

08 Cybersecurity



Security First!

General Principles

- Treat all external input as untrusted
- Prefer simple and explicit architectures
- Fail safely and log security-relevant failures (log level WARNING)
- Minimize attack surface and unnecessary dependencies
- Apply the principle of least privilege (both when developing and in the final product)
- Do not reinvent the wheel (use existing encryption algorithms etc)

Secrets and Credentials

- Never hardcode:
 - passwords
 - API keys
 - certificates
 - tokens
 - database credentials
- 2. Do not store secrets in:
 - block diagram constants
 - typedef defaults
 - test VIs
 - screenshots
 - source-controlled config files
- 3. Do not log sensitive information
- 4. Solutions
 - “AES” VI Package by VIGods
 - Better: Store secrets in a place the regular user cannot access

Networking

- Prefer encrypted communication (TLS, HTTPS, SSH)
- Avoid plaintext credentials and insecure protocols
- Validate remote peers (eg using certificates)
 - restrict access on a network level (IP ranges etc)
- 4. Use existing protocols
- 5. Explicitly document:
 - timeout behavior
 - reconnect strategy
 - retry handling

Databases and SQL

- Use parameterized queries whenever possible
- Avoid dynamically concatenating SQL strings from user input
- Validate and sanitize externally provided data before database operations
- Restrict database permissions to the minimum required access
- Never store database credentials directly in source code
- Log database failures without exposing sensitive query data
- Explicitly handle connection loss and timeout behavior

LabVIEW Misc

VI Server

- Disable VI Server if not required
- Restrict network access and permissions
- Never expose unrestricted VI Server access on production systems

Dynamic VI Loading

- Only load trusted plugins or VIs
- Avoid loading VIs from user-writable directories
- Use strict connector pane contracts and versioning

System Exec

- Treat all command line input as untrusted
- Avoid constructing shell commands from unchecked user input
- Document all external tool dependencies

File Handling

- Validate file paths and filenames
- Avoid path traversal vulnerabilities
- Restrict writable and executable locations

DQMH and Modular Architectures

- Encapsulate queues, references, and communication resources
- Avoid exposing internal module resources
- Validate message payloads and typedef compatibility
- Keep security-related blocking operations outside the EHL

Logging and Error Handling

- Log security-relevant events:
 - failed authentication
 - rejected certificates
 - malformed packets

- permission violations
- 2. Logs should include:
 - timestamps
 - module names
 - severity
 - connection information
- 3. Avoid excessive logging in real-time loops

Deployment

- Remove development tooling from production systems
- Avoid unnecessary administrator privileges
- Keep dependencies updated
- Prefer signed installers and executables

Resources

- [EU Cyber Resilience Act \(CRA\)](#)

The HSE Way of Working:

A set of guidelines that recommend programming style, better practices, and methods for all our LabVIEW projects. We ask all our peers to follow these guidelines to help improve the readability of our shared source code and make software maintenance easier.

[<< 07 Networking](#) [11 DQMH >>](#)

From:

<https://dokuwiki.hampel-soft.com/> - HAMPEL SOFTWARE ENGINEERING

Permanent link:

<https://dokuwiki.hampel-soft.com/kb/bestpractices/codingconventions/cybersecurity?rev=1782381935>

Last update: 2026/06/25 10:05

