BULLETIN OFFICIEL DES ARMÉES

Édition Chronologique n° 9 du 8 mars 2018

PARTIE PERMANENTE
Administration Centrale

Texte 1

DIRECTIVE N° 40/DEF/DGSIC
portant sur le développement des applications informatiques et des logiciels robustes du ministère de la défense.

Du *17 mai 2017*

D I R E C T I O N G É N É R A L E D E S S Y S T È M E S D ' I N F O R M A T I O N E T D E COMMUNICATION : *sous-direction de la sécurité des systèmes d'information.*

DIRECTIVE N° 40/DEF/DGSIC portant sur le développement des applications informatiques et des logiciels robustes du ministère de la défense.

Du 17 mai 2017

NOR A R M D 1 7 5 2 5 3 9 X

Référence :

Instruction ministérielle n° 7326-2/DEF/CAB du 7 août 2014 (n.i. BO).

Pièce(s) Jointe(s) :

Quatre annexes et trois appendices.

Classement dans l'édition méthodique : BOEM 160.3

Référence de publication : BOC n° 9 du 8 mars 2018, texte 1.

SOMMAIRE

5.1. Exigences relatives au développement.

5.2. Habilitation des acteurs du développement informatique.

5.3. Exigences de sécurité dans le cadre du développement informatique.

5.4. Documentation de sécurité du développement informatique.

5.5. Suivi et gestion de la sécurité dans le cadre du développement informatique.

5.6. Méthodologie de développement informatique.

5.7. Choix du langage.

5.8. Outils de développement.

5.9. Exigences de sécurité vis-à-vis des tiers.

6. CONCEPTION ET ARCHITECTURE.

6.1. Identification et gestion des menaces.

6.2. Architecture applicative générale.

6.3. Principes de conception.

6.4. Cloisonnement.

6.5. Logique applicative.

6.6. Authentification.

6.7. Gestion des erreurs.

6.8. Gestion des traces.

6.9. Gestion des fichiers.

6.10. Données de test et de débogage.

7. IMPLÉMENTATION ET PROGRAMMATION.

7.1. Conventions de codage.

7.2. Vérification des données.

7.3. Interactions avec l'environnement.

7.4. Protection des données sensibles.

7.5. Crypographie.

7.6. Gestion des droits et privilèges.

7.7. Gestion des ressources informatiques.

1. PRÉSENTATION GÉNÉRALE ET GUIDE D'USAGE.

1.1. **Contexte.**

Une approche périmétrique (chiffrement, pare-feu, détection d'intrusion, etc.) de la sécurité des systèmes d'information n'apporte pas de réponse satisfaisante aux menaces visant les couches applicatives : injections SQL (1), injections de codes malveillants, etc.

L'agence nationale de la sécurité des systèmes d'information fait régulièrement état d'attaques ou de vulnérabilités de ce type permettant une intrusion pouvant conduire à un vol massif de données.

C'est dans ce contexte de lutte contre ces menaces que s'inscrivent la sécurité applicative et la directive sur le développement robuste et sécurisé.

1.2. **Présentation du document.**

Ce document s'intègre dans les missions de la direction générale des systèmes d'information et de communication (DGSIC), aux termes du décret n° 2006-497 du 2 mai 2006 modifié, portant création de la direction générale des systèmes d'information et de communication et fixant l'organisation des systèmes d'information et de communications du ministère de la défense.

Il s'agit d'un ensemble d'exigences et de règles permettant de réaliser au profit du ministère de la défense des développements d'applications informatiques ou des logiciels robustes capables de fonctionner correctement en présence d'événements inattendus (valeurs hors normes ou mal formatées, etc.).

Les règles *infra* ont pour objectif final de renforcer la robustesse des applications informatiques ou des logiciels en phase d'utilisation (au sens de [IM 2008] ou [IG 125-1516]) ou de production. Cette robustesse participe à la sécurité de ces applications informatiques ou de ces logiciels.

1.3. **Périmètre du document.**

Cette directive permettra d'établir un caractère normatif (ou référentiel) applicable à tous les développements informatiques tant internes qu'externes faits au profit du ministère de la défense.

Elle définit des principes, des méthodes, des règles voire des bonnes pratiques afin d'améliorer la robustesse et la confiance dans la sécurité des applications informatiques et des logiciels.

Cependant, elle n'a pas vocation à se substituer aux normes qualités ou aux standards déjà existants dans certains domaines notamment en matière de sûreté (logiciels aéronautiques embarqués par exemple) mais à les compléter le cas échéant.

Les règles s'appliqueront dans le cadre des activités de réalisation d'un développement informatique (application informatique ou logiciel) telles que définies dans les instructions ministérielles, à savoir :

- les activités de réalisation, d'intégration et de déploiement telles que décrites succinctement au paragraphe 2 de [IM 2007] ;

- les activités de la phase de réalisation dans le cadre d'une démarche de programme d'armement telles que décrite dans [IG 125-1516] ;

- les phases et les disciplines (construction et transition) de la méthode PHARE (2) correspondant aux activités entre les jalons J3 et J4 tels que décrits dans [IM 2008] ;

- les activités de la phase de réalisation telle que décrite dans [GUIDE 07].

Le vocable « développement informatique » recouvre de manière non exhaustive les notions suivantes :

- l'élaboration d'applications informatiques et de logiciels ;

- l'élaboration de modules informatiques constituants d'un système d'information ou d'un programme d'armement de plus grande ampleur ;

- l'élaboration de fichiers d'instructions (3) (ou de commandes) informatiques visant à automatiser certaines tâches récurrentes dans les processus de développement et de codage, d'intégration et de

déploiement de tests et d'évaluation de sécurité ;

- l'élaboration de fichiers d'instructions (ou de commandes) informatiques visant à automatiser certaines tâches récurrentes dans le cadre du développement agile ;

- etc.

1.4. **Acteurs concernés.**

Ce document s'adresse principalement aux acteurs participant au développement d'applications et de logiciels en particulier :

- les membres de l'équipe de projet assignés au suivi des développements informatiques telle que décrite dans les instructions ministérielles en phase conduite de projet ;

- les développeurs responsables du codage et de la programmation ;

- les acteurs responsables de la conception et de la modélisation d'applications informatiques ou de logiciels ;

- les acteurs responsables de l'intégration et du déploiement d'applications informatiques ou de logiciels ;

- les acteurs responsables des tests et des évaluations de sécurité d'applications informatiques ou de logiciels.

1.5. **Applicabilité.**

Ce document doit être décliné et appliqué par les entités du ministère impliquées dans le développement d'applications informatiques ou de logiciels.

De plus, il constitue un référentiel pour l'élaboration de clauses contractuelles pour les prestations de développement externalisé d'applications informatiques ou de logiciels au profit du ministère de la défense.

1.6. **Processus de gestion du document.**

Ce document fait l'objet d'un cycle de rédaction, vérification et approbation des directives et autres documents officiels en conformité avec les usages en cours au sein de la DGSIC.

Il peut être amendé ou complété à l'issue d'un processus de retour d'expérience formalisé à travers la directive RETEX (cf. [DIR RETEX]).

1.7. **Niveaux de préconisation.**

Les règles définies dans ce document ont différents niveaux de préconisation conformes à la [RFC 2119] :

OBLIGATOIRE.	Ce niveau de préconisation signifie que la règle édictée indique une exigence absolue de la directive.
RECOMMANDÉ.	Ce niveau de préconisation signifie qu'il peut exister des raisons valables, dans des circonstances particulières, pour ignorer la règle édictée, mais les conséquences doivent être comprises et pesées soigneusement avant de choisir une voie différente.
DÉCONSEILLÉ.	Ce niveau de préconisation signifie que la règle édictée indique une prohibition qu'il est toutefois possible, dans des circonstances particulières, de ne pas suivre, mais les conséquences doivent être comprises et le cas soigneusement pesé.
INTERDIT.	Ce niveau de préconisation signifie que la règle édictée indique une prohibition absolue de la directive.
CONSEILLÉ.	

	Ce niveau de préconisation signifie que la règle édictée est une bonne pratique. Il n'est pas nécessaire d'instruire une dérogation lorsqu'elle n'est pas respectée.

2. CADRE DOCUMENTAIRE.

2.1. **Documents applicables.**

[RGI]	Référentiel Général d'Interopérabilité, version 2.0 du 20 avril 2016. http://references.modernisation.gouv.fr/interoperabilite
[RGS]	Référentiel Général de Sécurité, version 2.0 du 13 juin 2014. http://www.ssi.gouv.fr/rgs
[IGI 1300]	Instruction générale interministérielle n° 1300 du 30 novembre 2011 sur la protection du secret de la défense nationale. https://www.ssi.gouv.fr/administration/reglementation/protection-des-systemesinformations/
[PSSIE]	Politique de Sécurité des Systèmes d'Information de l'Etat : circulaire du premier ministre n° 5725/SG du 17 juillet 2014. http://circulaire.legifrance.gouv.fr/pdf/2014/08/cir_38641.pdf
[PSSI-M]	Instruction ministérielle n° 7326/DEF/CAB du 7 aout 2014 relative à la politique de sécurité des systèmes d'information du ministère de la défense. http://synoptic.intradef.gouv.fr/sites/default/files/20140807_np_cab_im-7326_pssimindef.pdf
[PSSI-M-T]	Instruction ministérielle n° 7326-2/DEF/CAB relative au volet technique de la politique de sécurité des systèmes d'information du ministère de la défense. http://synoptic.intradef.gouv.fr/sites/default/files/160128_pssi_m_t_dr.pdf
[II 901]	Instruction interministérielle n° 901/SGDSN/ANSSI du 28 janvier 2015 relative à la protection des systèmes sensibles. http://synoptic.intradef.gouv.fr/sites/default/files/ii901sensible_signee.pdf
[IM 900]	Instruction ministérielle n° 900/DEF/CAB/-- du 26 janvier 2012 relative à la protection du secret de la défense nationale au sein du ministère de la défense. http://synoptic.intradef.gouv.fr/sites/default/files/55308737d01.pdf
[IM 2007]	Instruction ministérielle n° 2007/DEF/DGSIC du 24 mars 2014 relative au pilotage d'un système d'information outillant les processus de fonctionnement du ministère, pendant tout le cycle de vie jusqu'au retrait de service. http://synoptic.intradef.gouv.fr/sites/default/files/20140324_np_dgsic-cs_200-im2007.pdf
[IM 2008]	Instruction ministérielle n° 2008/DEF/DGSIC du 10 juillet 2013 fixant les modalités d'approbation et de suivi des SIC. http://synoptic.intradef.gouv.fr/sites/default/files/20130710_im_2008_fixant _les_m odalites_approbation_et_suivi_des_sic_0.pdf
[IG 125-1516]	Instruction générale n° 125/EMA-1516/DGA du 26 mars 2010 relative au déroulement et à la conduite des opérations d'armement.
[PSI]	Politique du système d'information du ministère de la défense version 6 du 31 janvier 2013. http://synoptic.intradef.gouv.fr/sites/default/files/20140207_np_dgsic-sds_90lettre-politique_du_si.pdf
[DIR SGBDR]	Directive n° 5/DEF/DGSIC du 7 avril 2007 portant sur les systèmes de gestion de bases de données relationnelle (SGBDR). http://synoptic.intradef.gouv.fr/sites/default/files/20080407_np_dgsic_05-directivesystemes-gestion-base-de-donnee-relationnelle.pdf
[DIR INTERAPPLI]	Directive n° 19/DEF/DGSIC du 24 aout 2011 portant sur les échanges inter-applicatifs du ministère de la défense. http://synoptic.intradef.gouv.fr/sites/default/files/20110824_np_dgsic_19-directiveechanges-iterapplicatifs-mindef.pdf
[DIR SYNC TPS]	Directive n° 24/DEF/DGSIC du 9 mars 2012 portant sur le service de synchronisation horaire au standard IP.

	http://synoptic.intradef.gouv.fr/sites/default/files/20120309_np_dgsic_24-directiveservice-synchronisation-horaire-standart-internet-protocol.pdf
[DIR HSI]	Directive n° 27/DEF/DGSIC du 24 janvier 2013 relative à l'homologation des systèmes d'information du ministère de la défense. http://synoptic.intradef.gouv.fr/sites/default/files/20130124_np_dgsic_sdssi_directi ve_27_homologation _si_mindef.pdf
[DIR TRACES]	Directive n° 29/DEF/DGSIC du 12 novembre 2013 portant sur les traces et leur gestion au sein du ministère de la défense. http://synoptic.intradef.gouv.fr/sites/default/files/20131112_np_29_dgsic_directive _traces-et-gestion-au-sein-du-mindef.pdf
[DIR SEC.HEB]	Directive n° 32/DEF/DGSIC du 11 mars 2014 portant sur la sécurité de l'hébergement des SI au sein du ministère. http://synoptic.intradef.gouv.fr/sites/default/files/20140311_np_dgsic-sdssi_32-dirdirective-securite-hebergement.pdf
[DIR RETEX]	Directive n° 33/DEF/DGSIC/NP du 5 février 2015 relative au retour d'expérience en cybersécurité au sein du ministère de la défense. http://synoptic.intradef.gouv.fr/sites/default/files/20150205_np_dgsic-sdssibmr_033-directive-retex-ed1.pdf
[DIR S.INDUS]	Directive n° 39/DEF/DGSIC/DR du 1er juillet 2016 portant sur la sécurité des systèmes industriels.
[NOTE DCP]	Note n° 598/DEF/DGSIC/SDSSI/DR du 25 octobre 2016 relative aux recommandations en matière de protection de donnés à caractère personnel.
[GUIDE 07]	Guide n° 7/DEF/DGSIC/NP du 21 novembre 2014 relatif à l'intégration de la sécurité des systèmes d'information dans les projets de systèmes d'information.
[GUIDE 11]	Guide n° 11/DEF/DGSIC/NP du 7 octobre 2016 relatif au document des projets et des systèmes d'information régis par les instructions 2007 et 2008.
[DIRISI 61]	Directive n° 61/DEF/DIRISI/SICO du 10 septembre 2014 relative à la constitution et au rôle du comité de décision des mises en production. http://www.dirisi.defense.gouv.fr/IMG/pdf/Directive_no61.pdf
[GUIDES CODAGE]	Cadres techniques de développement spécifiques édités par la DGSIC. Guides et règles de codage pour l'implémentation de logiciels édités par DGA/MI. http://synoptic.intradef.gouv.fr/node/703/documents
[CCT]	Cadre de cohérence technique des systèmes d'information et de communication du ministère de la défense. http://synoptic.intradef.gouv.fr/sites/default/files/20160728_dr_dgsic_cctmindef_v2_0.pdf
[GT TESTS]	Document de travail du groupe de travail sur les tests piloté par le Centre d'analyse des systèmes d'information de la défense.

