EUROPEAN COMMISSION
DIRECTORATE-GENERAL
HUMAN RESOURCES AND SECURITY
Directorate DS - Security
Informatics Security

Brussels, 04/07/2011
HR.DS5/GV/ac ARES (2011) 719444
SEC20.10.05/04 - Standards

European Commission

Information System Security Policy

C(2006) 3602

STANDARD ON SECURE SYSTEMS DEVELOPMENT

ADOPTED BY MRS. IRENE SOUKA,

DIRECTOR-GENERAL OF DG HUMAN RESOURCES AND SECURITY, ON 04/07/2011

Version 0.4_04/05/2011

Commission européenne, B-1049 Bruxelles / Europese Commissie, B-1049 Brussel - Belgium. Telephone: (32-2) 299 11 11.

Commission européenne, L-2920 Luxembourg. Telephone: (352) 43 01-1.

TABLE OF CONTENTS

1. ADOPTION PROCEDURE

This Security Standard is adopted in accordance with Article 10(3) of Commission Decision C(2006) 3602 concerning the security of information systems used by the European Commission, adopted on 16 August 2006.

It is drawn up under the responsibility of the Security Directorate pursuant to Article 9(1)(b) and takes into account the items listed in Article 10(2) of Commission Decision C(2006)3602, in particular internationally recognised norms and standards applicable in the field of information systems security.

Under Article 10(3) of Commission Decision C(2006) 3602, the implementing rules may be supplemented by measures of a technical, physical, procedural or organisational nature proposed by the Director of the Security Directorate and adopted by the Director-General of the Directorate-General for Human Resources and Security in consultation with departments that have a legitimate interest. These supplementary measures are called 'security standards' where their application is mandatory, or 'security guidelines' where their application is optional or where they provide guidance on security standards implementation.

2. INTRODUCTION

Much of the European Commission's activity is supported by information technology, and software applications are constantly in the process of development or acquisition to support the evolving needs. The confidentiality, integrity and availability of these applications and their data are always important, and the increasing use of applications whose user community extends beyond the Commission's boundaries means that the risks relating to these applications are also growing.

Software applications must be secured in order to protect them from accident or abuse. The necessary controls must be built into information systems, and the earlier that this is done, the better, since adding them later on is more costly and likely to leave vulnerabilities if the applications have not been designed with security in mind.

3. OBJECTIVES

This standard provides instructions for the processes relating to the acquisition, development and maintenance of information systems. The objective is to ensure that appropriate security controls are identified and included in such systems, and that risks such as the introduction of unauthorised code are minimised. The standard also contains specific controls for outsourced software development.

4. SCOPE

This standard applies to all information systems that are acquired or developed by or on behalf of the European Commission. The measures mandated by this standard must be followed by all relevant personnel, including all Commission officials, contractors and third parties involved in this process.

These standards are minimum requirements for all systems. Additional requirements may be applied for systems handling EUCI, as mandated by Commission Decision 2001/844/EC, ECSC, Euratom.

5. THREATS COVERED

Security controls defined in this information security standard will help to reduce the impact of the following threats (their description is in the *Standard on Information Security Risk Management*):

T31 – Software malfunction

T36 – Corruption of data

T38 – Error in use

T39 – Abuse of rights

T40 – Forging of rights

T41 – Denial of actions

6. TERMINOLOGY

Accreditation: A formal declaration by a Security Accreditation Authority according to Commission Decision 2001/844/EC, ECSC, Euratom that an IS is approved for operation at an acceptable level of risk, based on the implementation of an approved set of technical, managerial, and procedural safeguards.

Buffer overflow: A condition where a computer storage field's limits are exceeded, causing data to overflow into adjacent memory. This can cause unexpected results such as application crashes or execution of malware.

Certification: A comprehensive assessment of the management, operational, and technical security controls in an information system, made in support of security accreditation, to determine the extent to which the controls are implemented correctly, operating as intended, and producing the desired outcome with respect to meeting the security requirements for the system. Certification is normally performed by an external body and results in a formal certificate of compliance.

Check digit: A number (usually one or two digits long) used to validate a numerical field, such as a bank account number or national identity number. Check digits are the result of calculations involving the digits in the numerical field.

Checksum: A calculated number used to verify a set of numbers.

CIA: Confidentiality, Integrity and Availability – the three major aspects of Information Security. EC systems must be classified for each of these three aspects (see Commission Decision C(2006) 3602).

Common Criteria: The Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408) constitute a framework for specifying functional and assurance requirements.

Cross-site scripting: A web application attack method whereby an attacker can execute scripts in the victim's browser which can hijack user sessions, deface web sites or redirect the user to malicious sites.

Digital Signature: Normally, a digest or hash of a message encrypted with a private key to assure message integrity and non-repudiation.

Evaluation Assurance Level (EAL): A numerical grade (from 1 to 7) assigned following the completion of a Common Criteria security evaluation. Higher EALs indicate greater levels of formality in the design and testing processes.

Information System (IS): A set of software that handles information (or data). Other components may also be included in the overall scope of an IS, including hardware, other software (operating systems, middleware etc.), networks, personnel, administrative processes and external entities. See section 7 for further information.

Input data: Information that is entered into an application. This is normally either information entered directly into the application by a user, or data coming from another application.

Input data validation: The process of checking that input data appears to be correct (within defined parameters) and does not contain malicious input (such as code fragments trying to bypass application security).

Message Authentication Code: A specific type of (symmetric) cryptographic hash function used to provide assurance of message integrity.

Message digest: A summary version of a message that is often encrypted or hashed and used to verify the integrity of the received message.

SOA: "Service Oriented Architecture", in which Information System(s) is/are realised in sets of independent and interoperable services. XML (eXtensible Markup Language) is commonly used for interfacing with such services.

