MITIGATE

MITIGATE

Multidimensional, IntegraTed, rIsk assessment framework and dynamic, collaborative risk manaGement tools for critical information infrAstrucTrurEs

www.mitigateproject.eu

Grant Agreement No.653212
Topic: H2020-DS-2014-01
Risk Management and Assurance Models
Innovation Action

Deliverable D4.2

Report on Standards and Regulations Compliance

Contractual Date of Delivery:	M15 / November 2016
Editor:	Armend Duzha (MAGG)
Work-package:	4
Distribution / Type:	PU (Public)
Version:	1.0
File:	Storage location / File-direction

Project co-funded by the European Union within the Horizon 2020 Programme

Abstract

Indicative text as an example (the final text will be relevant to the final context of the respective Deliverable)

Despite the importance of Critical Information Infrastructures (CIIs) and dynamic ICT-based maritime supply chains (SCs) for port operations, state-of-the-art Risk Management (RM) methodologies for maritime environments pay limited attention to cyber-security and do not adequately address security processes for international SCs. Motivated by these limitations, MITIGATE will introduce, integrate, validate and commercialize a novel RM system, which will empower stakeholders' collaboration for the identification, assessment and mitigation of risks associated with cyber-security assets and SC processes. This collaborative system will boost transparency in risk handling, while enabling the generation of unique evidence about risk assessment and mitigation. At the heart of the RM system will be an open simulation environment enabling stakeholders to simulate risks and evaluate risk mitigation actions. This environment will allow users to model, design, execute and analyze attack-oriented simulations. Emphasis will be paid on the estimation of cascading effects in SCs, as well as on the prediction of future risks. MITIGATE will be compliant with prominent security standards and regulations for the maritime sector (i.e. ISO27000, ISO28000, ISPS).

The MITIGATE system will be built based on readily available technologies of the partners, which will enable the project to produce a mature (high-TRL) system at an optimal value-for-money. The system will be validated based on real-life pilot operations across five EU ports (Bremen, Piraeus, Valencia, Ravenna, Livorno) with the active participation of over 500 users (security officers, terminal operators, facility operators, standardization experts and more). Also, the project's approach will be contributed as a blueprint to the NIS public-private platform. Finally, significant effort will be devoted to the commercialization of the MITIGATE system based on pragmatic business plans and market launch actions.

Executive Summary

The report reflects the work conducted in task T4.4 “Standards and Regulations Compliance”. In particular, the deliverable reports on the MITIGATE’s system compliance to various cybersecurity-related standards, frameworks, models, programs, best practices and initiatives (including ISO27001, ISO27005, ISO28000, ISPS and more). The examined existing security-related approaches were assessed against the following seven (7) domains covered by the MITIGATE approach:

1. Risk Management.
2. Asset, Change, and Configuration Management.
3. Threat and Vulnerability Management.
4. Situational Awareness.
5. Information Sharing and Communications.
6. Supply Chain and External Dependencies Management.
7. Cybersecurity Program Management.

According to the evaluation results, the developed system satisfies, implements and comports to different areas of the examined solutions.
In addition, in order for the system to keep pace with the increasing demand for usage of well-recognized vulnerability identifiers, it adopts specific industry standards (Common Vulnerabilities and Exposures (CVE) and Common Attack Pattern Enumeration and Classification (CAPEC)) for vulnerability and exposure names. In this vein, it takes into consideration and satisfies a set of criteria, recommendations and conditions imposed by these standards, however, it has to complete the CVE and CAPEC compatibility processes in order to be registered as CVE and/or CAPEC Compatible.
Finally, the report presents the MITIGATE software development and integration procedure adopted that has been adopted and will be followed throughout the entire life of the project. Also we presented the tools that were selected and are used to implement the defined development lifecycle.

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	16/09/2016	First Draft ToC	A. Duzha
0.2	22/09/2016	Revised ToC	N. Polemi
0.3	24/11/2016	Contribution to chapters 2, 3 & 4	S. Papastergiou, N. Polemi, P. Gouvas, C. Douligeris, A. Karantjias, K. Patsakis, G. Exarchou, S. Glykos, A. Duzha
0.4	25/11/2016	Introduction and conclusions	S. Papastergiou, N. Polemi
0.5	25/11/2016	Integrated document ready for internal review	S. Papastergiou, N. Polemi, P. Gouvas
1.0	30/11/2016	Final version of the Deliverable	S. Papastergiou, N. Polemi, P. Gouvas

Contributors

First Name	Last Name	Partner	Email
Armend	Duzha	MAGG	armend.duzha@maggioli.it
Nineta	Polemi	UPRC	dpolemi@gmail.com
Panagiotis	Gouvas	SILO	pgouvas@gmail.com
Athanasis	Karantjias	UPRC	thanos.karantjias@gmail.com
Spyros	Papastergious	UPRC	paps@unipi.gr
Constantinos	Patsakis	UPRC	kpatsak@unipi.gr
Christos	Douligeris	UPRC	cdouliq@unipi.gr
Stamatios	Glykos	UPRC	StamatisGlykos@gmail.com
Georgios	Exarchou	UPRC	gexarchou@gmail.com

Glossary

Acronym	Explanation
CAPEC	Common Attack Pattern Enumeration and Classification
CVE	Common Vulnerabilities Enumeration
SCS	Supply Chain Service

Table of Contents

List of Figures

List of Tables

1 Introduction

The MITIGATE project has introduced a risk assessment methodology targeting to deal with the interdependent cyber-related threats and risks of the maritime Supply Chain (SC) Services. The main objective of the methodology is to assist the SC operators to assess the risks and their possible effects deriving from an external security threat realized in an interdependent entity (e.g., ships, port authorities, maritime / insurance companies, customs, ship-industry) or other Critical Infrastructures (e.g. railroads, airports). Such effects are the result of the multiple, divergent interdependencies and interconnections that exist among the SC stakeholders which in many cases are complex, non-obvious and hard to identify.

On top of that, it should be noted that a set of acknowledged security standards, methodologies, best practices, frameworks, legal and regulatory regime (e.g. ISO 27001, ISPS code) impose specific obligations and rules that should also be taken into consideration for the effective addressing of all the cascading risks and effects.

1.1 Purpose

The main goal of this report is to check and ensure the implementation of security standards and regulations. Thus, we need to validate that specific requirements imposed by them have been taken into consideration in the design and development of the MITIGATE system

1.2 Scope

This document reports on the MITIGATE's system compliance to various cybersecurity-related widely-used standards, frameworks, models, programs, best practices and initiatives (including ISO27001, ISO27005, ISO28000, ISPS and more). Also, the report indicates the compatibility of MITIGATE system with the Common Vulnerabilities and Exposures (CVE) and the Common Attack Pattern Enumeration and Classification (CAPEC). Finally, the deliverable will specify the main steps and phases of the adopted software development and integration procedure.

1.3 Structure of the deliverable

Section 2 sets the main objectives and activities of the MITIGATE approach logically grouped in seven (7) domains and provides a mapping of a set cybersecurity standards, best practices, specifications and best practices to these seven (7) domains. Section 3 shows to what extent the MITIGATE system satisfies the requirements and recommendations defined by the Common Vulnerabilities and Exposures (CVE) and the Common Attack Pattern Enumeration and Classification (CAPEC)**.** Section 4 presents the MITIGATE software development and integration procedure adopted that guarantees and assures quality during the entire lifetime of the project. Finally Section 5 draws conclusions.

2 Standards and Regulations in Information Technology, Maritime & Supply Chain Security

2.1 Mitigate Security Domains

The MITIGATE system implements a targeted Supply Chain (SC) risk assessment approach, introduced in the Deliverable D2.2 [MITIGATE D2.2], suitable for the involved organizations' ICT infrastructures that is in accordance to their corresponding identified security needs, requirements, particularities and obstacles. By combining the knowledge gained from existing projects (e.g. S-PORT, CYSM, MEDUSA, SecureTropos) and coupling them with new game theory and simulation techniques and mathematical models for predicting and analyzing threats patterns, the produced system provides a sound decision making towards the effective and efficient security and risk management, guiding the SC business partners on analysing, assessing and managing organization-wise and interdependent effects, threats and risks.

In this context, the MITIGATE system offers a bundle of added-value security management services described in the Deliverables D3.1 [MITIGATE D3.1] and D3.2 [MITIGATE D3.2], contributing to:

- elicit, analyse, model and document multiple and divergent cyber interdependencies between the ports and the maritime-related entities (e.g. ships, port authorities, maritime / insurance companies, customs, ship-industry);
- generate common approaches in identifying, monitoring and handling common risks;
- lower common cyber related risks to acceptable levels, constantly identifying vulnerabilities and continuously informing the SC operators about the current and upcoming cyber threats;
- predict all possible vulnerabilities paths and patterns and measure their effectiveness and applicability;
- analyse and document threats rising from the interdependency of the ports with other maritime entities (e.g. other ports, maritime companies, railways) which may cause cascading effects and identify measures for reducing the produced risks;
- aid the SC business partners to deal with the danger of diffusing threats that come from their interconnections with various entities and critical infrastructures;
- determine the exploitation, resilience and reliability level of ports' supply chains;
- raise the cyber intelligence and culture and harmonize the corporate and SC cyber security practices, drawing a more security-aware strategy that can be incorporated into the existing business operation and logic, and can improve the trust in the maritime environment;
- increase predictability and reduce uncertainty of SC business operations by lowering cyber risks to definable and acceptable levels, continuously informing them on current and upcoming threats reducing the possibilities of operation and business disruption;
- provide information security awareness guiding the organizations on selecting security countermeasures that fit to their needs balancing the cost with the business benefits;

Thus, considering the main capabilities and aspects of the proposed approach, we have extracted and summarized a set of cybersecurity-related attributes, indicators and functions which have been logically grouped in seven (7) domains [C2M2 Model]. Each of the 7 domains contains a structured set of cybersecurity objectives that represents the activities required for establishing and ensuring increased capability in the domain. These domains will be used as reference points to check and evaluate the MITIGATE system's compliance to various selected standards and regulations (including ISO27001, ISO27005, ISO28000, ISPS and more).

A brief description of the 7 domains is presented in the following table.

Domain	Description
Risk Management	Establish, operate and maintain a cybersecurity risk management program to identify, analyze, and mitigate cybersecurity risks to the organization taking into consideration the related interconnected infrastructures, and stakeholders.
Asset, Change, and Configuration Management	Identify and manage all cyber assets which are necessary in the provision of the supported business processes and needed to be protected commensurate with the risk and impact resulting from various threats
Threat and Vulnerability Management	Identify, analyze, manage, and respond to cybersecurity threats and vulnerabilities commensurate with the risk to the involved ICT infrastructure and organizational objectives.
Situational Awareness	Collect, analyze, correlate, and use cybersecurity security and risk related information, including information retrieved from online repositories, to form the security state of the cyber assets.
Information Sharing and Communications	Establish and maintain relationships with internal and external entities which will reveal their commitment to identify all of their organizations' cyber assets, the controls they have undertaken and provide cybersecurity information, including threats and vulnerabilities
Supply Chain and External Dependencies Management	Identify, analyze, and mitigate the cybersecurity risks associated with assets that are dependent on other entities, commensurate with the risk to the involved ICT infrastructure and organizational objectives.
Cybersecurity Program Management	Establish and maintain an enterprise cybersecurity program that is aligned with the indentified risk to the examined infrastructure.