2.2. **Normes et standards applicables.**

[RFC 2119] Mots-clés pour niveaux d'obligation.

2.3. **Autres documents et sites.**

[SynopTIC]	Site de référence des SIC du ministère. http://synoptic.intradef.gouv.fr/ (intranet)
[GUIDES CONFIG]	Guides de configuration et de sécurité validés par le ministère de la Défense. http://synoptic.intradef.gouv.fr/guides-de-configuration-et-de-securite (intranet)

[GUIDES ANSSI]	Guides et recommandations sur la sécurité des développements JAVA et des sites web publiés par l'Agence nationale de la sécurité des française (ANSSI). https://www.ssi.gouv.fr/bonnes-pratiques (internet)
[GUIDE ENISA]	Guide sur la sécurité et la confidentialité du Web 2.0 de l'Agence européenne pour la sécurité des réseaux et de l'information (ENISA). https://www.enisa.europa.eu/publications/archive/reportfr/at_download/fullReport (internet)
[GUIDES OWASP]	Guides de sécurisation des développements du programme libre de sécurité des applications informatiques Web (OWASP). https://www.owasp.org (internet)
[GUIDES NIST]	Guides (série 800-XXX) de sécurisation des développements des systèmes d'information de l'Institut national des standards et technologies américain (NIST). http://csrc.nist.gov/publications (internet)
[GUIDES DISA]	Guides de sécurisation des développements informatiques de l'Agence des systèmes d'information de la Défense américaine (DISA). http://iase.disa.mil/stigs/Documents/Forms/AllItems.aspx (internet)
[GUIDES SANS]	Guides de sécurisation des développements informatiques édités par l'institut SANS. https://www.sans.org/reading-room/whitepapers/securecode (internet)
[SDL MICROSOFT]	Cycle de vie du développement sécurisé de Microsoft. http://www.microsoft.com/sdl (internet)

3. GESTION DES DÉROGATIONS.

La présente directive est applicable à tous les développements (internes ou externes) d'applications informatiques ou de logiciels au profit du ministère de la défense. Cependant, des contraintes inhérentes à ces développements peuvent engendrer des difficultés dans l'application de certaines règles.

Dès lors, toute dérogation à l'application des règles techniques ou organisationnelles devra être documentée et traitée dans le cadre de la recette, et le cas échéant de l'homologation de sécurité du système d'information (cf. [DIR HSI]) : les risques induits par cette dérogation devront être acceptés dans le cadre des travaux du groupe de travail de sécurité d'applications informatiques ou de logiciels associé aux développements informatiques sous la responsabilité du chef de projet en concertation avec le RSSI (Responsable de la sécurité des systèmes d'information).

Toutes les dérogations devront être listées et insérées dans le dossier d'homologation, et les documentations exigées dans le cadre du développement informatique d'applications informatiques ou de logiciels (cf. [GUIDE 11]).

4. LISIBILITÉ DU DOCUMENT.

Les règles, et recommandations, sont regroupées par domaine d'activités-types rencontrées dans les développements informatiques :

- les activités-types de spécification et de méthodologie ;
- les activités-types de conception et d'architecture ;
- les activités-types d'implémentation et de programmation (codage) ;
- les activités-types d'intégration et de déploiement ;
- les activités-types de tests et d'évaluation de sécurité.

La typographie utilisée pour distinguer une note d'information, une règle technique ou organisationnelle est donnée *infra*.

Ce cadre indique une note d'information plus détaillée.

DEV-RT-NN énumère une règle technique, indique son niveau de préconisation et sa description.

DEV-RO-NN énumère une règle organisationnelle, indique son niveau de préconisation et sa description.

Pour la suite, on utilisera les termes :

- « Application » pour désigner une application informatique, un logiciel, un module informatique ou toute suite logique d'instruction informatique visant à automatiser toute tâche ou tout traitement de données ;

- « Développement » pour désigner le développement informatique.

5. SPÉCIFICATION ET MÉTHODOLOGIE.

Les règles *infra* en matière de spécification et de méthodologie sont applicables quelle que soit la nature des applications développées. Elles peuvent être adaptées au contexte.

5.1. **Exigences relatives au développement.**

DEV-RO-1 Il est obligatoire que les exigences (sécurité, techniques, conceptuelles, etc.) relatives au développement soient formalisées au sein de documents spécifiques conformes aux instructions ministérielles.

Les exigences de sécurité seront formalisées :

- au sein du « plan contractuel de sécurité » (cf. [IGI 1300]) pour les développements classifiés ;

- au sein du « plan d'assurance de sécurité » (cf. [NOTE DCP]) ou le cas échéant du plan étatique de management pour les développements sensibles ;

- au sein d'un plan de sécurité sous une forme adaptée pour les développements « non protégés ».

Les autres exigences seront formalisées au sein d'un fonds documentaire technique conforme aux instructions ministérielles régissant les développements informatiques concernés.

5.2. **Habilitation des acteurs du développement informatique.**

DEV-RO-2 Il est obligatoire que tous les acteurs impliqués aient fait l'objet d'une habilitation en cohérence avec la confidentialité ou la sensibilité du développement. Elle est indiquée au sein des clauses contractuelles de sécurité (cf. DEV-RO-1).

Cette exigence concerne tant les personnes morales (entreprises, prestataires, etc.) que les personnes physiques (développeurs, chef de projet, etc.), que ce soient des développements internes ou externes au ministère.

Les besoins en habilitation ou en criblage élémentaire devront être clairement précisés dans l'annexe de sécurité jointe au contrat.

DEV-RO-3 Il est recommandé que tous les acteurs ministériels impliqués dans un développement de traitement automatisé de données à caractère personnel aient fait l'objet d'une habilitation confidentiel défense.

En fonction du contexte et sous la responsabilité des contractants ministériels, cette recommandation peut être applicable dans le cadre des prestations externalisées.

5.3. **Exigences de sécurité dans le cadre du développement informatique.**

DEV-RO-4 Il est obligatoire d'identifier et de hiérarchiser les exigences de sécurité applicables au développement de l'application. Les arbitrages et les évolutions seront formalisés et tracés dans la documentation technique de réalisation ou le dossier de sécurité et d'homologation.

Les exigences de sécurité applicables au développement sont constituées des éléments suivants :

- la politique de sécurité des systèmes d'information applicable ;

- la sensibilité (confidentialité, intégrité, disponibilité) des données traitées et traitantes (code source, paramétrages, etc.) ;

- l'analyse de risques exprimant les besoins et identifiant les objectifs de sécurité ;

- les exigences de sécurité issues de la fiche d'expression rationnelle des objectifs de sécurité (FEROS) ;

- les éventuelles menaces prises en compte au cours de l'analyse de risque ;

- les éventuelles réglementations applicables (protection du secret, données personnelles, etc.). Ces exigences de sécurité sont exprimées par les commanditaires (autorités « clientes » au sens de [IM 2007]) aux équipes de projet (ou aux prestataires) concernées en amont du développement.

Elles sont accompagnées d'une métrique convenue permettant de les hiérarchiser, et d'effectuer des arbitrages le cas échéant, en termes d'impacts métier.

DEV-RO-5 Il est obligatoire d'identifier et de définir l'ensemble des rôles (métier et technique) et des privilèges strictement nécessaires et suffisants, pour le développement et la mise en œuvre de l'application.

Les rôles et privilèges associés aux acteurs du développement, aux utilisateurs et aux exploitants de l'application tant sur le plan fonctionnel (métier), que sur le plan technique (MOE [4], équipe projet, administrateur, opérateur sauvegarde, etc.) doivent être formalisés au sein de la documentation technique de réalisation (dossier d'architecture, etc.) ou du dossier de sécurité et d'homologation (dossier d'administration et d'utilisation, etc.).

DEV-RO-6 Il est obligatoire d'identifier et de définir les responsabilités (métier, technique, sécurité) des différents acteurs impliqués dans le développement.

Les responsabilités des différents acteurs doivent être formalisées au sein du plan de management (ou équivalent) de projet de développement :

- dans le domaine fonctionnel pour la partie métier ;

- dans le domaine technique pour la partie architecture, programmation, intégration, etc. ; Ø dans le domaine de la vérification (revue de code, tests, etc.).

5.4. **Documentation de sécurité du développement informatique.**

DEV-RO-7 Il est obligatoire de documenter pour le dossier de sécurité et d'homologation les éléments d'information décrivant les arbitrages et les choix décidés pendant les phases de développement en matière de robustesse et de sécurité, y compris les dérogations à la présente directive.

Tout arbitrage ou choix décidé par les équipes de projet en concertation avec les commanditaires concernés, en matière de robustesse et sécurité des développements doit être formalisé et documenté, afin d'enrichir le dossier de sécurité et d'homologation (plan de sécurité, dossier justificatif de choix, dossier de vulnérabilités

résiduelles, etc.).

Ces éléments d'information pourront aussi enrichir la documentation technique (dossier d'architecture, etc.) ou de management du projet tel que décrit dans [GUIDE 11] ou [IG 125-1516].

5.5. **Suivi et gestion de la sécurité dans le cadre du développement informatique.**

DEV-RO-8 Il est obligatoire d'identifier au sein de l'équipe de projet un responsable de la sécurité du système d'information garant de la prise en compte de l'ensemble des questions de sécurité pour le projet de développement.

Intégré au sein de l'équipe de projet de réalisation du développement, le responsable de la sécurité du système d'information est chargé de coordonner les actions internes ou externes garantissant la prise en compte des exigences de sécurité et de leur déclinaison dans toutes les phases du projet : besoin de l'autorité « cliente », prise en compte par les acteurs du développement, etc.

De plus, il est chargé d'animer la démarche d'homologation de l'application en développement.

5.6. **Méthodologie de développement informatique.**

DEV-RO-9 Il est conseillé d'opter pour une méthodologie de développement informatique.

Les méthodes de développement informatique itératives présentent plusieurs intérêts pour la qualité des développements. Elles seront donc préférées si le contexte le permet, dès lors que la robustesse et la sécurité sont prises en compte à chaque itération. L'intérêt de ce type de démarche réside dans :

- la diminution des erreurs et l'amélioration de la qualité, les livrables réalisés étant régulièrement testés et améliorés ;

- la revue des différentes étapes de sécurité permet de vérifier qu'elles sont toujours en cohérence avec le contexte, en particulier lorsque celui-ci évolue au cours du projet ;

- la maîtrise de la complexité quelle que soit la taille du projet : le découpage du projet en phases plus courtes permet d'éviter un effet tunnel ou des étapes longues et complexes à planifier, la fatigue des équipes se fait moins ressentir et la phase de rédaction des spécifications s'en trouve simplifiée.

DEV-RO-10 Il est obligatoire d'intégrer la sécurité dans toutes les phases du développement à partir d'un plan de management de projet ou équivalent.

