Final Project Specification: AI-Powered Malware Analysis Tools

Course Project

Project Duration: January 13, 2026 – March 22, 2026

1 Overview

This final project spans **2 months** with a final deadline of **March 22, 2026**. Students must select one of three project ideas focused on applying AI and LLMs to cybersecurity analysis tasks. All projects include both an implementation component and an optional research component for comparing results with state-of-the-art approaches.
Submission link: `https://forms.gle/zAarhGAJQKzzALTH9`

1.1 Project Deliverables

1. **Project Plan** (Due: **January 25, 2026**)
   - Project title
   - Team members (for group projects only)
   - Methodology description
   - Tools and frameworks to be used
   - For research component:
     - Clear research questions
     - Related work survey with comparison table
     - Identification of papers/implementations for comparison
     - Dataset selection and justification
     - Evaluation plan with metrics
2. **Implementation** (Due: **March 22, 2026**)
   - Complete working system
   - Docker containerization (Dockerfile and docker-compose)
   - `start.sh` script for environment setup
   - Documentation and README
3. **Final Report & Presentation** (Due: March 22, 2026)
   - Technical report documenting implementation
   - Evaluation results
   - Research component results (if applicable)
   - Potential manuscript draft (if results warrant publication)

2 General Requirements

2.1 Version Control

- All code must be tracked using **GitHub** with regular commits
- For group projects (max 2 students), commits must clearly show individual contributions
- Commit messages should be descriptive and follow best practices

2.2 Containerization

All projects must be shipped as Docker containers with:

- `Dockerfile` defining the container image
- `docker-compose.yml` for multi-container orchestration (if needed)
- `start.sh` script that:
  - Checks for required environment variables
  - Validates dependencies
  - Starts the workflow/service

Rationale: Docker ensures reproducibility and proper dependency management across different environments.

2.3 Research Component

Each project includes an optional but encouraged research component involving:

- Comparison with state-of-the-art approaches
- Quantitative evaluation using established metrics
- Potential for manuscript preparation if results are significant
- **Regular supervision meetings** available for research-focused students
- **Flexible deadlines** that can be adjusted based on conference submission cycles

3 Project Ideas

3.1 Project 1: Agentic Workflow for Automated Binary/APK Analysis

3.1.1 Project Description

Develop an AI-powered agentic workflow system that automates the analysis of either binary executables or Android APK files. The system should leverage multiple specialized tools orchestrated by LLM-based agents to perform comprehensive security analysis.

3.1.2 Technical Requirements

Core Technologies

- **MCP Servers:** Use FastMCP to implement analysis tools as Model Context Protocol servers
- **Agent Framework:** Choose one of:
  - Agno framework (recommended)
  - CrewAI
  - LangChain
  - SmolAgents

3.1.3 Tool Design Philosophy

Do not simply wrap existing command-line tools. Instead, create **specialized, high-level tools** that simplify analysis for agents.

Examples of good tool design:

- **For Binary Analysis:**
  - `extract_crypto_constants()` - Identifies cryptographic constants in binary
  - `find_suspicious_syscalls()` - Detects unusual system call patterns
  - `analyze_control_flow_anomalies()` - Identifies obfuscation patterns
  - `extract_strings_with_context()` - Returns strings with their usage context
- **For APK Analysis:**
  - `extract_permissions_with_risk()` - Returns permissions with risk assessment
  - `find_hardcoded_secrets()` - Identifies API keys, tokens, credentials
  - `analyze_network_behavior()` - Extracts and analyzes network communication
  - `detect_obfuscation_techniques()` - Identifies code obfuscation methods

Bad examples (too simplistic):

- `run_radare2_command(cmd)` - Just wraps a raw command
- `execute_jadx(args)` - Direct tool execution without abstraction

3.1.4 Research Component (Optional)

Goal: Develop a framework that automatically solves CTF challenges in binary exploitation or Android reverse engineering categories.