SQL Injection: A method of attacking applications by entering database commands in data entry fields in order to bypass security controls or cause application errors. This is one of the most common ways of attacking applications.

Systems Development Life Cycle (SDLC): the process of creating or altering information systems, and the models and methodologies that people use to develop these systems. Many different SDLCs have been developed with different emphases and phases; this standard uses a basic model consisting of five phases: Initiation, Development, Implementation, Operation and Disposal.

Target System: An Information System, including all relevant components, that is the subject of a security policy, plan or review.

7. BACKGROUND INFORMATION

Information systems are essential to many of the administrative processes of the European Commission. The vast majority of the information handled by the Commission is computerised, and the quantity of this information and the complexity of the systems that handle it are ever increasing. Clearly, it is necessary

to protect these systems and the information that they handle, and studies have shown that this is best achieved as early as possible in the lifetime of a system. The goal of this standard is to define when and how this should be done, for both new and existing systems.

It is important to correctly understand the scope of this standard so that its rules are applied whenever they are relevant. One difficulty in doing this is the definition of what is meant by an Information System (IS) for the purposes of this standard. The Commission Decision C(2006) 3602 upon which this standard is based defines an information system as:

"A set of equipment, methods and procedures, and where relevant also persons, personnel, organised to perform information processing functions[1]*."*

In the context of the current standard, it may be helpful to expand upon this description. An Information System is fundamentally a set of software that in some way handles information (or data). Around this core of software and data may be found other components, including hardware, other software (operating systems, middleware etc.), networks, personnel, administrative processes and external entities (normally as providers or users). A more precise definition is difficult since "information system" is a rather abstract concept.

It must also be noted that the definition of Information Systems is intended to be inclusive rather than exclusive, encompassing all types of software (application software, middleware, operating systems etc.). As a general principle, all computer hardware[2] should be part of an identified Information System, and all ISs must have a System Owner[3] and be subject to the rules of this standard.

The current standard covers the process of developing, acquiring and maintaining information systems, and is intended to ensure that a well managed process is in place to assure that all ISs contain adequate controls to protect their confidentiality, integrity and availability. The security considerations for each system are documented in a Security Plan.

The software components of ISs may be acquired in two principal ways: by acquiring existing third party products (whether commercial or open-source), or by developing bespoke software; this standard contains rules for both situations. There are also combinations of the two, such as external software development or products requiring extensive customisation (beyond simple configuration), and in these cases the rules applying to both situations must be applied as relevant.

---

[1] C(2006) 3602, Article 3, paragraph 3

[2] The precise definition of what constitutes computer hardware is also difficult, particularly at the increasingly blurred boundaries between computers and other devices or storage media. For example, a PDA is clearly a computer system, but what about a mobile telephone? A fixed telephone may not be a computer system on its own, but may be part of a larger IS, which includes the exchange and all telephones connected to it.

[3] As mandated by the Standard on Asset Management

The scope of the security requirements should be defined by determining the boundaries of the IS; the system so described is often referred to as the "security target", or the "target system". It is important to define the target system so that it is clear where controls need to be applied, particularly those that are intended to protect the system from external threats. The definition of the target system – and its perimeter – may include different types of elements, such as geographic location, network segments, hardware or organisational boundaries.

This standard also contains rules on technical aspects of secure systems development. However, detailed information on the types of application attacks and vulnerabilities is outside the scope of this standard, since this information is constantly changing as new issues are identified. Extensive information is available from reputable sources, such as the OWASP[4] which publishes information on the most common application security risks. The Commission follows its own development methodology for internally developed systems, with which this standard is aligned[5].

The requirements of an information system are often described in a series of "business rules" describing its functionality and use. Some of these business rules should address the security requirements, and should be expressed in the security requirements, such as the validation, processing and integrity rules that are discussed in this standard.

8. SECURITY REQUIREMENTS OF INFORMATION SYSTEMS

Policy objective 7.1.1 – Security requirements of information systems – System owners must ensure that their security requirements are properly implemented. Therefore, all security requirements must be identified at the requirements phase of a project and justified, agreed and documented as part of the project documentation.

8.1. General rules

8.1.1. Information systems security governance

Any information system that is receiving, processing or outputting data is at risk of violating Confidentiality, Integrity, and Availability constraints. The design and implementation of the application must ensure that the risks of breaches of Confidentiality, Availability and Integrity are reduced to an acceptable level.

All information systems must have a System Owner (SO)[6], whose responsibilities include system security (see the *Standard on Asset Management*). The System Owner may appoint a System Security Officer (SSO) who is then responsible for the day-to-day security issues, although

[4] OWASP specifically addresses web applications (see http://www.owasp.org/)

[5] The "RUP@EC" methodology is currently used, and that methodology is being updated to ensure its compatibility with this standard.

[6] See Annex II of C(2006) 3602 for the definition of the roles mentioned in this section.

the SO retains overall responsibility. The LISO should also be involved on a consultative basis.

Some systems may also have a Data Owner separately from the SO. In this case, the Data Owner should also take part in the definition of the security requirements as regards the data for which he or she is responsible.

Other people or entities such as project managers, IRMs and system suppliers may also be involved in the definition, development or provision of security measures. Their roles are outlined in section 13 below.

8.1.2. Defining security requirements

Formally defined and documented business requirements must exist for each information system. The system must also be classified in accordance with the *Standard on Asset Management*. The security requirements must be documented in a Security Plan[7].

System requirements for information security and processes for implementing security must be integrated in the early stages of information system acquisition or development projects.

