ROP 공격 및 SGX 취약점

한성대학교

김경호

https://youtu.be/DpFQ3ehX6Sw

ROP(Return Oriented Programming) Attack

- ROP 란?
  - strcpy, scanf와 같은 취약점 있는 함수를 이용한 Buffer Overflow 취약점이 있는 코드를 Gadget을 이용하여 함수 호출 및 조작하는 공격
  - 메모리 취약점을 막기 위한 방어 기법인 ASLR, DEP 등도 우회 가능

  - Gadget
    - 공격을 하기 위한 코드 조각

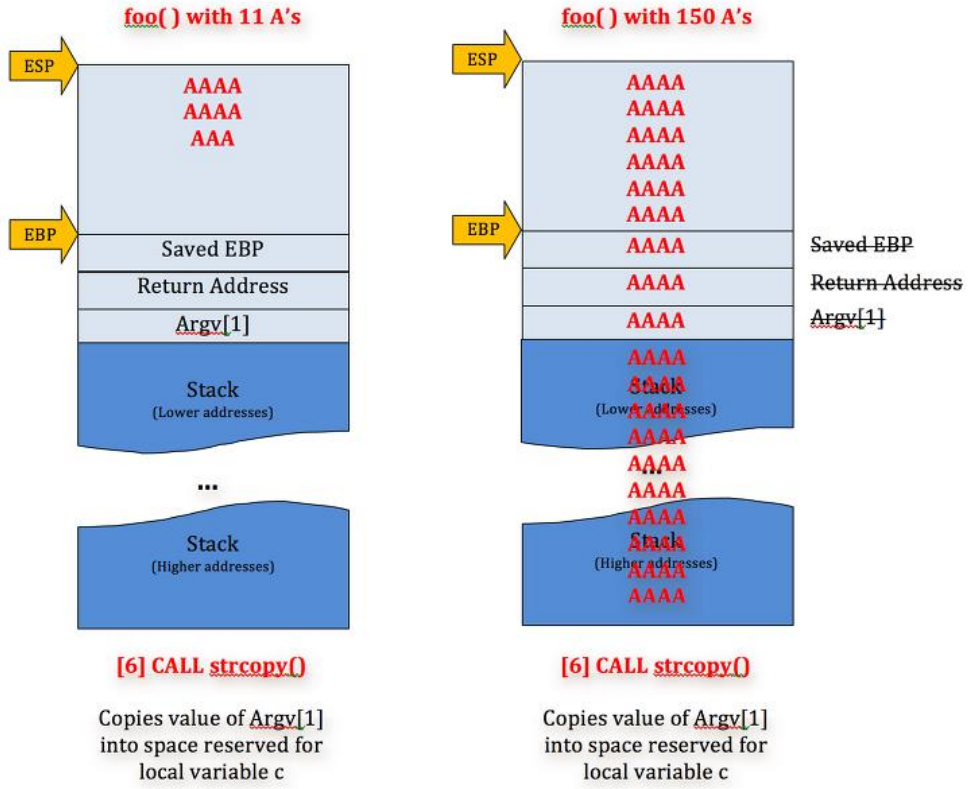
  - ASLR(Address Space Layout Randomization)
    - 실행 및 호출 할 때 마다 주소 배치를 무작위로 배정하는 기법

  - DEP/NX(Data Execution Protection/Non Executable)
    - 코드 영역 제외한 다른 영역에서 실행 권한 x

Background (BufferOverflow)

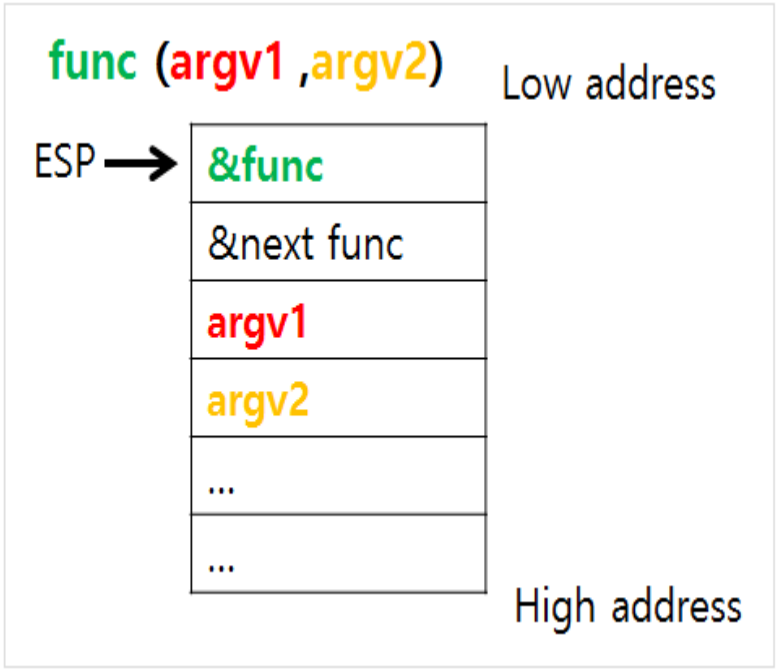
- Strcpy 함수의 취약점을 이용한 Buffer Overflow 공격
- 함수 호출 후 return 해야 하는 주소를 공격 코드로 분기

