Green Pace

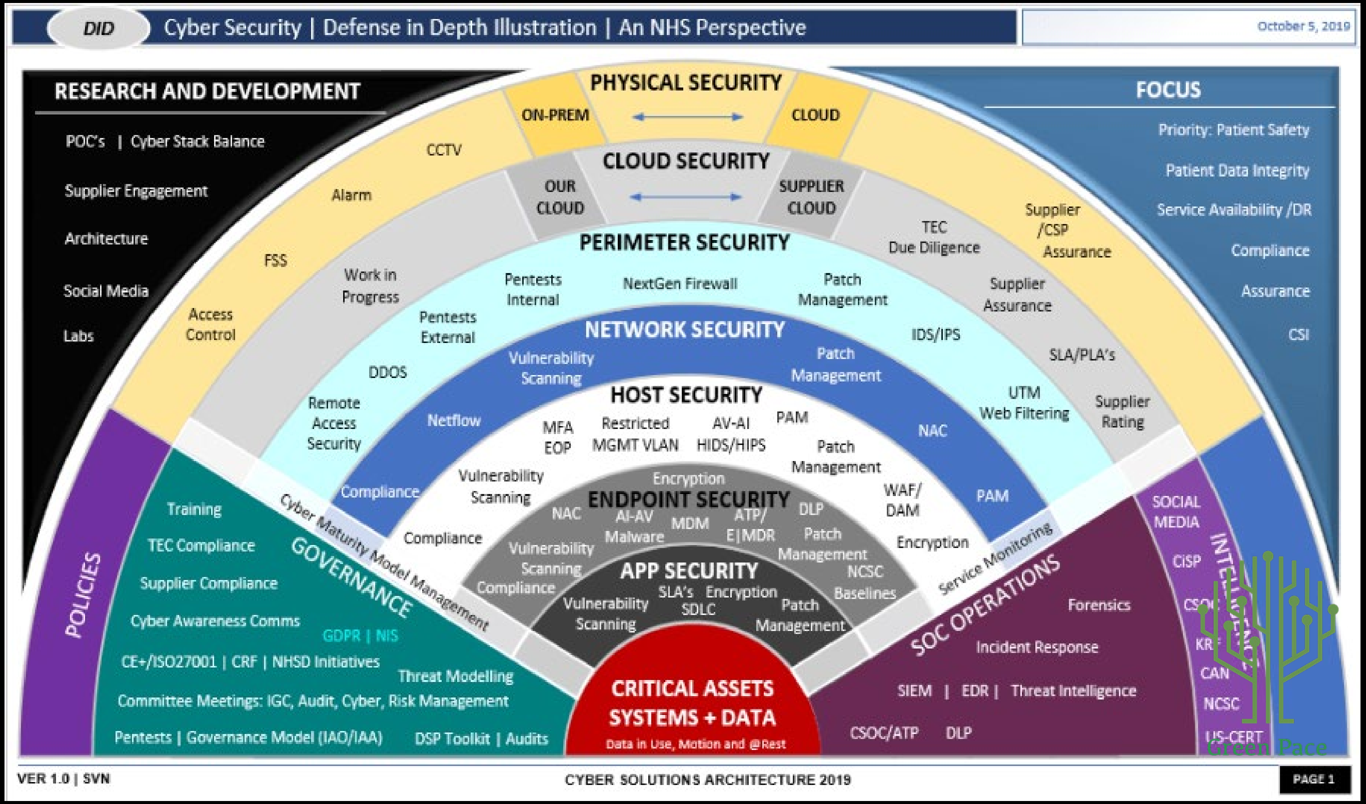
Security Policy Presentation
Developer: *Cortney Messinger*

CS 405 – *Project Two*
Security Policy Implementation
02/15/2026

OVERVIEW: DEFENSE IN DEPTH

- Purpose of the Security Policy
- Standardized Secure Coding Practices
- Defense-in-Depth Strategy
- Integration into DevSecOps

Needed to standardize secure practices as the development team scales.



