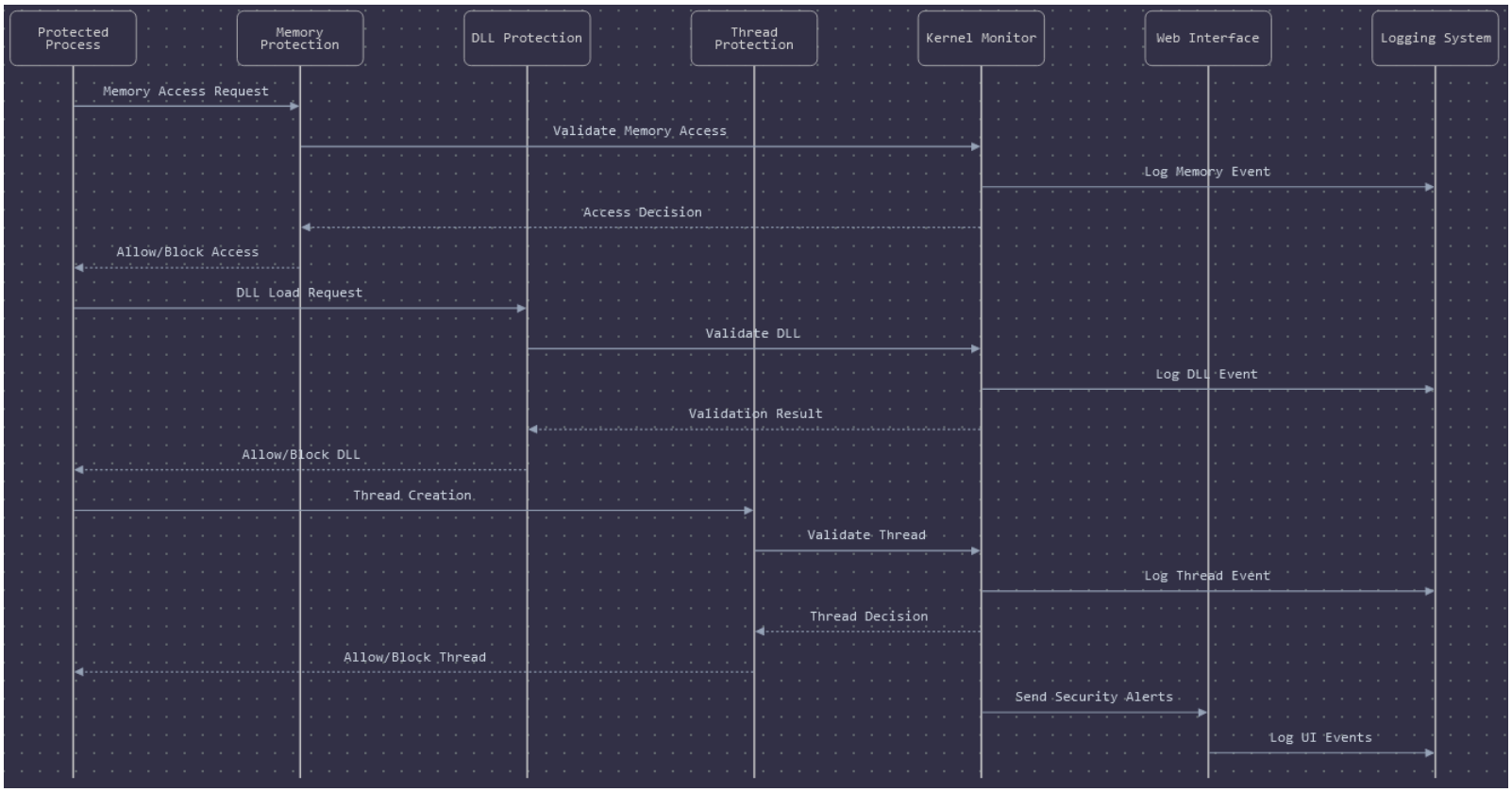