Chaque phase du développement fera l'objet de préconisations de sécurité adaptées aux menaces et aux risques déterminés en amont du projet. Seront déclinés et décrits :

- dans le plan de management ou équivalent, les modalités de formation et de sensibilisation des acteurs clés ;

- dans le dossier-type de conception et d'architecture du projet, les éléments de sécurisation de l'architecture de l'application ;

- en introduction du dossier-type des codes sources, les règles des [GUIDES CODAGE] applicables au langage de programmation (ou de codage) et au cadriciel (ou *framework*) choisi pour le développement ;

- dans le dossier-type des processus d'intégration, la sécurisation d'une part de l'environnement d'intégration des modules unitaires et d'autre part des processus associés ;

- dans le dossier-type des configurations, le durcissement des configurations de l'application et des plateformes (production, pré-production, développement, etc.) associées ;

- dans le dossier-type et plans de tests, les tests liés à la robustesse et à la sécurité de l'application ;

- dans le dossier-type de déploiement et de mise en œuvre, les modalités de déploiement de l'application.

Les dossiers-types décriront les modalités de traitement et d'évaluation des anomalies détectées sur la robustesse et la sécurité de l'application, afin d'enrichir le dossier des vulnérabilités résiduelles.

DEV-RO-11 Il est recommandé de contrôler l'existence et l'efficacité des mesures de sécurité mises en œuvre lors de chaque phase du développement, et le cas échéant de les réviser afin de les améliorer.

Il s'agit de vérifier l'existence des mesures de sécurité dans chaque phase du développement au travers, par exemple, de la prise en compte des menaces, de revues de code régulières, de recherche de vulnérabilités, de vérification de configuration, etc.

Les modalités pratiques seront décrites dans les dossiers-types ou les plans de test, voire dans le dossier de sécurité et d'homologation. Les résultats et les recettes de tests seront joints afin d'évaluer le niveau de sécurité (via le dossier des vulnérabilités résiduelles) du développement.

DEV-RO-12 Il est obligatoire de définir et de formaliser un suivi rigoureux des dérogations, des bogues, des anomalies et des vulnérabilités tout au long du développement. Ce suivi permettra d'évaluer le niveau de sécurité de l'application à la livraison, et de hiérarchiser les corrections à apporter.

Les outils de suivi de développement utilisés doivent intégrer les aspects bogues, anomalies et vulnérabilités afin de suivre l'état de sécurité de l'application tout au long de son cycle de vie, de la conception au retrait de service. Ils doivent intégrer les critères de causes et de conséquences des bogues de sécurité.

Les critères de causes et de conséquences des bogues de sécurité doivent s'inspirer des sources ouvertes (OWASP, CWE, etc.).

Ces éléments d'information viendront enrichir le dossier de vulnérabilités résiduelles du dossier de sécurité et d'homologation.

5.7. **Choix du langage.**

DEV-RO-13 Il est obligatoire de justifier le choix du langage de développement notamment vis-à-vis de la robustesse et de la sécurité (bibliothèque ou API de sécurité, existence d'options de sécurité des outils de développement, etc.).

Le choix du langage utilisé pour le développement de l'application dépend des ressources en place, de l'existence de documentation adaptée, des impératifs fonctionnels, ainsi que de besoins techniques tels que la gestion des ressources informatiques (mémoire, espace de stockage, réseau, etc.) ou la disponibilité de bibliothèques tierces.

Les critères à prendre en compte, du point de vue de la robustesse et de la sécurité, pour le choix d'un langage sont les suivants :

- la présence de fonctions intrinsèques de sécurité dans le langage et les bibliothèques associées (typage, fonctions sécurisées, etc.) ;

- le niveau de maitrise des équipes techniques (développeurs, exploitants, architectes, etc.) ;

- la gestion de la mémoire (automatique ou déléguée au développeur) ;

- la disponibilité de bibliothèques largement testées et éprouvées ;

- le caractère compilé ou interprété (sensibilité aux injections) ;

- le support de connectivités sécurisées ;

- la disponibilité de documentations liées à la sécurité ;

- l'intégration en environnement de production (interopérabilité avec les services, etc.) ;

- l'existence de suivis pérennes des vulnérabilités (détection, remédiation) et des correctifs (disponibilité, distribution) de l'ensemble des outils de développement (compilateurs, bibliothèques, etc.) associés à ce langage.

Pour aller plus loin : les sites https://wiki.mozilla.org/Bugzilla:Languages et https://www.tiobe.com/tiobeindex/ proposent des comparatifs entre les différents langages de programmation.

5.8. **Outils de développement.**

DEV-RT-1 Il est obligatoire d'utiliser un outil de gestion de version pour gérer et stocker les fichiers (codes sources, scripts, etc.) des applications.

Les outils de gestion de version permettent de stocker l'ensemble des fichiers du projet en conservant la chronologie de toutes les modifications. Cela permet notamment de pouvoir effectuer des retours en arrières afin d'identifier les modifications ayant introduit un dysfonctionnement.

De surcroit, ce type d'outil permet de tracer les modifications afin d'en déterminer l'auteur à chaque changement et de détecter des éventuelles modifications malveillantes.

DEV-RT-2 Il est obligatoire d'utiliser un compilateur (et un éditeur de lien) à jour de correctifs de sécurité et de mettre en œuvre les mécanismes de sécurité disponibles.

Certains langages de développement sont particulièrement sujets aux corruptions de mémoire. C'est pourquoi il est important d'utiliser des compilateurs qui fournissent des mécanismes de défense contre la corruption de mémoire à la compilation ou lors de l'exécution.

Ce type de mécanisme permet de rendre plus difficile l'exécution prédictible d'un exploit, en limitant par exemple le droit en exécution sur certaines pages ou en augmentant le caractère aléatoire de l'adressage mémoire.

DEV-RT-3 Il est recommandé de mettre en œuvre les mécanismes de sécurité offerts par le langage, le *framework* ou l'environnement de développement utilisé.

Certains langages, *frameworks* ou environnement de développement proposent des fonctionnalités ou mécanismes permettant d'éviter ou de réduire les risques liés à la gestion de la mémoire ou au traitement des données. Ces derniers peuvent être utilisés, dans une logique de défense en profondeur, afin de se prémunir contre certains types de vulnérabilités (exemple de mécanisme des compilateurs ou ASLR [5]).

DEV-RT-4 Il est recommandé d'activer l'ensemble des messages d'avertissement et d'erreur lors de la compilation (et de l'édition de lien).

Les messages d'avertissement des compilateurs permettent de signaler des erreurs potentielles.

Les messages d'erreurs permettent de vérifier de manière systématique l'absence de certaines erreurs syntaxiques pouvant engendrer des vulnérabilités intrinsèques pour les applications.

5.9. **Exigences de sécurité vis-à-vis des tiers.**

DEV-RO-14 Il est obligatoire d'inclure des clauses liées à la sécurité dans tout contrat ou convention impliquant un tiers, au travers des documents de sécurité adaptées à la sensibilité des développements (cf. point 5.1.).

Les contrats ou conventions établis pour la réalisation de prestations par des tiers doivent intégrer des documents de sécurité couvrant toutes les phases (au sens des [IM 2007], [IM 2008] et [IG 125-1516]) du développement informatique faisant l'objet d'une externalisation, à savoir :

- la protection des informations mises à disposition et échangées avec les tiers ;

- l'habilitation des tiers impliqués dans les développements informatiques sensibles ou classifiés ;

- la vérification de l'effectivité et l'efficacité des mesures de sécurité contractualisées ;

- les modalités de prise en compte des constats de manquements aux mesures de sécurité contractualisées.

Ces documents complètent ceux décrits au point 5.1.

6. CONCEPTION ET ARCHITECTURE.

Les règles *infra* en matière de conception et d'architecture sont applicables quelle que soit la nature des applications développées. Elles peuvent être adaptées au contexte.

6.1. **Identification et gestion des menaces.**

L'analyse de risque, réalisée en amont, permet d'identifier et de hiérarchiser les menaces à prendre en compte, et de manière cohérente, lors du développement de l'application :

- les menaces spécifiques pesant sur les enjeux métier et l'application concernée ;

- la base de connaissance des menaces sur les logiciels de la méthode EBIOS (6) de l'ANSSI ;

- les menaces issues de l'analyse de risque faite dans le cadre de l'élaboration de la fiche d'expression rationnelle des objectifs de sécurité (FEROS) au profit de la sécurité des systèmes d'information.

DEV-RO-15 Il est obligatoire que l'équipe de projet (et plus particulièrement le chef de projet) prenne en compte les menaces issues de l'analyse de risques pesant sur l'application dont elle est responsable. Ces menaces sont hiérarchisées afin de prioriser leur traitement.

Ces menaces pesant sur l'application identifiées et retenues par l'autorité commanditaire seront communiquées à l'équipe de projet pour les prendre en compte tout au long du développement, leur hiérarchisation permettra de prioriser leur traitement.

L'ensemble des informations relatives à ces menaces sera formalisé et documenté afin d'enrichir le dossier d'homologation et le fonds documentaire du projet.

DEV-RO-16 Il est obligatoire de maintenir à jour le référentiel de menaces en tenant compte des évolutions du projet de développement et de son contexte.

Ces évolutions doivent être réalisées en concertation avec l'ensemble des acteurs concernés (autorité « cliente », équipe de projet, prestataires). Elles doivent être formalisées et documentées au profit du dossier de sécurité et d'homologation accompagnant les développements.

6.2. **Architecture applicative générale.**

DEV-RT-5 Il est recommandé de séparer et de cloisonner les couches de présentation (interface homme-machine, etc.), de traitement et d'accès aux données (bases de données, fichiers, etc.) constituant les applications.

Une architecture applicative ou logicielle en couches indépendantes permet de séparer et de cloisonner les différentes logiques via des interfaces claires et bien définies. Des interfaces simples facilitent les tests de sécurité, réduisent les possibilités d'erreurs ou de vulnérabilités.

En dissociant les différents traitements dans des modules spécifiques, une telle architecture applicative permet également, par la suite, d'isoler si nécessaire les composants logiciels sur des machines différentes dans des zones réseaux de niveau de sécurité adapté. De ce fait, les flux entre ces différentes couches peuvent être filtrés par des dispositifs physiques et/ou logiques (équipements réseau, etc.).

Cette architecture a également l'avantage de permettre une rupture protocolaire entre les différentes couches de l'application.

Le découpage choisi doit être précisé dans le dossier-type de conception et d'architecture de l'application informatique ou du logiciel.

DEV-RT-6 Il est recommandé de minimiser la surface d'attaque de l'application en réduisant le nombre de fonctionnalités et les services offerts au strict besoin ainsi que d'en limiter l'accès et l'utilisation.

La conception des applications doit décliner le principe de minimisation de la surface d'attaque : il s'agit de réduire au strict nécessaire le nombre de fonctionnalités à développer et l'accès aux fonctionnalités développées.

Ce principe est applicable à l'espace d'exécution (hébergement, machine virtuelle, etc.) de l'application : il s'agit de supprimer les services inutiles, ou à défaut de les rendre inaccessibles.

DEV-RT-7 Il est interdit d'utiliser des protocoles, services ou algorithmes obsolètes pour la conception de nouvelles applications.

Les protocoles, services ou algorithmes obsolètes ne doivent pas être retenus pour les nouveaux développements.

Les applications existantes doivent évoluer vers des versions de protocoles, services et algorithme pérennes et soutenues dans le cadre d'un maintien en condition de sécurité.

Les technologies obsolètes comportent souvent des faiblesses de conception avérées pouvant être exploitées pour contourner les fonctions de sécurité mises en place.

Exemples de services, protocoles ou algorithmes obsolètes et vulnérables :

- rlogin, rsh, telnet, SNMP v1, SSH v1. ;

- SSL, TLS v1.0, TLS v1.1. ;

- RC4, MD5, SHA1, DES, 3DES. ;

- LM, NTLM v1.

Les applications utilisant des protocoles ou des services obsolètes doivent faire l'objet d'un plan de remédiation, via un maintien en condition de sécurité adapté, afin de réduire l'exploitation des vulnérabilités induites.

DEV-RT-8 Il est obligatoire de réaliser et maintenir à jour un inventaire des codes externes (bibliothèques, *framework*, API, etc.) utilisés par l'application.

Les codes externes sont des modules développés (et maintenus) par des tiers. Leur utilisation est commune dans le cadre du développement.

En cohérence avec la règle DEV-RT-7, tout code externe doit être testé, maintenu, pérenne, et recensé dans les bases de données de vulnérabilités afin d'en évaluer les risques pour l'application.

Leur utilisation doit donc être inventoriée, et documentée dans le dossier-type de conception et d'architecture du projet de développement.

6.3. **Principes de conception.**

DEV-RT-9 Il est recommandé d'appliquer le principe de défense en profondeur à la conception de l'application.

Le principe de défense en profondeur consiste à déployer différents mécanismes de sécurité sur les différentes couches (ou modules) de l'application afin de prévenir, détecter, limiter, bloquer une attaque via l'exploitation de vulnérabilités, et de réparer les couches endommagées de l'application.

DEV-RT-10 Il est recommandé de concevoir des interfaces simples, minimales et bien définies.

Une définition d'interface précise simplifie l'analyse, l'inspection et les tests du code. Elle réduit également la surface d'attaque de l'application.