Table 1. Mitigate Security Domains

2.1.1 Risk Management

The Risk Management domain comprises three main objectives and includes a set of activities presented in Table 2:

1. **Establishment of a Cybersecurity Risk Management Strategy**: A cybersecurity risk management strategy includes a well-defined risk assessment methodology that aims to systematically evaluate the cyber risks affecting the examined ICT infrastructure. The risk assessment approach determine the value of the corporate assets and estimate the potential impact of threats in terms of specific criteria (availability confidentiality, integrity) and based on various organizational scenarios.
2. **Management of the Cybersecurity Risks**: This objective includes activities for identifying and assessing, responding to (e.g. accepting, mitigating), and monitoring risks. These should be performed in a manner that aligns with the business needs.
3. **Management of the Risk-related activities**: All risk management relevant activities are well-defined and documented.

Risk Management	
Establishment of a Cybersecurity Risk Management Strategy	A documented and well-defined cybersecurity risk management strategy exists
	An approach for effective risk prioritization considering various parameters including probability and/or impact of the risks.
	A number of criteria have been defined and proposed for evaluating and categorizing the operational risks
	The risk management strategy takes into account the new challenges of the threat landscape
	An risk taxonomy is documented and used in risk management activities
Management of the Cybersecurity Risks	Identification of the Cybersecurity risks
	Mitigation and handling of the identified risks
	The evaluation and management of the Cybersecurity risks are performed in accordance with the risk management strategy
	Risk monitoring is an integral part of the Cybersecurity Risk Management Strategy
	A structured repository of identified risks has been defined to support the risk management activities
Management of the Risk-related activities	All risk management activities are well-defined and documented
	Stakeholders/Business Partners/Resources involved in the risk management activities have been identified

	Risk management approach include compliance requirements for specified standards and/or guidelines

Table 2. Risk Management Objectives and Activities

2.1.2 Asset, Change, and Configuration Management

The Asset, Change, and Configuration Management domain comprises four objectives and includes a set of activities presented in Table 3:

1. **Asset Inventory Management:** Formulation of an inventory of assets required for the provision of the supported business processes.
2. **Asset Security Configuration**: Identification and documentation of the security controls and measures that can reduce the identified risks and minimize the corresponding consequences.
3. **Asset Changes Management**: Activities related to analyzing changes to assets in order to ensure they do not introduce unacceptable vulnerabilities into the examined ICT infrastructure, ensuring all changes follow the change management process, and identifying unauthorized changes.
4. **Asset Management Activities**: All asset management relevant activities are well-defined and documented.

Asset, Change, and Configuration Management	
Asset Inventory Management	An inventory of cyber assets required for the provision of the Supply Chain processes and services.
	An inventory for all cyber assets required for the provision of the Supply Chain processes and services.
	Inventory attributes of the cyber (e.g., asset name, Vendor Name, product, version, location, run privilege)
	Types of dependencies that exist between the cyber assets
	Existing interconnections that exist between the cyber assets
Asset Security Configuration	Establishment of an inventory of security controls applicable to the cyber assets
	Security controls to prevent, detect, counteract or minimize consequences of threats against cyber assets
Asset Changes Management	The changes (deployment of new security controls) to the cyber assets are evaluated in terms of their impact before being implemented
	The changes (deployment of new security controls) to the cyber assets are evaluated in terms of their effectiveness before being implemented
Asset Management	All asset management activities are documented

Activities	Stakeholders/Business Partners/Resources involved in the asset management activities have been identified
	Asset management approach include compliance requirements for specified standards and/or guidelines

Table 3. Asset, Change, and Configuration Management Objectives and Activities

2.1.3 Threat and Vulnerability Management

The Threat and Vulnerability Management domain comprises three objectives and includes a set of activities presented in Table 4:

1. **Threat Management**: Identification of threats relevant to the examined ICT infrastructure. These activities include the collection and analysis of threat information from online repositories and other trusted sources interpreting that information in the context of the risk analysis and management process.
2. **Vulnerability Management**: Vulnerability analysis begins with analyzing information gathered from different sources (e.g. automatic scanning tools, online repositories and web sites) in order to infer the exposure of each asset to various risk factors. In addition, appropriate mitigating controls should be proposed to resolve the indentified security and cover the corresponding weaknesses.
3. **Threat and Vulnerability Management Activities:** All threat and vulnerability management relevant activities are well-defined and documented.

Threat and Vulnerability Management	
Threat Management	Identification of threats related with the cyber Assets
	An inventory of threat related with the cyber Assets
	Cybersecurity threat information is retrieved from the online databases (e.g. https://web.nvd.nist.gov, http://www.cvedetails.com)
	Measurement of the severity of the threats in terms of expected frequency of appearance (based on the history of previous incidents)
	Measurement of the severity of the threats based on the participants' intuition and knowledge.
	Measurement of the severity of the threats based on information retrieved from social media and existing repositories.
	Threats are addressed according to their severity
	Identified threats are analyzed and prioritized

Vulnerability Management	Identification of vulnerabilities related with the cyber Assets
	Known vulnerabilities are retrieved from the online databases (e.g. https://web.nvd.nist.gov, http://www.cvedetails.com)
	vulnerability assessments are performed
	Vulnerabilities are addressed (e.g. deployment of mitigating controls, enforcement of cybersecurity patches) according to their severity
	Identified vulnerabilities are analysed and prioritized
	Validation of the effectiveness of the proposed security controls (e.g., deployment of patches) to respond to the identified vulnerabilities.
Threat and Vulnerability Management Activities	All threats and vulnerabilities management activities are documented
	Stakeholders/Business Partners/Resources involved in the threats and vulnerabilities management activities have been identified
	Threat and vulnerability management approach include compliance requirements for specified standards and/or guidelines

Table 4. Threat and Vulnerability Management Objectives and Activities

2.1.4 Situational Awareness

The Situational Awareness domain comprises two objectives and includes a set of activities presented in Table 5:

1. **Establishment and Maintenance of an appropriate Security Picture**: Activities that provide accurate knowledge of the dynamic ICT environment under examination. The security state of all cyber assets should be defined and communicated to all responsible operators.
2. **Situational Awareness Management Activities**: All threat and vulnerability management relevant activities are well-defined and documented.

Situational Awareness Management	
Establishment and Maintenance of an appropriate Security Picture	Evaluation of the current security state of the cyber assets
	Communication of the current security state of the cyber assets to the appropriate stakeholders.
	Aggregation of the security and risk related information to provide an understanding of the security state of the cyber assets
	Information from the online databases (e.g. https://web.nvd.nist.gov, http://www.cvedetails.com) is collected to derive and infer the security state of the cyber assets

	Comparison of different states of the cyber assets to determine whether security progress has taken place.
Situational Awareness Management Activities	All Situational Awareness management activities are well-defined and documented
	Stakeholders/Business Partners/Resources involved in the Situational Awareness management activities have been identified

Table 5. Situational Awareness Management Objectives and Activities

2.1.5 Information Sharing and Communications

The Information Sharing and Communications domain comprises three objectives and includes a set of activities presented in Table 6:

1. **Cybersecurity Information Sharing**: Activities that promote sharing of risk-related information among all the internal and external operators and organizations by establishing the principles and maintaining an appropriate framework for interaction among them. The aim of these activities is to strengthen cybersecurity and improve the security posture of the involved entities.
2. **Cybersecurity Information Sharing Management Activities**: All Cybersecurity Information Sharing management relevant activities are well-defined and documented.

Information Sharing and Communications Management	
Cybersecurity Information Sharing	Security and risk related Information is collected from and provided to the organization and entities involved in the risk assessment
	Principles have been established and maintained to enable secure sharing of sensitive security and risk related Information
	Technical repositories have been identified that serves as primary source of consultation on cybersecurity issues
	Information-sharing stakeholders and entities are identified based on shared interest in and risk to the examined infrastructure.
	Specific principles have been defined for the effective sharing and exchange of security and risk related Information
	A network of internal trust relationships among all entities and organizations has been established to validate the provided security and risk related Information
Cybersecurity Information Sharing Management Activities	All Cybersecurity Information Sharing management activities are well-defined and documented
	Stakeholders/Business Partners/Resources involved in the Cybersecurity Information Sharing management activities have been identified

Table 6. Information Sharing and Communications Management Objectives and Activities

2.1.6 Supply Chain and External Dependencies Management

The Supply Chain and External Dependencies Management domain comprises three objectives and includes a set of activities presented in Table 7:

1. **Dependencies Identification**: Identification of the interdependencies and interconnections among the involved ICT infrastructures, establishing and maintaining a comprehensive understanding of key relationships.
2. **Dependency Risk Management:** Identification and measurement of combined interdependent vulnerabilities/threats paths and patterns arising from the interconnections that exist between the organizations (e.g., ships, port authorities, maritime / insurance companies, customs, ship-industry) managing the associated cybersecurity risks.
3. **Dependency Risk Management Activities:** All dependency risk management relevant activities are well-defined and documented.

Supply Chain and External Dependencies Management	
Dependencies Identification	All cyber dependencies that exist between the organizations and entities are identified
	All interconnections between the cyber assets are identified
	All cyber dependencies and interconnections are grouped according to their nature, effects and functionality
Dependency Risk Management	Cybersecurity risks due to the existing dependencies and interconnections (at asset and organization level) are identified and addressed
	Mitigation and handling of the identified dependency risks
	The evaluation and management of the Cybersecurity dependency risks are performed in accordance with the risk management strategy
	Supply chain threats are identified and assessed taking into consideration security-related information collected from online repositories.
	Cybersecurity dependency risk monitoring is an integral part of the Cybersecurity Risk Management Strategy
	Agreements among the involved organizations and entities include cybersecurity requirements
	Contracts and agreements among the involved organizations and entities incorporate sharing of cybersecurity threat and risk information
	Cybersecurity requirements (regarding impact, threat and vulnerability assessment) are established for all identified dependencies based on the risk management strategy
Dependency	All cybersecurity dependency risk management activities are well-defined and

Risk Management Activities	documented
	Stakeholders/Business Partners/Resources involved in the cybersecurity dependency risk management t activities have been identified
	Dependency risk management approach include compliance requirements for specified standards and/or guidelines

Table 7. Supply Chain and External Dependencies Management Objectives and Activities

2.1.7 Cybersecurity Program Management

The Cybersecurity Program Management domain includes activities (see Table 8) regarding the Establishment of a Cybersecurity Strategy Program. This program should take into account priorities aligned to the objectives of the adopted risk analysis and mitigation process.

