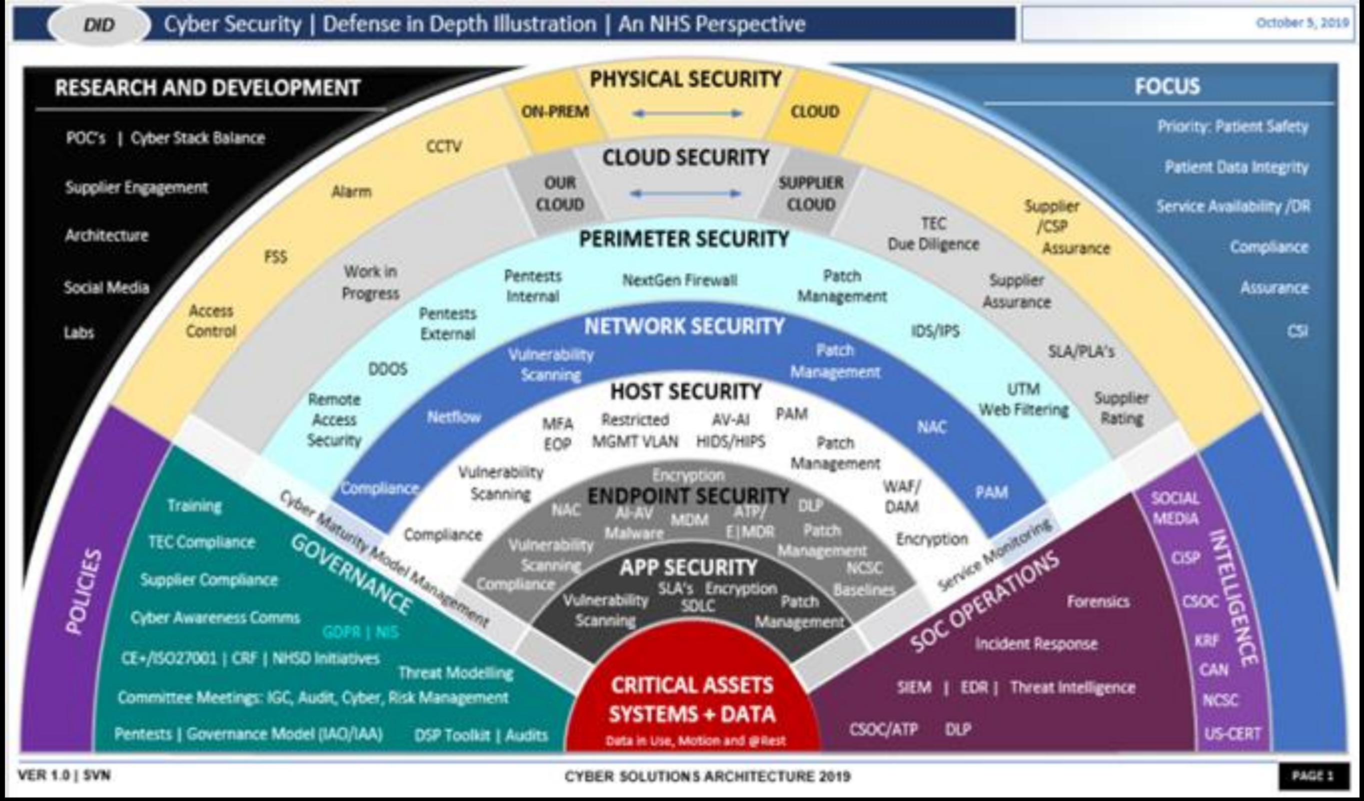
Green Pace

Security Policy Presentation
Developer: Eric Diaz

Green Pace

OVERVIEW: DEFENSE IN DEPTH

My security policies cover the whole Defense in Depth. We protect the users by ensuring no leakage happens and vulnerabilities are stopped.



