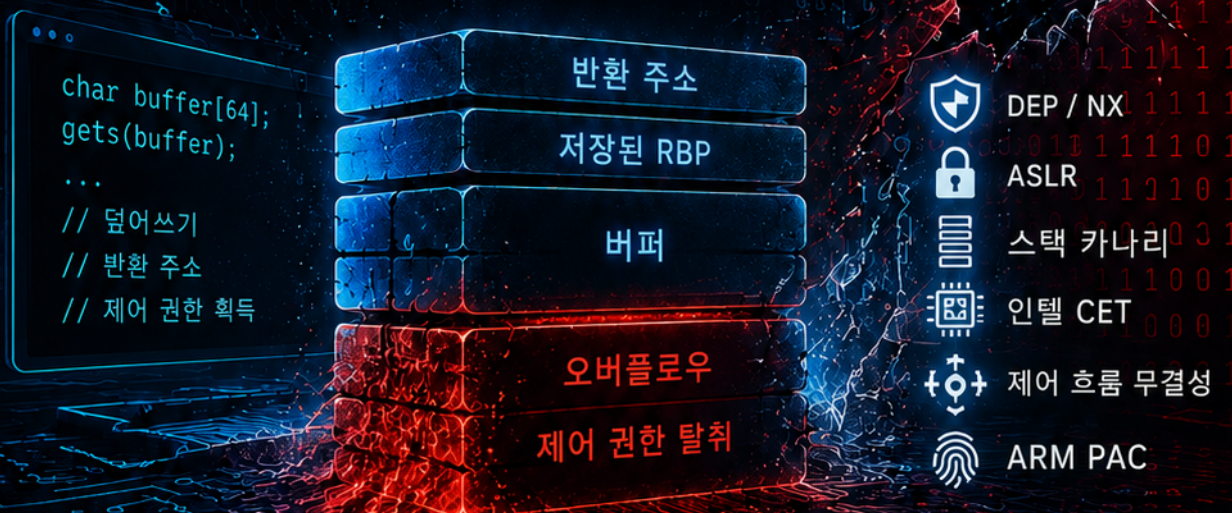
버퍼 오버플로우

악용 및 방어 우회

그리고

기억 영역 손상 및 방어 우회

기억 영역 손상, 취약점 악용,
악용 기법 및 방어 우회를 깊이 있게 탐구하다



익스플로잇 개발 | ROP / JOP 셸코드 | 취약 및 분석 | 퍼징 및 분석 | 방어 전략

Steve T.

버퍼 오버플로우 악용 및 방어 우회

메모리 손상 취약점, 익스플로이팅 및 방어에 대한 심층 연구

Steve T. Publications

이 책은 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 할인 중입니다.

이 버전은 2026-07-16에 출간되었습니다.

이것은 Leanpub 책입니다. Leanpub은 작가들과 출판자들에게 Lean 출판 과정을 가지고 힘을 실어 줍니다. Lean Publishing은 가벼운 도구들과 많은 독자들의 피드백과 지지를 통해 일단 시작하면 앞으로 나아갈 힘을 구축하고, 당신이 제대로 된 책을 쓸때까지 계속해서 책을 출간하도록 하는 행위입니다.

© 2026 Steve T. Publications

차례

메모리 손상 취약점, 익스플로이팅 및 방어에 대한 심층 연구

이 책에 대하여: 이 책은 CPU 아키텍처 기반부터 현대의 보안 환경을 정의하는 고급 익스플로이팅 기법 및 현대적 방어 수단까지, 메모리 손상 취약점에 대해 포괄적이고 기술적으로 엄밀한 내용을 제공합니다. 컴퓨터 메모리 아키텍처, 프로세스 레이아웃, 어셈블리 언어 기초부터 시작하여 스택 기반 및 힙 기반 버퍼 오버플로, 사용 후 해제(use-after-free), 형식 문자열 취약점 및 관련 메모리 손상 버그로 진행됩니다. 이 책은 쉘코드, 리턴 지향 프로그래밍, 점프 지향 프로그래밍, 그리고 Windows와 Linux 플랫폼 모두에 걸친 정보 누출을 포함한 익스플로잇 개발 개념을 다룹니다. DEP/NX, ASLR, 스택 카나리, Intel CET, ARM PAC, 제어 흐름 무결성과 같은 현대적 완화 기법을 심층적으로 검토하고, 익스플로이팅 연구와 방어 메커니즘 사이의 지속적인 군비 경쟁도 함께 다룹니다. 마지막 장들은 취약점 연구 방법론, 퍼징(fuzzing), 안전한 코딩 관행 및 사건 대응을 다룹니다. 이 책은 방어적 목적으로 메모리 손상을 가장 깊은 수준에서 이해하고자 하는 보안 전문가, 리버스 엔지니어, 취약점 연구자 및 고급 학습자를 위한 것입니다. 서술된 모든 기법은 윤리적 보안 연구와 책임 있는 공개의 맥락에서 제시됩니다.

서론

1988년 11월, 코넬 대학의 로버트 모리스라는 이름의 대학원생이 인터넷을 통해 확산된 최초의 주요 컴퓨터 웜인 모리스 웜으로 알려지게 될 프로그램을 출시했습니다. 이 웜은 수천 대의 유닉스 시스템에서 실행되던 서비스인 fingerd 데몬의 버퍼 오버플로 취약점을 악용했습니다. 프로그램의 고정 크기 버퍼가 감당할 수 있는 양보다 더 많은 데이터를 전송함으로써, 이 웜은 호출 스택상의 리턴 어드레스를 덮어씌워 원격 코드 실행에 성공했습니다. 당시 인터넷에 연결된 컴퓨터의 약 10%가 감염되어 광범위한 혼란을 초래했으며, 이는 현대 보안 분야의 탄생을 의미했습니다.

30년이 지난 지금까지도 버퍼 오버플로 취약점은 가장 위험한 소프트웨어 버그 유형 중 하나로 남아 있습니다. Common Vulnerabilities and Exposures 데이터베이스에는 경계 밖 쓰기(out-of-bounds write), 사용 후 해제(use-after-free) 조건, 힙 오버플로 및 관련 메모리 손상 결함과 관련하여 매년 수천 개의 CVE 항목이 등록됩니다. 현대 운영 체제는 이러한 공격에 대응하기 위해 다양한 방어수단을 배치합니다. 비실행 메모리 페이지, 주소 공간 무작위화, 스택 카나리, 제어 흐름 무결성, 그리고 Intel의 제어 흐름 강제 기술(Control-Flow Enforcement Technology)과 ARM의 포인터 인증 코드(Pointer Authentication Codes)와 같은 하드웨어 기반 보호 기능이 그것입니다. 그러나 공격자는 각 새로운 방어 계층을 우회하는 방법을 계속 찾아내고 있으며, 익스플로잇 기법과 완화 조치 사이의 군비 경쟁은 둔화될 조짐이 없습니다.