A contrario, la complexité au niveau des interfaces augmente les risques d'exploitation malveillante due à des comportements fonctionnels non prévus ou mal définis.

Les interfaces doivent implémenter les fonctionnalités strictement nécessaires au besoin fonctionnel. Toute fonctionnalité non utilisée doit être supprimée des interfaces (cf. DEV-RT-6).

Une interface doit être décrite en termes d'entrée/sortie, de services offerts ou accessibles répondant au strict besoin. Elle doit faire l'objet d'une formalisation dans le dossier-type de conception et d'architecture associé au développement informatique.

DEV-RT-11 Il est obligatoire d'appliquer les principes de moindres privilèges, de séparation des droits et de séparations des privilèges dans la conception de l'application à savoir : les droits et privilèges utilisés sont réduits et strictement nécessaires au fonctionnement des fonctionnalités et des services offerts, il en va de même pour leur exploitation.

L'application du principe du moindre privilège permet de réduire, de limiter voire de neutraliser les effets d'une attaque exploitant une vulnérabilité de l'application.

Ainsi, un code malveillant exploitant une vulnérabilité avec des droits et privilèges réduits ne pourra pas accéder à certaines ressources ou fonctionnalités offertes par le système d'exploitation : destruction ou modification de fichiers, modifications des configurations, etc.

Chaque droit ou privilège doit être attribué de manière adaptée et limitée dans le temps et l'espace lors de l'exécution des fonctionnalités de l'application. Ceci permet de qualifier le niveau de sensibilité du code associé, et de prioriser (éventuellement) son analyse au sens de la robustesse et de la sécurité.

En la matière, les fonctions critiques de sécurité ou complexes de l'application doivent faire l'objet d'une attention particulière.

L'ensemble des droits et privilèges nécessaires au fonctionnement de l'application et fonctionnalités associées doivent être recensés et décrits dans le dossier de sécurité et d'homologation associé au projet.

6.4. **Cloisonnement.**

DEV-RT-12 Il est obligatoire de limiter au strict besoin l'accès aux interfaces des modules embarquant ou utilisant des fonctionnalités (métier ou de sécurité) sensibles au sens de l'analyse de risque.

Afin de réduire les risques liés aux attaques par recherche exhaustive, les interfaces exposant des fonctionnalités (métier ou de sécurité) sensibles de l'application doivent être protégées, leur accès strictement contrôlé et conforme à la politique de sécurité idoine : réseau dédié, interface dédiée, authentification spécifique, cloisonnement logique des flux, restriction d'adresses réseau, etc.

Ces composants ou modules embarquant ou utilisant des fonctionnalités sensibles doivent être recensés à partir des résultats de l'analyse de risque. Ces fonctionnalités seront décrites dans le dossier-type de conception et d'architecture ou le dossier de sécurité et d'homologation du projet informatique.

Ainsi, les fonctions de sécurité ou à fort privilèges de l'application devront être considérées comme sensibles, et bénéficiées d'un accès limité et contrôlé.

De même, les fonctions d'accès aux données sensibles de l'application devront faire l'objet d'une réflexion particulière en termes de protection des bases de données et de leur positionnement dans la structure d'hébergement de l'application.

DEV-RT-13 Il est obligatoire d'identifier les composants (ou modules) de l'application interfacés avec des services, des applications ou des ressources tierces (ou externes).

En application du principe de défense en profondeur, les composants de l'application interfacés, ou connectés, avec des systèmes ou des services informatiques externes doivent être identifiés au cours de la conception, et faire l'objet de mesures de sécurité (cloisonnement, accès, etc.) spécifiques et adaptées. De plus, ces composants doivent faire l'objet d'une attention particulière lors du développement.

Ces composants ou modules applicatifs doivent être recensés et décrits dans le dossier-type de conception et d'architecture ou le dossier de sécurité et d'homologation du projet.

6.5. **Logique applicative.**

DEV-RT-14 Il est recommandé d'utiliser un format unique pour normaliser la représentation de toutes les données traitées par l'application.

Lorsqu'il est nécessaire de comparer ou de faire des opérations sur des noms de ressources (chemin du fichier, URI [7], etc.), il est recommandé d'utiliser les formes normalisées de la donnée et ce pour éviter que les formes différentes d'une expression ne contournent les contrôles de sécurité.

Par exemple, pour les noms de fichier sous Windows, il existe plusieurs manières de les représenter :

- C:Dossiertest.exe ;
- C:DoSsIERTEST.EXE ;
- C:/Dossier/test.exe ;
- %homedrive%Dossiertest.exe ;
- 127.0.0.1c$Dossiertest.exe ;

- 127.0.0.1c$Dossier...Dossiertest.exe.

DEV-RT-15 Il est déconseillé d'effectuer des conversions d'un type d'encodage de caractères vers un autre type d'encodage de caractères.

De manière générale, un caractère est encodé à partir d'une table (ou type) d'encodage de caractère pour pouvoir être représenté sous la forme d'une suite de bits (0 ou 1). Lors de la conversion, cette suite de bit peut représenter un autre caractère, par exemple la représentation 0x97 est « ù » en encodage DOS Europe de l'Ouest, et « - » (tiret cadratin) en encodage Windows Occidental. La conversion d'un encodage vers un autre peut permettre de contourner les mécanismes de protection d'accès aux répertoires sur les systèmes de fichier insensible à la casse (différence entre les majuscules et les minuscules).

DEV-RT-16 Il est obligatoire de vérifier la conformité, la qualité et la cohérence des données en entrée.

Toutes les données en entrée de l'application, y compris celles qui ne sont pas traitées, doivent être considérées comme non fiables et donc potentiellement malveillantes.

Il est nécessaire d'en vérifier la conformité au format attendu, aux valeurs attendues et cohérentes avec le contexte des données en entrée. En fonction du contexte les données en sorties peuvent être aussi concernées.

DEV-RT-17 Il est recommandé de procéder à la vérification et à la validation de la conformité des données en entrée au plus proche dans le temps et dans l'espace du code traitant ces données.

Afin de se prémunir contre les risques d'injection ou de contournement des contrôles de sécurité, les données d'entrée doivent être validées juste avant utilisation, afin de limiter le risque de modification entre les deux. Pour le modèle client/serveur, il s'agit de vérifier les données d'entrée au niveau serveur.

Par exemple, un filtrage de caractère implémenté dans le code javascript d'une application web peut être contourné par un attaquant utilisant un proxy web en modifiant le code javascript sur le client qu'il contrôle.

6.6. **Authentification.**

DEV-RT-18 Il est recommandé de déléguer le mécanisme d'identification et d'authentification à un module externe s'appuyant sur un référentiel centralisé de gestion des droits et des privilèges.

L'authentification est une fonction de sécurité critique pour une application. Toute erreur dans la logique d'implémentation de cette fonction peut aboutir à un accès non autorisé.

Il est donc préférable de s'appuyer sur un composant fiable et largement testé. Ce module d'authentification pourra être centralisé et ainsi bénéficier aux autres applications faisant appel à ce service. Un module d'authentification centralisé facilite également les tests et vérifications de sécurité.

DEV-RT-19 Il est obligatoire de faire précéder l'accès aux fonctionnalités et aux ressources de l'application par une phase d'authentification.

La phase d'identification et d'authentification doit précéder tout accès aux fonctionnalités de l'application pour réduire le risque d'un accès non autorisé et permettre une traçabilité des actions.

Dans le cas d'une architecture n-tiers, la phase d'identification et d'authentification intervient lors de la connexion à la couche présentation via le site Web, lors de l'utilisation des fonctionnalités de la couche applicative et lors de l'accès aux données de la couche données.

DEV-RT-20 Il est obligatoire de restreindre l'accès à toute ressource non publique de l'application via l'utilisation d'un mécanisme d'authentification.

Seules les ressources explicitement destinées à être publiques peuvent être accessibles sans une authentification préalable. L'accès à toute autre ressource sans authentification doit être interdit.

L'attribution de droits d'accès aux ressources doit respecter le principe de strict nécessité, de traçabilité et de non-répudiation des actions grâce un mécanisme d'identification et d'authentification. Ainsi la surface d'attaque et les risques d'accès non autorisés s'en trouvent réduits.

DEV-RT-21 Il est interdit de stocker ou de faire transiter les authentifiants (mots de passe, jetons d'authentification, etc.) en clair (cf. [PSSI-M-T]) lors des procédures d'authentification de l'application**.**

Les authentifiants permettant à un utilisateur de se connecter à l'application ou de maintenir sa session applicative doivent être stockés de manière sécurisée à l'aide de moyens adaptés : utilisation de conteneurs logiques cryptographiques, etc.

De plus, ils seront échangés avec l'application uniquement sur des canaux chiffrés.

Il est préférable d'utiliser un canal sécurisé par TLS (8) ou SSH (9) afin d'éviter toute fuite d'information dans le cadre des échanges de données.

DEV-RT-22 Il est obligatoire de générer des identifiants techniques de session uniques et imprédictibles.

Afin d'éviter la collision ou l'usurpation d'identifiant de session, l'application doit utiliser pour chaque utilisateur authentifié un jeton unique et difficile à prédire par un utilisateur malveillant.

En particulier, les identifiants de session ne doivent jamais contenir d'information personnelle. Ils doivent être assignés à l'utilisateur et jamais choisis par lui.

DEV-RT-23 Il est recommandé de limiter dans le temps les sessions des utilisateurs ou des ressources clientes, et de libérer à l'issue des sessions les ressources de l'application utilisées.

Afin de limiter les possibilités pour un utilisateur malveillant de voler une session, toutes les sessions doivent avoir une durée de vie limitée dans le temps. Les identifiants de session doivent être détruits en cas de déconnexion ou d'expiration de la session. Cela concerne également les sessions d'échanges de données entre les applications.

À l'issue, toutes les ressources applicatives ou logicielles utilisées durant la session doivent être libérées afin d'éviter la saturation de l'application.

6.7. **Gestion des erreurs.**

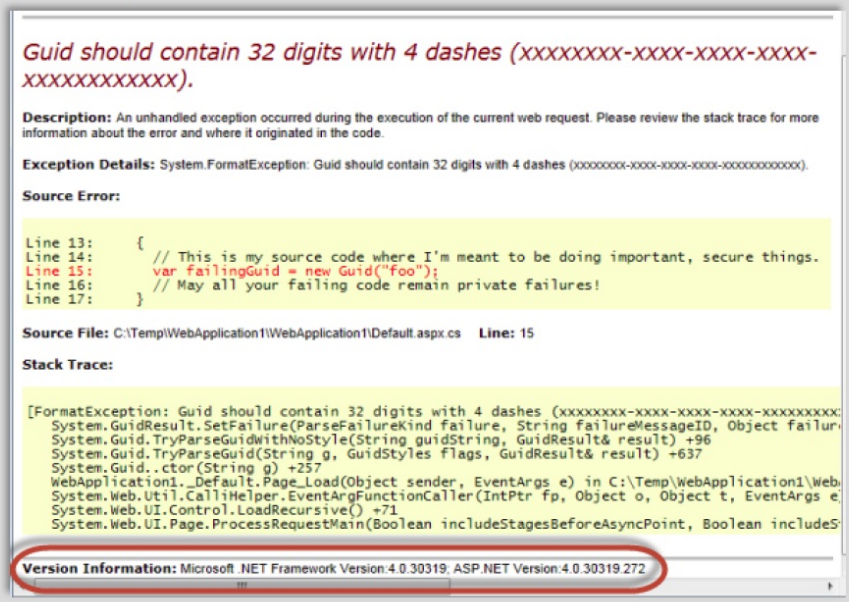
DEV-RT-24 Il est interdit de divulguer dans les messages d'erreur des informations techniques non pertinentes et inutiles pour gérer les erreurs rencontrées par les utilisateurs de l'application

Les informations présentes dans les messages d'erreurs doivent être spécifiques, claires, cohérentes, courtoises, strictement nécessaires et facilement exploitables par les utilisateurs de l'application.

De plus, les informations présentes dans les messages d'erreurs par défaut peuvent divulguer à un utilisateur malveillant des informations sur les technologies utilisées par l'application, à savoir leur architecture, l'existence de comptes particuliers, etc.

Afin de pallier ce problème, il est nécessaire de gérer les exceptions de manière centralisée et retourner des messages d'erreurs génériques et laconiques ne délivrant que le minimum d'informations et de détails.

Exemple d'une page web exposant la version du serveur web :



6.8. **Gestion des traces.**

DEV-RT-25 Il est obligatoire de mettre en place une gestion des journaux identifiant et traçant les événements et les incidents de sécurité de l'application.

L'application doit tracer les authentifications et les actions réalisées par l'utilisateur. Les traces générées doivent au minimum faire apparaître les éléments qui suivent :

- l'identifiant de l'utilisateur ou de la ressource cliente ;

- l'horodatage et la sévérité de l'incident de sécurité ;

- la source de la connexion ;

- le résultat des actions (succès ou échec) liées à la sécurité ;

- les anomalies de fonctionnement.