Green Pace

THREATS MATRIX

Likely This problem will most likely arise	Priority One of the top issues we need to cover
Low priority Bottom of the list to focus on	Unlikely Will probably not come across this

Green Pace

10 PRINCIPLES

1. Validate Input Data
2. Heed Compiler Warings
3. Architect and design for Security Polices
4. Keep It Simple
5. Default Deny
6. Adhere to the Principle of Least Privilege
7. Sanitize Data Sent to Other Systems
8. Practice Defense in Depth
9. Use Effective Quality Assurance Techniques
10. Adopt a Secure Coding Standard

CODING STANDARDS

1. SQL Injection
2. Memory Protection
3. Assertions
4. Error Handling
5. String Correctness
6. Exceptions
7. Data Type
8. Data Value
9. Object Oriented Programming
10. Input Output

Green Pace

ENCRYPTION POLICIES

Encryption at Rest: When data is stored and isn't being used, it is still protected by encryption to ensure there is no unauthorized access.

Encryption in Flight: When data is being transmitted it has a encryption on it and without the right key it will remain unreadable.

Encryption in Use: When data is being used encryption is still active on it which protects it from unauthorized access or modifications during use

Green Pace

TRIPLE-A POLICIES

Authentication: Verifies the users account ensuring nobody can get in unless they have the right password and username.

Authorization: The specific resources and services a user can access and what they can do with them, based on their assigned permissions.

Accounting: Tracks and records user activity on the network.

Green Pace

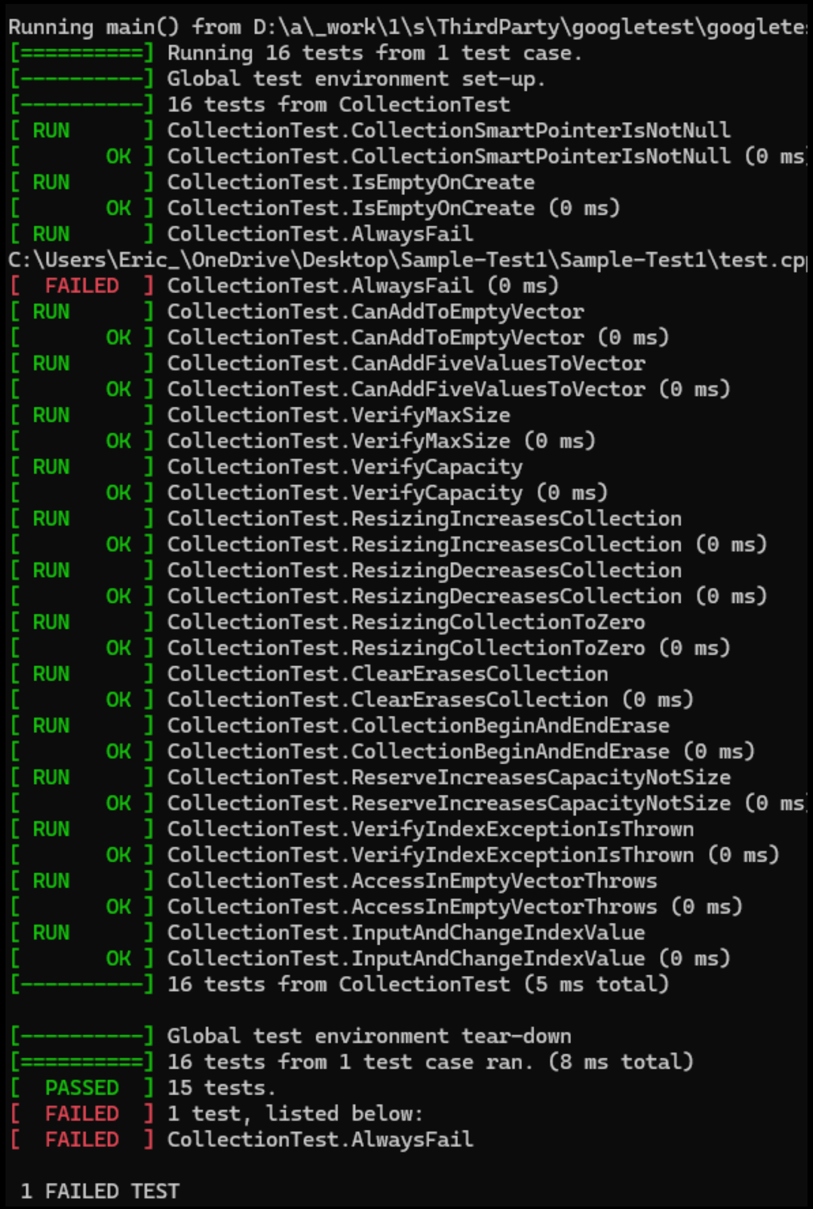
Unit Testing

Unit testing takes parts of the code and tests if that segment works as intended. This is very important because it finds bugs that can be fixed in early stages of development.