THREATS MATRIX

- Injection & Memory = Highest Risk
- Arithmetic & Type = High but lower exploitability
- Assertions & Randomness = Medium
- Level 5 items can cause breach/RCE or systemic outage

Impact \ Likelihood	**Low**	**Medium**	**High (Likely)**
High Impact	—	**STD-001** (Data Type) **STD-002** (Overflow) **STD-003** (String) **STD-007** (Exceptions) **STD-008** (Concurrency) – Level 5 (higher risk) **STD-009** (I/O Format Strings)	**STD-004** (SQL Injection) **STD-005** (Memory Protection)
Medium Impact	—	**STD-006** (Assertions) **STD-010** (Secure Randomness)	—
Low Impact	—	—	—

Impact = Severity | Likelihood = Probability of exploit | Level = Overall risk (1–5)

Fix order: STD-004/005 → STD-008 → remaining Level 4 → Level 3

Green Pace

10 PRINCIPLES

1. Validate Input Data → STD-001/002/003/004/009
2. Heed Compiler Warnings
3. Architect for Security
4. Keep It Simple
5. Default Deny
6. Least Privilege
7. Sanitize Data
8. Defense in Depth
9. Quality Assurance
10. Secure Coding Standards

Each coding standard maps to one or more principles.
Grounded in SEI CERT Secure Coding Framework

Green Pace

CODING STANDARDS

Priority Order:

1. SQL Injection (STD-004)
2. Memory Protection (STD-005)
3. Concurrency (STD-008)
4. String Safety (STD-003)
5. Integer Overflow (STD-002)
6. Data Type Safety (STD-001)
7. I/O Safety (STD-009)
8. Exception Safety (STD-007)
9. Assertions (STD-006)
10. Secure Randomness (STD-010)

Ranked by exploitability + impact + remediation cost (risk level)

Green Pace

ENCRYPTION POLICIES

- Encryption at Rest → AES-256
- Encryption in Flight → TLS 1.3
- Encryption in Use → Secure memory / protected runtime
- Key management required → Keys stored/managed via approved key management process

Green Pace

TRIPLE-A POLICIES

Authentication
- Strong identity verification

Authorization
- Role-Based Access Control

Accounting
- Logging & auditing of:
  - Logins
  - Database changes
  - Access levels
  - File access
  - Logs are tamper-resistant and access-controlled

Green Pace

Unit Testing

Integer Overflow or SQL Injection

- Test 1 Does the system detect integer overflow?
- Test 2 Does valid arithmetic execute correctly?
- Test 3 Are boundary values handled safely?
- Test 4 Does invalid negative input fail safely?

Green Pace

Test 1 Does the system detect integer overflow?

- Input: Max unsigned int + 1
- Expected: Exception thrown
- Result: PASS
- Next step: How to take it further

Test 2 Does valid arithmetic execute correctly?

- Input: Safe addition
- Expected: No exception
- Result: PASS
- Next step: add fuzz/boundary cases

Test 3 Are boundary values handled safely?

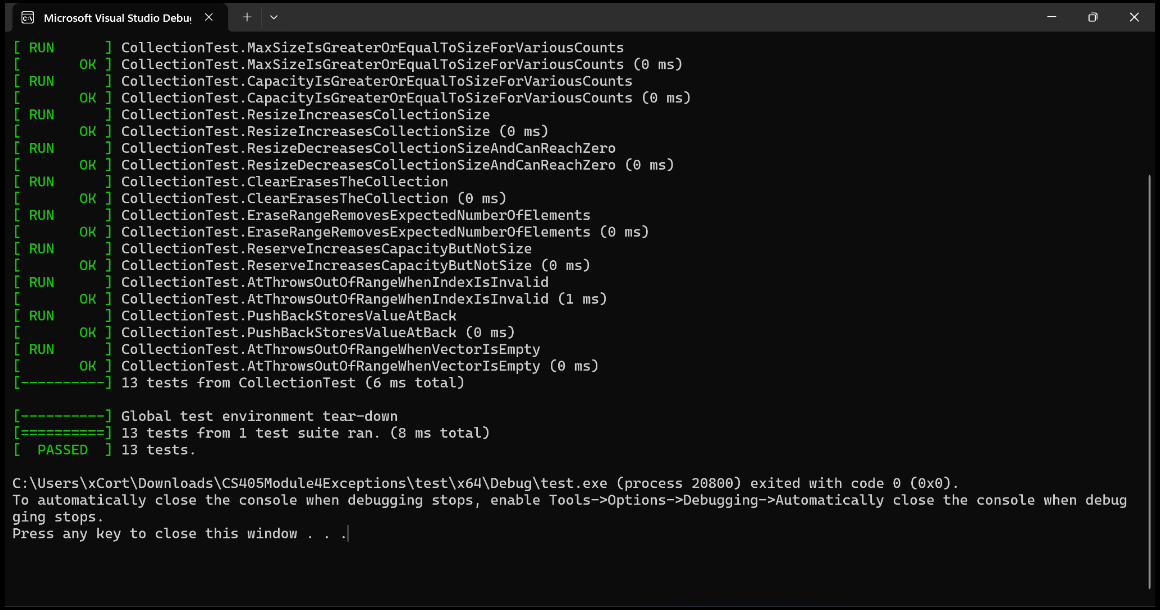
- Input: Maximum – 1
- Expected: Success
- Result: PASS
- Next step: test multiple integer types