Workforce Management	
Establishment of a Cybersecurity Strategy Program	A documented and well-defined cybersecurity strategy exist
	The cybersecurity program strategy are aligned with the indentified risk to the examined infrastructure
	A cybersecurity architecture is used to inform risk analysis
	The Cybersecurity strategy takes into account the new challenges of the threat landscape

Table 8. Cybersecurity Program Management Objectives and Activities

2.2 Cyber-related standards Compliance

The MITIGATE system should cover the main aspects of widely-used standards, frameworks, models, programs, best practices and initiatives. To this end, we studied and analyzed the capabilities and functions of a set of existing security-related approaches in order to check to what extent the system satisfies the requirements and rules imposed by them. The following table provides a mapping of the examined cybersecurity standards to the seven (7) domains covered by the MITIGATE approach as these defined in the previous section. In particular, this mapping shows the areas of the standards which the MITIGATE system implements.

Prerequisites	Risk Management	Asset, Change, and Configuration Management	Threat and Vulnerability Management	Situational Awareness	Information Sharing and Communications	Supply Chain and External Dependencies Management	Cybersecurity Program Management
ISO 27001:2013 Information Technology – Security techniques – Information security management systems requirements [ISO/IEC 27001:2005]		●	●	●	●		●
ISO 27002:2005 Information technology – Security techniques – Code of practice for information security management [ISO/IEC 27002:2005]		●	●	●	●		●
ISO 27005:2011 International Organization for Standardization. (2011). Information security risk management (ISO 27005:2011) [ISO 27005:2011]	●						●
ISO/IEC 21827:2008 International Organization for Standardization. (2008). Systems Security Engineering – Capability Maturity Model (SSE-CMM) (ISO/IEC 21827:2008). [ISO/IEC 21827:2008]			●		●		●

ISO 28001:2007 International Organization for Standardization. (n.d.). Security management systems for the supply chain - Best practices for implementing supply chain security, assessments and plans - Requirements and guidance (ISO/ IEC20001:2007). [ISO 28001:2007]						●	●
Security considerations in the information system development life cycle. [NIST Security Considerations in SDLC]	●						●
Information security training requirements: A role- and performance-based model [NIST SP800-16]							●
Guide for applying the risk management framework to federal information systems [NIST SP800-37]	●			●	●	●	●
Creating a patch management and vulnerability management program [NIST SP800-40]			●				
Recommended security controls for federal information systems and organizations [NIST SP800-53]	●	●	●		●		●
Computer security incident handling guide [NIST SP800-61]							●
Security considerations in the system development life cycle [NIST SP800-64]			●				●
Guide for security-focused configuration management of information systems [NIST SP800-128]		●					●
Information security continuous monitoring (ISCM) for federal information systems and organizations				●	●	●	●

[NIST SP800-137]							
National vulnerability database. [NIST NVD]	●		●	●	●		
Piloting supply chain risk management for federal information systems [NISTIR 7622]						●	●
Guidelines for smart grid cyber security: Vol. 1, smart grid cyber security strategy, architecture, and high-level requirements [NISTIR 7628]	●	●					●
Guidelines for smart grid cyber security: Vol. 3, Supportive analyses and references [NISTIR 7628]			●		●		●
Organisation for Economic Co-operation and Development [OECD Reducing Systemic Cybersecurity Risk]				●		●	
Key practices of the capability maturity model [SEI CMM]							●
Generic SCADA risk management framework for Australian critical infrastructure. [SCADA AU RMF]	●						●
[Situation Awareness in Dynamic Systems] Endsley, M. (1995). Toward a theory of situation awareness in dynamic systems. Human Factors, pp. 32-64.				●	●		●
Supply chain risk management awareness. Armed Forces Communication and Electronics Association Cyber Committee. [Supply Chain Risk Management Awareness]	●			●		●	●

Table 9. Cyber-related standards Compliance

3 Standards for Information Security Vulnerabilities

3.1.1 Common Vulnerabilities and Exposures (CVE) – Requirements and Recommendations for CVE Compatibility

The Common Vulnerabilities and Exposures (https://cve.mitre.org/) is a central repository that provides information for publicly known security vulnerabilities and exposures. This initiative runs a program named “Common Vulnerabilities and Exposures Compatibility Program” [CVE Compatibility] which has defined a list of requirements and recommendations against which any tool, service, Web site, database, or system that uses the CVE-related information can be evaluated in order to declare they are CVE-Compatible.

The defined requirements and recommendations fall into the following categories:

- **High-Level Requirements**: These are the high-level requirements for all capabilities. Many of them are described in detail in later sections.
- **Accuracy**: CVE compatibility only facilitates data sharing if the capability’s mapping is accurate. Therefore, CVE-compatible capabilities must meet minimum accuracy requirements.
- **Documentation**: The following requirements apply to documentation that is provided with the capability.
- **CVE Date Usage**: Users must know how "up-to-date" a capability’s repository is with respect to its mapping to CVE. The capability owner needs to indicate the currency of a mapping by providing the date of its last update of CVE information and indicate what portion of CVE content they utilize and where they gather the CVE content from.
- **Different Styles of CVE Name Support**: A capability MUST function with CVE names independent of the format of the CVE name’s representation in the capability, whether it is using the older style four-digit CVE ID Syntax or the, four-digit or higher-digit CVE ID Syntax (used after the CVE ID Syntax modification in use after 31 December 2013).
- **Revocation of CVE Compatibility**
- **Review Authority**
- **Appendix A: Type-Specific Requirements**: Since a wide variety of capabilities use CVE, certain types of capabilities may have unique features that require special attention with respect to CVE compatibility.
- **Appendix B: Media Requirements**

The following table shows to what extent the MITIGATE system satisfies the proposed CVE requirements.

High-Level Requirements		
Prerequisites		
2.1	The Owner MUST be a valid legal entity, i.e., an organization or a specific individual, with a valid phone number, email address, and street mail address.	N/A
2.2	The capability MUST provide additional value or information beyond that which is provided in CVE itself (i.e., name, description, references, and associated data).	✓
2.3	The Owner MUST provide the Review Authority with a technical point of contact who is qualified to answer questions related to the mapping and any CVE-related functionality of the capability.	✓
2.4	The capability MUST be available to the public, or to a set of consumers, in a production version.	✓
2.5	The Owner MUST provide the Review Authority with a completed "CVE Compatibility Requirements Evaluation Form."	✓
2.6	For a capability with a Repository, the Owner MUST provide the Review Authority with free access to the Repository so that the Authority can determine that the Repository satisfies all associated requirements.	✓
2.7	For a capability with a Repository, the Owner MUST allow the Review Authority to use the Repository to identify any vulnerabilities that must be added to CVE.	✓
2.8	The Owner MUST agree to abide by all of the mandatory CVE Compatibility Requirements, which includes the mandatory requirements for the specific type of capability.	✓
Functionality		
2.9	The capability MUST allow users to locate security elements using CVE names ("CVE-Searchable").	✓
2.10	When the capability presents security elements to the user, it MUST allow the user to obtain the associated CVE names ("CVE-Output").	✓
2.11	For a capability with a Repository, the capability's mapping MUST accurately link security elements to the appropriate CVE names ("Mapping Accuracy").	✓
2.12	The capability's documentation MUST adequately describe CVE, CVE compatibility, and how the CVE-related functionality in the capability is used ("CVE-Documentation").	✓
2.13	The capability MUST state the date of its currency with respect to CVE ("Date Usage")	✓
2.14	The capability MUST satisfy any additional requirements for the specific type of capability, as specified in Appendix A.	✓
2.15	The capability MUST satisfy all requirements for its distribution media, as specified in Appendix B.	✓
2.16	The capability is NOT REQUIRED to do any of the following: • use the same descriptions or references as CVE	✓

	• include every CVE name in its repository	
	Miscellaneous	
2.17	If the capability does not satisfy all requirements, then the Owner MUST NOT advertise that it is CVE-compatible.	N/A
	Accuracy	
3.1	For a capability with a Repository, the Repository MUST have an Accuracy Percentage of 90 percent or greater.	N/A
3.2	During the review period, the Owner MUST correct any mapping errors found by the Review Authority.	N/A
3.3	After the review period, the Owner SHOULD correct a mapping error within a reasonable time frame after the error was initially reported, i.e., within six (6) months for tools and three (3) months for on-line capabilities and services.	N/A
3.4	For a capability with a Repository, the Owner SHOULD prepare and sign a statement that, to the best of the Owner's knowledge, there are no errors in the mapping.	N/A
3.5	If the capability is based on, or uses, another CVE-compatible capability (the "Source" capability), and the Owner becomes aware of mapping errors in the Source capability, then the Owner MUST report those errors to the Owner of the Source capability.	N/A
3.6	The mapping accuracy for Advisory archives MUST be performed against all of the security elements of the archive repository subsequent to, and including, the archive's first use of a CVE name in a security element.	N/A
3.7	A capability MUST accurately reflect the status of deprecated CVE names within three (3) months for on-line capabilities and services.	N/A
	Documentation	
4.1	The documentation MUST include a brief description of CVE and CVE compatibility, which can be based on verbatim portions of documents from the CVE Web site.	✓
4.2	The documentation MUST describe how the user can find individual security elements in the capability's repository by using CVE names.	✓
4.3	The documentation MUST describe how the user can obtain CVE names from individual elements in the capability's repository.	✓
4.4	If the documentation includes an index, then it SHOULD include references to CVE-related documentation under the term "CVE."	✓
	CVE Date Usage	
5.1	Each new version of the capability MUST identify the most recent date of CVE content that was used in creating or updating the mapping through at least one of the following: change logs, new feature lists, help files, or some other mechanism. The capability is "up-to-date" with respect to that date.	✓
5.2	Each new version of the capability MUST be up-to-date with respect to a stated CVE date that is no more than three (3) months before the capability was made available to its users. If	✓

	a capability does not satisfy this requirement, then it is "out-of-date."	
5.3	The Owner MUST publicize how quickly it will update the capability's repository to include new CVE information.	✓
5.4	The Owner MUST describe the criteria and mechanism for selecting the CVE information they include in their capability.	✓
5.5	The Owner MUST describe where it gathers new CVE content from.	✓
Different Styles of CVE Name Support		
6.1	If a user performs a search using YYYY	✓
6.2	If the Capability contains the CVE name CVE	✓
Revocation of CVE Compatibility		
7.1	If a Review Authority has verified that a Capability is CVE-compatible, but at a later time the Review Authority has evidence that the requirements are not being met, then the Review Authority MAY revoke its approval.	N/A
7.1.1	The Review Authority MUST identify the specific requirements that are not being met.	N/A
7.2	The Review Authority MUST determine if the actions or claims of the Owner are "intentionally misleading."	N/A
7.2.1	The Review Authority MAY interpret the phrase "intentionally misleading" as it wishes.	N/A
7.3	Unless recommended by two CVE Editorial Board members who do not have a conflict of interest, the Review Authority SHOULD NOT consider revoking CVE compatibility for a particular Capability more often than once every six (6) months.	N/A
Warning and Evaluation		
7.4	The Review Authority MUST provide the Capability Owner and Technical POC with a warning of revocation at least two (2) months before revocation is scheduled to occur.	N/A
7.4.1	If the Review Authority has found that the Owner's actions or claims are intentionally misleading, then the Review Authority MAY skip the warning period.	N/A
7.5	If the Owner believes that the requirements are being met, then the Owner MAY respond to the warning of revocation by providing specific details that indicate why the Capability meets the requirements under question.	N/A
7.6	If the Owner modifies the Capability so that it complies with the requirements in question during the warning period, then the Review Authority SHOULD end the revocation action for the Capability.	N/A
Revocation		
7.7	The Review Authority MAY delay the date of revocation.	N/A
7.8	The Review Authority MUST publicize that CVE compatibility has been revoked for the capability.	N/A
7.9	If the Review Authority finds that the Owner's actions with respect to CVE compatibility requirements are intentionally misleading, then revocation SHOULD last a minimum of one year.	N/A

