2018/12/17, 3-2 Capstone Project

최종 발표 슬라이드

- 보안 취약점 분석 및 보완 프로그램 개발 -

지도교수 : 이종혁

Captain : 201621571 손상진

Sailors : 201621110 권순홍

201621136 서민지

201621173 최서윤

QBQB

목차

1. 현재 진행 사항
   - RabbitMQ 동작 분석
   - 퍼징 관련 논문 분석
   - 퍼징 툴 사용
2. 앞으로의 진행 계획

QBQB

1. 현재 진행 사항

- RabbitMQ 동작 분석
  - RabbitMQ 설치
    - RabbitMQ 관리 플러그인 설치
  - RabbitMQ 실습
    - Hello world
    - Work Queue
    - Publish/Subscribe
    - Routing
    - Topic
    - RPC

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/documents_rabbitmq/1_RabbitMQ_install_guid_in_KR/KR.md

1. 현재 진행 사항

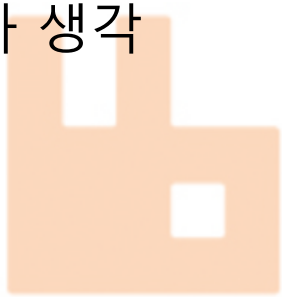
- 퍼징 관련 논문 분석
  - An Empirical Study of the Reliability of UNIX Utilities
    - 유닉스 유틸리티 프로그램의 신뢰성에 대한 연구
      1. 배경
      2. 도구
         - fuzz
         - ptyjig
      3. 도구를 통한 결과

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/study/1_An_Empirical_Study_of_the_Reliability_of_UNIX_Utilities/KR.md

1. 현재 진행 사항

- 퍼징 관련 논문 분석
  1. 배경
     - 비가 전화선에 영향을 미쳐, 터미널 라인에 잘못된 문자가 입력
     - 잘못된 문자가 입력되어 프로그램들이 중단되는 것을 발견
     - 해당 프로그램들은 유닉스 유틸리티
     - 안전하다고 생각하는 유닉스 유틸리티 같은 프로그램에도 심각한 버그가 있을 것이라 생각



1. 현재 진행 사항

- 퍼징 관련 논문 분석
  2. 도구

     유틸리티를 테스트 하기위해 프로그램 개발

     - fuzz
       - 목표 프로그램에 의해 소비될 랜덤 문자 스트림 생성
       - 여러가지 옵션 존재
     - ptyjig
       - 상호작용 유틸리티 프로그램을 테스트하기 위함
         - vi, nano와 같은 상호작용 유틸리티는 입력을 넣기가 어려움
     - 스크립트들
       - fuzz에서 출력된 임의의 문자열을 프로그램에 자동적으로 테스트 하기위한 스크립트

