메모리 취약점 공격 분석

-버퍼 오버플로우 공격-

안태진, 이현제, 최서윤, 권순홍, 이종혁*

({taejin, hyunje}@codecure.smuc.ac.kr,

{seoyun, soonhong, jonghyouk}@pel.smuc.ac.kr)

상명대학교 보안동아리 CodeCure

목 차

서론

- 배경
  - 현재 대부분의 취약점 공격은 메모리 공격 양상을 띰
  - 메모리 공격 양상의 큰 흐름은 버퍼 오버플로우(Buffer Overflow)
  - RTL은 대표적인 공격 기법

  - ASLR 보호 메커니즘으로 인해 RTL 공격 우회 가능
  - 이후 ROP 공격이 제안됨

  - 가젯을 만들 때 필요한 RET 명령어 검색의 어려움으로 인해 JOP 공격 기법이 새롭게 제안됨

서론

- 전개
  - 버퍼 오버플로우를 기반으로한 메모리 취약점 공격 기법 분석
  - 2장 관련 연구
    - 스택 프레임 구조, 쉘코드
  - 3장 메모리 취약점 공격 서술
    - 발전과정, RTL, ROP, JOP
  - 4장 결론 및 향후 연구

목 차

관련 연구

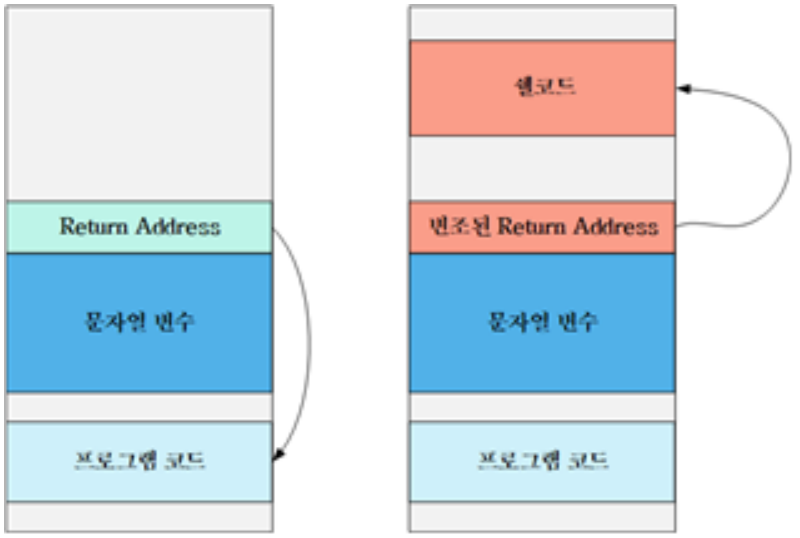
- 스택 프레임 구조
  - CALL(호출) 명령어 실행시 스택 프레임이 생성되고 스택 프레임이 스택에 저장
  - 스택 프레임에 저장되는 정보
    - 매개변수, 지역 변수, RET 주소
      - RET 주소 (반환 주소)
        - 호출된 함수를 호출한 명령어의 주소 값을 가지고 있음
        - 함수의 호출 완료시 RET 주소를 보고 원래 코드 실행 흐름으로 복귀
  - 스택은 높은 주소에서 낮은 주소로 할당

목 차

- 서론
- 관련연구
  - 스택 프레임 구조
  - 쉘코드
- 메모리 공격 기법
  - 발전 과정
  - RTL
  - ROP
  - JOP
- 결론 및 향후 연구

관련 연구

- 쉘코드 (Shellcode)
  - 시스템의 특정 명령을 실행하는 작은 크기의 프로그램
  - 소프트웨어 취약점 공격 이후 실행될 작은 크기의 프로그램으로 사용
  - 메모리에 쉘코드를 올리고 RET 주소를 쉘코드가 저장된 메모리의 주소로 덮어 씌워 쉘 실행



관련 연구

- 쉘코드 (Shellcode)
  - 종류
    - 로컬 쉘코드 (Local Shellcode)
      - 공격자가 시스템의 접근 권한을 가지고 있는경우 프로세스를 공격하여 해당 프로세스와 같은 높은 권한을 획득하기 위해 사용
    - 원격 쉘코드 (Remote Shellcode)
      - 공격자가 네트워크상의 다른 시스템 내부에 취약점이 있는 프로세스 공격하고자 하는 경우 사용
      - 연결 방식에 따라 두가지로 나뉨
        - 리버스 쉘코드 (Reverse Shellcode)
          - 목표 시스템으로부터 공격자에게 연결을 요청하도록 하는 쉘코드
        - 바인드 쉘코드 (Bind Shellcode)
          - 공격자가 목표 시스템의 특정 포트를 바인드하여 대상 시스템에 연결

