Safety Critical Software and Security – How Low Can You Go?

Karin Bernsmed, Martin Gilje Jaatun and Per Håkon Meland

Citation:
Karin Bernsmed, Martin Gilje Jaatun and Per Håkon Meland (2018), Safety Critical Software and Security – How Low Can You Go? In: 2018 IEEE AIAA 37th Digital Avionics Systems Conference (DASC) Proceedings pp.210-215. DOI: 10.1109/DASC.2018.8569579

Safety Critical Software and Security – How Low Can You Go?

Karin Bernsmed
SINTEF Digital
Trondheim, Norway
karin.bernsmed@sintef.no

Martin Gilje Jaatun
SINTEF Digital
Trondheim, Norway
martin.g.jaatun@sintef.no

Per Håkon Meland
SINTEF Digital
Trondheim, Norway
per.h.meland@sintef.no

Abstract— **The safety of aviation software is ensured by performing development according to the DO-178C standard. However, this standard has a blind spot in that it fails to consider software security aspects in development. The Building Security In Maturity Model (BSIMM) comprises a software security framework with 113 software security activities. This model is often used for measuring the maturity of an organization's software security lifecycle. In this paper we evaluate the ability of DO-178C to ensure also software security, by demonstrating how few BSIMM activities you can get away with performing, while remaining compliant with the different DO-178C assurance levels. The results indicate that organizations with very low software security maturity can still be able to perform well in accordance to DO-178C. Based on the results, we propose concrete activities that could be integrated into the DO-178C development process, to strengthen the security of the developed software.**

Keywords—DO-178C, ED-12C, BSIMM, software, security, development

I. Introduction

The aviation community has extensive experience in analysing safety hazards, defining safety requirements and certifying the safety characteristics of the software installed in the aircraft. Once the software has been certified, it is considered to be safe as long as no changes are made to its architecture, design or operation. These assumptions are problematic from a security perspective.

Due to the characteristics of malicious activities, the cyber security risk picture is constantly changing. Threats that are relevant today may be irrelevant tomorrow and new threats that cannot be foreseen may appear in the future. Hence, it is generally accepted in the security community that software systems will almost be vulnerable and that risks must be continuously assessed, monitored and responded to. Conversely, safety analysis tends to focus solely on unintentional actions and failures; the risk of malicious interference is often overlooked, even though there may be safety implications. An open question is therefore how existing standards and regulations on aviation safety can be adjusted to reflect this new reality.

The safety of aviation software is supposedly ensured by performing development according to the DO-178C standard [1]. However, this standard does not consider software security aspects in development. The Building Security In Maturity Model (BSIMM) [2] comprises a software security framework with 113 software security activities that real software development organizations have been observed performing. BSIMM is intended to be used for measuring the maturity of an organization's software security lifecycle. Even though it is not a standard per se, it can be used as a yardstick for comparison with similar organizations. Objective measurement of software security of a given piece of software is very difficult, if not impossible [12]. BSIMM is therefore based on second order metrics, which measure the various activities that are performed when creating secure software. DO-178C does precisely this, but in connection with the development of safe software. However, in a digital system, the safety of a system is also dependent on security, since security incidents may also have safety consequences (see e.g., [3] and [4]).

In this paper we evaluate the ability of DO-178C to ensure also software security, by demonstrating how few BSIMM activities you can get away with performing, while remaining compliant with the different DO-178C assurance levels. Even though BSIMM is not a process-based standard, and hence not directly comparable with DO-178C, we find this study area interesting because it will provide insight into development activities that are considered important for security but that are not required to achieve the different safety assurance levels.

The remainder of this paper is organized as follows. Section II introduces BSIMM and DO-178C in further detail. This section also provides an overview of related work. Section III presents the results from the analysis. Section IV concludes the paper and provides some recommendations for improvements of the DO-178C development process.

II. Background

A. BSIMM

The Building Security In Maturity Model (BSIMM) [2] is a study of real-world software security initiatives that is organised so that an organisation can use it to determine where they stand with their software security initiative. BSIMM provides an overview over the security of software by mapping how it was built, what kind of activities that were carried out while it was built and by measuring a number of artefacts that were created when it was developed. BSIMM can also be used to measure how an organisation's software security efforts evolve over time.

A central concept in BSIMM is the Software Security Group (SSG), which is the person (or persons) responsible for software security in an organisation. The SSG can be as small as a single person, it need not be a formal role, and need not be a full-time position.

