한 데이터를 수집합니다 . 이러한 분석을 사용 하여 POP 체인 식별자는 유망한 POP 체인을 계산하고 PUT을 생성합니다. 식별된 각 체인에 대해 fuzzer는 구체적인 익스플로잇을 찾을 때까지 이 PUT에 대해 퍼지 테스트를 수행합니다.

5 디자인

5.1 POI 감지기

POI 감지기는 동적 테스트를 통해 대상 PHP 애플리케이션에서 잠재적인 POI 취약성을 감지합니다. 특히 사전 정의된 입력 문자열을 가져와 PHP 개체로 변환하는 삽입 지점을 동적으로 감지 합니다.

URL이 있는 대상 PHP 애플리케이션이 주어지면 감지기 는 애플리케이션을 크롤링하고 방문할 URL 목록을 계산하는 것으로 시작합니다. 이 목록의 각 웹페이지를 방문할 때 감지기는 <a> 및 <form> 태그를 추출합니다. 각 <a> 태그 에서 감지기는 대상 URL에 대한 GET 요청 템플릿을 빌드합니다. 각 <form> 태그에서 감지기는 각각 키와 기본값 이 있는 액션, 메소드 및 입력 매개변수를 추출 합니다. 그런 다음 감지기는 이러한 구성 요소를 조합 하고 입력 매개변수가 있는 요청 템플릿을 생성합니다.

따라서 각 요청 템플릿에서 감지기 는 입력 문자열을 사용하여 테스트 요청 목록을 생성합니다. 이 요청 템플릿의 각 GET, POST 및 COOKIE 매개변수에 대해 감지기는 테스트 PHP 개체 를 나타내는 미리 정의된 직렬화된 문자열을 주입하여 테스트 요청 세트를 생성합니다. 이러한 각 요청에서 하나의 입력 매개변수는 사전 정의된 문자열을 보유합니다. 그런 다음 감지기는 각 테스트 요청을 보내고 사용자 입력을 역직렬화하는 PHP 내장 호출 사이트가 테스트 개체와 함께 호출되는지 관찰합니다. 이를 위해 FUGIO는 runkit [14] 또는 uopz [34] 를 사용하여 unserialize, is_file 및 file_exists 와 같은 사용자 입력의 역직렬화와 관련된 26개의 미리 정의된 PHP 내장 함수를 연결합니다.

간단히 말해서, 이러한 내장 함수 중 하나가 테스트 개체를 보유하는 실제 매개변수와 함께 호출되면 탐지기 는 이 호출을 잠재적인 POI 취약성으로 보고하고 이 호출 사이트 정보를 동적 분석기에 전달합니다.

5.2 정적 분석기

정적 분석기 는 POP 체인 식별자 (§ 5.4) 로 POP 체인 및 PUT 생성에 나중에 사용되는 사용자 정의 클래스 및 기능에 대한 정적 요약을 계산합니다.

분석기는 대상 응용 프로그램의 소스 코드를 가져와 각 파일에 대한 정적 요약을 출력합니다. 이 정적 요약은 이 파일 에 정의된 함수 및 클래스의 정보를 보유하는 함수 요약 및 클래스 요약 을 포함합니다. 각 PHP 파일을 AST(추상 구문 트리) 집합으로 구문 분석한 다음 기능 , 클래스, 인터페이스 및 특성에 대한 모든 정의를 분석합니다. 정의된 각 클래스에 대해 정적 분석기는 이름, 이름 속도, 상위 클래스, 구현된 인터페이스, 사용된 특성 및 정의 자체를 포함한 정보를 추출합니다. 그런 다음 이 정보를 클래스 를 정의하는 PHP 파일의 클래스 요약에 저장합니다. 이 요약 에는 정의된 각 속성 및 멤버 메서드 의 이름과 가시성도 포함됩니다 .

함수 또는 멤버 메서드 정의를 구문 분석할 때 정적 분석기는 함수 프로토타입을 추출하고 호출자-호출 수신자 관계를 포함하는 함수 요약을 계산합니다.

함수 또는 멤버 메서드 정의 내의 각 메서드 호출(예: $receiver->method()) 에 대해 분석기는 대상 메서드 이름과 해당 수신기 클래스 후보를 추출합니다.

호출이 명시적으로 $this를 사용하는 경우 정적 분석기는 수신자를 대상 메소드 또는 이 소유자의 상위 클래스의 소유자로 정확하게 유추합니다 . 그렇지 않으면 대상 메서드 이름을 가진 멤버 메서드를 포함하는 클래스 집합을 수신기 후보에 할당하여 가능한 모든 클래스의 보수 집합을 계산합니다. 그러나 new 키워드를 포함하는 새로 인스턴스화된 클래스의 할당 문에서 이 수신기로의 절차 내 데이터 흐름이 있는 경우(예: $receiver = new ClassA), 분석기는 이 클래스를 수신기로 사용합니다. 각 파일 요약 의 이 호출자-호출자 정보 는 나중에 가제트를 연결 하여 POP 체인을 구축할 때 사용됩니다.

각 멤버 메서드 및 함수에 대해 정적 분석기 는 형식 매개 변수 및 소유자 속성 에서 본문의 각 호출 호출 사이트 인수 까지 흐름에 둔감한 절차 내 데이터 흐름을 계산합니다 . 기능 요약 에는 이러한 데이터 흐름이 포함되어 있으며, 이는 나중에 공격이 주입 개체의 속성 값을 조정하여 후속 가젯의 실제 인수를 변경할 수 없는 악용할 수 없는 중간 가젯 을 제거하는 데 사용됩니다.

5.3 동적 분석기

동적 분석기 는 대상 애플리케이션 소스 코드 에서 정적으로 정의되지 않은 동적으로 생성된 클래스 및 함수에 대한 함수 요약 및 클래스 요약을 계산합니다. 이러한 요약은 나중에 체인 및 PUT 생성을 위한 POP 체인 식별자에서도 사용됩니다.

POI 취약점이 있는 경우 동적 분석기는 정적 분석기가 수집할 수 없는 추가 정보를 수집합니다.

얻다. 이를 위해 FUGIO는 runkit [14] 또는 uopz [34] 확장 을 사용하여 unserialize, file_exists 및 fopen 과 같은 사용자 입력을 내부적으로 역직렬화하는 PHP 내장 함수에 대한 후크를 설치하는 PHP 파일을 생성합니다 . .htaccess 파일을 사용하여 이 파일을 대상 웹 애플리케이션에 삽입 합니다.

주어진 POI 취약점이 트리거되면 설치된 후크가 정적 분석기 를 통해 얻은 모든 클래스를 로드하려고 시도합니다 . PHP는 언로드된 클래스 에 액세스할 때 개발자 지정 콜백을 호출 하는 자동 로드 기능을 지원합니다. 많은 PHP 응용 프로그램이 이 자동 로드 기능을 사용하기 때문에 동적으로 로드할 수 있는 클래스를 아는 것이 중요합니다 . 주어진 클래스가 로드 되었는지 확인하기 위해 class_exists 를 사용 합니다. 그렇지 않으면 이 함수 는 해당 자동 로드를 사용하여 지정된 클래스를 자동으로 로드하려고 시도합니다. 모든 클래스에 대해 class_exists 를 호출 하여 로드 가능한 클래스 목록을 얻습니다 .

동적 분석기는 본체가 정적으로 정의 되지 않은 경우 로드된 함수, 클래스, 인터페이스 및 특성을 후속적으로 검사합니다 . 분석 과정은 정적 분석기에서 수행되는 분석과 동일합니다 . 그러나 동적 분석기 는 소스 코드에 존재하지 않는 동적으로 정의된 클래스와 함수를 검사 합니다. Dahse et al. [13] 은 이러한 동적으로 생성 된 함수나 클래스를 고려하지 않았습니다.

마지막으로 동적 분석기는 이러한 후킹 메서드가 호출될 때 환경 변수 (예: $_ENV 및 $_SERVER) 와 전역 변수(예: $_GLOBAL) 를 저장합니다. 다음 단계에서 이러한 변수는 주어진 POI 취약성이 트리거 되는 실행 환경을 모방하는 PUT을 생성하는 데 사용됩니다 .

5.4 POP 체인 식별자

정적 및 동적 분석기의 정보를 기반으로 POP 체인 식별자는 사용 가능한 POP 체인 목록 과 퍼지 테스트를 수행하는 데 사용할 PUT를 내보냅니다.

5.4.1 POP 체인 식별

POP 체인은 대상 POI 취약점의 악용이 발생할 때 매직 방식에서 민감한 싱크로의 스택 추적을 반영하는 일련의 가젯입니다 . 이 민감한 싱크 의 유형에 따라 공격자는 다양한 유형의 공격을 수행할 수 있습니다. 이 문서에서는 파일 생성/수정/삭제, 셸 명령 주입 및 원격 코드 실행 을 유발 하는 총 26개의 민감한 싱크를 지정했습니다 .

민감한 싱크의 각 호출에 대해 FUGIO는 공격자가 호출 을 구현하는 함수 또는 메서드 (m) 의 기능 요약을 사용하여 호출 의 실제 인수를 변경할 수 있는지 여부를 확인합니다 (§ 5.2). m 의 형식 매개변수 또는 m의 소유자 클래스 속성 에서 절차 내 데이터 흐름이 있는지 확인 합니다 . 그렇다면 FUGIO는 POP 체인을 생성할 때 이 싱크 기능을 고려합니다. 그렇지 않으면 FUGIO는 이 싱크 기능을 제외합니다.

공격자가 개체 주입을 통해 이러한 인수를 직접 변경할 수 있는 방법이 없기 때문입니다 . 예를 들어 목록 1에서 FUGIO 는 각각 save 및 close 메소드 의 절차 내 데이터 흐름 분석을 확인하여 두 개의 민감한 싱크인 file_put_contents (20행) 와 unlink (23 행)를 찾습니다 . 그런 다음 FUGIO 는 파일 이름 속성에서 file_put_contents의 두 번째 인수와 unlink의 첫 번째 인수로 의 데이터 흐름이 있는지 확인합니다.

체인 식별자는 POP 체인을 생성하는 데 필요하며, 각 체인은 매직 메소드에서 각 대상에 민감한 싱크 까지의 경로를 따라 가젯으로 구성됩니다 . 이에 대한 한 가지 순진한 솔루션 은 각 마술 방법 에서 대상에 민감한 싱크까지 깊이 우선 검색을 수행하여 가능한 모든 체인을 생성하는 것입니다. 그러나 엔트리 매직 메서드에서 호출 수신자의 다운스트림을 순회하면 민감한 싱크 가 발생하지 않아 계산 리소스가 낭비될 수 있습니다. 또한 콜 체인에 사이클 이 존재하는 경우 알고리즘이 종료되지 않습니다.

대신, 우리는 그림 4 와 같이 너비 우선 방식으로 깊이 제한 호출 트리를 구축하는 알고리즘을 설계했습니다 . 특히, 대상 싱크의 각 호출에 대해 FUGIO의 체인 식별자는 호출 트리를 계산하며, 그 루트는 다음과 같습니다 . 호출을 구현하는 메서드. 그런 다음, 함수 또는 메서드(m)를 나타내는 각 리프 노드에 대해 새 노드를 반복적으로 연결하며, 각 자식 노드는 m의 잠재적 호출자에 해당합니다. 체인 식별자 는 높이가 감사자가 지정하는 매개변수인 지정된 높이까지 자랄 때까지 이 트리를 빌드합니다. 평가 에서 이 매개변수로 7개를 선택했습니다 (§ 7.4). 이 높이 제한 트리 검색을 통해 FUGIO는 호출 가제트에 주기가 있는 경우에도 POP 체인을 열거할 수 있습니다.