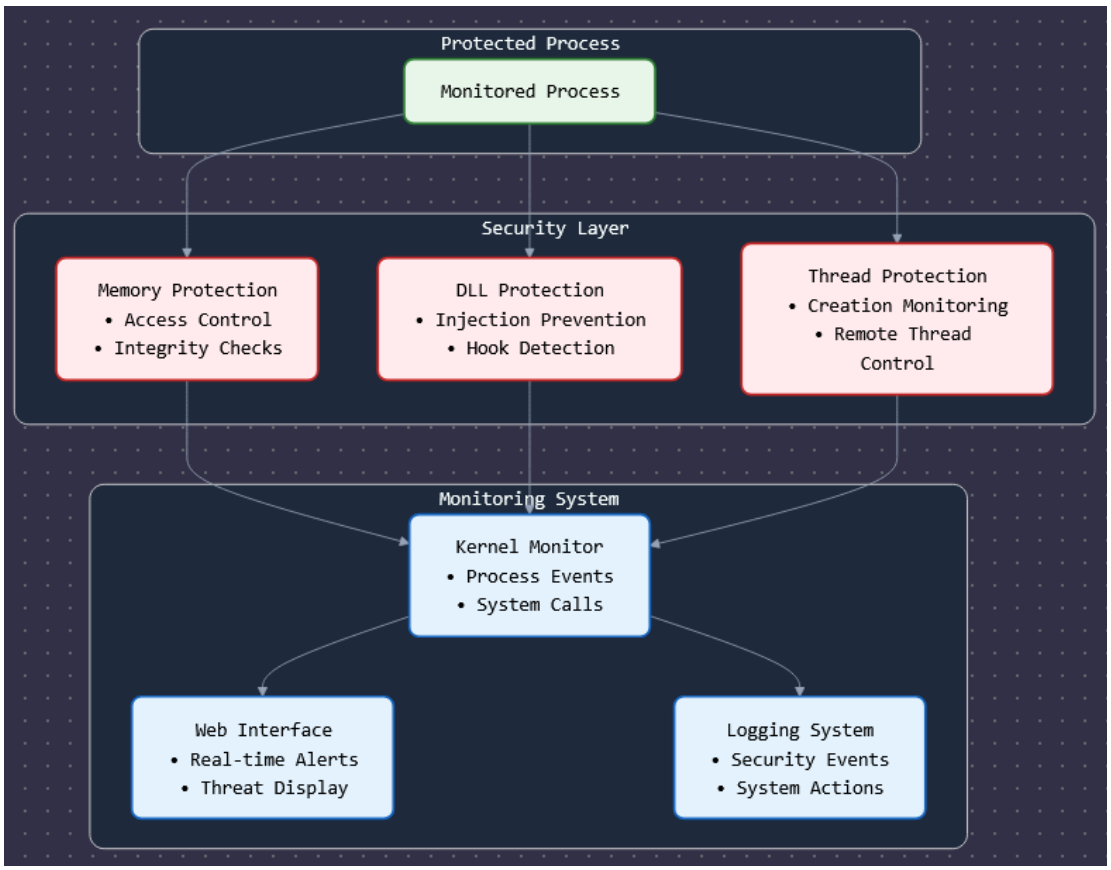
Windows Process Protector Test Plan