Test 4 Does invalid negative input fail safely?

- Input: Negative to unsigned
- Expected: std::invalid_argument
- Result: PASS
- Next step: integrate into CI pipeline gate

These tests were implemented using the Microsoft C++ Unit Testing Framework within Visual Studio.

Unit Test Execution Results – Visual Studio



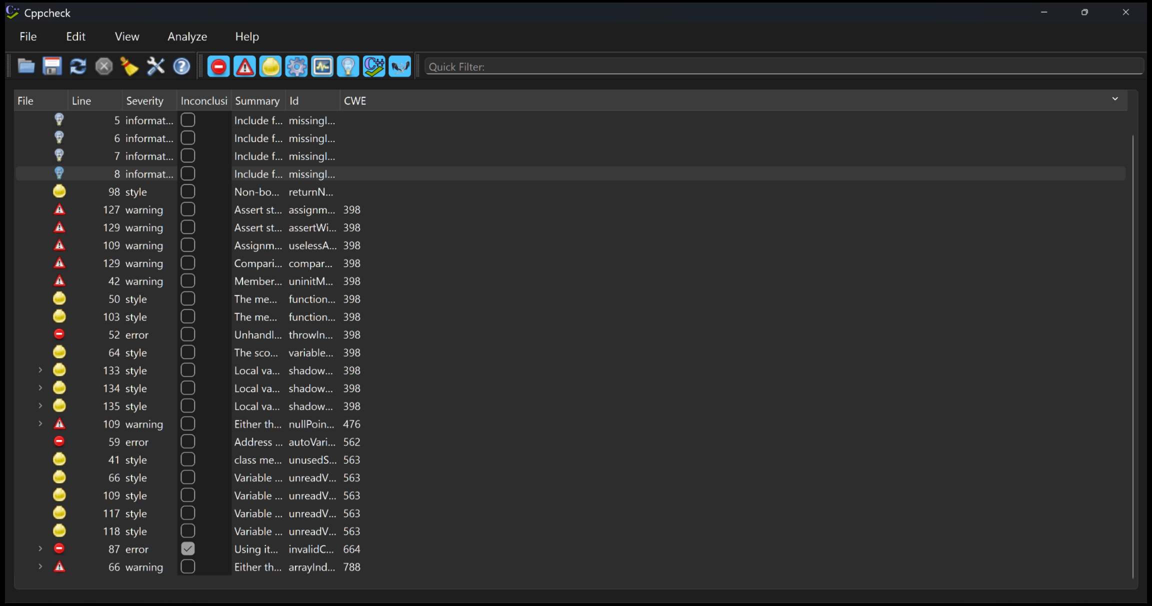
Microsoft Visual Studio Debu

```
[ RUN      ] CollectionTest.MaxSizeIsGreaterOrEqualToSizeForVariousCounts
[       OK ] CollectionTest.MaxSizeIsGreaterOrEqualToSizeForVariousCounts (0 ms)
[ RUN      ] CollectionTest.CapacityIsGreaterOrEqualToSizeForVariousCounts
[       OK ] CollectionTest.CapacityIsGreaterOrEqualToSizeForVariousCounts (0 ms)
[ RUN      ] CollectionTest.ResizeIncreasesCollectionSize
[       OK ] CollectionTest.ResizeIncreasesCollectionSize (0 ms)
[ RUN      ] CollectionTest.ResizeDecreasesCollectionSizeAndCanReachZero
[       OK ] CollectionTest.ResizeDecreasesCollectionSizeAndCanReachZero (0 ms)
[ RUN      ] CollectionTest.ClearErasesTheCollection
[       OK ] CollectionTest.ClearErasesTheCollection (0 ms)
[ RUN      ] CollectionTest.EraseRangeRemovesExpectedNumberOfElements
[       OK ] CollectionTest.EraseRangeRemovesExpectedNumberOfElements (0 ms)
[ RUN      ] CollectionTest.ReserveIncreasesCapacityButNotSize
[       OK ] CollectionTest.ReserveIncreasesCapacityButNotSize (0 ms)
[ RUN      ] CollectionTest.AtThrowsOutOfRangeWhenIndexIsInvalid
[       OK ] CollectionTest.AtThrowsOutOfRangeWhenIndexIsInvalid (1 ms)
[ RUN      ] CollectionTest.PushBackStoresValueAtBack
[       OK ] CollectionTest.PushBackStoresValueAtBack (0 ms)
[ RUN      ] CollectionTest.AtThrowsOutOfRangeWhenVectorIsEmpty
[       OK ] CollectionTest.AtThrowsOutOfRangeWhenVectorIsEmpty (0 ms)
[----------] 13 tests from CollectionTest (6 ms total)

[----------] Global test environment tear-down
[==========] 13 tests from 1 test suite ran. (8 ms total)
[  PASSED  ] 13 tests.

C:\Users\xCort\Downloads\CS405Module4Exceptions\test\x64\Debug\test.exe (process 20800) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debug
ging stops.
Press any key to close this window . . .
```