리프 노드를 연결합니다. 가제트 (m) 의 잠재적 호출자를 각 리프에 연결할 때 체인 식별자는 개발자가 m 에 대해 의도한 진정한 호출자 뿐만 아니라 공격자가 개체 속성을 조작 하여 m 에 적용하는 위조 호출자를 고려합니다. 특히, 체인 식별자는 1) 대상 호출자 이름이 m 과 동일하고 2) 실제 매개변수 수가 m의 형식 매개변수 수 와 동일한 호출 문이 있는 모든 함수 요약을 수집합니다 . 이 수집된 함수 요약 집합은 첨부할 m 의 잠재적 호출자 집합이 됩니다. 잠재적 호출자에서 체인 식별자는 호출 문이 m 의 소유자 클래스를 나타내지 않는 정적으로 결정적인 수신자를 갖는 호출자를 추가로 제거합니다 . 정적 분석을 통해 FUGIO는 이미 각 호출 (§ 5.2)에 대해 보수적인 수신기 세트를 계산했음을 상기하십시오.

체인 식별자는 m의 잠재적 호출자를 연결할 때 정적 정보를 활용한다는 점을 고려 하면 잠재적 호출자가 정적으로 결정할 수 없는 경우 호출 에지를 놓칩니다. $receiver->$method 와 같은 반사 호출 사이트 의 경우 체인 식별자는 가능한 호출 수신자를 결정할 수 없으므로 이 호출 에지를 포함하는 체인이 누락됩니다. § 7.3.2에서 이러한 반사 호출 사이트로 인한 위음성에 대해 논의 합니다.

체인 생성. 각 싱크에 대한 호출 트리를 구축한 후,

체인 식별자는 마법을 나타내는 리프 노드를 찾습니다.
행동 양식. 식별된 각 리프 노드에 대해 경로를 계산합니다.
루트 노드에 연결하여 트리 노드 체인을 방출합니다.
POP 가젯에 해당합니다.
식별된 POP 체인을 다음 단계로 전달하기 전에
체인 식별자는 절차 간 데이터 흐름 분석을 수행하여 데이터 흐름 이 없는 체인을 제거합니다.
민감한 싱크의 실제 인수에 대한 가젯 속성,
공격자가 이러한 실제를 변경하는 것을 불가능하게 만듭니다.
인수. 각 가젯의 기능 요약을 활용합니다.
이 POP 체인에 있으며 절차 간 데이터를 계산합니다.
POP 체인 내에서 흐릅니다.

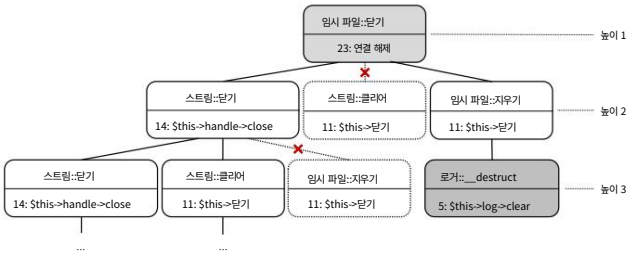


그림 4: 목록 1 의 링크 해제에서 호출 트리

5.4.2 PUT 생성

FUGIO는 상대적으로 작은 PHP에서 PUT을 합성합니다.
원래 응용 프로그램보다 있지만 악용을 생성하는 데 필요한 모든 가젯
이 포함되어 있습니다. FUGIO는 먼저 모든 클래스를 추출하고
분석기가 요약에서 함수 정의
계산하다. 클래스와 기능의 충돌을 피하기 위해
이름, FUGIO는 각 정의에 대해 하나의 파일을 내보냅니다. 우리는 이것을
생성된 파일 정의 파일.
그런 다음 FUGIO는 PUT의 머리와 몸 부분을 생성합니다.
헤드 부분에서 FUGIO는 모든 환경과 글로벌을 설정합니다.
동적 분석기에서 제공하는 변수. 후지오 그럼
다음을 포함하는 모든 정의 파일을 포함하도록 본문 부분을 씁니다.
악용할 때 로드되고 로드할 수 있는 모든 가젯
타겟 POI 취약점. 본문 부분 은 직렬화 된 입력 문자열을 받는 unserialize
호출로 끝납니다.
fuzzer는 나중에 제공합니다. 이 PUT은 이 입력 문자열을 받습니다.
실행 중인 PHP 인터프리터를 호출하는 쉘 명령에서
풋. 따라서 fuzzer는 다음 을 사용하여 직렬화 해제를 호출합니다.
실행을 시뮬레이션하는 이 PUT에서 생성된 입력
POI 취약점을 악용할 때의 환경.
피드백을 위한 도구. 피드백 기반 퍼징을 수행하기 전에 FUGIO
는 주어진 PUT를 계측하여 다음을 얻습니다.
각각의 퍼즈 테스트를 수행할 때 3가지 종류의 피드백
개체 입력: 1) 실행된 내용이 포함된 실행 추적
조건식 및 호출된 메서드, 2) 숫자
POP 체인에서 실행된 가젯의 수 및 3) 속성에 대한 힌트
조건식의 값.
첫 번째 및 두 번째 유형의 피드백을 얻기 위해
FUGIO는 매번 실행을 보고하는 코드를 삽입합니다.
모든 조건식 전후에 함수 입력,

알고리즘 1: 퍼징

```
입력 : 계장된 PUT(넣기)
       A POP 체인(pop_chain)
출력 : 페이로드

1 시드_풀 ←[]
2 init_seed ← GenerateInitialSeed(pop_chain)
3 AddSeed(seed_pool,init_seed)
4 동안 참 _
5     시드 ← SelectSeed(seed_pool)
6     입력 ← MutateSeed(시드)
7     실행 ← ExecutePUT(put,input)
8     result ← AnalyzeExecution(실행)
9     NewPath(결과) 인 경우
10        AddSeed(seed_pool,input)
11        SinkReached(결과) 인 경우
12            보고서(입력, "잠재적으로 악용될 수 있음")
13            Oracle(입력) == 성공 이면
14                보고서(입력, "악용 가능")
15                부서지다
16        또 다른
17            ExecutedGadgets (입력) >
                그런 다음 ExecutedGadgets(시드)
18                AddSeed(seed_pool,
                    GenerateNextGadgetSeed(입력))
19            Hinted_properties ← AnalyzeConditions(결과)
20            foreach 소품 ∈ hinted_properties 는
21                AddSeed(seed_pool,
                    GenerateHintedSeed(입력, 소품))
22    그렇지 않으면 참조 오류가 발생 하면
23        missing_property ← AnalyzeError(실행)
24        AddSeed(seed_pool,
            AddProperty(input,missing_property))
```

사용자 정의 함수의 모든 호출 문에서.
특히 각 계측 코드는 다음의 해시 값을 내보냅니다.
그 줄과 파일, 따라서 실행 주석을 남깁니다.
이 단계에서는 속성에 대한 힌트를 보고하는 코드도 삽입합니다.
fuzzer가 정확한 속성 값을 생성하기 위해 활용하는 값. 목록 1에서 $this->logtype 을 다음으로 설정해야 합니다.
"TEMPORARY"를 사용하여 지우기 를 호출하지만 (5행), 이 특정 문자열을 무작위로 생성하는 것은 불가능합니다. 이를 위해 in strumented 코드는 상수 및 유형 검사 내장을 보고합니다.
is_string() 및 is_int( ) 와 같은 함수
실행된 조건문에서. 이 보고된 상수
내장 함수는 나중에 속성 값을 생성하는 데 사용됩니다 . 이러한 유형의 속성 생성 속성 힌트를 호출합니다.

5.5 POP 체인 퍼저

PUT 및 POP 체인이 주어지면 fuzzer는 PUT에서 피드백 기반 퍼징을 수행하여 익스플로잇 개체를 생성합니다.
알고리즘 1 은 퍼징의 전체 프로세스를 설명합니다. 주어진
POP 체인 및 PUT, fuzzer 는 주어진 시간 동안 퍼징 캠페인을 시작합니다. 퍼징 프로세스가 시작됩니다.
초기 종자 풀을 준비하여(1-3행). 그런 다음 반복됩니다.
주입 대상을 찾을 때까지 다음 절차

공격 페이로드로 POP 체인에서 민감한 싱크를 실행합니다. fuzzer는 이전 실행 피드백을 기반으로 시드를 선택하고(라인 5) 이 시드를 변경하여 테스트할 새 주입 개체를 생성합니다(라인 6). 그런 다음 fuzzer는 변경된 시드로 계측된 PUT를 실행 하고 실행 피드백을 분석합니다(7-8행). 이 돌연변이된 시드 가 분기 적용 범위를 늘리는 데 기여하면 fuzzer는 이 시드를 시드 풀에 추가합니다(9-10행). 또한, 이 돌연변이된 시드가 기존 시드보다 더 많은 가젯을 실행하는 경우 fuzzer는 POP 체인에서 한 단계 더 깊은 대상을 대상으로 하는 새로운 시드도 파생합니다(17-18행). 또한 상수를 활용 하고 실행된 조건문에 나타나는 속성 의 사용을 기반으로 속성 유형을 유추하여 현재 돌연변이된 시드에 속성 값을 할당하여 새 시드를 계산 합니다(19-21행). 나머지 섹션에서는 퍼징 프로세스의 각 단계를 자세히 설명합니다.

종자 최적화. 우리는 돌연변이할 시드를 선택할 때 체인의 마지막 가제트 에서 보안에 민감한 싱크 또는 주어진 체인의 더 깊은 가젯에 도달하는 시드 입력을 우선시합니다 .

fuzzer는 빈 시드 풀(라인 1) 을 준비하여 시작 하고 초기 시드 입력을 생성합니다. 이 초기 시드 입력은 클래스가 지정된 POP 체인 의 첫 번째 가젯을 보유하는 개체입니다 . 그런 다음 이 초기 시드가 시드 풀에 추가됩니다(2-3행). 그런 다음 fuzzer는 시드 풀(라인 5)에서 시드를 선택합니다. 각 시드에 대해 fuzzer는 1) 시드 선택 수, 2) 속성 트리의 해시, 3) POP 체인에서 실행된 가젯의 최대 깊이와 같은 실행 결과를 저장합니다. 이 정보를 기반으로 fuzzer는 실행 횟수가 더 많은 가젯 이 있는 덜 자주 실행되는 시드의 우선 순위를 지정합니다 .

보다 형식적으로, 우리 는 POP 체인의 길이와 POP 체인 에서 실행되는 가제트 의 최대 깊이 사이의 차이로 diff 를 정의합니다. 그런 다음 방정식 1 을 사용 하여 대상 가젯 깊이에 대한 확률을 할당합니다. 실행된 가젯의 깊이가 클수록 대상 깊이가 선택될 확률이 높아집니다.

$$\text{점수} = \frac{1}{\frac{5\times dif}{1+e^{\max(cur\_depth)}}}, \quad \text{파이} = \frac{\text{스코어리}}{\Sigma\text{점수}} \tag{1}$$

대상 가젯 깊이를 선택한 후 fuzzer는 실행에서 이 목표 깊이에 도달한 시드 풀의 시드 중 하나 를 선택합니다.

시드 풀에 싱크에 도달하는 시드가 포함된 경우 FUGIO는 1.0의 확률을 0.9와 0.1로 나눕니다. 그런 다음 싱크에 도달한 시드에 0.9의 확률을 균일하게 분배합니다 . 0.1의 확률은 풀의 나머지 시드에도 균일하게 분포됩니다. 싱크에 도달하는 시드 가 없으면 fuzzer 는 대상 가젯 깊이를 공유하는 각 시드에 동일한 확률을 할당합니다.

또한 이 시드 선택의 수가 증가함에 따라 각 시드의 확률이 감소하므로 덜 선택된 시드에 대해 선택될 확률이 높아집니다.