7.10	The Review Authority MAY publicize the reason for revocation.	N/A
7.11	If the approval is revoked, the Owner MUST NOT apply for a new review during the period of revocation.	N/A
Review Authority		
8.1	The Review Authority MUST review the capability for CVE compatibility with respect to a specific CVE content date, i.e., the Review Date.	N/A
8.2	The Review Authority MUST clearly identify the Review Date that was used to determine compatibility for the capability.	N/A
8.3	The Review Authority MUST clearly identify the version of the CVE compatibility requirements document that was used to determine compatibility for the capability.	N/A
8.4	The Review Authority MUST define and publish a Sample Size.	N/A
8.4.1	The Review Authority SHOULD use a Sample Size of 50 elements plus 5 percent of the capability's repository, up to a maximum Sample Size of 400 elements.	N/A
8.4.2	The Review Authority MAY review every element in the capability's repository.	N/A
8.5	The Review Authority MUST publicize the Sampling Method.	N/A
8.6	The Review Authority MAY use a Review Sample that was not randomly selected.	N/A
8.7	The Review Authority MUST use the same Sampling Method and Sample Size for all capabilities that are evaluated within the same time frame.	N/A
Appendix A: Type-Specific Requirements		
A.1	The Capability MUST satisfy all additional requirements that are related to the specific type of capability.	X
A.1.1	If the Capability is a vulnerability assessment scanner, intrusion detection system (IDS), or a product which integrates the results of one or more scanners and IDSs, then it must satisfy the Tool Requirements, A.2.1 - A.2.8.	X
A.1.2	If the Capability is a service (such as a managed intrusion detection and response service, or a remote scanning service) then it must satisfy the Security Service Requirements, A.3.1 - A.3.5.	N/A
A.1.3	If the Capability is an online vulnerability or signature database, Web-based archive, or maintenance/patch site, then it must satisfy the Online Capability Requirements, A.4.1 - A.4.3.	N/A
A.1.4	If the Capability is an aggregation tool like a security information manager, a compliance reporting tool, or a service supplying these types of aggregations of vulnerability type information, then it must satisfy the Aggregation Capability Requirements, A.5.1 - A.5.6.	✓
Tool Requirements		
A.2.1	The Tool MUST allow the user to use CVE names to locate associated Tasks in that Tool ("CVE-Searchable") by providing at least one of the following: a "find" or "search" function, a mapping between that Tool's Task names and CVE names, or another mechanism.	N/A
A.2.2	For any report that identifies individual security elements, the Tool MUST allow the user to determine the associated CVE names for those elements ("CVE-Output") by doing at least one of the following: including CVE names directly in the report, providing a mapping between the Tool's Task names and CVE names, or using some other mechanism.	N/A
A.2.3	Any required reports or mappings MUST satisfy the media requirements as specified in Appendix B.	N/A
A.2.4	The Tool, or the Owner, SHOULD provide the user with a list of all CVE names that are associated with the Tool's Tasks.	N/A

A.2.5	The Tool SHOULD allow the user to select a set of Tasks by providing a file that contains a list of CVE names.	N/A
A.2.6	The interface of the Tool SHOULD allow the user to browse, select, and deselect a set of Tasks by using individual CVE names.	N/A
A.2.7	If the Tool does not have a Task that is associated with a CVE name as specified by the user in the A.2.5 or A.2.6 Tool requirements, then the Tool SHOULD notify the user that it cannot perform the associated Task.	N/A
A.2.8	The Owner MUST warrant that (1) the rate of false positives is less than 100 percent, i.e., if the Tool reports a specific security element, it is at least sometimes correct, and (2) the rate of false negatives is less than 100 percent, i.e., if an event occurs that is related to a specific security element, then sometimes the Tool reports that event.	N/A
Security Service Requirements Security services might use CVE-compatible tools in their work, but they may not provide their customers with direct access to those tools. Thus it could be difficult for customers to identify and compare the capabilities of different services. The Security Service Requirements address this potential limitation.		
A.3.1	The Security Service MUST be able to use CVE names to tell a user which security elements are tested or detected by the service ("CVE-Searchable") by doing one or more of the following: providing the user with a list of CVE names that identify the elements that are tested or detected by that Service, providing the user with a mapping between the Service's elements and CVE names, responding to a user-supplied list of CVE names by identifying which of the CVE names are tested or detected by the Service, or by using some other mechanism.	N/A
A.3.2	For any report that identifies individual security elements, the Service MUST allow the user to determine the associated CVE names for those elements ("CVE-Output") by doing one or more of the following: allowing the user to include CVE names directly in the report, providing the user with a mapping between the security elements and CVE names, or by using some other mechanism.	N/A
A.3.3	Any required reports or mappings that are provided by the Service MUST satisfy the media requirements as specified in Appendix B.	N/A
A.3.4	If the Service provides the user with direct access to a product that identifies security elements, then that product SHOULD be CVE-compatible.	N/A
A.3.5	The Owner MUST warrant that (1) the rate of false positives is less than 100 percent, i.e., if a Tool reports a specific security element, it is at least sometimes correct, and (2) the rate of false negatives is less than 100 percent, i.e., if an event occurs that is related to a specific security element, then sometimes the Service reports that event.	N/A
Online Capability Requirements		
A.4.1	The Online Capability MUST allow a user to find related security elements from the Online Capability's repository ("CVE-Searchable") by providing one of the following: a search function with returns CVE names for related elements, a mapping that links each element with its associated CVE name(s), or some other mechanism.	N/A
A.4.1.1	The Online Capability SHOULD provide a URL "template" that allows a computer program to easily construct a link that accesses the search function as outlined in Online Capability Requirements A.4.1. Examples: http://www.example.com/cgi-bin/db-search.cgi?cvename=CVE-YYYY-NNNN http://www.example.com/cgi-bin/db-search.cgi?cvename=CVE-YYYY-NNNNN http://www.example.com/cgi-bin/db-search.cgi?cvename=CVE-YYYY-NNNNNN http://www.example.com/cve/CVE-YYYY-NNNN.html http://www.example.com/cve/CVE-YYYY-NNNNN.html http://www.example.com/cve/CVE-YYYY-NNNNNN.html	N/A
A.4.1.2	If the URL template is for a CGI program, the program SHOULD accept the HTTP "GET" method.	N/A

A.4.2	For any report that identifies individual security elements, the Online Capability MUST allow the user to determine the associated CVE names for those elements ("CVE-Output") by doing at least one of the following: by allowing the user to include CVE names directly in the report, providing the user with a mapping between the security elements and CVE names, or by some other mechanism.	N/A
A.4.3	If the Online Capability does not provide details for individual security elements, then the Online Capability MUST provide a mapping that links each element with its associated CVE name(s).	N/A
Aggregation Capability Requirements		
A.5.1	The Aggregation capability MUST allow the user to use CVE names to locate associated elements in that capability ("CVE-Searchable") by providing at least one of the following: a "find" or "search" function, a mapping between that capability's names and CVE names, or another mechanism with the approval of the Review Authority.	✓
A.5.2	For any report that identifies individual security elements, the Aggregation capability MUST allow the user to determine the associated CVE names for those elements ("CVE-Output") by doing at least one of the following: including CVE names directly in the report, providing a mapping between the capability's names and CVE names, or using some other mechanism.	✓
A.5.3	Any required reports or mappings MUST satisfy the media requirements as specified in Appendix B.	✓
A.5.4	The Tool, or the Owner, SHOULD provide the user with a list of all CVE names that are associated with the Tool's Tasks.	✓
A.5.5	The Tool SHOULD allow the user to select a set of Tasks by providing a file that contains a list of CVE names.	✓
A.5.6	The interface of the Tool SHOULD allow the user to browse, select, and deselect a set of Tasks by using individual CVE names.	✓
Appendix B: Media Requirements		
B.1	The distribution media that is used by a CVE-compatible capability MUST use a media format that is covered in this appendix.	v
B.2	The media format MUST satisfy the specific requirements for that format.	X
Electronic Documents (HTML, word processor, PDF, ASCII text, etc.)		
B.3.1	The document MUST be in a commonly available format that has readers which support a "find" or "search" function ("CVE-Searchable"), such as raw ASCII text, HTML, or PDF.	✓
B.3.2	If the document only provides short names or titles for individual elements, then it MUST list the CVE names that are related to those elements ("CVE-Output").	✓
B.3.3	The document SHOULD include a mapping from elements to CVE names, which lists the appropriate pages for each element.	✓
Graphical User Interface (GUI)		
B.4.1	The GUI MUST provide the user with a search function that allows the user to enter a CVE name and retrieve the related elements ("CVE-Searchable").	✓
B.4.2	If the GUI lists details for an individual element, then it MUST list the CVE name (or names) that map to that element ("CVE-Output"). Otherwise, the GUI MUST provide the user with a mapping in a format that satisfies the B.3.1 Electronic Documents requirement.	✓
B.4.3	The GUI SHOULD allow the user to export or access CVE-related data in an alternate format that satisfies the B.3.1 Electronic Documents requirement.	✓

Table 10. Requirements and Recommendations for CVE Compatibility

3.1.2 Common Attack Pattern Enumeration and Classification (CAPEC) – Requirements and Recommendations for CAPEC Compatibility

Common Attack Pattern Enumeration and Classification (CAPEC) (https://capec.mitre.org/) provides a taxonomy of known threats and attacks that can be used to better understand their impact and to facilitate their mitigation. The Compatibility Program [CAPEC Compatibility] of this initiative has specified rules and requirements that a product, service or systems has to fulfill in order to be registered as "CAPEC-Compatible".

The CAPEC requirements and recommendations have been grouped as follows:

- **High-Level Requirements**: The following items define the concepts, roles, and responsibilities related to the proper use of CAPEC Identifiers to share data across separate security analysis, security testing, security operations and security management capabilities (tools, repositories, services, and standards) to allow these capabilities to be used together, and to facilitate the comparison of security-relevant tools and services.
- **Accuracy**: CAPEC compatibility only facilitates data sharing and correlation if the capability's mapping is accurate. Therefore, CAPEC-compatible capabilities must meet the following minimum accuracy requirements.
- **Documentation**: The following requirements apply to documentation that is provided with the capability.
- **CAPEC Version Usage**: Users must know what version of CAPEC is used in a capability's repository with respect to its mapping to CAPEC. The capability owner can indicate the currency of a mapping by referencing the relevant CAPEC version and optionally, the date the mapping was updated.
- **Revocation of CAPEC Compatibility**
- **Review Authority**
- **Appendix A**: **Type-Specific Requirements**: Since a wide variety of capabilities use CAPEC, certain types of capabilities may have unique features that require special attention with respect to CAPEC compatibility.
- **Appendix B: Media Requirements**

The following table shows the compatibility of the MITIGATE system with the CAPEC requirements and recommendations.