The BSIMM framework consists of twelve practices organised into four domains; Governance, Intelligence, SSDL Touchpoints and Deployment (see Table 1). Each practice comprises a number of activities on three levels, with level 1 being the lowest maturity and level 3 is the highest. For example, for practice Strategy and Metrics, SM1.4 is an activity on level

1, SM 2.5 is an activity on level 2, and SM 3.2 is an activity on level 3.

Table 1 The BSIMM Software Security Framework.

Governance	Intelligence	SSDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

BSIMM is the cumulative result of a multiyear study of real-world software security initiatives. As described by McGraw et al. [2], the model has been built directly out of data observed in 109 software security initiatives.

B. *DO-178C*

DO-178C Software Considerations in Airborne Systems and Equipment Certification [1] has been the main document used for approving commercial software-based aerospace systems. The initial version goes back to 1982, and the current version was released in 2012.

According to DO178C, software components must be certified according to a specific assurance level determined by a system safety assessment process. Table 2 shows a simplified overview of the failure conditions that the software should be verified against.

Table 2 DO-178C failure condition category descriptions.

Level	Description
A	**Catastrophic** failures that typically result in multiple fatalities and loss of the aeroplane.
B	**Hazardous** failures can cause a large reduction in the safety margins or functional capabilities of the aeroplane, excessive workload for the crew that will reduce performance and reliability of regular tasks, and possibly serious or fatal injuries to a small number of people.
C	**Major** failure conditions could cause a significant reduction in safety margins or functional capabilities of the aeroplane, including physical distress and injuries to the passengers or crew.
D	**Minor** failure condition could cause a slight reduction in safety margins or slight increase in crew work load.
E	**No safety effect**, hence not applicable for DO-178C.

For instance, a failure caused by a software component that is verified according to level D will typically only be a nuisance to the crew, and result in a reduced set of the functional capabilities which can be managed by a fall-back mechanism. For each level, there is a set of objectives that acts as guidance for the software production. They are there to make sure that the system is reliable and does not cause harm to the environment (meaning aeroplane, crew, passengers or anything outside of the aeroplane). Though security incidents can easily lead to events that fall into the categories described in Table 1, there are no security-related objectives in DO-178C. Instead, guidance about this can be found in other documents such as DO-326 Airworthiness Security Process Specification [13] and DO-356 Airworthiness Security Methods and Considerations [14].

Developing safety critical software using DO-178C is usually done using a waterfall development model, in which requirements are fed into a design that is then implemented in code, which is then verified and finally integrated into the final software system. The final output of the software development process is the executable object code together with a set of document artefacts that provides evidence that the objectives in the targeted assurance level have been met. It is important to note that the standard does not cover deployment and operation of the system itself.

C. *Related work*

The importance of also consider security when developing safety critical system is already well-know. Merged security and safety lifecycles are proposed in, for example, [15][16][17][18]. An overview over approaches that combine safety and security in the system lifecycle process is provided by Kriaa et al. [9]. However, as far as we are aware, there is no previous work that evaluates the DO-178C standard from the BSIMM perspective. Paulitsch et al. [8] discuss future security requirements in avionics and point out that "*A major question is to assess (ideally quantitatively) how effectively existing safety-oriented processes are performing from the viewpoint of security*", which is exactly what we are aiming for in this study.

III. Method

The BSIMM report [2] does not list the average number of activities performed in each practice, but we have calculated an average based on the total number of software development organizations that perform each activity. For instance, 55 out of 109 organizations perform SM1.1, which means this activity contributes 0,5 to the average activity count of the Strategy & Metrics practice. By doing this calculation for all activities in the practice, we arrive at an average of 4,2 activities (out of 11) in this practice across all the organizations in the BSIMM study. Note that this averaging does not take into account the *maturity level* of each activity; an activity counts as an activity whether it is level 1 or level 3. We have done this for all the practices, and the result can be observed in the red curve in Fig. 1. Note also in this figure that 7 is not a hard maximum, since many practices have more activities in total.

For the analysis we have assumed that the imaginary software developing organizations that we are evaluating are performing at their "worst" with respect to security, i.e., that they do not do any security related activities outside what is being required to meet the objectives for the targeted DO-178C assurance levels.

The work presented in this paper has been performed in the SoS-Agile project, which is funded by the Norwegian Research Council's IKTPLUSS program.

IV. Analysis

The mapping between the DO 178C and BSIMM demonstrates clearly that there are significantly different foci for the two documents. In Figure 1 we show the minimum software security maturity level of an imaginary software development organization that must comply with DO-178C Level A, compared to the average of firms that participate in the BSIMM study [2].

