

# 최종 발표 슬라이드

- 보안 취약점 분석 및 보완 프로그램 개발 -



지도교수 : 이종혁

Captain : 201621571 손상진

Sailors : 201621110 권순홍

201621136 서민지

201621173 최서윤



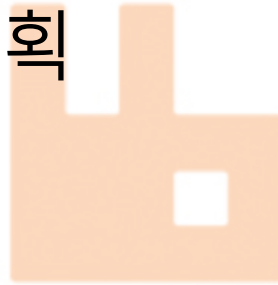
# 목차

---

## 1. 현재 진행 사항

- RabbitMQ 동작 분석
- 퍼징 관련 논문 분석
- 퍼징 툴 사용

## 2. 앞으로의 진행 계획



# 1. 현재 진행 사항

---

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



# 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

- ptjig

- 3. 도구를 통한 결과



QBQB

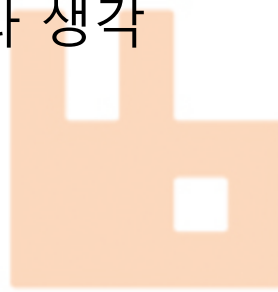
# 1. 현재 진행 사항

---

- 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
- ptijig
  - 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
  - 안정적이라고 생각했던 프로그램에서 많은 문제 발견
    - 24~33%의 경우가 프로그램을 중단, 중지



QBQB

# 1. 현재 진행 사항

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

```
orc_0369 orc_0754 orc_1139 orc_1524 orc_1909 orc_2294 orc_2679 orc_3064 orc_3449 orc_3834 orc_4219 orc_4604 orc_4989
orc_0370 orc_0755 orc_1140 orc_1525 orc_1910 orc_2295 orc_2680 orc_3065 orc_3450 orc_3835 orc_4220 orc_4605 orc_4990
orc_0371 orc_0756 orc_1141 orc_1526 orc_1911 orc_2296 orc_2681 orc_3066 orc_3451 orc_3836 orc_4221 orc_4606 orc_4991
orc_0372 orc_0757 orc_1142 orc_1527 orc_1912 orc_2297 orc_2682 orc_3067 orc_3452 orc_3837 orc_4222 orc_4607 orc_4992
orc_0373 orc_0758 orc_1143 orc_1528 orc_1913 orc_2298 orc_2683 orc_3068 orc_3453 orc_3838 orc_4223 orc_4608 orc_4993
orc_0374 orc_0759 orc_1144 orc_1529 orc_1914 orc_2299 orc_2684 orc_3069 orc_3454 orc_3839 orc_4224 orc_4609 orc_4994
orc_0375 orc_0760 orc_1145 orc_1530 orc_1915 orc_2300 orc_2685 orc_3070 orc_3455 orc_3840 orc_4225 orc_4610 orc_4995
orc_0376 orc_0761 orc_1146 orc_1531 orc_1916 orc_2301 orc_2686 orc_3071 orc_3456 orc_3841 orc_4226 orc_4611 orc_4996
orc_0377 orc_0762 orc_1147 orc_1532 orc_1917 orc_2302 orc_2687 orc_3072 orc_3457 orc_3842 orc_4227 orc_4612 orc_4997
orc_0378 orc_0763 orc_1148 orc_1533 orc_1918 orc_2303 orc_2688 orc_3073 orc_3458 orc_3843 orc_4228 orc_4613 orc_4998
orc_0379 orc_0764 orc_1149 orc_1534 orc_1919 orc_2304 orc_2689 orc_3074 orc_3459 orc_3844 orc_4229 orc_4614 orc_4999
orc_0380 orc_0765 orc_1150 orc_1535 orc_1920 orc_2305 orc_2690 orc_3075 orc_3460 orc_3845 orc_4230 orc_4615 orc_5000
orc_0381 orc_0766 orc_1151 orc_1536 orc_1921 orc_2306 orc_2691 orc_3076 orc_3461 orc_3846 orc_4231 orc_4616 Report_test.txt
```

```
user@ubuntu:~/Melkor_ELF_Fuzzer$ melkor -a test
[!] Dir 'orcs_test' already exists. Files inside will be overwritten !

<---- test
<x> <x>
I'll be corrupted 5000 times !

[+] Automatic mode
[+] ELF type detected: ET_DYN
[+] Selecting the metadata to fuzz

[+] Detailed log for this session: 'orcs_test/Report_test.txt'

[+] The Likelihood of execution of each rule is: Aprox. 10 % (rand() % 10 < 1)

[+] Press any key to start the fuzzing process...
```



