APPLICATION SECURITY AND DEVELOPMENT

SECURITY TECHNICAL IMPLEMENTATION GUIDE

Version 3, Release 10

23 January 2015

Developed by DISA for the DoD

This page is intentionally blank.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

TABLE OF FIGURES

SUMMARY OF CHANGES

GENERAL CHANGES

The previous release was Version 3, Release 9, dated 24 October 2014.

APP2060: The original Sun Java style guide reference is no longer available, guidance updated to reference the Google Java style guide instead.

Checklist Changes

APP3305: Removed references to old SRR.

This page is intentionally blank.

1. INTRODUCTION

1.1 Background

This Application Security and Development Security Technical Implementation Guide (STIG) provides security guidance for use throughout the application development lifecycle. This STIG provides the guidance needed to promote the development, integration, and updating of secure applications. Subjects covered in this document are: Development, design, testing, conversions and upgrades for existing applications, maintenance, software configuration management, education, and training. Defense Information Systems Agency (DISA) encourages sites to use these guidelines as early as possible in the application development process. Some vulnerabilities may require significant application changes to correct. The earlier the STIG requirements are integrated into the development lifecycle, the less disruptive the remediation process will be.

1.2 Authority

DoD Instruction (DoDI) 8500.01 requires that "all IT that receives, processes, stores, displays, or transmits DoD information will be […] configured […] consistent with applicable DoD cybersecurity policies, standards, and architectures" and tasks that Defense Information Systems Agency (DISA) "develops and maintains control correlation identifiers (CCIs), security requirements guides (SRGs), security technical implementation guides (STIGs), and mobile code risk categories and usage guides that implement and are consistent with DoD cybersecurity policies, standards, architectures, security controls, and validation procedures, with the support of the NSA/CSS, using input from stakeholders, and using automation whenever possible." This document is provided under the authority of DoDI 8500.01.

Although the use of the principles and guidelines in these SRGs/STIGs provide an environment that contributes to the security requirements of DoD systems, applicable NIST SP 800-53 cybersecurity controls need to be applied to all systems and architectures based on the Committee on National Security Systems (CNSS) Instruction (CNSSI) 1253.

1.3 Scope

This document is a requirement for all DoD developed, architected, and administered enterprise applications and systems connected to DoD networks. The STIG is not intended to be applied to firewalls or other network devices with application management interfaces when a more relevant STIG for that product already exists. These requirements assist Application Development Program Managers, Application Designers, Release Managers, Security Managers (SMs), Information Assurance Managers (IAMs), IAOs, and System Administrators (SAs) with configuring and maintaining security controls for applications. The requirements listed can be used to evaluate custom-developed applications and Commercial-Off-The-Shelf (COTS) software packages as well. Some requirements are not applicable to COTS or GOTS software packages. Requirements that are not applicable to COTS or GOTS are specifically identified as being not applicable.

Writing Conventions: Throughout this document, statements are written using words such as "**will**" and "**should**". The following paragraphs are intended to clarify how these STIG statements are to be interpreted.

A reference that uses "**will**" indicates mandatory compliance. All requirements of this kind will also be documented in the italicized policy statements in bullet format, which follow the topic paragraph. This makes all "**will**" statements easier to locate and interpret from the context of the topic. The IAO will adhere to the instructions as written.

For each italicized policy bullet, the text will be preceded by parentheses containing the STIG Identifier (STIGID), which corresponds to an item on the checklist and the severity code of the bulleted item. An example of this will be as follows: "(*G111: CAT II*)". Throughout the document, accountability is directed to the IAO to "ensure" a task is carried out or monitored. These tasks may be carried out by the IAO or delegated to someone else as a responsibility or duty.

A reference to "**should**" indicates a recommendation that further enhances the security posture of the site. These recommended actions will be documented in the text paragraphs but not in the italicized policy bullets. All reasonable attempts to meet this criterion will be made.

1.4 Vulnerability Severity Category Code Definitions

Severity Category Codes (referred to as CAT) are a measure of vulnerabilities used to assess a facility or system security posture. Each security policy specified in this document is assigned a Severity Code of CAT I, II, or III.

Table 1-1. Vulnerability Severity Category Code Definitions

	DISA Category Code Guidelines
CAT I	Any vulnerability, the exploitation of which will, **directly and immediately** result in loss of Confidentiality, Availability, or Integrity.
CAT II	Any vulnerability, the exploitation of which **has a potential** to result in loss of Confidentiality, Availability, or Integrity.
CAT III	Any vulnerability, the existence of which **degrades measures** to protect against loss of Confidentiality, Availability, or Integrity.

1.5 STIG Distribution

Parties within the DoD and Federal Government's computing environments can obtain the applicable STIG from the DoD IA Portal STIG page or Information Assurance Support Environment (IASE) web site. These sites contain the latest copies of any STIG, as well as checklists, scripts, and other related security information. The Uniform Resource Locator (URL)

for the DoD IA Portal STIG page is https://www.us.army.mil/suite/page/397960 and the URL for the IASE web site is http://iase.disa.mil/stigs/Pages/index.aspx.

1.6 Document Revisions

Comments or proposed revisions to this document should be sent via e-mail to disa.letterkenny.FSO.mbx.stig-customer-support-mailbox@mail.mil. DISA FSO will coordinate all change requests with the relevant DoD organizations before inclusion in this document.

1.7 Document Overview

This guide is intended to assist in the design, development, testing, deployment, and maintenance of secure applications. As operating system security postures continue to improve, attackers are turning their attention toward application vulnerabilities, increasing the need for secure application development practices.

This guide may be used for both in-house application development and to assist in the evaluation of the security of third-party applications. The guidance provided is not specific to any one platform, programming language, or application type. Some sections of this guide may not apply to all applications. In some cases, specific guidance has been provided based upon platform, programming language, or language types. The presence of specific guidance for a technology does not exempt applications utilizing other technologies from a requirement.

When using this guide to evaluate third-party products, some sections may not be applicable. Answers to some questions will need to be gathered through discussions with vendor representatives or from public information about the company.

For best results, the recommendations provided in this guide should be implemented as early in the development process as possible. When developing applications for the DoD, third-party developers should be encouraged to integrate these procedures into their lifecycle. DoD STIGS are available for existing third-party products, such as databases, web servers, web application servers, and desktop applications. Third-party product STIGs, used in conjunction with Operating System and Network STIG, provide a basis for a secure development baseline. DoD STIGS can be found at the DoD IA Portal web site (https://www.us.army.mil/suite/page/397960) or on the IASE web site (http://iase.disa.mil/stigs/index.html).

1.8 Document Organization

This document divides security requirements by roles found in the Software Development Life Cycle (SDLC) and is organized as follows:

- Section 1 provides document overview and background information.
- Section 2 is devoted to Program Management requirements. These requirements focus on the Program Manager in charge of the software development team.
- Section 3 focuses on design and development requirements geared toward the development team. The development team is responsible for the secure design of the application.

- Section 4 concentrates on Software Configuration Management (SCM) and the requirements needed for the Software Release Manager. The Software Release Manager is responsible for the builds and baselines of the deployed system.
- Section 5 deals with testing requirements for the Test Manager. The Test Manager is responsible for the security testing of the application. The Test Manager will create and use Test Plans and procedures to perform required security testing.
- Section 6 deals with deployment requirements for the IAO. The IAO may not perform the daily security analysis that may be required, but instead assign an SA or other individuals to perform these duties. Ultimately, the IAO is responsible for the security of the deployed application.

2. PROGRAM MANAGEMENT

This section describes the IA-related activities required of the Program Manager as the application progresses through the SDLC.

The Program Manager will follow the security guidance from the Defense Acquisition Guidebook that can be found at the following web site: https://akss.dau.mil/dag/DoD5000.asp?view=document&rf=GuideBook\IG_c4.4.4.asp.

A summary of the best practices for program management of software systems is as follows:

- Select individuals with domain experience in developing comparable software systems, with successful past performance, and with a mature SDLC
- Acquire the documentation for target production, test, and development platforms
- Plan for support and deployment of the software into a production environment
- Plan for emerging and evolving technologies
- Use COTS products, Government-Off-The-Shelf (GOTS) products, and reusable components before developing new software
- Acquire COTS and GOTS products and support services prior to design phase
- Track COTS software purchases and maintenance licenses
- Acquire SDLC tools, such as development environments, configuration management tools, requirements analysis tools, testing tools, and source code analyzers
- Use open system concepts and a modular design
- Select the programming language based on system performance, life-cycle costs, risks, and interoperability
- Use DoD standard data and follow data administrative policies in DoDD 8320.2
- Assess information operations risks (see DoDD S-3600.1) using techniques such as independent expert reviews

In addition, the Program Manager should apply the following security considerations to software design and management (see DoDD 5000.2):

- A documented impact analysis statement addressing software reliability must accompany any modifications to existing DoD software. This is to ensure that any potential reliability issues arising from the modification effort are identified and addressed
- Establish a formal software change control process, ensure the change control process indicates whether foreign nationals, in any way, participated in software development, modification, or remediation
- Select software quality assurance personnel to monitor the software change process
- Use an independent verification and validation team
- Analyze the technical risks and vulnerabilities as well as creation of a threat model
- Verify each foreign national employed by contractors/subcontractors to develop, modify, or remediate software code specifically for DoD use has a security clearance commensurate with the level of the program in which the software is being used

- Verify primary vendors on DoD contracts having subcontractors who employ cleared foreign nationals work only in a certified or accredited environment (DoD Information Assurance Certification and Accreditation Process (DIACAP))
- Verify DoD software with coding done in foreign environments or by foreign nationals is reviewed for malicious code by software quality assurance personnel
- Give preference during product selection and evaluation ,when employing COTS software, to those vendors who can demonstrate that they took efforts to minimize the security risks associated with foreign nationals who develop, modify, or remediate the COTS software being offered
- Ensure software quality assurance personnel review software sent to locations not directly controlled by the DoD or its contractors for malicious code when it is returned to the DoD contractor's facilities

2.1 Documentation

The Program Manager has oversight responsibility for creating the System Security Plan (SSP), Application Configuration Guide and Classification Guide, and Coding Standards, if applicable. The Program Manager is responsible for the communication of all system documentation to the application deployment personnel. The Program Manager should designate a lead developer to handle security and security-related issues.

2.1.1 System Security Plan (SSP)

The Program Manager is required to produce an SSP. These documents provide an overview of the security requirements of the application.

The SSP will include the following:

1. Technical, administration, and procedural IA program policies
2. Designation of IA personnel
3. Identification of the IA requirements
4. IA objectives (e.g., requirements for data handling or dissemination, system redundancy and backup, or emergency response)

The Program Manager will document the following:

1. Appointments to required IA roles (e.g., Authorizing Official (AO) IAM/IAO, etc.)
2. Assigned duties
3. Appointment criteria (e.g., training, security clearance, information technology (IT) designation, etc.)

(APP2010.1: CAT II) The Program Manager will ensure an SSP is established describing the technical, administrative, and procedural IA program and policies governing the DoD information system, and identifying all IA personnel and specific IA requirements and objectives.

(APP2010.2: CAT II) The Program Manager will ensure all appointments to required IA roles are established in writing to include assigned duties and appointment criteria, such as training, security clearance, and IT designation.

2.1.2 Application Configuration Guide

The overall responsibility for the development of the Application Configuration Guide belongs to the Program Manager. The Application Configuration Guide documents the secure deployment of the application being developed. This is critical information to the application hosting providers. The Program Manager will be the liaison between the application hosting providers and the development team and provide system documentation as needed.

The Program Manager may designate an application as critical in the Application Configuration Guide. If designated as a critical application, the application will not be hosted on a general purpose server with other applications.

The Program Manager will identify the list of all potential hosting enclaves in the Application Configuration Guide. The Application Configuration Guide shall also include any interconnected outsourced IT-based processes and interconnected IT platforms with connection rules and requirements.

(APP2020.1: CAT II) The Program Manager will provide an Application Configuration Guide to the application hosting providers.

(APP2020.2: CAT II) The Program Manager will provide a list of all potential hosting enclaves and connection rules and requirements.

2.1.2.1 Standardized Environments

Standard environment settings establish a secure baseline upon which applications are designed to function within the target secured environment. The standardized environments should include each development system, build system, and test system. Standardization should include the software load, system configuration, and network configuration. In addition, all DoD environments and systems will be configured to comply with all appropriate DoD STIGS.

By standardizing the software, any configuration settings interfering with the application can be identified. If the application cannot function with a certain configuration setting enabled, then, this fact, as well as an analysis of any potential risks, should be added to the Application Configuration Guide.

(APP2020.3: CAT II) The Program Manager will ensure development systems, build systems, and test systems have a standardized environment and are documented in the Application Configuration Guide.

2.1.3 Security Classification Guide

The Program Manager will create a Security Classification Guide if the system contains classified information. This guide should document the data elements and their classifications.

(APP2040.1: CAT II) The Program Manager will ensure a Security Classification Guide exists containing data elements and their classifications if the system contains classified information.

2.1.4 Mission Assurance Category and Confidentiality

The Program Manager will determine the MAC and Confidentiality levels for the application, as shown in Tables 2-1 and 2-2. For example: *This application was designed to operate on MAC II or MAC III systems and handle publicly releasable information only.*

Table 2-1. MAC Levels for IA Control

MAC Levels for IA Controls			
MAC	**Definition**	**Integrity**	**Availability**
1	These systems handle information that is determined to be vital to the operational readiness or mission effectiveness of deployed and contingency forces in terms of both content and timeliness.	High	High
2	These systems handle information that is important to the support of deployed and contingency forces.	High	Medium
3	These systems handle information that is necessary for the conduct of day-to-day business, but does not materially affect support to deployed or contingency forces in the short-term.	Basic	Basic

Table 2-2. Confidentiality Levels for IA Control

Confidentiality Levels for IA Control	
Confidentiality Level	**Definition**
Classified	Systems processing classified information.
Sensitive	Systems processing sensitive information as defined in DoDD 8500.01, to include any unclassified information not cleared for public release.
Public	Systems processing publicly releasable information as defined in DoDD 8500.01 (i.e., information that has undergone a security review and been cleared for public release).

(APP2050: CAT II) The Program Manager will ensure the system has been assigned specific MAC and Confidentiality levels.

2.1.5 Coding Standards

Implementing coding standards provides many benefits to the development process. These benefits include readability, consistency, and ease of integration.

Code conforming to a standard format is easier to read, especially if someone other than the original developer is examining the code. In addition, formatted code can be debugged and corrected faster than unformatted code.

Introducing coding standards can help increase the consistency, reliability, and security of the application by ensuring common programming structures and tasks are handled by similar methods, as well as, reducing the occurrence of common logic errors.

Coding standards also allow developers to quickly adapt to code which has been developed by various members of a development team. Coding standards are useful in the code review process as well as in situations where a team member leaves and duties must then be assigned to another team member. Coding standards often cover the use of white space characters, variable naming conventions, function naming conventions, and comment styles.

Many organizations have existing coding standards that can be tailored for individual projects. Table 2-3 provides some sample coding standards and where they can be found.

Table 2-3. Sample Coding Standards

Language	URL
Java	http://google-styleguide.googlecode.com/svn/trunk/javaguide.html
C# Visual Basic.Net	http://submain.com/?nav=products.guidelines
C++	http://www.possibility.com/Cpp/CppCodingStandard.html
Python	http://www.python.org/dev/peps/pep-0008/

The Program Manager is responsible for obtaining or creating a set of coding standards to be followed by the development team during the SDLC.

(APP2060.1: CAT II) The Program Manager will ensure the development team follows a set of coding standards.

2.1.5.1 Unsafe Functions

The Coding Standards documentation should also include a list of known unsafe functions. The Program Manager will designate a team member to maintain a list of unsafe functions. Unsafe functions are identified as presenting some risk to the application and should, therefore, not be used. Appendix B includes examples of functions to avoid for C and C++ programming languages. Automated checks shall be utilized to identify where these functions are used and to alert developers of their presence. For example, in Microsoft Visual C++, warning C4996 indicates an unsafe function call.

(APP2060.2: CAT II) The Program Manager will ensure the development team creates a list of unsafe functions to avoid and document this list in the coding standards.

2.2 Third-Party Software Products

Third-party software products and libraries may be used in the development of an application. These products and libraries may consist of database management systems, reporting tools, web servers, open source libraries, or a variety of other software products. If the application relies on software products or libraries outside the control of the development team, it is the Program Manager's responsibility to ensure the components comply with the requirements outlined in this document.

2.2.1 NIAP-Approved Products

Components being used to provide IA or IA-enabled functionality will, in addition to meeting all the requirements specified in this document, be evaluated in accordance with the NIAP approval process. For information on the National Information Assurance Partnership (NIAP) approval process or NIAP-approved products, refer to the following web site: http://www.niap-ccevs.org/cc-scheme/vpl/. These products have been evaluated and validated in accordance with the provisions of the NIAP Common Criteria Evaluation and Validation Scheme.

(APP2070.1: CAT III) The Program Manager will ensure any IA or IA-enabled products used by the application are NIAP approved or in the NIAP approval process.

IA and IA-enabled product definitions are provided from the DoD Instruction (DoDI) 8500.2, Information Assurance (IA) Implementation, dated February 6, 2003.

- IA product: Product or technology whose primary purpose is to provide security services (e.g., confidentiality, authentication, integrity, access control, or non-repudiation of data); correct known vulnerabilities; and/or provide layered defense against various categories of non-authorized or malicious penetrations of information systems or networks. Examples include such products as data/network encryptors, firewalls, and intrusion detection devices.
- IA-enabled product: Product or technology whose primary role is not security, but which provides security services as an associated feature of its intended operating capabilities. Examples include such products as security-enabled web browsers, screening routers, trusted operating systems, and security-enabled messaging systems.

2.2.2 Protection Profiles

Third-party software must meet security NSA-NIAP requirements based on the sensitivity of the data.

The requirements for products are defined in the Protection Profile Consistency Guidance for Robustness published under the Information Assurance Technical Framework (IATF).

(APP2080.1: CAT II) The Program Manager will ensure COTS IA and IA-enabled products, which are used to protect publicly released information, comply with National Security Agency (NSA)-endorsed Protection Profiles.

(APP2080.2: CAT II) The Program Manager will ensure COTS IA and IA-enabled products which are used to protect sensitive information when the information transits non DoD-owned networks, or the system handling the information is accessible by individuals who are not authorized to access the information on the system, comply with NSA-NIAP approved Protection Profiles.

(APP2080.3: CAT II) The Program Manager will ensure COTS IA and IA-enabled products which are used to protect classified information when the information transits networks, which are at a lower classification level than the information being transported, comply with NSA-NIAP approved Protection Profiles.

More information can be found about Protection Profiles at the following web site: http://www.niap-ccevs.org/cc-scheme/pp/index.cfm.

2.2.3 Categories of Third-Party Products

There are many classes of software that contain differing licensing, distribution agreements, and warranties.

The DoD memo entitled "Clarifying Guidance Regarding Open Source Software (OSS)" dated October 16, 2009 defines Open Source Software (OSS) and provides various clarifications. It can be found at: http://cio-nii.defense.gov/sites/oss/

Third-party products may fall into many categories as defined below:

- OSS: Software for which the human-readable source code is available for use, study, reuse, modification, enhancement, and redistribution by the users of that software
- Public domain software: Software not protected by any copyright laws providing the right to use, modify, and redistribute without permission or payment to the author.
- Shareware: Copyrighted software distributed under a license that provides a trial right to use and redistribute the binaries. For continued usage, users are required to pay a fee.
- Freeware: Copyrighted software distributed under a license that provides rights to use and redistribute the binaries. Unlike shareware, there is no charge for continued use.
- Commercial software: Includes software that is of the type customarily used by the general public or by nongovernmental entities for purposes other than governmental purposes and that has been sold, leased, or licensed to the general public, or has been offered for sale, least, or license to the general public.

Software products and libraries with both (1) no source code to review, repair, and extend, and (2) limited or no warranty, will not be used in DoD information systems unless they are necessary for mission accomplishment and there are no alternative IT solutions available. If these products are required, they must be assessed against information assurance requirements.

The product assessment addresses the risk of deploying software products with both (1) no source code to review, repair, and extend, and (2) limited or no warranty, because of difficulties patching or fixing software without having access to the original source or a vendor actively maintaining the software product.

Software products and libraries with both (1) no source code to review, repair, and extend, and (2) limited or no warranty, will not be used in DoD information systems unless they are necessary for mission accomplishment and there are no alternative IT solutions available. If these products are required, they must be assessed for IA impacts and be approved for use by the AO. The product assessment addresses the risk of deploying software products with both (1) no source code to review, repair, and extend, and (2) limited or no warranty , because of difficulties patching or fixing software without having access to the original source or a vendor actively maintaining the software product. Software products and libraries not specifically part of, or deployed with the application, are not required to be approved by the AO.
These products include compilers, testing tools, Configuration Management tools, and other products and libraries part of the application or deployed with the application. However, a risk assessment should be performed on the product or libraries to mitigate potential vulnerabilities.

(APP2090.1: CAT II) The Program Manager will obtain AO acceptance of risk for all public domain, shareware, freeware, and other software products/libraries with both (1) no source code to review, repair, and extend, and (2) limited or no warranty, but are required for mission accomplishment.

Since OSS has source code available, this category of software can be maintained for security fixes and patched for known vulnerabilities. The Program Manager can elect to maintain OSS with the same security requirements as other custom-developed application components.

2.3 Education and Training

Education should be provided to all levels of management, including Team Leads, Program Managers, and upper management.

A critical step in improving application security is educating managers, designers, developers, and testers. This education should focus on the security aspects of the development process relevant to an individual. It is possible individuals may fall into more than one of the listed categories. In such cases, those individuals should receive the training appropriate for all of their job functions.

Additional IA training requirements can be found in DoDD 8570.01.

2.3.1 Management

Program Managers need to fully support the effort of secure application development. Without the full support of management, this guide may be implemented in a chaotic fashion. It is important Program Managers realize implementing security into the development process will impact the development schedule. Program Managers must be willing to factor the additional

time needed into the development schedule as well as be willing to delay the release, or even remove functionality, if it cannot be implemented in a secure manner. An application release should be delayed if the risk level of known vulnerabilities is greater than the need for the deployment of a mission-critical application.

(APP2120.1: CAT II) The Program Manager will ensure all levels of program management receive security training regarding the necessity, impact, and benefits of integrating secure development practices into the development lifecycle.

2.3.2 Designers

Designers are responsible for decisions ranging from target operating system selection to the functional specification of program features. Since vulnerabilities often appear as a result of design decisions, designers should be educated about secure design practices. Since the application vulnerability landscape is continuously evolving, designers should be provided with, at a minimum, annual training to cover newly discovered vulnerability types and to review secure design practices.

(APP2120.2: CAT II) The Program Manager will ensure designers are provided training on secure design principles for the entire SDLC and newly-discovered vulnerability types on at least an annual basis.

2.3.3 Developers

Developers are responsible for implementing the application design as described by the designer. In order to implement the design securely, developers need to be educated in secure coding practices. This education should include an overview of known security flaw types, methods of identifying code vulnerabilities, and methods of correcting code vulnerabilities.

Developers should be provided with updated security training on at least an annual basis. This training should review the material presented in the initial training session and provide information on newly-discovered vulnerability types.

(APP2120.3: CAT II) The Program Manager will ensure developers are provided with training on secure design and coding practices on at least an annual basis.

2.3.4 Testers

Testers are responsible for ensuring the application meets the established requirements and functions correctly. Both functional testing and security testing should be performed during the testing phase of the software lifecycle.

Testers should be provided with training to enable them to formulate tests and data sets for the application. Training allows them to identify potential security flaws which are necessary to create and execute test plans and procedures. This training should be updated and provided at least once a year.

(APP2120.4: CAT II) The Program Manager will ensure testers are provided training on at least an annual basis.

2.4 Application Maintenance

2.4.1 Security Incident Response Process

The incident response process should be put in place to respond to reports of security flaws in the application. This process should include a method for individuals to submit potential security vulnerabilities to the development or maintenance team. The process should then dictate what is to be done with the reported vulnerabilities. Reported vulnerabilities must be tracked throughout the process to ensure they are triaged, corrected, and tested. The corresponding update is released to the user community. Also, the user community is notified of the availability of the application update.