Ces traces doivent être générées et stockées conformément à la directive [DIR TRACES].

DEV-RT-26 Il est interdit de stocker des informations sensibles dans les journaux de l'application.

Les journaux de l'application ne doivent pas contenir de données sensibles, comme par exemple des paramètres d'authentification ou des données métiers.

Dans le cas des applications informatiques mobiles, les éventuels journaux générés et stockés au niveau du client, ne doivent pas contenir d'informations sensibles au risque de favoriser leur accès à un utilisateur malveillant.

6.9. **Gestion des fichiers.**

DEV-RT-27 Il est obligatoire de procéder à une analyse d'innocuité de tout fichier téléchargé, téléversé ou transmis avant de le traiter ou de le stocker dans son conteneur final (répertoire disque dur, base de données, etc.).

Le téléchargement, le téléversement et la transmission de fichiers vers une application sont des moyens privilégiés pour déposer une charge malveillante à distance.

Les fichiers reconnus comme malveillants ou non conformes aux caractéristiques attendues (formats, structures, contenus, etc.) doivent être immédiatement rejetés et leur traitement interrompu. Un message d'erreur approprié doit être renvoyé par l'application.

6.10. **Données de test et de débogage.**

DEV-RT-28 Il est obligatoire d'identifier les fonctions, commentaires et données utilisés lors du développement et devant être supprimés lors du passage en production.

Toute fonction d'activation implémentée concernant les modes de débogage, tout commentaire ou toute donnée facilitant le développement et la validation des fonctionnalités de l'application doivent être supprimés lors du passage en phase de production.

Ainsi, ces éléments doivent être identifiés le plus tôt possible. Leur identification systématique pendant le développement selon une méthodologie éprouvée permet de réduire le risque d'un oubli lors du passage en production.

7. IMPLÉMENTATION ET PROGRAMMATION.

Les règles *infra* en matière d'implémentation et de programmation sont applicables quelle que soit la nature des applications développées. Elles peuvent être adaptées au contexte.

7.1. **Conventions de codage.**

DEV-RT-29 Il est obligatoire de disposer de conventions et de règles de codage adaptées au langage utilisé et de les appliquer dans le cadre du développement.

Ces règles et conventions contribuent à la lisibilité et à la qualité du développement de l'application. Elles doivent être jointes à la documentation technique du projet de développement et constituent un référentiel de vérification lors des revues de code. Elles sont idéalement vérifiables à l'aide d'un outil adapté.

Tout développement (interne ou externe) au profit du ministère doit utiliser et appliquer, dans la mesure du possible, les règles de codage pour l'implémentation de logiciels en vigueur au sein du ministère (cf. [Guides codage]).

Les conventions de codage seront précisées en introduction du dossier-type des codes sources.

7.2. **Vérification des données.**

DEV-RT-30 Il est obligatoire de contrôler les données, à traiter et traitées, en entrée et en sortie des modules et des fonctionnalités développés et codés.

Afin de prévenir les risques de types injection, les données doivent être :

1. filtrées en rejetant les caractères non autorisés et non pris en compte ;

2. normalisées en les réduisant à leur représentation la plus simple ;

3. validées en vérifiant le format, le type, l'origine ou encore la longueur des données attendues ;

4. encodées selon le contexte de traitement.

DEV-RT-31 Il est recommandé de filtrer les données en explicitant seulement les cas acceptés (filtrage par liste blanche).

Afin d'éviter les contournements des filtrages de sécurité, il est préférable d'utiliser les listes blanches en lieu et place des listes noires. Les listes blanches permettent d'identifier clairement les cas acceptés et de rejeter les autres.

7.3. **Interactions avec l'environnement.**

DEV-RT-32 Il est obligatoire de d'utiliser des API et des bibliothèques pour interagir avec les autres systèmes (système de gestion de base de données, système d'exploitation, etc.).

Il faut minimiser le recours à des interpréteurs et éviter de permettre à une application d'exécuter des commandes directement sur un système ou d'initier des invites de commandes. Seules, dans la mesure du possible, les fonctions disponibles dans les bibliothèques conçues à cet effet seront utilisées.

7.4. **Protection des données sensibles.**

DEV-RT-33 Il est interdit de laisser des paramètres d'authentification en dur dans les codes sources ou les fichiers de commandes et d'instruction.

Laisser en dur dans les codes sources de l'application informatique ou du logiciel des paramètres d'authentification ou des clefs secrètes est à proscrire dans le cadre du développement robuste.

Ces informations sensibles devront être protégées en confidentialité et en intégrité par des conteneurs ou des moyens sécurisés adaptés.

7.5. **Crypographie.**

DEV-RT-34 Il est interdit de développer ses propres routines de cryptographie en dehors des développements cryptographiques gouvernementaux.

L'implémentation correcte d'un algorithme cryptographique est une tâche très complexe. Il est fortement probable qu'une telle implémentation contienne des erreurs.

Il est donc nécessaire d'utiliser des routines cryptographiques définies dans des bibliothèques largement testées, éprouvées et maintenues.

7.6. **Gestion des droits et privilèges.**

DEV-RT-35 Il est recommandé de limiter au strict minimum l'utilisation de droits et de privilèges élevés en termes de fonctions (création et accès aux ressources informatiques sous-jacentes, etc.), de temps (obtenus au plus tard, rendus au plus tôt) et d'espace (réduction des fonctions à privilèges) pour mettre en œuvre les fonctionnalités de l'application.

Tout en respectant le principe du moindre privilège (cf. DEV-RT-11), l'utilisation de droits et de privilèges élevés nécessaires aux fonctionnalités de l'application développées doit être temporaire, strictement limitée au besoin fonctionnel (accès aux fichiers sensibles, création de *socket* réseau, etc.) et accordée à des fonctionnalités spécifiques, identifiées et en nombre restreint.

Ainsi, il s'agit :

- de limiter les portions de code ayant besoin de droits et de privilèges élevés pour leur exécution ;

- de donner les droits et privilèges élevés strictement nécessaires à l'exécution des portions de code concernées ;

- d'accorder les droits et privilèges élevés aux portions de code concernées juste avant leur exécution ;

- de retirer les droits et privilèges élevés aux portions de code concernées juste après leur exécution ;

- de s'assurer de l'exécution des portions de code sans droits et privilèges élevés lorsqu'ils ne sont pas nécessaires.

7.7. **Gestion des ressources informatiques.**

DEV-RT-36 Il est obligatoire d'utiliser des API et des bibliothèques éprouvées, sûres et maintenues pour gérer les ressources informatiques (système d'exploitation, accès réseau, etc.) nécessaires aux fonctionnalités de l'application.

Il est fortement déconseillé de permettre à une application d'interagir (exécution de commande, accès, etc.) directement avec des ressources informatiques sous-jacentes ou distantes : on utilisera, dans la mesure du possible, les fonctions disponibles dans les bibliothèques conçues à cet effet.

DEV-RT-37 Il est obligatoire de mettre en œuvre les fonctions de gestion (création, allocation, libération, etc.) et de manipulation (lecture, écriture, etc.) des ressources informatiques (mémoire, fichier, etc.) au plus près de leur utilisation effective.

Toute allocation de ressources informatique doit être faite juste avant son utilisation. Toute libération de ressources informatique doit être faite juste après son utilisation.

Il peut exister des exceptions dans le cas des systèmes temps réel, ou en fonction des besoins fonctionnels.

Ainsi, si nous invoquons la fonction fopen() pour ouvrir un fichier en langage C, il faut penser à le fermer avec la fonction fclose(), une fois que l'on a terminé de travailler avec le fichier.

De même, la mémoire allouée dynamiquement à la demande de l'application doit être explicitement libérée à la fin du traitement ou à la sortie du contexte.

DEV-RT-38 Il est interdit de mettre les fonctions de manipulation (écriture, lecture, etc.) des ressources informatiques (mémoire, fichiers, etc.) dont la gestion des données n'est pas sécurisée, ou obsolète

L'utilisation de fonctions de manipulation de données (tampon, chaines de caractères etc.) pouvant être altérées doit être proscrite, car elles sont des vecteurs d'écrasement de tampon ou de corruption de mémoire, en cas d'attaque réussie ou d'erreur.

Toute fonction qualifiée d'obsolète ne doit pas être utilisée pour le développement de nouvelles applications. Pour l'existant, elle doit être remplacée par une version sûre, récente et pérenne dans le cadre du maintien en condition de sécurité.

La fonction C strcpy() est vulnérables aux débordements de tampon car elle ne gère pas la longueur de la chaîne de caractère à copier. On lui préférera la fonction strncpy().

7.8. **Gestion des références.**

DEV-RT-39 Il est recommandé de réduire la visibilité, l'utilisation et la capacité de modification des données mutables (objet, variables, etc.) aux seules portions de code concernées et consommatrices.

Il s'agit de limiter le partage de référence et le risque de corruption des données :

- en utilisant les accès en lecture seule ou en fournissant une copie de la donnée ou de l'objet en l'absence d'éventuelles modifications devant être propagées ;

- de protéger plus généralement les fonctionnalités développées contre les surcharges (modification a posteriori du code exécuté) et les empoisonnements (modification de variables globales).

7.9. **Lutte contre les injections.**

DEV-RT-40 Il est obligatoire d'utiliser les requêtes paramétrées pour interagir avec les bases de données.

La concaténation de chaines de données en entrée pour construire dynamiquement des requêtes SQL favorise la compromission des bases de données via des attaques de types injections SQL. Il faut donc lui privilégier les requêtes paramétrées en utilisant les fonctions des bibliothèques (ou API) des systèmes de gestion de bases de données.

DEV-RT-41 Il est obligatoire de s'assurer de l'innocuité (contrôle, validation) des fichiers en entrée avant leur traitement ou leur stockage.

La vérification de l'innocuité des fichiers traités par l'application doit prendre en compte :

- le type de fichier ;
- son nom ;
- son extension ;
- sa taille ;
- son origine ;
- le contrôle de son contenu structurel et sémantique;
- l'analyse des entêtes spécifiques ou non ;
- le stockage final et les droits d'accès du ficher.

L'utilisation d'un antivirus peut s'avérer utile pour s'assurer de l'absence de codes malveillants dans les fichiers. Avant stockage, il faudra s'assurer que son transfert ne risque pas de saturer le système de fichier.

DEV-RT-42 Il est déconseillé de confier la gestion ou la modification des caractéristiques ou métadonnées techniques (nom, emplacement, droits d'accès, etc.) des fichiers à un tiers ou à un utilisateur.

L'application doit gérer, et vérifier, les métadonnées techniques des fichiers transmis ou reçus avant leur stockage à l'emplacement de destination afin d'éviter des comportements applicatifs non prévu (écriture dans des répertoires sensibles, modifications des fichiers de configuration, etc.).

Ainsi, seule l'application doit pouvoir modifier les noms, les emplacements et les droits d'accès des fichiers transmis, reçus ou traités.

7.10. **Gestions des exceptions et des erreurs.**

DEV-RT-43 Il est obligatoire d'intercepter et de traiter de manière adéquate et au juste niveau les exceptions interrompant le fonctionnement normal de l'application.

Toute exception interrompant l'exécution d'une portion de code doit être interceptée, analysée et traitée par l'application.

Ainsi, il est nécessaire d'utiliser des blocs de traitement et les gestionnaires d'exception éventuellement fournis par le langage utilisé (blocs d'instructions *try/catch/finally,handler-bind,try/except,* etc.).

7.11. **Contrôles des codes sources.**

DEV-RT-44 Il est recommandé d'effectuer des analyses statiques des codes sources pendant la phase d'implémentation et de programmation.

Les outils d'analyse statique de code peuvent aider à détecter les vulnérabilités les plus courantes. Il est important de réaliser ce type d'analyse lors des phases de développement afin de détecter et de corriger ces vulnérabilités le plus tôt possible dans le cycle de développement.

Certains environnements de développement intègrent directement des outils d'analyse statique de code, ces outils simplifient la correction de bug de sécurité et permettent d'habituer les développeurs à détecter les erreurs courantes.

DEV-R0-17 Il est obligatoire de réaliser des revues des codes sources pendant la phase d'implémentation et de programmation à partir des règles de codage en vigueur au sein du ministère.

Les revues de code doivent être réalisées en priorité sur les portions de code les plus critiques au sens de l'analyse de risque, selon les principes suivants :

- les codes sources sont conformes aux règles de codage définies en termes de présentation, de lisibilité et d'implémentation (cf. [Guide codage]) ;

- les revues de code sont réalisés par des développeurs formés au développement sécurisé ;

- les revues de code ne sont pas réalisées par les auteurs du code en cours d'analyse.

DEV-R0-18 Il est obligatoire de décrire et de formaliser les tests unitaires pour chaque unité de code lors de la phase d'implémentation et de programmation.

Les tests unitaires jouent un rôle important dans la détection des régressions lors des modifications du code. Ils seront décrits dans le dossier-type ou le plan de tests de l'application concernée.