QBQB

```
(SYM[23]->st_shndx = 0xc004) | SHT[26] SYM[23] rule [11] executed
(SYM[24]->st_shndx = 0x18) | SHT[26] SYM[24] rule [05] executed
(SYM[24]->st_other = 0x3a) | SHT[26] SYM[24] rule [15] executed
(SYM[25]->st_info = 0xd3) | SHT[26] SYM[25] rule [06] executed
(SYM[28]->st_name = 0x20) | SHT[26] SYM[28] rule [09] executed
(SYM[28]->st_size = 0x00000000000b3e8e) | SHT[26] SYM[28] rule [04] executed
(SYM[28]->st_other = 0xd6) | SHT[26] SYM[28] rule [15] executed
(SYM[35]->st_other = 0xe3) | SHT[26] SYM[35] rule [15] executed
(SYM[38]->st_other = 0x21) | SHT[26] SYM[38] rule [15] executed
(SYM[41]->st_info = 0xac) | SHT[26] SYM[41] rule [14] executed
(SYM[42]->st_name = 0x3119c5f2) | SHT[26] SYM[42] rule [02] executed
(SYM[43]->st_size = 0x0015e5e2359588a) | SHT[26] SYM[43] rule [04] executed
(SYM[47]->st_size = 0x3b5591245c77022) | SHT[26] SYM[47] rule [04] executed
(SYM[50]->st_size = 0x0000000007ffffff) | SHT[26] SYM[50] rule [04] executed
(SYM[58]->st_size = 0x0b1e3903c0000000) | SHT[26] SYM[58] rule [04] executed
(SYM[62]->st_shndx = 0x0) | SHT[26] SYM[62] rule [05] executed
(SYM[64]->st_value = 0x00000000c1006e20) | SHT[26] SYM[64] rule [10] executed

[+] Fuzzing the Dynamic section .dynamic with 31 entries
(DYN[2]->d_un_d_val = 0x00000000c1009041) | SHT[21] DYN[2] rule [06] executed

[+] Fuzzing the Note section .note.ABI-tag with 32 bytes

[+] Fuzzing the Note section .note.gnu.build-id with 36 bytes

[+] Fuzzing the String Table .dynstr with 164 bytes

[+] Fuzzing the String Table .strtab with 562 bytes

[+] Fuzzing the String Table .shstrtab with 254 bytes

[+] Fuzzing the Section Header Table with 29 entries
(SHT[0]->sh_offset = 0x0000000000000000) | SHT[0] rule [03] executed
(SHT[0]->sh_size = 0x00003337defc4f4) | SHT[0] rule [04] executed
(SHT[0]->sh_flags = 0x0000000000000000) | SHT[0] rule [10] executed
(SHT[3]->sh_addralign = 0x0defacedff00ff00) | SHT[3] rule [05] executed
(SHT[5]->sh_flags = 0x0051c8d8defc4f4) | SHT[5] rule [10] executed
(SHT[5]->sh_flags = 0x0051c8d8defc4f4) | SHT[5] rule [14] executed
(SHT[7]->sh_size = 0x0000000000000000) | SHT[7] rule [04] executed
(SHT[7]->sh_addralign = 0x00000000000000ff) | SHT[7] rule [05] executed
(SHT[7]->sh_entsize = 0x0000000000000000) | SHT[7] rule [06] executed
(SHT[8]->sh_flags = 0x00000000c4fed000) | SHT[8] rule [10] executed
(SHT[10]->sh_flags = 0x0000000000ff0002) | SHT[10] rule [32] executed
(SHT[12]->sh_type = 0xf) | SHT[12] rule [07] executed
(SHT[13]->sh_addralign = 0x00003337291551a2) | SHT[13] rule [05] executed
(SHT[14]->sh_addr = 0x00000000defc4f4) | SHT[14] rule [02] executed
(SHT[16]->sh_name = 0x0) | SHT[16] rule [01] executed
(SHT[16]->sh_type = 0x1, sh_flags = 0x4444444480000000, sh_size = 0x424242423294735b, sh_entsize = 0xc0000000010b00b5) | SHT[16] rule [19] executed

[+] Fuzzing the Program Header Table with 9 entries
(PHT[0]->p_flags = 0xffff0004) | PHT[0] rule [15] executed
(PHT[1]->p_flags = 0xb584) | PHT[1] rule [10] executed
(PHT[1]->p_flags = 0xffff0004) | PHT[1] rule [15] executed
```