입력 생성. 주입 개체를 생성하려면 1)이 필요합니다.

```
1 <?php 2 클
래스 로거 { 공개 $logtype ; 공개
#           $log ; function setProp
4           ($name , $value )
5           { $parent_class = get_parent_class () ; if ( $parent_class
6               && property_exists ( $parent_class 부모 :: setProp ($name ,
7               $value ) ;
8                                                                    ,    $이름 ) ) {
9
10              }
11              $this - > $name = $value ;
12          }
13          function getProp ( $name ) { return $this
14              - > $name ;
15 } } 16 클
래스 스트림 { 17
            ...
18 } 19
클래스 TempFile 확장 스트림 { 공개 $filename ; 20

21              ...
22 } 23
$input = 새로운 로거 ; 24 $input - >
setProp (' 로그 유형 ', "                                    TEMPORARY ") ; 25
$input - > setProp ('log ', new TempFile ) ; 26 $input - > getProp ('log ')

27                      -> setProp ('파일명 ', "                    FILE_PATH ") ; 28
echo base64_encode ( 직렬화 ( $input ) ) ;
```

목록 2: 페이로드 템플릿의 예

주어진 POP 체인을 반영하는 여러 클래스의 구조적 계층 구조 설계 및 2) 이러한 여러 클래스의 속성에 적절한 값을 할당 하여 공격 페이로드로 민감한 싱크에 도달하는 것을 용이하게 합니다. 이 주입 개체 를 생성하고 변경할 때 속성 트리라는 트리 데이터 구조를 활용했습니다 . 루트 노드는 해당 클래스가 매직 메소드를 보유하고 있는 클래스 객체, 즉 주어진 POP 체인의 엔트리 가젯을 나타냅니다. 하위 노드의 각 노드는 클래스 개체 속성을 나타냅니다. 여기에는 속성 이름, 가시성, 유형 및 값이 포함됩니다. 속성 유형이 클래스 개체인 경우 이 노드는 속성 이 가질 수 있는 클래스 후보를 보유합니다. 또한 이 노드는 POP 체인에 다른 가젯이 있는 다른 클래스 개체를 나타내는 하위 트리 의 부모 노드가 됩니다.

fuzzer는 속성 트리를 생성하고 개체를 인스턴스화하는 PHP 파일을 생성하여 이 트리 를 주입 개체로 변환합니다. 이 PHP 파일에서 FUGIO는 이 트리에서 사용되는 모든 클래스를 정의하고 속성 값을 설정하고 생성된 클래스를 Listing 2와 같이 직렬화합니다 . 이 파일을 실행하면 fuzzer 가 입력을 생성하고 PUT에 제공됩니다.

fuzzer는 정의된 클래스를 이 PHP 파일에 주입하지 않습니다. 이러한 클래스는 도우미 클래스이며 직렬화된 주입 문자열을 생성하기 위해서만 설계되었습니다. 주입된 직렬화된 문자열 은 실제로 대상 PHP 애플리케이션의 기존 클래스를 악용합니다.

돌연변이. fuzzer는 선택된 seed의 속성 트리를 변경합니다(Line 6). fuzzer는 속성 트리의 각 속성을 방문하여 유형을 확인합니다. 이 유형이 Object일 때 fuzzer 는 이 속성 의 후보 클래스에서 무작위로 하나의 클래스를 선택 합니다. 그렇지 않으면 fuzzer는 문자열, 정수, 부울, 파일, 배열 및 참조 중 하나의 PHP 유형을 무작위로 선택합니다 .

문자열, 정수, 부울 및 파일 형식의 경우 fuzzer는 다음과 같습니다.

선택한 유형의 임의 값을 이 속성에 서명합니다. 배열 속성의 경우 fuzzer는 배열 크기를 임의로 설정하고 배열의 키와 값에 임의의 값을 할당합니다.
참조 속성의 경우 fuzzer는 소유자와 정적 및 동적 분석기가 계산하는 기타 속성을 식별합니다 . 그런 다음 다른 속성 중 하나를 임의로 선택하고 해당 참조를 대상 참조 속성에 할당합니다. 속성을 변경한 후 속성 트리는 PHP 파일로 변환되고 fuzzer 는 이 파일에서 생성된 입력으로 계측된 PUT를 실행합니다(라인 7).

피드백. fuzzer는 주어진 입력의 실행 결과를 활용 하여 주어진 POP 체인의 민감한 싱크에 도달할 더 유망한 후보인 새 시드를 유도 하는 피드백 기반 대상 퍼징을 수행합니다 . fuzzer가 활용하는 피드백에는 1) 분기 적용 범위, 2) 도달한 가젯 깊이, 3) 속성 힌트, 4) 참조 오류의 네 가지 유형이 있습니다.

분기 커버리지의 경우 fuzzer는 이 돌연변이 입력이 새로운 분기를 덮을 때 시드 풀 에 돌연변이 시드를 추가합니다 (9-10행). 두 번째 유형의 피드백의 경우 fuzzer는 가제트 깊이를 활용합니다. 돌연변이된 시드의 실행이 POP 체인 에서 이 돌연변이된 시드의 원래 시드가 도달한 것보다 더 깊은 새 가제트의 실행을 가능하게 할 때 fuzzer는 이 돌연변이된 시드를 도달한 가젯의 업데이트된 깊이와 함께 시드 풀에 추가합니다. (17-18행).

is_string(), is_int(), is_array () 와 같은 특정 내장형 검사 함수나 비교 연산자를 사용하여 조건문에서 사용되는 속성을 관찰할 때 fuzzer는 추론된 유형 또는 상수 피연산자를 다음 위치에 저장합니다. 속성에 대한 이 조건문. 각 힌트 속성에 대해 힌트 속성의 값이 추론된 값으로 설정되거나 추론된 유형에 의해 무작위로 변형되는 힌트된 시드를 생성합니다 (19-21행).

게다가, fuzzer는 리시버(예: $receiver->method()) 를 활용한 메소드 호출의 참조 오류를 관찰합니다 . 관찰된 오류가 수신기의 누락된 속성 또는 잘못된 개체 로 인해 발생하는 경우 fuzzer는 현재 입력의 트리에 누락된 속성 노드를 추가 하거나 대상 메서드 호출이 있는 클래스 중에서 선택한 값을 Object 에 할당합니다. 이름 (22-24행).

오라클을 이용합니다. fuzzer 는 생성된 입력 개체가 POI 취약점을 악용할 수 있는지 여부를 확인하기 위해 익스플로잇 오라클을 활용합니다. 돌연변이된 시드가 민감한 싱크에 도달하면 fuzzer는 주어진 POP 체인이 생성된 입력으로 악용될 수 있다고 보고합니다(11-12행).

악용 가능성이 있는 것으로 식별된 개체의 악용 가능성을 확인하기 위해 fuzzer는 생성된 입력 이 민감한 싱크의 실제 인수를 제어할 수 있는지 여부를 확인합니다. 오라클은 입력 객체의 각 속성 값을 싱크 인수와 비교 하여 fuzzer 가 페이로드를 주입하는 후보 속성 집합을 유도합니다 . 각 후보에 대해 fuzzer는 속성 값을 대상 민감한 싱크에 따라 달라지는 공격 페이로드로 설정합니다. 예를 들어 유형이
민감한 싱크의 에코는 XSS 공격을 허용할 수 있으며 공격 페이로드는 <script>alert(1);</script>로 설정됩니다.
민감한 싱크로 인해 파일이 삭제되는 경우 공격 페이로드를 기존 파일 경로로 설정합니다.

마지막으로 fuzzer는 인젝션 개체가 인수에 공격 페이로드를 사용하여 민감한 싱크를 호출하는지 여부를 확인 합니다. FUGIO는 PHP에서 26개의 민감한 싱크 호출을 연결하고 각 싱크가 공격 페이로드를 포함하는 실제 매개변수로 호출되는지 확인합니다. 그렇다면 fuzzer는 생성된 페이로드로 주어진 POP 체인을 악용할 수 있다고 보고 하고 퍼징 캠페인을 종료합니다(13-15행). Listing 2 는 생성된 익스플로잇 객체를 정의하고 이 객체의 직렬화된 문자열을 인쇄 하는 PHP 스니펫인 최종 출력을 보여 줍니다 . FUGIO는 또한 탐지기가 POI 취약점 (§ 5.1) 을 유발하는 것으로 발견 한 원래 요청의 GET, POST 또는 COOKIE 입력 매개변수에 이 직렬화된 문자열을 채워 공격 HTTP(S) 요청을 생성할 수 있습니다.

요약하면, fuzzer는 공격 문자열을 생성하고 이 입력으로 PUT를 실행하며, 이는 나중에 악용 개체로 역직렬화됩니다. Exploit oracle은 이 주입 개체가 주어진 POP 체인에서 일련의 가젯을 호출하고 공격 페이로드와 함께 싱크 기능을 호출하는지 여부를 확인합니다.

관리자. FUGIO는 POP 체인 식별자와 POP 체인 fuzzer를 병렬로 실행합니다. POP 체인 식별자 가 각 민감한 싱크를 방문하는 동안 일련의 체인을 계산하면 FUGIO 는 각 체인에 대해 퍼징 프로세스를 호출하여 퍼징 캠페인을 관리합니다 . 따라서 POP 식별자 가 싱크에 대한 POP 체인을 계산하는 동안 FUGIO는 다른 싱크에서 생성된 체인에서 퍼징을 시작할 수 있습니다.

FUGIO는 퍼징할 POP 체인을 선택할 때 길이가 짧은 POP 체인을 우선시합니다. 우리는 익스플로잇 개체를 생성할 때 짧은 POP 체인을 선호하기 위해 이 퍼징 정책을 구현했습니다. 이와 같이 FUGIO 는 더 많은 계산 자원을 필요로 하는 긴 체인을 확인하기 전에 짧은 체인의 익스플로잇 개체가 발견되면 해당 싱크에 대한 퍼징 프로세스를 종료합니다. POP 체인 식별자와 fuzzer를 실행하기 위한 CPU 코어의 비율을 3:1로 설정했습니다. POP 체인 식별자가 완료되면 모든 CPU 코어 가 fuzzer에 할당됩니다.

6. 구현을 피하라

FUGIO는 Python 및 PHP의 20K+ LoC에서 구현됩니다.
PHP 내장 함수를 후킹하여 개체 주입 지점을 식별할 때 runkit [14] 및 uopz [34]를 사용했습니다. 그러나 이러한 확장은 후킹 평가를 지원하지 않으므로 동적으로 생성된 함수 및 클래스 를 추출하기 위해 이 기능 을 구현했습니다. 이 후킹 기능을 통합하기 위해 모든 PHP 파일을 실행하기 전에 .htaccess 파일을 사용하여 후킹을 구현하는 PHP 파일을 추가했습니다 .

또한 정적 분석 및 PUT 계측을 위해 PHP 파일을 AST로 구문 분석할 때 PHP-Parser [45] 를 활용했습니다.
서로 다른 모듈 간의 통신을 위해 우리는 Rab bitMQ [63]를 사용했습니다. 열린 과학과 추가 연구를 지원하기 위해 우리는

탈출 릴리스 : https://github.com/WSP-LAB/RUN .

7 평가

FUGIO는 익스플로잇 개체 생성의 효율성을 측정하고(§ 7.2) 이전 연구 및 오픈 소스 도구 (§ 7.3) 와 성능을 비교하여 평가 합니다. 그런 다음 몇 가지 매개변수 가 기능적 익스플로잇 생성에 영향을 미치는 정도를 보여줍니다 (§ 7.4). 우리는 또한 이전에 보고되지 않은 POI 취약점 과 그 익스플로잇을 찾는 데 FUGIO를 평가합니다 (§ 7.5). 보고된 익스플로잇 개체 (§ 7.6) 로 사례 연구를 시연 합니다.

7.1 실험 설정

30개의 PHP 애플리케이션에서 FUGIO를 평가했습니다. 각 앱에 대해 알려진 POI 취약점을 준비 하여 파일 삭제/수정/생성, 명령 실행 또는 원격 코드 실행 을 유발하는 취약점을 유발하는 익스플로잇 개체를 생성하는 FUGIO 작업 을 수행했습니다. 30개의 응용 프로그램 중 8개의 응용 프로그램은 Dahse et al. 평가에 사용 [13]. PHPGGC [2]의 21개 애플리케이션도 포함했습니다. PHPGGC에서 12개의 패키지는 PHP 라이브러리입니다.