이 책이 존재하는 이유는 소프트웨어 보안에 종사하는 모든 사람에게 메모리 손상을 근본적인 수준에서 이해하는 것이 필수적이기 때문입니다. 스택 오버플로, 힙 조작, 리턴 지향 프로그래밍 페이로드가 어떻게 연쇄적으로 연결될 수 있는지를 파악하지 못하는 방어 전문가들은 효과적인 보호 장치를 설계하기 어려울 것입니다. 취약점 연구자는 메모리 레이아웃, 할당자 내부 구조, 호출 규약에 대한 심층적인 지식을 바탕으로 새로운 버그를 발견하고 책임 있게 공개해야 합니다. 멀웨어를 분석하는 리버스 엔지니어는 익스플로이팅 원시 기법(primitives)을 이해하여 그 능력을 평가해야 합니다. 심지어 애플리케이션 개발자도 특정 코딩 패턴이 왜 위험한지, 컴파일러 경화 플래그가 내부적으로 어떻게 작동하는지를 아는 것에서 이익을 얻습니다.

이 책은 기초부터 고급 기법으로 진행하는 점진적인 여정으로 구성되어 있습니다. 초반 장들은 하드웨어와 운영 체제 맥락을 확립합니다. CPU가 메모리와 상호 작용하는 방식, 프로세스가 가상 주소 공간에 어떻게 배치되는지, 어셈블리 언어 명령어가 레지스터 수준에서 무엇을 하는지, 그리고 호출 규약이 플랫폼 간 함수 호출을 어떻게 규율하는지를 다룹니다. 이 기초가 없으면 이후의 익스플로이팅 논의는 근거를 잃게 됩니다.

중반 장들은 특정 취약점 클래스와 익스플로이팅 기법에 깊이 들어갑니다. 스택 기반 버퍼 오버플로는 근본적인 메모리 손상 버그로서 상세히 다루되며, 할당자 내부 구조 때문에 훨씬 더 복잡한 힙 익스플로이팅이 그 후에 이어집니다. 오프바이원(off-by-one) 오류, 사용 후 해제, 이중 해제, 형식 문자열 결함, 정수 오버플로 및 레이스 컨디션과 관련된 취약점 유형이 체계적으로 소개됩니다. 쉘코드 구성, 제어 흐름 하이재킹, 리턴 지향 프로그래밍, 점프 지향 프로그래밍, sigreturn 지향 프로그래밍과 같은 익스플로잇 개발 개념은 이러한 기초 위에 구축됩니다. ASLR을 무력화하는 정보 누출 및 주소 공개 기법은 별도의 집중적인 장에서 다룹니다.

후반 장들은 방어로 초점을 전환합니다. 현대적 완화 기법이 개별적으로 검토됩니다. DEP/NX가 하드웨어 수준에서 어떻게 작동하는지, ASLR이 어떻게 엔트로피를 제공하고 어디서 한계가 있는지, 스택 카나리가 손상이 확산되기 전에 손상을 어떻게 감지하는지, PIE와 RELRO가 코드 섹션을 어떻게 보호하는지, LLVM, Intel CET, Microsoft CFG, ARM PAC의 제어 흐름 무결성 구현이 제어 흐름 이전을 어떻게 제약하는지를 다룹니다. 각 완화 기법과 그 우회 기법 사이의 역사적 군비 경쟁을 다루는 전용 장도 포함되어 있습니다. 어떤 방어가 실패했는지를 이해하는 것이 그들이 무엇을 보호하는지를 이해하는 만큼 중요하기 때문입니다.

마지막 장들은 더 넓은 생태계를 다룹니다. 퍼징 전략과 크래시 트리아지를 포함한 취약점 연구 방법론, 안전한 코딩 관행과 컴파일러 경화 옵션, 그리고 운영 환경에서 활발한 익스플로이팅에 대응하는 방어자를 위한 익스플로잇 탐지, 사건 대응 및 포렌식 분석을 다룹니다.

책 전반에 걸쳐 C 언어의 코드 예제가 해당 어셈블리 목록과 함께 제공되어 취약점이 명령어 수준에서 어떻게 나타나는지를 보여줍니다. 메모리 다이어그램은 손상 전후의 데이터 레이아웃을 표시합니다. 디버거 출력은 실용적인 분석 워크플로우를 시연합니다. Linux와 Windows 간의 크로스 플랫폼 비교는 공유된 원칙과 플랫폼별 미묘한 차이 모두를 강조합니다. 학술 연구가 이해에 기여한 부분에서는 출판된 논문 및 학회지 참고문헌이 제공됩니다.

몇 가지 중요한 면책 조항을 명시해야 합니다. 이 책은 교육적·방어적 목적으로 익스플로이팅 기법을 기술합니다. 저자는 윤리적 보안 연구와 취약점에 대한 책임 있는 공개를 강력히 지지합니다. 자신이 소유하지 않은 소프트웨어에서 메모리 손상 버그를 발견한 독자는 확립된 책임 있는 공개 절차를 따라야 합니다. 벤더에게 비공개로 통지하고, 수정안이 개발될 때까지 합리적인 시간을 허용하며, 취약점이 패치된 후에만 세부 정보를 공개해야 합니다. 명시적인 권한 없이 시스템에 대해 익스플로이팅 기법을 사용하는 것은 대부분의 관할권에서 불법이며, 어떤 경우에도 윤리적이지 않습니다.

이 책은 포인터, 배열, 구조체 및 malloc/free를 사용한 동적 메모리 할당을 포함한 기본 C 프로그래밍에 대한 친숙함을 전제로 합니다. 사전의 어셈블리 언어 경험이나 익스플로잇 개발 배경은 필요하지 않습니다. 이러한 주제는 처음부터 구축됩니다. 초점은 지배적인 데스크톱 및 서버 플랫폼인 x86_64 아키텍처이며, 모바일 및 임베디드

컨텍스트와 관련 있는 경우 ARM64에 대해 논의합니다. Linux와 Windows가 예제의 주요 운영 체제로 사용되지만, 많은 개념이 광범위하게 적용됩니다.

이 책을 다 읽은 후에는, 독자는 메모리 손상을 고립된 트릭의 모음이 아니라 컴퓨터 아키텍처, 운영 체제 설계, 소프트웨어 공학에 뿌리를 둔 일관된 분야로 이해하게 될 것입니다. 그 이해는 효과적인 오펜시브 연구와 견고한 방어적 엔지니어링 모두를 구축하는 기반입니다.

제1장: 컴퓨터 메모리 아키텍처 및 CPU 기초

메모리 손상을 이해하려면 이를 가능하게 하는 하드웨어를 이해해야 합니다. 모든 버퍼 오버플로, 모든 사용 후 해제, 모든 형식 문자열 익스플로잇은 궁극적으로 프로그램이 메모리 레이아웃에 대해 가정하는 것과 CPU가 데이터를 실제로 처리하는 방식 사이의 불일치를 악용합니다. 이 장은 아키텍처적 기초를 확립합니다.