511.1 0%



# 1. 현재 진행 사항

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

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

```
user@user-virtual-machine:~/afl-2.52b$ ./afl-fuzz -i ./testcases/others/text/ -o ./afl-out/ ./test-instr
afl-fuzz 2.52b by <lcantuf@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!

```
american fuzzy lop 2.52b (test-instr)

process timing
  run time : 0 days, 0 hrs, 0 min, 25 sec
  last new path : 0 days, 0 hrs, 0 min, 25 sec
  last uniq crash : none seen yet
  last uniq hang : none seen yet

cycle progress
  now processing : 1 (50.00%)
  paths timed out : 0 (0.00%)

stage progress
  now trying : havoc
  stage execs : 290/384 (75.52%)
  total execs : 39.0k
  exec speed : 1470/sec

fuzzing strategy yields
  bit flips : 0/56, 0/54, 0/50
  byte flips : 0/7, 0/5, 0/1
  arithmetics : 0/392, 0/21, 0/0
  known ints : 0/39, 0/140, 0/44
  dictionary : 0/0, 0/0, 0/0
  havoc : 1/15.9k, 0/22.0k
  trim : 33.33%/1, 0.00%

map coverage
  map density : 0.01% / 0.01%
  count coverage : 1.00 bits/tuple

findings in depth
  favored paths : 2 (100.00%)
  new edges on : 2 (100.00%)
  total crashes : 0 (0 unique)
  total tmouts : 3 (1 unique)

path geometry
  levels : 2
  pending : 0
  pend fav : 0
  own finds : 1
  imported : n/a
  stability : 100.00%

[cpu000:128%]
```

```
user@user-virtual-machine:~/afl-2.52b$ ls
Makefile      afl-as      afl-cntln  afl-gcc.c  afl-showmap  alloc-inl.h  docs      llwn_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.c  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-whatshup  dictionaries  libtokencap  testcases
user@user-virtual-machine:~/afl-2.52b$ cd afl-out
user@user-virtual-machine:~/afl-2.52b/afl-out$ ls
crashes  hangs  plot_data  queue
user@user-virtual-machine:~/afl-2.52b/afl-out$ cd crashes
user@user-virtual-machine:~/afl-2.52b/afl-out/crashes$ ls
user@user-virtual-machine:~/afl-2.52b/afl-out/crashes$ cd ..
user@user-virtual-machine:~/afl-2.52b/afl-out$ cd queue
user@user-virtual-machine:~/afl-2.52b/afl-out/queue$ ls
id:000000,orig:hello_world.txt
user@user-virtual-machine:~/afl-2.52b/afl-out/queue$
```



QBQB

# 1. 현재 진행 사항

- 퍼징 툴 사용

- Radamsa

- 퍼징 테스트 도구, 설치가 쉬움

- `echo "100 * (1 + (2 / 3))" | radamsa -n 10000 | bc`

