COUNCIL OF THE EUROPEAN UNION

Brussels, 5 March 2013

7124/13

CSCI 19
CSC 26

NOTE

From :	The General Secretariat)
To :	Delegations
Subject :	Information Assurance Security Guidelines on Web Applications

1. The Council Decision on the security rules for protecting EU classified information[1] states that "The Security Committee may agree at its level security guidelines to supplement or support this Decision and any security policies approved by the Council." (cf. Article 6(2)).

2. The CSC approved the attached Information Assurance Security Guidelines on Web Applications on 15 February 2013.

[1] Council Decision 2011/292/EU, OJ L 141 of 27.5.2011, p. 17.

This page intentionally left blank

IA Security Guidelines on Web Applications

IASG 5-06

TABLE OF CONTENTS

I PURPOSE AND SCOPE

1. These guidelines, agreed by the Council Security Committee in accordance with Article 6(2) of the Council Security Rules (hereinafter 'CSR'), are designed to support implementation of the CSR.

2. The purpose of these guidelines is to establish minimal standard for mitigation of risks that are typical in a web environment processing EUCI.

3. The General Secretariat of the Council (GSC) will apply these security guidelines in their structures and communication and information systems (CIS).

4. When EU classified information is handled in national structures, including national CIS, the Member States will use these security guidelines as a benchmark.

5. EU agencies and bodies established under Title V, Chapter 2, of the TEU, Europol and Eurojust should use these security guidelines as a reference for implementing security rules in their own structures.

6. These guidelines are addressed to staff responsible for the accreditation, purchase, project management, testing, design and implementation of web applications handling EUCI. Depending on the risk levels defined in section IV paragraph 35, for any given project to provide web applications, there are corresponding controls (as defined in Annex 2, Annex 3 and Annex 4) to be applied.

7. This document covers the aspects of a web application that have potentially a security impact, particularly those that are web specific. Therefore it focuses mainly on design, implementation and testing. Other, generic obligations for a CIS (e.g. TEMPEST, accreditation, monitoring, physical security etc) are defined in the CSR and relevant security policies.

8. Deployment of web technologies, although relying on TCP/IP stack, does not automatically imply interconnection with any specific type of network (Internet in particular). This document, therefore, does not address such aspects. The conditions and rules regulating the interconnection are explained in [9] and [10] .

II INTRODUCTION

9. Web applications have become increasingly popular as the availability of appropriate standard, off-the-shelf solutions offer good flexibility for client environments.

10. For the purpose of this document a "web application" is any computer system (handling EUCI) which offers a user interface in the form of a web site[2], accessible via the HTTP protocol [3], regardless of the functionality offered. It may, for example, provide an administration service for an IP router or offer services such as streaming from a web camera, or it could implement a document management system.

11. There is no uniform web application architecture. Web applications may be a part of a larger system and/or themselves composed of other computer systems. Any of such subcomponents or composites that are not specifically designed to handle HTTP protocol and/or HTML documents, are not in the scope of these guidelines (see Figure 1).

12. The architecture of a web application is composed of at least two parts:

    (a) A web server which is a program interpreting HTTP requests (request methods [3]) and producing responses;

    (b) A web client (typically multiple clients) - a program (called user agent [3]) sending requests and interpreting the results returned by the server. In most cases the client is responsible for rendering the graphical user interface (GUI), usually represented by HTML documents (web pages) and associated technologies (see paragraph 14).

13. A client usually has limited functionality and the processing is performed by the server. The client interpreting HTML documents and rendering the user interface is called a "web browser". Other types of clients[3] include: Web Services consumers, REST clients, WebDAV clients, other non-GUI HTTP clients.

14. Modern web browsers can perform a part of the processing (most often associated with GUI rendering) using, for example, the following technologies:

    (a) active components (e.g. Java applets, ActiveX, flash applications, plug-ins etc),

[2] A set of HTML[4] documents (webpages) including active content and resources linked to the HTML document.
[3] Often in B2B context, typically using other protocols that can be transported via HTTP: SOAP, XML-RPC, REST, WebDAV, JSON etc

(b) scripting languages (e.g. Java script, Visual Basic script) and

(c) web browsers/HTML features (e.g. CSS, AJAX, HTML forms).

The expected introduction of HTML version 5 may increase even more the amount of processing on the client side.

15. The server is designed to handle HTTP requests and where applicable higher level protocols transported over HTTP (e.g. SOAP [5], REST, XML-RPC[6], JSON [7] etc). Each request invokes certain services on the server side (e.g. stores data in a database, produces data). In return, the client receives a response in a form of a document (HTML or other types of documents).

16. Each item of data or service accessible to the web client is available via a uniform resource locator (URL) [12] which provides information about the location of the resource and the way to reach it in a standardised format (e.g. http://hostname:port number/path?searchpart).

17. Apart from the front-end components accessible to web client, the web application usually also connects to back-end elements (e.g. relational databases, name services, messaging systems, operating system services, email systems, rule engines, other back office services) as illustrated by Figure 1.

