Automated Reconnaissance Orchestrator (ARO) - Phase 1

Lightweight Edition for S3 Mini Project

📋 Project Overview

- **Duration:** 14 Weeks (3-4 Months)
- **Objective:** Build a DevOps-compliant pipeline that automates subdomain discovery, live host detection, and basic vulnerability scanning using 6 core tools
- **Target Grade:** A+ (Exceeds all syllabus requirements)

🎯 Core Features & Tool Chain

Module	Function	Tools Used	Output Format	Success Metrics
Subdomain Discovery	Find root/subdomains	Subfinder, crt.sh	`subdomains.txt`	>95% known subdomain coverage
Live Host Screening	Filter active hosts	HTTPX	`live_hosts.txt`	Sub-5 second response validation
Endpoint Extraction	Extract URLs from JS files	unfurl	`endpoints.txt`	JavaScript parsing accuracy
Vulnerability Scanning	Identify CVEs/misconfigurations	Nuclei (10 templates)	`vulnerabilities.csv`	Zero false positives
Reporting	Generate summary reports	LaTeX + Pandoc	`report.pdf`	Professional-grade output
Orchestration	Chain tools with error handling	Python + Docker	Automated scan logs	100% pipeline reliability

---

🏗 System Architecture

mermaid

```
graph TD
    A[Target Domain Input] --> B[Subfinder Discovery]
    B --> C[Certificate Transparency Check]
    C --> D[Subdomain Validation]
    D --> E[HTTPX Live Host Filter]
    E --> F[JavaScript Endpoint Extraction]
    F --> G[Nuclei Vulnerability Scan]
    G --> H[Result Aggregation]
    H --> I[LaTeX Report Generation]
    I --> J[PDF Output + Logs]

    K[Error Handler] --> B
    K --> E
    K --> G

    L[Resource Monitor] --> M[Memory/CPU Limits]
    M --> N[Scan Queue Manager]
```

💻 Technology Stack

Category	Technology	Version	Justification
Core Language	Python	3.10+	Native async support, extensive security libraries
Containerization	Docker	24.0+	Syllabus compliance, tool isolation, reproducibility
Version Control	Git	2.40+	Mandatory for evaluation, CI/CD integration
CI/CD	GitHub Actions	Latest	Automated testing, Docker builds
Documentation	LaTeX (Overleaf)	2023	Syllabus requirement, professional output
Package Management	Poetry	1.6+	Dependency management, virtual environments
Testing	pytest	7.4+	TDD compliance, pipeline validation

🔧 Additional Tools & Libraries

python

```
# requirements.txt preview
asyncio==3.4.3      # Async tool orchestration
docker==6.1.3       # Container management
pyyaml==6.0.1       # Configuration management
pandas==2.0.3       # Data processing
jinja2==3.1.2       # Report templating
click==8.1.6        # CLI interface
```

📅 Implementation Timeline (14 Weeks)

Phase 1: Foundation Setup (Weeks 1-4)

Week 1: Environment Setup

- □ Development environment configuration
- □ Docker installation and testing
- □ Git repository initialization
- □ Tool compatibility testing

Week 2: Core Tools Integration

- □ Subfinder Dockerization
- □ HTTPX integration
- □ Basic Python orchestrator (`scan.py`)
- □ Initial CI/CD pipeline

Week 3: Pipeline Development

- □ Tool chain integration
- □ Error handling implementation
- □ Configuration management
- □ Logging system setup

Week 4: Testing & Validation

- □ Unit test development
- □ Integration testing
- □ Performance benchmarking
- □ Security validation

Time Allocation: 10 hrs/week (Focus: Learning curve + setup)

Phase 2: Pipeline Integration (Weeks 5-8)

Week 5-6: End-to-End Pipeline

- □ Complete scan workflow: `domain → subdomains → live hosts → Nuclei`
- □ Data flow optimization
- □ Result validation
- □ Performance tuning

Week 7-8: Testing & Refinement

- □ Comprehensive unit tests
- □ Tool handoff validation

□ Error recovery testing

□ Resource usage optimization

Time Allocation: 8 hrs/week (Focus: Integration)

Phase 3: Reporting & Documentation (Weeks 9-12)

Week 9-10: Report Generation

□ LaTeX template development

□ Automated report generation

□ Data visualization

□ Executive summary automation

Week 11-12: Error Handling & Robustness

□ Comprehensive error handling

□ Graceful degradation

□ Recovery mechanisms

□ Performance monitoring

Time Allocation: 5 hrs/week (Focus: Documentation)

Phase 4: Finalization & Polish (Weeks 13-14)

Week 13: Final Integration

□ One-liner execution: `./aro.sh example.com`

□ Complete system testing

□ Performance optimization

□ Security hardening

Week 14: Documentation & Delivery

□ Scrum Book completion

□ Final report generation

□ Code cleanup and commenting

□ Submission preparation

Time Allocation: 10 hrs/week (Focus: Polish + documentation)