High-Level Requirements		
Prerequisites		
2.1	2.1) The capability owner MUST be a valid legal entity, i.e., an organization or a specific individual, with a valid phone number, email address, and street mail address.	N/A
2.2	The capability MUST provide additional value or information beyond that which is provided in CAPEC itself (i.e., name, description, risks, references, and associated weakness information).	✓
2.3	The capability owner MUST provide the Review Authority with a technical point of contact who is qualified to answer questions related to the mapping accuracy and any CAPEC-related functionality of the capability.	✓

2.4	The capability MUST be available to the public, or to a set of consumers, in a production or public version.	✓
2.5	For CAPEC compatibility the capability owner MUST provide the Review Authority with a completed "CAPEC compatibility Requirements Evaluation Form."	✓
2.6	The capability owner MUST provide the Review Authority with free access to the Repository so that the Authority can determine that the Repository satisfies all associated mapping accuracy requirements.	✓
2.7	The capability owner MUST allow the Review Authority to use the Repository to identify any attack pattern that should be added to CAPEC.	✓
2.8	The capability owner MUST agree to abide by all of the mandatory CAPEC compatibility Requirements, which includes the mandatory requirements for the specific type of capability.	✓
Functionality		
2.9	For CAPEC compatibility the capability MUST allow users to locate security elements using CAPEC identifiers ("CAPEC-Searchable").	✓
2.10	For CAPEC compatibility when the capability presents security elements to the user, it MUST allow the user to obtain the associated CAPEC identifiers ("CAPEC-Output").	✓
2.11	For CAPEC compatibility the capability's mapping MUST accurately link security elements to the appropriate CAPEC identifiers ("Mapping Accuracy").	✓
2.12	For CAPEC compatibility the capability's documentation MUST adequately describe CAPEC, CAPEC compatibility, and how the CAPEC-related functionality in the capability is used ("CAPEC-Documentation").	✓
2.13	For CAPEC compatibility the capability's publicly available documentation MUST explicitly list the CAPEC identifiers that the capability owner considers the capability to cover as part of its functionality ("CAPEC-Coverage").	✓
2.14	For CAPEC compatibility the capability's publicly available web site SHOULD provide the capability's CAPEC-Coverage as a CAPEC Coverage Claim Representation (CCR) XML document(s).	✓
2.15	The capability MUST denote the dated CAPEC version used ("Version Usage").	✓
2.16	The capability MUST satisfy any additional requirements for the specific type of capability, as specified in Appendix A.	✓
2.17	The capability MUST satisfy all requirements for its distribution media, as specified in Appendix B.	✓
2.18	The capability is NOT REQUIRED to do any of the following: • use the same descriptions or references as CAPEC • include every CAPEC identifier in its repository	✓
Miscellaneous		
2.19	2.19) If the capability does not satisfy all of the applicable requirements above (2.1 through 2.18), then the capability owner shall not advertise that it is CAPEC-compatible.	N/A

Accuracy		
3.1	The Repository MUST have an accuracy of 100 percent.	N/A
3.2	During the review period, the capability owner MUST correct any mapping errors found by the Review Authority.	N/A
3.3	After the review period, the capability owner SHOULD correct a mapping error within a reasonable time frame after the error was initially reported, i.e., within two (2) versions of the capability repository or six (6) months, whichever is shorter.	N/A
3.4	The capability owner SHOULD prepare and sign a statement that, to the best of the capability owner's knowledge, there are no errors in the mapping.	N/A
3.5	If the capability is based on, or uses, another CAPEC-compatible capability (the "Source" capability), and the capability owner becomes aware of mapping errors in the Source capability, then the capability owner MUST report those errors to the capability owner of the Source capability.	N/A
Documentation		
4.1	The documentation MUST include a brief description of CAPEC and CAPEC compatibility, which can be based on verbatim portions of documents from the CAPEC Web site.	✓
4.2	4.2) The documentation MUST describe how the user can find individual security elements in the capability's repository by using CAPEC identifiers.	✓
4.3	The documentation MUST describe how the user can obtain CAPEC identifiers from individual elements in the capability's repository.	✓
4.4	If the documentation includes an index, then it SHOULD include references to CAPEC-related documentation under the term "CAPEC."	✓
CAPEC Version Usage		
5.1	The capability MUST identify the CAPEC version or update date that was used in creating or updating the mapping through at least one of the following: change logs, new feature lists, help files, or some other mechanism. The capability is "up-to-date" with respect to that version or update date.	✓
5.2	Each new version of the capability SHOULD be up-to-date with respect to a CAPEC version that was released no more than four (4) months before the capability was made available to its users. If a capability does not satisfy this requirement, then it is "out-of-date."	✓
5.3	The capability owner SHOULD publicize how quickly it will update the capability's repository after a new CAPEC version or update becomes available on the CAPEC Web site.	✓
Revocation of CAPEC Compatibility		
6.1	If a review authority has verified that a capability is CAPEC-compatible, but at a later time the Review Authority has evidence that the requirements are not being met, then the Review Authority MAY revoke its approval.	N/A
6.1.1	The review authority MUST identify the specific requirements that are not being met.	N/A

6.2	The review authority MUST determine if the actions or claims of the capability owner are "intentionally misleading."	N/A
6.2.1	The review authority MAY interpret the phrase "intentionally misleading" at its discretion.	N/A
6.3	The review authority SHOULD NOT consider revoking CAPEC compatibility for a particular capability more often than once every six (6) months.	N/A
Warning and Evaluation		
6.4	The review authority MUST provide the capability owner and technical POC with a warning of revocation at least two (2) months before revocation is scheduled to occur.	N/A
6.4.1	If the review authority has found that the capability owner's actions or claims are intentionally misleading, then the Review Authority MAY disregard the warning period.	N/A
6.5	If the capability owner believes that the requirements are being met, then the capability owner MAY respond to the warning of revocation by providing specific details that indicate why the capability meets the requirements under question.	N/A
6.6	If the capability owner modifies the capability so that it complies with the requirements in question during the warning period, then the Review Authority SHOULD end the revocation action for the capability.	N/A
Revocation		
6.7	The review authority MAY delay the date of revocation.	N/A
6.8	The review authority MUST publicize that CAPEC compatibility has been revoked for the capability.	N/A
6.9	If the review authority finds that the capability owner's actions with respect to CAPEC compatibility requirements are intentionally misleading, then revocation SHOULD last a minimum of one year.	N/A
6.10	The review authority MAY publicize the reason for revocation.	N/A
6.11	The capability owner MAY post a public statement regarding the revocation on the same site.	N/A
6.12	If the approval is revoked, the capability owner MUST NOT apply for a new review during the period of revocation.	N/A
Review Authority		
7.1	A Review Authority MUST review the Capability for CAPEC compatibility with respect to a specific CAPEC version, i.e., the Review Version.	✓
7.2	A review authority MUST clearly identify the Review Version that was used to determine compatibility for the capability.	✓
7.3	A review authority MUST clearly identify the version of the CAPEC compatibility requirements document that was used to determine compatibility for the capability.	✓
7.4	A review authority MUST review every element in the capability's repository for CAPEC mapping accuracy.	✓

7.5	A review authority SHOULD review a capability for mapping accuracy at least once per year.	✓
Appendix A: Type-Specific Requirements		
A.1	The capability MUST satisfy all additional requirements that are related to the specific type of capability.	✓
A.1.1	If the capability is an assessment tool, dynamic application security testing (DAST) tool, penetration testing tool, exploit framework tool, threat modeling tool, or a product that integrates the results of one or more of these types of items, then it must satisfy the Tool Requirements, A.2.1 - A.2.8.	✓
A.1.2	If the Capability is a service (such as a security assessment service and training service, or a code and design review service) then it must satisfy the Security Service Requirements, A.3.1 - A.3.5.	✓
A.1.3	If the Capability is an online security issues or weaknesses in code database, Web-based resource, or information site, then it must satisfy the Online Capability Requirements, A.4.1 - A.4.3.	N/A
Tool Requirements		
A.2.1	The tool MUST allow the user to use CAPEC identifiers to locate associated tasks in that tool ("CAPEC-Searchable") by providing at least one of the following: a "find" or "search" function, a mapping between that tool's task names and CAPEC identifiers, or another mechanism determined to be sufficient by the review authority.	N/A
A.2.2	For any report that identifies individual security elements, the tool MUST allow the user to determine the associated CAPEC identifiers for those elements ("CAPEC-Output") by doing at least one of the following: including CAPEC identifiers directly in the report, providing a mapping between the tool's task names and CAPEC identifiers, or using some other mechanism determined to be sufficient by the review authority.	N/A
A.2.3	The publicly available documentation MUST explicitly list the CAPEC identifiers that the capability owner considers the tool effective at instantiating ("CAPEC-Compatibility Claim Coverage").	N/A
A.2.4	The capability's publicly available web site MAY provide the capability's CAPEC-Compatibility Claim Coverage as a CAPEC Coverage Claim Representation (CCR) XML document(s).	N/A
A.2.5	Any required reports or mappings MUST satisfy the media requirements as specified in Appendix B.	N/A
A.2.6	The tool, or the capability owner, SHOULD provide the user with a list of all CAPEC identifiers that are associated with the tool's tasks.	N/A
A.2.7	The tool SHOULD allow the user to select a set of tasks by providing a file that contains a list of CAPEC identifiers.	N/A
A.2.8	The interface of the tool SHOULD allow the user to browse, select, and deselect a set of tasks by using individual CAPEC identifiers.	N/A
A.2.9	If the tool does not have a task that is associated with a CAPEC identifier as specified by the user in the A.2.5 or A.2.6 tool requirements, then the tool SHOULD notify the user that it cannot perform the associated task.	N/A
Security Service Requirements Security services might use CAPEC-compatible tools in their work, but they may not provide their customers with direct access to those tools. Thus it could be difficult for customers to identify and compare the capabilities of different services. The Security Service Requirements address this potential limitation.		
A.3.1	The Security Service MUST be able to use CAPEC identifiers to tell a user which security elements are tested or covered by the service offering ("CAPEC-Searchable") by doing one or	✓