그래서 각 라이브러리를 이용하여 간단한 PHP 애플리케이션을 만들고 이 애플리케이션에 POI 취약점을 주입했습니다. 나머지 9개 애플리케이션에 대해 알려진 POI 취약점 기능인 CVE-2018-20148, CVE-2019-6339 및 [64]를 활용했습니다. 이러한 벤치마크 의 선택 기준은 1) 취약한 버전의 애플리케이션에 여전히 액세스할 수 있고 2) 취약한 애플리케이션의 크기가 작지 않습니다.

환경. 우리는 2.10GHz 및 384GB RAM의 Intel Xeon Gold 6238 CPU 2개가 장착된 Linux 워크스테이션에서 평가를 수행했습니다 . FUGIO 실행을 위해 PHP 버전별로 Docker 컨테이너를 준비 하고 해당 버전에 따라 웹 애플리케이션을 설치했습니다.

7.2 회피의 수행

30개의 애플리케이션 각각에 대해 5개의 퍼징 캠페인을 수행했습니다 . 각 퍼징 캠페인은 12시간 동안 지속되었습니다. 각 체인에 대해 최대 100초 동안 퍼징을 수행 하고 최대 7개의 장치를 조립하도록 FUGIO를 설정했습니다. § 7.4에서 이러한 매개변수에 대한 FUGIO의 민감도를 평가합니다 .

표 1 은 평가 결과를 요약한 것이다. 세 번째 열은 식별된 POP 체인의 수를 나타내고, 네 번째 열은 익스플로잇을 생성하기 위해 퍼징을 수행한 POP 체인의 수를 나타냅니다. Covered sinks 열은 감지된 POP 체인에서 고유한 민감한 싱크의 수를 나타냅니다.

Exploitable chains 열은 익스플로잇 가능한 POP 체인의 수를 나타내며, 이는 FUGIO가 PHP 개체 익스플로잇을 생성할 수 있음을 의미합니다. 아마도 악용 가능한 체인 열은 싱크 기능에 도달하는 데 성공한 생성된 입력 개체가 있는 체인의 수를 나타냅니다.

그러나 FUGIO 는 생성된 익스플로잇이 익스플로잇 오라클을 통과하지 못하여 익스플로잇 가능성을 확인할 수 없었습니다.
이 열의 각 셀은 5번의 퍼징 시도의 중앙값을 나타냅니다. 최소값과 최대 값은 대괄호 안에 있습니다. 또한 괄호 안의 숫자는 5번의 퍼징 시도 동안 보고된 고유 체인의 수를 나타냅니다 . True positive chains 열은 우리가 수동으로 확인한 악용 가능한 체인의 수를 나타냅니다.

더하기 기호의 왼쪽과 오른쪽에 있는 숫자는 각각 익스플로잇 가능한 (E) 체인과 아마 익스플로잇 가능한 (PE) 체인의 트루 포지티브 체인을 나타냅니다. 이 두 숫자의 합은 5번의 퍼징 시도 동안 FUGIO가 생성한 총 악용 가능한 개체의 수를 산출합니다. 마지막 두 열은 각각 POP 체인을 식별하고 FUGIO를 실행하는 데 소요된 시간을 보여줍니다.

30개의 애플리케이션에서 FUGIO는 총 68개의 E 체인을 보고했습니다. 우리는 이러한 E 사슬을 수동으로 확인했고 FUGIO가 거짓 긍정을 산출하지 않았음을 확인했습니다. 66개의 PE 체인 중 실제로 악용 가능한 체인은 26개였습니다. FUGIO 는 27개의 취약한 애플리케이션 각각에서 익스플로잇 개체를 생성했습니다.

FUGIO는 GLPI, Vanilla 및 Yii에 대한 익스플로잇 생성에 성공하지 못했습니다. GLPI에서 FUGIO는 실제로 악용 가능한 체인이 없기 때문에 E 체인 을 찾지 못했습니다 . 이것은 공격자가 입력 개체를 주입할 수 있지만 GLPI의 취약점을 악용할 수 없음을 의미합니다. 바닐라에서 FUGIO는 LFI 취약점을 유발하는 악용 가능한 체인 하나를 놓쳤습니다. FUGIO 는 컴퓨팅 체인 에 대한 LFI 싱크를 지원하지 않기 때문 입니다. Yii에서 FUGIO는 악용 가능한 체인을 식별했지만 fuzzer는 이에 대한 악용 개체를 생성할 수 없었습니다. 이 익스플로잇을 생성하려면 적절한 인덱스와 해당 클래스 객체를 포함하는 배열 값 을 익스플로잇 개체의 속성에 할당해야 합니다. FUGIO는 퍼징 제한 시간 내에 이러한 조건을 충족하는 익스플로잇 개체를 생성 할 수 없습니다.

FUGIO는 1,637개의 민감한 싱크를 유발한 약 1천만 개의 POP 체인을 식별했습니다. 확인된 체인 범위는 0(Vanilla)에서 320만 체인(Joomla)입니다. 이러한 통계는 악용 개체 생성을 자동화해야 할 필요성을 보여줍니다.

우리는 FUGIO가 26개의 악용 가능한 체인을 PE 체인으로 보고한 이유를 추가로 분석했습니다. 1) 공격자가 직렬화된 문자열로 파일 자원을 주입할 수 없기 때문에 FUGIO는 fwrite 와 같은 파일 자원을 사용하는 싱크가 있는 PE 체인 에서 익스플로잇 오라클을 수행하지 않습니다. 그러나 우리는 파일 이름이 주입된 기존 fopen 호출 사이트를 활용하여 9개의 PE 체인을 악용할 수 있음을 발견했습니다. 2) 익스플로잇 오라클 은 주입된 페이로드가 손실 없이 타겟 싱크의 실제 인수에 나타날 때만 E 체인 을 보고합니다. 그러나 공격자 가 매개변수에 공격 문자열을 부분적으로 삽입 할 수 있는 경우에도 4개의 체인이 악용될 수 있었습니다. 나머지 13개 체인에 대해 FUGIO는 지정된 퍼징 제한 시간 내에 공격 페이로드를 주입하기 위해 개체 속성을 정확히 찾아낼 수 없었습니다.

표 1: FUGIO 평가: 악용 가능하고 악용 가능한 체인의 수는 기능적 체인의 수를 나타냅니다.
FUGIO가 생성하는 개체를 악용합니다(WP: WordPress).

PHP 버전	애플리케이션	확인됨 쇠사슬	퍼지 쇠사슬	덮음 싱크대	악용 가능 쇠사슬	아마 악용 가능 쇠사슬	진실 긍정적인 쇠사슬	체인 분석 시간	총 시간
PHP 5.4	콘타오 CMS	26,180	25,732	41	3 [3 – 3] (3) 2 [2 – 7] (8)		3 + 2	117m 49초	시간 초과
	피윅	445,384	14,739	40	0 [0 – 0] (0) 0 [0 – 4] (4)		0 + 1	407분 24초	시간 초과
	GLPI	544,776	8,898	8	0 [0 – 0] (0) 0 [0 – 0] (0) 0 [0 – 1]		0 + 0	시간 초과	시간 초과
	줌라	3,292,647	9,075	47	(1) 0 [0 – 5] (5)		1 + 0	시간 초과	시간 초과
	큐브카트	30	28	11	1 [1 – 1] (1) 0 [0 – 0] (0)		1 + 0	7초	1분 51초
	간편한 CMS	16	16	13	1 [0 – 1] (1) 0 [0 – 1] (0) 11 [9 – 12]		1 + 0	6초	1분 49초
	웹 분석 열기	886	765	18	(12) 5 [4 – 6] (5) 12 + 4 0 [0 – 0] (0) 0 [0 – 0] (0) 1			45초	16분 52초
	바닐라 포럼	0	0	0	[0 – 2] (2) 1 [0 – 2] (3) 1 [1 – 1] (1) 0 [0 – 0] (0)		0 + 0	12초	12초
	스위프트메일러 5.0.1	13,387	12,891	16	0 [0 – 1] (1) 3 [2 – 3] (8) 0 [0 – 1] (1) 1 [0 – 2]		2 + 2	5분 28초	289분 30초
	스위프트메일러 5.1.0	14,875	14,875	16			1 + 0	7분 45초	330분 37초
	똑똑한	15	15	7			1 + 0	4초	1분 57초
	젠드프레임워크	1,271,410	8,732	264			1 + 0	시간 초과	시간 초과
PHP 5.6	PHPExcel 1.8.1(w/WP)	2,787	2,679	37	3 [3 – 3] (4) 1 [1 – 1] (2) 3 [3 – 3]		4 + 0	1분 31초	55분 49초
	PHPExcel 1.8.2(w/WP)	3,333	2,677	37	(3) 1 [1 – 1] (1) 0 [0 – 0] (1) 3 [0 – 3] (3) 1 [0 – 1] (1) 1 [1 – 1] (1) 0 [0 – 0] (0)		3 + 0	1분 56초	56분 10초
	Dompdf (w / WP)	639,904	8,350	115			1 + 0	시간 초과	시간 초과
	총구(WP 포함)	80,285	27,948	43			3 + 1	47분 20초	시간 초과
	WooCommerce 2.6.0(WP 포함)	3,857	3,747	28			1 + 0	23초	76분 18초
	WooCommerce 3.4.0(WP 포함)	158,636명의 이메일	12,004	27	0 [0 – 1] (1) 0 [0 – 0] (0)		1 + 0	311분 56초	581분 25초
	구독자(WP 포함)	1,844	1,793	28	1 [1 – 1] (1) 0 [0 – 0] (0) 1 [1 – 1]		1 + 0	13초	37m
	EverestForms(w/WP)	2,081	2,032	25	(1) 0 [0 – 0] (0)		1 + 0	14초	42분 4초
PHP 7.2	TCPDF	2	2	2	2 [2 – 2] (2) 0 [0 – 0] (0) 1 [0 – 1]		2 + 0	4초	13초
	Drupal7	15	15	11	(1) 4 [4 – 5] (4) 1 [1 – 1] (1) 2 [1 – 3] (5) 1 [1 – 2] (3) 1 [0 – 3] (5) 1 [0 – 2] (6) 0 [0 – 1] (1)		1 + 0	7초	2분 2초
	SwiftMailer 5.4.12	14,977	14,977	16			1 + 4	5m 7초	330m 16초
	스위프트메일러 6.0.0	13,495	13,175	16			3 + 4	4분 45초	286분 25초
	독백 1.7.0 독백	147	144	34			6 + 1	3초	3분 36초
	1.18.0 독백 2.0.0 라	4,525	4,230	55	2 [2 – 4] (8) 1 [0 – 2] (4)		8 + 4	12초	84분 16초
	미나	11,842	11,701	17	0 [0 – 0] (0) 0 [0 – 1] (1) 2 [2 – 2]		0 + 1	2분 36초	262분 2초
		3,254	3,254	4	(2) 0 [0 – 0] (0)		2 + 0	3분 18초	68분 1초
	이것	2,428,535	9,033	453	0 [0 – 0] (0) 0 [0 – 2] (2)		0 + 0	시간 초과	시간 초과
	오타3	1,073,189	8,751	208	0 [0 – 5] (7) 0 [0 – 6] (7)		7 + 2	시간 초과	시간 초과
총		10,052,313	222,278	1,637	68	66	68 + 26		

7.3 최첨단 도구와의 비교

FUGIO의 성능을 성능과 비교했습니다.
그 Dahse et al. § 7.3.1 및 PHPGGC의 보고
§ 7.3.2 . Dahseet al. 보고하는 정적 도구를 제안했습니다.
악용 가능한 체인 [13]. 한편, FUGIO는
악용 가능한 체인뿐만 아니라 악용 개체도 포함됩니다. 에
§ 7.3.1, 우리는 FUGIO에 대한 세부적인 비교를 수행합니다.
Dahse et al. 보고했다.