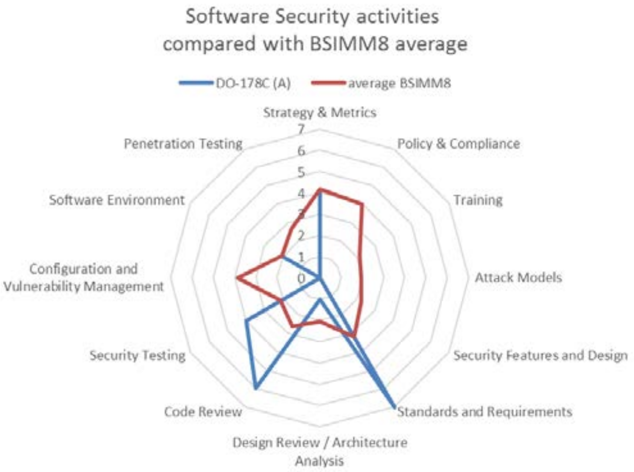

Fig. 1: Minimum maturity of an imaginary organization developing software w.r.t. DO-178C Level A compared to the maturity of the firms in the BSIMM study [2].

Level A is the highest assurance level and requires that all objectives defined in DO-178C are being met. As illustrated in Fig. 1, an organization complying with Level A will perform well in terms of the "Security Testing" activities defined by BSIMM. This is to be expected, since verification is a key part of DO-178C and the corresponding BSIMM security testing activities are to a high degree met when fulfilling the objectives in the DO-178C standard. More specifically, regarding security testing, the BSIMM activity "*ST3.4: Leverage coverage analysis*" will be covered by several of the objectives defined under the "Verification of verification process results" section of DO-178C. The BSIMM activities "*ST1.3: Drive tests with security requirements and security features*" and "*ST1.1: Ensure QA supports edge/boundary value condition testing*" will be covered by the DO-178C objectives to ensure that the executable object code complies with the high-level requirements and that the executable object code is robust with the high-level requirements. Further, regarding standards and requirements, an organization complying with Level A is required to define software development standards, thereby implementing the BSIMM activities "*SR1.1: Create security standards*". An organisation delivering Level A compliant software will also implement "*SR2.4: 25 Identify open source*" and "*SR3.1: Control open source risk*".

As can be seen from Figure 1, there are several weak spots of the Level A compliant organisation in terms of software security. Many of these are related to the BSIMM domains governance and deployment (see Table 1). According to BSIMM, governance is defined as "*Practices that help organize, manage, and measure a software security initiative*". BSIMM also consider staff development to be a central governance practice. Even though governance in its general meaning is a red thread in DO-178C, the focus is on compliance with the objectives of the standard rather than on compliance with external regimes and the score in this category is therefore low. Deployment is clearly out of scope of DO-178C, since this phase of the software lifecycle is covered by other airworthiness standards. Somewhat surprisingly, from the BSIMM point of view, DO-178C is also weak when it comes to design review and architecture analysis. Even though verification is a major component of the DO-178C standard, the standard does not require any specific competence from the persons who are performing such reviews. This is specifically recommended by BSIMM through the activities "*AA1.3 Have SSG lead design review efforts*" and "*CR1.2 Have SSG perform ad hoc review*". Neither does the DO-178C standard require the use of known bugs as input to the verification activities, as suggested by BSIMM through the activity "*CR2.7 Use a top N bugs list*". The use of automatic tools is also recommended in several of the activities in this category. These are the reasons why the BSIMM score is low.

An organization complying with **Level B** will implement the same BSIMM activities as the level A compliant organization. The main difference is that when going from Level B to Level A some of the DO-178C objectives needs to be fulfilled "with independence". There is no security activity defined in BSIMM that requires separation of responsibilities w.r.t verification and validation. Further, even though DO-178C Level A includes two objectives that are not required for Level B (these are "*Test coverage of software structure (modified condition/decision coverage) is achieved*" and "*Verification of additional code, that cannot be traced to Source Code, is achieved*"), the organization will not need to implement any additional BSIMM activities than the ones already required for Level B to meet these objectives. The BSIMM maturity measures for a Level A organization and a Level B organization will therefore be identical.

Level C is a lower assurance level than B, which manifests itself through requiring less independence when satisfying the objectives and fewer objectives that need to be fulfilled, compared with Level B. However, the objectives that are not needed for C, but that are required for B, are not covered by any of the BSIMM activities. This means that also the BSIMM maturity measure for the Level C organization will be identical to the maturity measures of the level A and B organizations.