De surcroit, l'ensemble des tests unitaires doit être rejoué après une modification du code afin de vérifier l'absence de régressions.

8. INTÉGRATION ET DÉPLOIEMENT.

Les règles *infra* en matière d'intégration et de déploiement sont applicables quelle que soit la nature des applications développées. Elles peuvent être adaptées au contexte.

8.1. **Processus d'intégration.**

DEV-R0-19 Il est obligatoire de mettre en œuvre un processus d'intégration des modules unitaires de l'application.

Le processus d'intégration permet de vérifier, lors de l'assemblage des modules unitaires de l'application :

- l'absence de dysfonctionnements des fonctionnalités et fonctions métier ;

- l'absence de dysfonctionnements lors des interactions avec les services externes ;

- l'absence de dysfonctionnements avec l'environnement d'accueil.

Le processus d'intégration, et l'ensemble des modalités pratiques associées, sont décrits dans le dossier-type d'intégration de l'application.

DEV-RT-45 Il est recommandé d'automatiser le processus d'intégration des modules unitaires de l'application.

Le processus doit notamment s'appuyer sur l'automatisation des taches d'intégration décrites dans le dossier-type d'intégration de l'application. L'automatisation garantit la reproductibilité des étapes d'intégration.

Le processus peut inclure des tests de validation d'intégration pour chaque module unitaire. Ces tests pourront être intégrés au dossier-type ou au plan de tests.

DEV-RT-46 Il est obligatoire de gérer les anomalies détectées lors du processus d'intégration via les outils de suivi des bogues et des vulnérabilités utilisés pour l'application.

Il s'agit *in fine* de recenser toutes les anomalies rencontrées, et de les traiter en fonctions de leur gravité et de leur impact sur la robustesse et la sécurité de l'application lors de l'intégration des modules unitaires.

8.2. **Mise en production.**

DEV-R0-20 Il est obligatoire de mettre en œuvre une gestion des mises en production de l'application.

La gestion des mises en production vise à concevoir un processus formel permettant de garantir que les nouvelles versions de l'application sont mises en production de manière contrôlée et après avoir été testée. La gestion des mises en production est décrite dans le dossier-type de mise en production de l'application.

DEV-RT-47 Il est recommandé d'automatiser le processus de mise en production (pré-production, production, etc.) de l'application.

Ce processus doit notamment s'appuyer sur l'automatisation des taches de déploiement et la formalisation des étapes dans des documents de type procédures. Il peut inclure des tests de validation des différentes tâches de la mise en production.

Il doit être conforme aux procédures de mise en production en vigueur au sein du ministère.

DEV-RT-48 Il est obligatoire de gérer les anomalies détectées lors de la mise en production via les outils de suivi des bogues et des vulnérabilités utilisés pour l'application.

Il s'agit *in fine* de recenser toutes les anomalies rencontrées, et de les traiter en fonctions de leur gravité et de leur impact sur la robustesse et la sécurité de l'application lors des mises en production.

DEV-RT-49 Il est interdit de livrer une application informatique ou un logiciel incluant des paramètres d'authentification par défaut dans les codes sources ou les fichiers de configuration.

Aucun paramètre d'authentification par défaut, ne doit être stocké dans les codes sources ou les fichiers de configuration inclus dans les livraisons au risque d'être utilisés pour les déploiements ultérieurs.

8.3. **Distribution.**

DEV-R0-21 Il est recommandé de vérifier l'authenticité et l'intégrité de la livraison récupérée avant son installation.

Il est fréquent de voir des attaquants substituer des codes sources authentiques par une version contenant du code malveillant. Ainsi, il est donc nécessaire de vérifier l'authenticité et l'intégrité de la livraison avant de procéder à son installation dans un environnement de production, afin de garantir le déploiement de la bonne version de l'application.

On pourra notamment s'appuyer sur des fonctions de signature du code.

8.4. **Déploiement.**

DEV-RT-50 Il est obligatoire de déployer les composants applicatifs de l'application dans des zones d'hébergement adaptées à leur sensibilité et leurs fonctionnalités conformément aux règles DEV-RT-5 et DEV-RT-9.

Une application conçue selon une architecture modulaire permet de réaliser un déploiement de chaque composant applicatif de manière distribuée sur différentes machines et différentes zones d'hébergement en fonction de leur sensibilité et leurs fonctionnalités.

En fonction de l'analyse de risque et de la sensibilité des données stockées, le serveur de base de données peut être hébergé dans une zone protégées par des moyens de sécurité spécifiques (pare-feu, sondes).

DEV-RT-51 Il est obligatoire de gérer les anomalies détectées lors des déploiements via les outils de suivi des bogues et des vulnérabilités utilisés pour l'application.

Il s'agit *in fine* de recenser toutes les anomalies rencontrées, et de les traiter en fonctions de leur gravité et de leur impact sur la robustesse et la sécurité de l'application.

Tout déploiement d'application doit être validé, décrit dans le dossier-type de déploiement et d'exploitation du projet et effectué en coordination avec l'entité responsable de l'hébergement.

8.5. **Installation.**

DEV-RT-52 Il est obligatoire d'utiliser les droits et les privilèges minimum et strictement nécessaires pour l'installation de l'application.

La conception d'application doit prendre en compte le principe du moindre privilège, comme décrit dans la règle DEV-RT-11. De même, seuls les droits et les privilèges minimum et strictement nécessaires seront utilisés pour l'installation. Ils doivent être documentés et formalisés au sein de la documentation technique du projet de développement.

DEV-RT-53 Il est obligatoire de documenter les procédures d'installation et de désinstallation de l'application concernée.

Il est obligatoire d'utiliser des procédures d'installation et de désinstallation documentées afin de contrôler les étapes du processus, de pouvoir reproduire ces opérations de manière uniforme, et d'éviter l'absence d'incohérence suite à la désinstallation.

DEV-RT-54 Il est recommandé d'automatiser les procédures d'installation et de désinstallation de l'application concernée.

L'automatisation des procédures d'installation et de désinstallation garantit leur conformité et leur uniformité lors des différents déploiements.

De plus, elle garantit l'absence d'incohérence suite à la désinstallation.

DEV-RT-55 Il est obligatoire de supprimer les journaux et les fichiers temporaires d'installation générés sur les serveurs de production une fois l'installation de l'application terminée.

Les journaux d'installation peuvent dévoiler des informations techniques à un potentiel attaquant. Ils doivent être supprimés de manière sécurisée après l'installation de l'application.

Les processus d'installation et de désinstallation sont décrits dans le dossier-type de déploiement et d'exploitation.

8.6. **Configuration et durcissement de l'installation.**

DEV-RT-56 Il est obligatoire de minimiser les composants et les fonctionnalités installés lors du déploiement de l'application

Selon le principe de la réduction de la surface d'exposition décrit dans la règle DEV-RT-6, il est nécessaire de supprimer tous les composants et fonctionnalités qui ne sont pas requis pour le déploiement et le bon fonctionnement de l'application. Lors de l'installation, il peut être nécessaire de désélectionner voire désinstaller certains composants installés par défaut.

On pourra également procéder à la suppression après installation de certains fichiers ou composants inutiles pour un déploiement en production.

Les composants et les fonctionnalités de l'application installés sont listés dans le dossier-type de déploiement et d'exploitation. Cette liste est cohérente dans les éléments contenus dans le dossier-type de conception et d'architecture.

8.7. **Défense en profondeur.**

DEV-RT-57 Il est obligatoire de filtrer et d'interdire tous les flux réseaux non nécessaires au bon fonctionnement de l'application et de sa plateforme.

Tout flux qui n'est pas spécifiquement autorisé doit être interdit. Pour implémenter ce filtrage, on utilisera une règle par défaut interdisant tous les flux et une liste blanche à laquelle seront ajoutés les flux autorisés.

DEV-RT-58 Il est recommandé de chiffrer par défaut toutes les données en transit.

Afin de limiter les risques d'écoute passive, d'interception ou d'altération de données, il est recommandé d'interdire le transit des données en clair. On pourra utiliser à cette fin des protocoles de chiffrement de flux réseau conformes aux préconisations de sécurité.

DEV-RT-59 Il est obligatoire de protéger les applications utilisant des technologies de type « Web » à l'aide d'un produit de sécurité embarquant une fonctionnalité de pare-feu applicatif.

Les solutions de filtrage réseau travaillant au niveau des couches réseaux ne permettent pas de bloquer les attaques utilisant des vulnérabilités propres à la couche applicative. Certaines de ces vulnérabilités sont bien connues et peuvent être rendues en partie inexploitables par un ensemble de règles implémentées par un équipement filtrant au niveau applicatif.

De nouvelles vulnérabilités impactant des bibliothèques, moteurs de bases de données ou *frameworks* sont régulièrement découvertes, et peuvent affecter les systèmes utilisant des technologies « Web ». Seules les solutions de pare-feu applicatif peuvent pallier certaines de ces vulnérabilités.

La défense en profondeur mise en œuvre est cohérente avec le dossier-type de conception et d'architecture.

8.8. **Durcissement des configurations des plateformes.**

DEV-RT-60 Il est obligatoire de durcir les configurations de tous les composants informatiques (y compris ceux de l'application) de la plateforme de production conformément aux guides de configuration ([GUIDES CONFIG]) de référence en vigueur au sein du ministère.

Les plateformes de production comprennent des composants devant obligatoirement faire l'objet d'un durcissement. Ce durcissement est réalisé pour chaque composant de la plateforme à partir d'une configuration

de référence validée en termes de sécurité.

Le durcissement est conforme au principe de réduction de la surface d'attaque de la règle DEV-RT-6 : tout ce qui n'est pas nécessaire ou autorisé de manière explicite doit être interdit.

Il est applicable à toutes les couches en support de l'application informatique, par exemple :

- les packages applications informatiques ou logiciels installés sur le serveur ;
- les services disponibles ;
- les ports ouverts ;
- les comptes utilisateurs ou applicatifs ;
- les groupes utilisateurs, etc.

DEV-RT-61 Il est obligatoire d'automatiser la procédure de durcissement des configurations de tous les composants informatiques (y compris ceux de l'application) de la plateforme.

Le durcissement d'une plateforme nécessite de configurer et vérifier un nombre important de points de contrôle afin de garantir sa conformité aux préconisations de sécurité. Afin de pouvoir déployer des plateformes durcies de manière uniforme et avec un niveau d'assurance satisfaisant, il est nécessaire d'automatiser les procédures de déploiement.

On pourra s'appuyer sur des configurations de référence ou plateformes de références afin de faciliter la maintenance dans le temps du durcissement des plateformes. Dans la mesure du possible, on veillera à contrôler la conformité de la configuration des plateformes vis-à-vis du gabarit de référence lors du déploiement et par la suite dans le cycle de vie de l'application informatique.

Le durcissement des configurations est décrit dans le dossier-type de configuration ou le dossier de sécurité et d'homologation de l'application.

8.9. **Gestion des traces.**

DEV-RT-62 Il est obligatoire de déployer une gestion et une centralisation des traces générées par les composants informatiques (y compris ceux de l'application) de la plateforme conformément à la directive [DIR TRACES].

La protection des journaux de l'application et des éventuels composants de la plateforme sous-jacente, doit être conforme à la directive en vigueur au sein du ministère.

Ainsi, les journaux d'installation générés sur le serveur de production pourront être centralisés avant d'être effacés une fois l'installation ou la désinstallation achevée.

DEV-RT-63 Il est obligatoire de synchroniser les composants informatiques (y compris ceux de l'application) des plateformes de production avec une référence de temps unique de référence conformément à la directive [DIR SYNC TPS].

Les traces applicatives doivent contenir parmi les informations obligatoires, la date précise de l'évènement tracé. Afin de s'assurer que la date des évènements produits est correcte, les plateformes de production doivent être synchronisées sur une source de temps unique de référence.

On utilisera un protocole dédié de synchronisation temporelle afin de contrôler la dérive des horloges.

9. TESTS ET NIVEAU DE ROBUSTESSE ET DE SÉCURITÉ.

Les règles infra en matière de tests et d'évaluation de niveau de robustesse et de sécurité sont applicables quelle que soit la nature des applications développées. Elles peuvent être adaptées au contexte.

9.1. **Stratégie de tests.**

DEV-R0-22 Il est obligatoire de mettre en œuvre une stratégie de tests afin d'évaluer la robustesse et la sécurité de l'application.

La stratégie de tests doit couvrir, *a minima*, les aspects suivants :

- les tests unitaires et de non régression des modules unitaires ;
- les tests d'intégration des modules unitaires ;
- les tests de robustesse des fonctions métier ([GUIDES OWASP], type OSSTM, etc.) ;
- les tests des fonctions de sécurité (authentification, journalisation, etc.) ;
- les tests d'analyse statique et de revue de code ([GUIDE CODAGE]) ;
- les tests d'injection de données aléatoires ;
- les audits de configuration des plateformes (développement, intégration, production, etc.) ;
- etc.

Pour assurer la complétude des tests, on se réfèrera à la documentation du groupe de travail sur les tests [GT TESTS].