PHPGGC는 특정 버전의 알려진 익스플로잇 체인에 대해서만 PHP 익스플로잇 개체 를 생성하는 오픈 소스 도구입니다.
PHP 애플리케이션 [2]. PHPGGC는 알려진 악용 가능한 POP 체인을 나열하기 때문에 이러한 체인을 사용하여 false를 측정했습니다.
FUGIO가 § 7.3.2에서 생성하는 네거티브. 참고로, 달리
PHPGGC, FUGIO는 gad get을 조합하고 모든 PHP 애플리케이션에 대한 익스플로잇 개체를 보고하기 위한 일반 AEG 도구입니다.

실험 설정. 각 애플리케이션에 대해 FUGIO를 실행했습니다.
12시간의 타임아웃으로 5번. 우리는 FUGIO를
각 체인에 대해 최대 100초 동안 퍼징을 수행합니다. 에
§ 7.3.1, 길이가 있는 POP 체인을 식별하도록 FUGIO를 설정합니다.
Dahse et al. 사용된. § 7.3.2 에서 우리는
길이가 최대 9인 체인으로 간주됩니다.
PHPGGC에 나열된 최대 악용 가능한 체인 길이입니다.

7.3.1 Dahse et al.과의 비교

대상 클래스. Dahseet al. 개체인지 여부를 분석했습니다.
스택을 정적으로 검사하여 주입 클래스를 로드할 수 있습니다.
포함된 파일의 [24]. 그러나 최소한 존재하는 경우
애플리케이션에서 하나의 자동 로더 콜백, 그들은 다음과 같이 가정합니다.
모든 기존 클래스를 로드할 수 있으며 더 이상 유효하지 않습니다.
이 버그가 PHP 5.4.24에서 패치되었고

표 2: 악용 가능한 POP 체인 수 및 해당
FUGIO 및 Dahse et al. 보고했다.

도구	Contao	Piwik	GLPI	Joomla	CubeCart	CMSMS	OWA	바닐라	토탈
탈출 11		1	0	2	1	1	17	0	33
[13] †	14	삼	0	삼	1	1	5	0	27

† SQLi, XXE 및 LFI 공격을 수행하는 POP 체인은 제외됩니다.

5.5.8 [12, 59, 60]. 반면에 FUGIO는 로드 가능 여부를 확인합니다.
모든 기존 자동 로더 콜백을 동적으로 호출 하여 가용성을 확인함으로써
클래스를 생성합니다. 공정한 비교를 위해,
다음과 같은 경우 모든 사용자 정의 클래스를 사용 가능한 가젯으로 간주합니다.
동일한 가정을 하는 하나 이상의 자동 로더가 존재합니다.
사용 가능한 가제트의 경우.
민감한 싱크. Dahseet al. 감지된 체인 분류
6가지 취약점 유형: 파일 삭제(FD), 파일 생성
(FC), 파일 수정(FM), SQL 인젝션(SQLi), 로컬 파일
포함(LFI) 및 XML 외부 엔터티 주입(XXE).
익스플로잇을 생성하기 위해 총 26개의 민감한 싱크를 지정했습니다.
FD, FC, FM 및 명령 주입 취약성의 경우
부록 11.1에 나와 있습니다. LFI 및
XXE는 이러한 취약점을 재현할 수 없기 때문입니다.
PHP 5.4 [61, 62]. SQLi의 경우 POP 식별 가능
쇠사슬; 그러나 fuzzer는 민감한 싱크에 도달할 수 없습니다.
이러한 체인에는 데이터베이스 계정 또는 인스턴스가 필요하기 때문에
이미 데이터베이스에 연결되어 있습니다. 우리는 이것을 남겨
미래의 일.

표 2 는 FUGIO 가 익스플로잇 객체를 식별하고 성공적으로
생성 한 E 체인의 수를 비교한 것입니다.
Dahse et al. 보고했다.
FUGIO는 6개에서 33개 E 체인 에 대한 익스플로잇 개체를 생성했습니다.
Dahse et al. 확인된 27개의 E 사슬
동일한 응용 프로그램. Dahse et al. 세부 사항 생략
각 체인의 악용 가능한 각 체인을 일치시킬 수 없습니다.
따라서 우리는 FUGIO가 사용하는 익스플로잇 개체의 수를 비교합니다.
그들의 논문에 보고된 숫자로 보고되었습니다.

Contao , Piwik 및
Joomla, FUGIO는 다음으로 인해 익스플로잇 개체를 생성할 수 없습니다.
다음 이유: 1) 싱크 기능에 도달하거나 제어하기 위해 복잡한 조건을 통
과해야 하는 4개의 체인
이러한 싱크 함수의 인수; 2) 2개의 체인이 필요합니다.
특정 OS 환경 및 특정 파일의 존재 또는
싱크 기능에 도달하는 디렉토리. 이러한 제한은
퍼지 테스트의 불건전한 특성에서; 우리는 더 많은 것을 믿습니다
고급 퍼징 최적화 및 계산 리소스
거짓 부정의 수가 감소합니다.

Dahse et al.의 정적 접근 방식에 유의하십시오. 보고된 10
정적으로 해결되지 않은 호출 사이트 방해로 인한 가양성
그들의 오염 데이터 흐름 분석 [13]. 이는 감사인이
다음을 포함한 모든 보고서의 악용 가능성을 확인해야 합니다.
거짓 긍정. 대조적으로, FUGIO는 거짓 양성을 보고하지 않았습니다.
생성된 객체로 PUT을 실행하고 확인하기 때문에
익스플로잇 오라클을 사용한 익스플로잇 가능성.

표 3: 다음에서 발견한 악용 가능한 POP 체인 비교
FUGIO 및 PHPGGC에 등재됨(WP: WordPress)

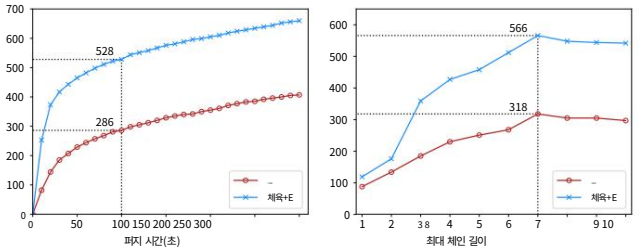
애플리케이션	PHPGGC	나는 도망친다		
	모두 다 아는 쇠사슬	감지됨 쇠사슬	익스플로잇	새로운 익스플로잇
SwiftMailer 5.0.1	1	1	0	1
SwiftMailer 5.1.0	1	1	0	삼
Smarty	2	1	1	0
ZendFramework	4	2	1	0
PHPExcel 1.8.1(WP 포함)	5	5	5	1
PHPExcel 1.8.2(w/WP)	5	5		1
Dompdf 0.8.0(w/WP)	1	1	5 0	0
총구(WP 포함)	5	4	2	2
WooCommerce 2.6.0(WP 포함)	1	1	1	0
WooCommerce 3.4.0(WP 포함)	1	1	1	0
이메일 구독자(WP 포함)	1	1	1	0
EverestForms(w/WP)	1	1	1	0
TCPDF 6.3.2	1	1	1	1
Drupal7	2	1	1	1
SwiftMailer 5.4.12	1	1	1	6
SwiftMailer 6.0.0 모노로	1	1	0	11
그 1.7.0 모노로그 1.18.0	2	2	0	1
모노로그 2.0.0 라미나	1	1	0	삼
	1	1	0	0
	1	1	1	1
이것	1	1	0	0
총	39	34	22	32

7.3.2 PHPGGC와 비교

추가 설정. FUGIO는 다음과 같은 경우 싱크에 대한 퍼징을 종료합니다.
fuzzer는 다른 싱크를 호출하는 다양한 체인을 탐색하기 위해 적어도 하나
의 익스플로잇 (§ 5) 을 생성합니다. 그러나 PH PGGC는 동일한 싱크를 공
유하는 여러 체인을 나열합니다. 따라서 fuzzing 프로세스가 종료되기 전에
종료되지 않도록 설정합니다 .
해당 싱크에 대해 열거된 모든 체인을 퍼징합니다.

표 3 은 의 능력에 대한 실험 결과를 보여줍니다.
FUGIO는 21개의 PHPGGC 애플리케이션에서 익스플로잇 개체를 생성합니다 .
표의 두 번째 열은 숫자를 나타냅니다.
PHPGGC가 나열하는 알려진 악용 가능한 체인. 세 번째
열은 FUGIO가 식별한 가제트 체인의 수를 나타내고 네 번째 열
은 이러한 식별된 체인에 대해 생성된 익스플로잇 수를 나타냅니
다. 마지막 열
FUGIO가 사용하는 새로운 악용 가능한 체인의 수를 보여줍니다.
익스플로잇 개체와 함께 발견되었습니다. 악용 가능한 39개의 경우
PHPGGC의 체인, FUGIO는 34개의 체인과 22개의 익스플로잇을 보고했습니다.
이러한 체인에 대한 개체입니다. 또한, FUGIO는 32개의 새로운
익스플로잇 가능한 체인 및 익스플로잇 개체, 총 생성
54개의 악용 가능한 체인 및 해당 악용 개체.

PHPGGC의 총 39개 체인 중 FUGIO는 34개의 POP 체인을 식별했습니
다. 5가지 근본 원인을 분석했습니다.
거짓 부정. ZendFramework, Guzzle 및 Drupal7에서 FU GIO는 이러
한 응용 프로그램이 사용하기 때문에 4개의 체인을 찾지 못했습니다 .
두 개의 가제트를 연결하는 반사 호출. 예를 들어, 언제
Zend의 세 번째 가젯에서 다음 가젯 연결하기



(a) 퍼징 타임아웃 (b) 최대 체인 길이

그림 5: 퍼징 시간 초과 및 최대 체인 길이를 변경하는 동안 생성된 POP 체인 수(E: 악용 가능, PE: 악용 가능).

프레임워크 체인, 세 번째 가젯의 $view->$helper 는 Zend_Cache_Frontend_Function::call을 호출해야 합니다.
이 호출의 대상 호출 대상은 $helper 에 대한 가능한 문자열 값을 추상화하지 않는 한 정적으로 결정할 수 없습니다 .
반사 호출 사이트에서 대상 호출 수신자를 식별하는 것은 정적 분석 분야에서 해결되지 않은 문제로 남아 있습니다 . Dahse et al. [13] 같은 결과가 나왔을 것이다. 대안은 이 호출 사이트의 모든 가젯을 고려하는 것입니다. 그러나 이 정책 으로 인해 금지된 수의 체인이 생성됩니다.

나머지 하나의 거짓 부정은 우리 범위에 없는 PHP 내장 가젯을 활용합니다. FUGIO는 대상 응용 프로그램에서 사용자 정의 가젯을 조립하는 데 중점을 둡니다.

34개의 식별된 체인 중 FUGIO는 22개의 E 체인에 대한 익스플로잇 개체를 생성했습니다. 체인이 누락된 이유는 두 가지였습니다. 1) 익스플로잇 개체를 구축하는 데 9개의 체인이 필요했으며 , 그 중 일부 속성에는 다른 개체 를 보유하는 배열 값이 있었습니다 . 이러한 객체는 FUGIO가 적절한 인덱스와 해당 객체를 보유하는 배열 값을 객체 속성에 할당해야 했기 때문에 FUGIO는 주어진 퍼징 시간 제한 내에 이러한 복잡한 객체를 생성할 수 없었습니다.

2) 나머지 3개 체인에 대해 FUGIO는 다른 체인 을 테스트하기 위한 시간 예산이 부족하여 퍼즈 테스트를 수행하지 않았습니다. 이 세 가지 체인에 대해 퍼지 테스트를 수행할 때 FUGIO는 기능적 익스플로잇을 생성했습니다.

우리는 FUGIO가 PHPGGC에 나열되지 않은 새로운 32개의 E 체인을 보고했음을 강조합니다. 이러한 결과는 FUGIO 가 개발자가 어려운 속성 지향 프로그래밍 으로 인해 놓쳤을 수 있는 AEG를 통해 악용 가능한 체인을 찾도록 도울 수 있음을 보여줍니다.