An organization complying with **Level D** needs only to implement a single BSIMM activity: "*SM1.1: 55 Publish process (roles, responsibilities, plan), evolve as necessary*". The activity "*SR1.3: 71 Translate compliance constraints to requirements*" may also be (at least partly) covered when high-level requirements are developed, in case the system requirements are being derived from compliance constraints. There are no other BSIMM activities that must be done do to be compliant with Level D.

Table 3 The 12 BSIMM activities that "everybody" does mapped against the DO-178C assurance levels.

BSIMM: 12 core activities that "everybody does"		DO-178C assurance level			
Activity	**Description**	**Level A**	**Level B**	**Level C**	**Level D**
SM1.4	Identify gate locations and gather necessary artefacts.	Yes	Yes	Yes	No
CP1.2	Identify PII obligations.	No	No	No	No
T1.1	Provide awareness training.	No	No	No	No
AM1.2	Create a data classification scheme and inventory.	No	No	No	No
SFD1.1	Build and publish security features.	No	No	No	No
SR1.3	Translate compliance constraints to requirements.	Partly	Partly	Partly	Partly
AA1.1	Perform security feature review.	Partly	Partly	Partly	Partly
CR1.2	Have SSG perform ad hoc review.	Partly	Partly	Partly	Partly
ST1.1	Ensure QA supports edge/boundary value condition testing.	Yes	Yes	Yes	Yes
PT1.1	Use external penetration testers to find problems.	No	No	No	No
SE1.2	Ensure host and network security basics are in place.	No	No	No	No
CMVM1.2	Identify software bugs found in operations monitoring and feed them back to development.	No	No	No	No

BSIMM also identifies "*12 core activities that "everybody" does*", which the state are activities that are commonly found in highly successful software development programs. Even though they cannot conclude that these 12 activities are necessary for all software security initiatives, they recommend that everybody should consider them. Table 3 depicts these activities and map them to the difference DO-178C assurance levels.

As can be seen from the table, organizations that deliver software in compliance with DO-178C Level A, B and C will all perform the BSIMM activity "*SM1.4: Identify gate locations and gather necessary artefacts*". What this means, according to BSIMM, is that release gates/checkpoints/milestones are integrated in the software development lifecycle and that input necessary for making a go/no-go decision are collected. For DO-178C Level D it is only required that activities of the software life cycle process are defined; there is no need to explain or demonstrate how the transitions between the separate phases of the life cycle are being managed.

All the DO-178C assurance levels will implement the BSIMM activity "*ST1.1: Ensure QA supports edge/boundary value condition testing*". This will be done to meet the DO-178C objective that requires that the software (or more specifically, the executable object code) is robust with respect to the high-level requirements.

Three entries in the list of ubiquitous BSIMM activities are already partly covered in DO-178C. All the four assurance levels will partly implement the BSIMM activity "*SR1.3: Translate compliance constraints to requirements*". The set of high-level requirements are produced through analysis of system requirements and system architecture. The standard also allows the software development process to produce derived requirements. DO-178C does state that the high-level requirements include functional, performance, interface, and safety-related requirements, but do not discuss other types of requirements, such as requirements derived from compliance constraints. Further, the BSIMM activities "*AA1.1: Perform security feature review*" and "*CR1.2: Have SSG perform ad hoc review*" have also been partly implemented in DO-178C. Even though the standard requires a thorough Software Quality Assurance process, the use of dedicated risk and or threat analysis methods performed by people with security knowledge is not required by DO-178C.

The last two entries in the table above, which are related to the deployment and operation of the software, are clearly out of scope for DO-178C. Privacy, which is ensured through the second entry ("*CP1.2: Identify PII obligations*"), is also most likely out of scope when assessing a safety critical system. Further, whereas awareness training, data classification and penetration testing are generally considered efficient security countermeasures, such activities are unlikely to become candidates for inclusion in the set of DO-178C objectives.

Regarding the BSIMM activity "*SFD1.1: Build and publish security features*", it is somewhat concerning that the DO-178C standard often represents a showstopper to this approach. When using what is being referred to as "previously developed software" in the standard, DO-178C requires that a gap analysis is performed to identify the objectives that need to be satisfied. Filling these gaps is often an extremely time-consuming process, which in many cases means that it is easier to start over and develop the software from scratch [5]. This goes against the best practice identified by BSIMM, which recommends that the SSG identifies previously built software security features (e.g., for user authentication) that they like, and approve them for reuse in all other projects that need a similar functionality.