폰 노이만 아키텍처와 메모리 어드레싱

현대 컴퓨터는 1945년 존 폰 노이만이 제안한 폰 노이만 아키텍처를 따릅니다. 이 모델에서 명령어와 데이터는 동일한 메모리 공간을 공유하고 동일한 버스를 통해 접근됩니다 [9]. CPU는 메모리에서 명령어를 가져와 해석하고, 실행한 후 결과를 메모리에 다시 저장합니다. 이 통합된 메모리 모델은 컴퓨팅의 유연성의 근원이자 근본적인 보안 취약점의 원인입니다. 데이터가 CPU가 명령어를 기대하는 위치에 배치될 수 있다면, 공격자는 데이터를 손상시켜 임의의 코드를 실행할 수 있습니다.

메모리는 어드레싱 가능한 바이트의 선형 배열로 구성됩니다. 각 바이트는 0부터 시작하는 고유한 수치 어드레스를 가집니다. 32비트 시스템에서 어드레스 범위는 0x00000000부터 0xFFFFFFFF(4기가바이트)까지입니다. 64비트 시스템에서는 이론적 범위가 2^64 바이트(16엑사바이트)로 확장되지만, 실제 구현은 더 적은 어드레스 비트를 사용합니다. x86_64는 현재 48비트 가상 어드레스 비트를 사용하여 프로세스당 256테라바이트의 주소 공간을 제공합니다.

CPU는 모든 메모리 작업 동안 함께 작동하는 세 개의 상호 연결된 버스를 통해 메모리에 접근합니다. 어드레스 버스는 CPU가 읽거나 쓰려는 메모리 어드레스를 운반하며, 프로세서에서 메모리 서브시스템으로 단방향으로 흐릅니다. x86_64에서 어드레스 버스 폭은 CPU가 도달할 수 있는 고유한 물리적 어드레스 수를 결정합니다. 데이터 버스는 실제 전송되는 바이트를 운반하며 양방향으로 동작합니다. 읽기 작업 중에는 데이터가 메모리에서 CPU로 흐르고, 쓰기 작업 중에는 반대 방향으로 흐릅니다. 제어 버스는 타이밍 및 제어 신호를 운반하여 전송을 조정합니다. 현재 작업이 읽기인지 쓰기인지, 메모리 서브시스템이 데이터를 받을 준비가 되었는지, 전송이 완료되었는지를 나타냅니다.

버퍼 오버플로 익스플로잇 도중 이러한 버스는 공격 전반에 걸쳐 활동적입니다. 오버플로우가 발생하는 `strcpy`가 버퍼 경계를 넘어 쓸 때, 각 바이트는 스택상의 대상 어드레스로 데이터 버스를 통해 이동합니다. 복사 작업이 버퍼를 지나 저장된 프레임 포인터와 리턴 어드레스로 진행됨에 따라 어드레스 버스는 점점 더 높은 스택 어드레스를 전달합니다. 오버플로 데이터를 제어하는 공격자는 궁극적으로 이러한 버스를 통해 이동하는 값을 제어하며, 그 연장선상에서 CPU가 RET 명령어를 실행할 때 읽을 값도 제어합니다. 이 하드웨어 수준의 데이터 흐름을 이해하면 메모리 손상이 왜 직접적으로 제어 흐름 하이재킹으로 이어지는지 명확해집니다. CPU는 합법적인 스택 데이터와 유효한 어드레스처럼 보이는 공격자가 제공한 바이트를 구별할 방법이 없기 때문입니다.

x86_64의 메모리 어드레싱은 여러 모드를 지원합니다. 즉시 어드레싱(immediate addressing)은 상수 값을 명령어에 직접 임베딩합니다. 레지스터 어드레싱은 레지스터의 내용을 피연산자로 사용합니다. 직접 메모리 어드레싱은 절대 어드레스를 지정합니다. 가장 일반적으로, 베이스-플러스-변위 어드레싱(base-plus-displacement addressing)은 베이스 레지스터의 값에 변위를 더하여 유효 어드레스를 계산하며, 선택적으로 인덱스 레지스터와 1, 2, 4, 또는 8의 스케일 팩터로 스케일링됩니다. 이 마지막 모드는 배열이 접근되는 방식입니다. 베이스 레지스터가 배열의 시작 어드레스를 보유하고, 인덱스 레지스터가 요소 번호를 보유하며, 스케일 팩터가 요소 크기를 고려합니다.

CPU 레지스터와 그 역할

레지스터는 CPU 내부의 작고 빠른 저장 위치입니다. 프로세서 다이 위에 위치하기 때문에 메인 메모리보다 여러 단계 빠릅니다. 어셈블리 언어의 모든 명령어는 레지스터나 레지스터를 통해 어드레싱된 메모리에서 작동합니다. 익스플로잇 개발에서는 각 레지스터가 무엇을 하는지, 호출 규약이 그것을 어떻게 사용하는지를 이해하는 것이 중요합니다.

x86_64 아키텍처는 열여섯 개의 64비트 일반 목적 레지스터를 제공합니다 [13, 91]:

RAX (누산기/Accumulator): 전통적으로 산술 연산 및 함수 리턴 값에 사용됩니다. 시스템 콜 인터페이스에서 RAX는 시스템 호출 번호를 보관합니다. 이름은 AX(16비트)에서 유래했으며, 이는 다시 AL과 AH(8비트 하위 및 상위 절반)에서 유래했습니다. 32비트 EAX에 쓰면 RAX의 상위 32비트가 0으로 초기화되는 특성은 쉘코드에서 널-프리 페이로드를 생성할 때 활용됩니다.

RBX (베이스/Base): 데이터 접근을 위한 베이스 포인터로 자주 사용됩니다. System V AMD64 ABI에서는 RBX가 callee-saved 레지스터입니다. 이를 수정하는 함수는 리턴 전에 원래 값을 복원해야 합니다.

RCX (카운터/Counter): 문자열 및 반복 명령어에서 루프 카운터로 역사적으로 사용되었습니다. Windows x64 호출 규약에서 RCX는 첫 번째 함수 인수를 보관합니다.

RDX (데이터/Data): 입출력 작업에 사용되며, 곱셈 및 나눗셈 결과의 보조 누산기로도 사용됩니다(RDX:RAX에 걸쳐 저장되는 128비트 결과 생성).

RSI (소스 인덱스/Source Index)와 RDI (데스티네이션 인덱스/Destination Index): 문자열 작업에서 소스와 데스티네이션 어드레스를 지정하는 데 사용됩니다. System V AMD64 ABI에서는 RSI와 RDI가 각각 첫 번째와 두 번째 함수 인수를 보관합니다.

RSP (스택 포인터/Stack Pointer): 현재 스택 프레임의 상단(top)을 가리킵니다. x86에서 스택은 하향으로 성장하므로 값을 푸시하면 RSP가 감소합니다. 이 레지스터는 익스플로이팅의 핵심입니다. RSP를 제어한다는 것은 CPU가 리턴 명령어를 실행할 때 읽을 데이터를 제어한다는 것을 의미합니다.

