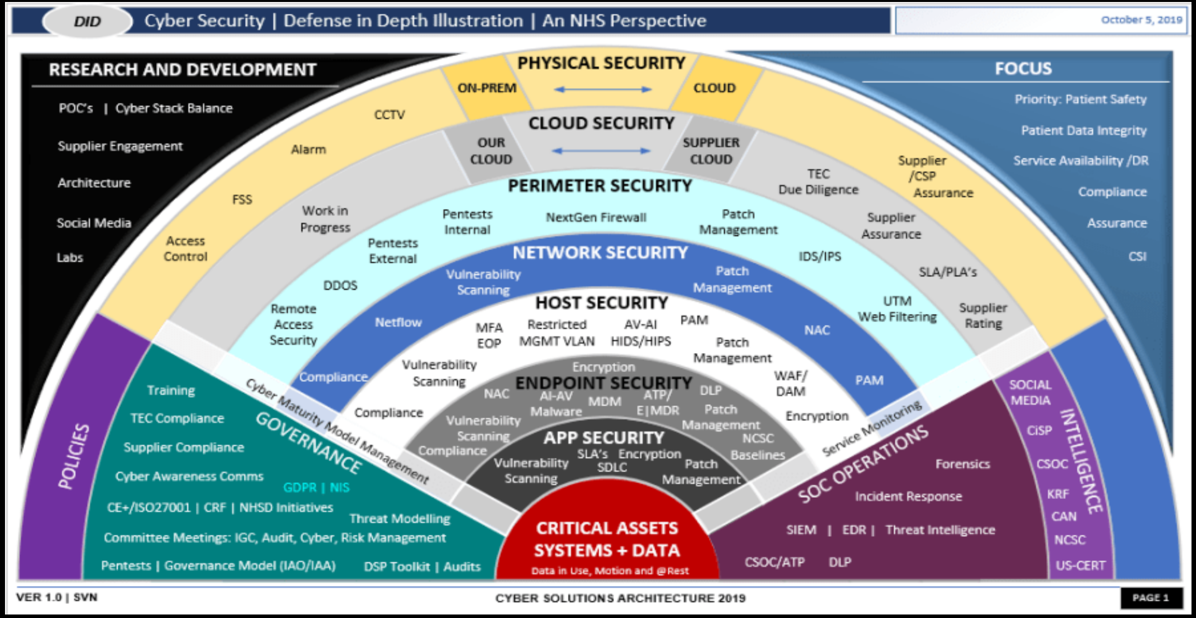
# Green Pace

Security Policy Presentation
Developer: *Onasis McCuien*

Green Pace

# OVERVIEW: DEFENSE IN DEPTH

Software development lifecycle security requirements may be effectively and consistently maintained by including automated procedures into the DevOpsPipeline. This alignment ensures that security is considered throughout the development process rather than being and afterthought



## THREATS MATRIX

| | |
|---|---|
| Likely<br>Medium | Priority<br>High |
| Low priority<br>No | Unlikely<br>Medium |

# 10 PRINCIPLES

| | |
|---|---|
| Validate Input Data | [Proper input validation strategies will help in the fight against malicious attacks and ensure that the integrity and security of the system is good. ] |
| Heed Compiler Warnings | [A compiler error means that your code has some kind of syntax error that causes the compiler to give up. No program is generated due to this error. This typically means that your compiler suspects that your program might crash or behave in a strange manner.] |
| Architect and Design for Security Policies | [Architectural and Design for Security encompass system values, Access control requirements, encryption and authentication requirements, and virus control requirements. These security structures are the heart of security against cyber-attacks. ] |
| Keep It Simple | [The “Keep it Simple” policy simply means that you should minimize the complexity of your program, use well established and proven security practices in your code, and most importantly, keep the code updated the most current security standards. ] |
| Default Deny | [The default deny principle is extremely important. This is what denies access to the cyber criminals. Default denies places huge importance on denying access by default and only allowing access to the rightly authorized. .] |
| Adhere to the Principle of Least Privilege | [You should only grant users the absolute minimum access needed in order for users to complete their issued tasks. This helps to greatly minimize the risk of security breaches and other possible lone wolf attacks.]] |
| Sanitize Data Sent to Other Systems | [This principle is utilized to clean and validate data in order to remove any harmful elements before sending it to other operating systems. ] |
| Practice Defense in Depth | [Practice defense in depth is a technique that gives multiple layers to your security. By introducing several layers of security such as encryption, firewalls, access controls, etc.. Adds more robust security to your system. ] |
| Use Effective Quality Assurance Techniques | [Effective Quality assurance techniques involve steps such as testing, code review, scanning the code, and actual SQL breach testing. This will ensure that your code shows no weak points. ] |
| Adopt a Secure Coding Standard | [Adopt a Secure coding standard simply means for a developer to adhere to the best practices and guidelines that help prevent vulnerabilities and weaknesses in their code. .] |