A minimum list of issues to be addressed at this stage includes:

- Access controls (mandatory or discretionary) (see the *Standard on Access Control and Authentication*)
- Administrator controls (see the *Standard on Access Control and Authentication*)
- Business continuity (see the *Standard on Business Continuity Management*)
- Data classification (see the *Standard on Asset Management*)
- Data encryption (see the *Standard on Cryptography and Public Key Infrastructure*)
- Data input controls (see section 9.2)
- Data output validation and access (see section 9.5)
- Data storage reliability and recovery (see the *Standard on Back-up*)
- Network security (see the *Standard on Network Security*)
- Operating system configuration

[7] See section 3.4 of the "Implementing Rules for Commission Decision C(2006) 3602 of 16.8.2006", and the related "Guidelines on Security Plan".

- Physical security of system devices (see the *Standard on Physical and Environmental Security*)
- Protective markings (see "C(2006) 3602 of 6.8.2006" for data classified as LIMITED, and Commission Decision 2001/844/EC, ECSC, Euratom of 29.11.2001 for EUCI).
- Segregation of duties (see the *Standard on Access Control and Authentication*)
- System level access (see the *Standard on Access Control and Authentication*)
- System-to-system identification (see the *Standard on Application Security*)
- User access methods (i.e. how users access data; whether it is directly, using application software, via a database interface etc.)
- User identification (see the *Standard on Access Control and Authentication*)
- User accountability (see the *Standard on Logging and Monitoring*), especially when dealing with sensitive data

See the standards mentioned for further details on many of these topics.

8.2. Developed Systems

8.2.1. Guidance on applicability

Where systems or applications are developed by or on behalf of the Commission, security requirements must be included at appropriate phases during the development as described below. This section does not provide a full description of the development phases, but only the key security-related issues that must be addressed at each stage.

For the purposes of this document, the rules will be based around a standard Systems Development Life Cycle (SDLC) consisting of five phases: Initiation, Development, Implementation, Operation and Disposal. A formal software development methodology must be used for developing systems, although it does not have to adhere to the model described in this section. Where a different methodology is used, these rules must be mapped onto that methodology and applied accordingly.

At the time of writing, the RUP methodology is the standard software development methodology used by the Commission. The following table shows the high-level correspondence between RUP and the SDLC used here.

SDLC phase	RUP phase
Initiation	Inception

Development	Elaboration Construction
Implementation	Transition
Operation	(not covered)
Disposal	(not covered)

The security requirements defined and implemented in these phases must make use of the general rules proscribed in section 8.1 above.

The Security Plan is the principal document that must be written for each system. It must be updated during the development process as relevant (see the *Guidelines on Security Plan*).

8.2.2. Initiation phase

The first phase covers the initial steps in a development project, up to the point where the development is approved. This typically covers steps such as a feasibility study, a high-level definition of business requirements and an estimate of the scope and budget.

During this phase, the following steps must be taken:

- The data to be handled by the system must be analysed to establish its level of sensitivity and classification for confidentiality, integrity and availability[8]
- Personal data must be identified and evaluated for privacy requirements in accordance with Regulation (EC) No 45/2001.
- The system's security level must be established as STANDARD or SPECIFIC[9]
- The geographical and organisational scope, and the network topology of the system must be reviewed to identify areas where data may pass through different security zones and therefore require additional protection such as encryption (e.g. data passing over public networks)
- High-level security requirements must be produced, based on an initial threat evaluation (when performing business impact analysis and starting the security plan)

[8] As described in Annex I of the Commission Decision C(2006) 3602 and the Standard on Asset Management

[9] Commission Decision C(2006) 3602, Annex I

- If the system's security level is classified as SPECIFIC, a risk assessment must be performed (see the *Standard on Information Security Risk Management*) and subsequently a Security Plan must be written, describing any additional security measures to be implemented

This step (or parts of it) may take place before a decision is made whether to develop the system or acquire an existing third party product. The result of this decision does not materially affect this phase from a security perspective, since the security requirements should serve either as a basis for the system design or as criteria for the selection of a third party product.

8.2.3. Development Phase

This phase covers the detailed design, development and testing of the system. Although the development phase normally focuses on the software being written, the accompanying processes and other measures should be included while considering the security of the administrative processes served by the system, particularly when designing non-automated security measures.

When software is developed by or on behalf of the Commission, source code must always remain available. For specific rules on outsourced software development, see section 12 below.

During this phase, the steps in the subsections below must be performed:

Detailed design

During this sub-phase, the requirements documented in the Initiation phase are expanded into a detailed description of the system to be developed (taking into account any subsequent modifications to the initial specifications).

- The security requirements from the Initiation phase must be included in the detailed system design.

- Security requirements that are not explicitly documented in the Initiation phase must also be included to ensure that the system will meet the generic requirements appropriate to the information classifications (for CIA) and the system security level.

Coding

In this sub-phase the program code is written and tested locally (unit or program testing). A certain amount of peer review and testing is performed to ensure that the quality of the code is sufficient for it to pass into the formal testing phase.

- Secure software development techniques must be followed during the development (see section 9)

- Coding standards appropriate to the development tools and techniques being used must be applied (see 0)

- Code reviews must be performed (see section 11.2)

- If system accounts are used, they must be designed according to the relevant rules (see section 10 of the Standard on Access Control and Authentication)

Testing

Testing must be performed on all software developments to ensure that they meet the requirements, operate correctly and securely, and can cope with expected volumes.

The subject of software testing is complex and there are many types of tests, such as unit tests (normally performed during the preceding sub-phase), system tests, volume tests, integration tests and acceptance tests; moreover, the nomenclature differs between different development methodologies.

Normally, separate System, Volume and Acceptance tests are required to meet all of the different testing needs. In addition, specific test procedures such as vulnerability assessments or penetration tests can provide greater assurance on system security. The system owner should determine what constitutes adequate testing for each system. See section 10 for more information on test environments.

