MINERALS MANAGEMENT SERVICE
INTERIM POLICY DOCUMENT

Effective Date: May 30, 2008
IPD No.: 08-11
Series: Administrative
Title: Secure Application and System Development

Originating Office: Information Management Division, Administration and Budget

1. Purpose. The purpose of this policy is to establish secure application and system development standards for the Minerals Management Service (MMS).

2. Objective. To ensure that applications and systems, hereafter referred to as systems, are developed securely and meet predefined security requirements prior to implementation on MMS production networks. The guidance set forth in this policy is designed to minimize potential risk to the MMS created by the introduction of insecure systems. The desired outcome is to ensure the confidentiality, availability, and integrity of the MMS information technology environment.

3. Authorities.

A. Departmental Manual (DM) 375 DM 19, Information Resources Management.

B. MMS Manual 375.19, Information Resources Management Program.

C. National Institute of Standards and Technology (NIST) 800-27, Rev. A. Engineering Principles for Information Technology Security (A Baseline for Achieving Security), June 2004.

D. NIST 800-64, Rev. 1. Security Considerations in the Information System Development Life Cycle, June 2004.

4. Scope. This policy applies to all MMS employees, contractors, vendors, and agents.

5. Definitions.

A. Security Assessment is a systematic examination of a system to determine the adequacy of security measures, validate documented security controls, identify security deficiencies, provide data from which to predict the effectiveness of proposed security measures, and confirm the adequacy of such measures after implementation. A security assessment may include a vulnerability assessment, penetration test, validation of security technical implementation guide(s), comparative analysis of documentation, source code validation, database audit, as well as other assessment techniques (see MMS Procedure entitled "MMS IT Security Assessment Procedure" for further clarification).

B. Systems Development Life Cycle is a detailed and specific set of procedures, steps, and documents that carry a project through its technical development. It includes an Initiation Phase, Planning Phase, Functional Design Phase, System Design Phase, Development Phase, Integration and Testing Phase, Installation and Acceptance Phase, and Maintenance Phase.

C. System, as referenced in this IPD, does not refer to the definition, interpretation, or meaning of "system" as defined by the Office of Management and Budget (OMB) or NIST. In its most general sense, system refers to any combination of hardware, software, and firmware.

6. Policy.

A. Program Enterprise Architects and IT Security Managers shall be notified prior to the development of any new system. This will ensure that appropriate Systems Development Lifecycle (SDLC) and security requirements are followed.

B. The MMSNet System Owner shall be notified prior to the connection of any new system to the MMS production network (MMSNet). Notification by email is adequate to meet this requirement. (This requirement does not include the connection of desktop and laptop systems for general user utilization.)

C. A security assessment shall be performed against any new system prior to connection to a production network. The security assessment shall be performed by an independent group or person. The results of the security assessment shall be reviewed to ensure:

(1) The running configuration matches the configuration defined in the System Security Plan (SSP) or System Design Specification document and the Security Technical Implementation Guide (STIG).

(2) There are no vulnerabilities associated with the system that are not already accounted for in the approved documentation and required for operational purposes.

D. Vulnerabilities discovered during the security assessment shall be remediated or mitigated prior to the connection of the system to a production network. The appropriate SSP shall be updated with all remediation and mitigation information. If any discovered vulnerabilities are part of normal operations and cannot be remediated or mitigated, this information shall be entered in the appropriate SSP, System Design Specification, STIG, Plan of Action and Milestones, and/or Risk Assessment.

E. The party with overall responsibility for the system development must certify that the following criteria have been met prior to requesting the connection of the system to a production network:

(1) Security assessment has been performed against the new system.

(2) All applicable STIGs have been applied to the new system.

(3) Security assessment has been reviewed by an independent, third party.

(4) Running configuration matches the documented configuration.

(5) Discovered vulnerabilities have been remediated, mitigated, or documented as part of normal system operations.

(6) Documentation has been updated to reflect changes made as a result of the security assessment.