목 차

메모리 공격 기법

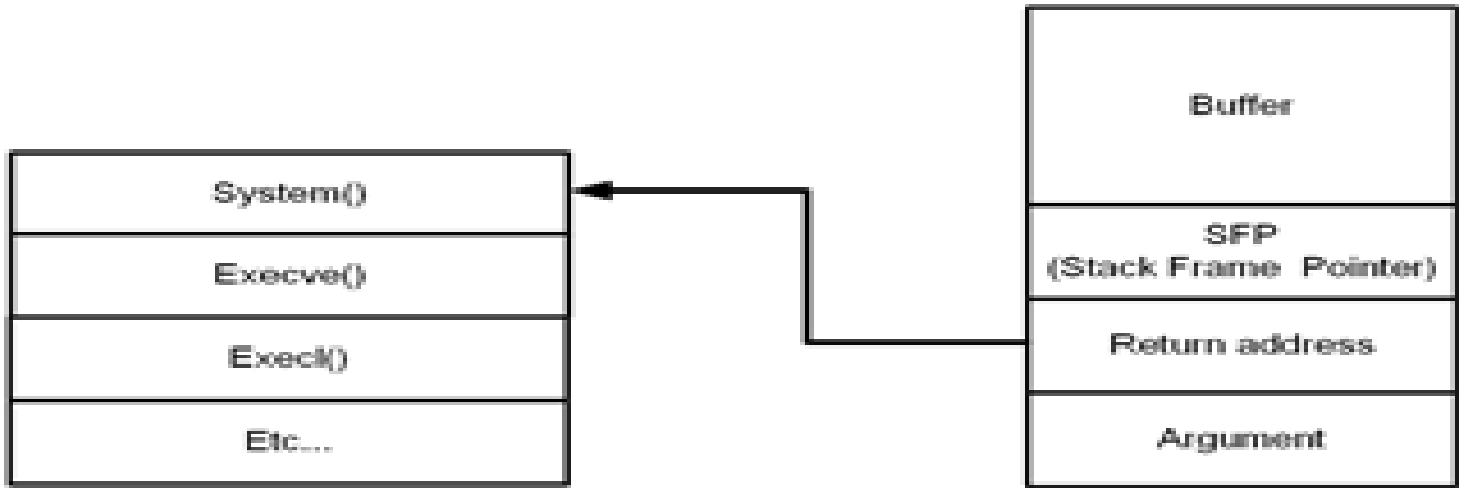
- 발전과정
  - 초기 메모리 공격
    - 코드 삽입 (Code Injection)
      - 악성코드 직접 메모리에 삽입, 스택을 그 코드로 향하게 수정하여 공격
    - W^X 보호 메커니즘으로 방어
      - W^X (Write XOR Execute)
        - 메모리를 쓰기 가능한 영역과 실행 가능한 영역으로 구분
  - 최근 메모리 공격
    - 코드 재사용 (Code Reuse)
      - 코드 영역에 존재하는 코드 재조합하여 사용
        - 코드 영역 (Code Segment)
          - 프로그램 실행하는 기계어 코드 저장되는 영역
    - W^X 보호 메커니즘 우회 가능

목 차

- 서론
- 관련연구
  - 스택 프레임 구조
  - 쉘코드
- 메모리 공격 기법
  - 발전 과정
  - RTL
  - ROP
  - JOP
- 결론 및 향후 연구

메모리 공격 기법

- RTL (Return To Libc)
  - 프로그램에 존재하는 시스템 라이브러리의 함수를 호출하여 제어 흐름 탈취, 쉘 코드 실행
  - 스택 프레임의 RET 주소를 변경하여 공격자가 원하는 시스템 함수의 위치로 제어 흐름 변경



메모리 공격 기법

- RTL (Return To Libc)
  - 공유 라이브러리에 올라가는 메모리 주소는 바뀌지 않고, 운영체제가 달라져도 스택의 크기가 변하지 않는다는 점을 활용한 공격
  - 한계
    - ASLR 보호 메커니즘으로 RTL 공격이 어려워짐
      - ASLR (Address Space Layout Randomization)
        - 리눅스 보호 메커니즘
        - 프로세스 내에 매핑되는 오브젝트에 대하여 호출 및 실행시 실행되는 주소를 랜덤화하는 보호 메커니즘