```
greendot@greendot-vmware:~/radamsa$ echo "aaa" | radamsa
radamsa: 명령을 찾을 수 없습니다
greendot@greendot-vmware:~/radamsa$ echo "aaa" | radamsa
aaa
greendot@greendot-vmware:~/radamsa$ echo "aaa" | radamsa
aaa%p$` \u1NaN!xcalc$!!;xcalc$&
greendot@greendot-vmware:~/radamsa$ echo "aaa" | radamsa
aaaaa
greendot@greendot-vmware:~/radamsa$ echo "aaa" | radamsa
aa
greendot@greendot-vmware:~/radamsa$ echo "aaa" | radamsa
aa
greendot@greendot-vmware:~/radamsa$ echo "aaa" | radamsa
0esaaaq
greendot@greendot-vmware:~/radamsa$ echo "aaa" | radamsa
aaaaaaaaa
aaaaaaaaa
aaaaaaaaa
greendot@greendot-vmware:~/radamsa$ echo "aaa" | radamsa
R/N/Naaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
greendot@greendot-vmware:~/radamsa$
```

```
(standard_in) 55054: syntax error
(standard_in) 55055: syntax error
(standard_in) 55056: syntax error
(standard_in) 55057: syntax error
(standard_in) 55058: syntax error
(standard_in) 55059: syntax error
(standard_in) 55060: syntax error
(standard_in) 55061: syntax error
(standard_in) 55062: syntax error
(standard_in) 55063: syntax error
(standard_in) 55064: syntax error
(standard_in) 55065: syntax error
(standard_in) 55066: syntax error

(standard_in) 6084: syntax error
(standard_in) 6086: syntax error
(standard_in) 6087: syntax error
(standard_in) 6088: syntax error
17014118346046923173168730371588410572900
-42227925164314062337396000
73730271949
Runtime error (func=(main), adr=28): Divide
(standard_in) 6092: syntax error
(standard_in) 6093: syntax error
100
(standard_in) 6096: syntax error
(standard_in) 6096: syntax error
(standard_in) 6096: syntax error
100
(standard_in) 6098: syntax error
(standard_in) 6098: syntax error
(standard_in) 6099: syntax error
(standard_in) 6099: syntax error
(standard_in) 6099: syntax error
(standard_in) 6099: syntax error
(standard_in) 6100: syntax error
(standard_in) 6100: syntax error
4294967298
EOF encountered in a comment.
(standard_in) 1: syntax error
greendot@greendot-vmware:~/radamsa$
```



행 걸린 프로그램

비정상 종료

## 2. 앞으로의 진행 계획

---

- 12월
  - RabbitMQ 문서 분석 및 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								

# # 백업 1

- Melkor SHT(Section Header Table)

Section Header	설명
SHT_NULL	<ul style="list-style-type: none"><li>• 섹션 헤더를 비활성으로 표시함</li></ul>
SHT_PROGBITS	<ul style="list-style-type: none"><li>• 프로그램에 의해 정의된 정보를 보유하며, 프로그램의 형식과 의미는 프로그램에 의해서만 결정</li></ul>
SHT_STRTAB	<ul style="list-style-type: none"><li>• 문자열 테이블이 있음</li></ul>
SHT_RELA	<ul style="list-style-type: none"><li>• Elf64_Rela 대한 유형 또는 오브젝트 파일의 64 비트 클래스에 대한 유형과 같은 명시적인 재배치 항목을 보유함</li></ul>
SHT_HASH	<ul style="list-style-type: none"><li>• 심볼 해시 테이블을 포함함</li></ul>
SHT_DYNAMIC	<ul style="list-style-type: none"><li>• 동적 연결에 대한 정보가 있음</li><li>• 현재, 오브젝트 파일은 동적 세션을 하나만 소유 가능</li></ul>
SHT_NOTE	<ul style="list-style-type: none"><li>• 파일을 어떤식으로든 표시하는 정보가 들어있음</li></ul>