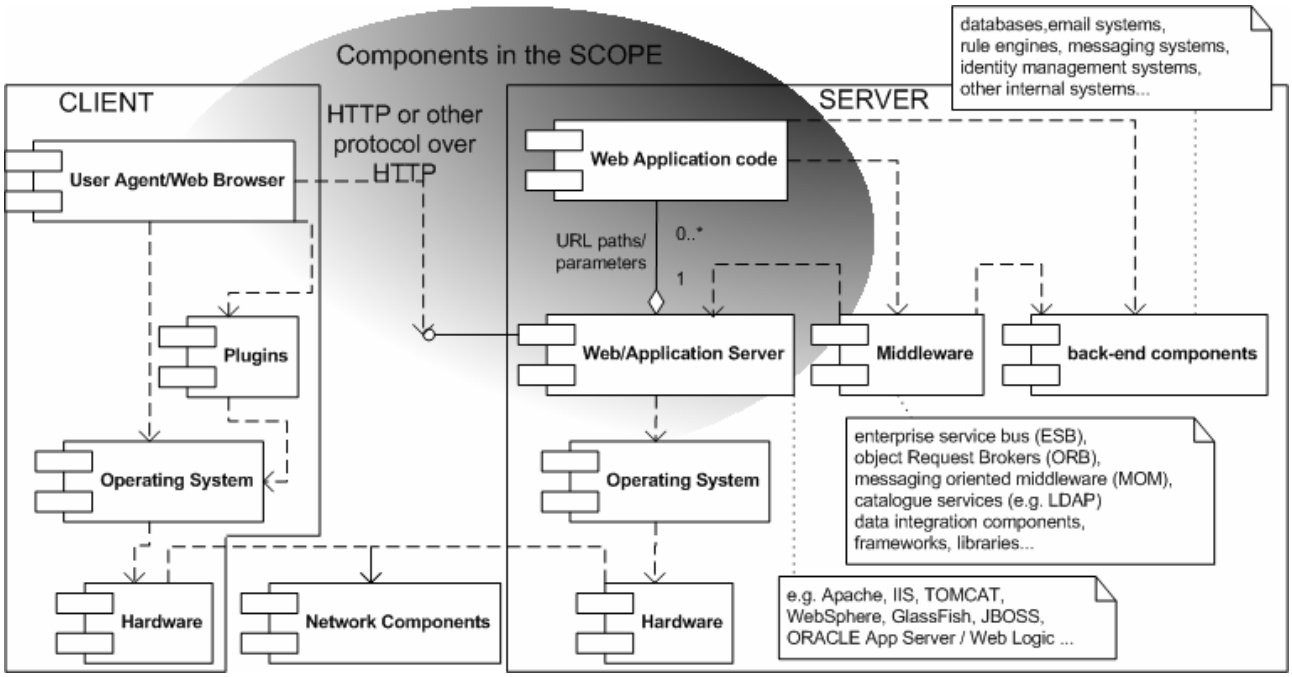

Figure 1 Example web application architecture and components in the scope of this document

III CHARACTERISTIC SECURITY THREATS FOR WEB APPLICATIONS

18. Detailed lists of generally relevant threats and vulnerabilities are made publicly available by CERT institutions and various commercial and non profit organisations (e.g. The Open Web Application Security Project (OWASP) [11], The Web Application Security Consortium (WASC) [13], etc).

19. The number of exploitable vulnerabilities and threats increases with the development of new products and advances in the research. This increase is so rapid that it is impossible to provide an up-to-date, exhaustive catalogue. Therefore, as with all systems, the list of threats and vulnerabilities identified as a result of the security risk assessment for a particular web application should be regularly revisited and revalidated in the light of future changes in technology.

20. The remaining part of this section contains a list of the most serious current vulnerabilities and threats that must be taken into consideration when conducting the security risk assessment of any web application.

21. **Eavesdropping** - The HTTP protocol does not offer any kind of encryption or confidentiality protection mechanisms. Therefore, by default, all data exchanged between the server and client can be intercepted on the client or server side, or inside the communication channel.

22. **Weak (or lack of) access control** - The data and/or services exposed by the web server may require access control. To provide the authentication and authorisation developers are often tempted[4] to rely solely on the secrecy of URLs or on vulnerable, *ad hoc* mechanisms of access control. As a result, the data can be accessed without any restrictions when the URL becomes known or the implemented protocol is exploited.

23. **Broken authentication and session management** - The HTTP is a stateless protocol thus storing the information about a state requires an additional implementation of a session concept. Such an implementation relies on higher level protocols and techniques implemented by the web application itself, often without a solid security analysis. These consequently may allow adversaries to take over authenticated session and impersonate legitimate users.

[4] HTTP offers, basic, digest, NTLM and negotiate authentication scheme with granularly higher level of security including the use of Kerberos (in negotiate scheme). In all cases however the user interface provided to enter the user's credential is under control of the web browser which is often inconvenient for developers.

24. **Replay attacks** - The HTTP protocol does not intrinsically ensure the authenticity of requests, consequently adversaries can successfully spoof the origin of a message by resending its prior eavesdropped copy (even if it is encrypted). In this way, a web application may be vulnerable for unauthorised access to data or other threats.

25. **Privacy** - The behaviour (e.g. changing colours, caching) of the web browsers may reveal personally identifiable information (PII) including: services accessed, parameters passed, URLs visited, downloaded images. Such data can be then accessed by unauthorised parties (including other web applications).

26. **Web page integrity/authenticity** - Both the content and the origin (URL) of a web page can be spoofed if no protection mechanisms are implemented. The attack can be based on visual similarity, mistyping the URLs, changing DNS entries, spoofing IP address, changing the content of a webpage in the communication channel or it can be performed by exploiting vulnerabilities in the web application (e.g. XSS).

27. **Web browser vulnerabilities** - There are a very limited number of web browsers on the market and typically the choice of web browser is not controlled by the server. Hence any security vulnerability found in one of the web browsers can be potentially applicable to all web applications accessed form a workstation which uses the affected web browser. The vulnerabilities may allow to execute an arbitrary programme or to access sensitive data on the client workstation (e.g. downloaded files, images, passwords, email addresses etc).

28. **Web servers vulnerabilities** - In most cases the basic web server functionality (handling network communication, processing requests, multitasking, administration etc) is provided by underlying COTS or FOSS products. Since the number of web server products is very limited, vulnerabilities found in popular server technologies may have an impact on substantial numbers of web applications.

29. **Plug-in vulnerabilities** - Some of the technologies commonly used in web applications rely on reusable, sometimes web-browser neutral, components called plug-ins (e.g. OLE components, adobe flash, adobe reader, media players etc). Any vulnerability of such a plug-in applies to all web applications using it, usually regardless of the web browser selected.

30. **Denial of Service (DoS) -** An intrinsic vulnerability of web servers is their susceptibility to denial of service attacks. Each web server has a limit for the maximum number of simultaneously processed requests, dictated by the server's design and available resources. Since processing is typically performed by the server, it is often possible for web clients (especially those with malicious intentions) to overload the server so that it stops responding to legitimate requests.

31. **Lack of input validation -** In any web application both graphical user interface (including executable parts) and data are mixed within the same document. If the web application fails to properly validate request data from the user and inserts it into a response, injection attacks such as for example cross-site scripting (XSS) and Cross-site request forgery (CSRF) may become possible.