## CODING STANDARDS

- Data Value
  - Severity: Low Likelihood: Unlikely Remediation Cost: Medium Priority: 2 Level:3
- String code
  - Severity: Medium Likelihood: likely Remediation Cost: Medium Priority: 2 Level:3
- SQL Injecting
  - Severity:Low Likelihood: Unlikely Remediation Cost: Medium Priority: 2 Level:3
- Memory Protection
  - Severity:Low Likelihood: Unlikely Remediation Cost: Medium Priority: 2 Level:3
- Assertions
  - Severity:Low Likelihood: Unlikely Remediation Cost: Medium Priority: 2 Level:3
- Exceptions
  - Severity:Low Likelihood: Unlikely Remediation Cost: Medium Priority: 2 Level:3
- Coding style
  - Severity:Low Likelihood: Unlikely Remediation Cost: Medium Priority: 2 Level:3
- Identifiers
  - Severity:Low Likelihood: Unlikely Remediation Cost: Medium Priority: 2 Level:3
- Documentation
  - Severity:Low Likelihood: Unlikely Remediation Cost: Medium Priority: 2 Level:3
- Code Length
  - Severity:Low Likelihood: Unlikely Remediation Cost: Medium Priority: 2 Level:3
- Don’t use go to statements
  - Severity:Low Likelihood: Unlikely Remediation Cost: Medium Priority: 2 Level:3

## ENCRYPTION POLICIES

- The use of encryption in flight and at rest is a crucial aspect of good cloud security architecture. It helps protect data during transmission (in flight) and when it is stored (at rest). Encryption is also required by many laws and regulations to ensure the privacy and security of sensitive information. Therefore, encryption is necessary in all cloud deployments to comply with these laws and regulations and to maintain a secure cloud environment.

## TRIPLE-A POLICIES

- [The triple A policies, authentication, authorization, and accounting are policies that support secure access to resources and track user activities.

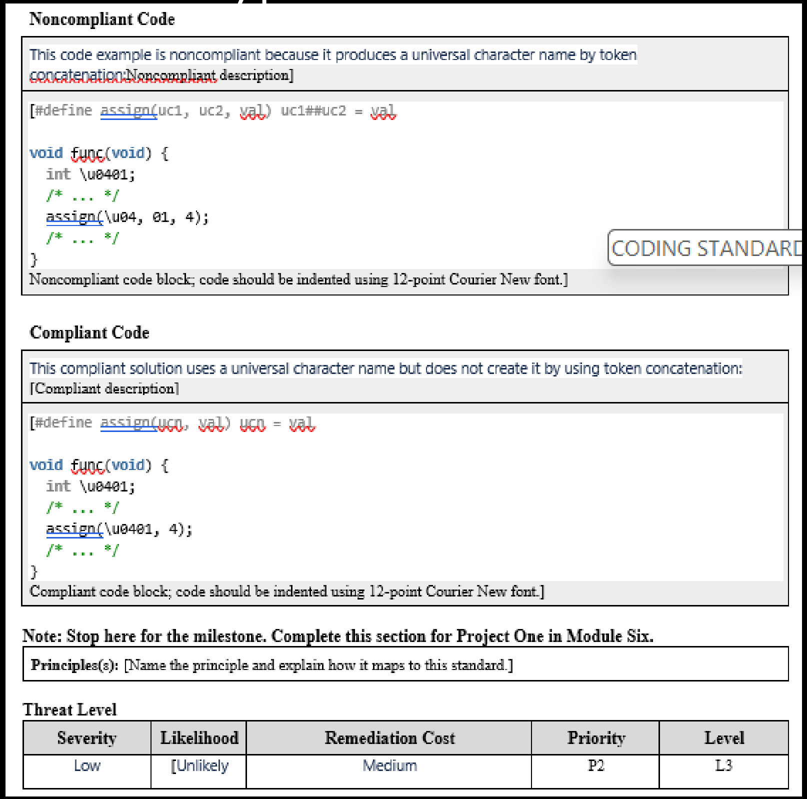
Authentication:
- User Id, Password, Biometric data]

Authorization:
- User roles

Accounting:
- User Id, timestamp

# Unit Testing

## Data Type



**Noncompliant Code**

This code example is noncompliant because it produces a universal character name by token concatenation:Noncompliant description]

```
[#define assign(uc1, uc2, val) uc1##uc2 = val

void func(void) {
  int \u0401;
  /* ... */
  assign(\u04, 01, 4);
  /* ... */
}
```

Noncompliant code block; code should be indented using 12-point Courier New font.]

CODING STANDARD

**Compliant Code**

This compliant solution uses a universal character name but does not create it by using token concatenation: [Compliant description]

```
[#define assign(ucn, val) ucn = val

void func(void) {
  int \u0401;
  /* ... */
  assign(\u0401, 4);
  /* ... */
}
```

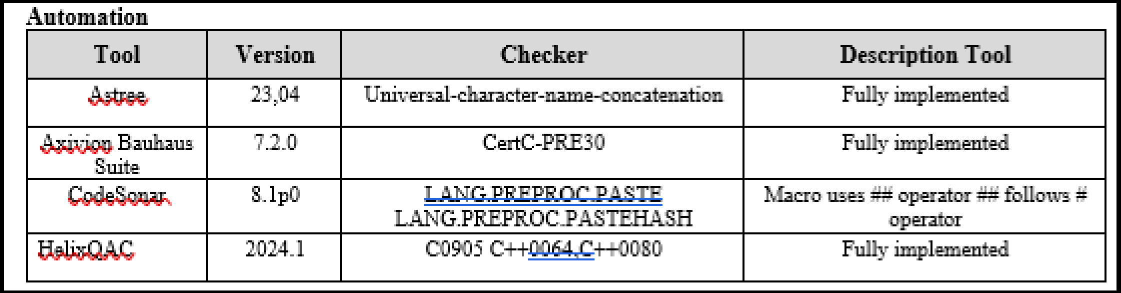
Compliant code block; code should be indented using 12-point Courier New font.]