	more of the following: providing the user with a list of CAPEC identifiers that identify the elements that are tested or covered by that Service, providing the user with a mapping between the Service's elements and CAPEC identifiers, responding to a user-supplied list of CAPEC identifiers by identifying which of the CAPEC identifiers are tested or covered by the Service, or by using some other mechanism.	
A.3.2	For any report that identifies individual security elements, the Service MUST allow the user to determine the associated CAPEC identifiers for those elements ("CAPEC-Output") by doing one or more of the following: allowing the user to include CAPEC identifiers directly in the report, providing the user with a mapping between the security elements and CAPEC identifiers, or by using some other mechanism.	✓
A.3.3	The publicly available documentation MUST explicitly list the CAPEC identifiers that the capability owner considers the Security Service to effectively cover in its offering ("CAPEC-Compatibility Claim Coverage").	✓
A.3.4	The capability's publicly available web site MAY provide the capability's CAPEC-Compatibility Claim Coverage as a CAPEC Coverage Claim Representation (CCR) XML document(s).	✓
A.3.5	Any required reports or mappings that are provided by the Service MUST satisfy the media requirements as specified in Appendix B.	✓
A.3.6	If the Service provides the user with direct access to a product that identifies security elements, then that product SHOULD be CAPEC-compatible.	✓
Online Capability Requirements		
A.4.1	The online capability MUST allow a user to find related security elements from the online capability's repository ("CAPEC-Searchable") by providing one of the following: a search function that returns CAPEC identifiers for related elements, a mapping that links each element with its associated CAPEC identifier(s), or some other mechanism.	N/A
A.4.1.1	The online capability SHOULD provide a URL "template" that allows a computer program to easily construct a link that accesses the search function as outlined in online capability Requirements A.4.1. Examples: http://www.example.com/cgi-bin/db-search.cgi?cweid=XXX http://www.example.com/cwe/xxx.html	N/A
A.4.1.2	If the site is publicly accessible without requiring login, then the cgi program SHOULD accept "GET" method.	N/A
A.4.2	For any report that identifies individual security elements, the online capability MUST allow the user to determine the associated CAPEC identifiers for those elements ("CAPEC-Output") by doing at least one of the following: by allowing the user to include CAPEC identifiers directly in the report, providing the user with a mapping between the security elements and CAPEC identifiers, or by some other mechanism.	N/A
A.4.3	The publicly available documentation MUST explicitly list the CAPEC identifiers that the capability owner considers the online capability's repository to cover ("CAPEC-Compatibility Claim Coverage").	N/A
A.4.4	The capability's publicly available web site MAY provide the capability's CAPEC-Compatibility Claim Coverage as a CAPEC Coverage Claim Representation (CCR) XML document(s).	N/A
A.4.5	If the online capability does not provide details for individual security elements, then the online capability MUST provide a mapping that links each element with its associated CAPEC identifier(s).	N/A
Appendix B: Media Requirements		
B.1	The distribution media that is used by a CAPEC-compatible capability MUST use a media format that is covered in this appendix.	✓
B.2	The media format MUST satisfy the specific requirements for that format.	✓
Electronic Documents (HTML, word processor, PDF, ASCII text, etc.)		

B.3.1	The document MUST be in a commonly available format that has readers which support a "find" or "search" function ("CAPEC-Searchable"), such as raw ASCII text, HTML, or PDF.	✓
B.3.2	If the document only provides short names or titles for individual elements, then it MUST list the CAPEC identifiers that are related to those elements ("CAPEC-Output").	✓
B.3.3	The document SHOULD include a mapping from elements to CAPEC identifiers, which lists the appropriate pages for each element.	✓
Graphical User Interface (GUI)		
B.4.1	The GUI MUST provide the user with a search function that allows the user to enter a CAPEC identifier and retrieve the related elements ("CAPEC	✓
B.4.2	If the GUI lists details for an individual element, then it MUST list the CAPEC identifiers that map to that element ("CAPEC	✓
B.4.3	The GUI SHOULD allow the user to export or access CAPEC	✓

Table 11. Requirements and Recommendations for CAPEC Compatibility

4 Compliance of MITIGATE system with selected design standards

4.1 Development Lifecycle

The MITIGATE software development and integration procedure constitutes a continuous process which contains all required discrete steps that re-assure quality during the entire lifetime of the project. It could be argued that this process is realized in a virtual circle (as shown in the following figure1) that contains the following functional components a) Source-Code-Versioning/Management, b) Continuous Integration, c) Quality Assurance, d) Persistent Storage of built (a.k.a. artefacts) and e) Issue/Bug Tracking .

Each part of the circle is supported by mature tools that are setup and interoperate smoothly. More specifically these tools are: a) Git for Source Versioning, b) Jenkins for Continuous Integration, c) Sonar for quality assurance, d) nexus for artefact-management and e) Bugzilla for Issue Tracking. In the following sections we will analyse the reason why our development lifecycle is implemented by these tools.

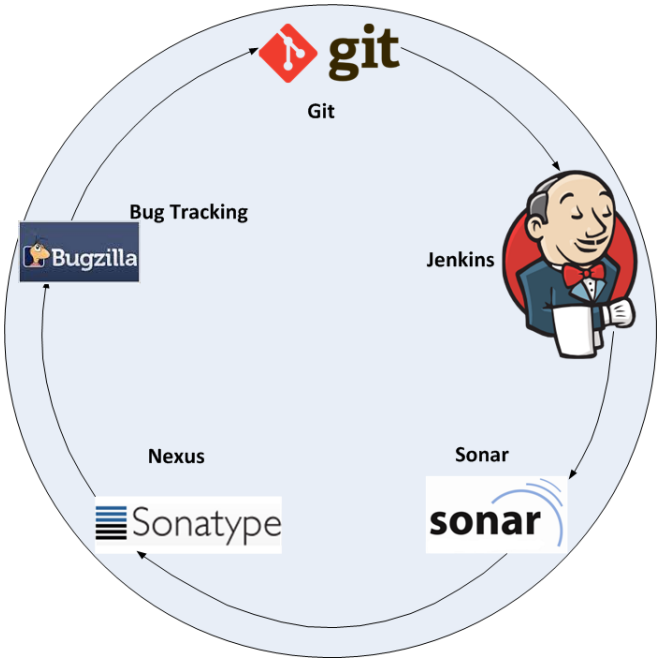

Figure 1 - Development Lifecycle adopted in MITIGATE

4.2 Supportive Tools

4.2.1 Version Control System

A Version Control System (also known as a Revision Control System) is a repository of files, often the files for the source code of computer programs, with monitored access. Every change made to the source is tracked, along with who made the change, why they made it, and references to problems fixed, or enhancements introduced, by the change. Version control systems are essential for any form of distributed, collaborative development. Whether it is the history of a wiki page or large software development project, the ability to track each change as it was made, and to reverse changes when necessary can make all the difference between a well managed and controlled process and an

uncontrolled 'first come, first served' system. It can also serve as a mechanism for due diligence for software projects.

Version control software, including the well known SVN and Git, was designed from the ground up to allow teams of programmers to work on a project together without wasting man-hours on paperwork. Instead of manually scanning branches of code and associated notes, version control allows for a central repository that is organized, logical, and facilitates file updates, notation, and even merging.

Git

First developed by Linus Torvalds of Linux fame, Git2 takes a radicalapproach that differs greatly from CVS and SVN. The original concepts for Git were to make afaster, distributed revision control system that would openly defy conventions and practices used in CVS. It is primarily developed for Linux and has the highest speeds on there. It will also run on other Unix-like systems, and native ports of Git are available for Windows as msysgit. As there is no centralized server, Git does not lend itself to single developer projects or small teams as the code may not necessarily be available when using a non-repository computer. Workarounds exist for this problem, and some see Git's improved speed as a decent tradeoff for the hassle. Git also comes equipped with a wide variety of tools to help users navigate the history system. Each instance of the source contains the entire history tree, which can be useful when developing without an internet connection. If any repository is lost due to system failure only the changes which were unique to that repository are lost. If users frequently push and fetch changes with each other this tends to be a small amount of loss, if any.

In a centralized VCS like Subversion only the central repository has the complete history. This means that users must communicate over the network with the central repository to obtain history about a file. Backups must be maintained independently of the VCS. If the central repository is lost due to system failure it must be restored from backup and changes since that last backup are likely to be lost. Depending on the backup policies in place this could be several human-weeks worth of work.

Due to Git being distributed, you inherently do not have to give commit access to other people in order for them to use the versioning features. Instead, you decide when to merge what from whom. That is, because subversion controls access, in order for daily check-ins to be allowed -for example - the user requires commit access. In Git, users are able to have version control of their own work while the source is controlled by the repository owner.

Branches in Git are a core concept used al the time. In Subversion they are more cumbersome and often used sparingly. The reason branches are so core in Git is every developer's working directory is itself a branch. Even if two developers are modifying two different unrelated files at the same time it's easy to view these two different working directories as different branches stemming from the same common base revision of the project. Consequently Git tracks the project revision the branch started from - this information is necessary to merge the branch back to trunk. It records branch merge events including: (a) author, time and date, (b) branch and revision information, (c) Changes made on the branches remain attributed to the original authors and the original timestamps of those changes, (d) changes which were made to complete the merge and are attributed to the merging user and (e) the reason that merge was done (optional; can be supplied by the user).In Subversion,

[2] http://git-scm.com/

branches and tags all are copies. Sometimes this is inconvenient; it is easy to checkout the whole repository by mistake. Branch path and file path lie in same namespace but they have different semantics - this can be confusing.

Furthermore, Git is extremely fast. Since all operations (except for push and fetch) are local there is no network latency involved to: i) Perform a diff, ii) View file history, iii) Commit changes, iv) Merge branches, v) Obtain any other revision of a file (not just the prior committed revision), vi) Switch branches.

Finally, Git's repository and working directory sizes are extremely small when compared to SVN.One of the reasons for the smaller repo size is that an SVN working directory always contains two copies of each file: one for the user to actually work with and another hidden in .svn/ to aid operations such as status, diff and commit. In contrast a Git working directory requires only one small index file that stores about 100 bytes of data per tracked file. On projects with a large number of files this can be a substantial difference in the disk space required per working copy.

As a full Git clone is often smaller than a full checkout, Git working directories (including the repositories) are typically smaller than the corresponding SVN working directories. There are even ways in Git to share one repository across many working directories, but in contrast to SVN, this requires the working directories to be co-located.

The Git repository (private) for MITIGATE is located here: https://github.com/singularlogic/mitigate-framework and is access is limited to the consortium developers for the time being. **After the finalization of the project the consortium will open the Git repository which will contain all modules.**

4.2.2 Continuous Integration

Continuous Integration is a software development practice where the members of a team frequently integrate their work – usually each contributor integrates his software code at least daily, leading to multiple integrations per day. Each integration cycle is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly. A Continuous Integration server exposes the following set of abilities:

- Contact a source code management (SVN, CVS, VSS, etc.) server and make updates of changes detected in a local directorys
- Launch one or more ant or maven script as needed for compilation and the packaging of archive files (jar, war and ear)
- Add plugins for code auditing, testing, and measurement of test coverage unit such as FindBugs, checkstyle, PMD, emma, cobertura, ...
- Execution of scripts launch tests
- Launch application server and deployment of applications through the ant and maven scripts
- Define inter-project dependencies
- Perform tasks of publications such as sending mail notification about the progress of the build process, the result of the builds and test results
- Send mail notification to people who have broken the code
- Prepare summaries and metrics on the build process

The greatest and most wide ranging benefit of Continuous Integration is reduced risk. The trouble with deferred integration is that it's very hard to predict how long it will take to do, and worse it's very hard to see how far you are through the process. The result is that you are putting yourself into a complete blind spot right at one of tensest parts of a project - even if you're one of the rare cases where you aren't already late. Continuous Integration completely finesses this problem. There's no long integration, you completely eliminate the blind spot. At all times you know where you are, what works, what doesn't, the outstanding bugs you have in your system.