By: Bartosz Kawalkowski and Harshil Patel





Part 1: Overall Test Plan / Overview

The testing strategy for the Windows Process Protector project involves a multi-phase approach, incorporating unit, integration, functional, and performance testing. First, we will start by testing individual components of the system in isolation to verify their correctness. This includes unit testing for core security mechanisms such as memory protection, code integrity verification, and DLL injection prevention**.** Once unit tests confirm the reliability of individual components, integration testing will validate the interaction between different subsystems, including the kernel-level process monitoring system and the web-based dashboard**.** Functional testing will ensure that the system correctly identifies unauthorized process modifications and alerts users as expected. Finally, performance testing will measure system efficiency, ensuring that security features do not significantly impact process performance. Boundary tests will evaluate system behavior under stress conditions, such as high memory usage or many monitored processes. The overall goal is to ensure robustness, security, and usability across different use cases, including gaming anti-cheat protection and enterprise security**.**

Part 2: Test Case Descriptions

Test Case 1	**WPPT1**
Purpose of Test	Verify unauthorized memory access detection
Description	Simulate an unauthorized attempt to access protected process memory.
Inputs	Inject a DLL into a protected process
Expected Output	Alert generated; process access blocked
Case Type	Abnormal
Test Type	Blackbox
Functional/ Performance	Functional
Unit/Integration	Unit

Test Case 2	**WPPT2**
Purpose of Test	Check code integrity verification
Description	Modify a protected process's executable code
Inputs	Modify process memory dynamically
Expected Output	Alert generated; process terminated
Case Type	Abnormal
Test Type	Blackbox
Functional/Performance	Functional
Unit/Integration	Unit

Test Case 3	**WPPT3**
Purpose of Test	Monitor unauthorized remote thread creation
Description	Test system response to unauthorized thread injection
Inputs	Execute remote thread into process
Expected Output	Alert generated; thread blocked
Case Type	Abnormal
Test Type	Blackbox
Functional/Performance	Functional
Unit/Integration	Unit

Test Case 4	**WPPT4**
Purpose of Test	Verify trust list functionality
Description	Ensure trusted processes are not flagged as threats
Inputs	Add process to trust list
Expected Output	Process not blocked
Case Type	Normal
Test Type	Blackbox
Functional/Performance	Functional
Unit/Integration	Unit

Test Case 5	**WPPT5**
Purpose of Test	Validate DLL injection blocking
Description	Attempt to inject a DLL into a monitored process
Inputs	Run an injector script
Expected Output	DLL injection prevented
Case Type	Abnormal
Test Type	Blackbox
Functional/ Performance	Functional
Unit/Integration	Unit

Test Case 6	**WPPT6**
Purpose of Test	Ensure web interface displays real-time alerts
Description	Test the web UI for accuracy in displaying detected threats
Inputs	Generate a fake process attack
Expected Output	UI displays correct threat details
Case Type	Normal
Test Type	Blackbox
Functional/ Performance	Functional
Unit/Integration	Integration

Test Case 7	**WPPT7**
Purpose of Test	Evaluate performance impact on system
Description	Measure CPU and memory usage of monitored processes
Inputs	Run multiple applications
Expected Output	System maintains optimal performance
Case Type	Normal
Test Type	Blackbox
Functional/ Performance	Performance
Unit/Integration	Integration

Test Case 8	**WPPT8**
Purpose of Test	Test system startup and initialization
Description	Verify that system initializes correctly on boot
Inputs	Restart system with monitoring enabled
Expected Output	Monitoring starts automatically
Case Type	Normal
Test Type	Whitebox
Functional/ Performance	Functional
Unit/Integration	Integration