In relation to security, standard testing should focus at a minimum on the issues below. The levels of protection applied must be appropriate to the information classification in each area.

Confidentiality – to ensure that information is protected against accidental or deliberate disclosure during processing, storage and transmission

Integrity – to ensure that information is correct and cannot be accidentally or deliberately corrupted during processing, storage and transmission

Availability – to ensure that the system will be available during the specified service window, that it can cope with the expected peak volumes and is resistant to attacks such as denials of service

Specifically, the following rules must be followed during testing:

- Test plans must be created, which describe the tests to be performed and the expected results.

- The results of tests must be documented

- Suitable test data must be designed and protected according to their classification (see section 10.1)

- The security requirements developed in the "Detailed Design" step must be specifically tested to ensure that they have been correctly developed

- Testing must provide assurance that the data classifications and system security level established during the Initiation phase are adequately supported by the system as developed

- Penetration tests must be performed for systems with a security level of SPECIFIC (they are recommended but not mandatory for STANDARD systems) – see section 11.3 for rules on performing penetration tests

- All software bugs or other issues identified during system, penetration, and acceptance testing must be recorded and tracked to resolution (correction or acceptance)

8.2.4. Deployment Phase

In this phase, the system is implemented into the live environment and tested to ensure that it works as expected in production. This phase often also includes the first few weeks or months of operation, when the development team is still available to troubleshoot errors and fix bugs.

- During implementation, care must be taken to ensure that the system and its associated security procedures are implemented as designed. In the event of any change, the security measures must be reviewed and updated if necessary.

- The system must be signed off as ready for use by the system owner or an approved delegate. The system acceptance process for this approval must include a review of the security measures.

8.2.5. Operation Phase

During this phase, the system is in regular use and the security measures must be operated as determined beforehand. In addition, most developed applications undergo regular changes or upgrades, and so the following steps must be taken:

- Security measures must be operated and regularly checked that they are working correctly

- Security requirements must be regularly reviewed and updated where necessary (e.g. in case the security target changes)

- Security requirements must be analysed for all application changes (see section 11.1 below on change control procedures)

- A process for identifying and addressing technical vulnerabilities must be in place (see the *Standard on Technical Vulnerability Management*)

- Systems must be subjected to periodic security reviews, depending on their sensitivity (see the Standard on Compliance).

8.2.6. Disposal Phase

When a system or a component reaches the end of its lifetime, care must be taken to ensure that any sensitive information is securely removed before the physical disposal of equipment. See the *Standard on Sanitisation of Media* for instructions on how to do this.

In some cases, information may need to be securely archived after a system is withdrawn from use. In this case, the security requirements for the archive must be analysed and agreed in the same way as for any other information system.

8.3. Third party Products

Third party products[10] acquired by the Commission should follow the same process as described in section 8.2 above, except that the Development phase (section 8.2.3) is replaced by a formal Acquisition phase.

Depending on the security requirements, consideration should be given to whether security certification or accreditation is required for such products, and at what level (e.g. Common Criteria EAL certifications).

During the Acquisition phase, the following steps must be performed:

- The security requirements from the Initiation phase must be reviewed and included in the formal system requirements (included in the Call for Tender or similar documentation sent to potential suppliers)

- Security requirements must be evaluated in the technical criteria, and should be given sufficient weight to have a significant impact on the product selection. In some cases (as determined by the SO in consultation with the LISO), certain security requirements may be essential so that their non-fulfilment mandates the rejection of a tender.

- If the system is classified as SPECIFIC, a risk assessment must be performed (see the *Standard on Information Security Risk Management*)

- The supplier should provide information about their testing procedures in order to provide assurance of the system's proper functioning and resistance to errors and attacks. If this is not possible (as may be the case, for example, for Open Source

[10] Often referred to as COTS (Commercial Off The Shelf) products. This category is intended to cover all products not made in-house, including software downloaded from third parties (e.g. via the Internet) and open source products that may be acquired for free, and therefore possibly without following a procurement procedure.

collaboration projects), then the above information should be collected through other means.

- The contract for the system(s) and/or services provided must include all relevant security specifications. If the software is acquired without a contract, then the security features should be thoroughly checked against the security requirements already documented.

- Where the security functionality in a proposed product does not satisfy the specified requirement then the risk introduced and potential compensating controls should be considered prior to acquiring the product.

- Where additional functionality is supplied and causes a security risk, this should be disabled or the proposed control structure should be reviewed to determine if advantage can be taken of the enhanced functionality available (without introducing new vulnerabilities).

The acquisition phase may focus primarily on the system being acquired, but the accompanying processes and other measures should not be neglected since they may need to be modified according to the capabilities of the system selected.

9. CORRECT PROCESSING IN APPLICATIONS

Policy objective 7.2.1 – Input Data Validation – Data input to applications must be validated to ensure that this data is correct and appropriate.

Policy objective 7.2.2 – Control of Internal Processing – Validation checks must be incorporated into applications to detect any corruption of information due to processing errors or deliberate acts.

Policy objective 7.2.3 – Message Integrity – Requirements for ensuring authenticity and protecting integrity of messages exchanged between applications must be identified, and appropriate controls identified and implemented.

Policy objective 7.2.4 – Output Data Validation – Data output from an application must be validated to ensure that the processing of information is correct and appropriate to the circumstances.

In order to best guarantee the correct processing of data in applications, the examination and validation of data should be automated as much as possible. Where manual checks are performed, clear responsibilities of all personnel involved in the process must be defined and communicated to the relevant personnel, and activity logs of the checks must be maintained. The rules in this section are intended to cover these goals.