32. **Insecure integration with backend services**- Web applications may use various internal services not exposed directly to end users but which are nonetheless controlled by them via parameters (e.g. redirect=url). An example of this is sending an email, redirection to other page, reading data from a backend database or download of data. Such internal services may pass limited security checks as not being considered usable by end users, and therefore they may contain more vulnerabilities. Consequently this could be exploited by adversaries to gain unauthorised access to data or a service, to direct users to a malicious web page or to change expected behaviour of a web page and/or its content.

33. **Indirect attacks on backend components -** Due to the increasing complexity of web architectures and technologies, overall systems have become very difficult to comprehend and analyse. Overlooked or misunderstood elements may allow for web applications to be used as means to mount a further attack on backend components. This include attacks like SQL injection, exploiting vulnerabilities of web frameworks (e.g. .NET, JEE, Hibernate, Spring or Struts), operating systems on which servers are run or even protection mechanisms (e.g. log analysers).

IV SPECIFIC GUIDELINES FOR WEB APPLICATIONS

34. Regardless of whether web application is purchased, used in a software as a service model (SaaS) or developed, **the security aspects must be taken into account from the beginning of its life cycle**. The countermeasures need to be implemented at the organisational, technical and physical levels.

35. While the choice of a risk evaluation method is not within the scope of these guidelines, it is expected that the risks can be at least categorised as to the levels: **high, enhanced, standard, basic**[5]. These reference levels will be further used in these guidelines when selecting security controls or countermeasures.

IV.1 Business and System Requirements Analysis Guidelines

36. The requirements analysis comprises the analysis of the business problem (what is the problem to be solved by a CIS) and the analysis of requirements for the IT system (how will the problem be solved by the use of the CIS). Typically this phase does not include discussions about specific technologies (web based in particular) unless this is a part of the explicit constraints (e.g. there is only a small subset of solutions permitted within the organisation).

37. It is recommended that no decision about the deployment of web technologies is taken without a risk assessment and full understanding of the business problem. The choice of technology must not take precedence over the business needs and in particular the security of EUCI.

38. The functional requirements for the web application should consider offering context-sensitive help for security related functions and settings. There should be a way the user can contact the operator of the application when a security incident is suspected (email, html form).

39. Annex 1 contains the business characteristics that must be considered before deciding whether or not to implement the CIS as a web application.

[5] It is assumed that whatever method is used for risk management it will be possible to map the level of risk into at least four values. The name (high, enhanced, standard, basic) is used as an internal reference only and it does not correspond to any concrete methodology or standard.

IV.2 Design Guidelines

40. The design of a web application or any CIS includes decisions about:

    (a) the type, functionality and number of components used as well as connections between them and

    (b) the choice of technologies (including protocols and data formats).

41. The quality of the design phase in a project must be ensured by the verification and validation of the design decisions (e.g. there might be a need for formal proofs or prototypes).

42. The minimum required features of the CIS must determine the choice of technology and architecture. Keeping complexity to a minimum may mean that a web application is replaced by something simpler (e.g. a distributed operating system or secure FTP/SCP server), if being web enabled does not bring any added value or it is not explicitly required.

43. The choice of design patterns (e.g. model view controller), modularisation, naming conventions etc are not within the scope of these guidelines, however the design style should support the security of the system as much as possible and facilitate the vulnerability assessment.

44. The architecture of the web application may consider deploying an additional component specialised in security functions (and others when needed), placed between the clients and web application as presented on Figure 2.

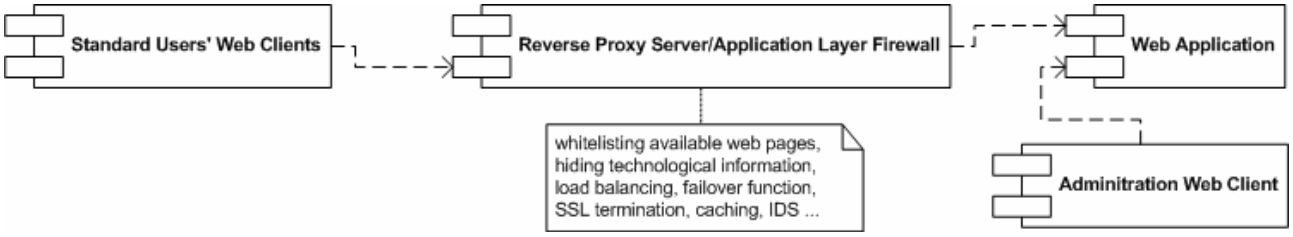

Figure 2 Example architecture with reverse proxy server

45. Annex 2 of this document provides solutions that must be considered in the design phase. The decision to select a particular countermeasure depends on how effective a countermeasure is in dealing with an identified risk (whether it mitigates the risk), and whether the value of the risk is sufficiently high to merit a particular countermeasure.

IV.3 Implementation Guidelines

46. As with most IT projects, the implementation phase includes system development (programming) and quality assurance processes (including security).

47. It is recommended that the technology used for the implementation is well known and proven from the security point of view (meaning that vulnerabilities have been studied, no new vulnerabilities have been found for a reasonably long time and there are known workarounds for existing problems). Technology which provides native support for security (e.g. by inbuilt access control mechanisms or session management) should be preferred.

48. It is impossible to define "one-size-fits-all" solution to guide the implementation of a secure web application. Therefore, the measures listed in Annex 3, should be used, taking into account the specificity of the concrete risk(s) addressed[6].

IV.4 Testing and Verification Guidelines

49. The objective of security testing and verification is to provide the required assurance that appropriate security controls are present and behave as intended.

50. The level of assurance must be proportional to the level of accreditation and the likelihood of the security incidents.

51. If permitted by the CSR, verification can be replaced by acceptance of a certification procedure performed by a third party (e.g. the presence of a product on the GSC list, one of the Member States or standardisation bodies). Such acceptance should be preceded by a detailed comparison between the security requirements for the web application and the requirements covered by the certificate issued by the third party.

52. The assurance should be based on the intensity of the security analysis and testing (e.g. number of security controls tested, percentage of lines of code analysed). Annex 4 defines example security verification levels. The requirements for higher levels include those defined for lower levels.

53. The tests have to be firstly performed on a test system (not on the production system) handling unclassified test data.