All four positive and negative tests executed successfully.

- All 13 security validation tests executed successfully
- Negative tests verified exception handling for unsafe inputs

Automation / External Detection



Cppcheck

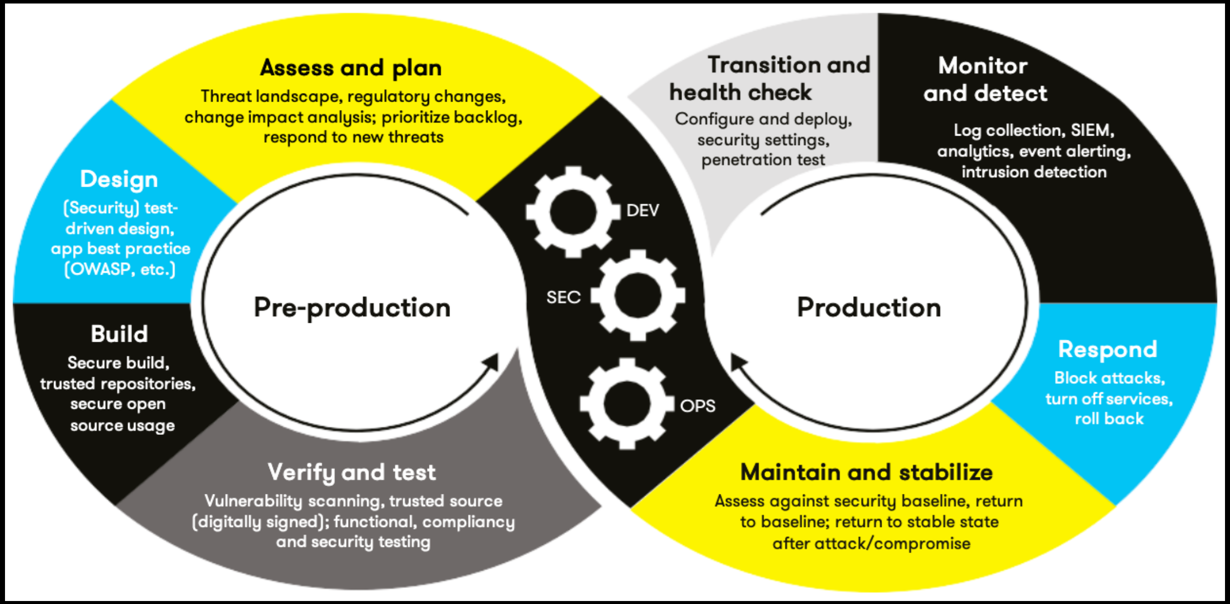
File	Line	Severity	Inconclusi	Summary	Id	CWE
	5	informat...	☐	Include f...	missingI...	
	6	informat...	☐	Include f...	missingI...	
	7	informat...	☐	Include f...	missingI...	
	8	informat...	☐	Include f...	missingI...	
	98	style	☐	Non-bo...	returnN...	
	127	warning	☐	Assert st...	assignm...	398
	129	warning	☐	Assert st...	assertWi...	398
	109	warning	☐	Assignm...	uselessA...	398
	129	warning	☐	Compari...	compar...	398
	42	warning	☐	Member...	uninitM...	398
	50	style	☐	The me...	function...	398
	103	style	☐	The me...	function...	398
	52	error	☐	Unhandl...	throwIn...	398
	64	style	☐	The sco...	variable...	398
›	133	style	☐	Local va...	shadow...	398
›	134	style	☐	Local va...	shadow...	398
›	135	style	☐	Local va...	shadow...	398
›	109	warning	☐	Either th...	nullPoin...	476
	59	error	☐	Address ...	autoVari...	562
	41	style	☐	class me...	unusedS...	563
	66	style	☐	Variable ...	unreadV...	563
	109	style	☐	Variable ...	unreadV...	563
	117	style	☐	Variable ...	unreadV...	563
	118	style	☐	Variable ...	unreadV...	563
›	87	error	✓	Using it...	invalidC...	664
›	66	warning	☐	Either th...	arrayInd...	788