RBP (베이스 포인터 / 프레임 포인터/Base Pointer / Frame Pointer): 스택 프레임 내에서 안정적인 참조점으로 전통적으로 사용됩니다. 컴파일러가 `-fomit-frame-pointer` 플래그로 이를 최적화하여 제거할 수 있지만, 대부분의 디버그 빌드와 많은 프로덕션 빌드에서는 여전히 RBP를 사용하여 스택 프레임의 연결 체인을 생성하고 백트레이스 기능을 가능하게 합니다.

R8부터 R15: 64비트 확장에서 추가된 여덟 개의 일반 목적 레지스터입니다. System V ABI에서 R8과 R9는 다섯 번째와 여섯 번째 함수 인수를 보관합니다. Windows x64에서는 R8과 R9가 세 번째와 네 번째 인수를 보관합니다.

일반 목적 레지스터 외에도 여러 특수 목적 레지스터가 관련됩니다:

RIP (명령어 포인터/Instruction Pointer): 실행할 다음 명령어의 어드레스를 보관합니다. 제어 흐름 하이재킹 공격의 주요 대상입니다. 다른 레지스터와 달리 RIP는 직접 쓸 수 없으며, call, jump, return, interrupt 명령어에 의해서만 수정됩니다. RIP를 제어한다는 것은 프로그램 실행을 제어한다는 것을 의미합니다.

RFLAGS (플래그 레지스터/Flags Register): 산술 및 논리 연산에 의해 설정되는 조건 코드를 포함합니다. 0 결과를 나타내는 Zero Flag(ZF), 레지스터 너비를 초과하는 오버플로를 나타내는 Carry Flag(CF), 음수 결과를 나타내는 Sign Flag(SF), 부호 있는 오버플로를 나타내는 Overflow Flag(OF), 문자열 작업 방향을 제어하는 Direction Flag(DF). 조건부 점프 명령어는 분기할지 여부를 결정하기 위해 이 플래그를 테스트합니다.

명령어 실행과 Fetch-Decode-Execute 사이클

CPU는 반복적인 사이클로 명령어를 실행합니다:

Fetch (가져오기): CPU는 RIP에 저장된 어드레스의 메모리에서 다음 명령어를 읽습니다. x86_64에서 명령어는 가변 길이(1~15바이트)이므로, CPU는 다음 명령어로 RIP를 진행하기 전에 첫 번째 바이트를 디코딩하여 명령어 길이를 결정해야 합니다.

Decode (**디코딩**): 가져온 바이트가 실행 유닛이 이해하는 내부 마이크로 연산으로 변환됩니다. 현대의 오더아웃 프로세서는 사이클당 여러 명령어를 디코딩하고 병렬 실행을 위해 재배열할 수 있습니다.

Execute (**실행**): 디코딩된 연산이 수행됩니다. 레지스터 또는 메모리에서 피연산자를 읽거나, ALU에서 산술을 수행하거나, 값을 비교하거나, 플래그를 업데이트할 수 있습니다.

Write-back (**결과 쓰기**): 결과가 데스티네이션 레지스터 또는 메모리에 다시 쓰입니다.

Commit (**커밋**): 오더아웃 프로세서에서 결과는 프로그램 순서대로 아키텍처 상태로 커밋되어, 명령어가 순서대로 실행되지 않았더라도 관찰 가능한 동작이 시퀀셜 실행과 일치하도록 보장합니다.

익스플로이팅 관점에서 핵심 통찰은 CPU가 RIP가 가리키는 어드레스를 신뢰한다는 것입니다. 하드웨어 수준에서 "합법적인" 코드와 "악의적인" 코드를 구별하는 고유한 메커니즘은 존재하지 않습니다. 버퍼 오버플로가 스택상의 리턴 어드레스를 덮어쓰고, 덮어씌워진 값이 유효한 명령어 바이트를 포함하는 공격자 제어 데이터로□□□한다면, CPU는 그 바이트를 의심 없이 실행합니다. 이는 모든 코드 실행 익스플로잇의 근본 전제입니다.

다음 x86_64 명령어들은 익스플로잇 개발에 특히 관련성이 있습니다:

- MOV: 레지스터, 메모리, 즉시 값 사이에서 데이터를 복사합니다.
- PUSH / POP: 값을 스택에 푸시(RSP 감소)하거나 스택에서 팝(RSP 증가).
- CALL: 현재 RIP를 스택에 푸시하고 대상 어드레스로 점프합니다. 함수 호출에 사용됩니다.
- RET: 스택 상단의 값을 RIP로 팝합니다. 리턴 지향 프로그래밍 가젯이 연결하는 명령어입니다.
- JMP: 대상 어드레스로의 무조건 점프입니다. 점프 지향 프로그래밍에 사용됩니다.
- XOR / AND / OR: 비트 논리 연산으로, 쉘코드에서 널 바이트 없는 값 초기화에 자주 사용됩니다.
- LEA (Load Effective Address): 실제로 메모리에 접근하지 않고 어드레스를 계산합니다. 산술 및 위치 독립 코드에서 현재 RIP를 얻는 데 유용합니다.
- SYSCALL / INT 0x80: 사용자 모드에서 커널 모드로 전환하여 시스템 호출을 수행합니다.

엔디안과 데이터 표현

엔디안은 다중 바이트 값이 메모리에 저장되는 바이트 순서를 결정합니다. x86_64는 리틀 엔디안(little-endian) 순서를 사용합니다. 가장 낮은 значимости 바이트가 가장 낮은 어드레스에 저장됩니다. 예를 들어, 64비트 값 0x4142434445464748은 다음과 같이 저장됩니다:

어드레스	바이트
0x1000	0x48 ('H')
0x1001	0x47 ('G')
0x1002	0x46 ('F')
0x1003	0x45 ('E')
0x1004	0x44 ('D')
0x1005	0x43 ('C')
0x1006	0x42 ('B')
0x1007	0x41 ('A')

버퍼 오버플로를 익스플로잇할 때 엔디안은 공격자가 올바른 바이트 순서로 다중 바이트 값(리턴 어드레스 등)을 제공해야 하므로 중요합니다. 리틀 엔디안 시스템은 인쇄될 때 나타나는 순서와 비교하여 역순으로 어드레스 바이트를 필요로 합니다.

이는 부분적 덮어쓰기도 영향을 받습니다. 공격자가 64비트 포인터의 가장 낮은 значимости 바이트만 덮어쓸 수 있다면, 리틀 엔디안 시스템에서 그 바이트는 가장 낮은 어드레스에 있으며 값의 하위 8비트를 나타냅니다. 빅 엔디안 시스템에서는 동일한 바이트가 상위 8비트를 나타낼 것입니다.

권한 레벨과 링 아키텍처

x86 프로세서는 네 가지 권한 레벨을 구현하며, 이를 링 0부터 링 3이라고 부릅니다. 링 0이 가장 높은 권한을 가지고 링 3이 가장 낮은 권한을 가집니다. 현대 운영 체제는 일반적으로 두 개만 사용합니다:

- **링 0 (커널 모드):** 운영 체제 커널이 여기서 실행됩니다. 모든 하드웨어, 메모리, CPU 기능에 무제한 접근 권한을 가집니다. 커널 모드 코드는 임의의 물리적 메모리 어드레스를 읽고 쓸 수 있으며, 페이지 테이블 엔트리를 수정하고, CLI(인터럽트 플래그 클리어)와 LGDT(글로벌 디스크립터 테이블 로드)와 같은 특권 명령어를 실행할 수 있습니다.