[6] Whether it derives from confidentiality, integrity or availability, what vulnerabilities are to be mitigated, effectiveness against particular threats etc

IV.5 Deployment and Maintenance Guidelines

54. Security related actions in the deployment and maintenance phase of a web application should include continuous detection and fixing of vulnerabilities, monitoring and reaction to attacks and ensuring information availability.

55. The regularity and depth of monitoring should be proportional to the level of identified risks.

56. Conflicts between availability and confidentiality or integrity risks (e.g. when applying patches) should be resolved at the business level by Sensitive CIS Manager (SCISM).

57. The web application documentation should include all requirements necessary for a secure deployment (network, infrastructure, servers, operating system, database, additional security controls) as well as maintenance and monitoring guidelines (e.g. URL for advisories and patches). All security-related settings should highlighted. The configuration options with a potential negative impact on security should be explained.

58. The web application should be deployed in a secure default configuration. If there are any steps necessary to get to a secure state, those should be well documented within the documentation and/or within the administrative interface (if possible this should be supported by the web application itself – e.g. change of administrator password should be enforced).

DEFINITIONS

Accreditation	Means the process leading to a formal statement by the Security Accreditation Authority (SAA) that a system is approved to operate with a defined level of classification, in a particular security mode in its operational environment and at an acceptable level of risk, based on the premise that an approved set of technical, physical, organisational and procedural security measures has been implemented;
Activation data	Data values, other than keys, that are required to operate cryptographic modules and that need to be protected (e.g., a PIN, a passphrase, or a manually-held key share).
Asset	Means anything that is of value to an organisation, its business operations and their continuity, including information resources that support the organisation's mission.
(Security)Assurance	Quantitative grounds for confidence that an entity meets its security objectives.
(Security)Assurance Level	Discrete value of assurance. In the context of Crypto Policy and cryptographic products there are only two levels - approved (which is expressed by putting a product on the list of products) or not approved (which is demonstrated by the fact product is not placed on the list).
Authentication	The process of establishing that individuals, organizations, or things are who or what they claim to be. In the context of a PKI, authentication can be the process of establishing that an individual or organization applying for or seeking access to something under a certain name is, in fact, the proper individual or organization.
Authenticity	The guarantee that information is genuine and from bona fide sources
Availability	The property of being accessible and usable upon request by an authorised entity.
Communication and information system (CIS)	Means any system enabling the handling of information in electronic form. A communication and information system shall comprise the entire assets required for it to operate, including the infrastructure, organisation, personnel and information resources. See Article 10(2) of CSR.

Confidentiality	The property that information is not disclosed to unauthorised individuals, entities or processes.
Cryptographic (Crypto) material	Means cryptographic algorithms, cryptographic hardware and software modules, and products including implementation details and associated documentation and keying material.
EU classified information (EUCI)	Means any information or material designated by an EU security classification, the unauthorised disclosure of which could cause varying degrees of prejudice to the interests of the European Union or of one or more of the Member States. See Article 2(1) of CSR.
Identification	The process of establishing the identity of an individual or organization, i.e., to show that an individual or organization is a specific individual or organization. In the context of a PKI, identification refers to two processes: (1) establishing that a given name of an individual or organization corresponds to a real-world identity of an individual or organization, and (2) establishing that an individual or organization applying for or seeking access to something under that name is, in fact, the named individual or organization. A person seeking identification may be a certificate applicant, an applicant for employment in a trusted position within a PKI participant, or a person seeking access to a network or software application, such as a CA administrator seeking access to CA systems.
Information Assurance (IA)	In the field of communication and information systems is the confidence that such systems will protect the information they handle and will function as they need to, when they need to, under the control of legitimate users. Effective IA shall ensure appropriate levels of confidentiality, integrity, availability, non-repudiation and authenticity. IA shall be based on a risk management process. See Article 10(1) of CSR.

Integrity	The property of safeguarding the accuracy and completeness of information and assets.
Non-repudiation	The ability to prove an action or event has taken place, so that this event or action cannot subsequently be denied.
Risk	Means the potential that a given threat will exploit internal and external vulnerabilities of an organisation or of any of the systems it uses and thereby cause harm to the organisation and to its tangible or intangible assets. It is measured as a combination of the likelihood of threats occurring and their impact.
Security risk management process	Means the entire process of identifying, controlling and minimising uncertain events that may affect the security of an organisation or of any of the systems it uses. It covers the entirety of risk-related activities, including assessment, treatment, acceptance and communication.
Threat	Means a potential cause of an unwanted incident which may result in harm to an organisation or any of the systems it uses; such threats may be accidental or deliberate (malicious) and are characterised by threatening elements, potential targets and attack methods.
Vulnerability	Means a weakness of any nature that can be exploited by one or more threats. A vulnerability may be an omission or it may relate to a weakness in controls in terms of their strength, completeness or consistency and may be of a technical, procedural, physical, organisational or operational nature.

REFERENCES

1. Council Security Rules	Council decision 2011/292/EU of 31 March 2011 on the security rules for protecting EU classified information
2. IASP 2	[RESTREINT UE/EU RESTRICTED] IA Security Policy on Cryptography
3. RFC 2616	Hypertext Transfer Protocol (HTTP) specification RFC2616 (http://www.ietf.org/rfc/rfc2616.txt)
4. HTML	HyperText Markup Language (HTML) specification version 4.01 (http://www.w3.org/TR/REC-html40/) version 5 draft (http://www.w3.org/TR/2011/WD-html5-20110525/)
5. SOAP	Simple Object Access Protocol (SOAP) version 1.2 specification (http://www.w3.org/TR/soap/)
6. XML-RPC	XML Remote Procedure Call specification (http://xmlrpc.scripting.com/spec.html)
7. RFC 4627	JavaScript Object Notation (JSON) specification (http://tools.ietf.org/html/rfc4627)
8. IASG BP-08	Information Assurance Guidelines on Reusable User-generated Passwords
9. IASP3	IA Security Policy on Interconnection
10. IASG3-01	IA Security Guidelines on Implementation of the Interconnection of CIS
11. OWASP	The Open Web Application Security Top 10 Project www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
12. URL	Uniform Resource Locator RFC 1738 (http://tools.ietf.org/html/rfc1738)
13. WASC	The WASC Threat Classification v2.0 http://projects.webappsec.org/f/WASC-TC-v2_0.pdf
14. Core J2EE design patterns	D. Alur, D. Malks, J. Crupi, Prentice Hall / Sun Microsystems Press ISBN:0131422464 2003(second edition)
15. RFC 2898 Paragraph 5.2	Password-Based Cryptography Specification Version 2.0, Password-Based Key Derivation Function 2 specification (http://tools.ietf.org/html/rfc2898#page-9)