9.1. Coding Standards

Coding standards must be documented for all coding technologies used in EC development projects. These coding standards must be followed by all

developers writing or reviewing code. Training may be required to ensure that all relevant personnel are sufficiently aware of the coding standards.

The coding standards should cover the issues described in the rest of this section below[11].

9.2. Input Data Validation

All data entered into Commission information systems, either manually or automatically, must be validated, when appropriate, with input checks such as boundary checking or limiting input fields to specific ranges or values in order to prevent or detect problems such as:

- values outside normal ranges
- invalid characters in data fields
- missing or incomplete data
- exceeding upper and lower data volume limits
- unauthorised or inconsistent control data
- attacks using malformed input, such as buffer overflows or SQL injection

Free text or numerical input fields are to be replaced by enumerated lists (whitelists) when possible. Applications must incorporate validation checks in business rules, to avoid errors or inconsistencies in multiple routines. Transactions which fail such checks must either be (a) rejected with a notification of the rejection sent to the submitter, (b) corrected and resubmitted, or (c) suspended pending further investigation.

In distributed architectures, data validation should be performed at the server level to protect against malicious data coming from compromised clients. Client-level checking may be implemented in addition to provide defence in depth[12].

9.3. Control of Internal Processing

Data that has been correctly entered can be corrupted by processing errors or through deliberate acts. The validation checks required will depend on the nature of the application and the impact of any corruption of data.

Areas of risks must be identified in the processing cycle and validation checks must be included for the identified risks. Appropriate controls must

[11] Although this standard is normally applicable to individual information systems, coding standards should be established for the whole Commission, not for each individual system.

[12] Client-side checking can have other benefits such as reducing unnecessary network traffic and server load.

be identified for applications to mitigate risks during internal processing. The architectural decisions on the functionality and implementation of these controls must be documented.

Examples of internal processing checks include:

- validating check digits in numerical fields (e.g. bank account numbers)
- session or batch controls, to reconcile data file balances before transaction commitment, and ensure the rolling back of failed transactions[13]
- balancing controls, to check opening balances against previous closing balances
- validation of system-generated data (e.g. checking whether values calculated by the system are within expected ranges)
- checks on the integrity, authenticity or any other security feature of data (or software) downloaded/uploaded, or verifying hash totals of records and files
- checks to ensure that application programs are successfully run at the correct time (for instance, if a job that is normally executed at night is run during the day, this may indicate an error or unauthorised activity)
- checks to ensure that programs are run in the correct order and terminate in case of a failure, and that further processing is halted until the problem is resolved.
- handling exceptions with meaningful error codes or messages preventing blocking situation and allowing debugging
- creating a log of the activities involved in the processing.

9.4. Message Integrity

Information systems often receive data in the form of messages from other systems (or modules of the same system), typically containing a number of data fields. Integrity checks must be performed on all such transmissions to ensure that the information has not been accidentally or deliberately corrupted.

Issues that can be detected through message integrity checks include data modification, substitution or replay, or incomplete data.

[13] This is often part of (distributed) transaction management. Additional checks must be coded in case this is not managed automatically.

If an integrity issue is detected, the system must log the error and take appropriate action, such as:

- ignore the message
- request the data again
- inform the administrator
- reroute traffic to other lines

There are several types of message integrity controls giving different types and levels of assurance, including:

- Checksums or check digits
- Message digests or hashes
- Message encryption
- Message Authentication Code (MAC)
- Digital signatures

If any form of encryption is used as a measure for message integrity, the rules in the *Standard on Cryptography and Public Key Infrastructure* must be followed.

9.5. Output Data Validation

Output data may be incomplete or incorrect, even after checks on input data, messages, and internal processing. Consequently, integrity checks must be in place to permit validation of output data.

The design and implementation of the application must ensure that the risks to information integrity are minimised. Specific output validation controls to consider are similar to those listed above for input data, messages and internal processing checks. These controls may include:

- plausibility checks to test whether the output data is reasonable
- reconciliation control counts to ensure processing of all data
- completeness checks to ensure that all intended output information is present
- displaying or printing additional information to permit manual validation of output data

10. SECURE DEVELOPMENT ENVIRONMENT

Policy objective 7.2.5 – Protection of System Test Data – Test data must be selected carefully and protected and controlled.

Policy objective 7.2.6 – Control of Operational Software – Installation of software on operational systems must be strictly controlled.

10.1. System Test Data

The adequate testing of information systems before implementation is critical to ensure their confidentiality, integrity and availability. Inadequate testing may lead to problems such as breaches of confidentiality, data corruption or system crashes, and so testing is a fundamental part of system security.

Test data must be selected and/or designed carefully to ensure that test procedures are adequate. In particular, testing must check that:

- Data processing operations are performed correctly
- Systems can deal with both correct and incorrect data, including malicious data inputs (such as SQL injection attempts)
- Systems can cope with the expected peak volumes of data
- Error-checking and validation routines are in place and functioning properly
- Error messages do not give information that could be useful to malicious users (e.g. software versions, names of database elements etc.) – instead, this information should be recorded in error logs that are not accessible by end users.

Test data should therefore include invalid data (e.g. text or special characters where numbers are expected, or invalid dates such as 31/02/2010) which must be created manually. Test data may also include copies or extracts of live data under the conditions given below. It may be necessary to have multiple sets of test data to test all aspects of a system.

The use of operational databases containing personal information or any other sensitive information for testing purposes should be avoided where possible. Testing with personal or otherwise sensitive information is permissible provided that the development is performed intra muros (within the Commission), confidentiality agreements are in place with any external staff having access to the data, and data access rights respect the "need to know" principle. If any of these conditions are not met, then all sensitive details and content must be removed or modified beyond recognition before use.