F. Management shall review and approve all system security assessment reports prior to the connection of the system to a production network. Management has final approval authority of all connection requests.

G. Internet Protocol (IP) addresses shall not be issued to new systems until all the criteria set forth in the policy are met.

H. System developers shall use secure development and programming practices throughout the entire SDLC process to facilitate the availability/production of evidence to support the security assessment. Such secure development practices include source code validation and integrity verification.

I. System developers shall ensure that all appropriate STIGs have been applied. STIGs are required as outlined in the IT Security Policy Handbook.

J. System developers shall complete the maximum number of IT security role-based training hours required by the annual training standard.

K. Federal, DOI, and MMS certification and accreditation requirements and standards shall be adhered to throughout all phases of system development.

7. Responsibilities.

A. The Chief Information Officer (CIO)/Deputy Chief Information Officer (DCIO) shall:

(1) Hold final approval authority for all connection requests.

(2) Review security assessment reports in a timely manner in order to approve/deny connection requests.

B. The Bureau and Program Information Technology Security Managers (BITSM/PITSM) shall:

(1) Review security assessment reports prior to submission to CIO/DCIO and provide recommendations on course of action.

(2) Submit completed security assessment documentation to CIO/DCIO for review.

C. Program Information Technology Security Managers (PITSM) shall:

(1) Review security assessment reports prior to submission to the BITSM and provide recommendations on course of action.
(2) Submit completed security assessment documentation to BITSM for review.

D. System Owners/System Security Managers shall:

(1) Engage an Independent Group or Person in order to schedule a security assessment of the new system prior to requesting connection to a production network.

(2) Ensure that a security assessment has been performed against any new system prior to requesting connection to a production network.

(3) Ensure that the security assessment has been reviewed by an independent, third party.

(4) Ensure that discovered vulnerabilities have been remediated or mitigated. If any of the vulnerabilities are part of normal operations, all documentation reflects this fact.

(5) Ensure that all documentation has been updated to reflect changes made as a result of the security assessment.

(6) Submit completed security assessment documentation to the PITSM for review.

E. The Independent Group or Person performing the security assessment shall:

(1) Perform an in-depth security assessment as outlined in the MMS IT Security Assessment Procedure.

(2) Provide the results of the security assessment to the independent, Third Party Reviewers and the System Owner.

F. Third Party Reviewers shall:

(1) Compare the results of the security assessment with the system's SSP or System Design Specification document and the STIG.

(2) Document any vulnerabilities listed in the security assessment and the differences between the security assessment and the documented configuration.

(3) Submit completed security assessment documentation to System Owner for review.

G. System Developers shall:

(1) Ensure secure programming practices are used throughout the entire SDLC process.

(2) Ensure the appropriate STIG has been applied.

(3) Ensure all configuration settings are accurately documented.

(4) Complete the maximum number of IT security role-based training hours required by the annual training standard.

8. Cancellation. This IPD will remain in effect until incorporated into the MMS Manual, canceled, or superseded with another IPD.

Robert E. Brown
Associate Director for
Administration and Budget

Minerals Management Service

Information Technology Security Assessment Procedure

March 2008

Security Assessment Requirements

Introduction

System developers shall use secure development practices throughout the SDLC. The use of appropriate secure development practices shall support the production of evidence. The evidence produced shall demonstrate that the security assessment requirements have been met.

Examples of vendor specific secure system development practices have been provided (see Attachment 2). The list is not exhaustive. The requisite standard or best practice needed for a specific system development shall be identified and implemented as appropriate.