목 차

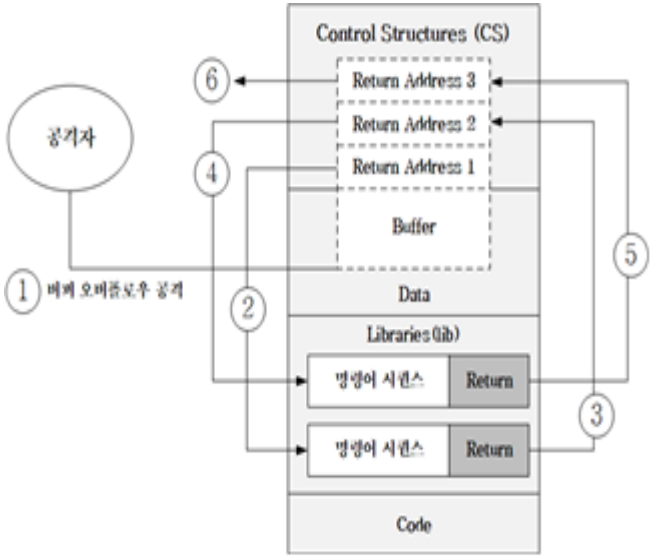
- 서론
- 관련연구
  - 스택 프레임 구조
  - 쉘코드
- 메모리 공격 기법
  - 발전 과정
  - RTL
  - ROP
  - JOP
- 결론 및 향후 연구

메모리 공격 기법

- ROP (Return-Oriented Programming)
  - ASLR로 인한 RTL의 한계점 보안하기 위해 제안
  - 코드 영역에 이미 존재하는 코드들을 재조합하여 제어 흐름 탈취
  - 가젯을 매개체로 원하는 코드 흐름을 연속적으로 제어
    - 가젯 (Gadget)
      - 코드 영역에 이미 존재하는 기계어 코드들을 재조합하여 만든 명령어 시퀀스
      - 하나의 가젯은 다음 가젯의 시작 주소를 반환하는 RET 명령어 존재
      - 따라서 하나의 가젯의 수행을 마치면 다음 가젯의 시작 위치로 제어 흐름 조작

메모리 공격 기법

- ROP (Return-Oriented Programming)
  - 공격 과정
    - 1. 가젯 생성
    - 2. 쓰기 가능한 메모리에 가젯의 시작 주소 기록
    - 3. 취약점 있는 함수 사용하여 SP를 가젯 1의 반환 주소로 변경
    - 4. 가젯 1은 SP 바꾸는 명령어 수행하여 SP에 가젯 2로 반환되는 주소값 저장
    - 5. 공격자의 반환 주소로 반환, 가젯 2 실행
    - 6. RET(반환 명령어) 반복 실행되면서 공격자가 원하는 실행 흐름 제어



메모리 공격 기법

- ROP (Return-Oriented Programming)
  - 장점
    - 코드 삽입, 수정 없이 공격
      - W^X 우회 가능
    - 프로그램에 존재하는 코드 재사용
      - 아키텍처의 제한 없이 적용 가능

목 차

- 서론
- 관련연구
  - 스택 프레임 구조
  - 쉘코드
- 메모리 공격 기법
  - 발전 과정
  - RTL
  - ROP
  - JOP
- 결론 및 향후 연구

메모리 공격 기법

- JOP (Jump-Oriented Programming)
  - ROP의 문제점 발생
    - RET 명령어에 대한 검색의 한계점 발생
    - ROP 공격에 대한 보호 메커니즘 역시 많이 제안
  - ROP의 문제점 보안하기 위해 새롭게 제안
  - RET 명령어보다 수가 많은 JMP 명령어 시퀀스 사용하여 제어 흐름 탈취
    - 이에 RET 명령어의 특징을 이용한 보호 메커니즘도 우회 가능

메모리 공격 기법

- JOP (Jump-Oriented Programming)
  - 공격 방법
    - 가젯을 만들어 가젯들로 분기하여 제어 흐름 탈취
    - 한 가젯에서 다른 가젯으로 지속적으로 분기하여 공격 수행
    - 두가지 방법으로 구현 가능
      - 1) 트램폴린 (Trampoline) 이용
      - 2) 디스패치 테이블 (Dispatch Table) 이용