La stratégie de test, et les modalités associées, sont décrites dans le dossier-type ou le plan de tests. Les résultats sont consignés dans des procès-verbaux de recette ou tests.

DEV-R0-23 Il est obligatoire de décrire chaque test, les résultats attendus, et le suivi des résultats pour chaque phase du développement informatique.

Ces éléments sont formalisés dans le dossier-type ou le plan de tests, et les procès-verbaux de recette.

DEV-RT-64 Il est recommandé d'industrialiser et d'instrumentaliser chaque test effectué à l'aide de plateformes et d'outils adaptés, afin d'assurer leur reproductibilité et leur automatisation.

Les tests de sécurité, et principalement ceux ayant lieux lors de la phase de développement, peuvent rapidement devenir couteux en termes de temps. Il est donc préconisé de s'appuyer sur des *frameworks* ou sur des outils de tests permettant d'automatiser la détection et la production de rapports sommaires.

Pour ce faire, il existe plusieurs outils commerciaux ou libres dédiés à la sécurité, ou incluant des plugins spécifiques à la tâche. Parmi ceux qui peuvent être utilisés dans le développement web (liste non exhaustive) :

- *Jenkins* ;
- *Mozilla Minion* ;
- *OWASP ZAP* ;
- *frameworks* de type *Unit.

DEV-RT-65 Il est obligatoire d'effectuer les tests automatisés de manière régulière et itérative à chaque phase du développement informatique.

Il est conseillé de planifier des tests automatisés (scans de vulnérabilités, tests unitaires, audit de code, etc.) sur l'application et son code source de manière régulière.

On peut par exemple planifier un test de robustesse sur une application web ainsi qu'un audit de son code source tous les jours entre 21 h 00 et 7 h 00.

9.2. **Gestion des résultats des tests.**

DEV-R0-24 Il est obligatoire de qualifier, via une métrique adaptée, les résultats des tests en fonction de leur impact sur la robustesse et la sécurité de l'application concernée.

Les résultats ayant un impact sur la robustesse et la sécurité de l'application concernée, seront qualifiés d'anomalies, de bogues, de faits techniques ou défauts en fonction du contexte culturel.

Ces éléments d'information seront précisés dans le dossier-type ou les plans de tests.

On retiendra la terminologie d'anomalies dans les lignes suivantes.

DEV-RT-66 Il est obligatoire de tracer et de traiter les anomalies détectées via les outils de suivi des bogues et des vulnérabilités utilisés pour l'application.

Le traitement d'anomalie doit donner lieu :

- à une correction effective de l'anomalie ;

- à la mise en œuvre d'une mesure de sécurité (absence de correction) ;

- à une vulnérabilité (absence de correction, absence de mesure de sécurité).

DEV-RT-67 Il est obligatoire de vérifier la correction effective des anomalies réputées « corrigées » via des tests adaptés, et d'en tracer les résultats via les outils de suivi des bogues et des vulnérabilités utilisés pour l'application.

Les mesures de sécurité mises en œuvre suite à une anomalie seront intégrées aux dossiers-types techniques (conception et architecture, intégration et déploiement, etc.) ou au dossier de sécurité et d'homologation en fonction de leur nature (évolution d'architecture, modification du code source, modification de la plateforme de production, etc.).

DEV-R0-25 Il est obligatoire d'enrichir le dossier des vulnérabilités résiduelles avec les vulnérabilités recensées dans les outils de suivi des bogues et des vulnérabilités utilisés pour l'application.

Les vulnérabilités et les arbitrages (dérogations, etc.) suivis à partir de l'outils de suivi des bogues et des vulnérabilités du développement doivent enrichir le dossier de vulnérabilités résiduelles (cartographie des vulnérabilités) du dossier de sécurité et d'homologation afin de fournir les informations pertinentes à la prise de décision en matière d'homologation de l'application.

9.3. **Niveau de robustesse et de sécurité.**

DEV-RT-68 Il est recommandé d'évaluer le niveau de robustesse et de sécurité d'une application à partir d'une analyse statique automatique du code, d'un test automatique de type OWASP et d'un audit de conformité des configurations.

Les résultats doivent être suivis dans l'outil de suivi des bogues et des vulnérabilités de l'application et enrichir le dossier de vulnérabilités résiduelles du dossier de sécurité et d'homologation.

Cette série de tests peut s'intégrer dans un cycle d'intégration continue.

Pour le ministre de la défense et par délégation :

L'ingénieur général hors classe de l'armement,
directeur général des systèmes d'information et de communication,

Marc LECLÈRE.

---

(1) Structured query language : langage de requête structurée permettant d'interagir avec les systèmes de gestion de base de données.

(2) Site http://phare.intradef.gouv.fr/phare/.

(3) Appelés aussi script ou batch.

(4) Maîtrise d'œuvre.

(5) Adress space layout randomization : distribution aléatoire de l'espace d'adressage.

(6) Expression des besoins et identification des objectifs de sécurité.

(7) Uniform ressource identifier : identifiant unique de ressource informatique.

(8) Transport layer security : protocole de sécurisation de la couche transport du protocole TCP/IP.

(9) Secure shell : ligne de commande sécurisée.

ANNEXE I.

RÉFÉRENTIELS MENACES ET RISQUES.

APPENDICE I.A.

RÉFÉRENTIEL DE MENACES DU CYCLE DE VIE DU DÉVELOPPEMENT SÉCURISÉ DE MICROSOFT.

Le tableau *infra* énumère les différents types de menaces utilisés dans le cadre du cycle de vie de développement sécurisé (SDL) (1) de Microsoft.

MENACE.	DESCRIPTION.
Usurpation d'identité /Mystification [*Spoofing*]	Cette menace renvoie à un attaquant prenant l'identité d'un utilisateur légitime, ou plus généralement une ressource illégitime (serveur, etc.) utilisant l'identification d'une ressource légitime à des fins malveillantes.
Modification malveillante des données [*Tampering*]	Cette menace renvoie à la modification non autorisée de données par une ressource, ou un utilisateur, illégitime ; données contenues dans des bases de données ou dans un flux d'échange transitant via le réseau.
Répudiation [*Repudiation*]	Cette menace renvoie à la contestation de l'exécution d'une action par une partie sans que le contraire puisse être prouvé par absence de traçabilité et d'imputation des actions.
Divulgation d'information [*Information disclosure*]	Cette menace renvoie à la divulgation, ou la mise à disposition, d'informations à des ressources, ou des utilisateurs, en l'absence de droits d'accès légitime.
Deni de service [*Denial of service*]	Cette menace renvoie à la saturation d'une ressource afin d'empêcher des ressources, ou des utilisateurs, d'y accéder ou d'en réduire la disponibilité et la fiabilité.
Élévation de privilièges [*Elevation of privileges*]	Cette menace renvoie à l'acquisition illégitime de droits ou de privilèges de la part d'une ressource, ou d'un utilisateur, ne les possédant pas.

APPENDICE I.B.
RÉFÉRENTIEL DE MENACES ANSSI.

Le tableau *infra* énumère les différents types de menaces sur les logiciels de la base de connaissance EBIOS (2) de l'agence nationale de sécurité des systèmes d'information (ANSSI).

MENACE.	DESCRIPTION.
Détournement de l'usage prévu du logiciel LOG-USG	Cette menace renvoie à l'utilisation des fonctionnalités du logiciel, sans le modifier ni l'endommager, pour réaliser des actions imprévues : accès illégitime en lecture ou écriture à des fichiers, élévation de privilèges d'un compte utilisateur, détournement pour un usage malveillant, etc.
Analyse du logiciel LOG-ESP	Cette menace renvoie à l'analyse illégitime sans l'endommager du logiciel (code source, fonctionnement, fonctionnalités, etc.) depuis l'intérieur ou l'extérieur du système d'information afin de compromettre éventuellement les données et le savoir-faire : divulgation de données sensibles, ingénierie inverse, recherche de vulnérabilités intrinsèques, exploitation malveillantes des messages d'erreurs, etc
Dépassement des limites du logiciel LOG-DEP	Cette menace renvoie aux tentatives de dépassement, ou saturation, des capacités de stockage ou de traitement du logiciel sans qu'il soit modifié, pouvant entraîner son dysfonctionnement, la réalisation de fonctions non prévues, des pannes ou un déni de service : injection de données non prévues, débordement de tampon, surexploitation des ressources informatiques allouées (réseau, connexion aux bases de données, etc.), etc.
Suppression de tout ou partie du logiciel LOG-DET	Cette menace renvoie à la disparition partielle, ou totale, de manière temporaire ou permanente, des fonctionnalités attendues du logiciel : effacement du logiciel en production, écrasement du code exécutable à des fins malveillantes, etc.
Modification du logiciel LOG-MOD	Cette menace renvoie à la modification des fonctionnalités attendues du logiciel pouvant entraîner des dysfonctionnements ou des pannes : activation ou désactivation de fonctionnalités, modification de fichiers de configuration, piégeage des codes sources, etc.
Disparition du logiciel LOG-PTE	Cette menace renvoie au vol de logiciel, sans l'endommager, appartenant à un propriétaire légitime : violation des droits d'utilisation, perte de savoir-faire, etc.

APPENDICE I.C.
RÉFÉRENTIEL DES RISQUES OWASP.

Le tableau *infra* énumère le TOP 10 des risques pesant sur la sécurité des applications informatiques du projet libre de sécurisation des applications informatiques « Web » (OWASP).

RISQUE.	DESCRIPTION.
Injection	Ce risque renvoie à des envois de données non fiables, ou non vérifiées, vers des interprétateurs visant à le duper et lui faire exécuter des commandes illégitimes afin d'accéder à des données : injection de requêtes d'accès aux données d'une base de données, etc.
Violation de gestion d'authentification et de session	Ce risque renvoie à l'exploitation de failles dues à une mauvaise implémentation ou à une mauvaise mise en œuvre des mécanismes d'authentification ou de gestion des sessions, afin de compromettre les informations sensibles associées : mots de passe d'authentification, jetons de sessions, etc.
Exécution de codes arbitraires malveillants[*Cross-script scripting (XSS)*]	Ce risque renvoie à l'exécution de codes arbitraires au sein des navigateurs suite à l'absence de vérification des données envoyées à l'application à des fins malveillantes : détournement de sessions d'utilisateurs légitimes, défiguration de site web, redirection de l'utilisateur vers un site malveillant.
Références directes non sécurisées à un objet	Ce risque renvoie à l'exposition d'une référence vers un objet d'exécution interne ne bénéficiant pas d'un contrôle d'accès ou d'une protection adéquate : fichier de configuration, répertoire contenant des informations sensibles, connexion à une base de données, etc.
Mauvaise configuration de sécurité	Ce risque renvoie à l'absence de configuration sécurisée (durcissement de configuration) ou de maintien en condition de sécurité des briques logicielles déployées au profit de l'application conformément au dossier technique : serveur web, serveur d'application, plateforme d'hébergement, etc.
Exposition de données sensibles	Ce risque renvoie à l'absence de protection des données sensibles hébergées et stockées au sein de l'application, contre le vol ou la modification illégitime : numéro de carte bancaire, mots de passe applicatifs des utilisateurs, données à caractère personnel, etc.
Manque de contrôle d'accès au niveau fonctionnel	Ce risque renvoie à l'absence de vérification des demandes d'accès des utilisateurs aux fonctionnalités, et aux services associés offertes par l'application, sans quoi des demandes illégitime pourront être forgées : modification des droits d'accès via des requêtes forgées, accès à des informations sensibles via des requêtes forgées, etc.
Falsification de requêtes intersites [*Cross Site Request Forgery (CSRF)*]	Ce risque renvoie à l'utilisation à distance, par un utilisateur malveillant, du navigateur d'un utilisateur authentifié pour générer des requêtes légitimes pour accéder et interagir avec l'application.
Utilisation de composants avec des vulnérabilités connues	Ce risque renvoie à l'utilisation de composants et de briques logicielles embarquant des vulnérabilités connues et exploitables pouvant compromettre la sécurité et les défenses de l'application concernée : bibliothèques utilisées par l'application, systèmes d'exploitation de la plateforme d'hébergement, etc.
Redirections et renvois non validés	Ce risque renvoie à l'utilisation par l'application de données non fiables et non validées pour renvoyer les utilisateurs légitimes vers des pages ou des sites malveillants : sites d'hameçonnage (*phishing*), accès à des pages non autorisées, etc.

(1) Sécurity development lifecycle : cycle de vie de développement sécurisé.

(2) Expression des besoins et identification des objectifs de sécurité.

ANNEXE II.

POINTS DE CONTRÔLE ROBUSTESSE ET SÉCURITÉ DU DÉVELOPPEMENT.

La *checklist* ci-dessous présente de manière non exhaustive les points essentiels à vérifier lors d'une revue de code ou lors de vérifications concernant la robustesse et la sécurité des applications.