(APP2140.1: CAT II) The Program Manager will ensure a security incident response process for the application is established that defines reportable incidents and outlines a standard operating procedure for incident response.

2.4.2 Vulnerability Management Process

An important part of the maintenance phase of an application is managing vulnerabilities for updated versions of the application software discovered after the application was released.

(APP2130.1: CAT II) The Program Manager will ensure users are provided with a means of obtaining updates for the application.

When a security flaw is discovered in an application deployed in a production environment, notification to the user community must take place as quickly as possible. This notification should be planned for by the Program Manager in the design phase of the application. This notification should be a warning of any potential risks to the application or data. A notification mechanism will be established to notify users of the vulnerability and the potential risks, the availability of a solution, and/or potential mitigations reducing risks to the application.

(APP2130.2: CAT II) The Program Manager will ensure a mechanism is in place to notify users of security flaws and to provide users with the availability of patches.

(APP2130.3: CAT II) The Program Manager will ensure a comprehensive vulnerability management process, including systematic identification and mitigation of software vulnerabilities is in place.

2.4.3 Unsupported Software

As long as the application is supported, maintenance must be available to address any security flaws discovered in the application. New software vulnerability types are discovered on a regular basis, and it is always possible an application previously having no known exploits will have a vulnerability discovered.

The maintenance team does not need to be an active team devoted to the application, but resources must be available to address the flaw in a timely and efficient manner. If maintaining the application is no longer possible, users will be informed when maintenance will be discontinued so they may migrate to another solution.

(APP2135: CAT I) The Program Manager will ensure all products are supported by the vendor or the development team.

2.5 Workplace Security Procedures

The Program Manager is responsible for developing procedures that assure sensitive, classified, For Official Use Only (FOUO), and other data classification not included, meet the physical data handling and storage requirements.

These procedures should include:

1. End-of-day security checks
2. Unannounced security checks
3. Imposition of a two-person rule within the computing facility (where appropriate)

(APP2150.1: CAT II) The Program Manager will ensure procedures are implemented to assure physical handling and storage of information is in accordance with the data's sensitivity.

2.6 Compliance with DoD Standards

If a DoD STIG is available for one or more components of an application, its guidance will be applied. In addition, security guidance developed for deployment of the application will be applied as indicated in the Application Configuration Guide. The Program Manager should designate a security lead to address security issues found during application development.

(APP2160.1: CAT II) The Program Manager will ensure development systems, build systems, test systems, and all components comply with all appropriate DoD STIGS, NSA guides, and all applicable DoD policies.

DoD STIGs are available at the following web sites:

DoD IA Portal STIG Page – https://www.us.army.mil/suite/page/397960
DoD IASE – http://iase.disa.mil/stigs/index.html

NSA guidance is available at the following web site:

http://www.nsa.gov/ia/guidance/index.shtml

3. DESIGN & DEVELOPMENT

This section of the document illustrates design-related security requirements. The security-specific application requirements for this section are incorporated by the Application Designers. The high-level application security requirements are handled by cooperation between the Program Manager and the Designer, with the Program Manager having the ultimate decision-making authority.

3.1 Documentation

3.1.1 Design Document

The Designer is primarily responsible for the development and updates to the application design document. The Design Document needs to cover many aspects of the application design but should also document the minimal security requirements for sensitive information, external interfaces, roles, access for the roles defined, and any unique security requirements. If the application provides mission and business essential functions, these functions should be identified and prioritized for restoration for continuity planning purposes.

(APP3010: CAT II) The Designer will create and update the Design Document for each release of the application identifying the following:

- *all external interfaces (from the threat model)*
- *the nature of information being exchanged*
- *categories of sensitive information processed or stored and their specific protection plans*
- *the protection mechanisms associated with each interface*
- *user roles required for access control*
- *access privileges assigned to each role*
- *unique application security requirements*
- *categories of sensitive information processed or stored and specific protection plans (e.g., Privacy Act, Health Insurance Portability and Accountability Act (HIPAA), etc.)*
- *restoration priority of subsystems, processes, or information*

3.1.2 Application Configuration Guide

The developers will provide documentation detailing security controls, best practices, mitigations or concerns. The requirements and recommendations will be documented in the Application Configuration Guide and must be part of the deployment plan.

Any special configuration requirements needed for the application to run will be documented and analyzed for security implications. If a required configuration setting affects the security posture of the system, these configuration settings will be documented in the Application Configuration Guide and provided to the hosting providers of the application. These configuration settings are provided so any risks can be mitigated by taking additional precautions.

(APP2020.4: CAT II) The Designer will ensure known security assumptions, implications, system-level protections, best practices, and required permissions are documented in the Application Configuration Guide.

(APP2020.5: CAT II) The Designer will ensure deployment configuration settings are documented in the Application Configuration Guide.

3.1.3 Threat Model

Threat modeling is the process of identifying potential threats to the application, risk ranking these threats, and selecting appropriate countermeasures or mitigations for the threats. Threat modeling is a critical step in securing an application from attack.

The threat model will be reviewed for each application release and updated as required to reflect the changes in application design and functionality. As potential threats are discovered and the implementation details of the application become known, the threat model will be updated.

The countermeasures identified throughout this process will be implemented according to the guidance provided in subsequent sections of this document.

(APP3020.1: CAT II) The Designer will ensure threat models are documented and reviewed for each application release and updated as required by design and functionality changes or new threats are discovered.

(APP3020.2: CAT II) The Designer will identify potential mitigations to identified threats.

(APP3020.3: CAT II) The Designer will ensure appropriate mitigations are implemented to threats based on their risk analysis.

The following section provides a high-level overview of one of the many methods of performing threat modeling. The threat modeling process is described in the book "The Security Development Lifecycle" by Michael Howard and Steve Lipner. There are many other excellent references available on the process of threat modeling, and it is recommended the reader of this guide refer to one of these references for more in-depth coverage of the topic. The Microsoft Developers Network Blog (http://blogs.msdn.com/sdl/archive/tags/threat+modeling/default.aspx) provides useful information about threat modeling and techniques.

3.1.3.1 Define Common Usage

The Designer will document the most common usage scenarios for the application and the user types. This helps drive appropriate mitigations.

3.1.3.2 Identify External Dependencies

External dependencies are defined as anything outside the control of the development team and which the application depends upon. Examples of external dependencies include, but are not limited to:

- Other computer systems
- Databases
- External data feeds or telemetry

When enumerating external dependencies, specify the version numbers and configuration requirements of the dependencies.

3.1.3.3 Enumerate Security Assumptions

A security assumption is an assumption made about the operating environment of the application for the secure operation of the product. These assumptions may include, but are not limited to:

- Operating system or application configurations. For example, expected file permissions on the system or Internet Protocol (IP) restrictions on a web server.
- Additional software or Application Program Interfaces (APIs) available on the system. For example, the presence of a specific cryptographic library.
- Environmental conditions. For example, the presence of a firewall between the application and the network and any ports and protocols that the firewall is expected to block, as well as those specifically required to be open for the application.
- Presence of infrastructure components, such as authentication servers and Virtual Private Network (VPN) concentrators.

All security assumptions identified will be documented as the information may need to be provided to users and administrators to help ensure the security of the application.

3.1.3.4 Identify Objects and Interactions

The next threat modeling step is to identify all of the objects in the application. Objects will fall into the following categories:

- Processes – A logical representation: The performance of discrete tasks or activities. Examples include the authentication component of a daemon or a rendering component in a browser.
- External Entities: Something acting upon the application and outside the control of the application. Examples include system users and external processes.
- Data Stores: Persistent data storage such as files and databases.
- Data Flows: Means by which data moves throughout the system.
- Boundaries: Identifies where information or process control passes between areas of different trust levels, systems, physical locations, or address spaces.

It is important to identify all objects involved in the application, as the rest of the threat modeling process relies on the proper identification of these objects. It is recommended to construct Data Flow Diagrams (DFDs) to represent visual models of the system.

3.1.3.5 Identify Entry Points

Identify the entry points of the system being modeled. These are locations where the system provides or gathers data on behalf of external entities. Sample entry points include:

- Transmission Control Protocol (TCP) ports
- User Datagram Protocol (UDP) ports
- Remote Procedure Call (RPC) end-points
- Web server pages
- Database listeners
- Mail server ports

To identify potential threats, enumerate each entry point discovered earlier and determine what kind of access is offered by the entry point. Access types are defined across two axes. The first is network accessibility:

- Local machine access
- Local subnet access or access limited set of addresses
- Remote access

The second axis measures authentication and authorization requirements for the entry point:

- Anonymous access (no authentication and no authorization)
- Access limited to non-administrative accounts
- Access limited to administrative accounts

The program management team and development team should rank all the entry points into the system from most accessible to least accessible. The most accessible entry points pose the highest degree of risk to the application.

After identifying access points, threats to these entry points and to the underlying application must be ascertained. Each element in the application is subject to attack (for example, data flows are subject to specific threats and processes are subject to others).

Threats can be categorized in order to help identify potential mitigations. There are a variety of categorizations used when enumerating attacks. This document explains the Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege (STRIDE). Other examples of threat categorization include Confidentiality, Integrity and Availability (CIA). STRIDE focuses on the method of categorization, and CIA focuses on security qualities. Information regarding different categorization methodologies (listed as follows) may be found in books or magazines:

- Spoofing: Allows an attacker to pose as a different entity.
- Tampering: The malicious modification of data or code.
- Repudiation: An entity denying having performed an action.
- Information Disclosure: Data or code being revealed to an inappropriate entity.
- Denial of Service (DoS): Prevents a valid entity from accessing an application or system.
- Elevation of Privilege: Allows an entity to execute code or access data at a privilege level above the entity.

Processes are subject to all STRIDE threats. Data stores and data flows are subject to tampering, information disclosure, DoS, and potential repudiation if the data is logging or audit data. Finally, external entities are subject to spoofing and repudiation threats.

Every threat identified will fall into one or more of the above listed categories.

For example, Microsoft provides a threat modeling tool which utilizes the STRIDE methodology of threat modeling. This modeling tool can be found at: http://www.microsoft.com/downloads/details.aspx?FamilyID=62830f95-0e61-4f87-88a6-e7c663444ac1&displaylang=en.

3.1.3.6 Determine Threat Risk

After identifying potential threats, they should be evaluated and ranked to ensure the most critical threats may be addressed first. There are many different ranking formulas available, and the methodology presented in this document is not the only viable ranking formula.

One of the simplest formulas is Risk (*R*) = Damage Potential (*D*) x Chance of Attack (*C*). While this formula may work, it is often difficult to assess the proper value to assign to *C,* since this will change as new attacks and technologies become available.

A critical topic to take into consideration is end-point accessibility. The more accessible an end-point may be, the higher the likelihood of an attack at the end-point.

Every potential threat identified in the previous step should be ranked using the same criteria in order to address the highest risk threats first.

Threat tree patterns are useful to determine the security-related pre-conditions that could lead to security threats.

It is often helpful to construct threat trees to help in ranking potential threats. A threat tree is a specialized form of a decision tree breaking a high-level threat into the individual components or steps necessary to exploit the vulnerability. Using the leaf nodes of the tree, it is possible to develop a better value for *C* and also potentially prune branches of the tree. Threat trees are also helpful in selecting mitigations for vulnerability. The creation of threat trees is beyond the scope of this document; however, many excellent resources on this topic exist on the Internet or in books.

Figure 3-1 presents a sample threat tree.

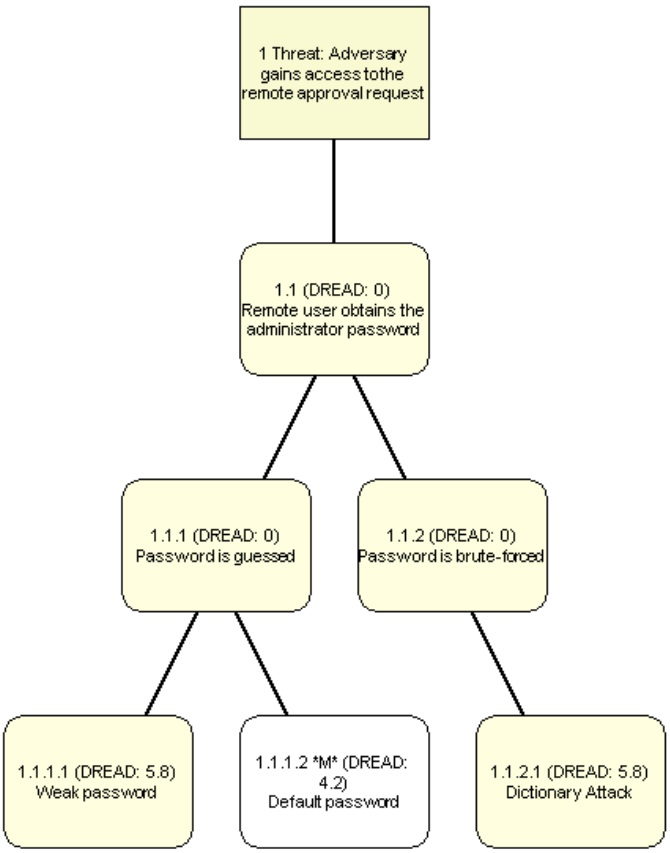

Figure 3-1. Sample Threat Tree

3.1.3.7 Identify Potential Mitigations

For every threat identified, there are several potential solution categories from which to choose:

- Warn the user – Leave the threat unmitigated, but warn the user. Warning the user does not remove the threat, and too many warnings lead to user's becoming "fatigued", so use warning judiciously.
- Disable the feature – Disable the feature, making it an optional application function. This is not a recommended course of action as the threat will still exist if the feature is enabled. Make sure users are aware of the security implications of the feature if it is enabled.
- Remove the feature – If it is impossible to implement a feature securely, it may be necessary to remove the feature rather than putting users at risk.
- Technological solution – There are often technical controls utilized to mitigate the threat.

A solution should be chosen for each threat identified and a document detailing all of the information generated should be created. This completed document is the threat model for the application.

3.1.4 Coding Standards

The development team will follow the coding standards either created or provided by the Program Manager. The development team may be involved in tailoring coding standards of an organization to the specific project.

(APP2060.3: CAT II) The Designer will follow the established coding standards established for the project.

3.1.4.1 Unsafe Functions

The unsafe functions documented in the coding standards will be avoided in the development phase of the application. Automated tools may be used to search and document cases of unsafe use.

(APP2060.4: CAT II) The Designer will not use unsafe functions documented in the project coding standards.

3.2 Using Static Analysis Tools

It is highly recommended that all software developed for DoD use be developed with the aid of static analysis tools. Such tools can find many vulnerabilities faster than humans can; although, they often suffer from issuing numerous false positives.

3.3 Third-Party Products

If the application relies on third-party software products or libraries selected by the development team, it is both the Program Manager's and the development team's responsibilities to ensure the components comply with all requirements in this document.

3.3.1 NIAP-Approved Products

In addition to meeting all the requirements specified in this document, products being used to provide IA or IA-enabled functionality will be evaluated in accordance with the NIAP approval process. For information on NIAP approval process or NIAP-approved products, refer to the following web site: http://www.niap-ccevs.org/cc-scheme/vpl/.

(APP2070.2: CAT III) The Designer will ensure any IA or IA-enabled products used by the application are NIAP-approved or in the NIAP approval process.

3.3.2 Categories of Third Party Products

Software products and libraries with limited or no warranty will not be used in DoD information systems unless they are necessary for mission accomplishment and there are no alternative IT solutions available. If these products are required, they must be assessed for IA impacts and be approved for use by the AO.

The product assessment addresses the risk of deploying software products with limited or no warranty because of difficulties patching or fixing software without having access to the original source or a vendor actively maintaining the software product.

(APP2090.2: CAT II) The Designer will document for AO approval all , public domain, shareware, freeware, and other software products/libraries with both (1) no source code to review, repair, and extend, and (2) limited or no warranty, but are required for mission accomplishment.

3.4 Application Registration

Depending of the branch of service, the application being developed may be required to be registered in several locations. The individual registration requirements for each of the military services are not addressed in this document.

3.4.1 Ports and Protocols

While an application may be written to utilize any ports and/or protocols securely, the DoD has established guidelines regarding the use of Ports, Protocols, and Services (PPS) traversing or connecting to the GiG. The application in a DoD environment will comply with the guidance provided in the DoDI 8551.1 and the companion PPS CAL as well as all associated mitigations, as defined in the appropriate vulnerability assessments at the following web site: http://iase.disa.mil/ppsm/
All IPs, data services, and associated ports (whether used internal or external to the enclave) used throughout planned, newly developed, acquired, and existing DoD networks, which include DoD information systems, applications, and boundary protection devices, must be limited to essential capabilities, assessed for vulnerabilities, documented in a vulnerability assessment (VA) report, and assigned an assurance category and documented in the Category Assurance List (CAL).

(APP2100.1: CAT II) The Designer will ensure the use of new IPs, data services, and associated ports used by the application are submitted to the appropriate approving authority for that organization, which in turn are submitted through the DoD Ports, Protocols, and Services Management (DoD PPSM)."

All DoD applications operating on DoD networks are required to be registered (aka declared) in the DoD Ports and Protocols database.

(APP2110.2: CAT II) The Designer will ensure the application is registered with the DoD Ports and Protocols database for operation on a DoD network and/or production site."

3.5 Best Practices

The establishment of various best practices is an important aspect of a secure development environment. These best practices will provide consistent quality code when developing custom applications.

Source code never executed by the application will be removed from the application. Software components and libraries from approved third-party products may contain code never invoked during operation.

Many compilers offer compile time and runtime protections against stack and buffer overflows. These compile time options should be enabled wherever possible. Visual C++ allows the use of "/GS (stack base overflows)," and "/SAFESEH (secure exception handling)/NXCOMPAT (hardware-based" options for software-enforced data execution prevention (DEP) also referred to as NX) and /DYNAMICBASE (image and stack randomization).

(APP3050: CAT II) The Designer will ensure the application does not contain source code that is never invoked during operation, except for software components and libraries from approved third-party products, which may include un-invoked code.

(APP3060: CAT II) The Designer will ensure the application does not store configuration and control files in the same directory as user data.

User interfaces components of the application should be physically or logically separated from data storage services (e.g., database servers). Separation may be accomplished through the use of different computers, different central processing units (CPUs), different instances of the operating system, different network addresses, and combinations of these methods.

(APP3070: CAT II) The Designer will ensure the user interface services are physically or logically separated from data storage and management services.

(APP3080: CAT II) The Designer will ensure the application does not contain invalid URL or path references.

(APP3100: CAT II) The Designer will ensure the application removes temporary storage of files and cookies when the application is terminated.

3.5.1 Secure Defaults

The practice of secure defaults helps to ensure the application is deployed in a secure state. This practice implies unneeded or potentially unsafe functionality is disabled by default, and the user must explicitly enable the functionality when required.

As part of secure defaults, the installation process should also assist to ensure any system configuration requirements and system level protections identified earlier are documented for the user or configured during the installation process. If default accounts are created as part of the application installation, the installation process will include a method of changing the passwords associated with these accounts.

(APP3110: CAT II) The Designer will ensure the application installs with unnecessary functionality disabled by default.

3.5.2 Error Handling

Improper error handling in an application can lead to an application failure or possibly result in the application entering an insecure state. Improper error handling is usually not directly attackable, unless a method of repeatedly causing the application to fail is discovered. If such a method is discovered, it is possible for an attacker to cause a DoS or to potentially exploit the insecure state of the application. The more likely effect of improper error handling is a self-inflicted DoS or random cases of the application entering an insecure state.

The following items may indicate potential error handling issues:

- Failure to check return codes or handle exceptions.
- Improper checking of exceptions or return codes.
- Handling all return codes or exceptions in the same manner.
- Error information that divulges potentially sensitive data to an un-trusted user.

The primary way to detect error handling vulnerabilities is to perform code reviews. If a manual code review cannot be performed, static code analysis tools should be employed in conjunction with tests to help force the error conditions by specifying invalid input (such as fuzzed data and malformed filenames) and by using different accounts to run the application. These tests may give indications of vulnerability, but they are not comprehensive.

In order to minimize error handling errors, ensure proper return code and exception handling is implemented throughout the application.

(APP3120: CAT II) The Designer will ensure the application is not subject to error handling vulnerabilities.

3.5.3 Secure Failure

In order to minimize error handling and ensure proper return code, exception handling is implemented throughout the application. The secure failure principle should be implemented.

This principle helps ensure predictable and secure application behavior in case of application failures. Applications often perform checks on the validity of data, user permissions, and resource existence before performing a function. Secure failure is defined if a check fails for any reason, the behavior of the application remains in a secure state. The following pseudo-code provides a simple example of a failure which is not secure:

```
If CheckAccessDenied()
     Display Error Message ()
     DenyAccess()
Else
     Perform Privileged Action ()
Endif
```

In the example above, if the *Check Access Denied* function encounters an error while running and does not return "TRUE" (for example, the function returns an "Out of memory" error), the privileged action is performed. While this example is very simple, complex behaviors are often modeled in an application, and it is important the most secure code branch is executed in the event of an application error. The application code should provide exception handling; thereby, leaving the application in a secure state.

The principle of secure failure design is intended to account for all possible exceptions leaving the application in a vulnerable state. Coding constructs vary from programming language to

programming language. The example above only illustrates the basic concept of secure failure design.

(APP3130: CAT I) The Designer will ensure the application follows the secure failure design principle.

(APP3140: CAT II) The Designer will ensure application initialization, shutdown, and aborts are designed to keep the application in a secure state.

3.6 Cryptography

Through this section, the term "protected with appropriate cryptography" will mean:

- Use of symmetric ciphers to protect data from disclosure.
- Use of Message Authentication Codes, hashes, or digital signatures to protect data from tampering.
- Use of digital signatures to provide data integrity.

3.6.1 FIPS 140-2

If cryptographic technologies are not implemented correctly, it may be possible for an attacker to attack protected data. The Designer will use Federal Information Processing Standard (FIPS)-certified algorithms. FIPS 140-2 Security Requirements for Cryptographic Modules can be found at the following web site: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

Annex A: Approved Security Functions for FIPS PUB 140-2, Security Requirements for Cryptographic Modules can be obtained at the following web site:
http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexa.pdf

(APP3150.1: CAT II) The Designer will ensure the application uses FIPS 140-2 validated cryptographic modules if the application implements encryption, key exchange, digital signature, and hash functionality.

For classified data, cryptographic modules will be NSA-approved cryptographic modules.

3.6.2 NSA-Approved Cryptography

NSA has developed Type 1 algorithms for protecting classified information. The Committee on National Security Systems (CNSS) National Information Assurance Glossary (CNSS Instruction No. 4009) defines Type 1 products as:

"Cryptographic equipment, assembly or component classified or certified by NSA for encrypting and decrypting classified and sensitive national security information when appropriately keyed. Developed using established NSA business processes and containing NSA approved algorithms are used to protect systems requiring the most stringent protection mechanisms."

NSA-approved cryptography is required to be used for classified information system processing.

3.6.3 Random Number Generation

Applications that use weak random numbers for security purposes are vulnerable to attack.

Use a FIPS 140-2 approved random number generator when the random number is to be used for cryptographic or security purposes. The list of approved random number generators can be obtained from the National Institute of Standards and Technology (NIST) web site: http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexc.pdf.

(APP3150.2: CAT II) The Designer will ensure the application uses a FIPS 140-2 validated random number generator to support cryptographic functions.

Random number generators not used in support of cryptographic functions are not required to be FIPS 140-2 validated.

NIST FIPS-validated cryptography should be used as a baseline for cryptographic random number generators. NSA-approved cryptographic random number generators are required for classified systems.

Note: Encryption requirements for data at rest are covered by two categories:
1. Unclassified systems require FIPS 140-2 encryption modules.
2. Classified systems require NSA/Computer Security (COMSEC)-approved Type-1 encryption.

3.6.4 Key Exchange

A common key exchange vulnerability is when the ends of a communication channel do not perform mutual authentication before exchanging a key, keys, or keying material. Without mutual authentication, it may be possible to perform a Man-In-The-Middle attack and intercept/modify the key communications or subsequent communications.