7.4 하이퍼파라미터

우리는 FUGIO의 두 매개변수인 퍼징 타임아웃과 최대 체인 길이의 효율성을 평가합니다.
퍼지 시간이 초과되었습니다. fuzzer 는 익스플로잇 개체를 보고할 때까지 생성된 입력으로 PUT을 실행합니다. fuzzer가 오래 실행될수록 더 많은 익스플로잇이 생성될 수 있습니다. 그러나 주어진 시간 예산 내에서 다른 POP 체인을 테스트하기 위해 퍼징 시간 제한을 설정했습니다. 이 섹션에서 우리는 실험을 수행했습니다.

fuzzer가 각 POP 체인에 대해 실행되어야 하는 시간을 결정합니다.

평가 데이터 세트를 구축하기 위해 퍼지할 POP 체인을 샘플링했습니다. 30개 애플리케이션의 각 민감한 싱크에 대해 최대 10개 체인을 샘플링했으며 그 중 마지막 가제트 에 이 싱크가 있습니다. 샘플링된 POP 체인의 총합은 13,582입니다.

샘플링된 각 체인에 대해 300초의 시간 제한으로 fuzzer를 5번 실행했습니다 . fuzzer는 총 408개의 E 체인과 662 개의 (PE+E) 체인을 생성했습니다. 각 PE 또는 E 체인에 대해 fuzzer가 익스플로잇 개체를 생성하는 데 소비한 시간을 측정했습니다 . 그런 다음 10초 단위로 0초에서 300초까지 지정된 특정 시간 내에 익스플로잇 개체가 생성된 체인 수를 계산했습니다 . 그림 5a 에 결과를 표시했습니다 . 그런 다음 300초 이내에 익스플로잇 개체가 생성된 전체 체인 중 fuzzer 가 지정된 시간 내에 익스플로잇 개체를 생성한 체인의 백분율을 계산했습니다. 그런 다음 퍼징 타임아웃을 백분율이 70%보다 커지는 시간으로 설정합니다. 악용 가능한 체인의 경우 백분율 은 100초에서 70.1%이고, 악용 가능하고 악용 가능한 체인의 경우 백분율은 50초에서 70.2%입니다.

따라서 모든 실험 (§ 7)에서 퍼징 제한 시간을 100초로 설정했습니다.

체인 길이. POP 체인을 식별할 때 체인 식별자는 호출 트리의 높이를 매개변수로 사용합니다 (§ 5.4). 높이는 FU GIO가 계산 하는 POP 체인의 최대 길이를 의미 합니다 . 익스플로잇 개체 를 보고할 때 최대 체인 길이의 효율성 을 측정하기 위한 실험을 수행했습니다 .

이 평가를 위한 POP 체인을 준비하는 방법은 앞서 언급한 퍼징 타임아웃 실험과 유사합니다. 최대 체인 길이를 1에서 10 사이의 값으로 설정하고 각 조건에 대해 POP 체인을 샘플링했습니다. 예를 들어, 최대 체인 길이를 3으로 설정했다면 길이가 1, 2, 3인 체인 사이에서 샘플링이 수행됩니다.

FUGIO 는 100초의 제한 시간을 사용하여 샘플링된 체인에 대해 5번의 퍼징 캠페인을 수행했습니다. 각 최대 체인 길이에 대해 생성된 E 체인과 PE 및 E 체인의 총계 를 계산했습니다. 그림 5b 는 실험 결과를 보여줍니다. 최대 체인 길이를 7에서 8로 변경 시 체인 수가 감소합니다. 퍼징 타임아웃 예산이 모든 유망한 체인의 퍼징을 줄여 익스플로잇 개체를 생성하기 때문입니다. 따라서 § 7.2 에서 최대 체인 길이를 7로 설정했습니다.

7.5 필드 테스트

우리는 최신 버전의 두 PHP 애플리케이션(WooCommerce가 포함된 WordPress 5.4.2 및 Concrete5 8.5.4)에서 FUGIO의 효율성을 평가했습니다. WooCommerce는 5백만 건 이상의 활성 설치가 있는 가장 인기 있는 WordPress 플러그인 중 하나입니다 [3].

WordPress에서 FUGIO는 7개의 민감한 싱크 에 대해 39개의 POP 체인을 식별하고 익스플로잇 개체가 있는 익스플로잇 가능한 체인 1개를 보고했습니다. 체인은 WC_Log_Handler_File: :__destruct 로 시작 하고 두 명의 사용자 가 있는 call_user_func 를 트리거합니다 .

```
1 <?php
2 클래스 WC_Log_Handler_File 확장 WC_Log_Handler {
#         function __destruct () { // 첫 번째 가젯
4             foreach ($this -> $handle 로 처리 ) {
5                 if ( is_resource ( $handle ) ) {
6                     fclose ( $핸들 ) ;
7 } } } }
8 클래스 Requests_Utility_FilteredIterator 확장
                배열 반복자 {
9 public function current () { // 두 번째 가젯
11              $value = 부모 :: 현재 () ;
12              $value = call_user_func ($this -> 콜백                    ,
13                                                  $값 ) ; // 싱크대
14              반환 $ 값 ;
15 } }
16
17 // 주입 객체 생성
18 $obj = 새로운 WC_Log_Handler_File ;
19 $obj - > setProp ('핸들 ',
20                                 새로운 Requests_Utility_FilteredIterator (
21                                     [" 인수 ) ) ;     = >     ARGUMENT_OF_CALLBACK "]
22
23 $obj - > getProp ('핸들 ')
24              -> setProp ('콜백 ', "                    CALLBACK_TO_BE_CALLED ") ;
25
26 // POI 취약성 트리거
27 $data = 직렬화 해제 ( $input ) ;
```

목록 3: WordPress의 악용 가능한 POP 체인

제어 가능한 인수. 따라서 공격자는 원격으로
그녀가 선택한 인수로 기존 함수를 호출합니다.
예를 들어, 공격자는 위조된 인수로 시스템 을 호출 하여 쉘 명령 주입 공격을 수행합니다.

Concrete5에서 FUGIO는 다음과 같은 4개의 악용 가능한 체인을 발견했습니다.
201개의 민감한 싱크에 대해 5,016개 체인 중 개체를 악용합니다.
3개의 사슬과 그들의 악용 가능한 개체는 공격자가
임의의 파일과 그와 함께 남은 하나의 체인을 삭제하려면
익스플로잇 개체를 통해 공격자는 사용자 정의된 모든
그녀의 인수 선택으로 기능합니다.

폭로. 우리는 HackerOne [21]에 익스플로잇 개체가 있는 취약점을 보고했습니다. WordPress 팀은 다음과 같이 알렸습니다.
취약점은 2018년 9월에 보고되었지만
공개되지 않은 버그로 남아 있습니다. Concrete5 팀 패치
보고된 취약점 및 할당된 CVE-2021-40102.

7.6 사례 연구

FU GIO가 WooCommerce를 사용하여 WordPress 5.4.2에서
보고한 두 개의 E 체인과 해당 익스플로잇 개체를 소개합니다.
최신 버전의 Concrete5.

WooCommerce와 WordPress. 목록 3 은 POP를 보여줍니다.
7.5절 에서 WordPress 취약점을 탐지 할 때 FUGIO가 생성한 체인 및 해당 익스플로잇 개체 . 악용 가능
POP 체인은 WC_Log_Handler_File: 의 두 가지 가젯으로 구성됩니다.
__destruct 및 Requests_Utility_FilteredIterator:
:call_user_func 싱크 함수를 호출하는 현재.

ArrayItera tor 를 상속하는 클래스 의 현재 메서드는 해당 클래스의 객체에 대한 foreach 호출에 의해 호출됩니다 .
4 행 의 핸들 속성 이 Requests_- 로 설정된 경우
Utility_FilteredIterator 인스턴스, 현재 메서드
호출됩니다.

fuzzer는 이에 대한 익스플로잇 객체를 생성하는 것으로 시작합니다.

```
1 <?php
2 네임스페이스 simplehtmldom_1_5 {
3 클래스 simple_html_dom {
4         function __destruct () { // 첫 번째 가젯
5             $this -> 지우기 () ;
6         }
7         function clear () { // 두 번째 가젯
8             if ( isset ($this - > parent ) ) {
9                 $this - > 부모 - > 지우기 () ;
10                설정되지 않음 ($this -> parent ) ;
11 } } } }
12 네임스페이스 스태시 \ 드라이버 {
13        클래스 FileSystem 은 AbstractDriver { 를 확장합니다.
14            public function clear ( $key = null ) { // 세 번째 가젯
15                $path = $this -> makePath ( $key ) ;
16                if ( is_file ( $경로 ) ) {
17                    $반환 = 참 ;
18                    링크 해제 ( $path ) ; // 싱크대
19            } }
20            보호된 함수 makePath ( $key = null ) {
21                if (! isset ($this - > cachePath ) ) {
22                    새로운 LogicException 발생 ('오류 ') ;
23                }
24                $basePath = $this -> cachePath ;
25                $경로 = $베이스 경로 ;
26                반환 $경로 ;
27 } } }
28
29 // 주입 객체 생성
30 $obj = 새로운 simplehtmldom_1_5 \ simple_html_dom ;
31 $obj -> setProp ('부모 ', 새로운 Stash \ Driver \ FileSystem ) ;
32 $obj - > getProp ('부모 ')
33              -> setProp ('cachePath ', "                    FILE_TO_DELETE ");
34
35 // POI 취약성 트리거
36 $ 데이터 = is_dir ( $ 입력 ) ;
```

목록 4: Concrete5의 악용 가능한 POP 체인

첫 번째 가제트 클래스인 WC_Log_Handler_- 를 선택하여 체인
파일 (18행). 다음 가젯을 연결하기 위해 fuzzer가 변경됩니다.
WC_Log_Handler_ 에 선언된 각 속성의 값
파일 클래스. 4 행 의 핸들 속성 이 변경된 경우
객체 유형 으로 해당 값이 Requests_Utility_- 로 설정됩니다.
FilteredIterator (19-22행), 현재 방법 은
호출되고 fuzzer는 싱크 함수 call_user_- 에 도달합니다.
func in Line 13. 그런 다음 fuzzer는 mutated 속성이 이 민감한 싱크의 인수를 제어할 수 있는지 여부를 확인합니다.
call_user_func 의 첫 번째 인수는 Requests_Utility_FilteredIterator 클래스 의 콜백 속성입니다 .
공격자가 선택한 콜백으로 설정해야 합니다(23-24행).
이 콜백의 인수는 $value 를 통해 전달할 수 있습니다 .
call_user_func 의 두 번째 인수는 다음을 나타냅니다.
현재 반복되는 항목의 값(21행). 이후
공격자는 이 호출의 첫 번째 인수를 제어할 수 있습니다 _-
user_func 싱크, FUGIO는 보고하여 퍼즈 테스트를 종료합니다.
이 익스플로잇 개체.

콘크리트5. 목록 4 는 POP 체인과 해당 익스플로잇 개체를 보여줍니다.
FUGIO는 Concrete5 (§ 7.5)에서 보고했습니다. 체인 길이
세입니다. 이 체인은 simple_hmtl_dom::__de struct,
simple_hmtl_dom::clear 및 Filesystem::clear 로 구성됩니다.
연결 해제 기능을 호출합니다.

이 체인에 대한 익스플로잇 개체를 생성하기 위해 fuzzer는
첫 번째 가제트 클래스인 simple_hmtl_dom 을 선택합니다 (30행).
첫 번째 가젯인 simple_hmtl_dom::__destruct는 무조건적으로 두 번째 가제트인 simple_hmtl_dom::clear를 호출합니다.