- **링 3 (사용자 모드)**: 애플리케이션 프로그램이 여기서 실행됩니다. 사용자 모드 코드는 하드어나 커널 메모리에 직접 접근할 수 없습니다. 특권 명령어 실행 또는 커널 어드레스 접근 시도는 일반 보호 오류(general protection fault)를 발생시키며, 커널은 위반 프로세스를 종료하여 이를 처리합니다.

링 간 전환은 하드웨어에 의해 제어됩니다. 시스템 호출은 SYSCALL 명령어(x86_64) 또는 INT 0x80(레거시 x86)를 사용하여 링 3에서 링 0으로 전환합니다. 인터럽트와 예외도 링 전환을 유발합니다. CPU는 커널 핸들러로 제어를 이전하기 전에 현재 상태(RIP 및 RFLAGS 포함)를 커널 스택에 저장합니다.

이 권한 분리는 익스플로이팅과 관련이 있습니다. 사용자 모드 익스플로잇은 링 3에서 수행할 수 있는 작업으로 제한되기 때문입니다. 사용자 애플리케이션에서 성공적인 버퍼 오버플로는 해당 프로세스의 제어 흐름을 하이재킹할 수 있지만 다른 프로세스의 메모리나 커널 데이터에 직접 접근할 수는 없습니다. 권한을 상승하려면 공격자가 커널 자체의 취약점(커널 모드 버퍼 오버플로)을 악용하거나 시스템 호출 인터페이스를 남용하는 방법을 찾아야 합니다.

하드웨어 수준의 메모리 매핑

메모리 관리 유닛(MMU)은 소프트웨어가 사용하는 가상 어드레스를 RAM이 사용하는 물리적 어드레스로 변환하는 하드웨어 구성 요소입니다 [11]. MMU는 운영 체제가 유지하는 계층적 데이터 구조인 페이지 테이블을 사용하여 이 변환을 수행합니다 [10, 14].

x86_64의 4레벨 페이징에서 48비트 가상 어드레스는 다음과 같이 분할됩니다:

- 비트 47:39 (9비트): Page Map Level 4 (PML4) 테이블 인덱스
- 비트 38:30 (9비트): Page Directory Pointer Table (PDPT) 인덱스
- 비트 29:21 (9비트): Page Directory (PD) 인덱스
- 비트 20:12 (9비트): Page Table (PT) 인덱스
- 비트 11:0 (12비트): 4-KiB 페이지 내 오프셋

각 페이지 테이블 레벨은 512개의 엔트리(2^9)를 포함하며, 각 엔트리는 8바이트입니다. CPU는 CR3 레지스터에 저장된 루트 포인터에서 시작하여 이 테이블들을 순회해서 가상 어드레스에 해당하는 물리적 프레임을 찾습니다.

각 페이지 테이블 엔트리(PTE)에는 물리적 프레임 번호뿐만 아니라 권한 플래그도 포함됩니다:

- **Present (P)**: 페이지가 현재 물리적 메모리에 있는지 여부

- Read/Write (R/W): 쓰기가 허용되는지 여부
- User/Supervisor (U/S): 사용자 모드 코드가 페이지에 접근할 수 있는지 여부
- Execute Disable (NX/XD): 이 페이지에서의 코드 실행이 허용되는지 여부

AMD의 확장 페이지 테이블과 Intel의 Execute Disable 기능에서 x86_64에 추가된 NX 비트는 DEP의 하드웨어 기반입니다 [5, 6]. 페이지 테이블 엔트리에 NX 비트가 설정되면, 해당 페이지에서 명령어 실행 시도가 보호 오류를 발생시킵니다. 이는 스택이나 힙에 배치된 쉘코드가 직접 실행되는 것을 방지합니다.

번역 탐색 버퍼(TLB)는 최근의 가상-물리 번역을 캐싱하여 모든 메모리 접근마다 다중 레벨 페이지 테이블 워크를 피합니다. TLB는 일반적으로 작습니다(현대 프로세서에서 64~128 엔트리). 따라서 프로세스 매핑의 일부만 보유합니다. TLB 미스 시 CPU는 하드웨어 페이지 워크를 수행하며, 이는 수십 사이클이 소요됩니다.

페이지 레벨 권한을 이해하는 것은 익스플로이팅에 필수적입니다. 많은 기법이 특정 권한 조합을 가진 메모리 영역을 찾는 데 의존하기 때문입니다. 리턴 지향 프로그래밍은 실행 가능한 코드 페이지(텍스트 세그먼트)가 이미 실행 가능으로 표시되어 있기 때문에 작동합니다. 공격자는 새 코드를 쓰기 전용이지만 실행 불가능한 영역에 삽입하는 대신 기존 코드를 재사용합니다.

제2장: 프로세스 메모리 레이아웃 및 운영 체제 메모리 관리

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

가상 메모리 및 주소 변환

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

텍스트, 데이터, BSS, 힙 및 스택 세그먼트

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

공유 라이브러리와 동적 링크

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

메모리 할당: malloc, free 및 할당자

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

페이지 테이블과 MMU

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

크로스 플랫폼 차이: Linux와 Windows 메모리 레이아웃

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제3장: 어셈블리 언어 필수 개념 및 호출 규약

어셈블리 언어는 머신 코드의 사람이 읽을 수 있는 표현입니다. 현대 익스플로잇 개발에서□□부터 어셈블리를 작성하는 경우는 드물지만, 디어셈블된 바이너리를 읽고, 크래시 덤프를 분석하며, 메모리 손상이 어떻게 제어 흐름 하이재킹으로 이어지는지 추론하는 데에는 어셈블리 이해가 필수적입니다. 이 장은 익스플로잇 컨텍스트에서 사용되는 x86_64 명령어 세트를 다룹니다.

x86 및 x86_64 명령어 세트 개요

x86_64는 가변 길이 명령어를 가진 CISC(Complex Instruction Set Computing) 아키텍처입니다. 각 명령어는 선택적 접두사 바이트, 1바이트 오퍼코드, 선택적 ModR/M 바이트, 선택적 SIB(Scale-Index-Base) 바이트, 선택적 변위, 그리고 선택적 즉시 데이터로 구성됩니다. 이 복잡성은 강력한 단일 명령어를 가능하게 하지만, RISC 아키텍처보다 디어셈블리 및 분석을 더 어렵게 만듭니다.