System testing must not be performed on operational data unless the required safeguards are in place. Rules must exist for the use of production data during system testing and must include the following:

- When systems are tested, the resulting data and test results must be handled as sensitive information until the information can be disposed of properly.

- Any software tools specifically used for testing code and data must be protected from access by unauthorised people.

- Every copy of operational information to a test application must be authorised.

- The copying and use of operational information must be logged to provide an audit trail.

- The use of operational information for testing purposes must be approved by the System Owner (or Data Owner, if separate), or an approved delegate.

Test data must be protected and controlled by suitable physical and technical controls and procedures. The classification of the original data, where used, must be respected and appropriate security measures taken in testing environments. If data sets used for testing are modified in a way that reduces their classification level[14], security measures may be applied to the resulting test data sets at a lower level (whilst respecting the minimum rules stipulated by this standard); however, care must be taken to ensure that the original information cannot be recreated or inferred from the modified test data.

Guidance on issues that should be tested may be found in Annex 1.

10.2. Operational Software

Security procedures must be in place to cover the implementation of software on operational systems (both servers and workstations), and the associated movement of code.

When establishing controls on production or operational systems the following must be considered:

- Operating systems must be hardened, notably by removing powerful software utilities that are not needed for normal system operations (e.g. compilers or debuggers).

- Production servers must have formal change management processes in place (see section 11.1). These systems may only hold approved executable code, and not development code or compilers (unless required for normal operation).

- The updating of the operational software, applications, and program libraries must only be performed by trained administrators upon appropriate management authorisation and following extensive and successful testing (see section 10.1 above). An audit log must be maintained of all updates to operational systems files.

[14] Reducing the classification level for test data must be done with due care and consideration of the risks involved. In particular, the damage to reputation of a breach of confidentiality of test data may be as great as a breach of real data, even if the information has been anonymised.

- Previous versions of software must be retained as a contingency measure.

- Old versions of software should be archived, together with all required information and parameters, procedures, configuration details, and supporting software for as long as data requiring these versions is retained in archive.

- All change control operations must include roll back plans and event logging.

Wherever there is a potential impact, changes to operating system and any other software running on live systems must be tested for compatibility with the application software before being implemented.

Any third party supplied software used in operational systems must be maintained at a level supported by the supplier and software patches should be applied (after testing) when they can help to remove or reduce security weaknesses.

Physical or logical access may only be given to suppliers for support purposes when necessary, and with management approval. The supplier's activities must be monitored.

Where possible, automated integrity checking mechanisms should be implemented to ensure the integrity of the software.

11. SECURITY IN DEVELOPMENT AND SUPPORT PROCESSES

Policy objective 7.3.1 – Change Control Procedures – The implementation of changes must be controlled by the use of formal change control procedures. Modifications to software packages must be discouraged and limited to necessary changes, and all changes must be strictly controlled. Access to program source code must be restricted.

Policy objective 7.3.2 – Code Review – Software code developed for Commission services must be reviewed using a formal process in order to guarantee that its functions are implemented in accordance with the specifications and that there is no malicious code.

11.1. Change Control Procedures

11.1.1. General Rules

This section gives rules for controlling changes to software[15] and related materials (such as design and test documents and data). This includes software developed by or on behalf of the Commission, as well as

[15] Change management is also covered, at a more general level, by the Standard on Operational Management. The change control rules given here are compatible with this standard, but contain details that are specific to systems development processes.

commercial products that require changes (updates, patches etc.), although some of the measures are not applicable for commercial products (as noted in the text).

Change control is a complex subject, and this standard focuses only on the high level security requirements for the software change control process. The procedures used must be designed carefully on the basis of information such as the system's requirements, CIA classifications and operating environment. The rules in this section are the minimum that must be followed.

Formal change control procedures must be in place for all systems owned by the Commission. The objective of these procedures is to protect systems from unauthorised or accidental changes and to ensure that systems are adequately tested before implementation (i.e. primarily to safeguard the integrity of the source code, although its confidentiality and availability are also protected).

Key security-related aspects of change control are found in section 11.1.5 below. A complete audit trail must be maintained for all change control processes.

11.1.2. Segregation of Duties

The different roles involved in making and implementing changes must be segregated to ensure that unauthorised changes are not made. At a minimum, developers, system/acceptance testers and operational staff must be separate, and developers must not have write access to production systems. Where possible, personnel administering the change control procedures and system testers should also be segregated from the other roles.

The appropriate segregation of duties should be determined on the basis of the CIA classifications of the system and risks such as potential conflicts of interest or opportunities for fraud, and approved by the System Owner.

11.1.3. Protection of technical environments

Separate environments must be used for Development, Testing and Production (other environments may also exist, although generally they will fit into one of these three categories). The transfer of files between these environments must be closely controlled as described in section 11.1.5 below. The Testing environment must be as similar as possible in its technical configuration to the Production environment to reduce the risk of systems behaving differently in Production than expected.

Access to source code, compiled code, development tools and test data must be limited to authorised individuals in all environments. Where an "open source" approach is taken, the same principles must be followed even though the development community authorised to have read access to code may be larger (and possibly in some cases extending outside the Commission).

11.1.4. Software Repository

A software repository must be implemented for all software developed by or on behalf of the Commission to protect all relevant electronic records from loss or unauthorised changes. The repository should hold copies of documents, code and other items such as:

- System requirements
- System design
- Source code
- Test data
- Compilers
- External libraries
- Licences
- Encryption keys (where appropriate; see the *Standard on Cryptography and Public Key Infrastructure* for more information on when keys may or may not be copied)
- User and operational documentation

The repository must be protected adequately and access to it restricted on a need-to-know basis.