The following items may indicate key exchange vulnerabilities in an application:

- Use of in-house developed key exchange or authentication protocols.
- Key exchange is not part of the authentication protocol.
- Key exchange does not include authentication of both parties.

Use an established authentication protocol to minimize key exchange vulnerabilities.

(APP3170: CAT II) The Designer will ensure the application uses encryption to implement key exchange and authenticate end-points prior to establishing a communication channel for key exchange.

Note: Encryption requirements for data at rest are covered by two categories:
1. Unclassified systems require FIPS 140-2 encryption modules.
2. Classified systems require NSA/COMSEC-approved Type-1 encryption.

(APP3180: CAT II) The Designer will ensure private keys are accessible only to administrative users.

3.7 Data

Improper data handling, storage, and transmission can lead to information disclosure or malicious modification. Depending on what information is disclosed and/or modified, the consequences may be insignificant or could result in a system compromise or sensitive data leakage.

3.7.1 Database Management System

Many applications are designed with a third-party Database Management System (DBMS) for data storage and retrieval. Additional requirements are placed on applications using DBMS COTS packages. The Database STIG imposes additional requirements on DBMS and can be found at the following web site: https://www.us.army.mil/suite/page/397960.

(APP3190: CAT II) The Designer will ensure the application does not connect to a database using administrative credentials or other privileged database accounts.

(APP3200: CAT III) The Designer will ensure transaction-based applications implement transaction roll-back and transaction journaling.

3.7.2 Data Storage

The information owner may elect to require certain information be treated as sensitive data. If required, this data must be protected using appropriate cryptography.

(APP3210.1: CAT II) The Designer will ensure NIST-certified cryptography is used to protect stored sensitive information if required by the information owner.

(APP3210.2: CAT II) The Designer will ensure NIST-certified cryptography is used to store classified non-Sources and Methods Intelligence (SAMI) information if required by the information owner.

If a classified enclave contains SAMI data, this data must protected using appropriate cryptography.

(APP3210.3: CAT II) The Designer will ensure a classified enclave containing SAMI data is encrypted with NSA-approved cryptography.

Note: Encryption requirements for data at rest are covered by two categories:

1. Unclassified systems require FIPS 140-2 encryption modules.
2. Classified systems require NSA/COMSEC-approved Type-1 encryption.

3.7.3 In-Memory Data Handling

In order to minimize in-memory data handling disclosure vulnerabilities in the application, implement the following procedures:

- Encrypt sensitive data held in physical or virtual memory when not being used.
- Clear all memory blocks used to process sensitive data prior to releasing the memory.
- Revoke access authorizations to data prior to initial assignment, allocation, or reallocation to an unused state so information produced by a previous user is not available to a subsequent user that obtains access to an object that has been released back to the system.
- Ensure that memory clearing code is not removed by the compiler when compiler optimization is selected.
- Use memory clearing in supporting development environments (e.g., SecureSting in the .Net Framework Class Library).

(APP3220.1: CAT II) The Designer will ensure sensitive data held in memory is cryptographically protected when not in use if required by the information owner.

(APP3220.2: CAT II) The Designer will ensure classified data held in memory is cryptographically protected when not in use.

(APP3230.1: CAT II) The Designer will ensure the application properly clears or overwrites all memory blocks used to process sensitive data if required by the information owner.

(APP3230.2: CAT II) The Designer will ensure the application properly clears or overwrites all memory blocks used to classified data.

(APP3240: CAT II) The Designer will ensure all access authorizations to data are revoked prior to initial assignment, allocation or reallocation to an unused state.

3.7.4 Data Transmission

In order to minimize data transmission disclosure or manipulation vulnerabilities in the application, implement the following procedures:

- Protect all sensitive data and unclassified data transmitted on a commercial (non-DoD) or wireless network using appropriate cryptography.
- Protect all classified data when network is cleared for lower classification.
- Encrypt all need-to-know data separated for need-to-know reasons.
- Encrypt all SAMI data in transit through a network at the same classification level.

Classified data transmitted through a network, which is cleared to a lower level than the data being transmitted, is separately encrypted using NSA-approved cryptography.

Unclassified, sensitive data transmitted through a commercial or wireless network is encrypted using NIST-certified cryptography.

(APP3250.1: CAT I) The Designer will ensure unclassified, sensitive data transmitted through a commercial or wireless network is protected using NIST-certified cryptography.

(APP3250.2: CAT I) The Designer will ensure classified data, transmitted through a network that is cleared to a lower level than the data being transmitted, is separately protected using NSA-approved cryptography.

Data in transit through a network at the same classification level separated for need-to-know reasons will be encrypted minimally with NIST-certified cryptography.

SAMI data in transit through a network at the same classification level will be encrypted using NSA-approved cryptography.

(APP3250.3: CAT II) The Designer will ensure information in transit through a network at the same classification level, but which must be separated for need-to-know reasons, is protected minimally with NIST-certified cryptography.

(APP3250.4: CAT II) The Designer will ensure SAMI information in transit through a network at the same classification level is protected with NSA-approved cryptography.

Note: Encryption requirements for data in transit are covered by two categories:
1. Unclassified systems require FIPS 140-2 encryption modules.
2. Classified systems require NSA/COMSEC-approved Type-1 encryption.

3.7.5 Data Integrity

Data is subject to manipulation and other integrity related attacks whenever that data is transferred across a network. To protect data integrity during transmission, the application must implement mechanisms to ensure the integrity of all transmitted information. All transmitted information means that the protection of transmitted information is not restricted to just the data itself. Identifying and implementing the application data protection mechanisms are the responsibility of the application designer and are based on evaluated data protection requirements. Protection mechanisms must be extended to include data labels, security parameters or metadata when application data protection requirements specify. Modern web application data transfer methods can be complex and are not necessarily just point to point in nature. Service Oriented Architecture (SOA) and RESTFUL web services allow for XML based application data to be transmitted in a manner similar to network traffic wherein the application data is transmitted along multiple hops. In such cases, point to point protection methods like TLS or SSL may not be the best choice for ensuring data integrity and alternative data integrity protection methods like XML Integrity Signature protections where the XML payload itself is signed may be required as part of the application design. Overall application design and architecture must always be taken into account when establishing data integrity protection mechanisms. Custom-developed solutions that provide a file transfer capability should also implement data integrity checks for incoming and outgoing files. Any transmitted information

will require mechanisms to ensure the data integrity (e.g. digital signatures, cryptographic hashing, TLS or SSL encryption).

(APP3260: CAT II) The Designer will ensure the application uses mechanisms assuring the integrity of all transmitted information (including labels and security parameters).

3.7.6 Data Marking

If the application is designed to handle non-publicly releasable data, it will mark all output. Output includes, but is not limited to, video display, printed materials, database exports, and files. Proper labeling of data is the responsibility of the data creator or the application developer. Guidelines for appropriate marking can be found in DoD 5200.1-R, Information Security Regulation, and DoD 5200.1-PH, DoD Guide to Marking Classified Documents.

(APP3270: CAT I) The Designer will ensure the application has the capability to mark sensitive/classified output when required.

3.8 Authentication

Authentication is a security mechanism designed to verify the identity of a user or other service wishing to access the application. Authentication safeguards against unauthorized access and use. Authentication establishes a trust mechanism between the user or other service and the application.

The sample application shown in Figure 3-2 has many potential authentication points. Many factors need to be considered to determine which potential authentication points are required to safeguard the application.

Access Control is a security mechanism designed to enforce the applications access control policies as they pertain to successfully authenticated users. Access control policies specify what paths of execution through the application a particular user is authorized to execute. All applications and access points should be examined to determine what if any access control is required and for the type of authentication. Adding authentication has a benefit of reducing the amount of code that is accessible to un-trusted users.

Sample Application - Potential Authentication Points

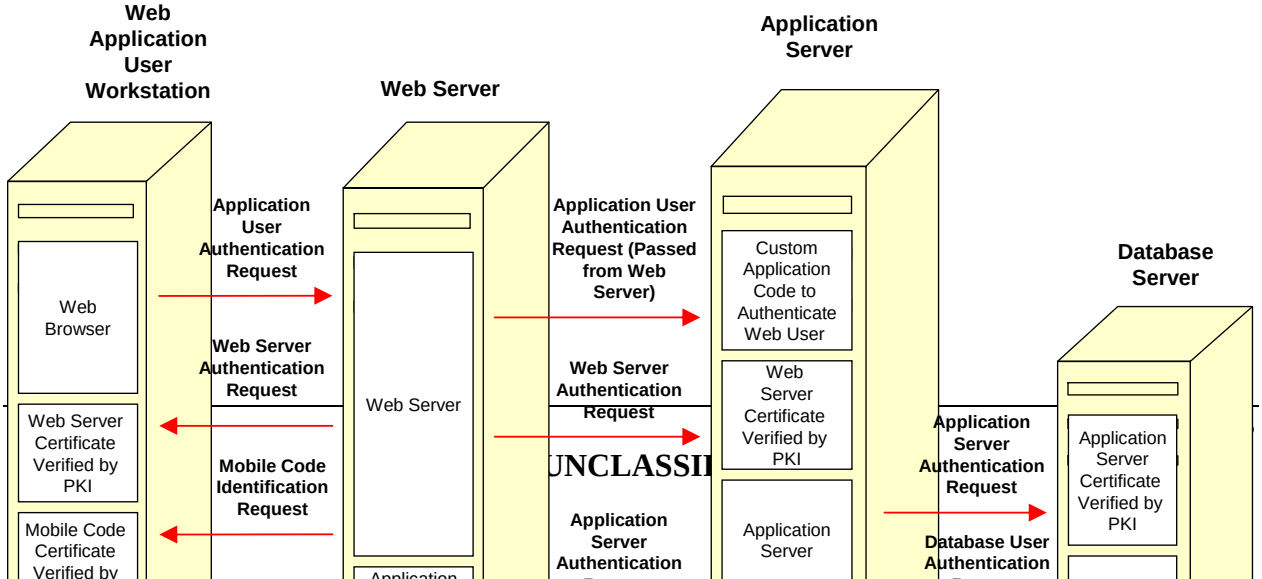

Figure 3-2. Sample Potential Authentication

3.8.1 Types of Authentication

3.8.1.1 Server Authentication

When server authentication is required, it must be performed using DoD Public Key Infrastructure (PKI)-approved credentials as detailed within DoDI 8520.2.

3.8.1.2 User Authentication

"DoD private web servers providing access to DoD sensitive information except those protecting access to personal information by information-privileged individuals shall be PK-Enabled to rely on certificates for client authentication issued by DoD-approved PKI. Information systems that reside behind web servers requiring authorization based on individual identity shall use the identity provided by certificate-based authentication to support access control decisions." If application users are required to be authenticated, then this authentication must be performed using DoD-approved PKI credentials. Certain applications providing read-only public releasable information may not require user authentication.

3.8.1.3 Signed Code Identification

All signed code must be signed with a DoD PKI mobile code signing certificate. Signed code certificates must be validated as indicated in the PKI Certificate Validation section before they are executed on a workstation.

3.8.1.4 Device Authentication

Devices such as Cell Phones, Personal Digital Assistants (PDAs), or Voice over Internet Protocol (VoIP) phones may require device authentication. For VoIP device authentication requirements, refer to the Voice and Video over Internet Protocol (VVoIP) STIG. For Cell Phones and PDA device authentication requirements, refer to the Wireless STIG.

3.8.2 Application Architecture Types

This section addresses authentication requirements based on the application architecture and access control requirements places on the data stored in the application.

3.8.2.1 Standalone Application Authentication

Standalone applications are applications that do not provide, use, or otherwise interact with the network or application services, as shown in Figure 3-3. Standalone applications can, therefore, make use of other methods such as operating system authentication to establish identity to the application and its resources.

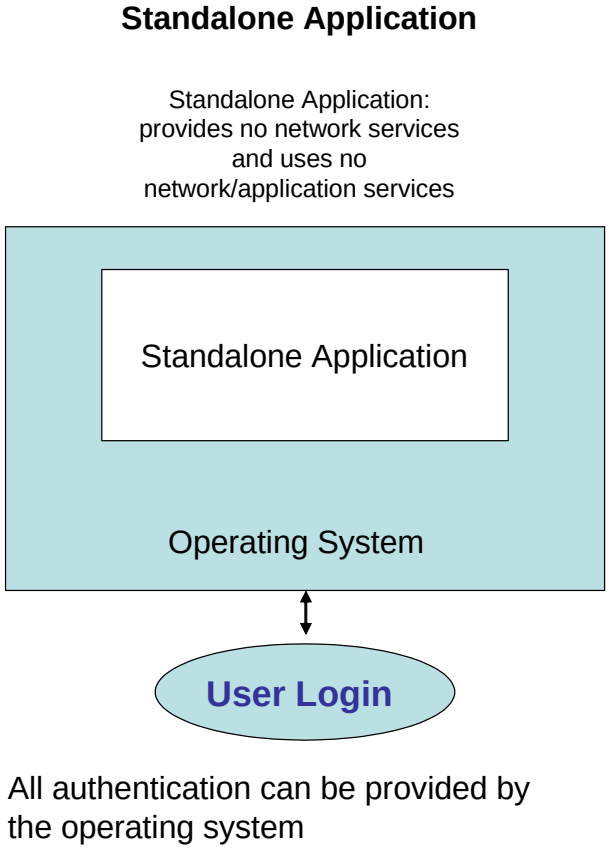

Figure 3-3. Standalone Applications

3.8.2.2 Server Application Authentication

Server applications are applications providing a service to remote users. These applications provide application or network services to their client applications on the network (e.g., File Transfer Protocol (FTP) Servers, Apache Web Server (serving only static data), etc.), as shown in Figure 3-4.

These applications may or may not require authentication based on the data being served. Refer to the Application Component Authentication section for details.

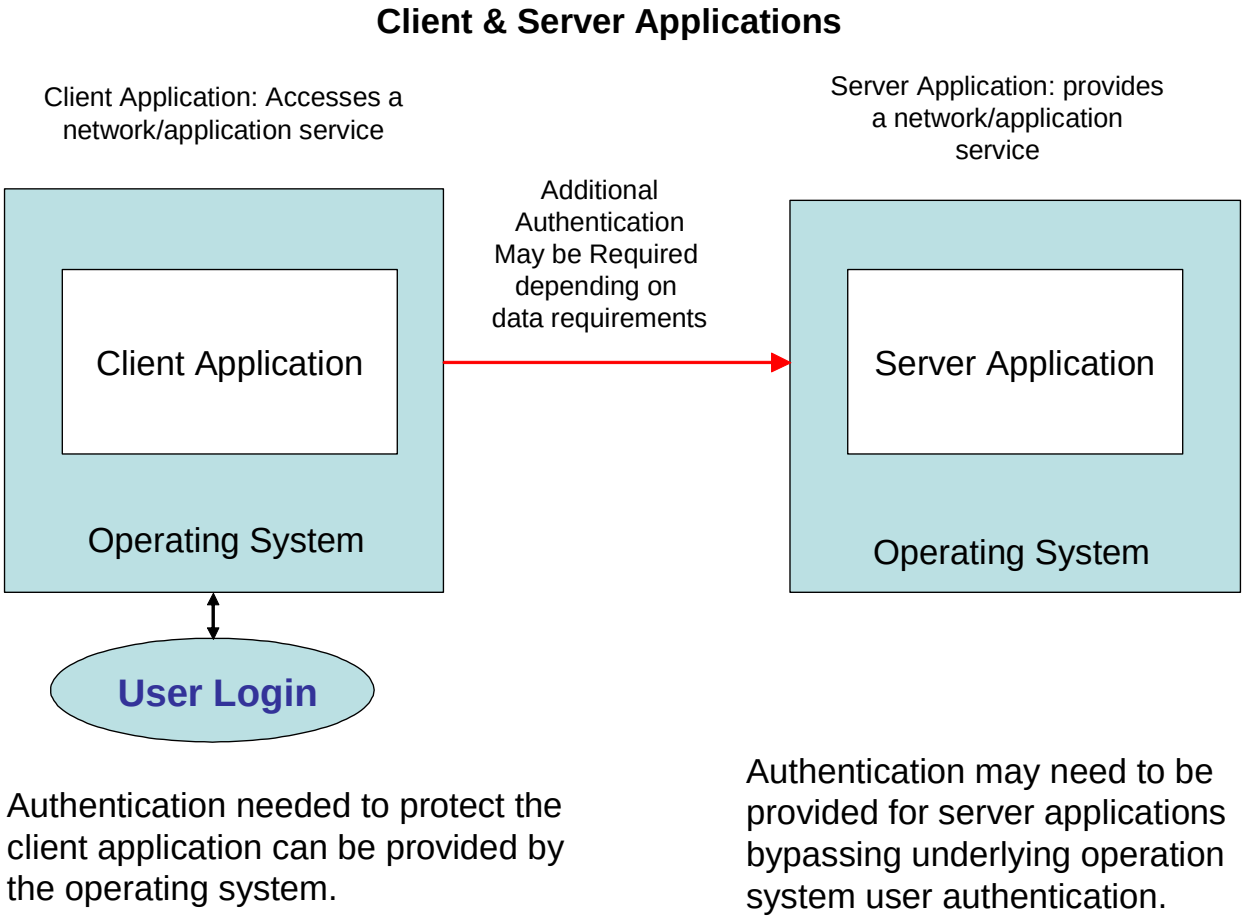

Figure 3-4. Client and Server Applications

3.8.2.3 Client Application Authentication

Client applications provide an interface to server applications. These client applications may be web browsers or traditional client-server applications with each client being physically installed on each user's workstation requiring access. These types of applications do not provide an application or network services (e.g., Microsoft Outlook Client, Internet Explorer, Netscape, etc.).

These applications are protected by the authentication provided by the operating system. These applications may be required to support authentication back to another server. For example, Internet Explorer used to browse a Non-classified Internet Protocol Router Network (NIPRNet) web site without any additional authentication. The operating system login is sufficient to protect this type of application. Microsoft Outlook Client is also protected by the operating system login; however, the Microsoft Outlook Client also connects to the Microsoft Exchange Server, so this application will require need to support authentication from the server application.

3.8.2.4 Combination Client Server Application Authentication

Most applications do not fit solely into server or client applications. Many applications have both server components and client components, as shown in Figure 3-5. Applications can be decomposed into server components and client components and each must be analyzed separately for authentication requirements (e.g., Tomcat Application Server, WebLogic Application Server, etc.).

Client Server Combination Applications

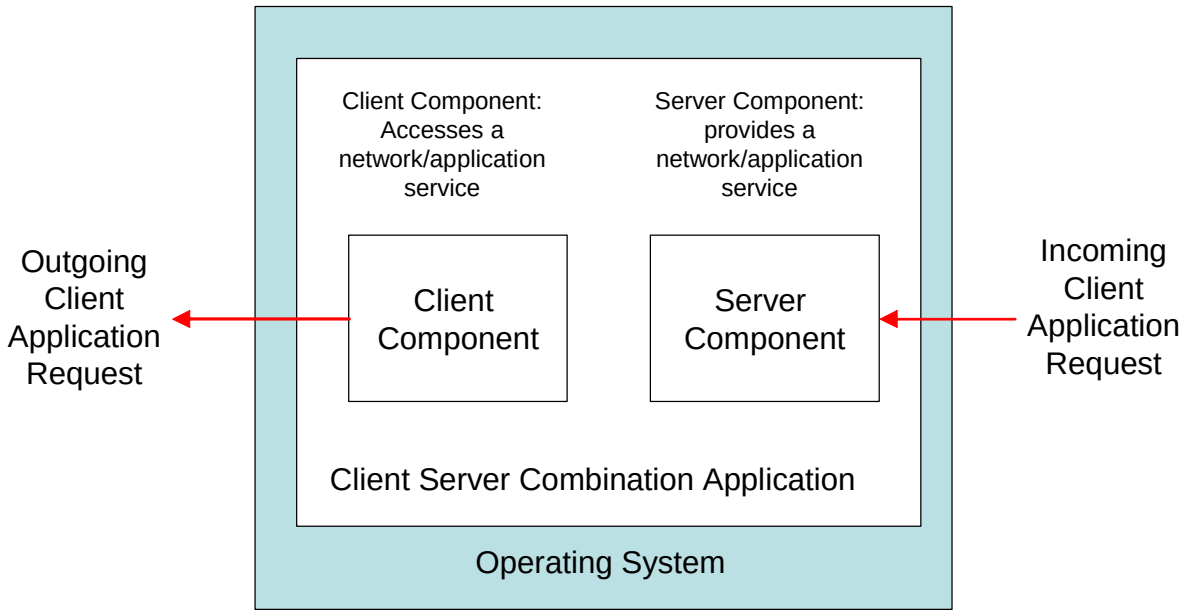

Figure 3-5. Client Server Combination Applications

3.8.2.5 Application Component Authentication

All server application and server components data need to be examined to determine if access control mechanisms are required for data access. If the application data or data owner requires additional access control, then user authentication must be implemented. This information will already be documented in the application's design document.

- User authentication and authorization is not required for the following situations: Reading publicly releasable information

User authentication and authorization is required for the following situations:

- Modifying any class of information
- Reading or modifying any information deemed to be sensitive by the data owner
- Reading or modifying any Classified or Sensitive information
- Any privileged access

3.8.3 PKI Authentication

Both DoDI 8500.2, "Information Assurance Implementation" and DoDI 8520.2, "Public Key Infrastructure (PKI) and Public Key (PK) Enabling" require the use of DoD-approved PKI certificates for identification and authentication as detailed in reference A. This is applicable to:

- *"All DoD unclassified and classified information systems including networks (e.g. Non-secure Internet Protocol Router Network, Secret Internet Protocol Router Network, web servers, and e-mail systems.*"

- *"DoD private web servers providing access to DoD sensitive information except those protecting access to personal information by information-privileged individuals shall be PK-enabled to rely on certificates for client authentication issued by DoD-approved PKIs".*

3.8.3.1 PKI User Authentication

All applications requiring user authentication to access sensitive data must be PK-enabled in compliance with DoDI 8520.2 PKI & PK Enabling and are required to credentials approved under the DoD PKI program. While the Common Access Card (CAC) with a DoD-issued PKI certificate is the primary credential for use within the DoD, other cases allow for the use of alternative tokens with PKI certificates. These groups may include; system/network administrators, privileged individuals, administrative users, volunteers, groups/role based, and reservists involved with the transfer of personal information.

Applications hosted on both NIPRNet and Secret Internet Protocol Router Network (SIPRNet) are required to be PK-enabled and should support and use credentials authorized under the DoD PKI program.

(APP3280.1: CAT II) The Designer will ensure applications requiring user authentication are PK-enabled.

(APP3280.2: CAT II) The Designer will ensure applications requiring user authentication are designed and implemented to support hardware tokens (e.g., CAC for NIPRNet).

(APP3290.1: CAT II) The Designer will ensure PK-enabled applications are designed and implemented to use approved credentials authorized under the DoD PKI program.

3.8.3.2 PKI Server Authentication

Networks and web servers accessed by DoD eligible users and DoD partners shall be enabled to use certificates for authenticating users and to support access control decisions.

For information systems requiring authentication other than network login or web servers, the system owner shall perform a business case analysis to determine if PK-enabling is warranted.

If a web server is required to authenticate with the client application, the authentication must be PK-enabled.

(APP3300: CAT II) The Designer will ensure applications requiring server authentication are PK-enabled.

3.8.3.3 PKI Certificate Validation

When validating credentials approved for use within the DoD, the following requirements apply. Details on how to meet these requirements are provided in the following sub-sections.

- Validate the certificate has not expired.
- Validate the certificate is used for its intended purpose.
- Validate the certificate is from a trusted DoD-authorized Certificate Authority (CA).
- Validate the chain of trust or path for a certificate.
- Validate the certificate has not been revoked using either Certificate Revocation Lists (CRLs) or Online Certificate Status Protocol (OCSP).
    - If caching the CRL, the cache will be updated at least daily.
    - Reject the certificate if its certificate status is not available or current.

(APP3305: CAT I) The Designer will ensure the application using PKI validates certificates for expiration, confirms origin is from a DoD-authorized CA, and verify certificate has not been revoked by CRL or OCSP, and CRL cache (if used) is updated at least daily.