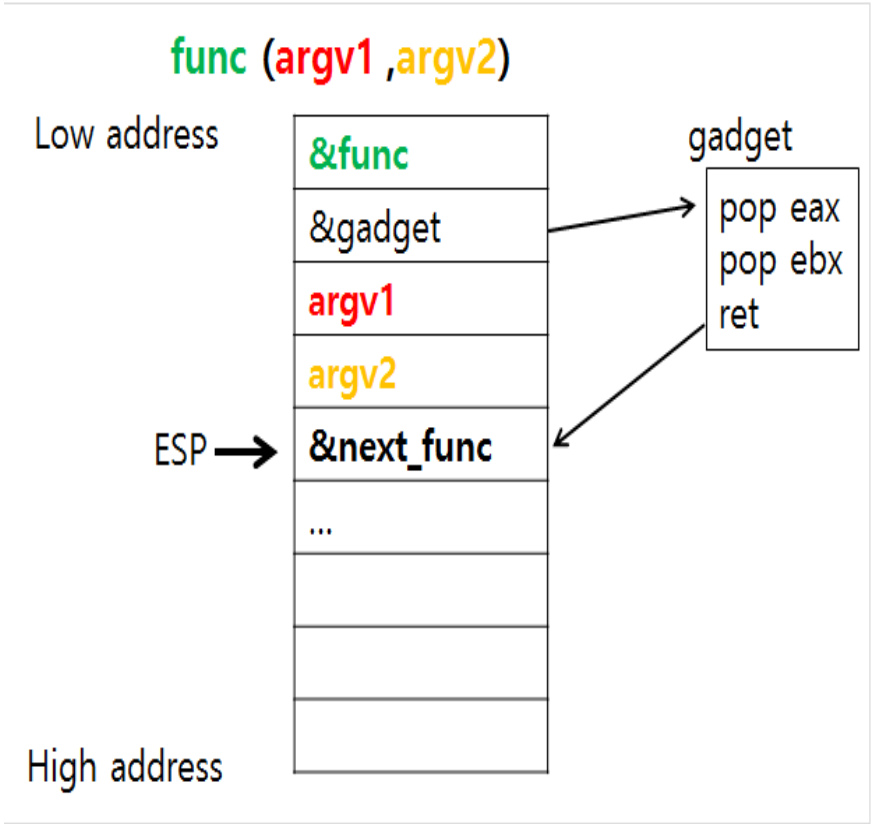
```c
#include <stdio.h>

int main(int argc, char * argv[]) {
  char buf[16];
  if(argc != 2)
    return -1;
  strcpy(buf, argv[1]);
  return 0;
}
```