11.1.5. Change Control Processes

Change Request

All changes must be initiated with a formal change request. In order to obtain approval, changes must:

- be adequately documented
- be submitted by authorised personnel
- be reviewed for any potential impact on system security or related processes
- be managed so that they do not conflict with or overwrite other changes to the same or related systems
- be restricted to the minimum necessary for business and security needs in order to limit the work, disruption and potential risks involved

- be approved by the SO[16] before code is released or work begins on the change

Release Source Code

For changes where source code (or similar objects) must be updated, the code must be released to the developer(s). Source code that is developed by or on behalf of the Commission must be protected (e.g. through the use of a change control system). The measures taken to protect source code must ensure that:

- The production system is built from code in the versioning system, and developers must not be allowed to directly manipulate the code or binaries of the production system
- Developers must be specifically authorised to access source code or binaries on a need-to-know basis
- Developers must not interfere with version management systems by directly modifying source code in the repository
- All source code is marked with version numbers
- Controls must be in place to ensure that legitimate changes are not accidentally overwritten, e.g. due to simultaneous changes
- At least the last six versions or two years' history of source code must be retained (whichever is longer)

Accept changes

Modified source code must be submitted for formal testing when the development phase is completed. For code to be accepted into formal testing, the following conditions must be fulfilled:

- The changes must have successfully[17] completed local testing (unit testing, program testing etc)
- All modified source code must have been subjected to a code review (see section 11.2 below)

[16] ITIL recommends the use of a Change Advisory Board (CAB) consisting of a number of members to approve change requests. This is recommended for larger systems in the Commission.

[17] The definition of a successful test is not straightforward, since some errors or issues identified during testing may not be serious enough to require remediation. The System Owner should review the test results and decide whether the test results are acceptable. From a security perspective, there should be no significant exploitable vulnerabilities in the code for it to be acceptable.

Transfer to Production

For the change to be implemented, the code[18] must be transferred to the Production environment. Before this can take place, the change must:

- have successfully passed all required testing phases in the Test environment (e.g. system tests, volume tests, acceptance tests, penetration tests)
- be fully documented, including updates to existing system documentation, operating procedures, user instructions etc.
- be signed off by the project leader and the SO or an approved delegate.
- be properly scheduled, and all personnel and users concerned notified of the change

Emergency Changes

Occasionally, changes to systems are necessary at very short notice and there is insufficient time to go through the normal change process. An emergency change control procedure must be drawn up for these situations. This procedure must include:

- An evaluation of the risks involved in the emergency change before it is performed
- Appropriate testing to the extent possible within the timeframe
- Formal approval (including acceptance of the risks involved) from the SO or an approved designate before an emergency change is made
- A back-up of the elements to be changed
- A review of the emergency change shortly after it is implemented
- Application of other key measures in the change control process after the emergency change, including a formal review and acceptance
- Identification of any ongoing normal changes that are affected by the emergency change and notification of the personnel involved.

11.1.6. Changes to Third Party Products

As a general rule, vendor-supplied or open-source software packages should be used with as little modification possible. When updates to acquired

[18] This standard will not address the question as to whether source code or compiled code should be transferred since this may depend on the technology used. However, whichever method is chosen should be used consistently and in accordance with secure best practices.

products are released by the publisher, they should be evaluated to determine whether the change needs to be applied to the Commission's production systems. If so, the change must follow the relevant processes described in section 11.1.5 above (i.e. omitting any steps that relate only to systems developed by or on behalf of the Commission).

Whenever it is proposed to modify an acquired product for the Commission in a way that may make it incompatible with the published version, the following points should be considered:

- the risk of built-in controls and integrity processes being compromised
- whether the consent of the vendor should be obtained in cases where the publisher does not perform the changes
- the possibility of obtaining the required changes from the publisher as standard program updates
- the impact if the organisation becomes responsible for the future maintenance of the software as a result of changes

Changes to packaged software should be minimised and carried out in such a manner that the change will not introduce new problems. Security requirements relating to the change must be designed and implemented.

Where the standard packages are modified, the original software must be retained and the changes applied to a clearly identifiable copy.

All such changes must be fully tested and documented, so that they can be re-applied if necessary to future software upgrades.

11.2. Code Review

Code reviews have been shown to improve the quality of source code and to reduce the cost of subsequent maintenance, as well as identifying potential bugs, design flaws, unauthorised functionality and backdoors. All systems that are developed by or on behalf of the Commission must be subjected to code reviews as part of the system development assurance process. The level of code reviews performed should be proportional to the perceived risks of the information system.

Developers and reviewers must be trained in secure coding methods appropriate to the languages used, and coding standards must be documented and put in practice. Code reviews must be performed by personnel other than the developers who wrote the code (they may be performed as peer reviews by other developers).

There are several different methods for performing code reviews. The project leader should select an appropriate method during the project initiation phase, and include the code reviews in the work estimates. Automated code review tools may be used but may not fully replace human review, since they cannot cover all aspects of a code review (particularly

identifying malicious code). Where possible, code review tools should be configured to support the coding standards.

Clear objectives must be set for each code review by understanding the business requirements of the system, its environment, the relevant coding standards and any potential security issues relating to the code under review.

For further information relating to code reviews, see Annex 1 of this document.

11.3. Penetration Tests

Penetration tests are aggressive security tests whereby the tester simulates the actions of a potential attacker in attempting to cause the system to fail, usually with the objective either of gaining unauthorised access or of causing a denial of service. They are also commonly referred to as ethical hacking.

For the purposes of this document, penetration tests are considered to be any tests where the actions of a potential attacker are simulated in identifying or attempting to exploit vulnerabilities in an information system. This includes vulnerability scans, which are less aggressive than full penetration tests.