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

**Principles(s):** [Name the principle and explain how it maps to this standard.]

**Threat Level**

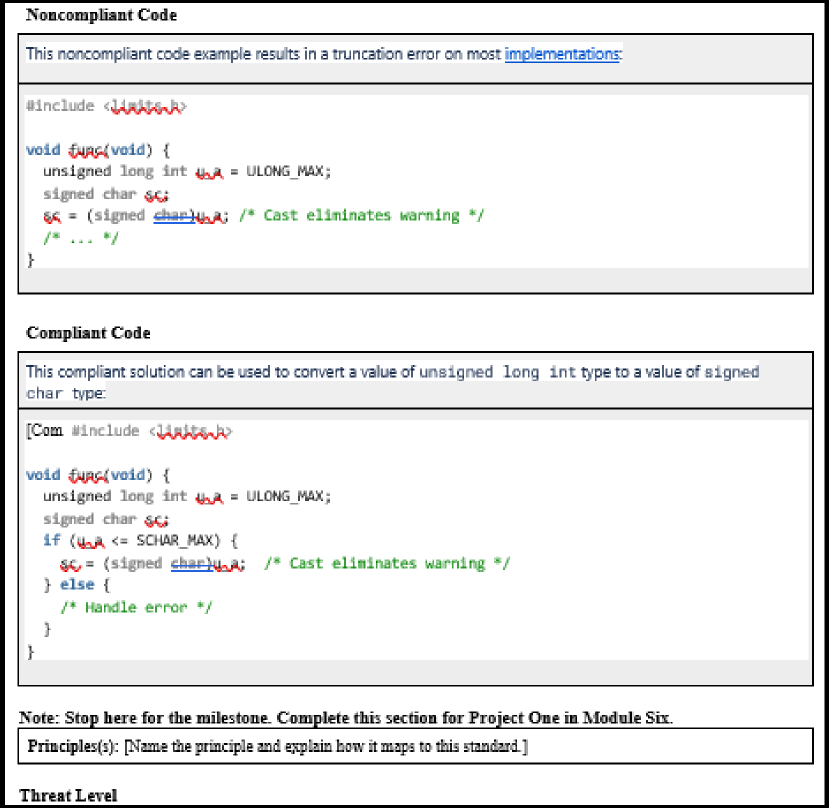
| Severity | Likelihood | Remediation Cost | Priority | Level |
|---|---|---|---|---|
| Low | [Unlikely | Medium | P2 | L3 |



**Automation**

| Tool | Version | Checker | Description Tool |
|---|---|---|---|
| Astree | 23,04 | Universal-character-name-concatenation | Fully implemented |
| Axivion Bauhaus Suite | 7.2.0 | CertC-PRE30 | Fully implemented |
| CodeSonar | 8.1p0 | LANG.PREPROC.PASTE LANG.PREPROC.PASTEHASH | Macro uses ## operator ## follows # operator |
| HelixQAC | 2024.1 | C0905 C++0064,C++0080 | Fully implemented |

- Data Value



**Noncompliant Code**

This noncompliant code example results in a truncation error on most implementations:

```
#include <limits.h>

void func(void) {
  unsigned long int u_a = ULONG_MAX;
  signed char sc;
  sc = (signed char)u_a; /* Cast eliminates warning */
  /* ... */
}
```

**Compliant Code**

This compliant solution can be used to convert a value of `unsigned long int` type to a value of `signed char` type:

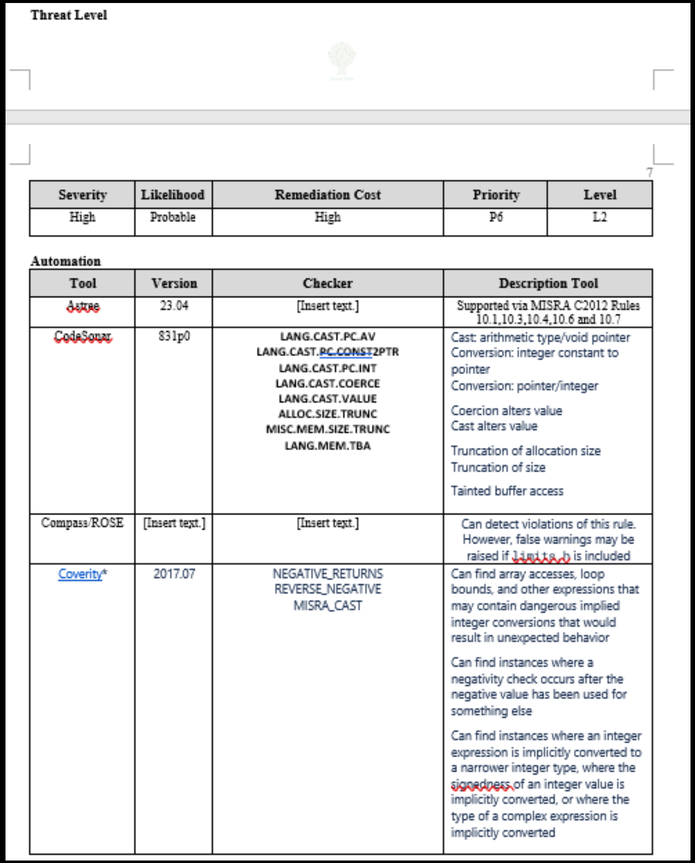
```
[Com #include <limits.h>

void func(void) {
  unsigned long int u_a = ULONG_MAX;
  signed char sc;
  if (u_a <= SCHAR_MAX) {
    sc = (signed char)u_a;  /* Cast eliminates warning */
  } else {
    /* Handle error */
  }
}
```

