ASM-Lite: Attack Surface Discovery Tool

Phase 1 - Mini Project Proposal

Project Overview

Project Title: ASM-Lite - Ethical Attack Surface Discovery with Trust Scoring
Duration: 16 weeks (8 Scrum sprints × 2 weeks each)
Core Purpose: Build a lightweight tool that helps small/medium organizations discover their internet-facing assets safely and ethically

What makes this project unique:

- **Ethical scanning** - Respects robots.txt and rate limits
- **Trust scoring** - Gives confidence levels for discovered assets
- **SME-focused** - Designed for smaller organizations without huge security teams

---

Technical Implementation

Core Technology Stack

```
Backend: Python 3.10+ with FastAPI
Database: SQLite (simple, no server setup needed)
CLI Interface: Click framework
Testing: pytest with coverage reporting
Documentation: LaTeX for academic report
Deployment: Docker containerization
```

Key Features to Build

Sprint 1-2: Foundation

- Basic CLI tool structure
- Database schema for storing discovered assets
- Simple web scraping with ethical constraints

Sprint 3-4: Discovery Engine

- Certificate transparency log querying
- DNS enumeration (subdomain discovery)
- Port scanning with rate limiting

Sprint 5-6: Trust Scoring

- Algorithm to score asset confidence
- Multiple source verification
- False positive filtering

Sprint 7-8: Polish & Deploy

- Error handling and logging
- Docker containerization
- Final testing and documentation

Scrum Framework Setup

Team Structure

- **Product Owner:** Your faculty supervisor
- **Scrum Master:** Assigned faculty member
- **Development Team:** You (student)
- **External Validators:** 3-5 security professionals who will review your work

Scrum Book Requirements

Create a physical notebook with these sections:

Section 1: Product Backlog

- User stories for each feature
- Sprint planning notes
- Priority rankings with faculty approval

Section 2: Database & UI Design

- Entity-relationship diagrams
- CLI command structure
- Screen mockups (even for CLI tools)

Section 3: Testing Documentation

- Test cases written before code (TDD approach)
- Bug reports and fixes
- Code coverage reports

Section 4: Version Control

- Git commit summaries
- Docker image tags
- Release notes

💡 *Tip: Date and get faculty signatures on major decisions*

Stakeholder Engagement Plan

Required Interactions (for CO2 compliance)

Week 2: Interview 3 academic researchers about attack surface management needs
Week 5: Workshop with 2 security practitioners to validate requirements
Week 9: Ethics board review for GDPR compliance
Week 13: Open-source community feedback session

Documentation Requirements

- Record all meetings (audio/video with permission)
- Keep signed feedback forms
- Archive email conversations
- Include stakeholder quotes in final report

💡 *Tip: Reach out to local cybersecurity companies, they often welcome student projects*

Timeline & Milestones

Phase Breakdown

```
Weeks 1-2:   Setup & Planning
Weeks 3-6:   Core Development (Sprints 1-2)
Weeks 7-10:  Feature Development (Sprints 3-4)
Weeks 11-14: Advanced Features (Sprints 5-6)
Weeks 15-16: Finalization (Sprints 7-8)
```

Critical Milestones

- **Week 2:** Scrum Book initialized, Git repo approved
- **Week 4:** Basic OSINT engine working
- **Week 8:** CLI tool can discover subdomains
- **Week 12:** Trust scoring algorithm implemented
- **Week 15:** LaTeX report complete, Docker image published

Testing Strategy (TDD Approach)

Write Tests First

python

```python
# Example test structure
def test_ethical_scanning():
    # Test that tool respects robots.txt
    assert check_robots_compliance("example.com") == True

def test_trust_scoring():
    # Test confidence calculation
    asset = Asset(dns_verified=True, cert_valid=True, port_open=True)
    assert calculate_trust_score(asset) > 0.8
```

Coverage Requirements

- Minimum 85% code coverage
- All critical functions must have tests
- Integration tests for CLI commands

Deliverables Checklist

Academic Requirements

- [ ] Scrum Book with dated entries and faculty signatures
- [ ] Git repository with 70+ meaningful commits
- [ ] LaTeX report (25 pages, IEEE format)
- [ ] Stakeholder interaction evidence (5+ documented meetings)
- [ ] Test coverage reports

Technical Deliverables

- [ ] Working CLI tool
- [ ] Docker image published to Docker Hub
- [ ] Source code with comprehensive documentation
- [ ] User manual
- [ ] Demo video

Professional Artifacts

- [ ] Project portfolio entry
- [ ] GitHub repository with proper README
- [ ] Professional presentation slides

Risk Management

Common Risks & Solutions

Risk: Scrum Master unavailable
Solution: Backup faculty member assigned, virtual meetings setup

Risk: LaTeX learning curve
Solution: Pre-built templates provided, workshop in Week 1

Risk: API rate limiting
Solution: Implement exponential backoff, use multiple data sources

Risk: Stakeholder scheduling
Solution: Offer virtual participation, record sessions

Risk: Docker deployment issues
Solution: Start containerization early, use GitHub Actions

Resources & Support

Development Environment

- **Minimum:** i5 processor, 8GB RAM
- **Recommended:** i7 processor, 16GB RAM, SSD
- **Cloud option:** GitHub Codespaces for consistent environment

External APIs (Free tiers)

- Certificate Transparency logs (crt.sh)
- DNS resolution (Quad9, Cloudflare)
- Port scanning (with rate limits)

Learning Resources

- Scrum Guide (official PDF)
- Python ethical hacking tutorials
- LaTeX documentation and templates
- Docker beginner guides

Success Metrics

Technical Metrics

- Tool can discover 90%+ of known assets in test environment
- Trust scoring accuracy >85% compared to manual verification
- CLI tool runs without errors on fresh Linux installation
- Docker image size <500MB

Academic Metrics

- All syllabus requirements met (98%+ compliance)
- Stakeholder feedback positive (>4/5 rating)
- Faculty approval for all major milestones
- LaTeX report passes plagiarism check

Professional Metrics

- Code quality suitable for production use
- Documentation sufficient for handover
- Portfolio piece ready for job applications
- Potential for open-source community adoption

Getting Started

Week 1 Action Items

1. Set up development environment
2. Create GitHub repository
3. Initialize Scrum Book
4. Schedule first stakeholder meeting
5. Install LaTeX and practice with templates
6. Draft initial product backlog

Faculty Approval Required

- [ ] Project scope and timeline
- [ ] Stakeholder engagement plan
- [ ] Technical architecture
- [ ] Risk mitigation strategy

Remember: This is your project, but faculty guidance is crucial for success. Don't hesitate to ask for help!