Note: For server to server authentication where an administrator has responsibility for both server certificates stores, certificate status checking may be disabled.

3.8.3.3.1 Certificate Expiration

DoD PKI certificates are typically issued with a maximum lifetime of three years. There are exceptions however. For example, a DoD contractor whose current contract duration is only one year would receive certificates that are only good for the life of his/her current contract (one year in this case). The certificate lifetime is controlled by two validity fields within the certificate, the Valid from and Valid to fields. The Valid from field indicates the date on which the certificate becomes valid (not before). The Valid to field indicates the expiration date of the certificate (not after).

3.8.3.3.2 Certificate Intended Purpose

Certificates should be validated that it is used for its intended purpose with the key usage field in the X.509 Certificate. Below are examples of key usage types.

- Key Encipherment
- Digital Signature

- Certificate Signing

3.8.3.3.3 Trusted DoD Certificate Authorities

The application must be configured to use only trusted DoD-approved CA.

3.8.3.3.4 Certificate Path Validation

All PKIs are based on a trust model providing a clear trust path between the issued certificate and the root CA. The application must elect to trust the original CA from which the end entity certificate was issued. For DoD PKI, applications must be configured to trust the DoD root CAs. By trusting the DoD root CAs, the application will implicitly trust any intermediate CAs and certificates issued by the intermediate CAs.

When a DoD PKI certificate is accepted by the application, it must validate all of the following to successfully recognize:

- End entity certificate.
- Certificate of the intermediate CA that issued the end entity certificate.
- Certificate of the DoD root CA which issued the intermediate CA's certificate.

3.8.3.3.5 Revocation Checking

3.8.3.3.5.1 OCSP and CRL

OCSP for client applications is the preferred method of revocation checking for servers. However, OCSP creates a significant number of certificate validations. Although, an OCSP responses and requests are small, an application would need to check the revocation information for all certificates OCSP creates a large volume of traffic and can impact network and system performance. CRL servers pull data once a day and can be located close to the OCSP requestor reducing network traffic. The DoD advises server administrators store revocation data close to reduce network traffic. Support for revocation checking differs among application types. Some applications do not natively support OCSP or robust validation checking. Designers are advised to review their server documentation, as well as guidance published on the DoD Public Key Encryption (PKE) DKO site (https://www.us.army.mil/suite/page/474113), to determine the best configuration for their particular system.

3.8.3.3.5.2 Local CRL Caching

Local CRL Caching is recommended to lower local network traffic. In order to lessen the local bandwidth consumption due to CRL downloads, it is recommended that Combatant Commanders/Services/Agencies' (CC/S/A's) cache their CRL files. The DoD PKI Program Management Office (PMO) provides several tools to download CRLs on a scheduled basis during off-peak hours. CRLAutoCache allows administrators to schedule transfers of CRLs from multiple sources to multiple destinations. The DoD PKI PMO has developed scripts for CRL downloads for Windows and Linux platforms. Many DoD networks have web proxies deployed that can cache DoD PKI CRL contents. However, most web proxies are not configured to support caching the dynamic URLs to publish CRLs. The DoD PKI PMO has created documentation for enabling caching of DoD CRLs for the following proxy products:

- Bluecoat
- ISA
- Squid

3.8.4 Password Authentication

PK-enabled applications using public key certificates for authentication are not required to implement password authentication, enforce password complexity and maintenance requirements. Applications may be designed to use both PKI-based and password-based authentication.

Note: DoDI 8520.2 states web servers require a minimum of password-based authentication depending on the conditions outlined:

"The DoD Components that conduct web server-based transactions with information privileged individuals, volunteers, or Reservists involving the transfer of personal information, shall use encryption to ensure confidentiality of these transactions, and shall require, at a minimum, that the information-privileged individual present a user-id and password."

DoDI 8520.2 outlines when password-based authentication is allowed; however, password-based authentication is highly discouraged.

Password authentication may be used as an underlying certificate to user account mapping mechanism. If password authentication is used, password complexity and maintenance requirements must still be enforced.

3.8.4.1 Password Masking

When users enter passwords for authentication to the application, the passwords should not be visibly displayed. Characters such as an asterisk or other characters can be displayed instead of the actual password.

(APP3310: CAT I) The Designer will ensure the application does not display account passwords as clear text.

3.8.4.2 Password Complexity and Maintenance

All account passwords, both administrative and non-administrative, must comply with the following:

- Passwords must be at least 15 characters long.
- Passwords must contain a mix of upper case letters, lower case letters, numbers, and special characters.
- When a password is changed, users must not be able to use personal information such as names, telephone numbers, account names, or dictionary words.
- Passwords must expire after 60 days.
- Users must not be able to reuse any of their previous 10 passwords.
- Users must not be able to change passwords more than once a day, except in the case of an administrator or a privileged user. Privileged users may be required to reset a user's forgotten passwords and the ability to change passwords more than once per day.

- When a password is changed, the new password must differ from the previous password by at least four characters.

(APP3320.1: CAT II) The Designer will ensure the application has the capability to require account passwords having a minimum of 15 alphanumeric characters in length.

(APP3320.2: CAT II) The Designer will ensure the application has the capability to require account passwords contain a mix of upper case letters, lower case letters, numbers, and special characters.

(APP3320.3: CAT II) The Designer will ensure the application has the capability to require account passwords be changed every 60 days or more frequently.

(APP3320.4: CAT II) The Designer will ensure passwords do not contain personal information such as names, telephone numbers, account names, birthdates, or dictionary words.

(APP3320.5: CAT II) The Designer will ensure the application has the capability to limit reuse of account passwords within the last 10 password changes.

(APP3320.6: CAT II) The Designer will ensure the application has the capability to limit user changes to their account passwords once every 24 hours with the exception of privileged or administrative users.

(APP3320.7: CAT II) The Designer will ensure the application has the capability to require new account passwords differ from the previous password by at least four characters when a password is changed.

3.8.4.3 Password Transmission

In general, password use is highly discouraged in favor of PKI authentication. However, if applications transmit account passwords, they must be transmitted in an encrypted format. Applications that transfer passwords should be thoroughly analyzed for security weaknesses.

(APP3330: CAT I) The Designer will ensure the application transmits account passwords in an approved encrypted format.

Note: Encryption requirements for data in transit are covered by two categories:

1. Unclassified systems require FIPS 140-2 encryption modules.
2. 2. Classified systems require the use of Federal Information Processing Standards (FIPS) encryption algorithms AND ENCRYPTION KEY LENGTHS that are NSA approved for the classification level being utilized. Type-1 Hardware encryption is another approved method of protecting classified data.

3.8.4.4 Password Storage

It is very difficult to protect stored secrets including passwords, so storing passwords is discouraged. In many instances, verifying the user knows a password is performed using a password verifier. In its simplest form, a password verifier is a hash of a password. A more

secure version of verifying a user knowing a password is to store the result of an iterating hash function and a large random salt value as follows:

```
H0 = H(pwd, H(salt))

Hn = H(Hn-1,H(salt))
```

Where *n* is a very large number, at least 100,000. *Hn* is stored, along with the salt. When the application wishes to verify that user knows a password, it simply repeats the process and compares *Hn* with the stored *Hn*.

If an application must store a password, the password must be encrypted using an appropriate encryption function, key and salt, and the encryption key stored in a secure location accessible only to trusted users and applications.

(APP3340: CAT I) The Designer will ensure the application stores account passwords in an approved encrypted format.

Note: Encryption requirements for data at rest are covered by two categories:
1. Unclassified systems require FIPS 140-2 encryption modules.
2. Classified systems require NSA/COMSEC-approved Type-1 encryption.

3.8.5 Authentication Credentials Protection

Embedding authentication data (such as passwords, usernames, and private keys in an application executable or data stores) could lead to unauthorized access to other application or system resources.

The following items may indicate potential embedded authentication data issues:

- Usernames, passwords, and private keys are stored in the executable.
- Usernames, passwords, and private keys are stored in a user-accessible data store.

The primary way of identifying embedded authentication data is through reviews of the application source code, data stores, binary files, and installation routines.

In order to eliminate embedded authentication data in the application, implement the following procedures:

- Use operating system protection mechanisms for securely storing sensitive data.
- Ensure strong access control permissions on data files containing sensitive information.
- Use appropriate cryptographic defenses when storing authentication data.

(APP3350: CAT I) The Designer will ensure the application does not contain embedded authentication data.

(APP3360: CAT II) The Designer will ensure the application protects access to authentication data by restricting access to authorized users and services.

3.8.6 User Accounts

The Designer will ensure the application installation procedures install the application with critical application accounts enabled and any unnecessary accounts disabled or deleted by default.

(APP3370: CAT II) The Designer will ensure the application installs with unnecessary accounts disabled or deleted by default.

The application will provide a mechanism to ensure duplicate user account names are not created. Using operating systems functions to manage user accounts may prohibit the creation of duplicate accounts.

(APP3380: CAT II) The Designer will ensure the application prevents the creation of duplicate accounts.

The application should allow only three consecutive incorrect logon attempts in a one hour period prior to locking the account.

(APP3390: CAT I) The Designer will ensure users' accounts are locked after three consecutive unsuccessful logon attempts within one hour.

The application should only allow the application administrator to manually unlock application accounts. The application administrator should have a policy to confirm the user identity through secret questions and other mechanisms before unlocking the locked users account.

(APP3400: CAT II) The Designer will ensure locked users' accounts can only be unlocked by the application administrator.

3.8.7 Sessions

The application should provide system definable parameters for the following:

- The total number of user sessions open for the entire application.
- The total number of concurrent sessions that can be opened by a single user.
- The total amount of idle time before the user session is forced to terminate.

These system-defined parameters may vary greatly with the type of application being developed. There are no upper or lower bounds required for these parameters, but they must be configurable on deployment of the application.

Session implementations must comply with the following:

- If the application allows multiple logon sessions, the application should limit the number of user sessions to the system defined parameter.
- The application should limit the total number of sessions across all users to the system-defined parameter.
- A capability to terminate a session and logout must be provided.
- A capability to terminate a session and logout after system-defined session idle time limit has been exceeded must be provided.
- Authentication credentials must be removed on client computers after a session terminates.

(APP3405: CAT I) The Designer will ensure the application supports detection and/or prevention of communication session hijacking.

(APP3410.1: CAT II) The Designer will ensure the application provides a capability to limit the number of logon sessions per user.

(APP3410.2: CAT II) The Designer will ensure the application provides a capability to limit the total number of logon sessions for the application.

(APP3415: CAT II) The Designer will ensure the application provides a capability to automatically terminate a session and logout after a system defined session idle time limit is exceeded.

(APP3420: CAT II) The Designer will ensure the application provides a capability to terminate a session and logout.

(APP3430: CAT I) The Designer will ensure the application removes authentication credentials on client computers after a session terminates.

3.8.8 Logon Banner

The Policy on Use of DoD Information Systems Standard Consent Banner and User Agreement Memorandum requires the use of a standard Notice and Consent Banner. The Logon Banner is one of two standard banners to be used based on the character limitations imposed by the system. The banner is mandatory and deviations are not permitted except as authorized in writing by the Deputy Assistant Secretary of Defense for Information and Identity Assurance.

Use the following banner for desktops, laptops, and other devices accommodating banners of 1300 characters. The banner shall be implemented as a click-through banner at logon (to the extent permitted by the operating system), meaning it prevents further activity on the information system unless and until the user executes a positive action to manifest agreement by clicking on a box indicating "OK.". The text of this banner should be customizable in the event of future updates, so those deploying the application can reconfigure the application with any additional banner updates, such as the following banner sample:

You are accessing a U.S. Government (USG) Information System (IS) that is provided for USG-authorized use only.

By using this IS (which includes any device attached to this IS), you consent to the following conditions:

- The USG routinely intercepts and monitors communications on this IS for purposes including, but not limited to, penetration testing, COMSEC monitoring, network operations and defense, personnel misconduct (PM), law enforcement (LE), and counterintelligence (CI) investigations.
- At any time, the USG may inspect and seize data stored on this IS.
- Communications using, or data stored on, this IS are not private, are subject to routine monitoring, interception, and search, and may be disclosed or used for any USG-authorized purpose.
- This IS includes security measures (e.g., authentication and access controls) to protect USG interests--not for your personal benefit or privacy.
- Notwithstanding the above, using this IS does not constitute consent to PM, LE or CI investigative searching or monitoring of the content of privileged communications, or work product, related to personal representation or services by attorneys, psychotherapists, or clergy, and their assistants. Such communications and work product are private and confidential. See User Agreement for details.

For Blackberries and other PDAs/Personal Electronic Devices (PEDs) with severe character limitations, use the following banner:

I've read & consent to terms in IS user agreem't.

(APP3440: CAT II) The Designer will ensure the application is capable of displaying a customizable click-through banner at logon which prevents further activity on the information system unless and until the user executes a positive action to manifest agreement by clicking on a box indicating "OK".

3.9 Access Control

An access control flaw exists if users or processes can view or modify data to which they should not be permitted. This could result in situations ranging from information disclosure to system compromise and could potentially result in the compromise of other systems on the network.

The following items may indicate inadequate access control:

- Objects are created and protected using custom permissions.
- Configuration files are readable and/or modifiable by users.
- Unprivileged user has rights to Add/View/Modify User Information or Personally Identifiable Information (PII).

In order to minimize access control flaws in an application, implement the following procedures:

- Specify the restrictive permission sets for objects on creation.
- Protect application configuration and executable files with the permission sets allowing only the application administrator to modify these files.

(APP3450.1: CAT II) The Designer will ensure application resources are protected with permission sets which allow only an application administrator to modify application resource configuration files.

3.9.1 Name Resolution

A name resolution vulnerability exists when an application trusts the results of a name resolution service, such as the hosts file, Domain Name System (DNS) or Windows Internet Name Service (WINS), as the sole means of authenticating a server or a client. Since there are many, well-known vulnerabilities in these protocols, most notably a lack of server authentication and protections on the resulting data, it is possible for an attacker to compromise the name resolution service and redirect the application to an alternate host. Depending on the design of the application, redirecting the application can lead to issues ranging from information disclosure to application compromise.

The following items may indicate name resolution vulnerabilities in an application:

- Using name resolution as the only means of identifying a server.
- Using name resolution as the only means of identifying a client.

Testing should be performed by configuring a malicious server and/or client, intentionally causing the name resolution to redirect to the malicious server/client and examining the behavior of the application when it is connected to the wrong system.

In order to minimize name resolution vulnerabilities, implement the following procedures:

- Use a cryptographic method of verifying the identity of a system prior to establishing any trust relationship.
- Use preferred methods with established protocols such as Transport Layer Security (TLS) or Internet Protocol Security (IPSec).

(APP3460: CAT I) The Designer will ensure the application does not rely solely on a resource name to control access to a resource.

3.9.2 Role-Based Access

The application should be divided into functionality based on roles. Users will be assigned to specific roles.

(APP3470.1: CAT II) The Designer will ensure the application is organized by functionality and roles to support the assignment of specific roles to specific application functions.

The user roles will determine which functionality the user can access. The application should provide access control mechanisms to ensure users only have access to data or information for the role (or roles) they have been granted.

(APP3480.1: CAT I) The Designer will ensure access control mechanisms exist to ensure data is accessed and changed only by authorized personnel.

(APP3480.2: CAT II) The Designer will ensure the access procedures enforce the principles of separation of duties and "least privilege".

3.9.3 Excessive Privileges

An application executing with more privileges than are required for it to function is considered to have excessive privileges and is violating the "Principle of Least Privilege." An application with excessive privileges greatly increases the risk to the system in the event the application suffers a security breach. The type of attack performed will vary based on the privileges granted to the application account.

The following items may indicate the presence of excessive privileges in an application:

- Applications executing as Administrator, System, root or any other privileged accounts.
- Applications requesting full control when accessing resources.

The primary methods of detecting excessive privileges are threat modeling and code reviews.

In order to minimize excessive privileges used by an application, implement the following procedures:

- Identify all resources the application needs to access, and ensure permissions or Access Control Lists (ACLs) are in place.
- Identify what types of access each user role needs for a resource, and ensure proper permissions or ACLs are in place.
- Identify any special privileges needed by the application.
- Where possible, run as the Local Service or Network Service accounts instead of SYSTEM (Windows).
- Remove all unneeded privileges from the process token when the process starts.

(APP3500: CAT II) The Designer will ensure the application executes with no more privileges than necessary for proper operation.

3.10 Input Validation

A major cause of software vulnerabilities is a failure to validate an un-trusted input. Any data crossing a trust boundary, as identified in the threat modeling process, will be checked to ensure validity before being used. Input may come from a user, data store, network socket, or other source.

When validating data, the application will check for known-good data and reject any data not meeting these criteria. Checking for known-bad data may still allow an attacker to get bad data through the validation by using alternate data encoding or patterns the developers have not considered. Allowing only known good data through may increase the possibility of valid data being rejected; however, this significantly lessens the possibility of invalid data being passed on to the application.

(APP3510: CAT I) The Designer will ensure the application validates all input.

For web applications, setting the character set on the web page reduces the possibility of receiving unexpected input that uses other character set encodings by the web application.

(APP3530: CAT II) The Designer will ensure the web application assigns the character set on all web pages.

3.10.1 SQL Injection Vulnerabilities

Structured Query Language (SQL) injection vulnerability allows an attacker to modify a database query to access or modify data to which they are not permitted. SQL injection vulnerabilities are exploited through an un-validated user input.

The following items may indicate potential SQL injection vulnerabilities in an application:

- Un-validated user input utilized to build queries or as parameters to stored procedures.
- Use of string concatenation to build queries.
- Use of string replacement to build queries.
- Use of SQL exec function.

The primary method of detecting SQL injection vulnerabilities is a code review. It is critical that all SQL-based applications be thoroughly code-reviewed for SQL injection vulnerabilities.
In order to minimize SQL injection vulnerabilities, implement the following procedures:

- Use static analysis tools that are known to find this class of vulnerability with few false positives.
- Validate all user input, allowing only known good input through.
- Use prepared or parameterized statements.
- Do not use string concatenation or string replacement to build SQL queries.
- Allow access to the database through views not directly to underlying tables in the database.

- Use sp_executesql and quotename inside stored procedures to help ensure input is valid (Microsoft SQL Server).

(APP3540.1: CAT I) The Designer will ensure the application is not vulnerable to SQL injection.

(APP3540.2: CAT II) The Designer will ensure the application uses prepared or parameterized statements.

(APP3540.3: CAT II) The Designer will ensure the application does not use concatenation or replacement to build SQL queries.

(APP3540.4: CAT II) The Designer will ensure the application does not directly access the tables in a database.

3.10.2 Integer Arithmetic Vulnerabilities

Integer arithmetic vulnerabilities result from inconsistent and/or incorrect results from calculations due to the handling of integer data types. These errors can lead to symptoms ranging from crashes and incorrect results to allowing an attacker to exploit buffer overflows. There are three kinds of integer arithmetic issues that could lead to security vulnerabilities:

- Signed vs. unsigned mismatches (e.g., comparing a signed integer to an unsigned integer)
- Truncation (e.g., incorrectly truncating a 32-bit integer to a 16-integer)
- Underflow and overflow (e.g., the sum of two numbers exceeds the largest possible value for the integer size in question)

The following items may indicate the presence of integer overflows in the application:

- Mixing signed and unsigned data types in calculations or comparisons.
- Mixing data types of different sizes in calculations or comparisons.
- Comparisons between variables and literal values.
- Use of un-validated input.
- Calculations not validated before the result is utilized.

The primary method of detecting integer overflows in an application is a code review. It is critical that all dynamic memory allocations and array indexing that uses arithmetic be code reviewed for correctness. The following items list some of the tests to be performed to help uncover integer-related vulnerabilities.

- Input negative values for numeric input
- Input border case values (i.e., 0, 7, 8, 254, 255, 16353, 16354)
- Input extremely large string values (> 64k)
- Input strings whose lengths equal border cases (32k, 32k-1, 64k, 64k-1)

As a general best practice, fuzz testing can uncover many integer-related vulnerabilities. Fuzz testing is the process of sending large and invalid data blocks as input to an application in an attempt to cause an application error.

In order to minimize integer overflows, implement the following procedures:

- Use static analysis tools that are known to find this class of vulnerability with few false positives.
- Use unsigned values whenever possible.
- Use only unsigned integers in memory allocation.
- Use only unsigned array indexing functions.
- Validate user input of numeric value, allowing only known good data to pass.
- Compile with the highest warning level possible.
- (C++) Use size_t types to hold size variables.
- (C++) Use the SafeInt class available from Microsoft.
- (C#) Compile with the */checked* compiler option.
- (GCC) Compile with the –ftrapv option.

(APP3550: CAT I) The Designer will ensure the application is not vulnerable to integer arithmetic issues.

3.10.3 Format String Vulnerabilities

Format string vulnerabilities occur when specially crafted format strings passed to a function allow flow control information to be viewed or modified. In a worst-case scenario, format string vulnerabilities can allow attackers to execute code of their choice on the system, resulting in complete system compromise. The primary avenue of attack for format string vulnerabilities is un-validated input.

An application taking input and passing it to a formatting function may indicate the presence of format string vulnerabilities. The primary method of detecting this vulnerability is a source code review or use of static analysis tools.

The %p (pointer) specifier should be used with caution, and the %n (number of characters written) should be avoided.

To test the application, insert format string specifiers in all areas of string input and observe the output looking for anomalies. The format specifiers will vary based upon the language used; however, in addition to language specific specifiers, the C and C++ language specifiers should also be tested.

In order to minimize format string vulnerabilities, implement the following procedures:

- Use static analysis tools that are known to find this class of vulnerability with few false positives.
- Validate all input before passing it to a function, allowing only good data to pass.
- Format strings used by the application should only be accessible by privileged users.

- (C++) Use stream operators instead of the printf family of functions.
- (GCC) Compile with -Wformat -Wformat-security to detect insecure use of format strings.
- Compile with Microsoft Visual C++ 2005 and later to detect insecure use of format strings at runtime.

(APP3560: CAT I) The Designer will ensure the application does not contain format string vulnerabilities.

3.10.4 Command Injection Vulnerabilities

Command injection attacks are attempts to inject unwanted data into an application for the purpose of executing operating system shell commands. This can allow an attacker to execute code, possibly at a higher privilege level, resulting in system compromise. Command injection vulnerabilities are most often exploited through un-validated input.

Potential command injection vulnerabilities include the use of functions spawning an interpreter or compiler. Table 3-1 lists some common functions vulnerable to command injection.

Table 3-1. Functions Vulnerable to Command Injection

Language	Functions/Characters
C/C++	system(), popen(), execlp(), execvp(), ShellExecute(), ShellExecuteEx(), _wsystem()
Perl	system, exec, `,open, |, eval, /e
Python	exec, eval, os.system, os.popen, execfile, input, compile
Java	Class.forName(), Class.newInstance(), Runtime.exec()

In addition to code reviews, testing should be performed to help identify command injection vulnerabilities. In order to test for command injection vulnerabilities, implement the following procedures:

- Identify all potential interpreters or compilers used to pass data.
- Identify the characters modifying the interpreters' behavior.
- Construct input strings containing these characters causing a visible effect on the system, pass them to the application and observe the behavior. Ensure all input vectors are tested in this manner.

In order to minimize command injection vulnerabilities, validate all input before passing to an interpreter or compiler, allowing only known good input through and using static analysis tools that are known to find this class of vulnerability with few false positives.

(APP3570: CAT I) The Designer will ensure the application does not allow command injection.

3.10.5 Cross Site Scripting (XSS) Vulnerabilities

XSS is a vulnerability where input is accepted by a web site and then sent back through a web page. This input can include code, such as JavaScript, to be executed by the user's browser. Since this code is seen as originating from the web server, it can access data from the servers' domain, such as a cookie, or modify the behavior of the web page by modifying links and other malicious actions. An XSS vulnerability can lead to an attacker gaining personal information or directing a user to a site of the attacker's choice.

If user input is echoed back into the browser, the application may have a potential XSS issue.

Note: This data may go through intermediate processes before being sent back to a browser.

