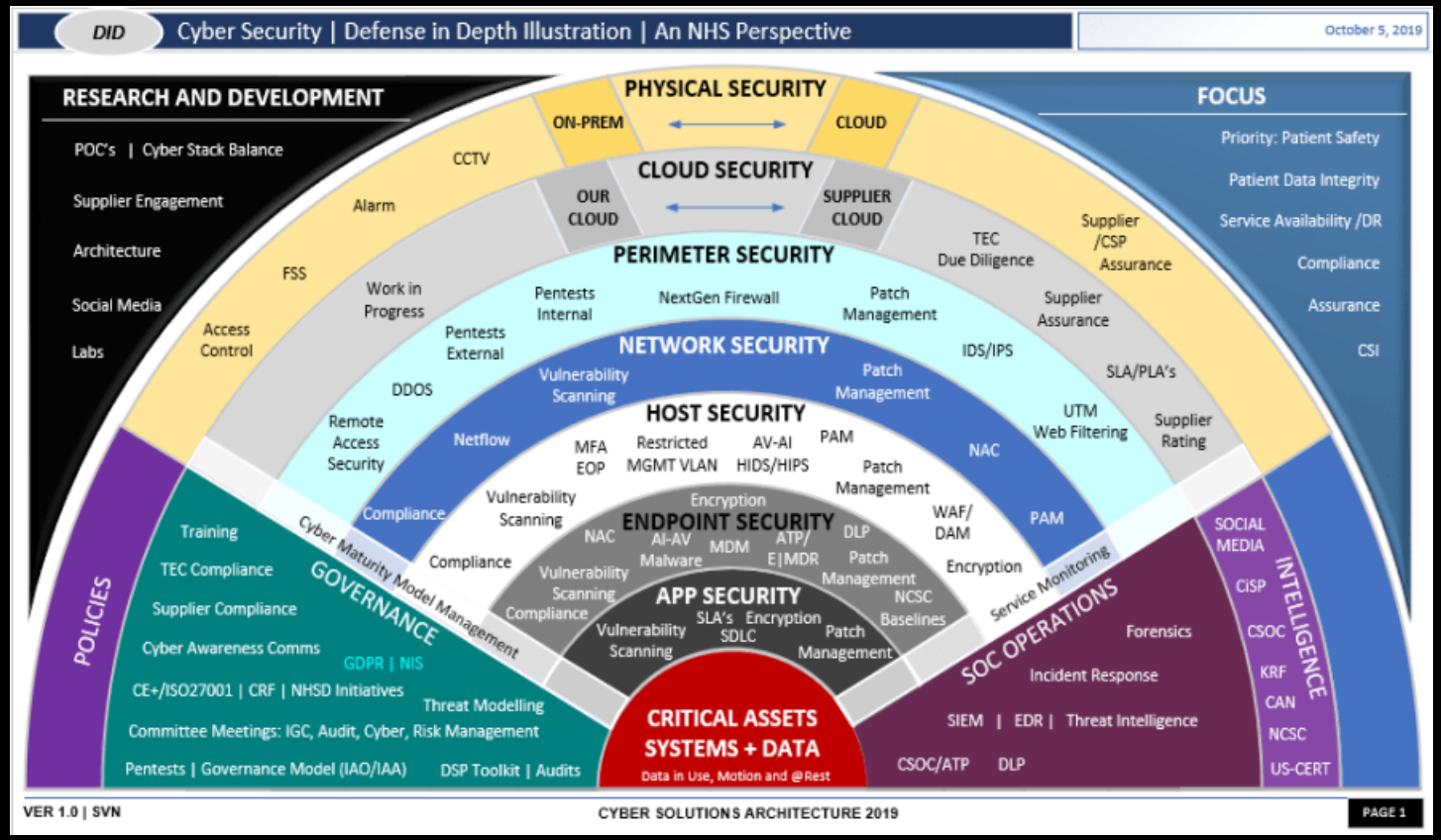
Green Pace

Security Policy Presentation

Developer: *Bryce Jensen*

OVERVIEW: DEFENSE IN DEPTH

- Introduction
- Role in Defense-in-Depth



Green Pace

OVERVIEW: THREATS MATRIX

LIKELY

STD-002-CPP (Signed Integer Overflow)
STD-003-CPP (String Storage Overflow)
STD-004-CLG (Sanitize Data)
STD-007-CPP (Exception Safety)
STD-010-CPP (Abrupt Program Termination)

UNLIKELY

STD-001-CLG (One Definition Rule)
STD-005-CLG (Double-Free Vulnerability)
STD-006-CLG (Static Assertions)
STD-008-CPP (Virtual Function Calls)
STD-009-CPP (File Resource Leaks)