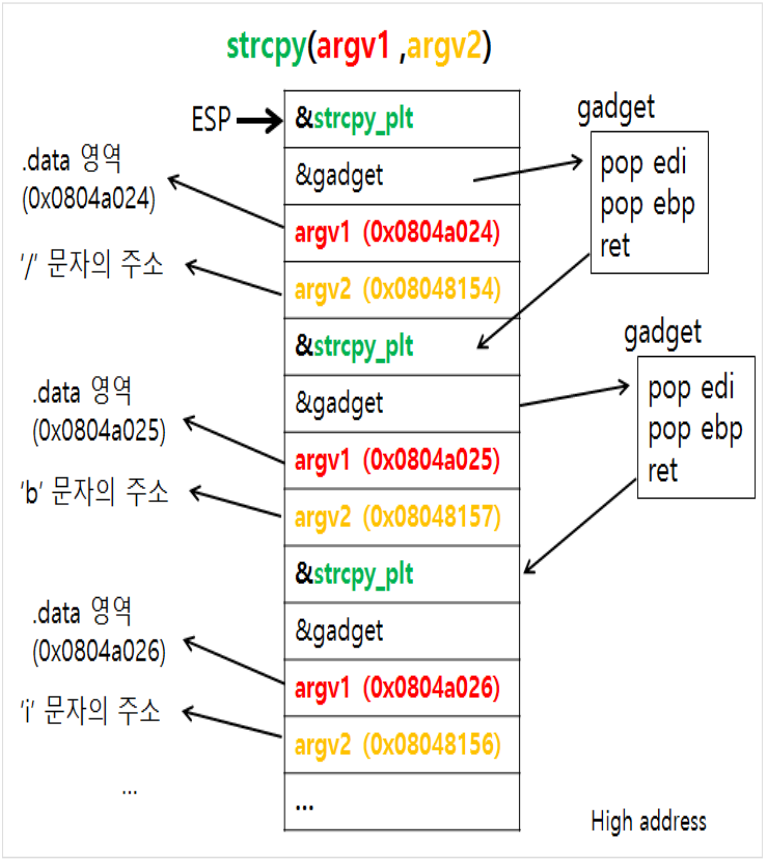


Background (ROP 기본)

- 함수 호출 과정에서 Stack Memory를 이용한 Parameter 전달
- 함수 끝나면 이전 실행했던 Code로 Jump





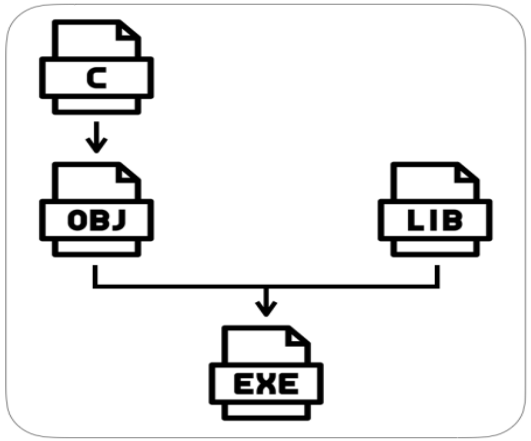


ROP 메모리 보안 우회 기법 (Link)