1.0 Software Development Requirements for ALL Systems

Applications designers and coders shall practice defensive programming in order to avoid, at a minimum, the following "Unforgivable Weaknesses" (from the Common Weakness Enumeration (http://cwe.mitre.org/)):

Unforgivable Weaknesses are defined as those that:

- Take precedence
- Are well documented
- Obvious
- Simple
- Quick to Find

CWE 120 - Buffer Overflow
CWE 79 - XSS (Cross-site Scripting)
CWE 23 - Directory transversal
CWE 98 - Remote file inclusion
CWE 89 - SQL Injection
CWE 425 - Direct Request
CWE 472 - Authorization Bypass
CWE 327 - Using a Broken or Risky Cryptographic Algorithm
CWE 271 - Privilege Dropping / Lowering Errors
CWE 61 - UNIX Symbolic Link (Symlink) Following
CWE 259 - Hard-Coded Password
CWE 190 - Integer Overflow (Wrap or Wraparound)

System developers shall practice defensive programming in order to avoid weaknesses specifically identified in design and coding standard practices issued for specific major application developments.

2.0 Assessment Requirements for All Systems

The System developers shall be able to demonstrate that the following tests have been addressed in all systems:

Technical Configuration - System developers shall verify the application of the appropriate MMS approved security technical implementation guides (STIGs), or their approved equivalents, for all Operating Systems and Major Applications used.

Port Scan - System developers shall verify that all open ports listed in the output of the port scan are reconciled/correlated to the documented ports as required in the MMS approved system's System Security Plan (SSP) and/or system design documentation.

Vulnerability Remediation - System developers shall verify that the vulnerabilities, known and/or potential system weaknesses, identified during the vulnerability assessment have been remediated or mitigated.

3.0 Assessment Requirements for All Servers

The System developers shall be able to demonstrate that the following tests have been addressed in all servers (database, web and application):

Parameter Tampering - System developers shall verify that countermeasures have been implemented correctly to address the threat of the parameter tampering, e.g. Query strings, POST parameters, and hidden fields cannot be modified in an attempt to gain unauthorized access to data or functionality.

User Privilege Escalation - System developers shall verify that countermeasures have been implemented correctly to address the threat of users attempting to gain unauthorized access to administrator or other users' privileges.

Credential Manipulation - System developers shall verify that countermeasures have been implemented correctly to address the threat of modifications being made to identification and authorization credentials in an attempt to gain unauthorized access to other users' privileges.

Session Hijacking - System developers shall verify that countermeasures have been implemented correctly to address the threat of attempts to take over a session established by another user to assume the privileges of that user.

Backdoors and Debug Options - System developers shall verify that countermeasures have been implemented correctly to address the threat of applications containing code left by developers for debugging purposes. Debugging code typically runs with a higher level of access, making it a target for potential exploitation. Application developers may leave backdoors in their code. These backdoors, if discovered, could potentially allow an intruder to gain additional level of access.

Input Validation Bypass - System developers shall verify that countermeasures have been implemented correctly to address the threat of client side validation routines and bounds-checking being removed, by ensuring that such controls are implemented on the server.

4.0 Assessment Requirements for Web Servers

The System developers shall be able to demonstrate that the following tests have been addressed for web servers:

Cookie Poisoning - System developers shall verify that countermeasures have been implemented correctly to address the threat of modified data being sent in cookies. Applications may enter an insecure state when unexpected modifications are made to data in cookies. There is a need to test the application response to receiving unexpected cookie values.

Forceful Browsing - System developers shall verify that countermeasures have been implemented correctly to address the threat of mis-configured web servers sending any file to a user, as long as the user knows the file name and the file is not protected. The countermeasures will address the hacker exploiting this security hole, and "jumping" directly to pages.

Configuration Subversion - System developers shall verify that countermeasures have been implemented correctly to address the threat of mis-configuring web servers and application servers, a very common mistake. The most common mis-configuration is one that permits directory browsing. Hackers can utilize this feature in order to browse the application's directories (such as cgi-bin/) by simply typing in the directory name.

SQL Injection - System developers shall verify that countermeasures have been implemented correctly to address the threat of specially crafted SQL commands being submitted in input fields. Need to ensure validation of user input type controls.