✅ Syllabus Compliance Matrix

Requirement	Implementation	Evidence Location	Validation Method
Individual Project	Solo development with git attribution	Git commit history	Commit timestamps and authorship
Scrum Methodology	Bi-weekly sprints with documentation	`docs/scrum_book.md`	Sprint retrospectives
Git Version Control	Daily commits with meaningful messages	GitHub repository	Commit frequency analysis
LaTeX Report	Auto-generated from scan results	`reports/report.tex`	Compilation success
Docker Usage	All tools containerized	`dockerfiles/` directory	Container builds
TDD	Comprehensive test suite	`tests/` directory	Coverage reports
Documentation	Inline + external docs	`README.md` + `/docs`	Documentation coverage

---

🖥️ Hardware Optimization

Dell Inspiron i5-1135G7 Specifications:

- **CPU:** Intel Core i5-1135G7 (4 cores, 8 threads)
- **RAM:** 16GB DDR4
- **Storage:** 512GB NVMe SSD
- **Network:** Wi-Fi 6

Resource Management Strategy:

```python
# Resource governor implementation
import resource
import psutil

class ResourceGovernor:
    def __init__(self):
        self.max_memory = 10_000_000_000  # 10GB limit
        self.max_cpu_percent = 80

    def enforce_limits(self):
        # Memory limit
        resource.setrlimit(resource.RLIMIT_AS,
                          (self.max_memory, self.max_memory))

        # CPU monitoring
        if psutil.cpu_percent() > self.max_cpu_percent:
            self.throttle_operations()
```

Scan Safety Parameters:

- **Max Subdomains:** 50 per scan
- **HTTP Request Delay:** 2 seconds
- **Concurrent Threads:** 4 maximum
- **Timeout Values:** 30 seconds per tool

📊 Expected Outputs

Terminal Interface:

```bash
$ ./aro.sh example.com
[INFO] Starting ARO scan for example.com
[+] Subdomain Discovery: Found 23 subdomains
[+] Live Host Detection: 14 hosts responding
[+] Vulnerability Scan: 2 critical findings
    └── CVE-2023-1234: SQL Injection (CVSS: 9.8)
    └── CVE-2023-5678: XSS Vulnerability (CVSS: 8.2)
[+] Report generated: reports/example_com_2025-07-06.pdf
[INFO] Scan completed in 4m 32s
```

Report Structure:

```latex
% Auto-generated LaTeX report
\documentclass{article}
\usepackage{graphicx}
\usepackage{booktabs}

\title{Security Assessment Report: example.com}
\author{ARO v1.0}
\date{\today}

\begin{document}
\maketitle

\section{Executive Summary}
% Automated summary generation

\section{Findings}
% Vulnerability table with CVSS scores

\section{Recommendations}
% Prioritized remediation steps
\end{document}
```

🔍 Risk Management

Risk	Probability	Impact	Mitigation Strategy
Tool Compatibility Issues	Medium	High	Use official Docker images, version pinning
Time Overrun	High	Medium	Prioritize core features, defer enhancements
Learning Curve	High	Low	Weekly mentor meetings, documentation
Hardware Limitations	Medium	Medium	Resource monitoring, scan throttling
Network Restrictions	Low	High	University network approval, VPN testing

---

📈 Success Metrics

Technical Metrics:

- **Code Coverage:** >90%
- **Test Success Rate:** 100%
- **Build Success Rate:** >95%
- **Documentation Coverage:** 100%

Academic Metrics:

- **Syllabus Compliance:** 100%
- **Deliverable Quality:** Grade A target
- **Innovation Factor:** Use of modern DevOps practices

🎯 Scope Comparison

Component	Original Vision	Phase 1 Scope	Justification
Tools	15+ integrated tools	6 core tools	Focus on quality over quantity
Asset Graphing	Interactive D3.js	Text-based summary	Complexity reduction
CI/CD	Multi-stage pipeline	Single GitHub Actions	Sufficient for validation
ML Components	Planned integration	Removed	Phase 2 expansion
Development Time	150+ hours	80-100 hours	Realistic for S3 timeframe

🚀 Phase 1 Advantages

1. **Realistic Timeframe:** 5-10 hours/week manageable alongside other coursework
2. **Complete Syllabus Coverage:** Exceeds all requirements with buffer
3. **Foundation Building:** Establishes base for Phase 2 expansion
4. **Hardware Compatibility:** Optimized for available resources
5. **Learning Focused:** Emphasizes understanding over complexity

📋 Next Steps

Pre-Development Phase:

1. **Week 0:** Submit for supervisor approval
2. **Week 0:** Finalize development environment
3. **Week 1:** Initialize Git repository and CI/CD

Development Kickoff:

1. **Day 1:** Docker environment setup
2. **Day 3:** First tool integration (Subfinder)
3. **Day 5:** Basic orchestrator framework
4. **Day 7:** Initial commit and CI test

📞 Support Structure

Academic Support:

- **Supervisor:** Weekly progress meetings
- **Scrum Master:** Bi-weekly methodology guidance
- **Peer Review:** Monthly code reviews

Technical Support:

- **Documentation:** Comprehensive inline comments
- **Community:** Tool-specific GitHub issues
- **Mentorship:** Senior student guidance

This proposal represents a carefully balanced approach to delivering a high-quality, syllabus-compliant project within realistic constraints while building a foundation for future expansion.