Research Questions:

1. How effective are LLM-based agents at solving binary analysis CTF challenges compared to human experts?
2. What types of challenges are most/least amenable to automated agentic approaches?
3. How does tool design impact agent performance in security analysis tasks?

Key References:

- *CTFusion: A CTF-based Benchmark for LLM Agent Evaluation* (`https://openreview.net/pdf?id=2zQJHLbyqM`)
- *Hacking CTFs with Plain Agents* (`https://arxiv.org/abs/2412.02776`)
- *LLMs as Firmware Experts: A Runtime-Grown Tree-of-Agents Framework* (`https://arxiv.org/pdf/2511.18438`)
- *ClearAgent: Agentic Binary Analysis for Effective Vulnerability Detection* (`https://dl.acm.org/doi/10.1145/3759425.3763397`)

Motivation Papers:

- *Cybersecurity AI: The World's Top AI Agent for Security CTF* (`https://arxiv.org/pdf/2512.02654`)
- *Cracking CTFs and Finding Zero-Days with AI-Agents* (Medium article)

Evaluation Plan:

- Select a benchmark set of CTF challenges (e.g., 50-100 challenges from picoCTF, HackTheBox, etc.)
- Measure: solve rate, time to solution, number of tool invocations
- Compare with baseline approaches (GPT-4 without tools, traditional automated tools)
- Qualitative analysis of failure modes

3.2 Project 2: Automated YARA Rule Generation using LLMs

3.2.1 Project Description

Develop a tool that automatically generates YARA rules for malware detection given a collection of malware samples. The tool should leverage LLMs to analyze binary characteristics and produce high-quality, discriminative YARA signatures.

3.2.2 Dataset Selection

Choose one of the following datasets:

1. **MOTIF Dataset** (`https://github.com/boozallen/MOTIF`)
   - Focus: Static analysis features
   - Best for: Learning structural patterns
2. **Custom Dataset** (Provided by instructor)
   - Focus: Both static and dynamic analysis
   - Best for: Hybrid rule generation
3. **AVAST-CTU Dataset** (`https://github.com/avast/avast-ctu-cape-dataset`)
   - Focus: Dynamic analysis with CAPE sandbox reports
   - Best for: Behavior-based signatures

3.2.3 Approach

YARA rules can be generated from:

- **Static analysis only** - Binary patterns, strings, PE headers
- **Dynamic analysis only** - Behavioral patterns from sandbox execution
- **Hybrid approach** - Combining both static and dynamic features

3.2.4 Required Viewing

- `https://youtu.be/jxM-WjtcmFo?si=jLJ1OghtUflASJNB`
- `https://youtu.be/35Exd9GrR5I?si=OKbPMsPiCQepa1QE`
- `https://youtu.be/zzpz3VYKzUw?si=RkjGbKB765nrfGUJ`

3.2.5 Required Reading

- *APIARY: Automatic Yara Rule Generation*
  `https://dl.acm.org/doi/epdf/10.1145/3576915.3616625`
- *Recent work on YARA rule generation*
  `https://www.sciencedirect.com/science/article/pii/S0167404825000860`
- *YAMME and related approaches*
  `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10177752`

3.2.6 Research Component (Optional)

Goal: Compare your LLM-based approach against state-of-the-art YARA generation tools.

Baselines for Comparison:

- **PackGenome** - Genetic algorithm-based generation
- **APIARY** - Automated rule generation from PE files
- **YAMME** - Yet Another Malware Manual Extractor

Research Questions:

1. Can LLMs generate more discriminative YARA rules than existing automated methods?
2. How do LLM-generated rules compare in terms of precision, recall, and false positive rates?
3. What role does the underlying analysis approach (static/dynamic/hybrid) play in rule quality?

Evaluation Metrics:

- **Detection Rate:** Percentage of malware samples correctly identified
- **False Positive Rate:** Percentage of benign files incorrectly flagged
- **Rule Specificity:** How narrowly rules target specific malware families
- **Rule Complexity:** Number of conditions, string patterns, etc.