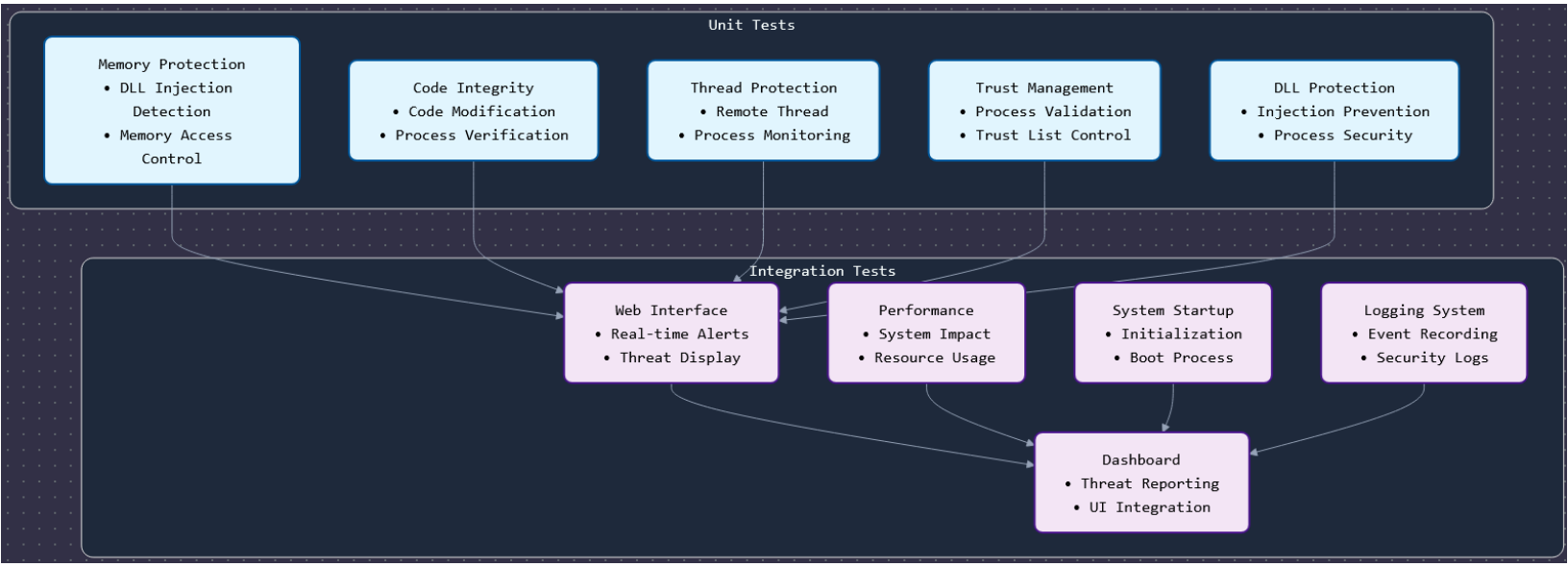
Test Case 9	**WPPT9**
Purpose of Test	Check system logging accuracy
Description	Validate whether all security events are logged correctly
Inputs	Trigger various process attacks
Expected Output	Logs contain accurate records
Case Type	Normal
Test Type	Whitebox
Functional/ Performance	Functional
Unit/Integration	Integration

Test Case 10	**WPPT10**
Purpose of Test	Validate dashboard threat reporting
Description	Ensure all threats detected are reflected in the UI
Inputs	Simulate multiple threat scenarios
Expected Output	UI displays all detected threats correctly
Case Type	Normal
Test Type	Blackbox
Functional/	Functional