Annex 1

IMPACT OF THE WEB TECHNOLOGIES ON BUSINESS REQUIREMENTS

Business characteristic	Impact
Security of business processes	The web environment allows for a limited control over client activities. Consequently, hazards such as the rapid transfer of large amounts of data, difficulties in the enforcement of the sequence of user inputs, uncontrolled delays between user activities, the possibility to stop a process at any stage, may be possible. The business process must therefore be designed to securely handle all deviations from its correct flow. For example, the input and attribution of the level of classification of EUCI must be performed in a single business transaction otherwise unauthorised users could gain access to data in the time frame between the action of storing EUCI and the recording its classification level or when the attribution of classification level is never completed;
The amount of data to be exchanged and the frequency of the exchange	Web based applications tend to generate more network traffic than "fat client" systems. Therefore, web applications dealing with extremely large documents, or extensive use of many fine-grained services may encounter problems with availability;
Constraints and sophistication of requirements for the user interface.	The difficulty to modify standard web browsers may limit or complicate the implementation of some requirements such as cryptography. This may also be an issue for cryptographic products used by the web browser (e.g. to establish an SSL tunnel), standard functionalities (e.g. printing), storage of data (e.g. pictures caching), hardware access, use of sophisticated authentication methods (e.g. biometrics), protocols used etc.;
Mode of operation	Use of standard web browsers in the CIS operating in a multilevel security mode may cause additional technical problems[7] ;
Scalability and further extensions needs	Even though the web application may be more susceptible to DoS attacks, there is a great choice of mechanisms to scale the systems;

[7] This is due to the fact that web browsers or their plug-ins can implement techniques of caching (i.e. temporary, uncontrolled storage of data) and anybody having physical access to the particular workstation could gain an access to such data.

DESING LEVEL COUNTERMEASURES

Measure / design decision	Description	Risk
Defence in depth	The web application must use a range of countermeasures addressing the same risk in various layers of the system (e.g. use of data leak prevention tools, double checks etc). Use of web application firewalls (as opposed to generic firewalls) is recommended.	all risks at the level standard or above
Defensive design	At least the following techniques must be used: - all data, even unclassified, must be considered sensitive unless it is proven that its compromise has no security impact. Consequently the data and communication channel should be appropriately protected; - the amount of data sent to client (or available in any way) must be limited to the strict minimum; - access to all services, including those not directly exposed to end users, must be controlled and monitored; - system must be designed to behave securely in case of malfunction or depletion of resources (safe fail); - security components must be designed in such way that they actively grant access to services or data, i.e., all access is blocked unless actively permitted; - within legal limitations, the deployment of honeypots and other techniques to detect and/or deceive potential attackers is recommended.	all levels of risk
Use of simplest possible technologies	The technologies chosen for the implementation of the web application must be the simplest possible and solid from a security point of view. Priority must be given to security rather than graphical attractiveness. Use of sophisticated frameworks, the full functionality of which would be only marginally used, is to be avoided. Unused features and configuration items (e.g. modules, plug-ins etc) must be switched off/uninstalled.	all levels of risk

Standardisation	COTS or FOSS products, design patterns and protocols must, where possible, comply with recognised standards respecting security. If needed products may be certified at an appropriate assurance level (e.g. Common Criteria).	all levels of risk
Resilient design	Resistance to security attacks must not rely on hiding system design decisions (e.g. algorithms, components etc) to internal controls ("no security by obscurity") nor does it mean that the design will be made publicly available.	all levels of risk
Least privilege principle	Any component acting on behalf of users must have only those privileges required to achieve its legitimate objectives. In the context of a web application the following design patterns must be used: - if the server is implemented by a process run on an operating system, the privileges of the process must not be higher than the necessary minimum required, to complete all legitimate tasks; - access from the web server to any back-end components (other systems, networks etc) must be limited to the minimum and strictly controlled in a form of a white list (e.g. by placing the infrastructure in a DMZ); - the access from the web server to back-end services must be either dedicated to each user or at least segregated according to the privileges of the users. If a web server implements any technique of resources/services connections pooling (e.g. to relational databases) the accounts used to access such resources may be grouped according to the privileges (e.g. for a database connection, instead of using one most powerful account there would be a couple of accounts: for reading unclassified data, reading classified data, inserting new data or for modifying and deleting etc).	all risks at the level standard or above
Maintainability	The web application should be designed to facilitate patching taking into account its availability needs (e.g. it may make use of redundant test environment).	all levels of risk
Intuitive and readable user interface	The user interface must be designed to prevent users from making unintended mistakes in all allowed web browsers. For example: insecure or irreversible actions should require double confirmation, security parameters (e.g. use of SSL) should be visible to the users, the information should be clearly presented using understandable terminology and classification markings, use of default buttons, keys or settings should lead to secure states only, complex actions or interfaces larger than one screen should be divided into smaller parts.	all levels of risk