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| |
|---|
| Principles(s): [Name the principle and explain how it maps to this standard.] |

**Threat Level**



**Threat Level**

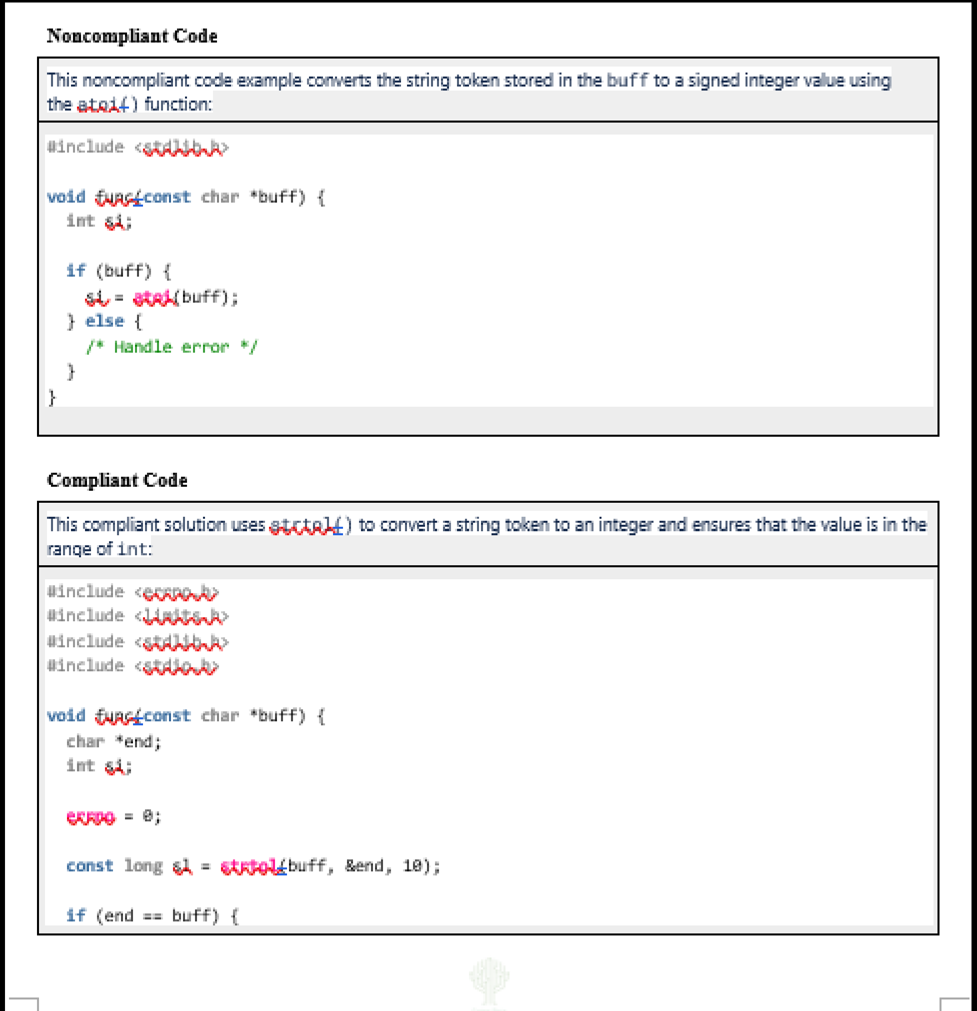
7

| Severity | Likelihood | Remediation Cost | Priority | Level |
|---|---|---|---|---|
| High | Probable | High | P6 | L2 |

**Automation**

| Tool | Version | Checker | Description Tool |
|---|---|---|---|
| Astree | 23.04 | [Insert text.] | Supported via MISRA C2012 Rules 10.1,10.3,10.4,10.6 and 10.7 |
| CodeSonar | 831p0 | LANG.CAST.PC.AV<br>LANG.CAST.PC.CONST2PTR<br>LANG.CAST.PC.INT<br>LANG.CAST.COERCE<br>LANG.CAST.VALUE<br>ALLOC.SIZE.TRUNC<br>MISC.MEM.SIZE.TRUNC<br>LANG.MEM.TBA | Cast: arithmetic type/void pointer<br>Conversion: integer constant to pointer<br>Conversion: pointer/integer<br>Coercion alters value<br>Cast alters value<br>Truncation of allocation size<br>Truncation of size<br>Tainted buffer access |
| Compass/ROSE | [Insert text.] | [Insert text.] | Can detect violations of this rule. However, false warnings may be raised if `limits.h` is included |
| Coverity* | 2017.07 | NEGATIVE_RETURNS<br>REVERSE_NEGATIVE<br>MISRA_CAST | Can find array accesses, loop bounds, and other expressions that may contain dangerous implied integer conversions that would result in unexpected behavior<br>Can find instances where a negativity check occurs after the negative value has been used for something else<br>Can find instances where an integer expression is implicitly converted to a narrower integer type, where the signedness of an integer value is implicitly converted, or where the type of a complex expression is implicitly converted |