Method	Approach	Dataset	DR (%)	FPR (%)
PackGenome	Genetic Algorithm	?	?	?
APIARY	Static Analysis	?	?	?
YAMME	Manual + Auto	?	?	?
Your Method	LLM-based	Selected	TBD	TBD

Table 1: Comparison with state-of-the-art (to be completed)

- **Generalization:** Performance on unseen samples from same malware family

Comparison Table (to be completed in project plan):
Implementation Reproducibility:

- Check if papers provide GitHub repositories
- If not available, contact authors or implement from paper description
- Document any deviations from original implementations

3.3 Project 3: Automated Dynamic Analysis Signature Generation

3.3.1 Project Description

Develop a tool that automatically generates CAPEv2 signatures from sandbox execution logs. The tool analyzes behavioral patterns in malware execution traces and produces Python-based signatures that can detect similar behaviors in future analyses.

Extension Goals:

- Automatically map behaviors to MITRE ATT&CK techniques
- Generate comprehensive malware analysis reports similar to those published by security vendors (Palo Alto Unit42, ESET WeLiveSecurity, Symantec, etc.)

3.3.2 Background: CAPEv2 Signatures

CAPEv2 signatures are Python classes that detect specific malware behaviors. See documentation: `https://capev2.readthedocs.io/en/latest/customization/signatures.html`

Example signature structure:

```
class SuspiciousNetwork(Signature):
    name = "suspicious_network"
    description = "Connects to suspicious domains"
    severity = 3
    categories = ["network"]

    def run(self):
        for call in self.get_raw_argument("network", "dns"):
            if self.check_domain(call["domain"]):
                self.mark_call()
                return True
        return False
```

3.3.3 Dataset Selection

Choose one of:

1. **AVAST-CTU Dataset** (`https://github.com/avast/avast-ctu-cape-dataset`)
   - Large-scale dataset with CAPEv2 sandbox reports
   - Contains diverse malware families
2. **Custom Dataset** (Provided by instructor)
   - Curated samples with known behaviors
   - May include additional metadata

3.3.4 Technical Approach

Phase 1: Signature Generation

1. Parse CAPEv2 JSON logs (API calls, network activity, file operations)
2. Extract behavioral patterns using LLMs
3. Generate Python signature classes
4. Validate signatures against test set

Phase 2: MITRE ATT&CK Mapping

1. Analyze detected behaviors
2. Map to relevant ATT&CK techniques (e.g., T1055 for process injection)
3. Generate structured ATT&CK reports

Phase 3: Report Generation (Optional)

1. Synthesize findings into human-readable reports
2. Include: executive summary, technical analysis, IOCs, recommendations
3. Style similar to industry threat reports

3.3.5 Research Component (Optional)

Research Questions:

1. How accurate are LLM-generated behavioral signatures compared to manually crafted ones?
2. Can automated signature generation reduce the time required for malware analysis?
3. How well do generated signatures generalize to new variants of known malware families?

Evaluation Metrics:

- **Signature Quality:** Precision, recall, F1-score
- **Coverage:** Percentage of malware behaviors detected
- **ATT&CK Mapping Accuracy:** Correctness of technique assignments

- **Report Quality:** Assessed via expert review (if feasible)
- **Efficiency:** Time savings compared to manual analysis

Baseline Comparisons:

- Existing CAPEv2 community signatures
- Manual signature creation by security analysts
- Rule-based behavior detection systems

4 Project Plan Template

Your project plan (due January 25, 2026) should include the following sections:

4.1 Project Information

- Project title and chosen project number (1, 2, or 3)
- Team members and roles (for group projects)
- GitHub repository URL

4.2 Methodology

- High-level approach description
- System architecture diagram
- Key components and their interactions
- Technology stack justification

4.3 Implementation Plan

- Timeline with milestones
- Task breakdown and assignments
- Dependencies and risk mitigation

4.4 Research Component (if applicable)

4.4.1 Research Questions

List 2-4 specific, measurable research questions.