세 번째 가젯인 Filesystem::clear 를 연결하려면 8 행 의 조건 이 전달되
어야 합니다. 따라서 fuzzer는 다음을 할당합니다.
상위 속성 에 임의의 값 . 이 조건을 통과한 후 fuzzer는 9 행에
서 참조 오류가 발생합니다.
부모 속성 으로 clear 를 호출하려고 할 때 .
이제 fuzzer는 명확한 클래스 객체를 할당합니다.
방법. fuzzer 가 이 클래스에 대해 FileSystem 을 선택할 때
개체가 있으면 마지막 가젯이 실행됩니다(31행). 닿다
민감한 싱크, $path 변수는 파일 경로여야 합니다.
(16행). fuzzer는 설정 방법을 모르지만
$path, 속성을 변경하는 동안 올바르게 설정할 수 있습니다.
파일 시스템. fuzzer는 임의의 값을 할당하려고 시도합니다.
cachePath 속성 에서 . cachePath 를 파일로 설정 하여
delete(줄 32–33), fuzzer는 생성에 성공했습니다.
식별된 POP 체인에 대한 익스플로잇 개체.

8 논의 및 제한 사항

적대적 개체를 역직렬화하는 보안 위협은
PHP뿐만 아니라 Python [31, 41], Java [25, 68], Ruby [33],

Android [48] 및 .NET [18, 43, 55]. 사용 가능 여부에 따라
가젯트를 통해 공격자는 다양한 악의적인 행동을 합니다.
각 언어에는 완화를 위한 자체 권장 사항이 있습니다.
이 위협. 기본 테이크아웃은 데이터를 직접 역직렬화하지 않는 것입니다.
신뢰할 수 없는 출처 [46]. 이 권장 사항 을 따르는 일반적인 관행은 사용자 입력을
삭제하는 것입니다 [17, 46].
불행히도 살균 로직은 대상에 따라 달라야 합니다.
종종 구현으로 이어지는 역직렬화 방법
잘못된 위생 검사 [20, 54].
또 다른 권장 사항은 개발자가 JSON(예: json_-

PHP에서 디코딩 하고 Python에서 json.loads), YAML(예:
.NET의 SnakeYAML , Python의 PyYAML) 또는 XML(예:
직렬화 해제를 호출하지 않는 JAVA의 XMLDecoder)
콜백 [46, 53]. 불행히도, Muñoz와 Mirosh [43]
.NET 및 Java의 많은 JSON 라이브러리 가 개체 필드를 채우기 위
해 setter를 호출했기 때문에 악용될 수 있음을 발견했습니다.
다른 연구 [16, 55] 는 유사한 취약점을 보고했습니다.
XML 및 YAML 형식.

역직렬화할 클래스를 화이트리스트 또는 블랙리스트에 추가하는 것은
수동적 완화 방법 입니다[39]. 이 접근 방식은 다음을 제한합니다.
클래스는 각 언어에서 지원하는 기능을 에이징하거나

블랙리스트에 있는 클래스를 역직렬화할 때 오류가 발생합니다. 그러나 이
접근 방식을 지정하려면 상당한 엔지니어링 비용이 필요합니다.
(역)직렬화를 위한 허용된 클래스 [20].
기존 도구는 안전하지 않은 역직렬화를 감지하는 데 중점을 두었습니다
[2, 8, 19, 42]. Burp Suite는 Java의 경우 [19] , PHP 의 경우 [2] 를 사
용하여 사전 정의된 페이로드를 전송하여 취약점을 감지합니다.
애플리케이션 [15, 51]. SerialDetector 는 오염 데이터 흐름 분석을 사용
하여 안전하지 않은 역직렬화를 식별하고 식별된 유효성을 검사합니다.
알려진 가제트에 대해서만 페이로드를 생성하여 취약점
체인 [55]. 그들은 식별하기 위해 사용 가능한 가제트를 조립하지 않습니다.
유망한 체인을 묶습니다. 몇 가지 정적 접근 방식이 집중되었습니다.
악용 가능한 가제트 체인 식별에 대해 [13, 22]. 하지만,
이 연구는 거짓을 제거하기 위해 수동 검사가 필요합니다
긍정적인. 대조적으로, FUGIO는 일반적인 AEG 도구입니다.
유망한 POP 체인을 식별하고 익스플로잇을 생성합니다.
ESCAPE에는 한계가 있습니다. ESCAPE는 가제트만 조립합니다.
PHP의 가젯이 아닌 대상 PHP 애플리케이션에서 추출
내부 수업. 따라서 다음을 사용하여 익스플로잇을 생성할 수 없습니다.
PHP 내부 가제트. 이러한 익스플로잇을 생성하려면 다음이 필요합니다.
PHP에서 내부 가젯을 명시적으로 제공하기 위한 수동 노력
후지오에게. 같은 이유로 FUGIO는 활용할 수 없습니다.
PHP 소스가 있는 PHP 바이너리 모듈 [49] 의 가제트
코드가 변환됩니다
FUGIO는 또한 반사 호출을 커버할 수 없습니다.
대상 호출 수신자는 POP를 열거할 때 정적으로 결정할 수 없습니다.
쇠사슬. 이 대상 호출 수신자에 대한 모든 기존 가젯 고려
퍼징을 수행하기 위해 금지된 수의 체인이 발생합니다.
테스트. 이 대상 호출 수신자에 대해 가능한 값을 계산하는 정교한 정적 분석이 한
가지 대안적 접근 방식입니다.
거짓 음성을 줄이기 위해. 퍼즈 테스트 [38]의 특성으로 인해 익스플로잇
을 찾는 데는 여러 캠페인이 필요할 수 있습니다 .
대상 체인에 많은 수의 시간 초과가 있을 때
정황.

9 관련 작업

웹 애플리케이션의 취약점을 찾습니다. 광대하다
PHP 응용 프로그램의 취약점을 찾는 연구의 양 . Huang et al. 안전하지 않은 것
을 감지하는 WebSSARI 도입
유형 상태 기반 정적 분석 알고리즘을 사용한 정보 흐름 [29, 30]. Xieet al. 3단계
분석을 제시했다.
블록 내, 절차 내,
및 절차 간 수준 [66]. 픽시 추가 수행
별칭 및 리터럴 분석을 통해 보다 포괄적이고
정확한 결과 [35, 36]. Sonet al. 제시된 정적 분석
의미 버그를 식별 하고 수정 하는 기술 [57]
액세스 제어 버그 [56]. Backes et al. 코드 속성 그래프 기반 절차 간 분석 기법 제
안
단일 그래프 구조에서 프로그램의 구문, 제어 흐름 및 데이터 종속성을 나타내는
[6].
AEG. AEG는 보고된 버그가 보안에 중요한지 여부 를 자동으로 확인하는 검증 프
로세스로 사용되었습니다 .
이는 오탐을 제거하는 데 기여하고 개발자 가 패치할 버그의 우선 순위를
지정하는 데 도움이 됩니다 [5]. 바이너리 애플리케이션에서 AEG
접근 방식은 주로 제약 조건을 해결하여 익스플로잇을 생성합니다.
1) 사용자 입력이 유발하는 경로 제약 조건
충돌하는 주어진 프로그램 및 2) 실행에 대한 제약
쉘코드 [4, 11, 23, 26, 27, 47, 65, 67].
AEG 기술은 웹 분석에도 적용됩니다.
응용 프로그램. Balzarotti et al. Static을 사용하여 입력 소스에서 민감한 싱크
로 의심스러운 프로그램 경로를 식별하여 살균 프로세스를 검증하기 위해 Saner를
도입했습니다.
동적 분석을 사용하여 공격 문자열을 포함하는 입력으로 프로그램을 분석하
고 시뮬레이션합니다 [7]. Kieyzun et al.

및 Huang et al. con colic 실행 에 기반한 제안된 AEG 접근 방식 [28, 37]. Alhuzali et al. 취약한 싱크로 프로그램 실행을 지시 하는 일련의 악성 HTTP 요청을 구성하기 위해 추가 정적 분석을 수행했습니다 [1].

많은 기호 실행 연구 에서 교차 사이트 스크립팅 , SQL 삽입 또는 파일 포함 과 같은 다양한 유형의 취약점 을 검증하려고 시도했지만 이전 연구에서는 POI 취약점에 대한 익스플로잇 생성을 다루지 않았습니다. 게다가 이러한 제약 조건 해결 접근 방식은 수천 개의 내장 기능 을 모델링해야 하기 때문에 상당한 엔지니어링 노력이 필요 합니다. 우리는 기호 실행 대신 퍼징을 통해 이 문제를 해결하기로 결정했습니다.

10 결론

POI 취약점에 대한 최초의 AEG 도구인 FUGIO를 제안합니다. POP 체인을 계산하고 익스플로잇을 생성 하기 위한 일련의 정적 분석, 동적 분석 및 퍼징 기술을 제시 합니다. FUGIO는 알려진 POI 취약점이 있는 30개의 실제 PHP 애플리케이션에서 68개의 익스플로잇 개체를 보고했습니다.

또한 FUGIO는 기능적 익스플로잇 개체와 함께 이전에 알려지지 않은 두 가지 POI 취약성을 보고했으며, 이는 힘든 속성 지향 프로그래밍 부담을 크게 완화하는 FUGIO의 효능을 보여줍니다.

승인

논문 개선을 위해 유익한 논평과 제안을 해주신 익명의 심사위원과 Xinyu Xing 목자께 감사드립니다 . 이 작업은 한국연구 재단(NRF) 보조금 번호: 2020R1C1C1009031 의 지원을 받았으며 부분적으로 Google Faculty Fellowship의 지원을 받았습니다.

참고문헌

[1] Abeer Alhuzali, Rigel Gjomemo, Birhanu Eshete 및 VN Venkatakrishnan. NAVEX: 동적 웹 애플리케이션을 위한 정확하고 확장 가능한 익스플로잇 생성. USENIX 보안 심포지엄 회보, 377–392 페이지 , 2018.

[2] 앰비오닉스 보안. PHPGGC: PHP 일반 가제트 체인. https://github.com/ambionics/phpggc.

[3] 자동. WooCommerce WordPress 플러그인. https: //wordpress.org/plugins/woocommerce/.

[4] Thanassis Avgerinos, Sang Kil Cha, Brent Lim Tze Hao, David Brumley. AEG: 자동 익스플로잇 생성. 2011 년 네트워크 및 분산 시스템 보안 심포지엄 회보에서.

[5] Thanassis Avgerinos, 차상길, Alexandre Rebert, Edward J Schwartz, Maverick Woo, David Brumley. 자동 익스플로잇 생성. ACM 통신 , 57(2):74–84, 2014.

[6] Michael Backes, Konrad Rieck, Malte Skoruppa, Ben Stock, Fabian Yamaguchi. PHP 애플리케이션 취약점의 효율적이고 유연한 발견. 보안 및 개인 정보 보호에 관한 IEEE 심포지엄 회보, 2017 년 334~349페이지 .

[7] Davide Balzarotti, Marco Cova, Vika Felmetsger, Ne nad Jovanovic, Engin Kirda, Christopher Kruegel 및 Giovanni Vigna. Saner: 정적 및 동적 분석을 구성하여 웹 애플리케이션에서 완전 삭제를 검증합니다. 보안 및 개인 정보 보호에 관한 IEEE 심포지엄 회보 , 2008년 387-401페이지 .

[8] 모리츠 베츨러. 자바 언마샬러 보안. https: //github.com/mbechler/marshalsec.

[9] Tyler Bletsch, Xuxian Jiang, Vince W Freeh, Zhenkai Liang. 점프 지향 프로그래밍: 새로운 종류의 코드 재사용 공격. 컴퓨터 및 통신 보안에 관한 ACM 아시아 회의 회보 , 2011년 30~40페이지.

[10] Marcel Böhme, Van-Thuan Pham, Manh-Dung Nguyen, Abhik Roychoudhury. 그레이박스 퍼징을 지시했습니다. 컴퓨터 및 통신 보안에 관한 ACM 회의 회보 , 2329–2344페이지, 2017.

[11] 차상길, Thanassis Avgerinos, Alexandre Rebert, David Brumley. 바이너리 코드에 혼란을 야기합니다. 보안 및 개인 정보 보호 에 관한 IEEE 심포지엄 회보 , 380–394페이지, 2012.

[12] 요하네스 다세. 줌라! 3.0.2 POI(CVE-2013-1453) – 가제트 체인. https://websec.wordpress.com/ 2014/10/03/joomla-3-0-2-poi-cve-2013-1453- 가젯-체인/, 2014.