- String Correctness



**Noncompliant Code**

This noncompliant code example converts the string token stored in the buff to a signed integer value using the atoi() function:

```c
#include <stdlib.h>

void func(const char *buff) {
  int si;

  if (buff) {
    si = atoi(buff);
  } else {
    /* Handle error */
  }
}
```

**Compliant Code**

This compliant solution uses strtol() to convert a string token to an integer and ensures that the value is in the range of int:

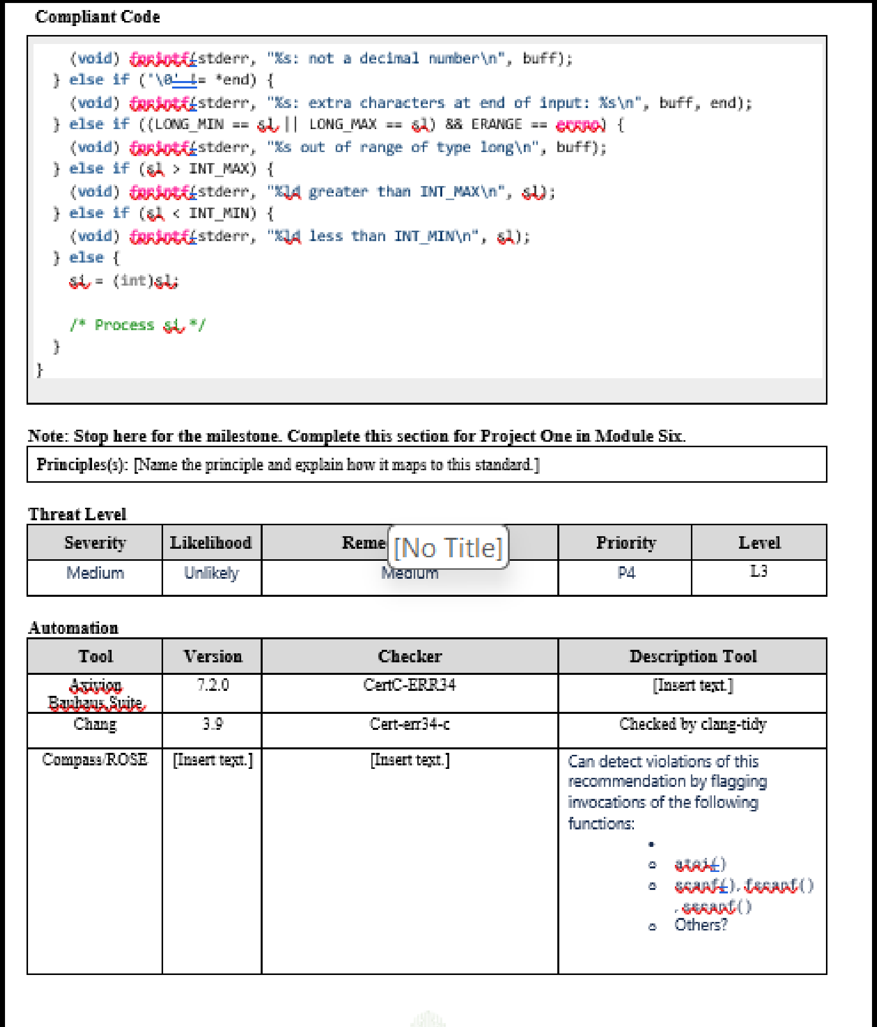
```c
#include <errno.h>
#include <limits.h>
#include <stdlib.h>
#include <stdio.h>

void func(const char *buff) {
  char *end;
  int si;

  errno = 0;

  const long sl = strtol(buff, &end, 10);

  if (end == buff) {
```



**Compliant Code**

```c
    (void) fprintf(stderr, "%s: not a decimal number\n", buff);
  } else if ('\0' != *end) {
    (void) fprintf(stderr, "%s: extra characters at end of input: %s\n", buff, end);
  } else if ((LONG_MIN == sl || LONG_MAX == sl) && ERANGE == errno) {
    (void) fprintf(stderr, "%s out of range of type long\n", buff);
  } else if (sl > INT_MAX) {
    (void) fprintf(stderr, "%ld greater than INT_MAX\n", sl);
  } else if (sl < INT_MIN) {
    (void) fprintf(stderr, "%ld less than INT_MIN\n", sl);
  } else {
    si = (int)sl;

    /* Process si */
  }
}
```