4.4.2 Related Work

Create a comparison table:

4.4.3 Implementations for Comparison

- List papers with available implementations
- Note GitHub repositories or other code sources
- Plan for reproducibility if code is unavailable

Paper/Tool	Approach	Key Contribution	Our Difference
Paper 1	Brief description	What they did	What we add
Paper 2	Brief description	What they did	What we add
...			

Table 2: Related work comparison

4.4.4 Dataset Selection

- Chosen dataset(s) and justification
- Data statistics (size, composition, etc.)
- Preprocessing requirements
- Train/validation/test split strategy

4.5 Evaluation Plan

- Metrics to be measured
- Baseline methods for comparison
- Experimental setup and parameters
- Statistical significance testing plan
- Expected outcomes and success criteria

5 Submission Guidelines

5.1 Project Plan (Due: January 25, 2026)

- Submit as PDF via course management system
- Include all sections from template above
- **Maximum 2 pages** (excluding references)
- For research component: Use **Google Scholar** to identify related work
- Prioritize papers from **high-quality security conferences** (A* or A ranking):
  - Examples: IEEE S&P, USENIX Security, CCS, NDSS, BlackHat, DEF CON
  - Check conference rankings at CORE or similar sources

5.2 Final Submission (Due: March 22, 2026)

- GitHub repository with:
  - Complete source code
  - Dockerfile and docker-compose.yml
  - start.sh script
  - Comprehensive README
  - Documentation

- Final report (PDF):
  - **Maximum 10 pages** (excluding references)
  - Technical implementation details
  - Evaluation results
  - Research findings (if applicable)
  - Conclusions and future work
- Presentation slides (PDF or PPTX)
- Optional: Draft manuscript (if results warrant publication)

5.3 Research Component Deadlines

- **Note:** For students pursuing the research component, deadlines may vary depending on conference submission cycles
- Students with research components can schedule **regular supervision meetings** to discuss:
  - Progress on research questions
  - Experimental design and results
  - Paper writing and submission strategies
  - Conference deadline alignment
- Contact instructor to arrange supervision schedule
- Conference-aligned deadlines will be negotiated on a case-by-case basis

6 Evaluation Criteria

Your project will be evaluated based on:

1. **Implementation Quality (30%)**
   - Code quality and organization
   - Proper use of frameworks and tools
   - Docker containerization
   - Documentation completeness
2. **Functionality (60%)**
   - System works as specified
   - Tool design quality (for Project 1)
   - Output quality (YARA rules, signatures, reports)
   - Practical utility
3. **Presentation & Documentation (10%)**
   - Clarity of final report
   - Quality of presentation
   - README and user documentation

4. **Research Component (Bonus +30%)**
    - Thoroughness of literature review
    - Quality of experimental design
    - Rigor of evaluation
    - Insights from results

7 Getting Started

7.1 Recommended Timeline

Week	**Activities**
1 (Jan 13-19)	Project selection, team formation, initial research
2 (Jan 20-26)	Complete project plan, finalize methodology
3-4 (Jan 27-Feb 9)	Setup development environment, implement core components
5-6 (Feb 10-23)	Continue implementation, begin testing
7-8 (Feb 24-Mar 9)	Complete implementation, run evaluations
9 (Mar 10-16)	Finalize results, prepare report and presentation
10 (Mar 17-22)	Final polishing, submission

7.2 Resources

- Google Scholar for literature review
- CORE Conference Rankings: `http://portal.core.edu.au/conf-ranks/`

8 Additional Notes

- Start early! These projects require significant time investment
- For group projects, ensure equitable work distribution
- Regular commits to GitHub will help track progress
- Don't hesitate to reach out for guidance or clarification
- The research component is optional but highly encouraged
- Publication-worthy results will receive additional support from instructors
- **Research component students:** Schedule regular supervision meetings to ensure progress and alignment with conference deadlines
- **Conference submissions:** If pursuing publication, discuss target conferences early to align project timeline

Good luck with your projects!