In addition to code reviews, testing should be performed to identify potential XSS vulnerabilities. In order to test for XSS vulnerabilities, implement the following procedures:

- Make a request against the application, setting all input parameters to known-bad values.
- View the Hypertext Markup Language (HTML) response, looking for the known-bad values sent as input. The response containing the values submitted as input may not be returned immediately, but may be sent in response to a future request or possibly to a different entity.

In order to minimize XSS vulnerabilities in the application, implement the following procedures:

- Use static analysis tools that are known to find this class of vulnerability with few false positives.
- Validate all input, allowing only known good input.
- If special characters are required for input, HTML encode user input.
- Set a known character set for all web pages to eliminate unexpected characters.

(APP3580: CAT I) The Designer will ensure the application does not have XSS vulnerabilities.

3.10.6 Cross Site Request Forgery (CSRF) Vulnerabilities

CSRF vulnerability allows an attacker to execute an unwanted command through a vulnerable web application form where the user is already authenticated to the web application. This vulnerability happens because of links sent through social engineering techniques.

(APP3585: CAT II) The Designer will ensure the application does not have CSRF vulnerabilities.

When designing web application forms which are critical to the web application, use a cryptographic nonce (number used once), or other message token, for web form transactions to prevent CSRF attacks.

3.10.7 Buffer Overflow Vulnerabilities

A buffer overflow is a vulnerability where data is written beyond the end of an allocated memory block or below an allocated blow (a buffer underflow). There are several types of buffer overflows, all leading to the same potential issues in the application; the ability to crash the application, or execute arbitrary code on the system. If the vulnerable application is running as an elevated account, such as system or the root account, this could lead to a compromise of the system. Buffer overflows are usually exploited through un-validated input.

The following items may indicate potential buffer overflows within the application:

- Cases where input is not checked before being copied into a buffer.
- Incorrect use of some of the functions listed in Appendix B.
- Incorrect calculations to determine buffer sizes.
- Incorrect calculations to determine array indexes.

The primary methods of detecting buffer overflows are code reviews and fuzz testing. Fuzz testing is the process of sending large and invalid data blocks as input to an application in an attempt to cause an application error.

In order to minimize buffer overflows, implement the following procedures:

- Use static analysis tools that are known to find this class of vulnerability with few false positives.
- Validate all input before use, allowing only known-good input through.
- Replace known-insecure functions with safer functions. (See Appendix B.)
- Recheck all calculations to ensure buffer sizes are calculated correctly.
- Recheck all array access and flow control calculations.
- (C++) Replace character arrays with Standard Template Library (STL) string classes.
- Use compile-time options that add compiler buffer overrun defenses. For example, in Visual C++ 2005 Service Pack 1 and later, use /GS, /SAFESEH, /NXCOMPAT and /DYNAMICBASE; in gcc 4.1.2-25 and later, use the –stack-protector option.

(APP3590.1: CAT I) The Designer will ensure the application does not have buffer overflows.

(APP3590.2: CAT II) The Designer will ensure the application does not use functions known to be vulnerable to buffer overflows.

(APP3590.3: CAT II) The Designer will ensure the application does not use signed values for memory allocation where permitted by the programming language.

3.11 Canonical Representation

Canonical representation issues arise when the name of a resource is used to control resource access. There are multiple methods of representing resource names on a computer system. An application relying solely on a resource name to control access may incorrectly make an access control decision if the name is specified in an unrecognized format.

For example, in Windows, notepad.exe may be represented by the following file and path name combinations:

```
C:\Windows\System32\notepad.exe
%SystemRoot%\System32\notepad.exe
\\?\C:\Windows\System32\notepad.exe
\\host\c$\Windows\system32\notepad.exe
```

An application attempting to restrict access to the file based solely on the file path and name may improperly grant or deny access. The same issue may apply to other named resources on a system, such as a hard- and soft-links, URL, pipe, share, directory, device name, or within data files, if alternate encoding mechanisms are used with the data.

The following items may indicate potential canonical representation issues in an application:

- Access control decisions based upon a resource name.
- Failure to reduce a resource name to its canonical form before use.

In order to minimize canonical representation issues in the application, implement the following procedures:

- Do not rely solely on resource names to control access.
- If using resource names to control access, validate the names to ensure they are in the proper format; reject all names not fitting the known-good criteria.
- Use operating system-based access control mechanisms such as permissions and ACLs.

(APP3600: CAT II) The Designer will ensure the application has no canonical representation vulnerabilities.

3.12 Hidden Fields in Web Pages

A “hidden” field vulnerability results when hidden fields on a web page, values in a cookie, or variables included in the URL can be used for malicious purposes. While these fields are not normally visible or editable by the user of a web browser, they can be viewed and/or modified by looking at the source. Unencrypted hidden fields can be used, provided they cannot be used maliciously.

The following items may indicate the presence of hidden field vulnerabilities in the application:

- HTML being sent to the client including a form field(s) where type=HIDDEN, type ='HIDDEN' or type = "HIDDEN".
- The application uses values stored in a cookie to elevate user access privileges.
- The application uses variables in the URL to elevate user access privileges.

The primary method of detecting hidden fields is to view the HTML sent to clients and search for the HIDDEN specifier. Hidden fields used in an application must be examined to determine if the disclosure or modification of the hidden values will have an adverse effect on the application, application data, or the user.

Implement the following procedures to minimize hidden field vulnerabilities:

- Remove hidden elements from web pages if they are not needed.
- Do not use hidden elements to store values effecting user access privileges.

(APP3610: CAT I) The Designer will ensure the application does not use hidden fields to control user access privileges or as a part of a security mechanism.

3.13 Application Information Disclosure

Information disclosure vulnerabilities are leaks of information from an application which are used by the attacker to perform a malicious attack against the application. This information itself may be the target of an attacker, or the information could provide an attacker with data needed to compromise the application or system in a subsequent attack. Information disclosure vulnerabilities are most often the result of programming errors, insufficient authentication, poor error handling, or inadequate data protection.

The following items may indicate the presence of information disclosure vulnerabilities in the application:

- Information about the operating environment is disclosed to a user.
- Information about a user's operating environment is disclosed to another user.
- Data access requests are not subject to permission checks.
- Error messages revealing information to the users through their wording or timing.
- Application responses revealing internal information to the user through their wording or timing.
- Authentication error messages identifying userids on the system.

The primary method of identifying information disclosure vulnerabilities is to perform a code review. In addition to a code review, the application may be tested by:

- Inducing errors in the program to verify the contents of error messages.
- Attempting variations of invalid input combinations to determine if the response contents or timing reveal information about the system. For example, is the response time or error

message different for an invalid username/invalid password and a valid username/invalid password combination? If so, this would allow an attacker to find valid usernames.
- Attempting to access data the user should not be able to access.

In order to minimize information disclosure vulnerabilities, implement the following procedures:

- Ensure an access control policy is in place to enforce access control of the data.
- Display generic error messages to end users.
- Log specific error information for application administrators.
- Ensure the application responses do not divulge un-needed details.

(APP3620: CAT II) The Designer will ensure the application does not disclose unnecessary information to users.

3.14 Race Conditions

A race condition occurs when two applications, processes, or threads attempt to manipulate the same object. The effects of a race condition can range from an application crash to a compromise of the system. Race conditions occur when developers do not consider what will happen if another entity modifies an object while in use.

The following items may indicate a potential race condition in an application:

- Global objects and resources
- Multiple threads or processes are accessing the same object
- Resources created in common areas
- Overly permissive ACLs

The primary method of identifying race conditions is through a code review. Testing may also be performed to help identify race conditions. Finding race condition through testing can be very difficult as they are often time critical. When testing, run the application as quickly as possible, preferably on a multi-processor system, by launching multiple instances of the application, and sending data to the application as quickly as possible.

Implement the following procedures to minimize race condition vulnerabilities:

- Minimize the use of global variables.
- Use thread-safe and re-entrant version of functions.
- Ensure ACLs are enforced on application resources that restrict the users who can access the protect resources

(APP3630.1: CAT II) The Designer will ensure the application is not vulnerable to race conditions.

Any race conditions specifically contributed by the operating system itself should also be documented. These vulnerabilities should be immediately reported to the operating system vendor for resolution.

(APP3630.2: CAT III) The Designer will ensure the application does not use global variables when local variables could be used.

The Designer will use thread safe functions when threads accessing the same object or data in a multi-threaded application.

(APP3630.3: CAT II) The Designer will ensure a multi-threaded application uses thread safe functions when threads are accessing the same object or data.

(APP3630.4: CAT II) The Designer will ensure global resources are locked before being accessed by the application.

3.15 Auditing

Auditing is the process of monitoring and recording events to provide accountability, track usage, and alert the administrator of potential problems. As part of the design process, auditing requirements and capabilities will be established for the application. A design decision to use third-party products or external auditing capabilities (such as, operating system, database, and web server) must be documented. The auditing capabilities provided by the external component must meet the minimum requirements specified.

This section details the minimum level of auditing required for a DoD application. Applications may require additional auditing to comply with legal requirements or to satisfy other custom requirements going beyond the basic security requirements that are included in this document.

Auditing capabilities, whether provided by an external component or the application itself, will meet the following minimum requirements in this section. Many of the auditing requirements are often handled by a backend database which, in many cases, may be more suitable then other alternatives.

(APP3640: CAT II) The Designer will ensure the application supports the creation of transaction logs for access and changes to the data.

3.15.1 Audit Notifications

The auditing system will provide an alert to administrative personnel when the audit log sizes are approaching their limits. Warnings may be accomplished by using appropriate operating system functions or network monitoring functions. A custom application is not necessarily required to be developed to implement warnings to administrators.

(APP3650: CAT III) The Designer will ensure the application has a capability to notify an administrator when audit logs are nearing capacity as specified in the system documentation.

The Designer will ensure the application has a capability to notify the user on login the following information:

- Date and time of the user's last unsuccessful logon
- IP address of the user's last unsuccessful logon
- Date and time of the user's last successful logon
- IP address of the user's last successful logon
- Number of unsuccessful logon attempts since the last successful logon

(APP3660: CAT III) The Designer will ensure the application has a capability to notify the user on login of date and time of the user's last unsuccessful logon, IP address of the user's last unsuccessful logon, date and time of the user's last successful logon, IP address of the user's last successful logon, and number of unsuccessful logon attempts since the last successful logon.

(APP3670: CAT II) The Designer will ensure the application has a capability to display the user's time and date of the last change in data content.

3.15.2 Access for Need-to-Know

Access to need-to-know information requires individual authentication with authorized or validated need-to-know access established. All access to need-to-know information will be audited.

(APP3680.1: CAT II) The Designer will ensure the application design includes audits on all access to need-to-know information.

Restricted access to need-to-know information is made available only to an authorized community of interest. Authorized users must present an individual authenticator and have either an authorized or validated need-to-know.

(APP3680.2: CAT II) The Designer will ensure the application logs all failed access attempts to need-to-know information.

3.15.3 Audit Record Content

The minimum contents for audit records are based on data classifications of:

- Classified
- Sensitive
- Publicly Releasable

Some events may not be collected automatically and may be required to be captured through a manual method such as blocking a terminal or a userid.

3.15.3.1 Publicly Releasable Audit Record Content

(APP3680.3: CAT II) The Designer will ensure the application's publicly releasable data audit records include:

- *Userid*
- *Successful and unsuccessful attempts to access security files*
- *Date and time of the event*
- *Type of event*

3.15.3.2 Sensitive Audit Record Content

(APP3680.4: CAT II) The Designer will ensure the application's sensitive data audit records include:

- *Userid*
- *Successful and unsuccessful attempts to access security files*
- *Date and time of the event*
- *Type of event*
- *Success or failure of event*
- *Successful and unsuccessful logons*
- *Denial of access resulting from excessive number of logon attempts*
- *Blocking or blacklisting a userid, terminal or access port and the reason for the action*
- *Activities that might modify, bypass, or negate safeguards controlled by the system*

3.15.3.3 Classified Audit Record Content

(APP3680.5: CAT II) The Designer will ensure the application's classified data audit records include:

- *Userid*
- *Successful and unsuccessful attempts to access security file*
- *Date and time of the event*
- *Type of event*
- *Success or failure of event*
- *Successful and unsuccessful logons*
- *Denial of access resulting from excessive number of logon attempts*
- *Blocking or blacklisting a userid, terminal or access port, and the reason for the action*
- *Activities that might modify, bypass, or negate safeguards controlled by the system*
- *Data required to audit the possible use of covert channel mechanisms*
- *Privileged activities and other system-level access*
- *Starting and ending time for access to the system*
- *Security relevant actions associated with periods of activity where security labels or categories of information are processed or changed*

3.15.4 Audit of Security Label Changes for Classified Data

The application automatically records the creation, deletion, or modification of confidentiality or integrity labels, if required by the information owner.

(APP3680.6: CAT III) The Designer will ensure the application creates an audit trail for addition, deletion, or change of the confidentiality or integrity labels as designated by the information owner.

3.15.5 Audit Trail Protection

The contents of audit trails are protected against unauthorized access, modification, or deletion.

(APP3690.1: CAT II) The Designer will ensure the audit trail is readable only by the application and auditors.

(APP3690.2: CAT II) The Designer will ensure the audit trail is protected against modification or deletion except by the application and auditors.

3.16 Mobile Code

The DoDI 8552.01 establishes and implements policy on using mobile code in DoD information systems.

DoDI defines mobile code as "software obtained from remote systems, transferred across a network, and then downloaded and executed on a local system without explicit installation or execution by the recipient." Mobile code falling outside the scope of the mobile code policy is not exempt to the security requirements contained in this document or other STIGs.

Mobile code originating from and traveling exclusively within a single enclave boundary is exempt from the requirements of the mobile code policy.

The Mobile Code Policy can be found on the following web site:
http://www.dtic.mil/whs/directives/corres/pdf/855201p.pdf.

3.16.1 Category 1 Mobile Code

There are two subgroups of Category 1 mobile code technologies:

- Category 1A
- Category 1X

3.16.1.1 Category 1A Mobile Code

The use of Category 1A digitally signed mobile code is allowed in DoD information systems.

All Category 1A mobile code that resides on DoD-owned or DoD-controlled servers shall be digitally signed with a DoD code-signing certificate prior to being installed on the servers.

All Category 1A mobile code which was signed with a code-signing certificate shall be obtained from a trusted source and designated as trusted by the recipient's component.

All Category 1A mobile code's digital signature is properly validated by the client runtime environment prior to the execution of the mobile code.

The following mobile code technologies are assigned to Category 1A:

- ActiveX controls
- MS Internet Explorer browser
- PCMan IETab Add-on
- Mobile code scripts executing in Windows Scripting Host (WSH) (e.g., JavaScript, VBScript downloaded via URL file reference or e-mail attachments, etc.). When JavaScript and VBScript execute within the browser, they are Category 3; however, when they execute in WSH, they are Category 1A.

The use of unsigned Category 1A mobile code in DoD information systems shall be prohibited.

(APP3700.1: CAT II) The Designer will ensure unsigned Category 1A mobile code is not used in the application.

(APP3700.2: CAT II) The Designer will ensure Category 1A mobile code used in an application is signed with a DoD-approved code-signing certificate.

(APP3700.3: CAT II) The Designer will ensure signed Category 1A mobile code used in an application is obtained from a trusted source and is designated as trusted.

(APP3710.1: CAT II) The Designer will ensure signed Category 1A mobile code signature is validated before executing.

Signed mobile code may need to be periodically signed again as the certificate expires.

3.16.1.2 Category 1X Mobile Code

The following mobile code technologies are assigned to Category 1X:

- HTML applications (e.g., hta files) downloaded as mobile code.
- Scrap objects (e.g., .shs and .shb files).
- Windows and Microsoft Disk Operating System (MS-DOS) batch scripts (.cmd and .bat).
- UNIX shell scripts.
- Binary executables (e.g., .exe files) downloaded as mobile code.
- Mozilla Active X Plugin
- Netscape 8.0/8.1 browser
- Shockwave movies (including Xtras)

(APP3700.4: CAT II) The Designer will ensure Category 1X mobile code is not used in applications.

3.16.2 Category 2 Mobile Code

The following mobile code technologies are assigned to Category 2:

- Java applets and other Java mobile code
- Visual Basic for Applications (VBA) (e.g., Microsoft Office macros, also used by Corel Office)
- LotusScript (e.g., Lotus Notes scripts)
- PerfectScript (e.g., Corel Office macros)
- Postscript
- Mobile code executing in .NET Common Language Runtime
- Rich Internet Applications
- Portable Document Format (PDF)
- Flash animations (e.g., .swf and .spl files) executing in the Shockwave Flash Plugin

3.16.2.1 Category 2 Mobile Code in Constrained Environment

Unsigned Category 2 mobile code executing in a constrained execution environment without access to local system and network resources (e.g., file system, Windows Registry, network connections other than to its originating server, etc.) may be used in DoD information systems.

(APP3720: CAT II) The Designer will ensure unsigned Category 2 mobile code executing in a constrained environment has no access to local system and network resources.

3.16.2.2 Category 2 Mobile Code not in Constrained Environment

The DoDI 8552.01 states "Category 2 mobile code not executing in a constrained execution environment may be used in DoD information systems if the mobile code is obtained from a trusted source over an assured channel using at least one of the following measures:

- Code Signing: The mobile code was digitally signed with a code-signing certificate that is designated as trusted by the recipient's Component. The mobile code's digital signature is properly validated by the client runtime environment prior to the execution of the mobile code.
- Secure Sockets Layer (SSL) Connection: The mobile code was downloaded over an SSL connection from a trusted SSL web server using a DoD or trusted commercial SSL server certificate.
- TLS Connection: The mobile code was downloaded over a TLS connection from a trusted TLS web server using a DoD or trusted commercial TLS server certificate.
- IPSec Combined with Mutual Authentication: The mobile code was downloaded from a trusted Web server over an encrypted IPSec connection that establishes mutual authentication using a DoD or trusted commercial certificate."

(APP3700.5: CAT II) The Designer will ensure signed Category 2 mobile code used in an application is signed with a DoD-approved code-signing certificate.

(APP3700.6: CAT II) The Designer will ensure Category 2 mobile code not executing in a constrained execution environment is obtained from a trusted source over an assured channel using at least one of the following measures:

1. *The mobile code was digitally signed with a code-signing certificate that was designated as trusted by the recipient's component.*
2. *The mobile code was downloaded over an SSL connection from a trusted SSL web server using a DoD or trusted commercial SSL server certificate.*
3. *The mobile code was downloaded over a TLS connection from a trusted TLS web server using a DoD or trusted commercial TLS server certificate.*
4. *The mobile code was downloaded from a trusted web server over an encrypted IPSec connection that establishes mutual authentication using a DoD or trusted commercial certificate.*

Note: In an SSL, TLS, or IPSec connection, the application is susceptible to downloading malicious code from phishing attacks on applications having XSS errors.

(APP3710.2: CAT II) The Designer will ensure the signed Category 2 mobile code signature is validated before executing.

Signed mobile code may need to be periodically signed again as the certificate expires.

3.16.3 Category 3 Mobile Code

Category 3 mobile code technologies may be used without restrictions in DoD information systems.

The following mobile code technologies are assigned to Category 3 mobile codes:

- JavaScript, including Jscript and ECMAScript variants, when executing in the browser
- VBScript, when executing in the browser

3.16.4 Emerging Mobile Code

The DoDI 8552.01 states "Emerging mobile code technologies refer to all mobile code technologies, systems, platforms, or languages whose capabilities and threat level have not yet undergone a risk assessment and have not been assigned to one of the a risk categories above. Because of the uncertain risk, the use of emerging mobile code technologies in DoD information systems is prohibited."

(APP3730: CAT II) The Designer will ensure uncategorized or emerging mobile code is not used in applications.

If an emerging mobile code technology is used by a DoD application, the DoD organization shall nominate the mobile code technology to undergo a risk assessment performed by NSA.

3.16.5 Mobile Code in E-mail

Applications embedding mobile code in e-mail attachments should only use mobile code which does not automatically execute when the user opens the e-mail message and the e-mail attachment.

(APP3740: CAT II) The Designer will ensure the application only embeds mobile code in e-mail that does not execute automatically when the user opens the e-mail body or attachment.

3.16.6 New Procurement and Development Efforts

All development of new types of mobile code technologies require a mobile code risk mitigation plan detailing the measures incorporated to mitigate the risk.

(APP3750: CAT II) The Designer will ensure development of new mobile code includes measures to mitigate the risks identified.

3.17 Service Oriented Architecture (SOA)

SOA is the underlying infrastructure supporting communication between services discoverable through a service broker providing programmatically-defined interfaces. Requirements for web services are based on NIST's Guide to Secure Web Services.

Figure 3-6 represents the basic components and service requests of an SOA.

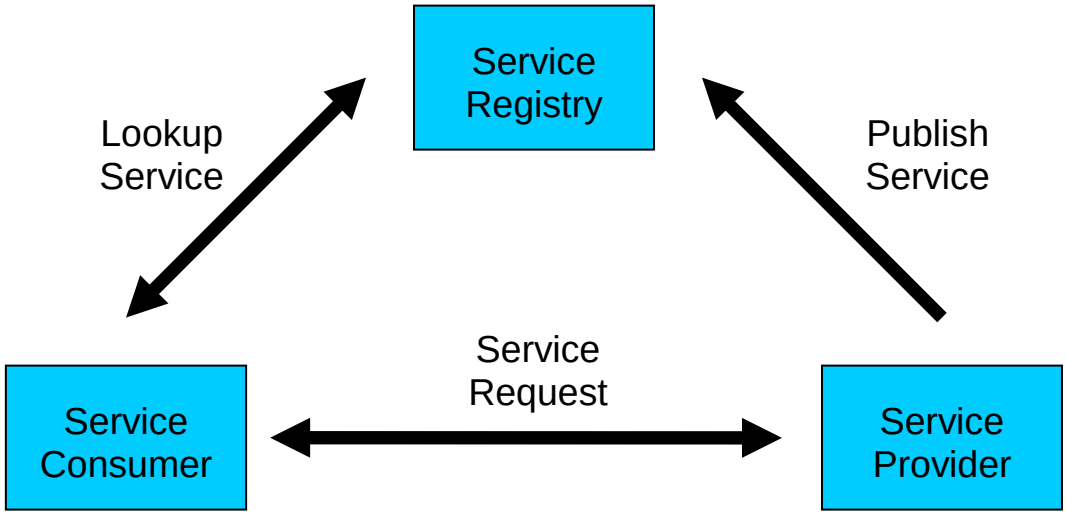

Figure 3-6. SOA Components and Service Requests

Designers can implement SOA using various technologies, including Remote Procedure Call (RPC), Distributed Component Object Model (DCOM), Common Object Requesting Broker Architecture (CORBA), Windows Communication Foundation (WCF) or Web Services. Most SOAs are implemented with web services technologies.

3.17.1 Web Services

Organization for the Advancement of Structured Information Standards (OASIS) is a consortium that produces web services and security standards. Also the World Wide Web Consortium (W3C) develops specifications, guidelines, software, and tools for various Internet technologies. W3C defines a web service as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically, Web Services Description Language (WSDL)). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HyperText Transfer Protocol (HTTP) with an Extensible Markup Language (XML) serialization in conjunction with other web-related standards."

The requirements in this section apply to both classified and unclassified applications.

3.17.1.1 Web Service Availability

Before the introduction of web services, a web application could only be accessed through its supporting web server. Now, with the advent of web services, each web services can be accessed individually. A critical concern of splitting a web application into separate web services is that the application will now provide multiple attack points and multiple points of failure. Web services increase the need to be redundant and highly available.

Because multiple web services may be required in a deployed web application, web services should be designed to detect application level attack patterns and react to these attacks. Network DoS attacks are addressed in the Network STIG.

(APP3760: CAT II) The Designer will ensure web services are designed and implemented to recognize and react to the attack patterns associated with application-level DoS.

Because of the significant possibility of DoS attacks, web services should be designed to have redundancy of critical functions. A web service may be deployed across multiple servers, virtual machines, or network interfaces.

(APP3770: CAT II) The Designer will ensure the web service design includes redundancy of critical functions.