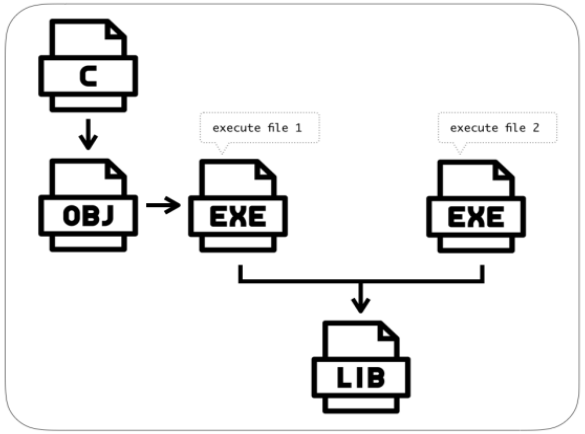
- Link 란?
  - 필요한 오브젝트 파일을 연결시키는 작업

- Static Link
  - 라이브러리 코드를 실행 파일 내부에 함께 컴파일
  - 함수 액세스 속도 빠르지만 실행 파일 커짐

- Dynamic Link
  - 공유 라이브러리 사용
  - 코드 크기 작지만 실행 속도 느림
  - 라이브러리 파일 없으면 실행 x



Static Link

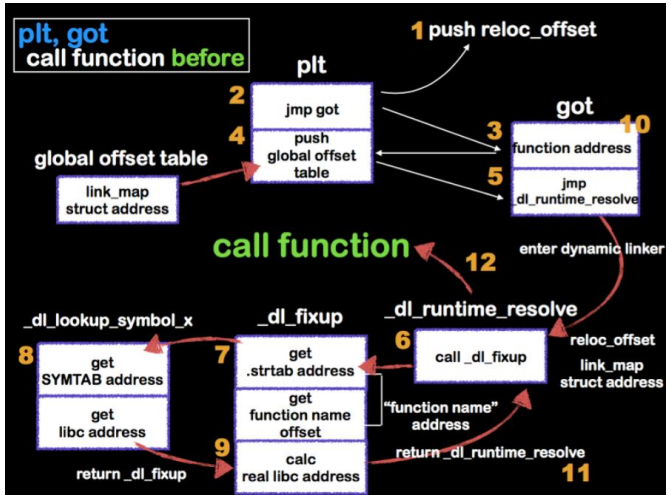


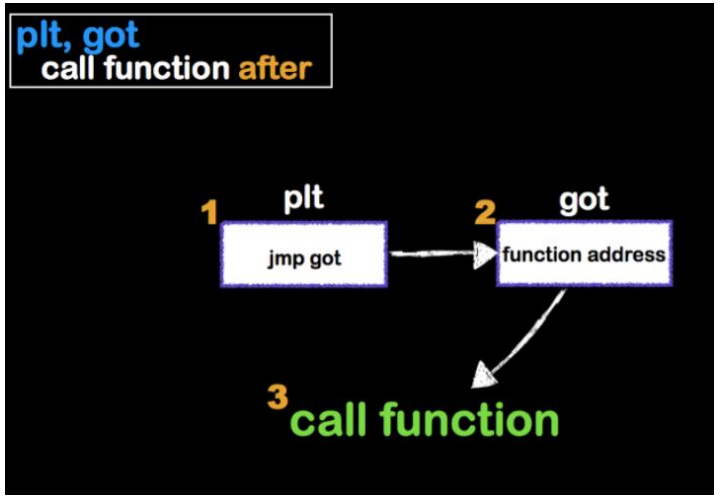
Dynamic Link

ROP 메모리 보안 우회 기법 (PLT & GOT)

- PLT (Procedure Linkage Table)
  - 외부 프로시저 연결 테이블.
  - PLT를 통해 다른 라이브러리에 있는 프로시저를 호출해 사용할 수 있다.

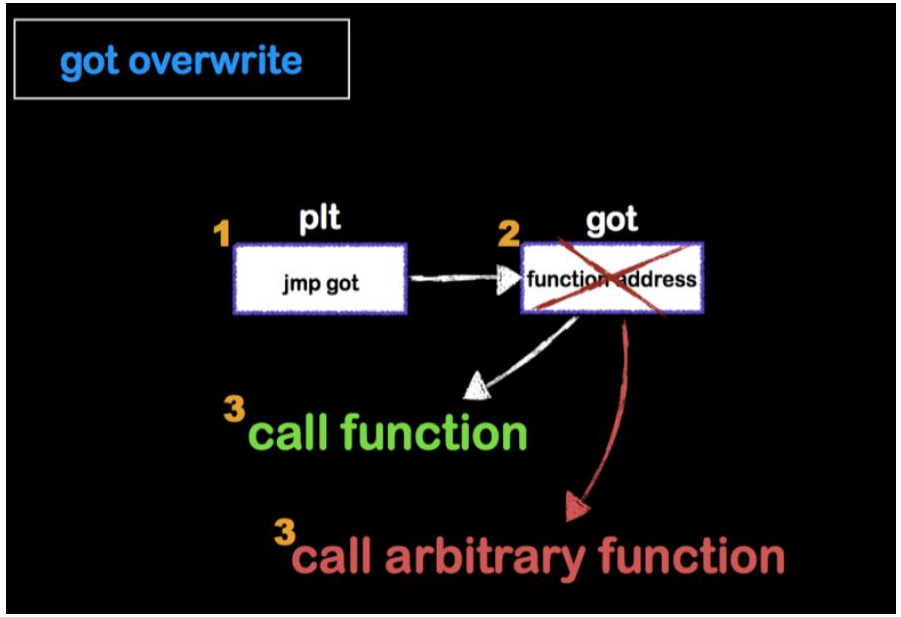
- GOT (Global Offset Table)
  - PLT가 참조하는 테이블. 프로시저들의 주소 저장

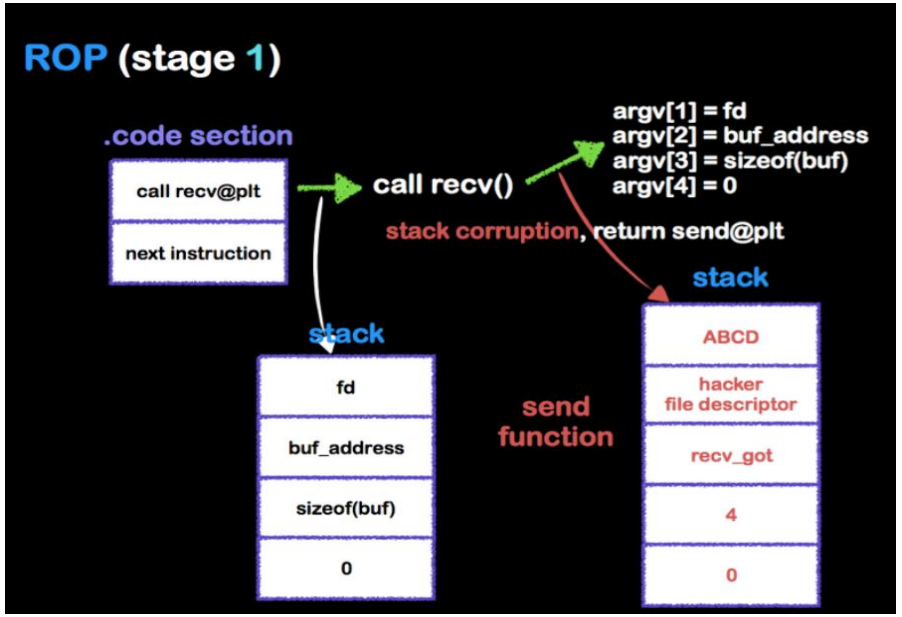




ROP 메모리 보안 우회 기법 (PLT & GOT)