Performance	
Unit/Integration	Integration

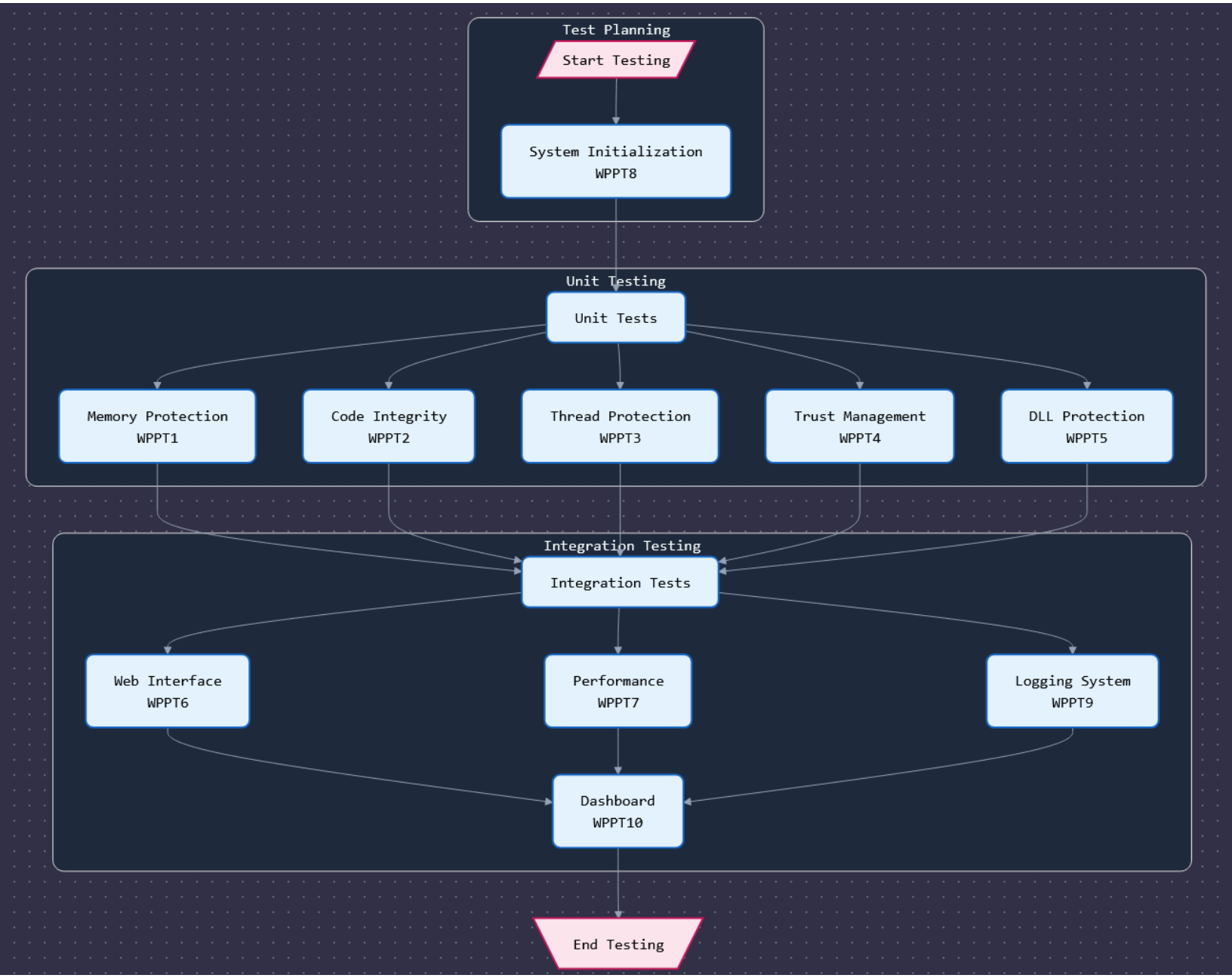
Part 3: Test Case Matrix

Test Case	Case Type	Test Type	Functional/Performance	Unit/Integration
WTTP1	Abnormal	Blackbox	Functional	Unit
WTTP2	Abnormal	Blackbox	Functional	Unit
WTTP3	Abnormal	Blackbox	Functional	Unit
WTTP4	Normal	Blackbox	Functional	Unit
WTTP5	Abnormal	Blackbox	Functional	Unit
WTTP6	Normal	Blackbox	Functional	Integration
WTTP7	Normal	Blackbox	Performance	Integration
WTTP8	Normal	Whitebox	Functional	Integration
WTTP9	Normal	Whitebox	Functional	Integration
WTTP10	Normal	Blackbox	Functional	Integration