명령어는 여러 범주로 분류됩니다:

데이터 이동: MOV, PUSH, POP, LEA, XCHG **산술:** ADD, SUB, MUL, IMUL, DIV, INC, DEC, NEG **논리:** AND, OR, XOR, NOT, SHL, SHR, ROL, ROR **제어 흐름:** JMP, JE/JZ, JNE/JNZ, JL, JG, CALL, RET, INT **문자열 작업:** MOVS, CMPS, SCAS, REP 접두사 **시스템 호출:** SYSCALL, SYSRET, INT 0x80

익스플로잇 개발에서 가장 중요한 명령어는 스택을 조작하는(PUSH, POP), 실행 흐름을 제어하는(CALL, RET, JMP), 그리고 레지스터와 메모리 간에 데이터를 이동하는(MOV, LEA) 명령어입니다.

익스플로잇 개발용 공통 명령어

MOV: 핵심 명령어입니다. 소스에서 데스티네이션으로 데이터를 복사합니다. 최소한 하나의 피연산자는 레지스터 또는 메모리 위치여야 합니다(두 즉시 값은 불가). MOVL(32비트), MOVQ(64비트), MOVABS(레지스터에 64비트 즉시 값 로드) [91, 92] 등의 변형이 있습니다.

```
mov rax, 0x41414141          ; 즉시 값을 레지스터로 이동
mov rax, rbx                 ; 레지스터에서 레지스터로
mov [rsp], rax               ; 레지스터에서 메모리(스택)로
mov rax, [rbp-8]             ; 메모리에서 레지스터로
```

PUSH와 POP: 스택을 조작합니다. PUSH는 RSP를 감소시키고 값을 저장하며, POP은 값을 로드하고 RSP를 증가시킵니다. 이는 ROP 체인 구성의 핵심입니다. 가젯이 종종 POP을 사용하여 공격자 제어 스택 데이터에서 값을 레지스터로 로드하기 때문입니다.

```
push rax                     ; 스택에 8바이트 푸시
pop rdi                      ; 스택에서 8바이트 팝하여 RDI로
ret                          ; 스택에서 8바이트 팝하여 RIP로
```

LEA (Load Effective Address): 참조 해체(dereference) 없이 어드레스를 계산합니다. 메모리 피연산자를 가진 MOV와 달리, LEA는 메모리에 접근하지 않습니다. 어드레스 계산을 수행하고 결과를 저장합니다. 이는 산술(스케일링된 덧셈) 및 위치 독립 코드에서 현재 명령어 포인터를 얻는 데 유용합니다.

```
lea rax, [rbp-8]             ; RAX = RBP - 8 (메모리 접근 없음)
lea rax, [rip]               ; RAX = 현재 RIP (PIC 기법)
lea rax, [rcx*4+rdx+0x10]    ; RAX = RCX*4 + RDX + 0x10
```

RET: 스택 상단을 RIP로 팝합니다. 리턴 지향 프로그래밍을 가능하게 하는 명령어입니다. 함수가 리턴할 때, 에필로그는 RET를 실행하여 스택 상단에 있는 값을 읽고 그곳으로 실행을 전환합니다. 오버플로가 그 값을 덮어썼다면, 실행이 공격자가 선택한 위치로 점프합니다.

XOR: 배타적 OR입니다. 레지스터를 자체와 XOR하면 효율적으로 0이 됩니다(0과 MOV보다 더 짧은 인코딩 생성). 즉시 값이 필요하지 않기 때문에 널-프리 쉘코드에 중요합니다.

```
xor rax, rax                 ; RAX = 0 (널-프리 방식으로 0 설정)
xor byte [rax], al           ; RAX 어드레스에 널 바이트 쓰기
```

스택 프레임: 프롤로그, 에필로그 및 로컬 변수

함수가 호출되면 로컬 상태를 보유하기 위해 스택 프레임이 생성됩니다. 프레임을 낮은 어드레스에서 높은 어드레스 방향으로 보면 다음과 같은 일반적인 레이아웃을 가집니다:

```
높은 어드레스
+------------------+
|  인수            |  (호출자에 의해 푸시됨, 저장된 리턴 어드레스 아래)
+------------------+
|  리턴 어드레스   |  <--- RET가 점프할 위치
+------------------+
|  저장된 RBP      |  <--- 프롤로그 이후 RBP가 가리키는 위치
+------------------+
|                  |
|  로컬 변수       |  ([rbp - offset]으로 접근)
|                  |
+------------------+ <-- RSP가 여기(또는 그 아래)를 가리킴
낮은 어드레스
```

함수 프롤로그는 이 프레임을 설정합니다:

```
push rbp                    ; 호출자의 프레임 포인터 저장
mov rbp, rsp                ; 새 프레임 포인터를 현재 스택 상단으로 설정
sub rsp, 0x20               ; 로컬 변수용 공간 할당
```

함수 에필로그는 이를 해제합니다:

```
leave                       ; mov rsp, rbp; pop rbp와 동일
ret                         ; 리턴 어드레스를 RIP로 팝
```

로컬 변수는 RBP로부터의 음수 오프셋을 사용하여 접근됩니다. `char buf[64]`로 선언된 변수는 일반적으로 `[rbp-0x40]`에 위치합니다. 첫 여섯 개의 인수(레지스터에 들어감) 이후의 함수 인수는 호출자가 CALL 명령어 전에 스택에 푸시하며, RBP로부터의 양수 오프셋에서 접근할 수 있습니다.

스택 프레임은 스택 기반 버퍼 오버플로의 주요 표적입니다. `buf`가 경계 검사 없이 쓰여지면, 64바이트 이상을 작성하면 높은 어드레스의 데이터가 덮어씌워집니다. 먼저

다른 로컬 변수, 그 다음 저장된 RBP, 마지막으로 리턴 어드레스 순입니다. 리턴 어드레스가 덮어씌워지면 RET가 제어를 공격자 제어 위치로 이전합니다.

System V AMD64 ABI 호출 규약

System V AMD64 Application Binary Interface는 x86_64에서 실행되는 유닉스 계열 시스템(Linux, macOS, FreeBSD, Solaris)에서 함수가 호출되는 방식을 정의합니다 [1, 12]. 핵심 규칙:

인수 전달: 첫 여섯 개의 정수 또는 포인터 인수는 RDI, RSI, RDX, RCX, R8, R9 순서로 레지스터를 통해 전달됩니다. 여섯 번째 이후의 인수는 오른쪽에서 왼쪽으로 스택에 푸시됩니다. 부동소수점 인수는 XMM0부터 XMM7을 사용합니다.

리턴 값: 64비트 이하의 정수 리턴 값은 RAX에서 반환됩니다. 128비트 값은 RDX:RAX(상위:하위)를 사용합니다. 부동소수점 리턴은 XMM0을 사용합니다. 16바이트보다 큰 구조체는 숨겨진 포인터 인수를 통해 반환됩니다.

Callee-saved 레지스터: RBX, RBP, R12, R13, R14, R15는 함수 호출을 가로질러 보존되어야 합니다. 함수가 이를 수정하면 저장 및 복원해야 합니다. 나머지 모든 레지스터(RAX, RCX, RDI, RSI, RDX, R8, R9, R10, R11)는 caller-saved이며 callee가 자유롭게 수정할 수 있습니다.

