

네트워크 보안 게임 프로젝트 계획서

프로젝트 개요

프로젝트명

TCP 패킷 분석 기반 네트워크 방어 게임

팀 구성

- 권장 인원: 3-4명
- 역할: 서버 개발, 클라이언트 개발, 네트워크 분석, UI/게임 로직

프로젝트 목적 및 목표

학습 목표

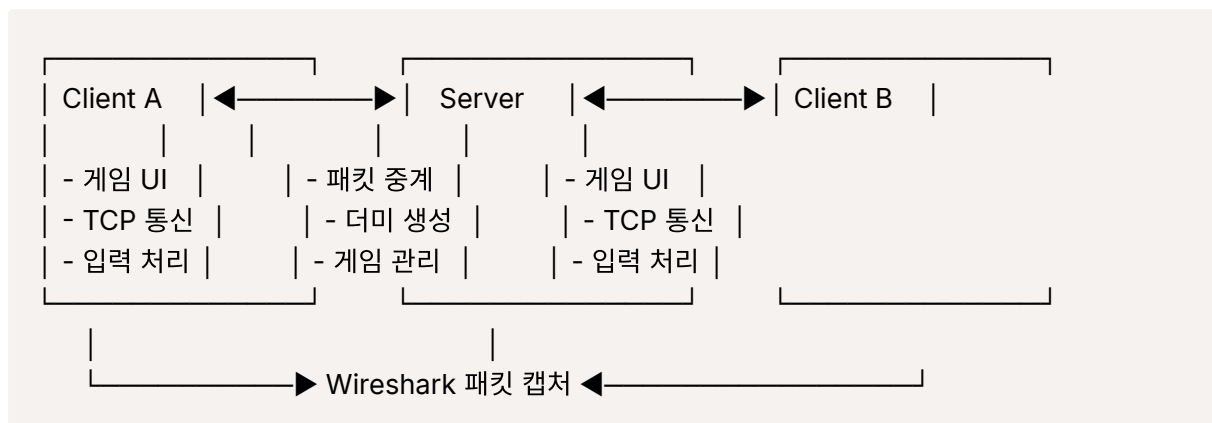
1. TCP/IP 프로토콜 이해 및 소켓 프로그래밍 실습
2. 패킷 분석 도구(Wireshark) 활용 능력 배양
3. 네트워크 보안 기초 개념 학습 (트래픽 분석, 공격 탐지)
4. 멀티스레드 네트워크 프로그래밍 경험

프로젝트 목표

- 실시간 멀티플레이어 네트워크 게임 구현
- TCP 패킷 송수신 및 분석 메커니즘 구현
- 교육용 네트워크 보안 시뮬레이션 완성

시스템 구조

전체 아키텍처



게임 플로우

1. [연결] 클라이언트들이 서버에 접속
↓
2. [대기] 플레이어 대기실 (최소 2명)
↓
3. [라운드 시작] 90초 타이머 시작
↓
4. [게임 진행]
 - 서버: 더미 패킷 지속 전송 (1-2초 간격)
 - 플레이어: 다른 플레이어에게 공격 패킷 전송
 - 각자 Wireshark로 자신을 향한 공격 패킷 탐지
↓
5. [방어 단계] 공격자 IP 입력 (20초)
↓
6. [점수 계산] 정답/오답 판정
↓
7. [다음 라운드] 또는 게임 종료

기술 스택

개발 환경

- 언어: Python 3.8+
- OS: Windows / Linux / macOS (크로스 플랫폼)
- IDE: VSCode, PyCharm 권장

서버 기술

- socket (TCP 통신)
- threading (멀티클라이언트 처리)
- json (메시지 직렬화)
- time (타이머 관리)

클라이언트 기술

- socket (서버 통신)
- tkinter 또는 PyQt5 (GUI)
- threading (비동기 처리)

네트워크 분석

- **Wireshark**: 패킷 캡처 및 분석
- **scapy** (선택): 패킷 생성 및 자동 분석

버전 관리

- **Git/GitHub:** 협업 및 코드 관리



프로토콜 설계

TCP 통신 포트

- 게임 포트: 9999
- 프로토콜: 커스텀 JSON 기반

메시지 타입

1. 더미 패킷 (서버 → 클라이언트)

```
{
  "type": "DUMMY",
  "timestamp": 1234567890,
  "payload": "DUMMY_ABCD1234"
}
```

2. 공격 패킷 (클라이언트 → 서버 → 타겟)

```
{
  "type": "ATTACK",
  "from": "192.168.1.10",
  "to": "192.168.1.20",
  "payload": "ATTACK_TARGET_PlayerB"
}
```

3. 방어 입력 (클라이언트 → 서버)

```
{
  "type": "DEFENSE",
  "attacker_ip": "192.168.1.10"
}
```

4. 점수 업데이트 (서버 → 클라이언트)

```
{
  "type": "SCORE",
  "player_id": "PlayerA",
  "score": 85,
}
```


```
"correct": true
}
```