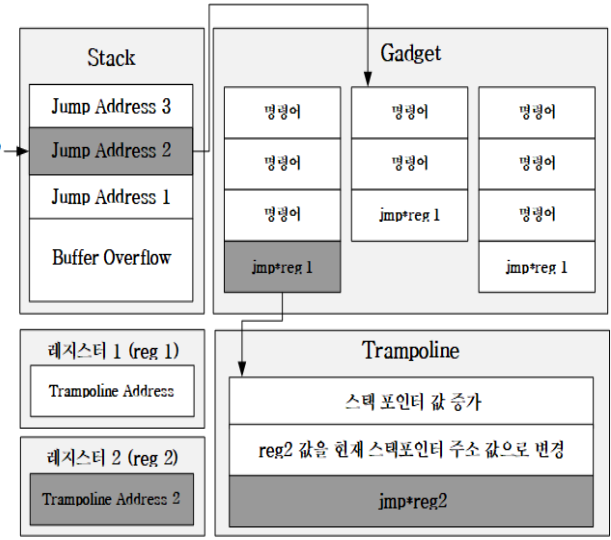
메모리 공격 기법

- JOP (Jump-Oriented Programming)
  - 공격 방법
    - 1) 트램폴린 (Trampoline) 이용
      - 트램폴린 (Trampoline)
        - 프로그램은 트램폴린 체크 명령어를 가지고 있음
        - 이 명령어 실행을 통해 트램폴린의 상태 확인
        - 트램폴린이 참(True)이면 분기,

        거짓(False)이면 분기하지 않음

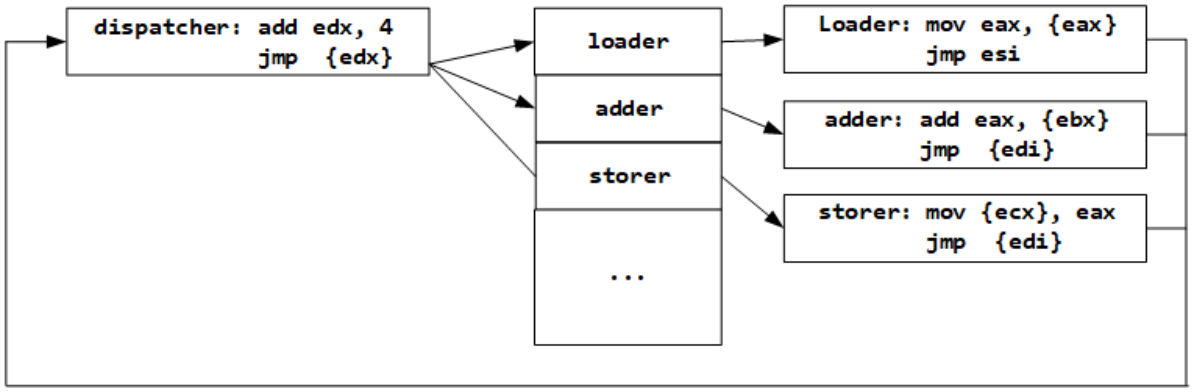
      - 분기시 Update-Load-Branch 기능 수행
        - Update 기능: SP 증가
        - Load 기능: 특정 레지스터에 SP가 참조하는

        메모리 값 저장
        - Branch 기능: 특정 레지스터로 흐름 변경



메모리 공격 기법

- JOP (Jump-Oriented Programming)
  - 공격 방법
    - 2) 디스패치 테이블 (Dispatch table) 이용
      - PC 변경시키는 JMP 명령어의 특징 이용한 공격 기법
        - PC (Program Counter)
          - 다음 실행될 명령어의 주소를 저장하고 있는 레지스터
      - SP 사용하지 않고 디스패처가 제어 흐름 변경
      - 가젯들은 디스패치 테이블에 들어가 있음
      - 디스패처 안에서 다음 가젯의 주소로 PC 증가시키며 가젯들 실행



목 차

- 서론
- 관련연구
  - 스택 프레임 구조
  - 쉘코드
- 메모리 공격 기법
  - 발전 과정
  - RTL
  - ROP
  - JOP
- 결론 및 향후 연구

결론

- 결론
  - 컴퓨터의 취약점 중 메모리 취약점,
    공격 기법 중 버퍼 오버플로우 중점으로 연구

- 향후 연구
  - 다음에는 방어 기법에 대한 연구를 수행할 예정
  - 위 세 가지 공격 기법을 실제 코드로 구현하는 연구도
    수행할 예정

감사합니다!