Web fuzzer

연락처

qjrm1430 : qjrm1430@naver.com

shsec : coolsunny1004@naver.com

sean(slay) : qtt1536@gmail.com

---

study

dirbuster

set pentest env

퍼징(Fuzzing)

소프트웨어 취약점을 찾을 때 사용하는 방법 중 하나로, 애플리케이션이 에러를 발생하게 만들기 위해 비정상적인 데이터를 만들어 전달하는 방법이며, 퍼징 테스트라고도 한다.

자동화 테스트로 기형/반기형적 데이터를 주입하여 SW 버그를 찾는 블랙박스 테스팅 기술이다.

크래시를 찾기 위해서 프로그램을 이용하여, 입력할 파일(또는 값)을 만들고(**생성), 프로그램 실행, 프로그램이 파일을 열고(입력), 크래시가 발생하는지 관찰 후 크래시 발생 여부에 따라 현재 사용한 파일과 로그를 저장하거나 폐기(분류) 절차를 자동화(또는 반자동화)한 테스트 방법으로 동적이다.**

퍼저와 스캐너

취약점 스캐너는 운영 중인 서비스의 약점을 찾아내는 과정이다. 목표로 하는 웹 서버의 IP, 열린 포트, 동작하는 서비스와 버전 정보를 알고 있다면, 이러한 서비스의 취약점을 확인하는 것이다.

퍼징을 "컴퓨터 프로그램의 입력에 유효하지 않거나 예상치 못한 또는 임의의 데이터를 제공하는 것"으로 정의합니다.

웹 애플리케이션 취약성 스캐너는 SQL 주입 또는 교차 사이트 스크립팅과 같은 취약성을 식별하기 위해 유효하지 않은 예기치 않은 데이터를 보냅니다.

퍼저는 일반적으로 비정상적인 데이터를 애플리케이션에
전달하여 에러를 유도한다.

공격대상 시스템의 보안 취약점을 탐지하기 위하여 목표 애플리케이션에 대한 다양한 입력값을 전송하는 결함 주입에 근거한다.

퍼징은 테스트 기법에 인데 취약점을 찾는것으로 의미가 변질되었다.

즉 스캐너 도구에 퍼징 기술, 모듈 혹은 플러그인을 이용한다면 퍼저라는 것이고 이를 사용하지 않는다면 단지 스캐너의 역할만 수행하는 것이다.

---

history

1988년 위스콘신 대학교 매디슨의 Barton Miller에 의해 생겼다.

(http://www.cs.wisc.edu/~bart/fuzz)

주로 커맨드라인 인터페이스 및 UI 퍼징을 지향하며 최신 운영체제에서도 단순한 퍼징으로 취약점을 찾아내었습니다.

---

fuzz의 탄생 유래

- Miller 교수가 직접 이메일 회신 해준 내용

폭풍우가 몰아치던 날, 모뎀을 통해 네트워크 작업을 하고 있었습니다. 비바람 때문인지 회선에서 불필요한 잡음이 섞여서 들려왔습니다. 그러던 중 제가 사용하던 프로그램에서 쓸데없는 문자열들이 의도치 않게 삽입되었고, 결국 오류가 발생하여 종료가 되는 현상을 겪게 되었습니다. 그때 들려오던 ‘파지지직’하던 잡음 소리에서 영감을 받아서, fuzz라고 이름을 붙였습니다.

크래시를 터트리는 이유

프로그램이 크래시가 터질 때는 내부에서 코드가 꼬여서 터지는 것인데, 꼬이는 코드를 잘 활용하면 공격자가 마음대로 프로그램을 동작시킬 수 있기 때문이다.

퍼저란 그리고 퍼저의 필수 기능

퍼징할 때 사용하는 툴을 퍼저(fuzzer)라고 한다.

fuzzer는 크게 3가지 기능이 필요하다.

1. 입력값을 생성하는 기능
2. 프로그램을 실행하는 기능
3. 결과를 분류하는 기능

- 2012 codeengn 발표 자료 Everyone has his or her own fuzzer

```
import os, sys, random

def go():
    return random.randrange(0, 0x100)

filesize = os.path.getsize("./sample.xxx")
fp = open("./sample.xxx", "rb++")
tmpoffset = random.randrange(0, filesize)
fp.seek(tmpoffset, 0)
fp.write("%c%c%c%c" % (go(), go(), go(), go()))  # 값 조작(입력값 생성)
fp.close()
os.system("target_binary sample.xxx")            # 실행
```

퍼징의 동작 방식

1. input 생성 → 2. 프로그램 실행 → 결과 관찰

- 어떻게 좋은 input을 생성할 수 있나
- 어떻게 빠르게 실행하나
- 어떻게 bug를 최대한 놓치지 않을까

퍼징의 대상

- 파일 포맷
- 네트워크 프로토콜
- 커맨드라인 파라미터
- 환경 변수
- 웹 어플리케이션

등

퍼징의 분류

퍼징을 위한 input data를 어떻게 조절하는지에 따라 두 가지로 나뉜다.