17 개발 일정

기초 구현 및 환경 설정

목표: 기본 TCP 통신 구현


작업	담당	산출물
프로젝트 환경 설정	전원	README, .gitignore
기본 TCP 서버 구현	팀원 A	server.py
기본 TCP 클라이언트 구현	팀원 B	client.py
프로토콜 정의 문서 작성	팀원 C	protocol.md
테스트: 단일 클라이언트 연결	전원	테스트 성공

마일스톤 1:  서버-클라이언트 기본 통신 완료

핵심 기능 개발

목표: 멀티클라이언트 및 패킷 전송

작업	담당	산출물
멀티클라이언트 처리 (threading)	팀원 A	server.py 업데이트
더미 패킷 생성 및 브로드캐스트	팀원 A	dummy_generator.py
공격 패킷 전송 기능	팀원 B	client.py 업데이트
Wireshark 필터 가이드 작성	팀원 C	wireshark_guide.md
테스트: 3개 클라이언트 동시 연결	전원	테스트 성공

마일스톤 2:  패킷 송수신 메커니즘 완료

게임 로직 및 UI

목표: 게임 메커니즘 구현


작업	담당	산출물
라운드 타이머 구현	팀원 A	game_manager.py
점수 계산 시스템	팀원 A	score_system.py
클라이언트 GUI 개발	팀원 B	gui.py
방어 입력 검증 로직	팀원 C	validation.py
게임 상태 동기화	팀원 C	서버-클라이언트 업데이트

마일스톤 3:  완전한 게임 한 라운드 플레이 가능

통합 및 개선

목표: 버그 수정 및 기능 보완

작업	담당
전체 통합 테스트	전원
버그 수정	전원
예외 처리 강화	전원
로그 시스템 추가	팀원 A
UI/UX 개선	팀원 B
사용자 매뉴얼 작성	팀원 C

마일스톤 4:  안정적인 게임 플레이 가능

최종 완성 및 발표 준비

목표: 프로젝트 마무리

작업	담당
최종 테스트	전원
코드 정리 및 주석	전원
발표 자료 제작 (PPT)	팀원 B, C
시연 영상 제작	팀원 B
최종 보고서 작성	전원
리허설	전원

최종 산출물:

- 소스 코드 (GitHub)
- 사용자 매뉴얼
- 발표 자료
- 시연 영상
- 최종 보고서

역할 분담 (예시)

팀원 A: 서버 개발자

- TCP 서버 구현
- 멀티클라이언트 관리
- 더미 패킷 생성
- 게임 로직 관리
- 점수 시스템

팀원 B: 클라이언트 개발자

- TCP 클라이언트 구현
- GUI 인터페이스 개발
- 사용자 입력 처리
- 서버 통신 관리

팀원 C: 네트워크 분석 및 문서화

- 프로토콜 설계
- Wireshark 가이드 작성
- 패킷 검증 로직
- 테스트 시나리오 작성
- 문서화 총괄

(선택) 팀원 D: QA 및 UI/UX

- 통합 테스트
- 버그 리포팅
- UI/UX 개선
- 사용자 매뉴얼 작성

게임 규칙 (상세)

게임 설정

- 플레이어: 2-4명
- 라운드: 5라운드
- 라운드당 시간: 90초
- 초기 체력: 100 HP

점수 시스템

행동	점수
공격 패킷 정확히 탐지	+10점
잘못된 IP 입력	-5점
공격 탐지 실패 (놓침)	-10 HP
공격 성공 (상대가 놓침)	+5점

승리 조건

1. 최종 라운드 후 가장 높은 점수

2. 또는 마지막까지 생존 (HP > 0)

게임 진행

1. 준비 단계 (10초)

- Wireshark 필터 설정: `tcp.port == 9999`
- 게임 UI 확인

2. 공격/방어 단계 (60초)

- 서버가 모든 플레이어에게 더미 패킷 전송 (1-2초 간격)
- 각 플레이어는 원하는 타겟에게 공격 패킷 전송 (횟수 제한 없음)
- 각자 Wireshark로 자신을 향한 공격 패킷 찾기

3. 방어 입력 단계 (20초)

- 발견한 공격자의 IP 주소를 게임 UI에 입력
- 여러 공격이 있었다면 모두 입력 가능

4. 결과 발표

- 정답 여부 확인
- 점수 및 HP 업데이트

Wireshark 사용 가이드

필수 필터

```
tcp.port == 9999
```

더미 패킷 식별

- 출발지: 서버 IP (예: 192.168.1.1)
- 페이로드 패턴: `DUMMY_` 로 시작

공격 패킷 식별

- 출발지: 다른 클라이언트 IP
- 페이로드 패턴: `ATTACK_` 로 시작
- 목적지: 본인의 IP

분석 팁

1. Follow TCP Stream으로 전체 대화 확인
2. Statistics > Conversations로 IP별 통신량 확인
3. 시간 순서대로 정렬하여 최근 패킷 확인

! 예상 문제 및 해결 방안

문제 1: 방화벽 차단

증상: 클라이언트가 서버에 연결되지 않음

해결방안:

```
# Windows 방화벽 예외 추가
- 제어판 > Windows Defender 방화벽 > 고급 설정
- 인바운드 규칙 > 새 규칙 > 포트 9999 허용

# Linux (iptables)
sudo iptables -A INPUT -p tcp --dport 9999 -j ACCEPT
```