- ASLR 우회 기법
  - 라이브러리 코드의 주소를 이용
  - GOT 주소를 공격자가 원하는 함수로 변환하여 공격 코드 실행
  - ASLR로 인한 주소 랜덤화 우회 가능
  - RTL Chaining을 이용해서 Parameter 변환 후 공격 코드 실행





Haking in Darkness

- ROP를 이용한 메모리 손상 취약점 이용
- Enclave의 상태를 3개의 Oracle을 이용해 전송

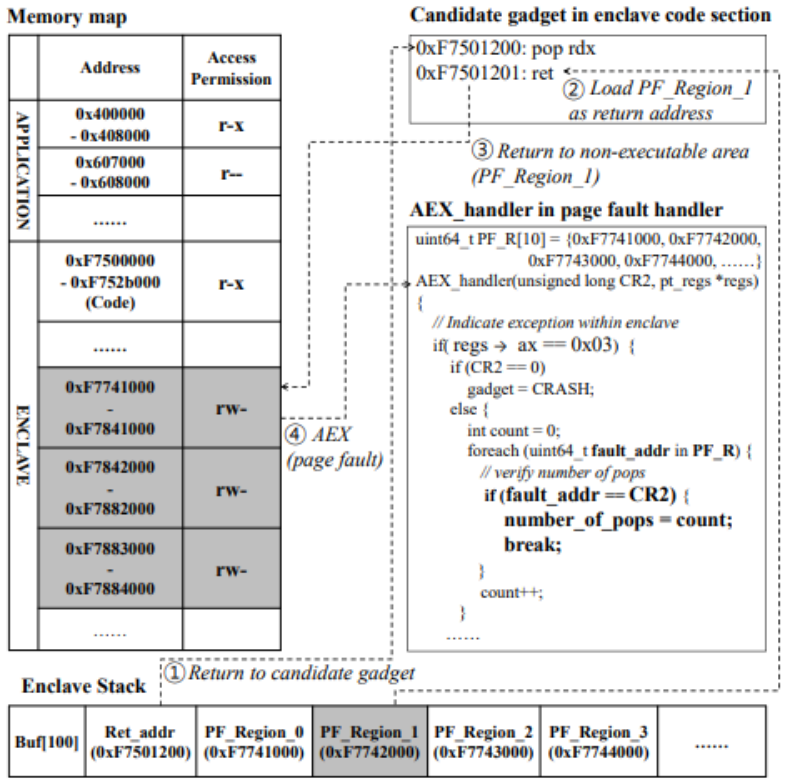
  (1) ret 명령 전에 레지스터 팝 수 감지 (2) Enclave 레지스터 값을 공개 (3) Enclave 메모리 내용 누출
- ROP를 이용한 Malware 실행
- Malware를 이용한 MITM(Man in the Middle) Attack 가능
- Malware를 활용한 Remote Attestation 조작 가능

Haking in Darkness (Assumption)

- 코드에 ENCLU 명령이 있어야함
  - Gadget을 찾을 때 사용
- 코드에 ROP gadget 이 있어야함 (pop register)
  - 레지스터를 이용해 Parameter 전달
- 코드에 취약점 함수가 있어야함 (memcpy, strcpy, etc..)
- Enclave 내부 접근을 제외한 System 전체 권한이 있어야함
- Intel의 표준 SDK로 컴파일 해야함

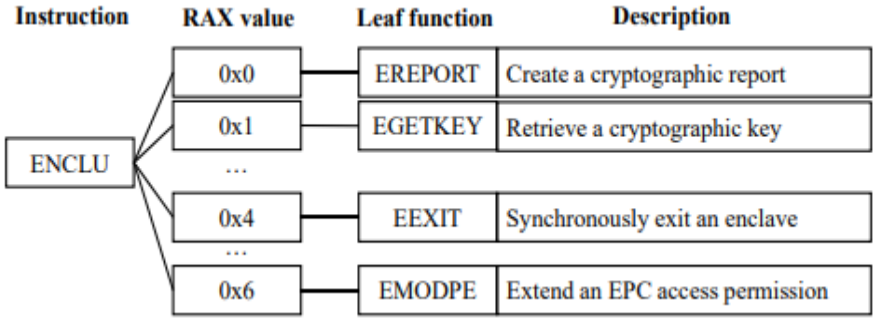
Haking in Darkness (Oracle 1)