These findings were analyzed and either remediated or documented as non-security-critical style issues

Cppcheck Static Analysis – Detected null pointer, invalid conversion, and memory safety risks.

- Cppcheck detected potential null pointer, uninitialized variable, and invalid conversion risks
- Findings demonstrate automated detection of memory safety and data validation issues before runtime

AUTOMATION SUMMARY


Assess and plan
Threat landscape, regulatory changes, change impact analysis; prioritize backlog, respond to new threats
Design
(Security) test-driven design, app best practice (OWASP, etc.)
Build
Secure build, trusted repositories, secure open source usage
Verify and test
Vulnerability scanning, trusted source (digitally signed); functional, compliancy and security testing
Pre-production
DEV
SEC
OPS
Transition and health check
Configure and deploy, security settings, penetration test
Monitor and detect
Log collection, SIEM, analytics, event alerting, intrusion detection
Production
Respond
Block attacks, turn off services, roll back
Maintain and stabilize
Assess against security baseline, return to baseline; return to stable state after attack/compromise

Static analysis detected narrowing conversion and arithmetic overflow risks before runtime.

Compiler warnings treated as errors during Create stage

Green Pace

TOOLS

- Plan → Create → Verify → Pre-Production → Release → Detect → Respond → Adapt
- Security Tools:
  - Static analysis during Create
  - Unit tests during Verify
  - Security scans in Pre-Production
  - Monitoring during Detect

Green Pace

RISKS AND BENEFITS

- Act Now:
  - Reduce breach risk
  - Improve compliance
  - Lower long-term cost
  - Reduces technical debt and audit exposure

- Wait:
  - Increased exposure
  - Legal risk
  - Higher remediation cost

Green Pace

RECOMMENDATIONS

- Recommendations:
  - Continuous training
  - Annual policy review
  - Mandatory secure code reviews
  - CI gate blocks merge on Level 5 findings
  - Formal exception approval via CISO

Green Pace

CONCLUSIONS

- Continue enforcement of SEI CERT C++ Secure Coding Standard
- Require automated static analysis in all builds
- Mandate secure code reviews before merge
- Implement annual security training for developers
- Conduct annual policy review and risk reassessment
- Expand monitoring and incident response automation

Green Pace

REFERENCES

- OWASP Foundation. (2023). OWASP Top 10. https://owasp.org
- SEI. (2016). CERT C++ Coding Standard.
- NIST. (2022). Digital Signature Standard.

Green Pace