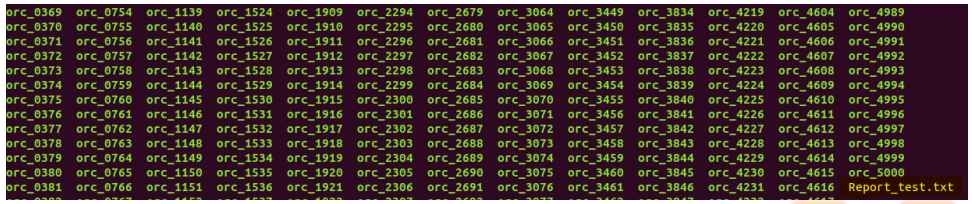
QBQB

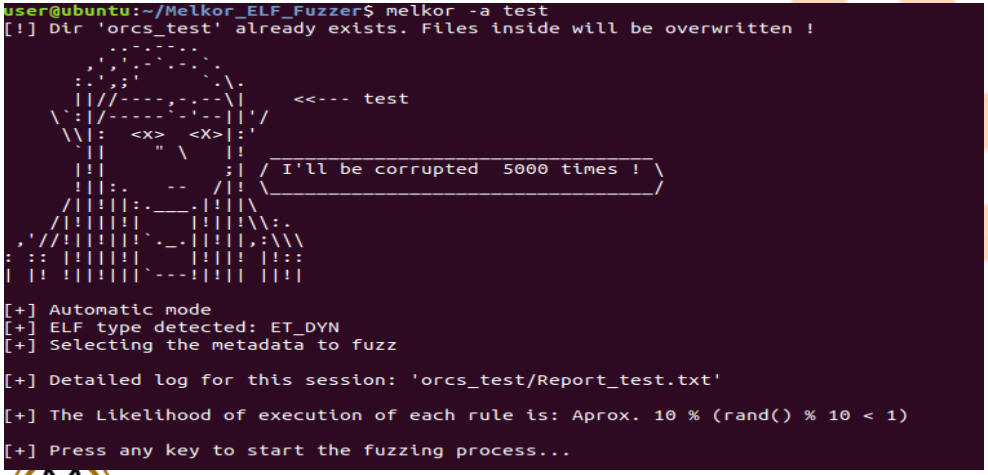
1. 현재 진행 사항

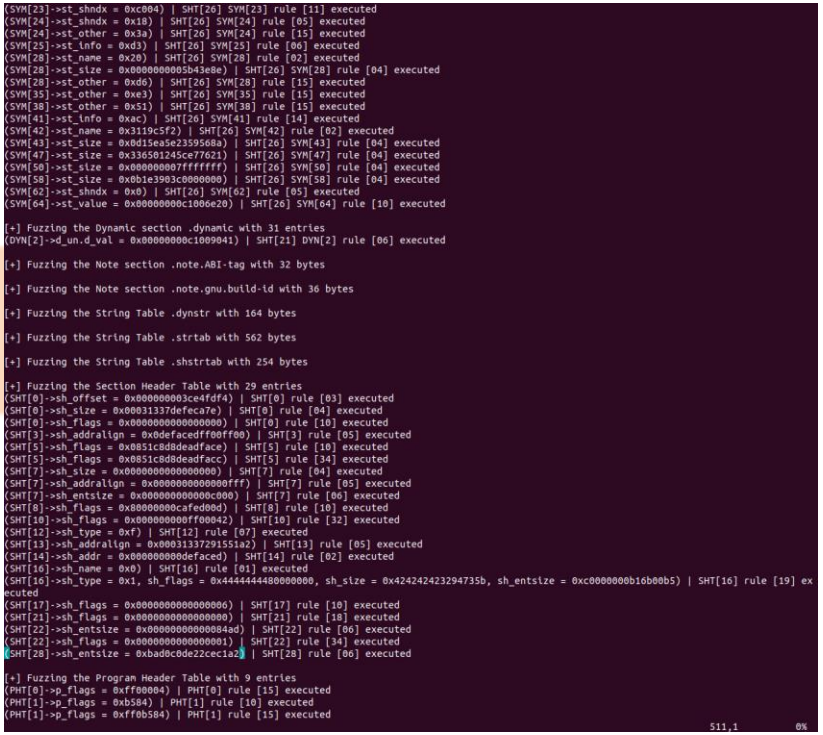
- 퍼징 관련 논문 분석
  3. 도구를 통한 결과
     - 3가지의 결론 도출
       - 충돌: 비정상적으로 종료된 프로그램이 core file 생성
       - 행: 프로그램이 무한 반복
       - 성공: 프로그램이 정상적으로 종료
     - 7가지 버전의 유닉스에서 88개의 유틸리티 프로그램 테스트
       - 안정적이다고 생각했던 프로그램에서 많은 문제 발견
         - 24~33%의 경우가 프로그램을 중단, 중지

1. 현재 진행 사항

- 퍼징 툴 사용
  - Melkor
    - Linux 시스템에서 ELF 파일에 대한 fuzzing을 위한 툴







QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/analysis_vuln/1_Analysis_fuzzing_tools/KR.md

