InterSystems Corporation
One Memorial Drive
Cambridge, MA 02142-1356

Main: +1 617 621 0600
Support: +1 617 621 0700
InterSystems.com

WHITE PAPER – Secure Coding Practices

InterSystems Development follows a series of policies and procedures to ensure that products developed by InterSystems are stable, secure, and reliable for intended customer use. While it is not our policy to share specific details of coding practices that we use to protect against security vulnerabilities, this document includes high level details on our policies regarding the development of our products.

All members of our development team are asked to confirm understanding of key policies on an annual basis. This includes our Secure Development policy.

This white paper highlights more specifics of our development and quality processes as they pertain to maintaining and delivering secure products.

Development Guidelines

Our developers work under and are mentored on coding standards to ensure security vulnerabilities are not introduced inadvertently. Code reviews are done as part of our standard process to ensure coding guidelines are followed and to ensure a high quality of code.

These guidelines include, but are not limited to, the following:

- Assuming that adversaries could have our source code, never store anything directly in source code that could be useful in an attack, such as a cryptographic key or password.
- Never check prototype or demo code into main source trees or branches accessible to main source trees. Prototype code may not yet have had a full security review and demo code may skip some (security-related) best practices in the interest of highlighting a particular feature or function.
- Never use client input directly to execute a statement or allow for generation of a dynamic SQL statement. Input provided by the client should only be used as raw data in specific elements and never as executable code.
- Avoid use of public variables and ensure all variables are scoped within appropriate procedure blocks to prevent external manipulation of variables.
- Avoid specific Cache Object Script functions that would allow for direct code execution or access of system executables.
- When working with files from an application, avoid use of parameters for filenames and check security resources before manipulating files.
- Do not construct Dynamic SQL from user-supplied strings, avoiding the potential for SQL injection.

- Do not allow external APIs to ask the server to execute specific methods by use of parameters.
- Avoid URL parameter vulnerabilities (the series of mechanisms to avoid are not outlined in this document for security reasons)
- Fail securely – Ensure that no diagnostic information could inadvertently expose secure information or information that could be used to form an attack. Also, avoid failing from a more trusted to a less trusted method or procedure.
- Keep sensitive data in memory for ONLY as long as needed and clear caches as appropriate. Mechanisms for immediately overwriting the memory are provided to developers.
- When submitting a code change, include testing instructions for our quality team that describe how to attempt to provoke a Denial of Access if this area allows for user invokable code and state the level of user access required.
- Utilize cryptography appropriately (guidelines not included in this document for security reasons) and utilize the standard cryptographic algorithms provided in our system APIs.
- Ensure all web services are secured and encrypt the username tokens.
- Clearly document the roles that an API or feature requires in order to minimize the chance that site developers and administrators over-privilege roles due to lack of understanding of discrete resource needs.

Quality Assurance Process

As part of our product qualification process we make use of a third party penetration testing tool to verify that our software is free from vulnerabilities such as (but not limited to):

- Cross-site scripting
- Cross-site request forgery
- SQL injection
- Unicode transformation

This tool is used for every release of IRIS Data Platform, Caché, Ensemble, HealthShare, and TrakCare.

All potential security concerns discovered by the tool are logged as potential InterSystems product issues, which then follow our standard processes for triage, resolution and re-verification by Product Management, Development and QD, respectively.

In addition, all potential security-related product issues are reviewed by the Director of QD and Product Manager for Security as an additional safeguard to ensure proper and timely mitigation.