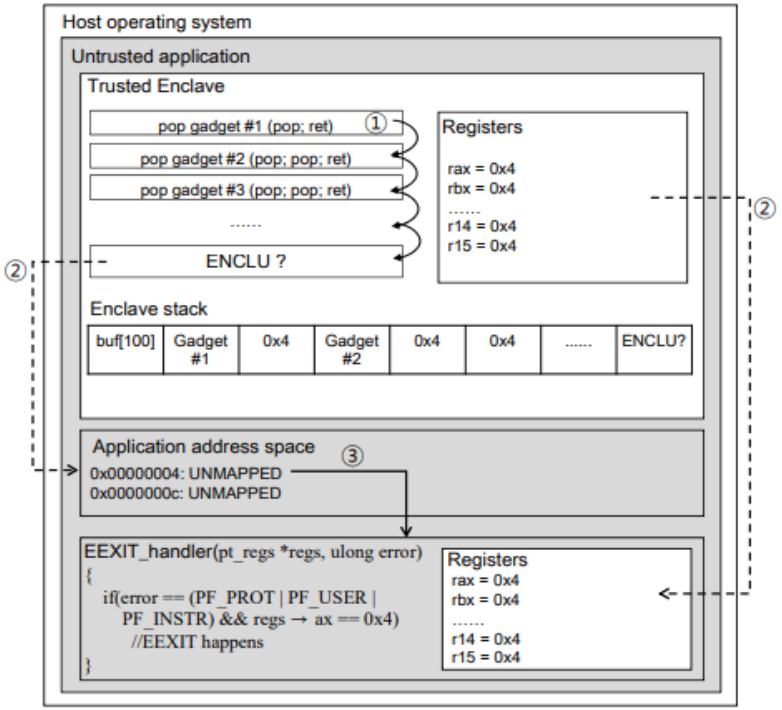
- 레지스터를 이용하는 Gadget 찾기
- AEX 및 Page fault를 이용
  - 예외 발생시 cr2레지스터를 이용하여 pop의 개수를 파악

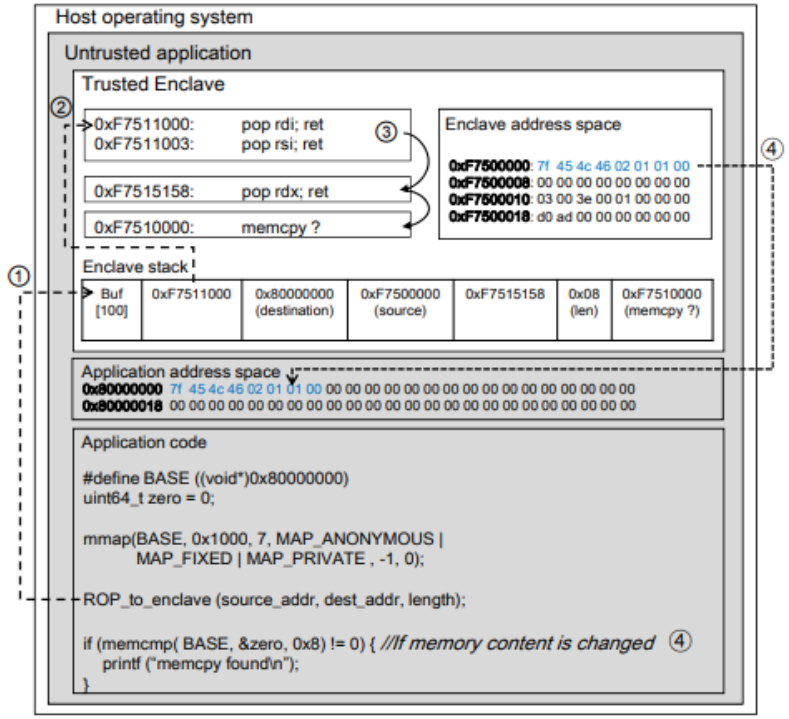


Haking in Darkness (Oracle 2)

- 찾은 Gadget 중 필요한 레지스터를 사용하는 Gadget 구분
- EEXIT 함수를 이용하여 식별
  - ENCLU 명령을 찾은 뒤 Parameter에 찾은 Gadget을 사용
  - EAX의 값이 0x4인 경우 Enclave가 종료됨