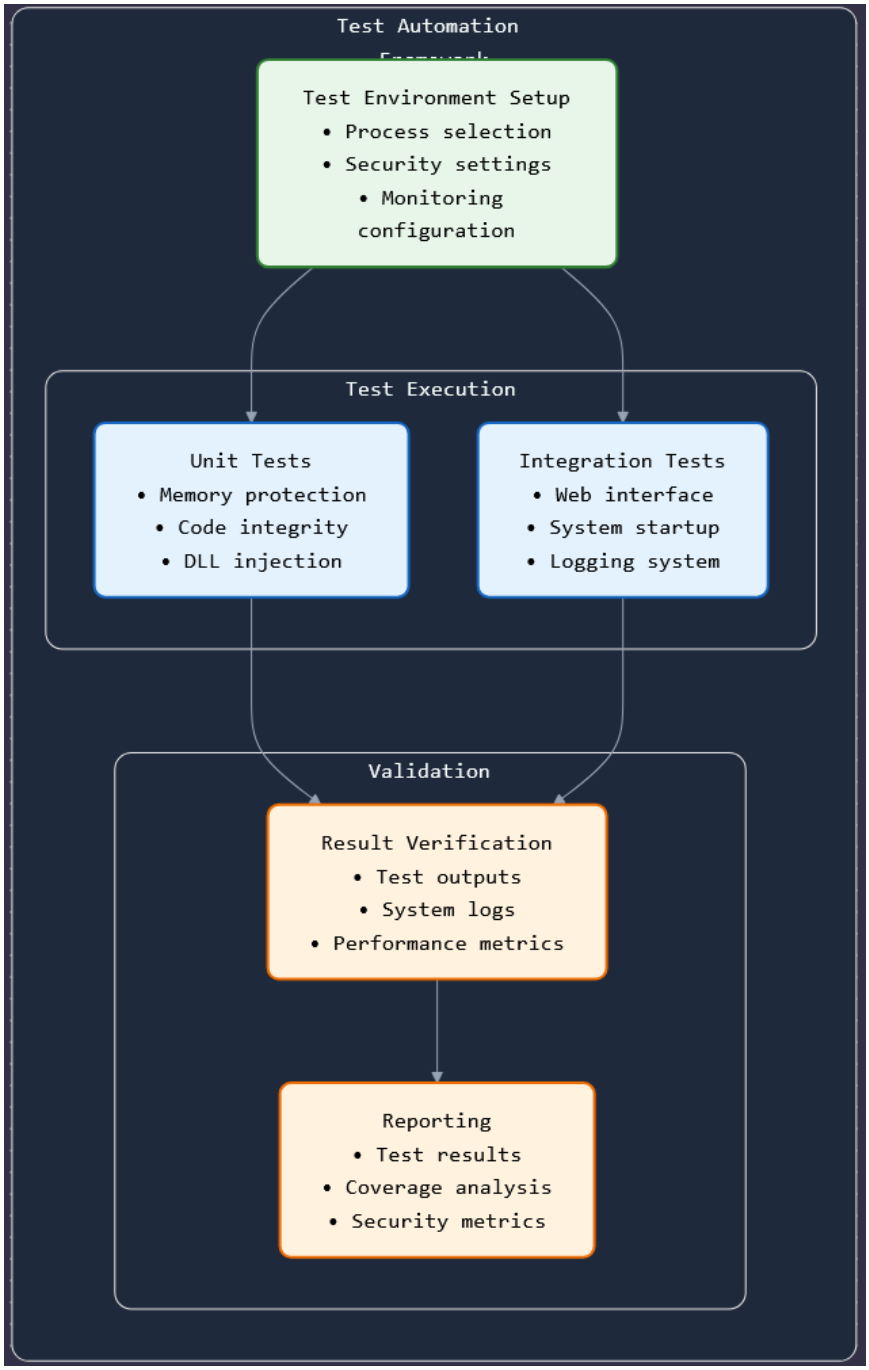
The diagram above shows the complete test case structure where blue boxes represent unit tests focusing on individual security features, purple boxes represent integration tests covering system-level functionality, arrows indicate dependencies, showing how unit test results feed into integration testing. All test ultimately contribute to the dashboard integration (WTTP10).

Testing Workflow Diagram:



This workflow diagram illustrates the sequential nature of the testing process, where testing begins with system initialization (WTTP8). All unit tests (WTTP1-5) must complete successfully before integration testing begins. Integration tests (WTTP6-10) run in parallel once unit tests are verified. The dashboard test (WTTP10) servers as the final integration point.

Test Automation Framework Diagram:



The diagram above shows the automation framework structure where green components (setup) handle test environment initialization, blue components (execution) run the actual tests, & orange components (validation) verify results and generate testing reports. This framework integrates with our existing test cases by automating execution of WTTP1-10 while adding systematic verification and reporting capabilities.