1. 현재 진행 사항

- 퍼징 툴 사용
  - AFL(American Fuzzy Lop)
    - 해당 프로그램을 통해 타깃 프로그램에 컴파일을 통해 퍼징

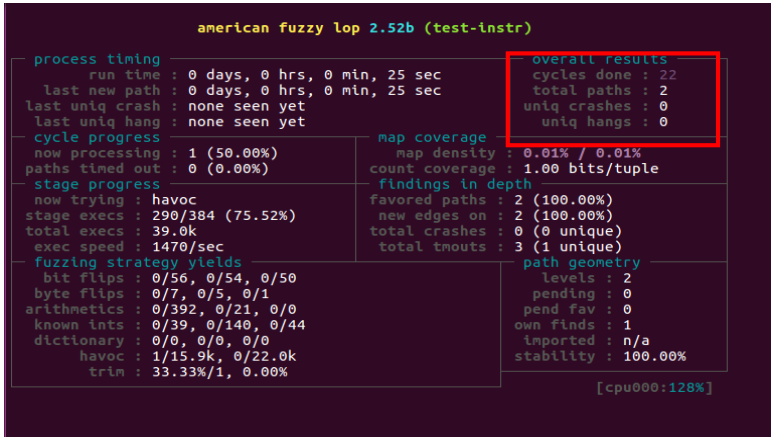
```
userm@userm-virtual-machine:~/afl-2.52b$ ./afl-gcc -o test-instr test-instr.c
afl-cc 2.52b by <lcamtuf@google.com>
afl-as 2.52b by <lcamtuf@google.com>
[+] Instrumented 6 locations (64-bit, non-hardened mode, ratio 100%).
```

```
userm@userm-virtual-machine:~/afl-2.52b$ ./afl-fuzz -i ./testcases/others/text/ -o ./afl-out/ ./test-instr
afl-fuzz 2.52b by <lcamtuf@google.com>
[+] You have 2 CPU cores and 3 runnable tasks (utilization: 150%).
[*] Checking CPU core loadout...
[+] Found a free CPU core, binding to #0.
[*] Checking core_pattern...
[*] Setting up output directories...
[+] Output directory exists but deemed OK to reuse.
[*] Deleting old session data...
[+] Output dir cleanup successful.
[*] Scanning './testcases/others/text/'...
[+] No auto-generated dictionary tokens to reuse.
[*] Creating hard links for all input files...
[*] Validating target binary...
[*] Attempting dry run with 'id:000000,orig:hello_world.txt'...
[*] Spinning up the fork server...
[+] All right - fork server is up.
    len = 6, map size = 4, exec speed = 2472 us
[+] All test cases processed.

[+] Here are some useful stats:

    Test case count : 1 favored, 0 variable, 1 total
       Bitmap range : 4 to 4 bits (average: 4.00 bits)
        Exec timing : 2472 to 2472 us (average: 2472 us)

[*] No -t option specified, so I'll use exec timeout of 20 ms.
[+] All set and ready to roll!
```



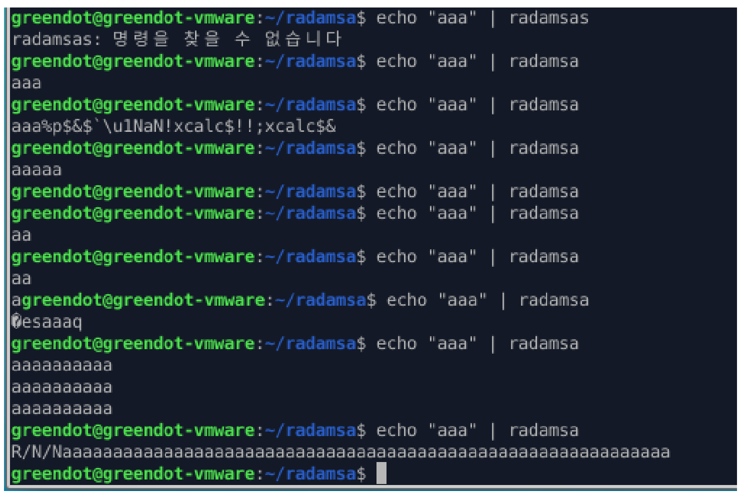

```
usern@usern-virtual-machine:~/afl-2.52b$ ls
Makefile            afl-as      afl-cmin    afl-gcc.c    afl-showmap   alloc-inl.h  docs          llvm_mode     types.h
QuickStartGuide.txt afl-as.c    afl-fuzz    afl-gotcpu   afl-showmap.c as           experimental  qemu_mode
README              afl-as.h    afl-fuzz.c  afl-gotcpu.c afl-tmin       config.h     hash.h        test-instr
afl-analyze         afl-clang   afl-g++     afl-out      afl-tmin.c    debug.h      libdislocator test-instr.c
afl-analyze.c       afl-clang++ afl-gcc     afl-plot     afl-whatsup   dictionaries libtokencap   testcases
usern@usern-virtual-machine:~/afl-2.52b$ cd afl-out
usern@usern-virtual-machine:~/afl-2.52b/afl-out$ ls
crashes  hangs  plot_data  queue
usern@usern-virtual-machine:~/afl-2.52b/afl-out$ cd crashes
usern@usern-virtual-machine:~/afl-2.52b/afl-out/crashes$ ls
usern@usern-virtual-machine:~/afl-2.52b/afl-out/crashes$ cd ..
usern@usern-virtual-machine:~/afl-2.52b/afl-out$ cd queue
usern@usern-virtual-machine:~/afl-2.52b/afl-out/queue$ ls
id:000000,orig:hello_world.txt
usern@usern-virtual-machine:~/afl-2.52b/afl-out/queue$
```