To meet the requirements for MAC I systems, critical web service functions should be implemented by different algorithms. All web services implemented with the same algorithms can be potentially disrupted using the same attack vector. Web services implemented using different algorithms decreases the possibility of a single attack vector disrupting the entire application. Critical functions are functions that prevent stated mission objectives of the application from being performed.

(APP3780: CAT II) The Designer will ensure web service design of critical functions is implemented using different algorithms to prevent similar attacks from a complete application level DoS.

For MAC I and MAC II systems to meet availability requirements, the web service should be designed to prioritize requests. Prioritizing requests may be accomplished based on location or IP address, user roles, user location, or other applicable prioritization.

(APP3790: CAT II) The Designer will ensure web services are designed to prioritize requests to increase availability of the system.

3.17.1.2 Web Service Deadlock

Because of the complexity and the highly distributive nature of web services, care should be taken to prevent the possibility of web services getting into a deadlock situation or a potential recursion situation.

(APP3800: CAT II) The Designer will ensure execution flow diagrams are created and used to mitigate deadlock and recursion issues.

3.17.1.3 Extensible Markup Language (XML) Injection

XML injection is an attack technique similar to SQL injection. This attack can inject content into the XML database and can also be used to compromise the integrity of an XML application or service.

(APP3810: CAT I) The Designer will ensure the application is not vulnerable to XML injection.

3.17.2 Representational State Transfer (REST) Web Services

REST is a style of software architecture commonly used to implement web services. REST web services concentrates scalability and performance of the overall architecture. Although, there are no specific requirements related to REST web services, all requirements in this STIG are applicable.

3.17.2.1 Conveying End User Authentication with REST

Security Assertion Markup Language (SAML) Assertions can be sent through a SAML HTTP Artifact Binding or SAML URI Binding using a REST-style web service. The HTTP artifact binding allows the SAML requestor and responder to communicate over HTTP. This allows REST-style web services to use SAML standard for security services. A URI binding uses a HTTP GET request to retrieve a SAML assertion.

3.17.2.2 Signed REST-Based Messages

For REST-styled web service messages, there is no direct standard to apply XML-Signature. Messages should analyzes carefully and address signature requirements. All integrity requirements must still be considered when using REST-styled web services.

3.17.2.3 Encrypted REST-Based Messages

For REST-styled web services messages, there is no direct standard to apply XML-Encryption. Thus, data modelers should build data schemas to account for security implications. Mutually-

authenticated TLS connections should be considered for confidentiality of REST-styled web service messages.

3.17.2.4 HTTP Artifact Binding

Artifact Resolution Protocol and SAML SOAP Binding (over HTTP) are used by the HTTP Artifact Binding to resolve a SAML message.

Samples, provided in the following subsections, show a service provider sending an artifact resolution request to an identity provide.

3.17.2.4.1 HTTP Redirect

The service provider initially issues an HTTP redirect to transmit the artifact.

https://www.disa.mil/SAML2/Artifact?SAMLart=artifact

3.17.2.4.2 Artifact Resolve Request

The identity provider sends an artifact resolve request directly to the service provided through a secondary channel.

```
<samlp:ArtifactResolve
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        ID="6474985"
        Version="2.0"
        IssueInstant="2009-12-31T10:00:00Z"
        Destination=https://www.disa.mil/SAML2/ArtifactResolution>
        <saml:Issuer>https://www.disa.mil/SAML2</saml:Issuer>
        <samlp:Artifact>artifact</samlp:Artifact>
</samlp:ArtifactResolve>
```

3.17.2.4.3 Artifact Response

The service provider returns an artifact response element containing the referenced authentication request.

```
<samlp:ArtifactResponse
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        ID="6474986"
        InResponseTo="6474985"
        Version="2.0"
        IssueInstant="2009-12-31T10:00:02Z "
        Destination="https://www.disa.mil/SAML2/Artifact">
```

```
<saml:Issuer>https://www.disa.mil/SAML2</saml:Issuer>

<samlp:Status>
        <samlp:StatusCode
        Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</samlp:Status>

<samlp:AuthnRequest
        ID="6474987"
        Version="2.0"
        IssueInstant="2009-12-31T10:00:02Z”
        ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
        AssertionConsumerServiceURL= https://www.disa.mil/SAML2/Artifact>

        <saml:Issuer> https://www.disa.mil/SAML2</saml:Issuer>

        <samlp:NameIDPolicy
        AllowCreate="false"
        Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"/>
</samlp:AuthnRequest>

</samlp:ArtifactResponse>
```

3.17.2.5 SAML URI Binding

The SAML URI Binding provides the ability to retrieve SAML assertions from a REST-style web service. The response is represented as a URI reference that contains a SAML assertion. The assertion is sent to a SAML responder for verification.

The following is an example of an URI Binding used to retrieve SAML assertions:

Request

```
GET /SamlIDProvider?ID=28738473 HTTP/1.1
Host: www.disa.mil
```

Response

```
<saml:Assertion ID=”28738473”
….
</saml:Assertion>
```

3.17.3 SOAP Web Services

Simple Object Access Protocol (SOAP) web services use XML as the base standard. SOAP web services use SOAP Envelopes as message standards. For discovery, SOAP (WSDL) is used for describing messages, operations, and where the web service can be accessed. Universal Description, Discovery, and Integration (UDDI) is an XML-based registry used to list available SOAP web services.

Figure 3-7 provides a graphical depiction of SOAP web services.

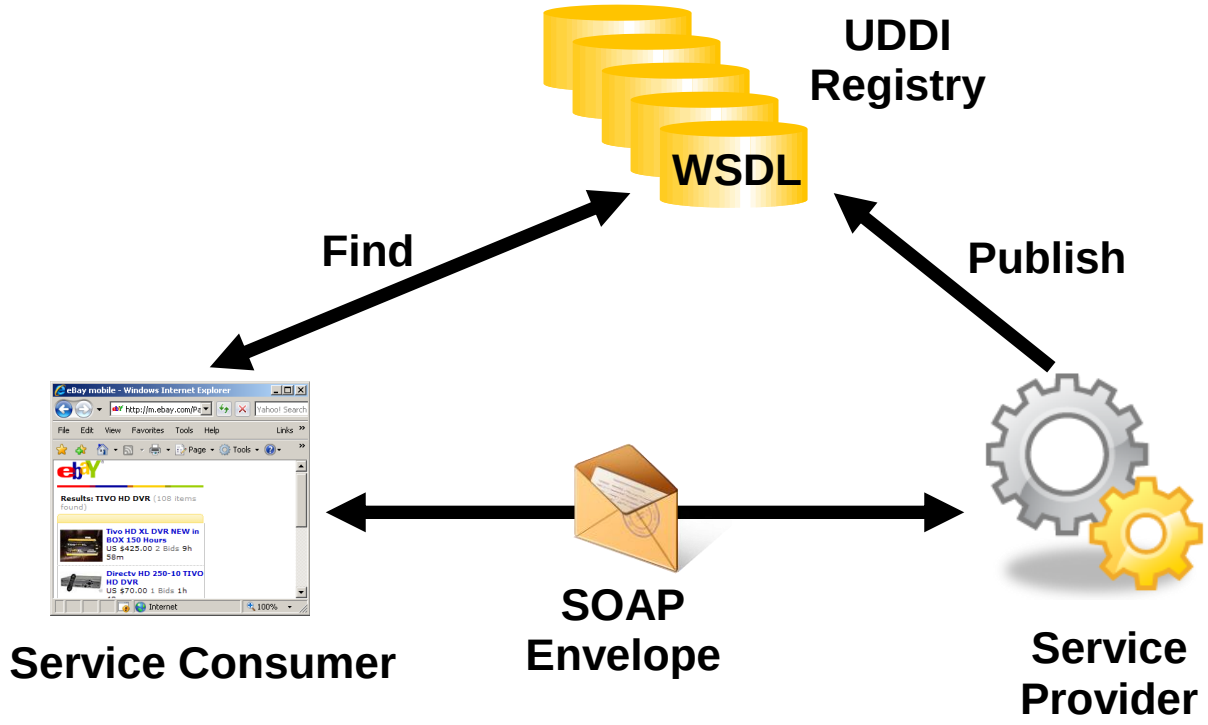

Figure 3-7. SOAP Web Services

3.17.3.1 SOAP Messaging

SOAP is an XML-based stateless messaging specification used to exchange messages between web services. The SOAP specification was developed by the W3C and the latest versions are available at the following web site: http://www.w3.org/2000/xp/Group/.

(APP3820: CAT I) The Designer will ensure web services provide a mechanism for detecting resubmitted SOAP messages.

3.17.3.2 Universal Description, Discovery, and Integration (UDDI)

UDDI is an open industry initiative, sponsored by OASIS. UDDI is used for defining web services in a repository using XML to describe services interfaces. The XML schema used to define web services is WSDL. WSDL describes web service end-points and their messages. UDDI communicates via SOAP messages.

UDDI registry entries must be signed to establish the identity of the publisher and ensure the integrity of the directory of web services.

(APP3830.1: CAT II) The Designer will ensure digital signatures exist on UDDI registry entries to verify the publisher.

Early versions of UDDI did not support the signing of registry entries. UDDI, Version 3 0 or newer, will be used to digitally sign web service registry entries.

(APP3840.1: CAT II) The Designer will ensure UDDI versions are used supporting digital signatures of registry entries.

UDDI registry entries should be protected from unauthorized modifications. Consequently, publishing registry entries is restricted to authenticated users only.

(APP3850.1: CAT II) The Designer will ensure UDDI publishing is restricted to authenticated users.

3.17.3.3 Message Integrity

Digitally-signed messages provide message integrity and authenticity of the message independent of the transport layer. Figure 3-8 illustrates an example of a signed SOAP message with the specific parts of the message being signed. The X.509 Certificate reference may also reference an external X.509 certificate. Also, SAML assertions are optional components to a signed message.

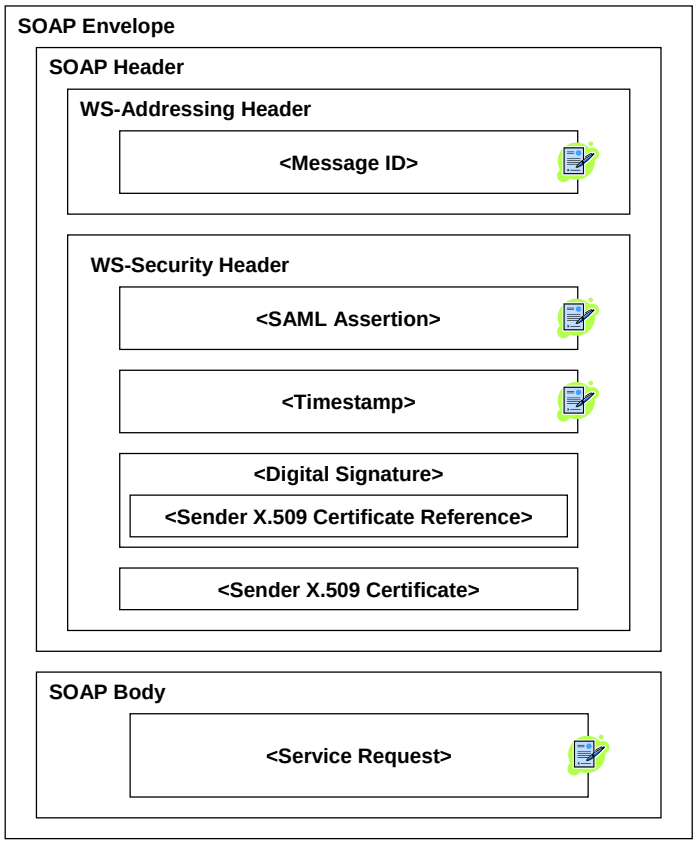

Figure 3-8. Example Signed SOAP Message

(APP3860: CAT II) The Designer will ensure SOAP messages requiring integrity sign the following message elements:

- Message ID
- Service Request
- Timestamp
- SAML Assertion.

Note: SAML assertions are optionally included in messages.

XML Signature can be used to sign the message so unauthorized modifications to a message cannot occur during transit.

3.17.3.4 Message Confidentiality

When using web services there are two options to provide confidentiality for service messages:

- Confidentiality can be provided at several layers of the communications stack, including the transport layer (e.g., TLS), network layer (e.g., IPSEC, inline network encryptors, etc.), and physical/link layer (e.g., link encryptors).
- Confidentiality can be provided at the message level by encrypting the message with XML encryption.

Web services messages may require encryption depending on the data being transmitted. Section 3.7.4 details encryption requirements for data in transit.

3.17.3.5 Timestamp Validations

WS-Security messages should utilize timestamps and expiration times to minimize possible replay attacks. Also, expiration times should be verified against time servers. Timestamps should also be validated as follows:

<wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id=" id-74389508">
<wsu:Created>2009-12-31T20:00:00.000Z</wsu:Created>
<wsu:Expires>2009-12-31T20:05:00.000Z</wsu:Expires>
</wsu:Timestamp>

(APP3870: CAT I) The Designer will ensure when using WS-Security messages use timestamps with creation and expiration times.

(APP3880: CAT I) The Designer will ensure validity periods are verified on all messages using WS-Security or SAML assertions.

3.17.4 SAML

Parts of SAML specifications are reproduced in this section. The SAML specification is made available by OASIS and used within the scope of their copyright notice furnished below.

"OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification, can be obtained from the OASIS Executive Director. OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary

rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director."

3.17.4.1 SAML CORE Specification

SAML enables identity providers to make assertions regarding the identity, attributes, and entitlements of an entity. In most cases, entities will be system users but they may also be other system entities. The SAML specification defines both the structure of SAML assertions associated set of protocols processing rules SAML messages are encoded in XML and use of XML namespaces.

The SAML Core Specification can be found at the OASIS web site:
http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

3.17.4.1.1 Schema Header and Namespace Declarations

The following schema fragment defines the XML namespaces and other header information for the SAML Core Specification:

<schema targetNamespace="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
elementFormDefault="unqualified"
attributeFormDefault="unqualified"
blockDefault="substitution"
version="2.0">
<import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd"/>
<import namespace=http://www.w3.org/2001/04/xmlenc#
schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd"/>
<annotation>
<documentation>
Location: http://docs.oasis-open.org/security/saml/v2.0/
Revision history:
V1.0 (November, 2002):
Initial Standard Schema.
V1.1 (September, 2003):
Updates within the same V1.0 namespace.
V2.0 (March, 2005):
New assertion schema for SAML V2.0 namespace.
</documentation>
</annotation>
…
</schema>

3.17.4.1.2 SAML Assertions

An assertion is a package of information that supplies zero or more statements made by an asserting party. There are three categories of assertion statements which can be created by an asserting party defined in the SAML Core Specification:

- Authentication: The assertion subject was authenticated by a particular means at a particular time.
- Attribute: The assertion subject is associated with the supplied attributes.
- Authorization Decision: The assertion subject has been granted or denied access to the specified resource.

3.17.4.1.3 Name Identifiers

Name identifiers contain descriptive identifiers for subjects and the issuers of assertions and protocol messages. SAML provides name qualifiers to provide a means to associate identifiers from different identity providers without collision.

Name identifiers may be encrypted to provide confidentiality in cases where the identifier may be transmitted via an intermediary.

3.17.4.1.3.1 Element <BaseID>

The <BaseID> element includes the following attributes for use by extended identifier representations:

- NameQualifier [Optional]
- SPNameQualifier [Optional]

The NameQualifier and SPNameQualifier attributes should be omitted unless the identifier's type definition explicitly defines their use and semantics.

3.17.4.1.3.2 Complex Type NameIDType

The NameIDType complex type is used when an element serves to represent an entity by a string-valued name:

- NameQualifier [Optional]
- SPNameQualifier [Optional]
- Format [Optional]
- SPProvidedID [Optional]

The NameQualifier and SPNameQualifier attributes should be omitted unless the element or format explicitly defines their use and semantics.

3.17.4.1.3.3Element <NameID>

The <NameID> element is of type NameIDType and is used in SAML assertion constructs <Subject> and <SubjectConfirmation> elements and protocol messages.

3.17.4.1.3.4Element <EncryptedID>

The <EncryptedID> element is of type EncryptedElementType and carries the content of an unencrypted identifier element in encrypted fashion. The <EncryptedID> element contains the following elements:

- <xenc:EncryptedData> [Required]
- <xenc:EncryptedKey> [Zero or More]

3.17.4.1.3.5Element <Issuer>

The <Issuer> element and the complex type <NameIDType> provide information about the issuer of a SAML assertion or protocol message.

3.17.4.1.4 Assertions

3.17.4.1.4.1Element <AssertionIDRef>

The <AssertionIDRef> element is a unique identifier for the SAML assertion.

(APP3890: CAT II) The Designer shall ensure each unique asserting party provides unique assertion ID references for each SAML assertion.

3.17.4.1.4.2Element <AssertionURIRef>

The <AssertionURIRef> element makes a reference to a SAML assertion by a URI reference.

3.17.4.1.4.3Element <Assertion>

The <Assertion> element is of the <AssertionType> complex type. This type specifies the following elements and attributes:

- ID [Required]
- IssueInstant [Required]
- <Issuer> [Required]
- <ds:Signature> [Optional]
- <Subject> [Optional]
- <Conditions> [Optional]
- <Advice> [Optional]

Zero or more of the following statement elements:

- <Statement>
- <AuthnStatement>
- <AuthzDecisionStatement>
- <AttributeStatement>

3.17.4.1.4.4Element <EncryptedAssertion>

The <EncryptedAssertion> element represents an assertion in encrypted by the XML Encryption Syntax and Processing specification [XMLEnc]. The <EncryptedAssertion> element contains the following elements:

- <xenc:EncryptedData> [Required]

The Type attribute should be present and, if present, must contain a value of http://www.w3.org/2001/04/xmlenc#. The encrypted content must contain an element that has a type of or derived from AssertionType <xenc:EncryptedKey> [Zero or More].

Each wrapped key should include a Recipient attribute that specifies the entity for which the key has been encrypted. The value of the Recipient attribute should be the URI identifier of a SAML system entity.

Encrypted assertions are intended as a confidentiality protection mechanism when the unencrypted value passes through an intermediary.

(APP3900: CAT II) The Designer shall ensure encrypted assertions or equivalent confidentiality when assertion data is passed through an intermediary and confidentiality of the assertion data is required to pass through the intermediary.

3.17.4.1.5 Subjects

This section defines the element subject.

3.17.4.1.5.1Element <Subject>

The optional <Subject> element specifies the entity of the subject statements (zero or more). It contains an identifier, a series of one or more subject confirmations, or both as follows:

- <BaseID>, <NameID>, or <EncryptedID> [Optional]
- <SubjectConfirmation> [Zero or More]

3.17.4.1.5.1.1 Element <SubjectConfirmation>

The <SubjectConfirmation> element provides the means for a relying party to verify the correspondence of the subject of the assertion with the party with whom the relying party is communicating. It contains the following attributes and elements:

- Method [Required]
- <BaseID>, <NameID>, or <EncryptedID> [Optional]
- <SubjectConfirmationData> [Optional]

3.17.4.1.5.1.2 Element <SubjectConfirmationData>

The <SubjectConfirmationData> element has the SubjectConfirmationDataType complex type as follows:

- NotBefore [Optional]
- NotOnOrAfter [Optional]
- Recipient [Optional]
- InResponseTo [Optional]
- Address [Optional]
- Arbitrary attributes
- Arbitrary elements

(APP3910: CAT I) The Designer shall use the NotOnOrAfter when using the SubjectConfirmation element in a SAML assertion.

3.17.4.1.5.1.3 Complex Type KeyInfoConfirmationDataType

The KeyInfoConfirmationDataType complex type constrains a <SubjectConfirmationData> element to contain one or more <ds:KeyInfo> elements that identify cryptographic keys that are used in some way to authenticate an attesting entity. The particular confirmation method must define the exact mechanism by which the confirmation data can be used. The optional attributes defined by the SubjectConfirmationDataType complex type may also appear. This complex type, or a type derived from it, SHOULD be used by any confirmation method that defines it.

Each <ds:KeyInfo> element must identify a single cryptographic key in accordance with XML Signature. Multiple keys may be identified with separate <ds:KeyInfo> elements, such as when a principal uses different keys to confirm itself to different relying parties.

3.17.4.1.5.1.4 Example of a Key-Confirmed <Subject>

The following is an example of a key-confirmed <subject>:

```
<Subject>
        <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
        scott@example.org
        </NameID>
        <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
        <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
        <ds:KeyInfo>
        <ds:KeyName>Scott's Key</ds:KeyName>
        </ds:KeyInfo>
        </SubjectConfirmationData>
        </SubjectConfirmation>
</Subject>
```

3.17.4.1.6 Conditions

This section defines the SAML condition structure and places constraints on SAML assertions.

The <Conditions> element MAY contain the following elements and attributes:

- NotBefore [Optional]
- NotOnOrAfter [Optional]
- <Condition> [Any Number]
- <AudienceRestriction> [Any Number]
- <OneTimeUse> [Optional]
- <ProxyRestriction> [Optional]

(APP3920: CAT I) The Designer shall use the both the NotBefore and NotOnOrAfter elements or OneTimeUse element when using the Conditions element in a SAML Assertion.

Note: If SAML assertions are used, the NotBefore and NotAfter fields can be substituted for the WS-Security timestamp.

(APP3930: CAT II) The Designer shall ensure if a <OneTimeUse> element is used in an assertion, there is only one used in <Conditions> element of an assertion.

3.17.4.1.7 Advice

This section defines the SAML constructs that contain additional information about an assertion that an asserting party wishes to provide to a relying party.

3.17.4.1.7.1Element <Advice>

The <Advice> element contains any additional information that the asserting party wishes to provide.

The <Advice> element contains a mixture of zero or more <Assertion>, <EncryptedAssertion>, <AssertionIDRef>, and <AssertionURIRef> elements, and namespace-qualified elements in other non-SAML namespaces.

3.17.4.1.8 Statements

The following sections define the SAML authentication, attribute, and authorization statements.

3.17.4.1.8.1Element <Statement>

The <Statement> elements StatementAbstractType complex type is abstract and is only usable based on a derived type.

3.17.4.1.8.2Element <AuthnStatement>

The <AuthnStatement> element describes a statement by the asserting party asserting that the assertion subject was authenticated by a particular means at a particular time. Assertions containing <AuthnStatement> elements MUST contain a <Subject> element. It is of type AuthnStatementType, which extends StatementAbstractType with the addition of the following elements and attributes:from <AuthnStatement> for V2.0 of SAML:

- AuthnInstant [Required]
- SessionIndex [Optional]
- SessionNotOnOrAfter [Optional]
- <SubjectLocality> [Optional]
- <AuthnContext> [Required]

The asserting party should generate random session indexes (SessionIndex) to prevent attackers from calculation of the subsequent session index values. These unique values will be used for subsequent statements given to the same distinct session. These session indexes should be generated with FIPS-approved random number generators.

(APP3940: CAT II) The Designer will ensure the asserting party uses FIPS-approved random numbers in the generation of SessionIndex in the SAML Element <AuthnStatement>.

When the SessionIndex is tied to privacy data (e.g., attributes containing privacy data), the message should be encrypted.

(APP3950: CAT II) The Designer shall ensure messages are encrypted when the SessionIndex is tied to privacy data.

3.17.4.1.8.2.1 Element <SubjectLocality>

The <SubjectLocality> element specifies the DNS domain name and IP address for the system from which the assertion subject was authenticated. It has the following attributes:

- Address [Optional]
- DNSName [Optional]

The DNS name of the system from which entity identified by the subject was authenticated.

3.17.4.1.8.2.2 Element <AuthnContext>

The <AuthnContext> element specifies the context of an authentication event. The element can contain an authentication context class reference, an authentication context declaration or declaration reference, or both. The <AuthnContext> has the following elements:

- <AuthnContextClassRef> [Optional]
- <AuthnContextDecl> or <AuthnContextDeclRef> [Optional]
- <AuthenticatingAuthority> [Zero or More]

3.17.4.1.8.3Element <AttributeStatement>

The <AttributeStatement> element describes a statement by the asserting party asserting the entity has the specified attributes. Assertions containing <AttributeStatement> elements MUST contain a <Subject> element.