PRIORITY

STD-003-CPP (String Storage Overflow) – P18
STD-004-CLG (Sanitize Data) – P18
STD-002-CPP (Signed Integer Overflow) – P09
STD-007-CPP (Exception Safety) – P09
STD-005-CLG (Double-Free Vulnerability) – P09
STD-009-CPP (File Resource Leaks) – P04
STD-010-CPP (Abrupt Program Termination) – P04
STD-001-CLG (One Definition Rule) – P03
STD-008-CPP (Virtual Function Calls) – P02
STD-006-CLG (Static Assertions) – P01

LOW-PRIORITY

10 PRINCIPLES

- Sanitize Data Sent to Other Systems
- Validate Input Data
- Least Privilege
- Architect for Security
- Defense in Depth
- Heed Compiler Warnings
- Use Effective QA Techniques
- Adopt Secure Coding Standards
- Default Deny
- Keep It Simple

CODING STANDARDS

- Follow the One Definition Rule (ODR)
- Ensure Proper Use of Signed Data Types
- String Storage Should Have Enough Space for Character Data and the Null Terminator
- Sanitize Data
- Store New Values into Pointers After Being Freed
- Static Insertions Should Be Used to Test Constant Expression Values
- Ensure Exception Safety
- Do Not Invoke Virtual Functions from Constructors or Destructors
- Close Files That Are No Longer Needed
- Don’t Terminate a Program Abruptly

ENCRYPTION POLICIES

- Encryption at Rest
- Encryption in Flight
- Encryption in Use

Green Pace

TRIPLE-A POLICIES

- Authentication
- Authorization
- Accounting

Green Pace

UNIT TESTING

Memory and Storage of Data

Does Reserve Increase Capacity Without Changing Size?

Description:

- Reserving space in a collection should increase capacity without altering its size.

Further Testing:

- Reserve an equal amount of space, no change should be found.

```
// Unit test to verify that reserve increases capacity without changing the size
TEST_F(CollectionTest, ReserveIncreasesCapacityWithoutChangingSize)
{
    add_entries(5);  // Adding 5 elements to the collection

    // Store current capacity before reserving new space
    auto old_capacity = collection->capacity();
    collection->reserve(10);  // Reserve space for 10 elements
    EXPECT_GT(collection->capacity(), old_capacity);  // Ensure capacity increased
    EXPECT_EQ(collection->size(), 5);  // Ensure size remains the same (5 elements)
}
```

Green Pace

Does Accessing Out-of-Bounds Throw an Exception?

Description:

- Attempting to access out of the bounds of a collection should throw an exception.

Further Testing:

- Attempting to access a negative index, should also throw an exception.

```
// Unit test to verify that accessing out-of-bounds throws std::out_of_range exception
TEST_F(CollectionTest, AccessOutOfBoundsThrowsException)
{
    add_entries(1);  // Add a single entry to the collection

    // Access index 5 (out of bounds), expect exception
    EXPECT_THROW(collection->at(5), std::out_of_range);

    // Access negative index, expect exception
    EXPECT_THROW(collection->at(-1), std::out_of_range);
}
```

Green Pace

Can Duplicate Values Be Added to the Collection?

Description:

- Verify that adding a duplicate value to a collection works.

```
// Unit test to verify that duplicate values can be added to the collection
TEST_F(CollectionTest, CanAddDuplicateValues)
{
    collection->push_back(42);  // Add the value 42
    collection->push_back(42);  // Add the value 42 again (duplicate)
    ASSERT_EQ(collection->size(), 2);  // Ensure collection size is 2
    EXPECT_EQ(collection->at(0), 42);  // First element should be 42
    EXPECT_EQ(collection->at(1), 42);  // Second element should be 42 (duplicate)
}
```

Further Testing:

- Check whether adding duplicates unnecessarily inflates memory usage.

Green Pace

Can Duplicate Values Be Added to the Collection?

Description:

- Accessing an element in an empty collection should result in an exception.