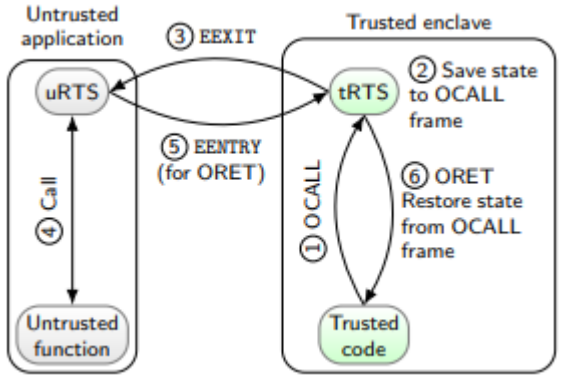


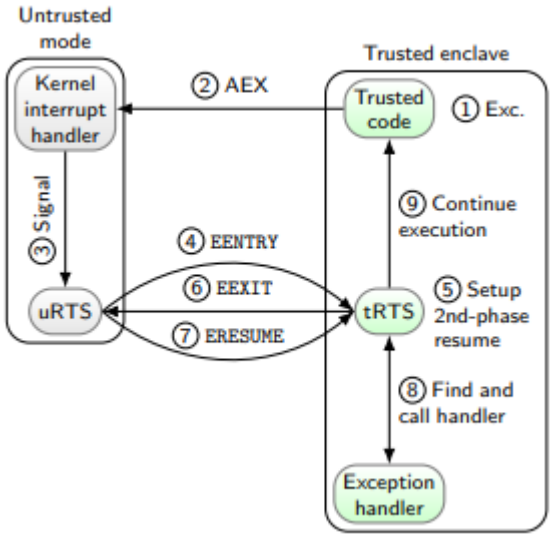
Haking in Darkness (Oracle 3)

- Memcpy() 함수를 이용하여 ROP Chain 제작
- 비 신뢰 메모리에 Enclave 메모리 Copy
- 비밀 데이터 해킹 뿐만 아니라 Malware 코드 실행 또한 가능

Efficient Code-Reuse Attacks

- ROP(Return-Oriented Programming)을 사용한 메모리 공격
- 기존 연구의 대응방안인 SGX-Shield를 통한 무작위화 구현도 뚫음
- SGX-SDK의 일부 무작위화 안되는 취약점을 이용
  - tRTS
- User 권한으로 공격 가능





Efficient Code-Reuse Attacks (Assumption)

- 메모리 취약점 코드가 존재해야함
- Malware를 Enclave가 접근할 수 있는 메모리에 적재해야 함
  - 응용 프로그램 또는 Enclave를 예측 가능한 주소에 데이터를 할당하도록 조정
- Enclave 외부 Memory layout 알아야 함 (가상주소)
- 실행파일 바이너리 분석이 가능해야 함
- 시스템 권한은 필요 없음

Efficient Code-Reuse Attacks (ORET & CONT)

- ORET
  - tRTS에 있는 함수로 OCALL 이후 CPU Context를 복원하는 명령
  - Stack 조작이 가능해야함 (코드 취약점)
  - Rdi 및 레지스터 서브 셋 제어 가능

- CONT
  - tRTS에 있는 함수로 Exception 이후 CPU Context를 복원하는 명령
  - 모든 레지스터를 완벽하게 제어 가능