Use of appropriate cryptographic products	Use of cryptographic products selected according to the crypto policy [2] is required in all cases where cryptography is involved in the protection of EUCI. In accordance with the CSR (see Council Security Rules [1] Article 10(6)) all electronic transmission of EUCI has to be protected by approved cryptographic products to ensure the confidentiality. Where there is a problem in finding a suitable cryptographic product for the protection of a transmission channel (e.g. approved implementation of TLS/SSL) the application designer may chose to encrypt the data off-line before sending them to the web application and vice versa.	all levels of risk
Web page integrity and authenticity protection	The integrity and authenticity of webpages must be protected (e.g. by mandatory use of TLS/SSL). This applies to the webpage's content and all elements displayed or used within the webpage (e.g. pictures, java scripts, data downloaded by applets or ActiveX components etc.) as well as any other documents returned to the clients. This protection is critical for all pages displayed within a session.	integrity and confidentiality risks at level standard or above
Access control	Strong access control must be provided for all EUCI handled by the web application. Access control must not rely solely on the secrecy of URLs identifying the protected resource. Use of white lists instead of blacklists (or combination of both techniques) is recommended. For all risks above standard level every access to the web application should be controlled.	all risks linked to confidentiality and integrity
Accountability	Each interaction with the system should be recorded. The information recorded should include at least, the following items: time, data/service accessed, account used, identification of a workstation, URL accessed. Appropriate log management techniques should be applied to prevent the logs from unauthorised modification or destruction. The logging system should be preferably centralised as described in the "Encapsulation and centralisation of security services" paragraph. User should be clearly informed about legal consequences of prohibited actions.	all risks at the level standard or above
Multifactor authentication	Authentication based on two (or more) types of credentials (e.g. knowledge of a secret and possession of a certain object) must be required to access EUCI or other sensitive data.	confidentiality and integrity risks at the level enhanced and above
Encapsulation and centralisation of security services	All the security services must be, where possible, encapsulated inside reusable components and centralised on the server side. This includes services like authentication and authorisation, identity management, logging, monitoring etc.	all risks at the level standard or above

Duplication of services and/or data, use of failover/ switchover infrastructure	The technique of duplication and/or failover infrastructure must be used for data and services having high availability needs. The time needed to switch to the failover system has to be determined by business users and tested in practice.	availability risk high
Separation of the identity management system	The web application should avoid sharing an identity management system (if any is used) with any other IT system that is not intended for the same users' group. For example, it should not use the identity management system provided by the operating system nor the relational database if access to the operating system or database is not intended for web application users, especially when the operating system or the database is used by other systems dedicated for different users' communities (on the contrary, the identity management offered by an operating system should be used if possible, when users of the web application are allowed to access operating system services (e.g. file sharing)). Use of specialised identity management systems designed for a distributed environment is recommended. The identity management systems must not be shared by many web applications accredited to different levels.	all risks at the level standard or above
Protection of back office components	Back-end components connected to the web server cannot be accessible by clients and, where possible, they must be hidden so that no information about the type and number of components is available on the client side.	all risk at the level standard or above
Removal of development and test data	Prior to release, all development toolkits, debugging routines, programmer comments (visible in the executable files) and test data must be removed from the CIS.	all levels of risk
Use of Hardware Security Modules	Use of Hardware Security Modules (HSM) on the server side is recommended for protecting classified information.	confidentiality and availability risks at the level enhanced or above
Protection of configuration settings	Configuration files must not be stored in the space accessible by the clients. Storing any plain text secrets in configuration files must be avoided and default secrets (passwords) must be changed in compliance with the Information Assurance Guidelines on reusable user-generated passwords [8].	all risks at level standard or above

Support for asynchronous processing	Asynchronous processing may be considered when using web applications with high availability and scalability needs.	availability risk high
Use of common character set	The character set (e.g. UTF-8) must be decided during the design phase and all relevant components must be designed with explicit support of chosen encoding.	all risks at level standard and above
Hazards and synchronisation problems	The design of the web application must securely handle the problems coming from the synchronisation of processes, in particular it has to address the following aspects: - receiving the same requests many times (e.g. refreshing a web page in a web browser may cause unwanted multiplication of actions like sending an email etc); - competition for resources between many clients (e.g. parallel modification of the same EUCI by two different clients may cause its loss or corruption); - timeouts and sessions expiration (e.g. timeout during a long lasting operation may leave a system in an unsecure state); - deadlocks - synchronisation of processes will avoid the possibility of a deadlock (a process is blocked by another one which respectively is blocked by the former); - starvation - a request may not be processed due to an unfair policy of resources management or request queuing; - data sharing - in most operating systems the thread shares the memory with the process that started it (therefore it can modify the data of the parent process), whereas the process receives a copy of all the data from parent the process.	availability risks basic or above

Annex 3

IMPLEMENTATION LEVEL COUNTERMEASURES

Measure	Description	Risks
Defensive programming	The components of the web application must make use of a defensive programming approach, at least the following techniques should be applied: - the code must be as simple and clear as possible (no tricks, version-dependent, or platform-dependent techniques must be used, (e.g. use of static variables, assumptions about memory organisation etc)); - the developer should expect incorrect input both of unintentional and malicious nature; - the developer should expect and account for errors even in unlikely situations (like memory allocation); - critical security conditions are to be checked and enforced (e.g. whether the connection is encrypted); - the developer should handle potential hazards (e.g. changing 64bits variable by two threads simultaneously, or generation of unique incremental numbers).	all levels of risk
Not leaving backdoors or workarounds	The source code of a web application in any stage of the development must not contain any backdoors or any other mechanisms bypassing security controls.	all levels of risk
Integrity of the code on the server side	The integrity of the executable code must be controlled (e.g. by digital signature)[8].	all risks at the level enhanced and above

[8] This applies recursively to all 3rd party libraries and components.

Validation of input on the server side	No validation of the input data on the client side is trusted. Each component on the server side must consider input data as malicious and must therefore use a positive validation pattern. This means, only correct data are accepted (in contrast to the strategy that known wrong input is rejected). For example the component accepting an email address would check the size of the data and the format and only correct email addresses would be further processed. Use of regular expressions might be useful to describe the correct input. The canonicalization of data is recommended for all calls to external parsers or services (e.g. conversion of relative paths to absolute paths).	all levels of risk
Handling of suspicious actions	Even though Intrusion Detection Systems (IDS) are commonly used, the best place to distinguish between a "normal" and suspicious behaviour of the system is the web application itself. The code on the server side must therefore securely handle suspicious behaviour of the clients at minimum the following situations must be handled: - detection of multiple failed login attempts; - detection of parallel sessions for the same user account; - detection of session id guessing; - detection of downloads or uploads of extreme amount of data (e.g. many gigabytes of data accessed by a single user); - suspicious data input - use of large, abnormal inputs or non alphanumeric characters; - variations from the agreed protocols and data flows (e.g. bypassing certain steps or repetitions); - detection of request forgery (e.g. by checking referrer header or preferably, by the use of techniques preventing such attacks like those mentioned below in the "Protection against request forgery" paragraph).	all risks at the level enhanced or above