Green Pace

Exception and Assertion Testing

Exceptions help with notifying when a failure has happened. An assertion terminates the processing

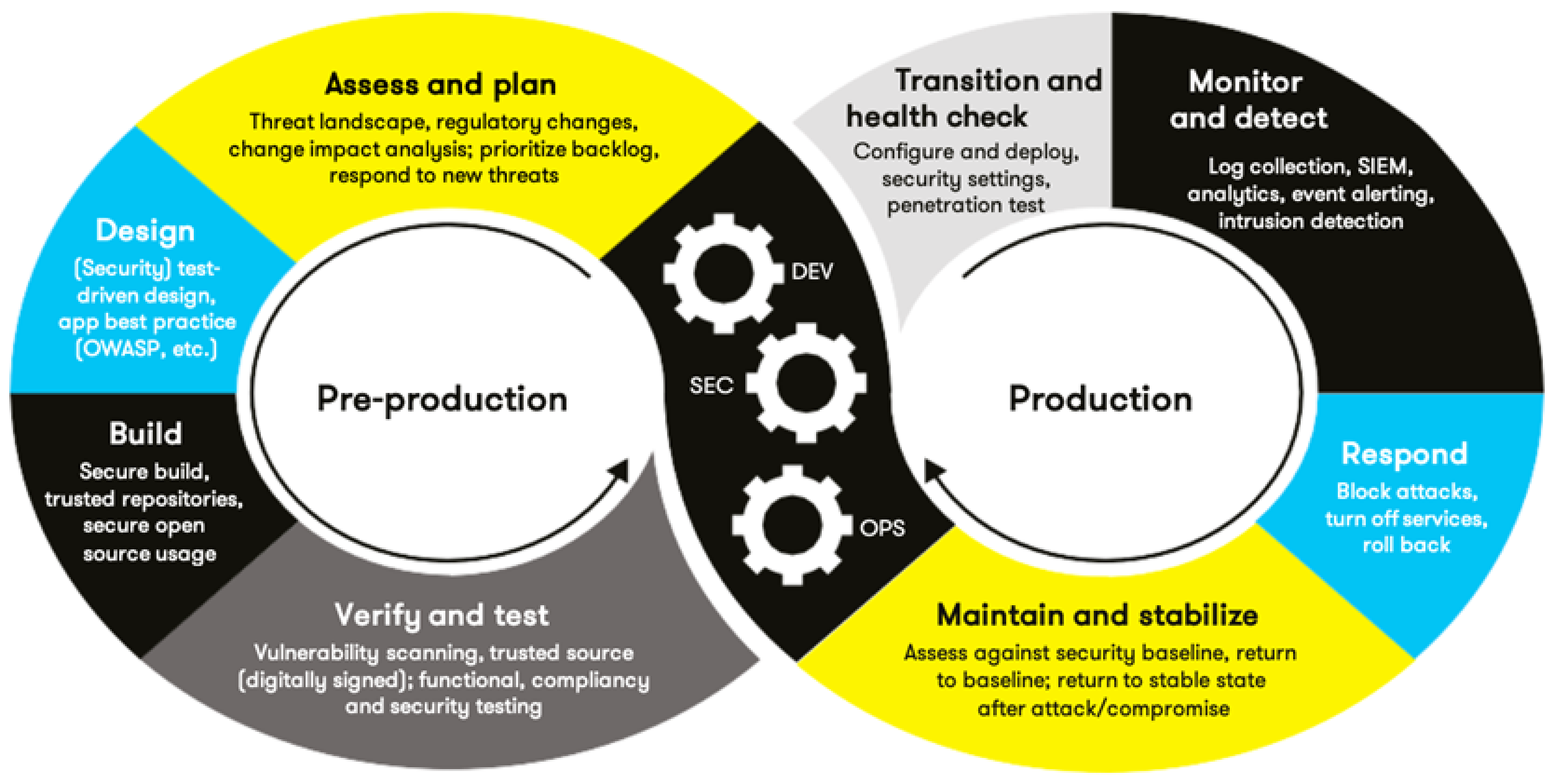


```
Running main() from D:\a\_work\1\s\ThirdParty\googletest\googlete
[==========] Running 16 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 16 tests from CollectionTest
[ RUN      ] CollectionTest.CollectionSmartPointerIsNotNull
[       OK ] CollectionTest.CollectionSmartPointerIsNotNull (0 ms
[ RUN      ] CollectionTest.IsEmptyOnCreate
[       OK ] CollectionTest.IsEmptyOnCreate (0 ms)
[ RUN      ] CollectionTest.AlwaysFail
C:\Users\Eric_\OneDrive\Desktop\Sample-Test1\Sample-Test1\test.cp
[  FAILED  ] CollectionTest.AlwaysFail (0 ms)
[ RUN      ] CollectionTest.CanAddToEmptyVector
[       OK ] CollectionTest.CanAddToEmptyVector (0 ms)
[ RUN      ] CollectionTest.CanAddFiveValuesToVector
[       OK ] CollectionTest.CanAddFiveValuesToVector (0 ms)
[ RUN      ] CollectionTest.VerifyMaxSize
[       OK ] CollectionTest.VerifyMaxSize (0 ms)
[ RUN      ] CollectionTest.VerifyCapacity
[       OK ] CollectionTest.VerifyCapacity (0 ms)
[ RUN      ] CollectionTest.ResizingIncreasesCollection
[       OK ] CollectionTest.ResizingIncreasesCollection (0 ms)
[ RUN      ] CollectionTest.ResizingDecreasesCollection
[       OK ] CollectionTest.ResizingDecreasesCollection (0 ms)
[ RUN      ] CollectionTest.ResizingCollectionToZero
[       OK ] CollectionTest.ResizingCollectionToZero (0 ms)
[ RUN      ] CollectionTest.ClearErasesCollection
[       OK ] CollectionTest.ClearErasesCollection (0 ms)
[ RUN      ] CollectionTest.CollectionBeginAndEndErase
[       OK ] CollectionTest.CollectionBeginAndEndErase (0 ms)
[ RUN      ] CollectionTest.ReserveIncreasesCapacityNotSize
[       OK ] CollectionTest.ReserveIncreasesCapacityNotSize (0 ms
[ RUN      ] CollectionTest.VerifyIndexExceptionIsThrown
[       OK ] CollectionTest.VerifyIndexExceptionIsThrown (0 ms)