The <AttributeStatement> contains the following elements:

- <Attribute> or <EncryptedAttribute> [One or More]

3.17.4.1.8.3.1 Element <Attribute>

The <Attribute> element identifies an attribute name and an optional value or values.

The <Attribute> element contains the following XML attributes:

- Name [Required]
- NameFormat [Optional]
- FriendlyName [Optional]
- Arbitrary attributes
- <AttributeValue> [Any Number]

3.17.4.1.8.3.2 Element <EncryptedAttribute>

The <EncryptedAttribute> element represents an encrypted SAML attribute. The <EncryptedAttribute> element contains the following elements:

- <xenc:EncryptedData> [Required]
- <xenc:EncryptedKey> [Zero or More]

3.17.4.1.8.4Element <AuthzDecisionStatement>

The <AuthzDecisionStatement> element describes a statement by the asserting party asserting a request for access by the assertion subject to the specified resource has resulted in the specified authorization decision on the basis of some optionally specified evidence. Assertions containing <AuthzDecisionStatement> elements MUST contain a <Subject> element.

Ambiguity in URI references can cause a resource to be accessed by changing or modifying part of the URI to access the system.

SAML authorities should encode all resource URI references in normalized form. Relying parties should convert resource URI references to normalized form prior to processing.

The <AuthzDecisionStatement> element has the following elements and attributes:

- Resource [Required]
- Decision [Required]
- <Action> [One or more]
- <Evidence> [Optional]

3.17.4.1.8.4.1 Simple Type DecisionType

The DecisionType simple type defines the possible values to be reported as the status of an authorization decision statement. They are as follows:

- Permit
- Deny
- Indeterminate

3.17.4.1.8.4.2 Element <Action>

The <Action> element specifies an action on the specified resource for which permission is sought. The <Action> element has the following attribute:

- Namespace [Optional]

3.17.4.1.8.4.3 Element <Evidence>

The <Evidence> element contains one or more assertions or assertion references that the asserting party relied on in issuing the authorization decision. The <Evidence> element contains a combination of one or more of the following elements:

- <AssertionIDRef> [Any number]
- <AssertionURIRef> [Any number]
- <Assertion> [Any number]
- <EncryptedAssertion> [Any number]

3.18 IPv6

A memorandum was issued by the Assistant Secretary of Defense on Internet Protocol Version 6 (IPv6) Policy Update, dated 16 August 2005. This memorandum defines milestone objectives for enterprise-wide deployment of IPv6. This memorandum also defines an IPv6-capable product as *a system or product capable of receiving, processing and forwarding IPv6 packets and/or interfacing with other systems and protocols in a manner similar to IPv4.*

Criteria to be considered IPv6-capable are:

1. *Conformant with the IPv6 standards profile contained in the DoD IT Standards Registry (DISR)*
2. *Maintaining interoperability in heterogeneous environments with IPv4*
3. *Commitment to upgrade as the IPv6 standard evolves*
4. *Availability of contractor/vendor IPv6 technical support*

DoD IPv6 Standard Profiles for IPv6 Capable Products, Version 3.0, dated 13 June 2008, is located at the Joint Interoperability Test Command (JITC) web site: http://jitc.fhu.disa.mil/apl/ipv6/pdf/disr_ipv6_product_profile_v3.pdf.

(APP3960: CAT II) The Designer will ensure the application is compliant with all DISR IPv6 profiles.

Supporting application services (e.g., FTP, Network File system (NFS), HTTP, etc.) should be upgraded to provide support for IPv6. Patch and update type services should also be examined to determine if these services can be executed in an IPv6 only mode. Operating systems and all network service interfaces should be updated to reflect IPv6 addressing.

(APP3970: CAT II) The Designer will ensure supporting application services and interfaces have been designed or upgraded for IPv6 transport.

The application should be compliant with RFC 4038 Application Aspects of IPv6 Transition.

When upgrading an IPv4 application using broadcast, IPv6 multicast addresses can be used as an alternative depending on the specific design and implementation.

(APP3980: CAT II) The Designer will ensure the application is compliant with IPv6 multicast addressing and features an IPv6 network configuration options as defined in RFC 4038.

The application should be compliant with RFC 1884 that describes the IPv6 addressing architecture.

The application should be designed to support IPv6 configuration options to support IPv6 specific addressing types:

1. Unicast Addresses
   - Global Addresses
   - Site-Local Addresses
   - Link-Local Addresses
   - Special addresses (e.g., loopback addresses, IPv4-address mapped spaces, IPv6 6to4 addresses, etc.)
2. Multicast Addresses
3. Anycast

All user interfaces and system management interfaces should be updated to support IPv6 address types.

(APP3990: CAT II) The Designer will ensure the application is compliant with the IPv6 addressing scheme as defined in with RFC 1884.

4. SOFTWARE CONFIGURATION MANAGEMENT (SCM)

SCM is the process of controlling software by managing the versions of all components and their relationships. All system software must be approved by the Configuration Control Board (CCB). The Release Manager has the primary responsibility of SCM.

Adherence to the SCM process will limit unauthorized individuals from accessing and making unauthorized modifications and potentially malicious changes to code.

(APP4010: CAT III) The Release Manager will ensure the access privileges to the configuration management (CM) repository are reviewed every 3 months.

4.1 SCM Plan

The Release Manager will develop a SCM plan to include the following:

(APP4030.1: CAT II) The Release Manager will develop an SCM plan describing the configuration control and change management process of objects developed and the roles and responsibilities of the organization.

(APP4030.2: CAT III) The Release Manager will ensure the SCM plan identifies all objects created during the development process subject to configuration control.

(APP4030.3: CAT II) The Release Manager will ensure the SCM plan maintains procedures for identifying individual application components, as well as, entire application releases during all phases of the software development lifecycle.

(APP4030.4: CAT III) The Release Manager will ensure the SCM plan identifies and tracks all actions and changes resulting from a change request from initiation to release.

(APP4030.5: CAT III) The Release Manager will ensure the SCM plan contains procedures to identify, document, review, and authorize any change requests to the application.

(APP4030.6: CAT III) The Release Manager will ensure the SCM plan defines the responsibilities, the actions to be performed, the tools, techniques and methodologies, and defines an initial set of baseline software components.

(APP4030.7: CAT III) The Release Manager will ensure the SCM plan objects have security classifications labels.

(APP4030.8: CAT II) The Release Manager will ensure the SCM plan identifies tools and version numbers used in the software development lifecycle.

(APP4030.9: CAT III) The Release Manager will ensure the SCM plan identifies mechanisms for controlled access of simultaneous individuals updating the same application component.

(APP4030.10: CAT II) The Release Manager will ensure the SCM plan assures only authorized changes by authorized persons are possible.

(APP4030.11: CAT III) The Release Manager will ensure the SCM plan identifies mechanisms to control access and audit changes between different versions of objects subject to configuration control.

(APP4030.12: CAT II) The Release Manager will ensure the SCM plan identifies mechanisms to track and audit all modifications of objects under configuration control. Audits will include the originator and date and time of the modification.

4.2 Configuration Control Board (CCB)

The Release Manager will establish a CCB to manage the CM process. The IAM will be a member of the CCB. The CCB will meet regularly to determine the priority change requests. The CCB determines when the change requests will be implemented. The Program Manager should designate a security lead to be part of the CCB.

(APP4040.1: CAT II) The Release Manager will establish a CCB managing the CM process.

(APP4040.2: CAT II) The Release Manager will ensure the IAM is a member of the CCB.

(APP4040.3: CAT III) The Release Manager will ensure the CCB meets at least every release cycle or more often.

4.3 File Integrity

When application files are transferred from one environment to another, there is the potential for malware to be introduced into the application code or the application binaries themselves. Care must be taken to validate the integrity of application files prior to deploying applications into a production environment.

To ensure file integrity can be determined at a later date, the application Release Manager must cryptographically hash application files and/or the application package using a strong hashing algorithm prior to releasing the application. The hash value must be protected so it cannot be modified and it must be made available to system/application administrators so they are able to use the value in order to validate the integrity of the application files they intend to install.

Prior to the application receiving an ATO/IATO for deployment into a DoD operational network, the application package/files must be validated for integrity to ensure no tampering of application files has occurred. Failure to validate the integrity of application files prior to deploying an application into a production environment may compromise the operational network.

(APP4050: CAT II) The Release Manager must ensure application files are cryptographically hashed prior to deploying applications to DoD operational networks.

5. TESTING

The application testing process is vital in identifying security flaws before the application is released. In addition to the standard functional testing performed, the development team will have at least one tester specifically testing the application for security flaws.

(APP5010: CAT III) The Test Manager will ensure at least one tester is designated to test for security flaws in addition to functional testing.

(APP2160.2: CAT II) The Test Manager will ensure both client and server machines are STIG compliant.

(APP5030: CAT II) The Test Manager will ensure the application does not modify data files outside the scope of the application.

The Test Manager will create, document, and perform test cases for changes in the application.

(APP5040: CAT II) The Test Manager will ensure the changes to the application are assessed for IA and accreditation impact prior to implementation.

5.1 Test Plans and Procedures

The Test Manager will create and update test plans and procedures for each production application release.

(APP5050: CAT II) The Test Manager will ensure tests plans and procedures are created and executed prior to each release of the application or updates to system patches.

(APP5060: CAT II) The Test Manager will ensure tests procedures are created and at least annually executed to ensure system initialization, shutdown, and aborts are configured to ensure the system remains in a secure state.

5.2 Fuzz Testing

Fuzz testing is a testing method that can help uncover reliability and security vulnerabilities in a software product. Fuzz testing relies on building or manufacturing deliberately malformed data and then having the application under test the data. This can often lead to an application crash and percentage of the crashes is, in fact, security vulnerabilities.

It is important that all critical applications, most notably those facing the Internet or those that consume and parse files, be fuzzed.

5.3 Code Coverage

An important aspect of all testing methods, including fuzz testing, is code coverage achieved through the testing process. Code coverage is the percentage of the application code exercised during the testing process. Security flaws often occur in areas of the code not regularly executed, so it is important to keep track of how often a code branch is executed and tested to ensure thorough testing is performed. If code coverage is low, then the tests must be evaluated to determine why code coverage is low, and the tests changed to increase the percentage of code covered by the test cases.

(APP5070: CAT III) The Test Manager will ensure code coverage statistics are maintained for each release of the application.

5.4 Code Reviews

A code review is the process of reviewing application code to locate potential problems with functionality. Security flaws should also be identified during the code review. Any security flaws found will be entered into the defect tracking system, clearly identified as a security defect, and fixed before the application is released. Code reviews are not usually conducted on third-party libraries or products because vendors do not often make their source code available to evaluate. Code reviews are either automated or manual in nature, and there are many commercial companies offering code review services. The most comprehensive reviews will implement both of types of code review.

Application reviewers are objective parties holding no responsibilities of developing the application being certified and accredited. The application reviewer should have a strong background in the languages used by the application and training in identifying security flaws.

If the code is considered proprietary, the application developers will need to provide evidence a code review was performed on the code base and all code review-related requirements are reviewed.

Also, the threat modeling process should feed the code review process, so more significant threats receive higher priority and attention than less significant threats.

(APP5080: CAT II) The Test Manager will ensure a code review is performed before the application is released.

(APP5090: CAT II) The Test Manager will ensure flaws found during a code review are tracked in a defect tracking system.

(APP5100: CAT II) The IAO will ensure active vulnerability scanning is performed.

(APP5110: CAT II) The Test Manager will ensure security flaws are fixed or addressed in the project plan.

Automated Code Review

Automated code reviews can quickly identify weak areas of an application. Depending on the sophistication of the analysis, static analysis tools may be prone to false positives and may miss some classes of security defect. Automated code review procedures should be put in place to disqualify false positives and manually check for security vulnerabilities the tool fails to identify. Static analysis tools are valuable if the code is very large, but it is not a replacement for manual code review. There should be manual code review for exposed code, such as network-facing code and mobile code.

5.4.1.1 Static Analysis Tools

Static analysis tools should provide the following:

- Support for programming languages required.
- Scan for vulnerabilities and report vulnerabilities with a minimum of false positives and false negatives.
- Support a centralized security policy management so all scans use established policies.
- Scan for malicious code detection.
- Support the use of an underlying DBMS to collect, report, export, and analyze scan results.
- Provide remediation for vulnerabilities found.
- Provide measurement metrics for long-term trending of applications.
- Enable collaboration between security teams and development and Quality Assurance (QA).
- Provide customization capabilities to accommodate unique coding styles.
- Correlate dynamic testing to assist in the prioritization of static results.

There are many commercially available static analysis tools. A list of such tools is as follows:

Ounce Labs
Ounce Labs' code review tool (formerly Prexis) analyzes the source code of applications and identifies confirmed vulnerabilities and other potential security issues. Ounce has interfaces for users across the SDLC, from development/QA through Certification and Accreditation (C&A).

Parasoft's Jtest & C++test
Parasoft's Jtest is a Java testing product for development teams building Java Enterprise Edition (EE), SOA, Web, and other Java applications. Parasoft's C++ test provides coding policy enforcement, static analysis, code review, unit, and component testing for C and C++ code.

Klocwork Insight
Klocwork Insight finds software bugs and security vulnerabilities in C, C++, and Java code.

Fortify Source Code Analyzer (SCA)
Fortify SCA finds programming errors and vulnerabilities in 12 programming languages: Ajax (JavaScript), C/C++, Classic ASP, COBOL, ColdFusion, Java, .NET, PHP, PL/SQL, T/SQL and VB6. Fortify also finds errors in code that combines any of these 12 programming languages.

GrammaTech CodeSonar
GrammaTech CodeSonar is a source code analysis tool that performs inter-procedural analysis on C/C++ code and identifies vulnerabilities in programming logic.

Coverity Prevent
Coverity Prevent identifies and resolves the critical security defects in C, C++, and Java source code.

5.4.1.2 Web Application Vulnerability Scanners

Web application vulnerability scanners allow testers and application developers the ability to scan the web applications in a fully operational environment and check for many known security vulnerabilities. Web application vulnerability scanners actively parse URLs from the target web site to find vulnerabilities. These scanners check web applications for common security problems such as SQL injection, cross site scripting, command injection, buffer overflow, session management, and other vulnerabilities.

These tools can be used to satisfy code review requirements based on the security checks provided by the tool. Web application scanners should be used on each web application release prior to deployment to a production environment and periodically thereafter. Scans usually occur after changes to the application are made. A list of such tools is as follows:

WebInspect
WebInspect is a web application vulnerability scanning tool that actively scans web applications for potential security flaws such as buffer overruns, weak cryptography, race conditions, SQL injection, XSS, and others. WebInspect is automatically updated with known hacking techniques with each assessment performed.

Watchfire's AppScan
Watchfire's AppScan is a web application security testing tool that scans and tests for all common web application vulnerabilities. Vulnerabilities scanned for include SQL injection, XSS, buffer overflow, and others.

Fortify Program Trace Analyzer (PTA)
Fortify PTA enables QA organizations to find security vulnerabilities and leaks while conducting functional testing enabling testers to uncover security vulnerabilities in the application with no additional effort. Fortify PTA also pinpoints vulnerabilities to specific lines of code, facilitating remediation. In addition, NIST maintains a list of vulnerability scanners. The list is available here: http://samate.nist.gov/index.php/Web_Application_Vulnerability_Scanners.html
Please note that reference to these tools does not imply that they have been tested and approved for use by DISA.

5.4.2 Manual Code Review

A manual code review is performed by one or more application reviewers. Manual reviews are more time consuming than automated reviews and are reliant upon the skill and experience of the

reviewer to find security flaws, but they have the advantage of being able to detect forms of vulnerability that might not be found through automated analysis.

5.4.3 Third-Party Code Review

Several commercial companies offer code review services. These companies use a mixture of automated reviews and manual reviews by experienced auditors. Third-party review is valuable if the application is highly exposed, is large, or cannot be manually reviewed in depth.

This page is intentionally blank.

6. DEPLOYMENT

This section covers general guidelines to be used when deploying an application. It may be used either alone or in addition to the guidance provided in the rest of this document. It consists of general guidelines and some specific requirements for operating in the DoD environment.

6.1 Documentation

6.1.1 System Security Plan (SSP)

The SSP will document all required IA roles and the specific personnel assigned to those roles. The SSP will document the assigned duties of the IA roles and the prerequisite training, security clearance, and any special requirements to perform the IA duties.

(APP2010.3: CAT II) The IAO will ensure all appointments to required IA roles are established in writing to include assigned duties and appointment criteria such as training, security clearance, and IT designation.

6.1.2 Classification Guide

The IAO will acquire the classification guide from the Program Manager and make it available to the application users.

(APP2040.2: CAT II) The IAO will ensure the classification guide for the application data exists and is available to users.

6.1.3 Application Configuration Guide

The IAO will configure the application according to the Application Configuration Guide provided by the Program Manager of the developed application. General purpose servers share similar resources across numerous applications. A general purpose web server may host many web sites on the same machine.

(APP6010: CAT II) The IAO will ensure if an application is designated critical, the application is not hosted on a general purpose machine.

(APP2020.6: CAT II) The IAO will ensure the application is deployed in a manner consistent with the Application Configuration Guide provided by the developers.

6.1.4 Threat Model

The countermeasures identified throughout this process will be implemented by the IAO according to the threat model.

(APP3020.4: CAT II) The IAO will ensure identified mitigations to identified threats are implemented.

6.2 Third-Party Software

If a DoD STIG or an NSA guide is not available, third-party products used in the application will be configured to commercially accepted practices, independent testing results, and/or vendor literature.

(APP6020: CAT II) The IAO shall ensure if a DoD STIG or NSA guide is not available, a third-party product will be configured by the following in descending order as available: (1) commercially accepted practices, (2) independent testing results, or (3) vendor literature.

6.3 Ports and Protocols

While an application may be written to utilize any ports and/or protocols securely, the DoD has established guidelines regarding the use of Ports, Protocols, and Services (PPS) traversing or connecting to the GiG. The application in a DoD environment will comply with the guidance provided in the DoDI 8551.1 and the companion PPS CAL as well as all associated mitigations, as defined in the appropriate Vulnerability Assessments found at the following web site: http://iase.disa.mil/ppsm/
All IPs, data services, and associated ports (whether used internal or external to the enclave) used throughout planned, newly developed, acquired, and existing DoD networks, which include DoD information systems, applications, and boundary protection devices, must be limited to essential capabilities, assessed for vulnerabilities, documented in a vulnerability assessment (VA) report, and assigned an assurance category and documented in the Category Assurance List (CAL).

(APP2100.2: CAT II) The IAO will ensure the application is configured to comply with the DoD Ports and Protocols guidance.

(APP2100.3: CAT II) The IAO will ensure mitigations have been applied from the vulnerability assessments for all ports used in the application.

The DISN Security Accreditation Working Group (DSAWG) may approve formal exceptions, in the form of waivers, to the PPS requirements; however, the vulnerability still exists and must be recorded in the certification and accreditation documentation noting the vulnerability is present but the risk has been accepted and a Plan of Action and Milestones (POA&M) is in place.

(APP2110.3: CAT II) The IAO will ensure the application and all associated PPS are registered with the DoD PPS database.

6.4 Workplace Security Procedures

The IAO is responsible for developing procedures that assure sensitive, classified, FOUO, and other data classification not included, meet physical data handling and storage requirements for the production deployment site.

These procedures should include:

- End-of-day security checks
- Unannounced security checks
- Imposition of a two-person rule within the computing facility (where appropriate)

(APP2150.2: CAT II) The IAO will ensure procedures are implemented to assure physical handling and storage of information is in accordance with the data's sensitivity.

6.5 Unnecessary Services

All unnecessary services installed by default in an application will be removed or disabled.

(APP6030: CAT II) The IAO will ensure unnecessary services are disabled or removed.

6.6 Application Maintenance

6.6.1 Vulnerability Management

If the application developers have provided an automated alert mechanism to inform users of the availability of security updates, at least one application administrator must be registered to receive these notices.

If a security update is released for the application, the update will be applied after it has been operationally tested. If the update does impair the operation of the application, then the application hosting providers will give the developers feedback to correct any operational deficiencies in the application. The developers will issue a new release of the application to the hosting providers as soon as an update is available.

Many COTS products have included open source products in the packaging of their own products. One example is the Apache web server which is bundled with the Oracle DBMS product. The COTS product vendor will send updates when required for both open source tools provided in their product and their own custom developed software.

(APP6040: CAT II) The IAO will ensure at least one application administrator has registered to receive update notifications or security alerts when automated alerts are available.

(APP6050: CAT II) The IAO will ensure the system and installed applications have current patches, security updates, and configuration settings.

6.6.2 Maintenance Availability

As long as the application is in use, maintenance must be available to address any security flaws discovered in the application. New software vulnerability types are discovered on a regular basis, and it is always possible an application previously having no known exploits will have a vulnerability discovered.

The maintenance does not need to have an active team devoted to the application, but the facilities necessary to address the flaw in a timely and efficient manner must be available. If

maintaining the application is no longer possible, users will be informed when maintenance will be discontinued so they may migrate to another solution. Software no longer under maintenance or security support will be decommissioned and removed.

(APP6060: CAT I) The IAO will ensure the application is decommissioned when maintenance or support is no longer available.

(APP6070: CAT III) The IAO will ensure provisions are in place to notify users when an application is decommissioned.

6.7 Security Incident Response Process

The incident response process should be put in place to respond to reports of security flaws in the application. This process should include a method for individuals to submit potential security vulnerabilities to the development team. The process should then dictate what is to be done with the reported vulnerabilities. Reported vulnerabilities must be tracked throughout the process to ensure they are triaged, corrected, and tested. The corresponding update should be released to the user community. Also, the user community should be notified of the availability of the application update.

(APP2140.2: CAT II) The IAO will ensure a security incident response process for the application is followed.

6.8 Denial of Service (DoS)

A DoS attack is an attempt to prevent legitimate users from accessing a resource. Attackers attempt to exhaust a particular resource pool of an application, thereby preventing it from functioning correctly. These resources include, but are not limited to: sessions, disk space, memory, process space, network connections, or bandwidth. DoS attacks may also target infrastructure components on which the application relies, such as routers or DNS servers.

DoS attacks include:

- Repeated attempts to login to a service or user account in order to lock the account disrupting the service or person
- Disrupting network traffic between server components creating unrecoverable timeout issues with the application
- Launching additional copies of the applications to consume memory, disk space, and/or fill the process tables
- Directly accessing individual application services to consume memory, disk space, and/or fill the process tables
- Generating excessive numbers of mail messages
- Generating errors that must be logged

The following steps can be taken to help prevent or reduce the impact of DoS attacks:

- Disable or remove unneeded services.

- Enable disk quota systems if available.
- Monitor the system to detect low resource situations.
- Limit the total number of concurrent sessions that can be opened by a single user.

(APP6080: CAT II) The IAO will ensure protections against DoS attacks are implemented.

(APP6090: CAT III) The IAO will ensure the system alerts an administrator when low resource conditions are encountered.

6.9 Access Control

The IAO will protect application configuration and executable files with the permission sets allowing only the application administrator to modify these files.

(APP3450.2: CAT II) The IAO will ensure application resources are protected with permission sets only allowing application administrator to modify these configurations and files.

(APP3450.3: CAT II) The IAO will ensure access to format strings used by the application are restricted to authorized users.

6.10 Database Exports

Production database exports are often used to populate development databases. Test and development environments do not typically have the same rigid security protections that production environments do. When production data is used in test and development, the production database exports will need to be scrubbed to prevent information like passwords and other sensitive data from becoming available to development and test personnel who may not have a need to know. Sensitive data should not be included in database exports because of classification, privacy, and other types of data protection requirement issues.

(APP6100: CAT II) The IAO will ensure production database exports have database administration credentials and sensitive data removed before releasing the export.

6.11 PKI Certificate Configuration

In a production environment, the application should be configured to accept only approved DoD PKI certificates. Invalid, expired, revoked, and test certificates should not be issued in a production environment.

(APP3290.2: CAT I) The IAO will ensure the PK-enabled applications are configured to honor only approved DoD PKI certificates.

Note: For server to server authentication where an administrator has responsibility for both server certificates stores, certificate status checking may be disabled.