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

**Principles(s):** [Name the principle and explain how it maps to this standard.]

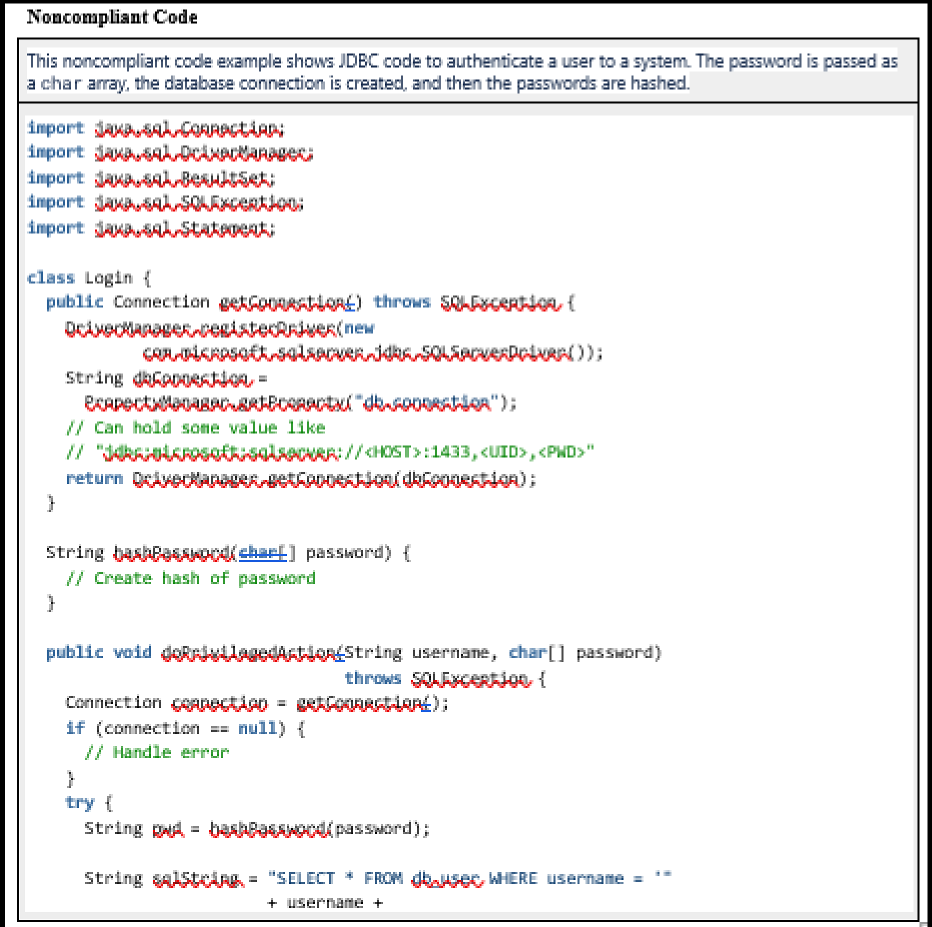
**Threat Level**

| Severity | Likelihood | Reme [No Title] | Priority | Level |
|---|---|---|---|---|
| Medium | Unlikely | Medium | P4 | L3 |

**Automation**

| Tool | Version | Checker | Description Tool |
|---|---|---|---|
| Axivion Bauhaus Suite | 7.2.0 | CertC-ERR34 | [Insert text.] |
| Clang | 3.9 | Cert-err34-c | Checked by clang-tidy |
| Compass/ROSE | [Insert text.] | [Insert text.] | Can detect violations of this recommendation by flagging invocations of the following functions: • ◦ atoi() ◦ scanf(), fscanf(), sscanf() ◦ Others? |

- SQL Injector



**Noncompliant Code**

This noncompliant code example shows JDBC code to authenticate a user to a system. The password is passed as a char array, the database connection is created, and then the passwords are hashed.

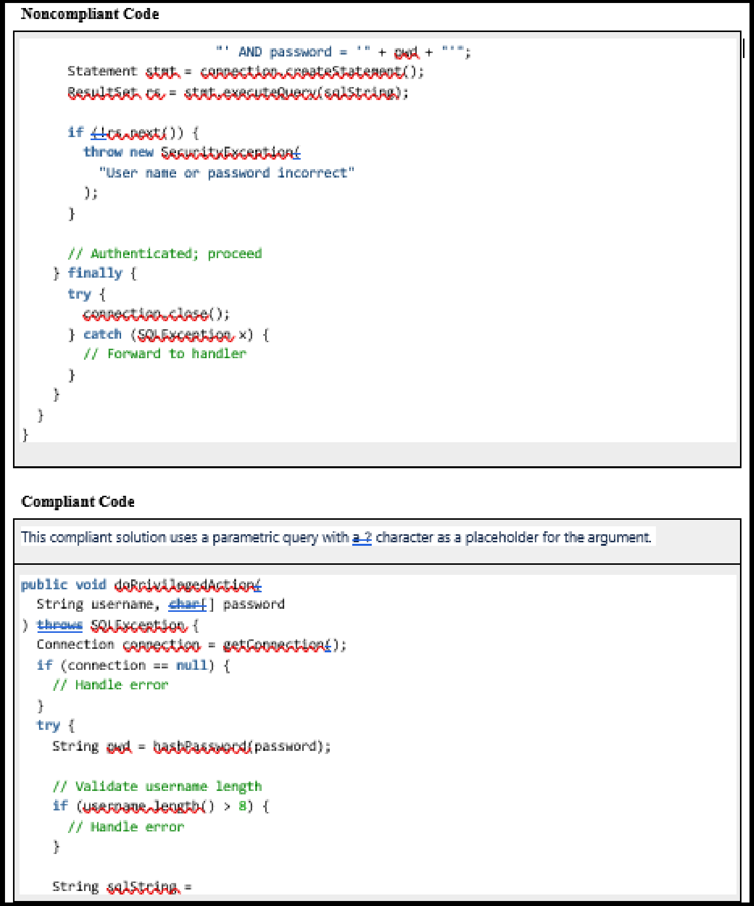
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

class Login {
  public Connection getConnection() throws SQLException {
    DriverManager.registerDriver(new
            com.microsoft.sqlserver.jdbc.SQLServerDriver());
    String dbConnection =
      PropertyManager.getProperty("db.connection");
    // Can hold some value like
    // "jdbc:microsoft:sqlserver://<HOST>:1433,<UID>,<PWD>"
    return DriverManager.getConnection(dbConnection);
  }