Cross-Site Scripting - System developers shall verify that countermeasures have been implemented correctly to address the threat of active content being submitted to the application in an attempt to cause a user's web browser to execute unauthorized code. Need to ensure validation of user input type controls.

Buffer Overflow - System developers shall verify that countermeasures have been implemented correctly to address the threat of “buffer overflow”. A “buffer overflow” is an anomalous condition where a process attempts to store data beyond the boundaries of a fixed-length buffer. The result is that the extra data overwrites adjacent memory locations. The overwritten data may include other buffers, variables and program flow data and may cause a process to crash, produce incorrect results or enter an insecure state. Buffer overflows can be triggered by inputs specifically designed to execute malicious code or to make the program operate in an unintended, insecure, way.

HTTP Response Splitting - System developers shall verify that countermeasures have been implemented correctly to address the threat of HTTP splitting. HTTP splitting results from the failure of the application or its environment to properly sanitize input values and can be used to perform cross-site scripting attacks, cross-user defacement, Web cache poisoning, and similar exploits.

Hidden Field Manipulation - System developers shall verify that countermeasures have been implemented correctly to address the threat of exploiting Hidden fields. Hidden fields are embedded within HTML forms to maintain values that will be sent back to the server. Such

hidden fields serve as a mean for the web application to pass information between different parts of one application or between different applications.

Stealth Commanding - System developers shall verify that countermeasures have been implemented correctly to address the threat of using a set of techniques which allow attackers to exploit parsing problems in server-side scripts in order to change the code executed by the server.

LDAP Injection - System developers shall verify that countermeasures have been implemented correctly to address the threat of attack that employs the compromise of web sites that construct LDAP (Lightweight Directory Access Protocol) statements from data provided by users. This exploit changes LDAP statements in order for dynamic web applications to run with invalid permissions, allowing the attacker to alter, add or delete content.

5.0 Assessment Requirements for Application Servers

The System developers shall be able to demonstrate that the following tests have been addressed:

Configuration Subversion - System developers shall verify that countermeasures have been implemented correctly to address the threat of mis-configuring web servers and application servers, a very common mistake. The most common mis-configuration is one that permits directory browsing. Hackers can utilize this feature in order to browse the application's directories (such as cgi-bin/) by simply typing in the directory name.

Buffer Overflow - System developers shall verify that countermeasures have been implemented correctly to address the threat of "buffer overflow". A "buffer overflow" is an anomalous condition where a process attempts to store data beyond the boundaries of a fixed-length buffer. The result is that the extra data overwrites adjacent memory locations. The overwritten data may include other buffers, variables and program flow data and may cause a process to crash, produce incorrect results or enter an insecure state. Buffer overflows can be triggered by inputs specifically designed to execute malicious code or to make the program operate in an unintended, insecure, way.

Stealth Commanding - System developers shall verify that countermeasures have been implemented correctly to address the threat of using a set of techniques which allow attackers to exploit parsing problems in server-side scripts in order to change the code executed by the server.

LDAP Injection - System developers shall verify that countermeasures have been implemented correctly to address the threat of attack that employs the compromise of web sites that construct LDAP (Lightweight Directory Access Protocol) statements from data provided by users. This exploit changes LDAP statements in order for dynamic web applications to run with invalid permissions, allowing the attacker to alter, add or delete content.

6.0 Assessment Requirements for Application Development

The System developers shall be able to demonstrate that the following has been complied with.

All code developed shall follow the MMS Solution Life Cycle guidance. All code development projects shall have a security quality assurance process. The QA process will define code review cycles during the process and ensure that appropriate security tests are performed at each point in the cycle where such tests shall be defined as required.

Secure development practices shall be employed at all times by the project team, assessed by the project QA team and tested by the program IT security branch for compliance. Actual practices shall be defined by the QA team, but shall include efforts to mitigate common vulnerabilities found in many web applications. The efforts shall include the items listed here, with the understanding that the list is not meant to be complete or that it cannot be further refined by the QA team, the security branch, or information that comes to light in the future.