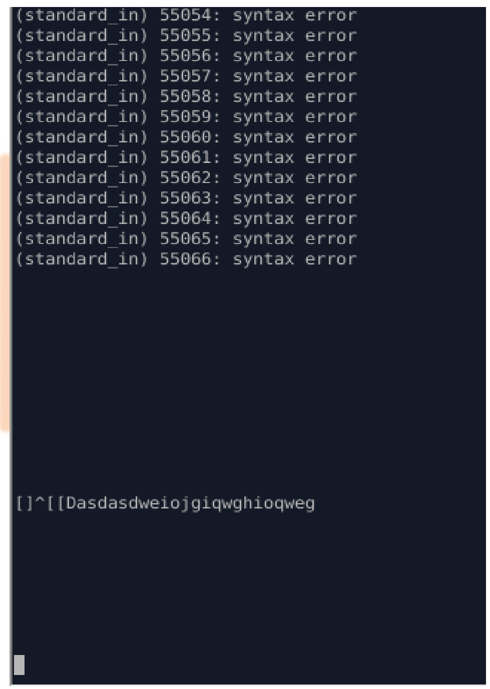
QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/analysis_vuln/1_Analysis_fuzzing_tools/KR.md

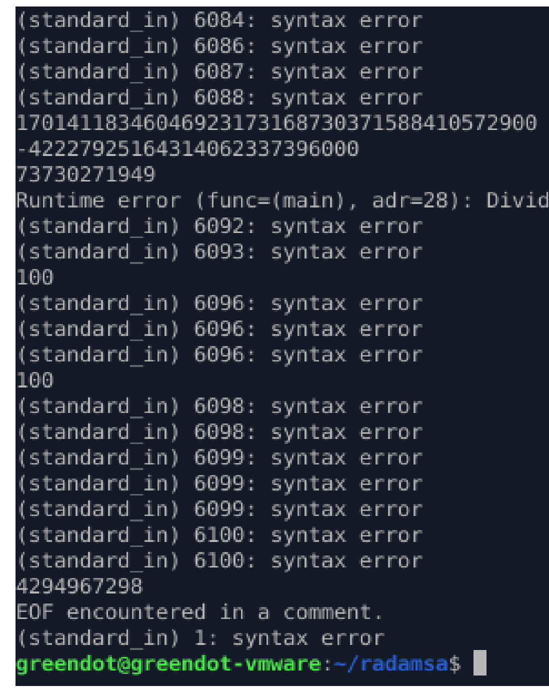
1. 현재 진행 사항

- 퍼징 툴 사용
  - Radamsa
    - 퍼징 테스팅 도구, 설치가 쉬움
      - echo "100 * (1 + (2 / 3))" | radamsa -n 10000 | bc





행 걸린 프로그램



비정상 종료

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/analysis_vuln/1_Analysis_fuzzing_tools/KR.md

2. 앞으로의 진행 계획

- 12월
  - RabbitMQ 문서 분석및 한글화 작업
  - AFL 퍼징 툴 분석
- 1월 ~ 2월
  - 소프트웨어 취약점 분석 방법 관련 논문 공부
  - 퍼징 툴을 통한 RabbitMQ 적용
  - github 공동화 작업 적용

2. 진행

- 앞으로의 진행 계획

	10월	11월	12월	1월	2월	3월	4월	5월
RabbitMQ 분석			→					
RabbitMQ 보안 취약점 분석								
취약점을 보완한 프로그램 개발								

QBQB

백업 1

- Melkor SHT(Section Header Table)

Section Header	설명
SHT_NULL	• 섹션 헤더를 비활성으로 표시함
SHT_PROGBITS	• 프로그램에 의해 정의된 정보를 보유하며, 프로그램의 형식과 의미는 프로그램에 의해서만 결정
SHT_STRTAB	• 문자열 테이블이 있음
SHT_RELA	• Elf64_Rela 대한 유형 또는 오브젝트 파일의 64 비트 클래스에 대한 유형과 같은 명시적인 재배치 항목을 보유함
SHT_HASH	• 심볼 해시 테이블을 포함함
SHT_DYNAMIC	• 동적 연결에 대한 정보가 있음 • 현재, 오브젝트 파일은 동적 세션을 하나만 소유 가능
SHT_NOTE	• 파일을 어떤식으로든 표시하는 정보가 들어있음