  String hashPassword(char[] password) {
    // Create hash of password
  }

  public void doPrivilegedAction(String username, char[] password)
                                 throws SQLException {
    Connection connection = getConnection();
    if (connection == null) {
      // Handle error
    }
    try {
      String pwd = hashPassword(password);

      String sqlString = "SELECT * FROM db_user WHERE username = '"
                         + username +
```



**Noncompliant Code**

```
                         "' AND password = '" + pwd + "'";
      Statement stmt = connection.createStatement();
      ResultSet rs = stmt.executeQuery(sqlString);

      if (!rs.next()) {
        throw new SecurityException(
          "User name or password incorrect"
        );
      }

      // Authenticated; proceed
    } finally {
      try {
        connection.close();
      } catch (SQLException x) {
        // Forward to handler
      }
    }
  }
}
```

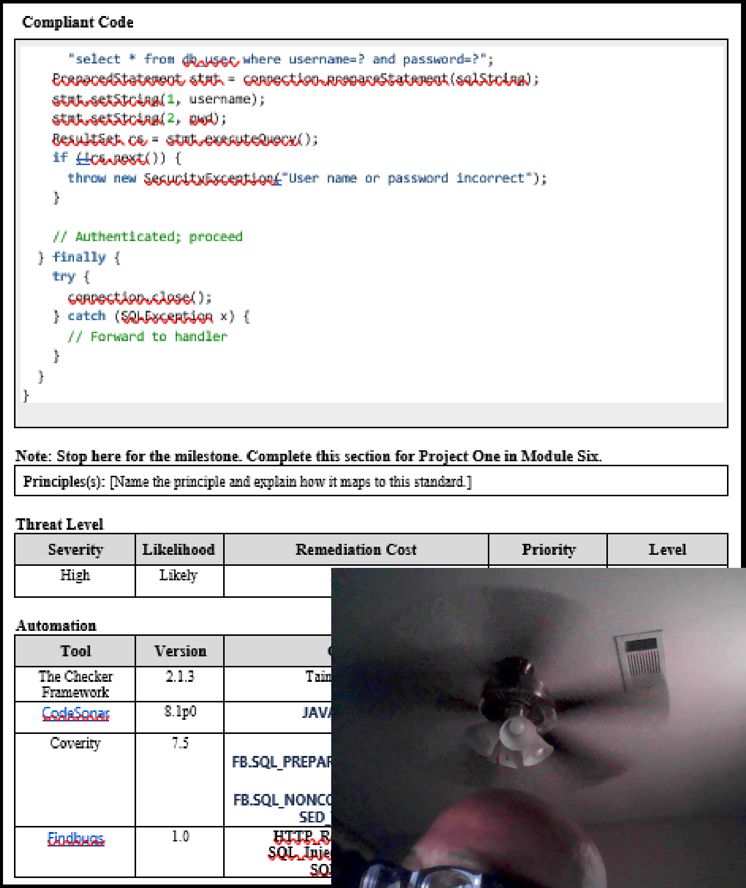
**Compliant Code**

This compliant solution uses a parametric query with a ? character as a placeholder for the argument.

```
public void doPrivilegedAction(
  String username, char[] password
) throws SQLException {
  Connection connection = getConnection();
  if (connection == null) {
    // Handle error
  }
  try {
    String pwd = hashPassword(password);

    // Validate username length
    if (username.length() > 8) {
      // Handle error
    }

    String sqlString =
```



**Compliant Code**

```
      "select * from db_user where username=? and password=?";
    PreparedStatement stmt = connection.prepareStatement(sqlString);
    stmt.setString(1, username);
    stmt.setString(2, pwd);
    ResultSet rs = stmt.executeQuery();
    if (!rs.next()) {
      throw new SecurityException("User name or password incorrect");
    }