문제 2: 다른 클라이언트 패킷이 보이지 않음

증상: Wireshark에서 자신의 패킷만 캡처됨

해결방안:

- **방법 1:** 서버가 모든 패킷을 각 클라이언트에 재전송 (권장)
- **방법 2:** 허브 사용 또는 네트워크 스위치 포트 미러링
- **방법 3:** 로컬 VM 환경 구축 (Host-Only Network)

문제 3: 실시간 분석이 너무 어려움

증상: 학생들이 제한 시간 내에 분석을 못함

해결방안:

- 라운드 시간 연장 (60초 → 90초)
- 더미 패킷 전송 빈도 감소 (0.5초 → 2초)
- 힌트 시스템 추가 (공격 받았을 때 알림)

문제 4: JSON 인코딩 오류

증상: 한글 또는 특수문자 전송 시 오류

해결방안:

```
# 서버/클라이언트 모두 UTF-8 인코딩 사용
message.encode('utf-8')
data.decode('utf-8')
```

문제 5: 동기화 문제

증상: 클라이언트마다 다른 게임 상태

해결방안:

- 서버 중심 상태 관리
 - 주기적인 상태 동기화 메시지 전송
 - 타임스탬프 기반 이벤트 순서 보장
-

평가 기준

기능 구현 (40%)

- ☐ TCP 서버-클라이언트 통신 (10%)
- ☐ 멀티클라이언트 처리 (10%)
- ☐ 더미/공격 패킷 송수신 (10%)
- ☐ 게임 로직 (점수, 타이머) (10%)

기술적 완성도 (30%)

- ☐ 코드 품질 (가독성, 구조) (10%)
- ☐ 예외 처리 및 안정성 (10%)
- ☐ 네트워크 프로토콜 설계 (10%)

창의성 및 난이도 (15%)

- ☐ 독창적인 기능 추가 (5%)
- ☐ 구현 난이도 (10%)

문서화 및 발표 (15%)

- ☐ 코드 주석 및 README (5%)
 - ☐ 사용자 매뉴얼 (5%)
 - ☐ 발표 자료 및 시연 (5%)
-

참고 자료

Python 소켓 프로그래밍

- [Python Socket 공식 문서](#)
- [Real Python - Socket Programming](#)

Wireshark

- [Wireshark 공식 사이트](#)
- [Wireshark Display Filters](#)

네트워크 기초

- TCP/IP 프로토콜 개론
- 컴퓨터 네트워크 교재

Git/GitHub

- [Git 기본 명령어](#)
- [GitHub 협업 가이드](#)

제출물

최종 제출 구조

```
project_root/
├── README.md          # 프로젝트 설명
├── requirements.txt    # 필요 라이브러리
├── server.py           # 서버 메인
├── client.py           # 클라이언트 메인
├── protocol.py         # 프로토콜 정의
├── game_manager.py     # 게임 로직
├── gui.py              # GUI (선택)
├── docs/
│   ├── protocol.md     # 프로토콜 문서
│   ├── user_manual.md  # 사용자 매뉴얼
│   └── wireshark_guide.md # Wireshark 가이드
├── tests/              # 테스트 코드
├── demo/               # 시연 영상/스크린샷
└── presentation.pptx   # 발표 자료
```

제출 방법

1. GitHub Repository 링크
2. 발표 자료 (PDF/PPTX)
3. 최종 보고서 (PDF)
4. 시연 영상 (MP4, 5분 이내)

체크리스트

개발 시작 전

- ☐ 팀 구성 완료
- ☐ 역할 분담 완료
- ☐ 개발 환경 설정 (Python, Git 설치)

- ☐ GitHub Repository 생성
- ☐ Wireshark 설치 및 사용법 학습

개발 중

- ☐ 주 1회 팀 미팅
- ☐ 코드 리뷰 진행
- ☐ 마일스톤별 테스트 통과
- ☐ 진행 상황 문서화

개발 완료 후

- ☐ 전체 통합 테스트 완료
 - ☐ 사용자 매뉴얼 작성
 - ☐ 발표 자료 제작
 - ☐ 시연 영상 녹화
 - ☐ 최종 보고서 작성
 - ☐ GitHub 정리 (README, 주석)
-

기대 효과

학습 효과

1. **네트워크 프로그래밍:** TCP/IP 소켓 프로그래밍 실전 경험
2. **패킷 분석:** Wireshark를 활용한 실제 트래픽 분석 능력
3. **협업:** Git을 활용한 팀 프로젝트 경험
4. **문제 해결:** 실시간 네트워크 문제 디버깅 능력

포트폴리오

- 완성도 높은 네트워크 프로젝트
 - GitHub 오픈소스 프로젝트
 - 기술 블로그 포스팅 소재
-