Anti-automation protection	The web application must not permit an attacker to automate a process that was originally designed to be performed only in a manual fashion, i.e. by a human web user (e.g. by the use of unpredictable URLs and parameters, use of CAPTCHA etc).	all risks at the level enhanced and above
Protection of information about the server environment	Information about technologies, versions of programs, operating systems on the server side should be hidden from the clients as much as possible[9]. This includes the following elements: - URL patterns - meaningful standard extensions (e.g. .php , .jsp, .aspx, .dll etc) must not be used; - standard look and feel of the system discloses minimum details about the technology used; - technology logos or licensing[10] - information about underlying technologies must be hidden (e.g. tags "GENERATED BY"); - standard files or services- standard resources located on the server side that would help to identify the technology (e.g. standard administration console , standard folders, or folder browsing services) must not be accessible by the user; - error messages - standard error messages must be trapped as described in "Handling of errors and exceptions"; - operating system information - information about the underlying operating system on the server side (e.g. statistical information coming from TCP sequence numbers or standard services) must be hidden from clients.	all risks at the level standard and above
Secure access to internal objects[11]	The use of URL/form parameters in referring to internal services must be controlled in the following way: - the access to internal objects must be always controlled by checking user's permission to access such object; - the reference to internal objects should be preferably indirect (the parameters controlled by the users should be converted on the server side and never processed without validation) and neutral (e.g. information shown to the user must not reveal the technical details about the object like: id in a database, database relation name, or name of a user).	all levels of risk

[9] If the technology used does not allow for hiding such details the technique of forwarding or rewriting can be used.
[10] Product procurement contracts must be written so as to permit this.
[11] The internal object is a component used internally by a web application, defined in the specific set of technologies used to build the system (e.g. a relation in a database, a java bean object or plain java object (POJO) , COM/OLE object, file or host name etc)

Protection of temporary data storages	If a web application stores (on the server or client side) any EUCI or any sensitive data in a temporary storage the confidentiality and integrity of such data must be protected. This includes using encryption and avoiding of predictable file names.	confidentiality and integrity risks high
Protection against reply attacks	If relevant, the implementation must guarantee the freshness of the requests. That means assuring that a request could be processed by the server only once[12].	confidentiality and integrity risks at level standard or above
Protection against request forgery	If relevant, the implementation must guarantee that the user is not maliciously deceived into sending a request or (s)he does not do it by an accident[13].	all levels of confidentiality and integrity risks
Session handling	The session handling should be preferably solved by the use of cookies[14]. The cookies which contain authenticated session tokens must have their domain and path set to an appropriately restrictive value for that site and an expiration date cannot be set[15]. It is recommended to use HTTPOnly cookies and to set "secure" flag. The session id cannot be stored in logs (also during exception handling) and all communications within the session have to be always carried via protected channel (e.g. SSL). Each generated web page that requires authentication must have a "logout" functionality. The mechanism of session id generation must assure appropriate randomisation[16] and session id must always change for new sessions (also during re-authentication). Each request within the session should be protected against reply and forgery (see above).	all levels of confidentiality and integrity risks
Session handling in higher risk systems	It is recommended to have a configurable timeout after which a session invalidates itself[17]. The system must block the use of the same session from two different sources (e.g. two different IP numbers or within two different SSL tunnels). Only web browsers with secure cookies handling can be used in the web application (to be enforced by the server and the local administrator).	confidentiality and integrity risk at the level enhanced and above

[12] There are many techniques to realise this, for example the server may implement the 'synchronizer token' design pattern [14]: the server includes a random number (token) in each returned HTML page, expecting this number in a next request (rejecting all mismatching requests). The confidentiality of such token must be protected.

[13] Among other techniques, it can be achieved by the 'synchronizer token pattern' mentioned above, asking user for a confirmation of the request or by the re-authentication of each request.

[14] If parameters (e.g. session id) or other mechanism are used they should ensure the same level of protection as described in this paragraph for cookies.

[15] The web browser is not supposed to store it.

[16] Having enough entropy so that guessing of session id would be detected by the IDS mechanisms.

[17] The timeout should invalidate the session on a web application regardless of the protocol, technologies or architecture used (SSO, proxies, client type).

Use of hash functions and salts to store secret credentials[18]	If authentication is based on a secret sharing technique, it is recommended that all such secrets are not stored in a plain text and instead that they are concatenated with a random string (a salt) and then stored as a hash generated by an approved cryptographic product. The hashing function may be additionally slowed down (e.g. by multiple calls of the hashing function). Use of Password-Based Key Derivation Function 2 (PBKDF2)[15] or equivalent techniques (if permitted by the IASP2 [2]) is recommended.	confidentiality risk basic or above
Secure handling of files	Files should only be included when necessary. Especially the usage of files stored in remote locations (such as CSS or WSDL) should be secured by using a whitelist. The names for the uploaded files and folders should be chosen on a random and not predictable basis. The path where files are stored should not be modifiable by the user (e.g. to prevent directory traversal). Uploaded files should be checked for plausibility and malware. Uploads should be restricted regarding size, file types and number of uploads. Uploaded files must not be executable on the server. For high risks, uploaded files (especially pictures) should be converted (to prevent malware distribution).	all risks at the level standard or above
Handling of errors and exceptions	The errors and exceptions have to be handled in the programme on the server side. The handler code has to make sure that the system is not left in an unsecure state[19] and that the information returned to the client does not reveal any information which could be useful to an attacker, including configuration entries, relation names, stack traces or software versions. The users should only receive technically neutral messages. For example a failed login attempt should not explain detailed reasons of the failure (e.g. it should not communicate that the password is incorrect or that the account does not exist). No error information from any back-end component should be passed directly to the client.	all risks at the level enhanced or above
Initialisation and deinitialisation of variables	The variables used inside the web applications must be initialised at the beginning of the software structure they belong to and all variables storing sensitive data (passwords, keys, EUCI) should be zeroed[20] before exiting the structure. No assumption about standard initial values or automatic deinitialisation should be made.	confidentiality risks standard or above
Secure memory management	It is recommended that technologies with direct memory management for the development of web applications (e.g. C/C++, PASCAL/Delphi, assemblers etc) are avoided. This includes calling such technologies via various APIs (e.g. calling to operating system routines or access to computation libraries) especially when the data passed to such technologies is controlled by the end users.	all risks at the level of enhanced or above