Another benefit of Continuous Integrations is that it facilitates bug fixing. It doesn't get rid of bugs, but it does make them dramatically easier to find and remove. In this respect it's rather like self-testing code. If a bug is introduced and detected it quickly, it's far easier to get rid of. It's also ease to diff debugging - comparing the current version of the system to an earlier one that didn't have the bug. Bugs are also cumulative. The more bugs you have, the harder it is to remove each one. This is partly because you get bug interactions, where failures show as the result of multiple faults - making each fault harder to find. It's also psychological - people have less energy to find and get rid of bugs when there are many of them - a phenomenon that the Pragmatic Programmers call the Broken Windows syndrome. As a result projects with Continuous Integration tend to have dramatically less bugs, both in production and in process.

However, the degree of this benefit is directly tied to how good the test suite is. It's not too difficult to build a test suite that makes a noticeable difference. Usually, however, it takes a while before a team really gets to the low level of bugs that they have the potential to reach. Frequent deployment is valuable because it allows your users to get new features more rapidly, to give more rapid feedback on those features, and generally become more collaborative in the development cycle.

This helps break down the barriers between customers and development - barriers are the biggest ones to successful software development. Continuous integration should occur frequently enough that no intervening window remains between commit and build, and such that no errors can arise without developers noticing them and correcting them immediately. Normal practice is to trigger these builds by every commit to a repository, rather than a periodically scheduled build.

Jenkins

Jenkins[3] is a continuous integration (CI) tool written in Java, which runs in a servlet container, such as Apache Tomcat or the GlassFish application server. It supports SCM tools including CVS, Subversion, Git, Perforce and Clearcase and can execute Apache Ant and Apache Maven based projects, as well as arbitrary shell scripts and Windows batch commands. The primary developer of Jenkins was Kohsuke Kawaguchi, who worked for Sun Microsystems at the time. Released under the MIT License, Jenkins is free software. The Jenkins project is supported by Oracle Corporation. Builds can be started by various means, including scheduling via a cron-like mechanism, building when other builds have completed, and by requesting a specific build URL.

Even though Cruise Control is more mature, with wide variety of choices and capabilities and finally with extensive documentation, Jenkins is a newer solution that exhibits several advantages over Cruise Control:

[3] http://jenkins-ci.org/

- Ease of install (unzip the file and that's it)
- Full-fledged configuration via its friendly Web UI (no XML required)
- More attractive look-n-feel in dashboard
- Extremely flexibility
- It can be extended via plug-ins
- It can execute Phing, Ant, Gant, NAnt and Maven build scripts
- It gives you clean readable URLs for most of its pages
- It has RSS, e-mail and IM integration
- It can distribute build/test loads to multiple computers

After2010, an issue arose in the community of Hudson with respect to the infrastructure used, which grew to encompass questions over Oracle's stewardship and perceived control of the project. Negotiations between the principal project contributors and Oracle took place, and although there were many areas of agreement a key sticking point was the control of the name "Hudson" itself, which Oracle claimed, and for which it submitted a trademark registration in early December 2010 (granted as of October 25, 2011). As a result, on January 11, 2011, a proposal was made to change the project name from "Hudson" to "Jenkins". The proposal was overwhelmingly approved by those that voted on January 29, 2011, creating the Jenkins project. On February 1, 2011, Oracle indicated that they, in partnership with others in the community intended to continue development of Hudson making the necessary infrastructure changes, confirming two development branches. Jenkins is now considered a more advanced tool than Hudson for a number of reasons: The developers who wrote 99% of the core of Hudson are now writing Jenkins, which, of course, is built on the same base. This will lead to more stability, better bug crushing and more new features faster. In addition, we have seen an increase of contributions to Jenkins from more people since the split. There is a very open community managing the Jenkins project. There is an independent board with long time Hudson developers from multiple companies including Yahoo, CloudBees, CloudEra and Apture. They hold regular and fully open governance meetings and post notes after each meeting for public comment. They are also in the process of donating all the code to the Software Freedom Conservancy to assure continued openness of this community. One of the unique features of Jenkins has been KK's well-known weekly schedule of releases. The project recently circulated the 5th release. Moreover, since the split happened there have been over 30 bug fixes to Jenkins. The project board and community have also decided to announce a stable release approximately every 3 months. The stable release will get patches. This will fit more user requirements for not upgrading on a weekly basis. The plug-in developer community has moved to Jenkins which now has 345 plugins. Jenkins has become the primary platform for 5 of the top 5 and 19 of the top 25 plug-ins. Clearly the community activity has also moved to Jenkins with both users and development list vastly outnumbering the corresponding ones in Hudson.

4.2.3 Quality Assurance

In the modern world of software development, quality measurement has become increasingly important. As in any technological project in scale, there is a need for a way to measure the quality and how the work progresses, when different people have different access to pieces of code. Even though quality is a perceptual, conditional and somewhat subjective attribute and may be understood differently by different people, software structural quality characteristics have been clearly defined by the Consortium for IT Software Quality (CISQ), an independent organization

founded by the Software Engineering Institute at Carnegie Mellon University. CISQ has defined 5 major desirable characteristics of a piece of software needed to provide business.

In the “House of Quality” model, these are "What’s" that need to be achieved:

- Reliability: An attribute of resiliency and structural solidity. Reliability measures the level of risk and the likelihood of potential application failures. It also measures the defects injected due to modifications made to the software (its “stability” as termed by ISO).
- Efficiency: The source code and software architecture attributes are the elements that ensure high performance once the application is in run-time mode. Efficiency is especially important for applications in high execution speed environments such as algorithmic or transactional processing where performance and scalability are paramount.
- Security: A measure of the likelihood of potential security breaches due to poor coding practices and architecture. This quantifies the risk of encountering critical vulnerabilities that damage the business.
- Maintainability: Maintainability includes the notion of adaptability, portability and transferability (from one development team to another). Measuring and monitoring maintainability is a must for mission-critical applications where change is driven by tight time-to-market schedules and where it is important for IT to remain responsive to business-driven changes. It is also essential to keep maintenance costs under control.
- Size: While not a quality attribute per se, the sizing of source code is a software characteristic that obviously impacts maintainability.

Sonar

Sonar[4] is an open source software quality platform. Sonar uses various static code analysis tools such as CheckStyle, to extract software metrics, which then can be used to improve software quality.

Sonar offers reports on duplicated code, coding standards, unit tests, code coverage, complex code, potential bugs, comments, design and architecture. The primary supported language is Java - other languages are supported with extensions. Today, several open source and commercial extensions can cover the following languages: C, C#, PHP, Flex, Groovy, JavaScript, Python, PL/SQL, COBOL and Visual Basic 6. It integrates with Maven, Ant and continuous integration tools and is expandable with the use of plugins.

[4] http://www.sonarqube.org/

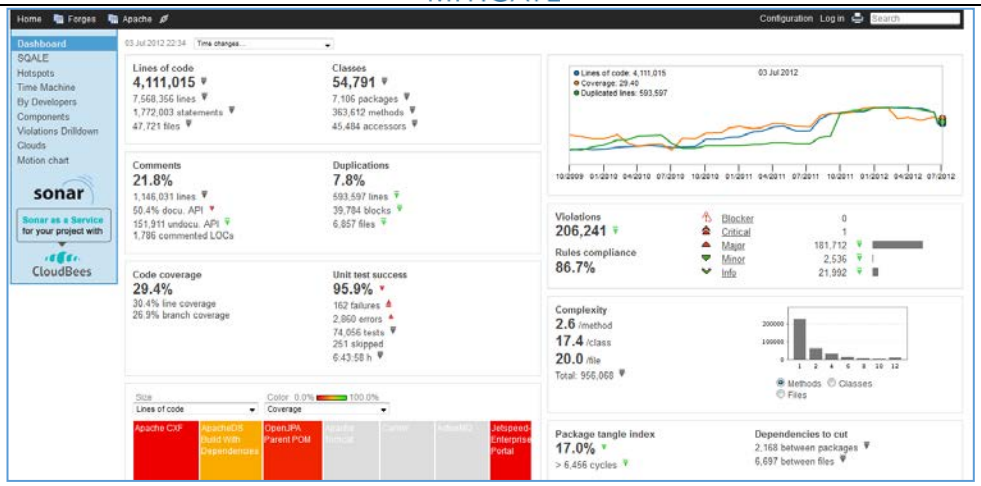

Figure 2 - Extract from an online installation of Sonar

4.2.4 Issue tracking

One main point in operating a service as the MITIGATE supply chain risk assessment is the issue tracking. Independent on how carefully and detailed the software was tested before going online, during the usage by the end-users there will occur bugs and problems. Therefore, it is necessary to install issue and bug tracking systems.

An issue tracker that is reachable for every developing partner needs to be included to collect development time issues like problem reports, feature requests, and work assignments. Users of the MITIGATE risk assessment toolkit have to be provided with bug reporting facility. For issues concerning coding, features and distribution the GitHub issue tracker5 is chosen while Bugzilla6 is applied to provide support for the end-users.

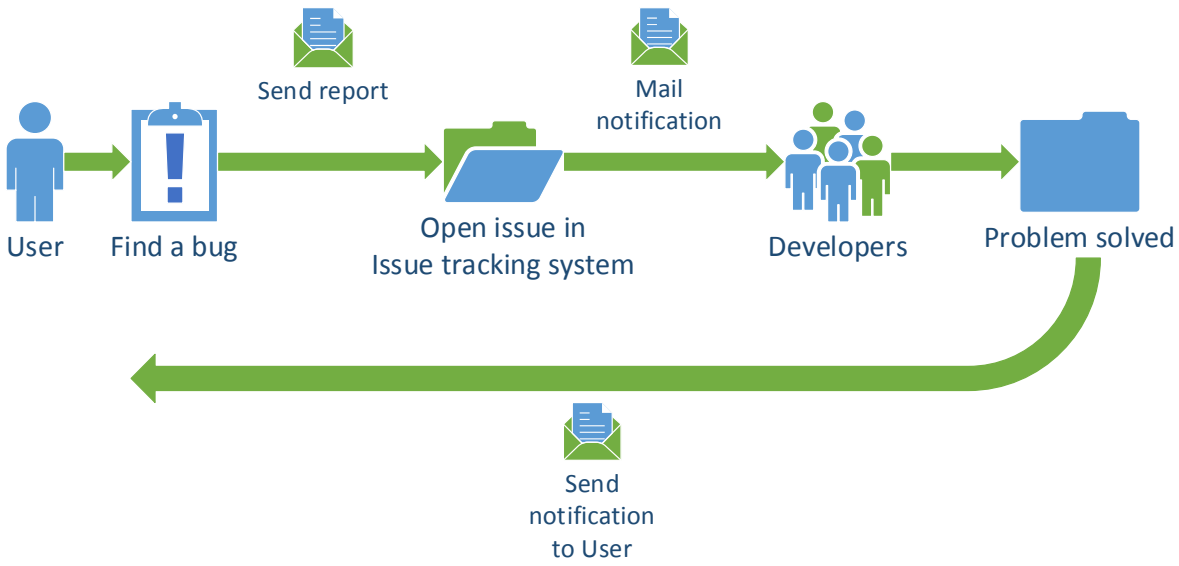

Figure 3 - Bug Reporting Mechanism

[5] https://github.com/blog/411-github-issue-tracker
[6] https://www.bugzilla.org/

Both the internal and external issue reporting processes should be implemented in the following way (cf. Figure 3). An end-user or a MITIGATE developer recognizes a problem or missing feature in the MITIGATE toolkit. He reports the bug or issue. The reporting is typically done by creating a new issue via the front end of the issue/bug tracker. The newly created issue is picked by a responsible and is typically assigned to a developer. A notification mechanism, usually implemented using e-mail, notifies the assignee about created issues and updates. The developer starts working on the ticket and documents the progress.