6.12 Auditing

6.12.1 Audit Trail Monitoring

Audit trail records from all available sources are regularly reviewed for indications of inappropriate or unusual activity.

(APP6110: CAT III) The IAO will review audit trails periodically based on system documentation recommendations or immediately upon system security events.

Suspected violations of IA policies are analyzed and reported in accordance with DoD information system IA procedures.

(APP6120: CAT II) The IAO will report all suspected violations of IA policies in accordance with DoD information system IA procedures.

For classified systems, automated monitoring of audit trails is required.

(APP6130: CAT III) The IAO will ensure, for classified systems, application audit trails are continuously and automatically monitored, and alerts are provided immediately when unusual or inappropriate activity is detected.

6.12.2 Audit Log Retention

Audit logs should be retained for a minimum of 1 year for applications not containing SAMI data. If the application contains SAMI data, the audit logs need to be retained for no less than 5 years.

(APP6140: CAT II) The IAO will ensure application audit trails are retained for at least 1 year for applications without SAMI data, and 5 years for applications including SAMI data.

6.12.3 Audit Trail Protection

The contents of audit trails are protected against unauthorized access, modification, or deletion.

(APP3690.3: CAT II) The IAO will ensure the audit trail is readable only by application administrators and auditors.

(APP3690.4: CAT II) The IAO will ensure the audit trail is protected against modification or deletion except by application administrators and auditors.

6.13 Recovery and Contingency Planning

Recovery and contingency planning is an important aspect of the deployment phase of the application. Documented backup and recovery procedures are required in the event of a system failure. Any circumstances prohibiting system recovery should be documented and included as a

part of the recovery procedures. System recovery needs to be performed in a secure and verifiable manner to protect the application and data.

(APP6160.1: CAT II) The IAO will ensure recovery procedures and technical system features exist so recovery is performed in a secure and verifiable manner.

(APP6160.2: CAT II) The IAO will document circumstances inhibiting a trusted recovery.

Back-up copies of the application software are stored in a fire-rated container or otherwise not collocated with the operational software.

(APP6170: CAT II) The IAO will ensure back-up copies of the applications software are stored in a fire-rated container and not collocated with operational software.

(APP6180: CAT II) The IAO will ensure procedures are in place to assure the appropriate physical and technical protection of the backup and restoration of the application.

Backup and recovery procedures vary based on the application MAC level.

For MAC III systems:

(APP6190.1: CAT II) The IAO will ensure data backup is performed at least weekly.

For MAC II systems:

(APP6190.2: CAT II) The IAO will ensure data backup is performed daily and recovery media is stored off-site at a location.

For MAC I systems:

(APP6190.3: CAT II) The IAO will ensure data backup is accomplished by maintaining a redundant secondary system, not collocated, that can be activated without loss of data or disruption to the operation.

Disaster recovery plans include business recovery plans, system contingency plans, facility disaster recovery plans and plan acceptance. Disaster recovery procedures for all systems vary based on their MAC level.

For MAC I systems:

(APP6200.1: CAT II) The IAO shall ensure a disaster plan exists providing for the smooth transfer of all mission or business essential functions to an alternate site for the duration of an event with little or no loss of operational continuity.

For MAC II systems:

(APP6200.2: CAT II) The IAO shall ensure a disaster plan exists providing for the resumption of mission or business essential functions within 24 hours of activation.

For MAC III Systems:

(APP6200.3: CAT II) The IAO shall ensure a disaster plan exists providing for the partial resumption of mission or business essential functions within 5 days of activation.

6.14 Account Management

The IAO is responsible for the overall account management of a production application.

Individual user accounts must be suspended, disabled, or removed from the application when they are terminated or no longer are required to have access to the application.

(APP6210: CAT II) The IAO will ensure an account management process is implemented, verifying only authorized users can gain access to the application and individual accounts designated as inactive, suspended, or terminated are promptly removed.

Password policies must comply with organizational password policies and not generate predictable passwords.

(APP6220: CAT I) The IAO will ensure passwords generated for users are not predictable and comply with the organizations password policy.

Individual users must be granted their own user accounts and not be allowed to share individual application user accounts.

(APP6230: CAT II) The IAO will ensure the applications users do not use shared accounts.

All user accounts are disabled which are authorized to have access to the application but have not authenticated within the past 35 days.

(APP6240: CAT III) The IAO will ensure all user accounts are disabled which are authorized to have access to the application but have not authenticated within the past 35 days

(APP6250: CAT II) The IAO will ensure unnecessary built-in application accounts are disabled.

(APP6260: CAT I) The IAO will ensure default passwords are changed.

(APP3320.8: CAT II) The IAO will configure the application to ensure account passwords conform to DoD password policy.

(APP3470.2: CAT II) The IAO will ensure access to privileged accounts is limited to privileged users.

(APP3470.3: CAT II) The IAO will ensure non-privileged accounts are limited to non-privileged users.

(APP3470.4: CAT II) The IAO will ensure the application account is established and administered in accordance with a role-based access scheme to enforce least privilege and separation of duties.

(APP3480.3: CAT II) The IAO will ensure the access procedures enforce the principles of separation of duties and "least privilege".

6.15 Infrastructure Compliance

If a DoD STIG is available for one or more components of an application, its guidance will be applied. The IAO should enforce STIG compliance of all deployment systems.

(APP2160.3: CAT II) The IAO will ensure deployment systems and all components comply with all appropriate DoD STIGS, NSA guides, and all applicable DoD policies.

6.16 Enclave DMZ

If an application spans between DoD enclaves and the Internet or other public or commercial wide area, a DMZ is required. Refer to the Enclave STIG for specific DMZ requirements found at the following web site: https://www.us.army.mil/suite/page/397960.

(APP6270: CAT II) The IAO will ensure connections between the DoD enclave and the Internet or other public or commercial wide area networks require a DMZ.

6.17 DoD DMZ

The DoD DMZ will host all Internet-facing applications (i.e., application that must be available from the Internet). Internet-facing applications fall into two categories: Unrestricted (available to anyone without authentication) and Restricted (available from the Internet but authentication is required for access). These requirements only apply to Internet-facing applications and do not apply to SIPRNet or classified systems.

The DoD DMZ implements a tiered architecture with a web front-end tier and an application/database back-end tier. When designing Internet-facing applications, designers must develop the application to be deployed in this tiered architecture. When the application is designed in a tiered style, the deployer will need to configure the application to operate inside the tiered DoD DMZ. Web servers must be segmented from back-end databases and application servers. This segmentation may be accomplished by use of logical separation if the application contains similar data types.

(APP6280: CAT I) The IAO will ensure web servers are on separate network segments from the application and database servers if it is a tiered application.

Applications that contain different data types must operate on physically separate motherboards. Operating systems with restricted data types may co-exist with other operating systems with restricted data types. Also, operating systems with unrestricted data types may co-exist with other operating systems with unrestricted data types.

(APP6290: CAT I) The Designer and the IAO will ensure Internet-facing web applications are designed for and operated on physically or virtually separate servers in accordance with the web tier requirements of Increment 1/Phase 1 deployment of the DoD DMZ.

Refer to the DoD DMZ STIG for specific DMZ Increment 1/Phase 1 requirements found at the following web site:
http://iase.disa.mil/stigs/net_perimeter/enclave-dmzs/Pages/dmz-imp.aspx

6.18 Web Services

6.18.1 Web Service XML Firewall

An XML firewall is a specialized firewall used to provide security for XML messaging, such as web services. XML firewalls are types of XML appliances that are separated from internal computer systems and frequently reside in an organization's DMZ.

(APP6300: CAT II) The IAO will ensure an XML firewall is deployed to protect web services.

6.18.2 Universal Description, Discovery and Integration (UDDI)

UDDI is an open industry initiative, sponsored by OASIS. UDDI is used for defining web services in a repository using XML to describe web service interfaces. The XML schema used to define web services is WSDL. WSDL describes web service end-points and their messages. UDDI communicates via SOAP messages.

When accessing the UDDI repository, browsing and listing should provide read-only access to the repository.

(APP6310: CAT II) The IAO will ensure web service inquiries to UDDI provide read-only access to the registry to anonymous users.

If the WSDL-formatted UDDI registry entry contains sensitive information, only authenticated users may have access to the sensitive UDDI registry entry.

(APP6320: CAT II) The IAO will ensure if the UDDI registry contains sensitive information, read access to the UDDI registry is granted only to authenticated users.

UDDI is an open industry initiative, sponsored by OASIS. UDDI is used for defining web services in a repository using XML to describe services interfaces. The XML schema used to define web services is WSDL. WSDL describes web service end-points and their messages. UDDI communicates via SOAP messages.

UDDI registry entries must be signed to establish the identity of the publisher and ensure the integrity of the directory of web services.

(APP3830.2: CAT II) The IAO will ensure digital signatures exist on UDDI registry entries to verify the publisher.

Early versions of UDDI did not support the signing of registry entries. UDDI, Version 3 0 or newer, will be used to digitally sign web service registry entries.

(APP3840.2: CAT II) The IAO will ensure UDDI versions are used supporting digital signatures of registry entries.

UDDI registry entries should be protected from unauthorized modifications. Consequently, publishing registry entries is restricted to authenticated users only.

(APP3850.2: CAT II) The IAO will ensure UDDI publishing is restricted to authenticated users.

This page is intentionally blank.

APPENDIX A. REFERENCES

Government References

Department of Defense Directive 8500.01E, “Information Assurance”, http://www.dtic.mil/whs/directives/corres/pdf/850001_2014.pdf

Department of Defense Instruction 8500.2, “Information Assurance Implementation”, 6 February 2003 http://www.dtic.mil/whs/directives/corres/pdf/850002p.pdf

Department of Defense Directive 8510.01, Risk Management Framework (RMF) for DoD Information Technology (IT). Formerly “DoD Information Assurance Certification and Accreditation Process (DIACAP),”
http://www.dtic.mil/whs/directives/corres/pdf/851001_2014.pdf

Department of Defense Directive 8570.01-M, “Information Assurance Training, Certification, and Workforce Management” 19 December 2005
http://www.dtic.mil/whs/directives/corres/pdf/857001p.pdf

Department of Defense Instruction 8520.2, "Public Key Infrastructure and Public Key Enabling", http://www.dtic.mil/whs/directives/corres/pdf/852002p.pdf

Department of Defense Instruction 8551.1, “Ports, Protocols, and Services Management” 13
http://www.dtic.mil/whs/directives/corres/pdf/855101p.pdf

Department of Defense Directive 5000.01, “The Defense Acquisition”
http://www.dtic.mil/whs/directives/corres/pdf/500001p.pdf

Defense Acquisition Guidebook
https://dag.dau.mil/Pages/Default.aspx

Department of Defense Security Technical Implementation Guides
https://www.us.army.mil/suite/page/397960

Ports, Protocols, & Services Management Database https://pnp.cert.smil.mil.

DKO Ports, Protocols, & Services Management https://www.us.army.mil/suite/page/396114

NIST Cryptographic Module Validation Program (CMVP) and Cryptographic Algorithm Validation Program (CAVP) http://csrc.nist.gov/cryptval

FIPS PUB 140-2 Security Requirements for Cryptographic Issued 25 May 2001
http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

Annex C: Approved Random Number Generators for FIPS PUB 140-2, Security Requirements for Cryptographic Modules issued 18 October 2007
http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexc.pdf

NIST Configuration Management in Security related Software Engineering Processes http://csrc.nist.gov/nissc/1996/papers/NISSC96/paper035/scm_kk96.pdf

National Vulnerability Database http://nvd.nist.gov/

National Information Assurance Partnership and Common Criteria Evaluation and Validation Scheme (CCEVS) information http://www.niap-ccevs.org/

United States (CERT) Computer Emergency Response Team http://www.us-cert.gov/ and http://www.cert.org/

Other References

Howard, Michael & Lipner, Steve. (2006) *The Security Development Lifecycle*. Redmond, Washington: Microsoft Press

Howard, Michael & LeBlanc, David (2003) *Writing Secure Code; Practical Strategies and Techniques for Secure Application Coding in a Networked World (2nd Edition).* Redmond, Washington: Microsoft Press

Howard, M., LeBlanc, D & Viega, J. (2005) *19 Deadly Sins of Software Security; Programming Flaws and How to Fix Them.* Emeryville, California: McGraw-Hill

Swiderski, Frank & Snyder, Window (2004) *Threat Modeling.* Redmond, Washington: Microsoft Press

Whittaker, James & Thompson, Herbert (2004) *How to Break Software Security.* USA: Pearson Education, Inc.

Gallagher, T., Jeffries, B., & Landauer, L. (2006) *Hunting Security Bugs.* Redmond, Washington: Microsoft Press

Larry Osterman's Threat Modeling 17 January 2005 http://blogs.msdn.com/larryosterman/archive/2005/01/17/354588.aspx

List of Common Vulnerabilities and Exposures http://cve.mitre.org/

Open Web Application Security Project http://www.owasp.org/

Open Web Application Security Project Threat Risk Modeling http://www.owasp.org/index.php/Threat_Risk_Modeling

The Security Development Lifecycle Blog http://blogs.msdn.com/sdl/rss.xml

Michael Howard’s Microsoft Software Security Blog http://blogs.msdn.com/michael_howard/

Microsoft STRIDE methodology Threat modeling Tool
http://www.microsoft.com/downloads/details.aspx?FamilyID=62830f95-0e61-4f87-88a6-e7c663444ac1&displaylang=en

Security Development Lifecycle (SDL) Banned Function Calls http://msdn2.microsoft.com/en-us/library/bb288454.aspx

SANS (SysAdmin, Audit, Network, Security) Institute Training http://www.sans.org/training/

This page is intentionally blank.

APPENDIX B. UNSAFE FUNCTIONS

Appendix B identifies functions that have a greater potential to cause application vulnerabilities. The presence of these functions does not immediately indicate a vulnerability; however, if they are used inappropriately, a vulnerability may exist. Table B-1 provides the unsafe functions in C and C++.

Table B-1. C and C++ Unsafe Functions

C and C++ Functions to Avoid		
Function	**Reason**	**Potential Replacements**
strcpy	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
wcscpy	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
_tcscpy	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
_mbscpy	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
StrCpy	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
StrCpyA	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
StrCpyW	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
lstrcpy	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
lstrcpyA	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
lstrcpyW	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
strcpyA	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
strcpyW	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
_tccpy	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
_mbccpy	Potential for buffer overflows	strcpy_s, StringCchCopy, StringCbCopy,strlcpy
strcat	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
wcscat	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
_tcscat	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
_mbscat	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat

C and C++ Functions to Avoid		
Function	**Reason**	**Potential Replacements**
strCat	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
strCatA	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
strCatW	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
lstrcat	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
lstrcatA	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
lstrcatW	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
strCatBuffW	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
strCatBuff	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
strCatBuffA	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
strCatChainW	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
strcatA	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
strcatW	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
_tccat	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
_mbccat	Potential for buffer overflows	strcat_s, StringCchCat, StringCbCat, strlcat
sprintf	Potential format string vulnerabilities	
wsprintf	Potential format string vulnerabilities	
fprintf	Potential format string vulnerabilities	

APPENDIX C. GLOSSARY

Application. Software program that performs a specific function directly for a user and can be executed without access to system control, monitoring, or administrative privileges. Examples include office automation, e-mail, web services, and major functional or mission software programs.

Automated Information System (AIS) Application. For DoD IA purposes, an AIS application is the product or deliverable of an acquisition program. An acquisition program is defined in DoDD 5000.1. An AIS application performs clearly defined functions for which there are readily identifiable security considerations and needs that are addressed as part of the acquisition. An AIS application may be a single software application (e.g., Integrated Consumable Items Support (ICIS)); multiple software applications that are related to a single mission (e.g., payroll or personnel); or a combination of software and hardware performing a specific support function across a range of missions (e.g., Global Command and Control System (GCCS), Defense Messaging System (DMS)). AIS applications are deployed to enclaves for operation and have their operational security needs assumed by the enclave. (**Note**: An AIS application is analogous to a "major application," as defined in OMB A-130; however, this term is not used in order to avoid confusion with the DoD acquisition category of Major Automated Information System (MAIS).)

Certificate. A digital representation of information that, at a minimum, identifies the certification authority issuing it, names or identifies its subscriber, contains the subscriber's public key, identifies its operational period, and is digitally signed by the certification authority issuing it.

Common Access Card (CAC). A department-wide smart card used as the identification card for active duty Uniformed Services personnel (to include the Selected Reserve), DoD civilian employees, eligible contractor personnel, and eligible foreign nationals. The primary platform for the public key infrastructure (PKI) authentication token used to access DoD computer networks and systems in the unclassified environment and, where authorized by governing security directives, the classified environment; and the principal card enabling physical access to buildings, facilities, installations, and controlled spaces.

Demilitarized Zone (DMZ). Perimeter network that adds an extra layer of protection between internal and external networks by enforcing the internal network's IA policy for external information exchange. A DMZ, also called a "screened subnet," provides external, untrusted sources with restricted access to releasable information while shielding the internal networks from outside attacks.

DoD-Approved External PKI. A PKI approved by the ASD (NII)/DoD CIO for use by DoD relying parties for assurance levels appropriate for the information being protected.

DoD Eligible Users. DoD eligible users are active duty Uniformed Services personnel, members of the Selected Reserve, DoD civilian employees, and personnel working on site at DoD facilities using DoD network and e-mail services.

DoD Information System. Set of information resources organized for the collection, storage, processing, maintenance, use, sharing, dissemination, disposition, display, or transmission of information. DoD Information System includes AIS applications, enclaves, outsourced IT-based processes, and platform IT interconnections.

DoD Partners. DoD partners are Government or non-Government entities that process electronic transactions with the DoD, or exchange e-mail containing DoD sensitive information.

DoD Private Web Server. For unclassified networks, a DoD private web server is any DoD-owned, operated, or controlled web server providing access to sensitive information that has not been reviewed and approved for release in accordance with DoD Directive 5230.9 and DoD Instruction 5230.29. For SIPRNet and other classified networks that are not accessible to the public, a DoD private web server is any server that provides access to information that requires need-to-know control or compartmentation.

Enclave. Collection of computing environments connected by one or more internal networks under the control of a single authority and security policy, including personnel and physical security. Enclaves always assume the highest mission assurance category and security classification of the AIS applications or outsourced IT-based processes they support and derive their security needs from those systems. They provide standard IA capabilities, such as boundary defense, incident detection and response, and key management, and also deliver common applications, such as office automation and e-mail. Enclaves are analogous to general support systems as defined in OMB A-130. Enclaves may be specific to an organization or a mission and the computing environments may be organized by physical proximity or by function independent of location. Examples of enclaves include local area networks and the applications they host, backbone networks, and data processing centers.

Enclave Boundary. The point at which an enclave's internal network service layer connects to an external network's service layer.

Government off-the-shelf (GOTS). software and hardware products developed and tailored specifically for government agencies by a government agency itself or in cooperation with a external vendor or contractor. These products are developed for redistribution to other government agencies. Custom developed code being developed by an external source, does not automatically qualify as GOTS.

Hardware Token. A portable, user-controlled, physical device used to generate, store, and protect cryptographic information and to perform cryptographic functions.

IA Manager (IAM). The individual responsible for the IA program of a DoD information system or organization. While the term IAM is favored within the Department of Defense, it may be used interchangeably with the IA title Information Systems Security Manager (ISSM).

IA Officer (IAO). An individual responsible to the IAM for ensuring that the appropriate operational IA posture is maintained for a DoD information system or organization. While the term IAO is favored within the DoD, it may be used interchangeably with other IA titles (e.g.,

Information Systems Security Officer, Information Systems Security Custodian, Network Security Officer, or Terminal Area Security Officer).

IA Product. Product or technology whose primary purpose is to provide security services (e.g., confidentiality, authentication, integrity, access control or non-repudiation of data); correct known vulnerabilities; and/or provide layered defense against various categories of non-authorized or malicious penetrations of information systems or networks. Examples include such products as data/network encryptors, firewalls, and intrusion detection devices.

IA-Enabled Product. Product or technology whose primary role is not security, but which provides security services as an associated feature of its intended operating capabilities. Examples include such products as security-enabled web browsers, screening routers, trusted operating systems, and security-enabled messaging systems.

Information Owner. Official with statutory or operational authority for specified information and responsibility for establishing the controls for its generation, collection, processing, dissemination, and disposal.

Information-Privileged Individual. For purposes of this Instruction, an information-privileged individual is a person whom the DoD has authorized access to specified DoD information systems to provide them information about, or access to, benefits, entitlements or services, which may be available to them. The information may include information protected by the Privacy Act or Health Insurance Portability and Accountability Act (HIPAA), which may be lawfully displayed to them. Information-privileged individuals include retirees and dependents. These individuals are provided access to these systems to facilitate the delivery of benefits, entitlements, and services. Information-privileged individuals will not be provided access to other DoD sensitive systems, unless they are otherwise authorized such access.

Mobile Code. Software modules obtained from remote systems, transferred across a network, and then downloaded and executed on local systems without explicit installation or execution by the recipient.

National Information Assurance Partnership (NIAP). Joint initiative between NSA and NIST responsible for security testing needs of both IT consumers and producers and promoting the development of technically sound security requirements for IT products and systems and appropriate measures for evaluating those products and systems.

Need-to-Know. Necessity for access to, or knowledge or possession of, specific official DoD information required to carry out official duties.

Need-to-Know Determination. Decision made by an authorized holder of official information that a prospective recipient requires access to specific official information to carry out official duties.

Public Key Infrastructure (PKI). The framework and services that provide for the generation, production, distribution, control, accounting and destruction of public key certificates.

Sensitive Information. Information, the loss, misuse, or unauthorized access to or modification of, could adversely affect the national interest or the conduct of Federal programs, or the privacy to which individuals are entitled under Section 552a of title 5, United States Code, "The Privacy Act", but which has not been specifically authorized under criteria established by Executive order or an Act of Congress to be kept secret in the interest of national defense or foreign policy (Section 278g-3 of title 15, United States Code, "The Computer Security Act of 1987"). Examples of sensitive information include, but are not limited to, information in DoD payroll, finance, logistics, and personnel management systems.

Sensitive information sub-categories include, but are not limited to, the following:

For Official Use Only (FOUO). In accordance with DoD 5400.7-R, DoD information exempted from mandatory public disclosure under the Freedom of Information Act (FOIA).

Privacy Data. Any record that is contained in a system of records as defined in the Privacy Act of 1974 (5 U.S.C. 552a) and information the disclosure of which would constitute an unwarranted invasion of personal privacy.

DoD Unclassified Controlled Nuclear Information (DoD UCNI). Unclassified Information on security measures (including security plans, procedures, and equipment) for the physical protection of DoD Special Nuclear Material (SNM), equipment, or facilities in accordance with DoD Directive 5210.83. Information is Designated DoD UCNI only when it is determined that its unauthorized disclosure could reasonably be expected to have a significant adverse effect on the health and safety of the public or the common defense and security by increasing significantly the likelihood of the illegal production of nuclear weapons or the theft, diversion, or sabotage of DoD SNM, equipment, or facilities.

Unclassified Technical Data. Data that is not classified but is subject to export control and is withheld from public disclosure according to DoD Directive 5230.25.

Proprietary Information. Information that is provided by a source or sources under the condition that it not be released to other sources.

Foreign Government Information. Information that originated from a foreign government and that is not classified CONFIDENTIAL or higher but must be protected in accordance with DoD 5200.1-R.

Department of State Sensitive But Unclassified (DoS SBU). Information that originated from the Department of State that has been determined to be SBU under appropriate Department of State information security polices.

Drug Enforcement Administration (DEA) Sensitive Information. Information that is originated by the Drug Enforcement Administration and requires protection against

unauthorized disclosure to protect sources and methods of investigative activity, evidence, and the integrity of pretrial investigative reports.

Sources and Methods Intelligence (SAMI). Any classified non-SCI information that has been determined by the data or Information Owner to need the protection afforded by DCID 6/5 and bears a SAMI marking.

Web Server. An automated information system that manages a web site by passing web pages to web browsers over a network. The web server may provide information stored locally on the server or may act as a portal to access information from other linked information systems.

This page is intentionally blank.