[18] This technique will prevent from attacks using pre-calculated hashes (rainbow tables) and brute force attacks.
[19] "Fail in known state"
[20] The code should effectively zero the content of the data, not a reference only

Secure redirection	The redirection (use of status codes 300,301,302,303,307 in response) in the web applications should be used very carefully and avoided when possible. If used, the redirection should not be controlled by the URL parameters and the programme generating redirection response to the web browser should perform solid access control.	all risks at the level standard or above
Encoding of input data passed to other components	The input from users passed to another service/component, must be appropriately encoded/escaped[21] (e.g. SQL, LDAP queries, calls to rule engines, interpreters etc). All special characters within the called service must be handled and escaped on the server side[22].	all level of risks
Filtering /encoding outputs returned to users	The data returned to the users inside the HTML documents must be encoded and escaped to make sure that it does not contain any interpretable parts (no HTML tags or scripting languages[23]). This applies to the following elements: - CSS documents; - data elements of HTML documents; - attributes (of HTML or XML tags); - scripts.	all levels of risk
Avoiding use of HTTP GET method for sending EUCI	The use of the GET method to send a request makes all the data visible in the web browser URL text field and it is likely to be visible in the web server logs. Therefore EUCI must be sent using the POST method.	all confidentiality risks
Disabling "auto complete"	Although not a part of a standard HTML, the autocomplete attribute is respected by most browsers. It is therefore recommended to disable it in all HTML forms to avoid any data storage on the client side.	all levels of risk
Use of third party components	Third party components (e.g. open source or COTS) should not be used without verification (see IV.4 Testing and Verification Guidelines)	all levels of risk

[21] "escaping" means converting input or output data (typically strings) into a form compliant to a grammar of a particular formal language (e.g. programming or query language). It is often achieved by the use of special characters called 'escape characters'. The objective of escaping is to assure correct interpretation of the data in a particular context. For example all '<' characters will be converted into '<' phrase in HTML , all apostrophes (" ' ") will be preceded by an additional apostrophe in many SQL languages etc.

[22] This problem can be addressed at the design phase of the web application by introduction of components responsible (among others) for the conversion of input data, e.g. Value Object pattern[14], use of stored procedure to access database (instead of using SQL directly), use of object-relational-mapping layer like Hibernate, Java Entity Beans or TopLink.

[23] Unless deliberately send as an interpretable programme.

Annex 4

EXAMPLE SECURITY VERIFICATION LEVELS

Level	Security verification requirements	
basic	Load test	The web application is checked against the expected load. The load characteristic (the limits and dependency between the load and response time) of a web application could also be determined.
	Automated web application source code analysis	The source code of the web application is automatically searched for vulnerability patterns (for the web application only).
	Automated dynamic web application analysis	Automated tools should be used to detect vulnerabilities in a running web application. The findings may be verified by penetration tests.
	Functional test of security controls	The web application is tested to verify the presence and functioning of the security controls (e.g. the presence of access control while accessing EUCI). The tests check only the basic use cases defined during the business analysis.
standard	Manual analysis of the source code of the web application	Manual (with a help of tools) analysis of the source code of a web application and all third party components providing security services (e.g. cryptographic products) verifying the following aspects: - absence of vulnerabilities and implementation mistakes - correct use of security controls
	Security controls bypassing tests	The tests should aim to discover all paths to bypass security controls implemented within the web application. This includes the use cases that are not explicitly defined by business requirements or techniques not predicted or even not allowed in a design phase (e.g. trying to access web application without using SSL or with downgraded version of the SSL, trying to read sensitive data located in a web server space etc).

enhanced	Analysis of intermediate components handling HTTP	Analysis of HTTP implementation in all components involved in handling requests (web browser, web server, proxies). The objective of the analysis is to identify all discrepancies and check their impact on security of the web application (e.g. possibility to inject HTTP headers, splitting of requests etc).
	Manual analysis of all components associated with web application	Manual analysis of the source code, hardware implementation and design documentation of a web application and all third party components (including those not providing security functionalities) associated with the web application (computing libraries, web frameworks etc). The middleware components (web servers, application servers, relational databases, messaging systems, operating systems etc) do not have to be analysed if they received a substantial amount of public scrutiny or if they successfully passed certification officially recognized by the web application owner.
high	Manual analysis of all environment	Manual analysis of the source code, hardware implementation and design documentation of the web application and all components used by the application or associated with it. The analysis of a middleware can be substituted only by officially recognised certificates.

Annex 5

GLOSSARY

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CERT	Computer Emergency Response Team
CIS	Communication and Information System(s)
COM	Component Object Model
COTS	Commercial off-the-shelf
CSRF	Cross Site Request Forgery
CSS	Cascading Style Sheets
DMZ	DeMilitarized Zone
DNS	Domain Name System
DoS	Denial of Service
EAL	Evaluation Assurance Level
EUCI	EU Classified Information
FOSS	Free and open-source software
FTP	File Transfer Protocol
GSC	General Secretariat of the Council
GUI	Graphical User Interface
HSM	Hardware Security Module
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IA	Information Assurance
IDS	Intrusion Detection System
IP	Internet Protocol
ISO	International Organisation for Standardisation
JSON	JavaScript Object Notation
JEE	Java Enterprise Edition
LDAP	Lightweight Directory Access Protocol
OLE	Object Linking and Embedding
OSI	Open System Interconnection
PBKDF	Password-Based Key Derivation Function

PII	Personally Identifiable Information
PIN	Personal Identification Number
PKI	Public Key Infrastructure
POJO	Plain Old Java Object
REST	REpresentational State Transfer
RPC	Remote Procedure Call
SAA	Security Accreditation Authority
SaaS	Software as a Service
SCISM	Sensitive CIS Manager
SCP	Secure CoPy
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SG	Secretary General
SSL	Secure Socket Layer
SSO	Single Sign On
SSRS	System Specific Security Requirement Statement
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
WSDL	Web Service Description Language
XML	eXtensible Markup Language
XSS	Cross Site Scripting