스택 정렬: CALL 명령어 전에 스택은 16바이트로 정렬되어야 합니다. 이는 호출이 리턴 어드레스를 푸시한 후(스택을 8바이트만큼 미정렬 상태로 만듦), 호출된 함수의 프롤로그가 RSP를 조정하여 정렬을 복원해야 한다는 것을 의미합니다. 일반적으로 RSP에서 8의 홀수 배수를 빼서 수행됩니다.

레드 존(Red zone): System V ABI는 RSP 아래 128바이트의 "레드 존"을 정의합니다. 리프 함수(다른 함수를 호출하지 않는 함수)는 RSP를 조정하지 않고 임시 저장소로 사용할 수 있습니다. 시그널 핸들러와 비동기 이벤트는 이 영역을 손상시켜서는 안 됩니다. 익스플로잇 개발자는 레드 존이 스택에서 데이터를 신뢰할 수 있게 배치할 수 있는 위치에 영향을 미치므로 인지해야 합니다.

Windows x64 호출 규약

Windows는 x64에서 다른 호출 규약을 사용하며, System V와 여러 중요한 측면에서 다릅니다 [16]:

인수 전달: 첫 네 개의 정수 또는 포인터 인수는 RCX, RDX, R8, R9 순서로 전달됩니다. 네 번째 이후의 인수는 스택에 배치됩니다. 부동소수점 인수는 XMM0부터 XMM3를 사용합니다.

섀도 공간(Shadow space): 호출자는 callee가 실제로 사용하는 인수 개수와 상관없이 모든 호출 전에 스택에 32바이트의 "섀도 공간"(또는 "홈 공간")을 할당해야 합니다. 이 공간은 [RSP+0], [RSP+8], [RSP+16], [RSP+24]에 있습니다. callee는 스프일(spill) 목적으로 레지스터 인수를 복사하는 데 이 공간을 사용할 수 있습니다.

Callee-saved 레지스터: RBX, RDI, RSI, RBP, R12, R13, R14, R15가 callee-saved입니다. Windows에서는 RDI와 RSI가 callee-saved이지만 Linux에서는 caller-saved라는 점을 주목하세요. 이 차이는 쉘코드 및 ROP 체인에서 안전히 사용할 수 있는 레지스터에 영향을 미칩니다.

스택 정렬: System V와 동일하게 CALL 명령어 전에 16바이트 정렬이 필요합니다.

리턴 값: System V와 동일합니다. 정수는 RAX, 부동소수점은 XMM0.

이 차이들로 인해 Linux에서 개발된 익스플로잇 기법이 Windows에 직접 적용되지 않으며, 그 반대도 마찬가지입니다. 특히 ROP 가젯은 어떤 레지스터가 사용 가능한지와 스택이 어떻게 구성되는지에 크게 의존합니다.

위치 독립 코드 및 리로케이션

위치 독립 코드(PIC, Position-independent code)는 메모리에 어디에 로드되더라도 올바르게 실행될 수 있는 코드입니다. 공유 라이브러리는 임의의 어드레스에 로드될 수 있으므로 위치 독립이어야 합니다. PIE와 같은 현대 익스플로잇 완화 기법은 이 요구사항을 전체 실행 파일로 확장합니다.

PIC에서는 코드가 절대 어드레스를 사용할 수 없습니다. 라이브러리 또는 실행 파일이 다른 베이스에 로드되면 해당 어드레스가 유효하지 않기 때문입니다. 대신 상대 어드레싱을 사용합니다:

RIP-상대 어드레싱: x86_64에서 메모리 피연산자는 현재 명령어 포인터에 상대적인 어드레스를 지정할 수 있습니다. 유효 어드레스는 RIP + 변위로 계산되며, 변위는 명령어에 인코딩됩니다. 이는 절대 베이스 어드레스를 알지 않고도 동일한 모듈 내의 데이터에 접근할 수 있게 합니다.

```
1  mov rax, [rip+0x1234]      ; 현재 명령어 뒤 0x1234바이트 위치의 데이터 접근
2  lea rdi, [rip+0x5000]      ; printf용 문자열 어드레스를 RDI로 로드
```

PC-상대 점프 및 호출: CALL과 JMP 명령어는 상대적 타겟을 사용할 수 있습니다. 변위는 현재 RIP에 추가되어 데스티네이션이 계산됩니다. 이는 단일 모듈 내의 코드가 절대 어드레스 없이 자체 함수를 호출할 수 있음을 의미합니다.

현재 RIP 얻기: 현재 실행 어드레스를 찾는 일반적인 PIC 기법은 다음과 같습니다:

```
call get_pc
get_pc: pop rax             ; RAX는 이제 리턴 어드레스(현재 위치)를 포함
```

이는 CALL이 다음 명령어의 어드레스를 스택에 푸시하고 POP이 이를 가져오기 때문에 작동합니다. 이 기법은 쉘코드에서 위치 독립성을 달성하는 데 사용됩니다.

리로케이션(Relocations): 공유 라이브러리가 로드될 때, 동적 링커는 라이브러리 외부의 심볼을 참조하는 어드레스를 수정해야 합니다. 이는 바이너리에 리로케이션 엔트리로 기록됩니다. GOT와 PLT 메커니즘(제2장에서 논의)이 런타임에 이 리로케이션을 처리합니다. 리로케이션을 이해하는 것은 익스플로이팅에 중요합니다. GOT 엔트리는 실행 가능한 코드를 가리키는 쓰기 가능 포인터이므로 덮어쓰기 공격의 매력적인 대상이기 때문입니다.

제4장: 디버깅 워크플로우 및 바이너리 분석 기초

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

보안 연구를 위한 GDB 필수 개념

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Windows의 WinDbg, x64dbg 및 Mona.py

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

메모리 검사 및 브레이크포인트 전략

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

radare2 및 Ghidra를 사용한 리버스 엔지니어링

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

정적 분석: objdump, readelf 및 PE 도구

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Frida를 사용한 동적 분석 및 계측

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제5장: 스택 기반 버퍼 오버플로

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

스택 기반 오버플로의 해부학

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

리턴 어드레스 덮어쓰기 및 제어 흐름 하이재킹

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

오프셋 찾기: 패턴 생성 및 크래시 분석

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

NOP 슬레드와 쉘코드 배치

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

환경 변수와 스택 스프레이

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Windows 및 Linux 시스템에 미치는 영향

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

워크스루: 쉘코드 삽입을 통한 완전한 스택 오버플로 익스플로잇

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

취약한 프로그램

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 1: 보호 비활성화로 컴파일

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 2: GDB 및 사이클릭 패턴으로 오프셋 찾기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 3: 쉘코드용 스택 어드레스 찾기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 4: 쉘코드 준비