    // Authenticated; proceed
  } finally {
    try {
      connection.close();
    } catch (SQLException x) {
      // Forward to handler
    }
  }
}
```

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| Principles(s): [Name the principle and explain how it maps to this standard.] |
|---|

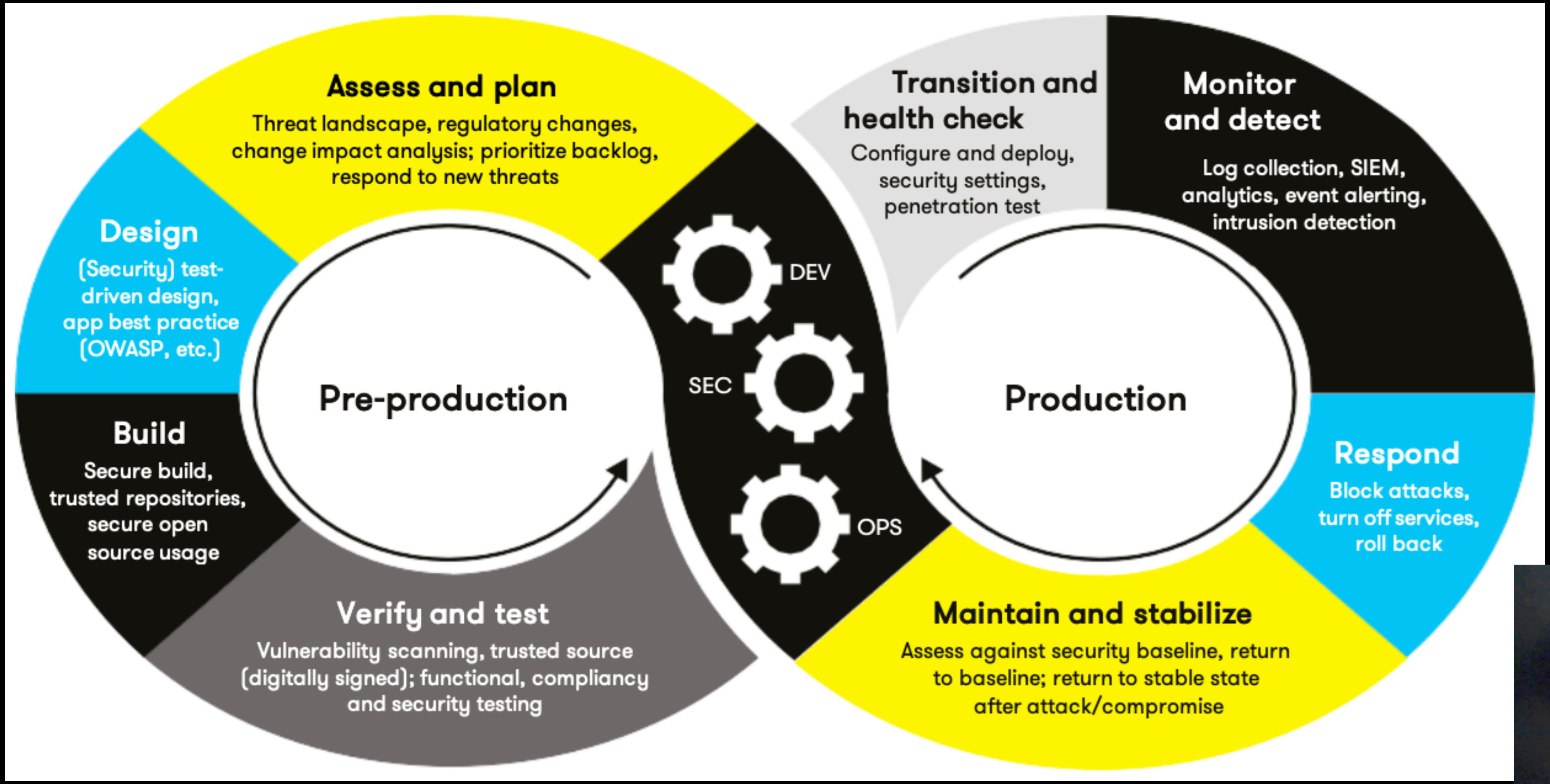
**Threat Level**

| Severity | Likelihood | Remediation Cost | Priority | Level |
|---|---|---|---|---|
| High | Likely | | | |

**Automation**

| Tool | Version | C |
|---|---|---|
| The Checker Framework | 2.1.3 | Tain |
| CodeSonar | 8.1p0 | JAVA |
| Coverity | 7.5 | FB.SQL_PREPAR<br>FB.SQL_NONCO<br>SED |
| Findbugs | 1.0 | HTTP_R<br>SQL_Inje<br>SQ |

# 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

## TOOLS

- The DevSecOps pipeline aims to integrate security practices into every stage of the software development process, ensuring that security is not an afterthought but a fundamental aspect of the development process. By incorporating security early on and automating security checks, organizations can build and deploy more secure software.]
- Analyzing risks
- Pre-Production: Security-driven procedures and testing are integrated in the design and early phases
- Build, Verification, and Testing: Using trustworthy repositories and guaranteeing safe use of open-source components, security controls are incorporated into the build process.
- Transition and Health Check: To guarantee resilience against assaults, configurations are protected, deployments are examined for security settings, and penetration tests are carried out before to deployment.
- Monitoring and Detection: To quickly discover security risks, monitoring systems are configured for log collecting, event analytics, and intrusion detection once they are in production.
- Response and Maintenance: Attacks are stopped and modifications soon as a threat is detected.

## RISKS AND BENEFITS

- The DevSecOps framework may be used to automate the enforcement of these requirements inside the DevOps process. To guarantee that security measures are regularly implemented, automation tools and procedures may be incorporated at every level.

## CONCLUSIONS

- Security becomes an essential component of the development process when automated security measures are incorporated into the DevOps pipeline. This guarantees the uniform and effective execution of security standards throughout the software development lifecycle.

## REFERENCES

- [Provide APA-style references with links to resources, articles, and videos that you used in your presentation.]