- Mutation(변이 기법, dumb 퍼징이라고도 함)
  - 기존의 데이터를 이용해 변형하여 대상에게 전달한다.
  - 주어진 입력 값이 무차별적으로 새롭게 바꿔가며 새로운 입력값을 만들어 내는 것.
  - 데이터의 형식이나 구조에 대한 명확한 이해가 어려울 때 사용하는 것이며, 노가다 같은 느낌이 강하다.
- Generation(Intelligent fuzzer라고도 함)
  - 새로운 데이터를 생성해 대상에게 전달한다.
  - 파일의 포맷(형식)이나 프로토콜(규약)을 이해하고 그것에 맞추어 적절한 input을 생성하는 방식.
  - 명세에 따라 맨 끝부터 철저히 구현해야 한다.

퍼징의 장점

퍼징은 테스팅을 무작위로 수행하는 기법으로, 정의된 것만 테스트 하는 접근 방식으로는 해결할 수 없는 지점을 공략하여 버그를 찾아내는 것인데, 쉽고 빠르게 수행하며 어떠한 작업을 여러 번 반복하는 것도 가능하다는 점과 자동화/반자동화이기 때문에 적은 시간으로 상당한 성과를 얻을 수 있다.

퍼징의 한계

블랙박스 테스팅을 할 때 유용하고, 퍼징을 위해 대상을 파악하는 것이 쉽지 않은 작업일 뿐더러 오탐과 트리거 할 수 없는 부분도 탐지하므로 분류가 힘들기도 하며 랜덤으로 작업하는 것 또한 경계값을 탐색하기가 어렵다.

대표적인 도구

1. Adress Sanitizer
   - 구글에서 개발하였고, 퍼징을 위한 표준 도구이다.

2. 웹 취약성 스캐너 역할
   - brup suite
   - wapiti

   - peach

3. 확장형 퍼징 프레임워크

4. 네트워크 레벨 프로토콜 퍼저
   - scapy

참고 URL

- 위키백과
- 퍼징 기법 https://cpuu.postype.com/post/589162
- 퍼저 정의. https://bloofer.net/95
- 퍼저와 스캐너 https://security.stackexchange.com/questions/124521/vulnerability-scanning-vs-fuzzing-a-web-application
- 퍼저 도구 https://blackarch.org/fuzzer.html

openvas, sulley

https://en.kali.tools/all/?category=fuzzer

퍼저 설계 론 - 어떤 절차 로직 인가. 한눈에 봐도 알수 있게 문서 정리, 오픈소스 분석 정리

shadow code, 다다음주까지 테스트 코드 작성

방법

제한적 방법론

1. 프로그램 실행 시 파라미터로 url 기입
2. 해당 url에 접속하여 beautifulsoup로 입력 관련 태그들 파싱
3. payload cheat sheet 파일에서 payload 하나씩 가져와 url+파싱된 태그의 이름 +payload 형태로 하여 request 모듈로 재전송
4. alert 객체를 얻어와서 조건문으로 여부를 확인한 뒤 alert 객체를 성공적으로 얻어왔으면 공격 포인트 발견, 얻어오지 못했으면 다시 되돌아가서 payload를 가져와 조합하여 재전송.
   이 작업을 반복

추구하는 방법론

1. 프로그램 실행 시 파라미터로 url 기입
2. 해당 url에 beautifulsoup로 파싱을 해오는데 서브 도메인으로 접속하는 부분이 있는 태그라면 urlparse를 이용하여 같은 도메인인지 확인한 뒤 접속을 하여 가능한 모든 입력 태그를 가져온다.
3. 가져온 입력 태그의 속성 중 이름 부분을 url+파싱된 태그의 이름+payload로 조합하여 공격 시도를 한다.
4. alert 객체를 얻어오는데 조건문으로 확인 여부를 검사하여 얻어올 수 있으면 공격 포인트 발견, 얻어오지 못하면 다시 되돌아가 코드 반복

pentestlab을 사용하여 유효성 검사

1. pentestlab으로 모의 해킹 환경을 open.
2. 위의 작업과 동일

---

python modules

- ptrace - 리눅스 기반 생성된 프로세스가 어떻게 움직이고 어떤식으로 데이터를 읽고 쓰는지 어떤 에러를 내는지 추적
- argparse
- requests - 딕셔너리 형태로 데이터 전송, 에러 정보 출력 안함, get post 지정
- urllib3 - 인코딩하여 바이너리 형태로 전송, 에러 정보 출력 함, get post 지정 안해도 됨 데이터를 보고 판단함
- beautifulsoup
- os
-