Penetration tests should be performed on systems as defined in section 8.2.3 earlier in this document, and may also be performed during the lifetime of a system in production.

Whenever a penetration test is performed which could affect other Commission systems[19], the following rules must be followed:

- The test must be documented in a formal convention which must be signed for authorisation before the test by the System Owner, the company performing the tests[20], DIGIT and the Security Directorate. The convention must include:
    - The purpose, scope and nature of the tests.
    - A detailed test plan and methodology.
    - The IP addresses of all target machines, as well as those of the source machines used for the tests. These IP addresses must be carefully checked by the System Owner and DIGIT.
    - The timeframe during which the tests will be performed.
    - A non-disclosure agreement signed by the tester.

[19] For example, if the tester has to pass through other parts of the Commission network before reaching the system to be tested. If the test takes place on an isolated system with no connection to the Commission's internal networks, then the rules should be considered as guidance instead.

[20] Penetration tests are normally performed by external security consultants.

- The company performing the penetration tests must provide evidence of its competence, and that of the personnel involved, to perform the tests.

- Appropriate controls must be established around the test environment.

- All penetration tests must be closely followed by an observer from the Commission.

- Tests must not be performed on any systems or components outside the documented scope.

- The results of the penetration test must be reported to the Security Directorate and to DIGIT, together with a description of the remedial actions taken or planned.

12. OUTSOURCED SOFTWARE DEVELOPMENT

Policy objective 7.3.3 – Outsourced Software Development – Outsourced software development must be supervised and monitored and its deliverables formally accepted before installation and usage.

12.1. General Rules

Information systems are often developed by third parties on behalf of the Commission, and so the Commission may not always have direct control of the software development process. In these cases, the rules in this section must be followed.

Source code must always be available to the Commission. Consequently, where software development is outsourced, the outsourcing contract must contain provisions for the code ownership and intellectual property rights or licensing arrangements, depending on the situation. If ownership of the code remains outside the Commission, then the contract must include an escrow arrangement.

The contractual agreement must specify the requirements for the quality and security functionality of the code. This must include provisions for ensuring the quality and accuracy of work done at the premises of the third party (for example, the right to audit).

All externally developed software must be formally tested and accepted by the system owner before it is installed on the production environment. All externally developed software must be tested for malware before installation on any EC environment.

13. ROLES AND RESPONSIBILITIES

System Owner: responsible for reviewing and approving system development and acquisition activities, and ensuring that they conform to the business and security requirements.

Developers: responsible for following secure development procedures and being aware of coding standards and security issues.

Project managers: responsible for ensuring that the security rules applying to the different phases of the SDLC are followed.

IRM teams: responsible for operating systems housed within the Commission, and for advising on technical aspects of software development and acquisition projects.

LISOs: responsible for advising on the security requirements for new and existing systems, and representing security interests in software projects.

Testers: responsible for making sure that information systems conform to the general and specific security requirements.

Procurement teams: responsible for ensuring that the specifications for systems to be acquired include appropriate security requirements.

System providers: responsible for operating systems housed within the Commission, and for advising on technical aspects of software development and acquisition projects.

Security Directorate: provide guidance on all special cases on request from DGs.

14. REFERENCES

- Commission Decision C(2006) 3602 of 16/8/2006

- Implementing rules for Commission Decision C(2006) 3602 of 16.8.2006

- Commission Decision 2001/844/EC, ECSC, Euratom

- International standard ISO/IEC 27001 – Second edition 2005-06-15

- International standard ISO/IEC 17799 – Second edition 2005-06-15

- Common Criteria for Information Technology Security Evaluation part 3: Security assurance components – v. 3.1 Revision 2

- ITIL (Information Technology Infrastructure Library)

- NIST SP 800-27 Rev A - Engineering Principles for Information Technology Security (A Baseline for Achieving Security), Revision A

- NIST SP 800-64 Security Considerations in the System Development Life Cycle

- OWASP (Open Web Application Security Project) - http://www.owasp.org/

15. RELATED DOCUMENTS

Note that standards marked (*) are in draft at the time of writing of this standard.

- Standard on Access Control and Authentication (*)
- Standard on Physical and Environmental Security
- Standard on Cryptography and Public Key Infrastructure
- Standard on Asset Management
- Standard on Information Security Risk Management (*)
- Standard on Business Continuity Management
- Standard on Network Security (*)

16. ANNEX 1 – SUPPLEMENTARY INFORMATION FOR CODE REVIEWS

This annex contains additional information for guidance on code reviews. Detailed information on how to perform the reviews is specific to each programming environment and is outside the scope of this document.

Reviewing source code is a complex process and there are many potential weaknesses. For example, when reviewing a source program, the following categories should be considered for review:

- SQL injection
- Cross-site scripting
- Input/output data validation
- Authentication
- Authorisation
- Handling of sensitive data
- Password security
- Exception management
- Data access
- Cryptography
- Use of uncontrolled code (e.g. external libraries)
- Configuration

- Threading
- Undocumented but accessible interfaces (backdoors)

All subsequent changes should be commented in the source code, and subsequent reviews may then focus on the code that has been changed in the latest version.

The following are examples of objectives that can be set for this exercise:

- Make sure that all untrusted input to the component is passed to a validation routine before it is used.
- Check error handling procedures to make sure that exceptions are caught consistently and caught close to their source.
- Check calculations whose results are used for memory allocation or buffer access for numeric overflow or underflow.
- Check cryptographic routines to make sure secrets are cleared quickly.
- Check for unauthorised functionality or backdoors

Code should be reviewed incrementally and iteratively and limited to small, manageable pieces of code.

After each code review, the coding standards may need to be updated to reflect the latest best practices of code review and programming standards, or to highlight issues that appear repeatedly.