FSCT 8561: Security Applications – Midterm Exam

Course: FSCT 8561 – Security Applications

Instructor: Dr. Maryam R. Aliabadi

Duration: 72 Hours (3 Days)

Format: Open Book / Individual Work

AI Policy & Academic Integrity

This is an open-resource exam. You may consult any online materials, including AI tools, as reference resources. However, you may not submit AI-generated architecture, code, or written justifications as your own work. You are fully responsible for every design decision and line of code in your submission. Submissions will be evaluated for internal coherence across design, protocol, implementation, and analysis. Inconsistent or non-defensible work may be subject to academic integrity review.

The Scenario: Secure Remote Diagnostic Service (SRDS)

You have been hired as a Security Architect for a firm that requires a Secure Remote Diagnostic Service (SRDS). The service allows administrators to connect to a server, authenticate, and run network health checks (scans) on remote nodes.

Your goal: Design and implement a system that remains secure even when the communication channel is monitored by an adversary.

DAY 1: System Design & Threat Modeling (40%)

Part A: Architecture Design (20%)

Provide a high-level overview of your system. You must submit a System Architecture Diagram (hand-drawn or digital) and a description of the following:

- *Client & Server Components*: How they establish a connection using Python Sockets.

- *The Gateway Logic:* Explain if your system uses a direct connection or an intermediary (like a WAP Gateway logic).
- *Justification:* Why did you choose your specific communication model (e.g., Stateful vs. Stateless)?

Part B: Adversarial Threat Model (20%)

Develop a structured threat model using the following table format. You must identify at least five unique threats, including at least one related to each topic covered in Labs 1–5 (Sockets, Scanning, MFA, Scapy, and Web).

Asset	Threat	Attack Vector	Impact	Mitigation
Ex: Admin Credentials	*Replay Attack*	*Sniffing Socket traffic*	*Unauthorized Access*	*Challenge-Response Nonce*

DAY 2: Protocol & Component Specification (30%)

Part C: Secure Protocol Design (15%)

Define your custom application-layer protocol. Your specification must include:

1. *Message Format:* JSON, TLV, Delimited,..
2. *The Authentication Handshake*: Explicitly incorporate Challenge-Response logic
3. *Adversarial Defense:* How does your protocol prevent an attacker from re-injecting a captured valid packet?

Part D: Component-Level Design (15%)

Detail the **Inputs, Outputs**, and **Security Controls** for these modules:

- *MFA Module:* Combining hashlib and PyOTP.
- *Recon Module:* Utilizing python-nmap.
- *Detection Module:* Utilizing Scapy.
- *Negative Space Decision:* State one piece of data you intentionally chose NOT to collect and explain how this reduces your attack surface.

DAY 3: Constrained Implementation & Analysis (30%)

Part E: Minimal Implementation (15%)

Provide a Python script that implements the Core Security Logic of your system. This should focus on the Authentication Handshake or the Attack Detection logic.

- **Constraint:** Use only the libraries covered in class (socket, hashlib, pyotp, scapy, requests, nmap).
- **Requirement:** Use comments to explain the *security purpose* of your code, not just the function.

Part F: Security Analysis & Reflection (15%)

Answer the following questions based *only* on your specific design:

1. *Fragility:* If an attacker targets your system with a **SYN Flood**, which component fails first?
2. *Bypass:* How could an attacker exploit the "Possession Factor" (MFA) in your design?
3. *Encryption Termination:* If your traffic passes through a translation gateway (WAP Gateway), where is your data most vulnerable to "sniffing"?

Submission Requirements

Submit **one single PDF** containing all diagrams, tables, link to code snippets, and reflections.

- *Filename:* FSCT8561-Midterm-FirstName-LastName-StudentNumber.pdf
- *Deadline:* 72 hours from the release time.

Grading Rubric

Category	Weight	Evaluation Criteria	Excellent (Full Marks)	Partial (Half Marks)	Poor / Missing (0)
System Architecture & Design	25%	Coherence, realism, justification, clarity of architecture diagram	Architecture diagram is complete and clear; all components described; rationale for sockets vs HTTP, stateful/stateless, push/pull is explicit and sound	Some components or rationale missing; minor inconsistencies between design and explanation	Diagram incomplete or missing; major components missing; rationale absent or contradictory
Threat Modeling	20%	Coverage, correctness, mitigation quality	Structured threat table includes all required threats (authentication, injection, replay, info leakage, scanning); mitigations are correct and justified; includes at least one threat per course topic	Partial coverage; some threats missing; mitigations partially justified	Missing table or very limited coverage; threats and mitigations incorrect or absent
Protocol Design	15%	Security reasoning, clarity, robustness	Protocol clearly defined; message format, authentication handshake, error handling, and session termination fully specified; replay, unauthorized commands, malformed input addressed	Some protocol elements incomplete; partial justification; minor security weaknesses	Protocol missing or poorly defined; no security reasoning
Component-Level Specification	15%	Inputs, outputs, failure modes, security controls	All required components described with inputs, outputs, failure modes, and security controls; includes explicit justification of excluded data	Partial coverage; some components missing or inadequately justified	Components missing; security controls absent; no explanation of excluded data
Minimal Implementation	15%	Alignment with design,	Code ≤200 lines; implements core security logic; matches	Minor deviations from design; code partially	Code missing or unrelated; major deviations from

Category	Weight	Evaluation Criteria	Excellent (Full Marks)	Partial (Half Marks)	Poor / Missing (0)
		correctness, clarity	design; comments explain **why** controls exist; no major errors	implements logic; comments incomplete	design; insufficient or missing comments
Reflection & Security Analysis	15%	Depth, insight, honesty	Thoughtful analysis of system fragility, attack vectors, defenses, and scalability; references own design decisions	Partial reflection; some points covered superficially	Reflection missing or generic; does not reference own design
Consistency & Internal Coherence	10%	Design ↔ code ↔ analysis alignment	All parts (design, protocol, code, reflection) are fully consistent	Minor inconsistencies between design, implementation, and analysis	Major inconsistencies; code contradicts design; reflection does not match submission

---

Good luck!