POINT DE CONTRÔLE.	CONFORMITÉ.
Vérifier que tous les composants (librairies, modules externes, etc.) identifiés sont nécessaires.	
Vérifier que tous les composants de l'application sont à jour et qu'ils ne sont pas vulnérables.	
Vérifier que l'accès aux fonctionnalités - qui ne sont pas explicitement publiques - requiert une authentification.	
Vérifier que l'application ne pré-remplit pas les champs contenant des secrets de connexion.	
Vérifier que les contrôles d'authentification se font de manière cohérente aux spécifications (par exemple côté serveur pour une application client-serveur).	
Vérifier que les contrôles d'authentification gèrent correctement les cas d'échecs et qu'ils ne divulguent pas d'informations techniques non génériques.	
Vérifier que l'application promeut et applique la politique de mots de passe en vigueur.	
Vérifier la robustesse des mécanismes de gestion de comptes (oubli de mot de passe, réinitialisation, etc.).	
Vérifier que le changement de mot de passe inclut le mot de passe actuel, le nouveau mot de passe ainsi qu'une confirmation.	
Vérifier que toutes les actions d'authentification sont correctement journalisées.	
Vérifier que les mots de passe sont correctement hachés. Ceci implique l'utilisation d'un algorithme de hachage robuste non réversible ainsi que d'un sel.	
Vérifier que l'authentification réussie génère de nouvelles informations (identifiants, *cookies*, jetons de session, etc.) de contexte de session et que la déconnexion détruit les informations de contexte de la session : les informations de contextes de session ne sont pas réutilisables d'une session à l'autre.	
Vérifier que l'application génère des identifiants de session complexes.	
Vérifier que la déconnexion d'un utilisateur invalide sa session.	
Vérifier qu'une session est limitée dans le temps.	
Vérifier que les secrets de sessions, comme les identifiants, ne sont pas divulgués dans les URL (1), dans les traces fonctionnelles, les messages d'erreurs, etc.	
Vérifier que le principe de moindres privilèges est correctement appliqué aux niveaux des groupes et des utilisateurs.	
Vérifier que les actions autorisant l'accès à des ressources (fichiers, réseau, mémoire, services tiers, etc.) sont correctement journalisées.	
Vérifier que les fonctions manipulant les données de l'utilisateur en entrée ne sont pas vulnérables à des attaques d'injection (dépassement de tampon, exécution de code arbitraire, etc.).	
Vérifier que le traitement des données de l'utilisateur en entrée est fait au plus près de leur consommation ou de leur traitement (côté serveur pour une application client-serveur).	
Vérifier qu'un seul mécanisme de filtrage et de validation des données de l'utilisateur en entrée est utilisé par l'application.	
Vérifier que toutes les données de l'utilisateur en entrée sont traitées et pas uniquement celles présentes dans les formulaires présentés à l'utilisateur.	
Vérifier que les données transférées d'un contexte DOM à un autre, sont correctement traitées, et que cela se fait en utilisant des méthodes JS robustes.	
Vérifier que le rejet d'une donnée de l'utilisateur en entrée est effectif et journalisé côté serveur.	
Vérifier que les requêtes de types SQL sont construites à partir de requêtes paramétrées et non à partir de la concaténation de chaînes de caractères alphanumériques.	
Vérifier que l'application est protégée contre l'injection de commandes	

Vérifier que les contextes d'encodage et d'affichage des variables et des données de l'utilisateur sont identiques.	
Vérifier la conformité d'implémentation et d'utilisation des modules cryptographiques dans les applications.	
Vérifier que les modules cryptographiques font l'objet d'une gestion et d'un traitement des erreurs sans risque de contournement des contrôles de sécurité.	
Vérifier que les aléas utilisés par l'application sont vraiment aléatoires et imprévisibles.	
Vérifier que l'entropie (nombre des possibles) des aléas utilisés par l'application est adaptée à leurs usages.	
Vérifier que les secrets (clefs privés, etc.) ou les données sensibles (mots de passe, etc.), utilisés par l'application bénéficient d'une protection adaptée à leurs usages.	
Vérifier que les zones de mémoire ayant stockée ou manipulé des secrets (clefs privés, etc.) ou des données sensibles (mots de passe, etc.) sont effacées et surchargées (écriture de données aléatoires dans les zones de mémoire) après leurs usages.	
Vérifier les implémentations correctes de la gestion des erreurs et du traitement des exceptions.	
Vérifier que les messages d'erreur sont génériques et qu'ils ne divulguent pas d'informations techniques ou sensibles.	
Vérifier que l'application journalise bien les différentes actions (action utilisateur ou administrateur, authentification, sécurité, etc.).	
Vérifier que la journalisation permet l'identification de l'utilisateur (ou de l'administrateur) son action et le résultat de son action.	
Vérifier que l'application ne journalise pas de secrets.	
Vérifier que l'application encode correctement les données stockées dans les journaux.	
Vérifier que tous les codes sources de l'application et des modules associés ne contiennent pas de portes dérobées, de fonctions cachées, etc.	
Vérifier que les redirections vers des pages ne se font que vers des URL strictement permises.	
Vérifier que les fichiers soumis par l'utilisateur sont analysés par un antivirus avant d'être traités par l'application.	
Vérifier que les données externes ne sont pas directement utilisées dans des inclusions, *class loader*, etc.	
Vérifier que les fichiers externes sont sauvegardés à l'extérieur de la racine de l'application dans un répertoire disposant de droits d'accès adaptés.	
Vérifier que l'application ne sauvegarde pas d'informations sensibles en clair côté client.	
Vérifier que l'application est compilée en utilisant les bonnes options de sécurité.	

(1) Uniform resource locator : localisateur uniforme de ressources.

ANNEXE III.

VÉRIFICATION EXIGENCES.

Le tableau *infra* donne des éléments de vérification du respect des exigences obligatoires la robustesse et la sécurité des applications.

EXIGENCES.	ÉLÉMENTS DE VÉRIFICATION.
DEV-RO-1	Liste des exigences de sécurité du dossier de sécurité et d'homologation. Listes des exigences fonctionnelles et métier du dossier technique conforme aux instructions ministérielles.
DEV-RO-2	Certificat de sécurité des acteurs impliqués dans le développement.
DEV-RO-4	Liste hiérarchisée des exigences de sécurité du dossier de sécurité et d'homologation.
DEV-RO-5 DEV-RO-6	Liste des rôles et privilèges du dossier de sécurité et d'homologation. Liste des rôles des responsables du dossier technique du projet de développement conforme aux instructions ministérielles.
DEV-RO-7 DEV-RO-8	Liste des responsabilités du dossier de sécurité et d'homologation.
DEV-RO-10 DEV-RO-12	Liste des modifications du dossier de sécurité et d'homologation. Liste des modifications du dossier du dossier technique.
DEV-RO-13	Dossier technique du projet de développement de l'application.
DEV-RT-1 DEV-RT-2	Dossier technique du projet de développement de l'application. Liste des outils de développement de l'application. Configurations des outils de développement.
DEV-RO-14	Annexes de sécurité du contrat.
DEV-RO-15 DEV-RO-16	Liste des menaces du dossier de sécurité et d'homologation.
DEV-RT-8	Dossier technique du projet de développement de l'application. Liste des outils de développement de l'application.
DEV-RT-11	Tableau des rôles et privilèges du dossier de sécurité et d'homologation. Dossier technique du projet de développement de l'application. Configuration des composants logiciels de l'application. Revue/Analyse statique du code source.
DEV-RT-12 DEV-RT-13	Dossier technique du projet de développement de l'application. Configuration des composants logiciels de l'application. Configuration de la plateforme de production ou de la structure d'hébergement. Revue/Analyse statique du code.
DEV-RT-16	Dossier technique du projet de développement de l'application. Revue/Analyse statique du code.
DEV-RT-19 DEV-RT-20	Dossier technique du projet de développement de l'application. Revue/Analyse statique du code.
DEV-RT-22	Dossier technique du projet de développement de l'application. Revue/Analyse statique du code.
DEV-RT-25	Dossier technique du projet de développement de l'application. Configuration des composants logiciels de l'application. Configuration de la journalisation conforme à l'instruction ministérielle. Configuration de la plateforme de production ou de la structure d'hébergement.
DEV-RT-27	Dossier technique du projet de développement de l'application. Dossier de sécurité et d'homologation de l'application. Procédures d'entrée/sortie des données et des fichiers. Revue/Analyse statique du code.
DEV-RT-28	Dossier technique du projet de développement de l'application. Liste des données de test et de débogage. Revue/Analyse statique du code.

DEV-RT-29 DEV-RT-30 DEV-RT-32 DEV-RT-36 DEV-RT-37 DEV-RT-40 DEV-RT-41 DEV-RT-43	Dossier technique du projet de développement de l'application. Revue/Analyse statique du code. Suivi des bogues et des anomalies.
DEV-RO-17 DEV-RO-18	Dossier technique du projet de développement de l'application. Revue/Analyse statique du code. Suivi des tests et des résultats. Suivi des bogues et des anomalies.
DEV-RO-19 DEV-RT-46 DEV-RO-20 DEV-RT-48 DEV-RT-50 DEV-RT-51 DEV-RT-52 DEV-RT-53 DEV-RT-55 DEV-RT-56 DEV-RT-57 DEV-RT-59 DEV-RT-60 DEV-RT-61 DEV-RT-62 DEV-RT-63	Dossier de sécurité et d'homologation de l'application. Dossier technique du projet de développement de l'application. Configuration des composants logiciels de l'application. Configuration de la journalisation conforme à l'instruction ministérielle. Configuration de la distribution du temps conforme à l'instruction ministérielle. Configuration de la plateforme de production ou de la structure d'hébergement. Suivi des bogues et des anomalies.
DEV-RO-22 DEV-RO-23 DEV-RT-65 DEV-RO-24 DEV-RT-66 DEV-RT-67 DEV-RO-25	Dossier de sécurité et d'homologation de l'application. Dossier technique du projet de développement de l'application. Suivi des tests et des résultats. Suivi des bogues et des anomalies.

ANNEXE IV.

DÉFINITIONS ET SIGLES.

API :	*Application programming interface* ou bibliothèque de fonctions, est un ensemble normalisé de fonctions et de services permettant à des logiciels d'accéder à des ressources informatiques externes (fichiers, base de données, services web, etc.).
ASLR :	*Adress space layout randomization* est une technique visant à répartir de manière aléatoire les zones de données d'un programme informatiques dans la mémoire des ordinateurs.
CWE :	*Common weakness enumeration* est une liste organisée des vulnérabilités rencontrées dans les logiciels. Cette liste est maintenue et mise à la disposition du public par l'organisation MITRE (organisation à but non lucratif spécialisée dans le conseil en matière de centre de recherche et de développement sponsorisé par le gouvernement fédéral) via le site https://cwe.mitre.org/.
Défense en profondeur :	Le principe de défense en profondeur consiste à déployer différents mécanismes complémentaires afin de prévenir, détecter, limiter bloquer une attaque via l'exploitation de vulnérabilités, et de réparer les éléments endommagées.
EBIOS :	La méthode d'Expression des besoins et d'identification des objectifs de sécurité est une méthodologie d'analyse de risques pesant sur les systèmes d'information, promue par l'Agence de sécurité des systèmes d'information
Framework :	le *framework* est un ensemble cohérent de composants logiciels et d'outils visant à faciliter la conception des applications informatiques ou des logiciels à partir d'un modèle architectural.
MOE :	la maîtrise d'œuvre assure la réalisation de l'application informatique ou du logiciel suivant les spécifications fonctionnelles et techniques.
OSSTM :	*Open source security testing methodology* est un projet libre de méthodologie d'audit de sécurité visant à fournir la documentation et les outils de réalisation de tests d'évaluation de sécurité pour les applications informatiques ou les logiciels.
OWASP :	Le projet libre de sécurisation des applications Web (*Open web applications security project*) pour objectif de fournir des méthodologies et des outils pour augmenter la sécurité des applications informatiques s'appuyant sur les technologies « Web ».
SQL :	*Structured query language* est un langage informatique normalisé servant à exploiter les bases de données relationnelles, à savoir la manipulation (recherche, ajout, suppression, etc.), la définition (structure et organisation) et le contrôle (autorisation d'accès, gestion des transactions) des données stockées dans la base.
SSH :	*Secure shell* vise à sécuriser la ligne de commande (*shell*) pour permettre son utilisation à distance. Le protocole SSH peut être aussi utilisé dans le cadre de la mise en œuvre de tunnel sécurisé (et authentifié) au profit de protocoles applicatifs (transfert de fichiers, interface graphique X11, etc.).
TLS :	*Transport layer security* est un protocole de sécurisation de la couche transport TCP (*Transmission control protocol*) du protocole d'interconnexion des réseaux IP (*Inter[connexion]net[work] protocol*).
URI :	*Uniform resource identifier* est un identifiant unique permettant de référencer une ressource informatique.
URL :	*Uniform resource locator* permet de désigner et d'accéder à une ressources référencée sur l'Internet : une fichier, une page Web, une photo, etc.