- Access controls must be strictly defined and enforced.
- Account and session management tokens must be protected and replaced when they are compromised or appear to be malfunctioning.
- All user input shall be validated to prevent buffer overflows. Any component which may be vulnerable to invalid input shall be assessed during the QA process and returned to the development staff when any defects are discovered.
- Many web applications pass parameters and data between modules. This type of data must be validated before it is accepted by another part of the application in order to prevent command injection attacks, cross-site scripting, or error conditions that may provide information to an attacker.
- Any encryption method used must comply with the appropriate government standard currently in force. The methods shall be tested in depth by the security branch to ensure that the session and data cannot be examined by an unauthorized agent.
- Remote administration of a web server or a web application shall not be used unless there is a risk analysis performed on the process and the risk of such administration is accepted by the system owner and the designated approving authority.
- Any server used to host a web application must conform to the standards and testing processes listed above in this document
- Underlying databases used by a web application shall be tested for security deficiencies as part of life cycle management. The QA team and security branch will define this cycle in the documentation for the QA process.
- All application developers shall obtain periodic training in writing secure code.
- Quality assurance team members shall obtain periodic training in code analysis with a focus on the security of the code.
- Security branch staff shall obtain periodic training in testing code, applications, databases, and servers for conformance with the policies of MMS and for secure application and system development.

Secure Development & Configuration Best Practices

Examples of specific vendor secure system development practices have been provided (see below). The list is not exhaustive. The requisite standard or best practice needed for a specific system development shall be identified by the system developers and used as appropriate.

There are a number of governmental initiatives from which additional information and status on the available standards and best practices for secure development can be obtained.

Government Development & Secure Configuration Best Practices

Department Homeland Security (DHS)
Build Security In,
https://buildsecurityin.us-cert.gov/daisy/bsi/home.html

National Institute for Standards (NIST)
National Checklist Program Repository
http://checklists.nist.gov/ncp.cfm?repository

Vendor Specific Development & Secure Configuration Best Practices

Apache HTTP Server
Security Tips for Apache HTTP Server Configuration v1.3
http://httpd.apache.org/docs/1.3/misc/security_tips.html

Security Tips for Apache HTTP Server Configuration v2.0
http://httpd.apache.org/docs/2.0/misc/security_tips.html

Microsoft ActiveX
ActiveX Security: Improvements and Best Practices
http://msdn2.microsoft.com/en-us/library/bb250471.aspx

Microsoft Internet Information Server
Security Guidance for IIS
http://www.microsoft.com/technet/security/prodtech/iis.mspx

Microsoft SQL
Microsoft SQL Security
http://www.microsoft.com/sql/technologies/security/default.mspx

SQL Server 2005 Security Overview for Database Administrators
http://download.microsoft.com/download/4/7/a/47a548b9-249e-484c-abd7-29f31282b04d/SQLSecurityOverviewforAdmins.doc

SQL Server 2005 Security Best Practices - Operational and Administrative Tasks
http://download.microsoft.com/download/8/5/e/85eea4fa-b3bb-4426-97d0-7f7151b2011c/SQL2005SecBestPract.doc

Oracle Databases
Oracle Database Security
http://www.oracle.com/technology/deploy/security/database-security/index.html

A Security Checklist for Oracle9i
http://www.oracle.com/technology/deploy/security/oracle9i/pdf/9i_checklist.pdf

Secure Configuration Guide for Oracle9iR2
http://www.oracle.com/technology/deploy/security/oracle9i/pdf/9ir2_checklist.pdf

Oracle Database Security Checklist for 10g & 11g
http://www.oracle.com/technology/deploy/security/database-security/index.html

Sun Java
Java SE Security
http://java.sun.com/javase/technologies/security/index.jsp

Secure Coding Guidelines for the Java Programming Language, version 2.0
http://java.sun.com/security/seccodeguide.html