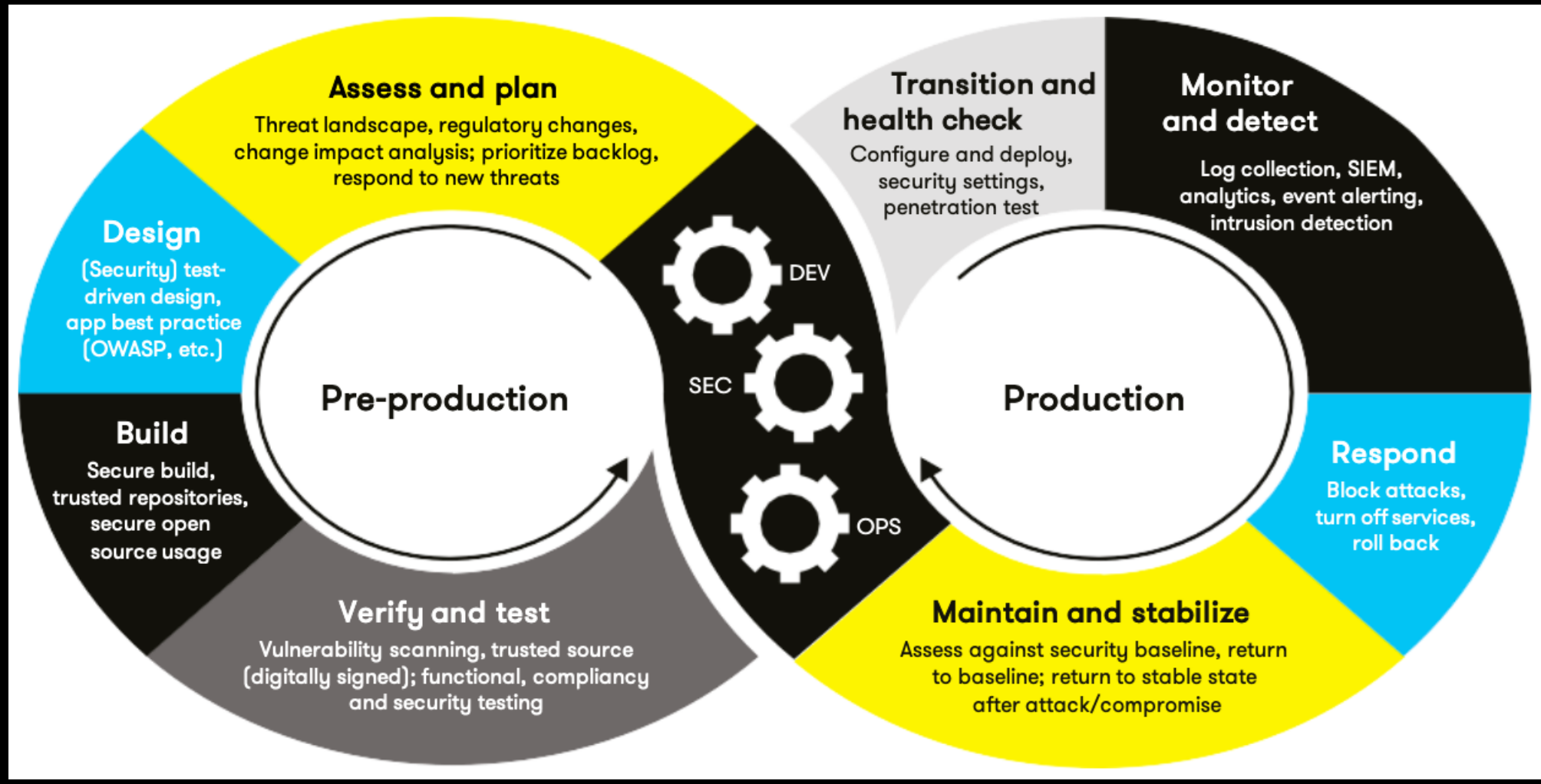
```
// Unit test to verify that accessing an empty collection
//  throws std::out_of_range exception
TEST_F(CollectionTest, AccessEmptyCollectionThrowsException)
{
    ASSERT_TRUE(collection->empty());  // Ensure the collection is empty

    // Try accessing element 0, expect exception
    EXPECT_THROW(collection->at(0), std::out_of_range);
}
```

Further Testing:

- After throwing an exception, add an element to the collection and check if it works correctly.

Green Pace

AUTOMATION SUMMARY AND TOOLS



Green Pace

RISKS, BENEFITS, & RECOMMENDATION

Current Problems	Solutions	Risks of Waiting	Benefits of Acting Now
Security Gaps	Integrate Security Early	Escalating the Threat	Risk Management
Increased Breaches	Automate Security Processes	Increased Costs	Streamlined Processes
Slow Responses	Enhance Collaboration	Compliance Violations	Enhanced Reputation

Green Pace

RECOMMENDATIONS

- Conduct a Risk Assessment
- Prioritize Vulnerabilities
- Invest in Training
- Implement Security Automation
- Foster Collaboration

Green Pace

CONCLUSIONS

- Beneficial and necessary to be on top of security

Green Pace