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 5: Python 익스플로잇 스크립트

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 6: GDB에서 검증

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

핵심 관찰 사항

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제6장: 힙 기반 버퍼 오버플로 및 힙 조작

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

메모리 할당자의 작동 방식: glibc ptmalloc 및 Windows 힙

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

힙 청크 메타데이터 및 프리 리스트 관리

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

힙 오버플로 익스플로이팅 기법

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

빈 분류 및 병합 공격

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

House of Force, House of Spirit 및 명명된 힙 기법

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

워크스루: 임의 쓰기 원시 기법을 통한 Fastbin 공격

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

취약한 프로그램

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

익스플로잇 전략: __free_hook 덮어쓰기를 위한 Fastbin 공격

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

GDB로 필요한 어드레스 찾기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Python 익스플로잇 스크립트 (pwntools)

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

GDB로 익스플로잇 검증

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제7장: 관련 메모리 손상 취약점

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

오프바이원 오류와 그 익스플로이팅

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

메모리 손상으로 이어지는 정수 오버플로

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

사용 후 해제(UAF) 취약점

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

이중 해제 및 해제된 포인터 조작

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

형식 문자열 취약점

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

경계 밖 읽기 및 쓰기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

메모리 안전에 영향을 미치는 레이스 컨디션

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

워크스루: vtable 덮어쓰기를 통한 Tcache 독성 UAF

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

취약한 프로그램

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

익스플로잇 전략: vtable 덮어쓰기를 위한 Tcache 독성

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

GDB로 필요한 어드레스 찾기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Python 익스플로잇 스크립트 (pwntools)

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

GDB에서 검증

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

워크스루: GOT 덮어쓰기를 통한 형식 문자열 임의 쓰기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

취약한 프로그램

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

익스플로잇 전략: libc 누출, GOT 덮어쓰기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

필요한 어드레스 찾기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Python 익스플로잇 스크립트 (pwntools)

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

GDB에서 검증

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제8장: 쉘코드 기초 및 제어 흐름 하이재킹

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

쉘코드란 무엇이며 왜 중요한가

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

위치 독립 쉘코드 작성

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Linux의 시스템 호출 기반 쉘코드

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Windows API 해결 및 LoadLibrary 기법

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

널 바이트 처리 및 인코딩

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

다형성 및 인코딩된 페이로드

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

워크스루: 스택 오버플로에서 쉘코드 빌드 및 실행

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

취약한 프로그램

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

쉘코드: execve("/bin/sh")

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

GDB로 오프셋 찾기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Python 익스플로잇 스크립트

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

GDB에서 검증

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제9장: 리턴 지향 프로그래밍 및 코드 재사용 공격

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

DEP가 해결한 문제와 ROP 솔루션

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

ROP 체인 구성: 가젯, 피벗 및 레지스터

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

ropper 및 ROPgadget으로 가젯 찾기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Linux의 ROP: 시스템 호출 구성

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Windows의 ROP: VirtualAlloc 및 WinExec

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

워크스루: 경화된 바이너리에 대한 완전한 ROP 체인

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

취약한 프로그램 (경화됨)

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 1: 보호 확인 및 가젯 찾기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 2: libc 어드레스 찾기

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 3: ROP 체인 구성

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 4: Python 익스플로잇 스크립트

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 5: GDB에서 검증

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

이 워크스루의 핵심 교훈

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

점프 지향 프로그래밍 (JOP)

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

신호 지향 프로그래밍 (SROP)

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

지향 리턴 구성 (ORC)

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제10장: 정보 누출 및 주소 공개

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

ASLR이 익스플로이팅을 더 어렵게 만드는 이유

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

힙 스프레이 및 결정론적 메모리 레이아웃

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

형식 문자열 정보 공개

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Return-to-PLT 및 GOT 덮어쓰기를 통한 누출

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

워크스루: ASLR 우회를 통한 다단계 익스플로잇 체인

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

취약한 프로그램 (ASLR + Partial RELRO)

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

2단계 공격 이해

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Python 익스플로잇 스크립트

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

오프셋 수학 이해

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

GDB에서 검증

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

왜 이 2단계 접근 방식이 작동하는지

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Windows ASLR 우회 기법

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

부분적 덮어쓰기와 감소된 엔트로피

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

크로스 프로세스 정보 수집

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제11장: 현대적 방어 및 완화 기법

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

DEP/NX: 비실행 메모리 페이지

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

ASLR: 주소 공간 레이아웃 무작위화

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

스택 카나리 및 ProPolice

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

PIE, RELRO 및 GOT 보호

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

SafeSEH 및 Windows의 CFG

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제어 흐름 무결성 (CFI)

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Intel CET: 섀도 스택 및 간접 브랜치 추적

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

ARM 포인터 인증 (PAC)

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

샌드박싱 및 프로세스 격리

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제12장: 우회 연구 및 익스플로잇-완화 군비 경쟁

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

타임라인: 스택 스매싱에서 현대 방어까지

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

카나리 우회 기법

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

ASLR 엔트로피 분석 및 우회

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

Full RELRO 및 GOT 쓰기 한계

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

CFI 우회 연구

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

CET 우회 접근 방식

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

ARM의 PAC 우회

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

미래: 다음은 무엇인가

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제13장: 취약점 연구 방법론

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

퍼징 전략: 커버리지 기반 및 뮤테이션 기반

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

워크스루: 퍼징 캠페인 설정 및 실행

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

타겟 프로그램: 취약한 JSON 유사 파서

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

옵션 A: AFL++로 퍼징

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

옵션 B: libFuzzer로 퍼징

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

크래시 트리아지 및 근본 원인 분석

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

바이너리 분석: 기호 실행 및 컨콜릭 테스트

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

워크스루: Angr로 패워드 체크 해결

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

메모리_SANITIZE: ASan, MSan, UBSan

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

책임 있는 공개 및 CVE 할당

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제14장: 안전한 코딩 관행 및 컴파일러 경화

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

안전한 문자열 및 메모리 함수

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

경계 검사 및 안전한 정수 산술

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

컴파일러 플래그: FORTIFY_SOURCE, StackProtector, CFProtection

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

메모리 안전 언어: Rust 대안

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

코드 리뷰 기법

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

제15장: 익스플로잇 탐지, 사건 대응 및 포렌식

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

실제 버퍼 오버플로 공격 탐지

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

EDR 및 동작 모니터링

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

코어 덤프 분석 및 사후 디버깅

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

네트워크 레벨 익스플로잇 탐지

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

RCE 이벤트용 사건 대응 플레이북

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

포렌식 타임라인 재구성

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

주요 익스플로잇 사건에서 배운 교훈

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

워크스루: 스택 오버플로 익스플로잇의 포렌식 분석

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

시나리오: 침해된 웹 서비스

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 1: GDB로 코어 덤프 분석

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 2: 익스플로이팅 아티팩트를 위한 스택 검사

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 3: 취약한 함수 식별

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 4: Volatility로 메모리 이미지 분석

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 5: 메모리에서 네트워크 연결 추출

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 6: 프로세스 메모리에서 삽입된 코드 추출

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 7: 공격 타임라인 재구성

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

단계 8: 귀속 및 침해 지표

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

핵심 교훈

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

결론

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.

참고문헌

이 내용은 샘플 책에 포함되어 있지 않습니다. 책은 Leanpub에서 https://leanpub.com/buffer-overflow-exploit-defense-kr에서 구매할 수 있습니다.