[13] Johannes Dahse, Nikola Kreini 및 Thorsten Holz. PHP의 코드 재사용 공격: 자동화된 POP 체인 생성. 컴퓨터 및 통신 보안 에 관한 ACM 회의 회보 , 2014년 42~53페이지.

[14] 드미트리 제노비치. Runkit(공식 PECL PHP Runkit 확장). https://github.com/zenovich/runkit.

[15] 페데리코 도타. 자바 역직렬화 스캐너(Burp Suite 플러그인). https://github.com/federicodotta/ Java-Deserialization-Scanner.

[16] OWASP Stammtisch 드레스덴. JSON 역직렬화 악용. https://owasp.org/www-pdf-archive/Marshaller_Deserialization_Attacks.pdf.pdf .

[17] Sondre Forland Fingan. 자바 역직렬화 취약점
담력. 2020년 석사학위논문.

[18] 제임스 포쇼. 당신은 내 유형입니까? 직렬화를 통해 .NET을 깨기 . 2012년
Black Hat USA 의 회보에서 .

[19] 크리스토퍼 프로호프. 이세리얼. https://github.com/frohoff/ysoserial .

[20] 아포스톨로스 지아나키디스. 역직렬화 문제: 역직렬화 취약점이란 무엇이며
솔루션을 제공할 때의 과제는 무엇입니까? https://www.waratek.com/
wp-content/uploads/2019/06/WP-Deserialization-20190610.pdf .

[21] 해커원. 해커 기반 보안 테스트 및 버그 현상금. https://
www.hackerone.com/.

[22] 이안 하켄. deserialization gad get chains의 자동 검색. 2018년
Black Hat USA의 회보에서.

[23] 숀 힐런. 소프트웨어 취약점에 대한 악용을 하이재킹 하는 제어 흐름 자동 생
성 . 2009년 옥스퍼드 대학교 박사 학위 논문 .

[24] 마크 힐즈, 폴 클린트, 위르겐 빈주. PHP 기능 사용에 대한 실증적
연구: 정적 분석 관점.
소프트웨어 테스트 및 분석 에 관한 ACM 국제 심포지엄 회보,
2013년 325~335페이지.

[25] 필립 홀징거, 스테판 트릴러, 알렉상드르 바르텔, 에릭 보든. 10년 이상의 자
바 익스플로잇에 대한 심층 연구. 컴퓨터 및 통신 보안에 관한 ACM 회의
회보, 779–790페이지, 2016.

[26] Hong Hu, Zheng Leong Chua, Sendroiu Adrian, Prateek Saxena
및 Zhenkai Liang. 데이터 지향 익스플로잇의 자동 생성. USENIX 보안
심포지엄 회보, 2015년 177–192페이지.

[27] Shih-Kun Huang, Min-Hsiang Huang, Po-Yen Huang, Chung-Wei
Lai, Han-Lin Lu, Wai-Meng Leong.
CRAX: 공격을 상징적 연속으로 모델링하여 자동 익스플로
잇 생성을 위한 소프트웨어 충돌 분석. 2012년 소프트웨
어 보안 및 안정성 에 관한 IEEE 국제 회의에서 .

[28] Shih-Kun Huang, Han-Lin Lu, Wai-Meng Leong, Huan Liu.
CRAXweb: 자동 웹 애플리케이션 테스트 및 공격 생성. 2013년 소프트웨
어 보안 및 안정성에 관한 IEEE 국제 회의에서.

[29] 황야오웬, 위팡유, 크리스티앙 항, 차이충훙, 이더차이, 궈사이엔. 정적 분석
및 런타임 보호를 통해 웹 애플리케이션 코드를 보호 합니다. World Wide
Web 에 관한 국제 회의 회보 , 2004년 40~52페이지.

[30] 황야오웬, 위팡유, 크리스티안 항, 차이충훙, 이더차이, 궈사이엔. 제한된 모
델 검사를 사용하여 웹 응용 프로그램을 확인합니다. 신뢰할 수 있는 시스
템 및 네트워크에 관한 국제 회의 , 199–208페이지, 2004.

[31] 불면증 보안. 역직렬화, 무엇이 잘못될 수 있습니까? https://
insomniasec.com/cdn-assets/ Deserialization_-
__What_Could_Go_Wrong.pdf.

32 인빅투스 아큐네틱스. https://www.acunetix.com/.

[33] 루크 얀케. Ruby 2.X 범용 RCE 역직렬화 가제트 체인. https://
www.elttam.com/blog/ruby deserialization/#content.

[34] 조 왓킨스. uopz: zend에 대한 사용자 작업입니다. https:
//github.com/krakjoe/uopz.

[35] 네나드 요바노비치, 크리스토퍼 크루겔, 엔긴 키르다.
Pixy: 웹 애플리케이션 취약점 을 탐지하기 위한 정적 분석 도구입니
다 . 보안 및 개인 정보 보호에 관한 IEEE 심포지엄 회보 , 258–263
페이지, 2006.

[36] 네나드 요바노비치, 크리스토퍼 크루겔, 엔긴 키르다.
웹 애플리케이션 취약점의 정적 탐지를 위한 정확한 별칭 분석. 프
로그래밍 언어 및 보안 분석에 대한 ACM SIGSAC 워크숍 회보 ,
27–36페이지, 2006년.

[37] Adam Kieyzun, Philip J Guo, Karthick Jayaraman 및 Michael D
Ernst. SQL 주입 및 교차 사이트 스크립팅 공격의 자동 생성. 소프트웨어
엔지니어링에 관한 국제 회의 회보 , 2009년 199-209페이지.

[38] George Klees, Andrew Ruef, Benji Cooper, Shiyi Wei, Michael
Hicks. 퍼지 테스트 평가. 컴퓨터 및 통신 보안에 관한 ACM 회의 진행 ,
2123–2138페이지, 2018.

[39] 윌 클리버. 신뢰할 수 없는 데이터의 역직렬화를 방지합니다. https://
wiki.sei.cmu.edu/confluence/display/java/SER12-J.
+Prevent+deserialization+of+ untrusted +data.

[40] 이택진, 위성일, 이수영, 손수엘.
FUSE: 침투 테스트를 통해 파일 업로드 버그를 찾습니다.
2020년 네트워크 및 분산 시스템 보안 심포지엄 회보에서.

[41] 댄 루스키. 파이썬으로 역직렬화 취약점을 설명하고 악용합니다. https://
dan.lousqui.fr/ explain-and-exploiting-deserialization 취약점-
with-python-en.html.

[42] 알바로 무노즈. ysoserial.net. https://github.com/
pwntester/ysoserial.net.

[43] 알바로 무뇨스와 올렉산드르 미로시. 13 일의 금요일
json 공격. Black Hat USA의 회보에서,
2017.

[44] 네르갈. 고급 return-into-lib(c) 익스플로잇: PaX
사례 연구. http://phrack.org/issues/58/4.html.

[45] 니키. PHP 파서. https://github.com/nikic/PHP
파서.

[46] OWASP. OWASP Top Ten 2017 A8: 불안정한 Dese rialization. https://
owasp.org/www-project-top ten / 2017 / A8_2017-
Insecure_Deserialization.

[47] Vartan A Padaryan, VV Kaushan 및 AN Fedotov. 스택 버퍼
오버플로에 대한 자동 익스플로잇 생성
취약점. 프로그래밍 및 컴퓨터 소프트웨어,
41(6):373–380, 2015.

[48] 또는 Peles와 Roee Hay. 그들을 모두 지배하는 하나의 클래스: 0-day
안드로이드의 역직렬화 취약점. 2015년 공격 기술에 관한 USENIX
워크숍 진행 과정에서 .

[49] 팔콘. 제피르. https://github.com/zephir-lang/
제퍼.

[50] 포트스위거. 버프 스위트. https://portswigger.net/
트림

[51] 포트스위거. PHP 객체 주입 슬링어. https://
github.com/portswigger/poi-slinger.

Ryan Roemer, Erik Buchanan, Hovav Shacham, Ste fan Savage. 반환 지향 프
로그래밍: 시스템,
언어 및 응용 프로그램. 정보 및 시스템 보안 에 관한 ACM 거래 ,
15(1):1–34, 2012.

[53] 루비-닥. 육군 원수. https://ruby-doc.org/core
2.6.3/Marshal.html.

[54] 샘 사눕. SuiteCRM: 코드 실행에 대한 PHAR 역직
렬화 취약성. https://
snyk.io/blog/suitecrm-phar-deserialization 취약점-코드 실행/.

[55] Mikhail Shcherbakov와 Musard Balliu. SerialDetector : 웹
에 대한 개체 주입 취약점에 대한 원칙적이고 실용적인 탐구 . 절차에
서
네트워크 및 분산 시스템 보안 심포지엄,
2021.

[56] 손수엘, 캐서린 S 맥킨리, 비탈리 슈마티코프.
Fix Me Up: 웹 애플리케이션의 액세스 제어 버그를 수정 합니다. 네
트워크 및 배포 절차에서
시스템 보안 심포지엄, 2013.

[57] 손수엘과 비탈리 슈마티코프. SAFERPHP: 찾기
PHP 애플리케이션의 의미론적 취약점. 프로그래밍에 대한 ACM SIGSAC 워
크숍 진행 중
보안을 위한 언어 및 분석, 2011.

[58] 소나 소스. PHP 코드 품질 및 코드 보안.
https://www.sonarsource.com/php/.

[59] PHP 그룹. PHP 5 변경 로그 버전 5.4.24.
https://www.php.net/ChangeLog-5.php#5.4.24.

[60] PHP 그룹. PHP 5 변경 로그 버전 5.5.8. https:
//www.php.net/ChangeLog-5.php#5.5.8.

[61] PHP 그룹. PHP 커밋: 옵션 추가
libxml 문서 로드 옵션을 전달하는 매개변수입니다.
https://github.com/php/php-src/commit/
cb72e23c147c5a93161c24428762b434dc58524d.

[62] PHP 그룹. 버그 62789: 자동 로더가 호출됨
유효하지 않은 클래스 이름으로. https://bugs.php.net/
bug.php?id=62789, 2012.

VM웨어. 래빗엠큐. https://www.rabbitmq.com/.

[64] WP스캔. WooCommerce 인증 Phar De 직렬화. https://wpscan.com/
vulnerability/
9567f575-529d-4d66-980c-73cba6726673.

[65] Wei Wu, Yueqi Chen, Jun Xu, Xinyu Xing, Xiaorui
Gong, Wei Zou FUZE: 착취를 촉진하는 방향으로
커널 use-after-free 취약점에 대한 생성. 에
USENIX 보안 심포지엄의 절차, 페이지
781–797, 2018.

[66] Yichen Xie 및 Alex Aiken. 보안의 정적 감지
스크립팅 언어의 취약점. 의 절차에서
USENIX 보안 심포지엄, 179–192페이지, 2006.

[67] Luhang Xu, Weixi Jia, Wei Dong 및 Yongjun Li. 버퍼 오버플로
취약점에 대한 자동 익스플로잇 생성 . 소프트웨어 품질, 신뢰성 및 보
안에 관한 IEEE 국제 회의의 절차에서

동반자, 2018년 463~468면.

[68] Yang Zhang, Yongtao Wang, Keyi Li, Kunzhe Chai.
Java 역직렬화 공격의 새로운 익스플로잇 기술. 에
Black Hat EU 절차, 2019.

11 부록

11.1 타겟에 민감한 싱크 기능

FUGIO가 각 웹에 대해 고려하는 싱크 기능을 나열합니다.
다음과 같이 공격합니다.

- 파일 삭제: 링크 해제, rmdir
- 파일 생성: fopen, fwrite, fputs, mkdir, copy, link,
  심볼릭 링크, file_put_contents
- 파일 수정: chmod, chown, chgrp, touch
- 쉘 명령 주입: popen, system, passthru,
  exec, proc_open, shell_exec, escapeshellcmd
- 원격 코드 실행: eval, mail, call_user_func,
  call_user_func_array, preg_replace