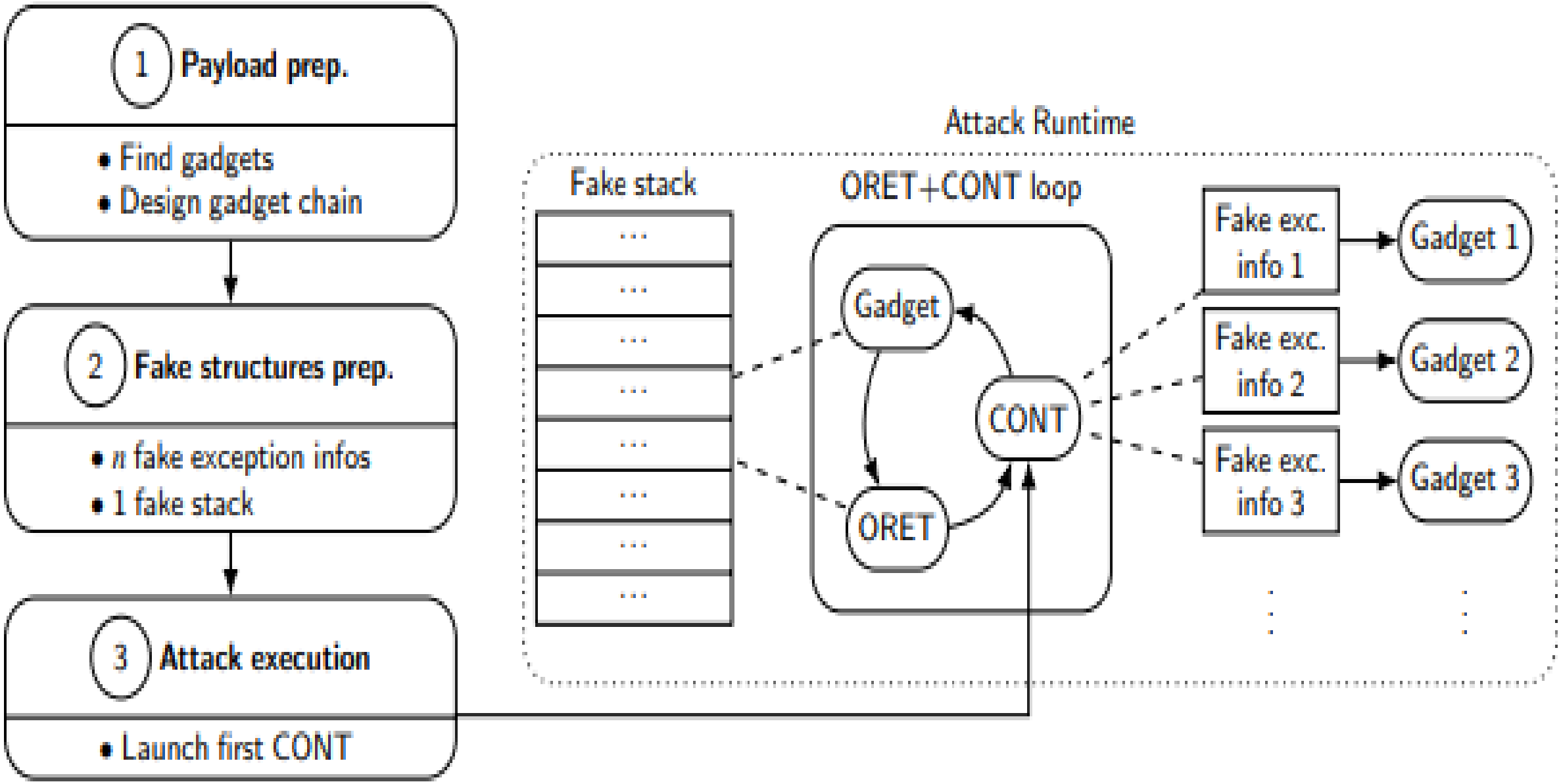
- ORET + CONT
  - ORET을 이용하여 rdi 레지스터 설정 후 CONT 이용하여 레지스터 값 호출

Efficient Code-Reuse Attacks (Work flow)

- 1. Payload Preparation
  - Enclave 바이너리 분석으로 Gadget 획득
  - Gadget을 chainning 하여 Payload 제작
  - Asm_oret() 함수 offset 확인

- 2. Fake structures Preparation
  - Enclave는 외부 메모리 접근도 가능한 점을 이용하여 가짜 structure 제작
  - CONT을 위한 가짜 예외 정보 구조 및 가짜 스텍 구조(ORET 재실행 위해)

- 3. Attack execution
  - 취약점 이용하여 CONT 실행 후 ORET + CONT loop 실행


1 Payload prep.
• Find gadgets
• Design gadget chain
2 Fake structures prep.
• n fake exception infos
• 1 fake stack
3 Attack execution
• Launch first CONT
Attack Runtime
Fake stack
ORET+CONT loop
Gadget
ORET
CONT
Fake exc. info 1
Fake exc. info 2
Fake exc. info 3
Gadget 1
Gadget 2
Gadget 3

SGX-Bomb

- Rowhammer 공격을 이용한 Bit flip 발생
- Bit flip을 이용하여 Enclave의 무결성 검사 실패 유도
- 무결성 실패 -> System 정지
- Dos 발생
- User 권한으로 공격 가능

SGX-Bomb (Assumption)

- 대상 머신의 DRAM 모듈은 Rowhammer 공격에 취약
- 유저 레벨의 실행 환경 및 Enclave 프로그램 실행 가능 환경
- 공격자는 시스템에 물리적 접근 할 수 없음.

SGX-Bomb (Row Hammer Attack)

- DRAM 의 하드웨어적 결함
- Cell 밀집도가 높아져서 한 Row에 반복적 접근을 할 경우 bit flip 발생
- DDR4에서도 동일한 취약점 발견 사례 있음

Q & A