In order to derive a suitable process time there should be some time restrictions. Every developing partner should check the issue tracking system once per week. The ticket processing should not take more than one week except the reparation of the reported bug cause high charge in developing. The committed tickets are marked such that there is the possibility to identify the ticket uniquely. In MITIGATE, the Github issue tracking system will be used for the project-internal issue tracking. For bug reporting by MITIGATE end-users, Bugzilla will be used.

4.2.5 Release Planning

Release planning includes also repository management. The release management in the MITIGATE software development plan is done with the help of Nexus Release Management7. Source code management, especially manage simultaneously the on-going development and the creation of a current release prototype in modern software development projects is done with the help of a branching model. A well-known one is the Git branching model8. It will be applied in MITIGATE. The aim of the introduction of different branches is to ensure the quality of the resulting source code and to decrease the number of failures. Starting with a master branch this is parted into a developing branch, a release branch and a possibly existing Hotfix-branch. Furthermore, the MITIGATE toolkit consists of several components. The development of each component happens in a particular feature branch stemming from the master development branch. Having finished the development of the feature, the new developed component is merged into the development tool.

The release branch consists of the development branch including already the particular components. Therefore, the development branch is merged back into the release branch. The development of new features starts from the release branch again in the same way as described before. According to the work plan for the MITIGATE project there will be several releases. In order to avoid huge integration problems after the merging of the developed feature in the current development branch small releases in between will be helpful.

The advantage of this approach is that all partners can use an executable actual version of the MITIGATE platform for starting the development of the new features. The components are developed according to the project's time table and merged into the development branch as soon as they are available step by step.

[7] http://books.sonatype.com/nexus-book/reference/
[8] http://git-scm.com/doc

5 Conclusions

The MITIGATE approach encapsulates a number of requirements and principles associated with the effective and efficient management of the cybersecurity issues and problems faced by the maritime industry. All the identified elements were organized in seven concrete domains as follows:

- Risk Management.
- Asset, Change, and Configuration Management.
- Threat and Vulnerability Management.
- Situational Awareness.
- Information Sharing and Communications.
- Supply Chain and External Dependencies Management.
- Cybersecurity Program Management.

These domains were used to identify to what extent the MITIGATE system that integrates the proposed risk assessment approach covers the main aspects of widely-used cybersecurity-related standards, frameworks, models, programs, best practices and initiatives. As presented in Section 3, the developed system satisfies and implements different areas of the examined solutions.

In addition, the MITIGATE system adopts specific industry standards (Common Vulnerabilities and Exposures (CVE) and Common Attack Pattern Enumeration and Classification (CAPEC)) for vulnerability and exposure names in order to support well-recognized vulnerability identifiers. The benefits of having the CVE and CAPEC identifiers are significant in terms of information timeliness and data accuracy and reliability. In Section 3, we evaluated the system against the conditions and recommendations defined by the corresponding CVE and CAPEC Compatibility Programs in order to check the conformance of the system with these standards. We saw that the system satisfied several requirements, however, it has to complete the CVE and CAPEC compatibility processes in order to be registered as CVE and/or CAPEC Compatible.

Finally, in Section 4, we described the MITIGATE software development and integration procedure that has been adopted and will be followed throughout the entire life of the project. Also we presented the tools that were selected and are used to implement the defined development lifecycle.

6 References

[MITIGATE D2.2] MITIGATE Deliverable 2.2 “Evidence-Driven Maritime Supply Chain Risk Assessment Approach”

[MITIGATE D3.1] MITIGATE Deliverable 3.1, “Specification of the Risk Assessment, Mitigation and Simulation Functionalities”, 2016

[MITIGATE D3.2] MITIGATE Deliverable 3.2, “Prediction, Forecasting, Visualization and Open Intelligence Services Specification”, 2016.

[C2M2 Model] D. Gonzalez, D. W. White, J. Stevens, J. Grundman, N. Mehravari, P. Curtis, T. Dolan (Contributors). Cybersecurity Capability Maturity Model Version 1.1 (C2M2). Department of Energy, Office of Electricity Delivery and Energy Reliability (DOE-OE). Available at http://energy.gov/sites/prod/files/2014/03/f13/C2M2-v1-1_cor.pdf

[ISO/IEC 27001:2005] International Organization for Standardization. (2008). Information security management systems (ISO/IEC CD 27001:2005).

[ISO/IEC 27002:2005] International Organization for Standardization. (2008). Code of practice for information security management (ISO/IEC27002:2005).

[ISO 27005:2011] International Organization for Standardization. (2011). Information security risk management (ISO 27005:2011)

[ISO/IEC 21827:2008] International Organization for Standardization. (2008). Systems Security Engineering – Capability Maturity Model (SSE-CMM) (ISO/IEC 21827:2008)

[ISO 28001:2007] International Organization for Standardization. (n.d.). Security management systems for the supply chain - Best practices for implementing supply chain security, assessments and plans - Requirements and guidance (ISO/ IEC20001:2007).

[NIST Security Considerations in SDLC] Radack, S. (2008). Security considerations in the information system development life cycle. National Institute of Standards and Technology. Retrieved from http://www.itl.nist.gov/lab/bulletns/bltndec03.htm

[NIST SP800-16] Wilson, M., Stine, K., & Bowen, P. (2009). Information security training requirements: A role- and performance-based model (NIST Special Publication 800-16, revision 1.0). National Institute of Standards and Technology. Retrieved from http://csrc.nist.gov/publications/drafts/800-16-rev1/Draft-SP800-16-Rev1.pdf

[NIST SP800-37] National Institute of Standards and Technology, Joint Task Force Transformation Initiative. (2010). Guide for applying the risk management framework to federal information systems (NIST Special Publication 800-37). Retrieved from http://csrc.nist.gov/publications/nistpubs/800-37-rev1/sp800-37-rev1-final.pdf

[NIST SP800-40] Mell, P., Bergeron, T., & Henning, D. (2005). Creating a patch management and vulnerability management program (NIST Special Publication 800-40, version 2.0). National Institute of Standards and Technology. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-40-Ver2/SP800-40v2.pdf

[NIST SP800-53] National Institute of Standards and Technology, Joint Task Force Transformation Initiative. (2009). Recommended security controls for federal information systems and organizations (NIST Special Publication 800-53, revision 3). Retrieved from http://csrc.nist.gov/publications/nistpubs/800-53-Rev3/sp800-53-rev3-final_updated-errata_05-01-2010.pdf

[NIST SP800-61] Scarfone, K., Grance, T., & Masone, K. (2008). Computer security incident handling guide (NIST Special Publication 800-61, revision 1). National Institute of Standards and Technology. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-61-rev1/SP800-61rev1.pdf

[NIST SP800-64] Kissel, R., Stine, K., Scholl, M., Rossman, H., Fahlsing, J., & Gulick, Jessica. (2008). Security considerations in the system development life cycle (NIST Special Publication 800-64, revision 2). National Institute of Standards and Technology. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-64-Rev2/SP800-64-Revision2.pdf

[NIST SP800-128] National Institute of Standards and Technology. (2011). Guide for security-focused configuration management of information systems (Special Publication 800-128). Retrieved from http://csrc.nist.gov/publications/nistpubs/800-128/sp800-128.pdf

[NIST SP800-137] Dempsey, K., Chawla, N. S., Johnson, A., Johnston, R., Jones, A.C., Orebaugh, A. ... Stine, K. (2011). Information security continuous monitoring (ISCM) for federal information systems and organizations (NIST Special Publication 800-137). National Institute of Standards and Technology. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-137/SP800-137-Final.pdf

[NIST NVD] National Institute of Standards and Technology. (2012). National vulnerability database. Retrieved from http://nvd.nist.gov/cvss.cfm

[NISTIR 7622] Swanson, M., Bartol, N., & Moorthy, R. (2010). Piloting supply chain risk management for federal information systems (Draft NISTIR 7622). National Institute of Standards and Technology. Retrieved from http://csrc.nist.gov/publications/drafts/nistir-7622/draft-nistir-7622.pdf

[NISTIR 7628] The Smart Grid Interoperability Panel – Cyber Security Working Group. (2010). Guidelines for smart grid cyber security: Vol. 1, smart grid cyber security strategy, architecture, and high-level requirements (NISITIR 7628). National Institute of Standards and Technology. Retrieved from http://csrc.nist.gov/publications/nistir/ir7628/nistir-7628_vol1.pdf

[NISTIR 7628] The Smart Grid Interoperability Panel – Cyber Security Working Group. (2010). Guidelines for smart grid cyber security: Vol. 3, Supportive analyses and references (NISITIR 7628). National Institute of Standards and Technology. Retrieved from http://csrc.nist.gov/publications/nistir/ir7628/nistir-7628_vol3.pdf

[OECD Reducing Systemic Cybersecurity Risk] Sommer, P., & Brown, I. (2011). Reducing systemic cybersecurity risk. Organisation for Economic Co-operation and Development. Retrieved from http://www.oecd.org/dataoecd/57/44/46889922.pdf

[SEI CMM] Paulk, M., Weber, C., Garcia, S., Chrissis, M.B., & Bush, M. (1993). Key practices of the capability maturity model (Version 1.1, Technical Report CMU/SEI-93-TR-25). Software Engineering Institute, Carnegie Mellon University. Retrieved from http://www.sei.cmu.edu/reports/93tr025.pdf

[SCADA AU RMF] IT Security Expert Advisory Group. (2012). Generic SCADA risk management framework for Australian critical infrastructure. Retrieved from http://www.tisn.gov.au/Documents/SCADA-Generic-Risk-Management-Framework.pdf

[Situation Awareness in Dynamic Systems] Endsley, M. (1995). Toward a theory of situation awareness in dynamic systems. Human Factors, pp. 32-64.

[Supply Chain Risk Management Awareness] Filsinger, J., Fast, B., Wolf, D.G., Payne, J.F.X., & Anderson, M. (2012). Supply chain risk management awareness. Armed Forces Communication and Electronics Association Cyber Committee. Retrieved from http://www.afcea.org/committees/cyber/documents/Supplychain.pdf

[CVE Compatibility] Requirements and Recommendations for CVE Compatibility. Available at https://cve.mitre.org/compatible/requirements.html

[CAPEC Compatibility] Requirements and Recommendations for CAPEC Compatibility. Available at https://capec.mitre.org/compatible/requirements.html