[ RUN      ] CollectionTest.AccessInEmptyVectorThrows
[       OK ] CollectionTest.AccessInEmptyVectorThrows (0 ms)
[ RUN      ] CollectionTest.InputAndChangeIndexValue
[       OK ] CollectionTest.InputAndChangeIndexValue (0 ms)
[----------] 16 tests from CollectionTest (5 ms total)

[----------] Global test environment tear-down
[==========] 16 tests from 1 test case ran. (8 ms total)
[  PASSED  ] 15 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] CollectionTest.AlwaysFail

 1 FAILED TEST
```

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

Transition and health check
Configure and deploy, security settings, penetration test

Monitor and detect
Log collection, SIEM, analytics, event alerting, intrusion detection

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

Production

Green Pace

TOOLS

- DevSecOps stands for development, security, and operations. This is a software development approach that integrates security practices into every stage of the software development process.
- Tools used are automation which help with testing and code development.
- Constant security testing is used throughout the development to catch bugs
- Scanning is used to help ensure there is no vulnerabilities.

Green Pace

RISKS AND BENEFITS

The way to handle a security problem is to act fast. When there is a data leakage or servers are down, fixing it right away is important. You want to minimize the damage of what happened. Also fixing the issues quickly gains the user's trust.

Green Pace

RECOMMENDATIONS

- Communication is the key to success.
- Training and testing the team.

CONCLUSIONS

- Validate Input Data: Validating input is very important, it helps ensure a system is safe with handling data. It protects the program from SQL injections and cross-site scripting.
- Adopt a Secure Coding Standard: Adopting a coding standard helps the team be on the same page on how to secure the program and implement good strategies.

Green Pace