In the following, we will illustrate how the imaginary DO-178C-compliant software development company would compare to the adoption rate of software security activities in the full BSIMM population [2]. Fig. 2 shows this for the Strategy & Metrics (SM) practice; the blue columns indicate which activities contribute to DO-178C Level A compliance, and the red curve indicates the percentage of BSIMM organizations that perform each activity. Note that in some of the following figures, two practices have been combined to conserve space. BSIMM practices that are not performed at all by our imaginary company have been omitted.

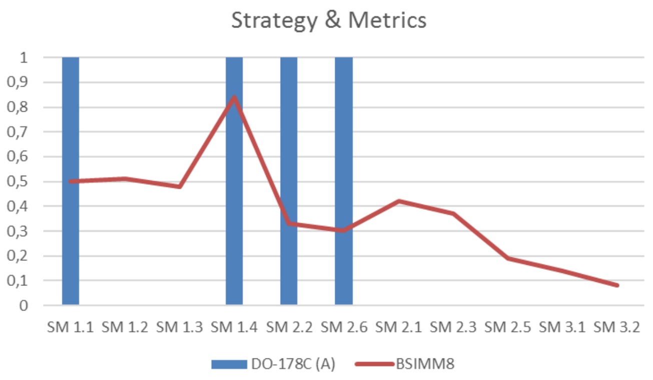

Fig. 2: DO-178C Level A Strategy & Metrics (SM) activities compared with the BSIMM compliance percentage [2].

In this case we can somewhat reassuringly conclude that the most common activity, "*SM1.4: Identify gate locations*" which is performed by more than 80% of the BSIMM organizations, is also performed by our imaginary DO-178C developer. However, it may be more disturbing that "*SM1.2: Create evangelism role*" and "*SM1.3: Educate executives*", which are both performed by about 50% of the BSIMM organizations, will *not* be performed in aviation.

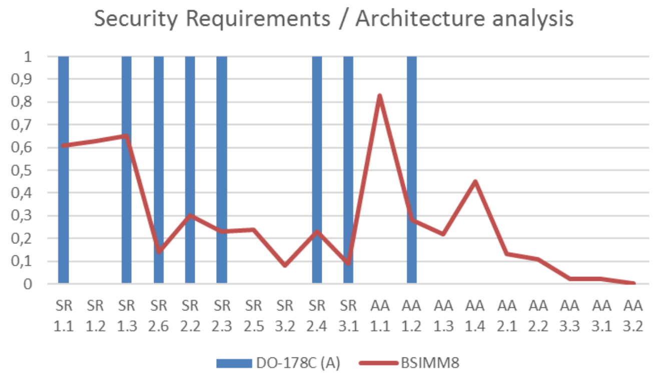

Fig. 3: DO-178C Level A Security Requirements (SR) and Architecture Analysis (AA) activities compared to the BSIMM compliance percentage [2].

In Fig. 3 we can notice that "*SR 1.2: Create a security portal*" and "*AA1.1: Perform security feature review*", performed by respectively more than 60% and 80%, of BSIMM organizations, are not performed by our imaginary developer.

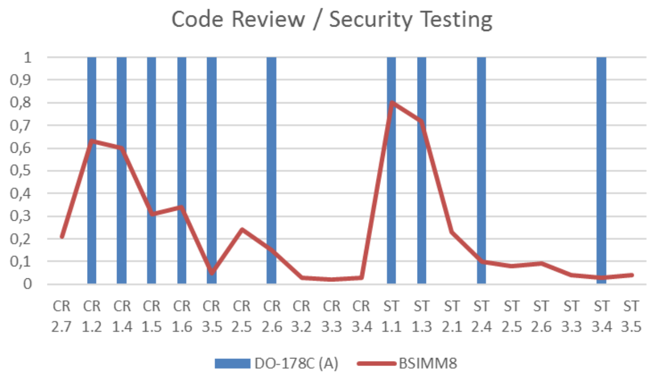

Fig. 4: DO-178C Level A Code Review (CR) and Security Testing (ST) activities compared with the BSIMM compliance percentage [2].

Fig. 4 shows that there are no major discrepancies when it comes to the practices Code Review and Security Testing; the only practices not covered have an adoption rate of among 20% or less among the BSIMM organizations.

Finally, Fig. 5 illustrates that the deployment domain is not DO-178C's strongest suit; "*SE1.2: Good practice network/host security*" which has more than 80% BSIMM adoption rate is not covered. DO-178C does, however, prescribe both the BSIMM activities "*SE2.2: Publish installation guides*" and "*SE2.4: Use code signing*".

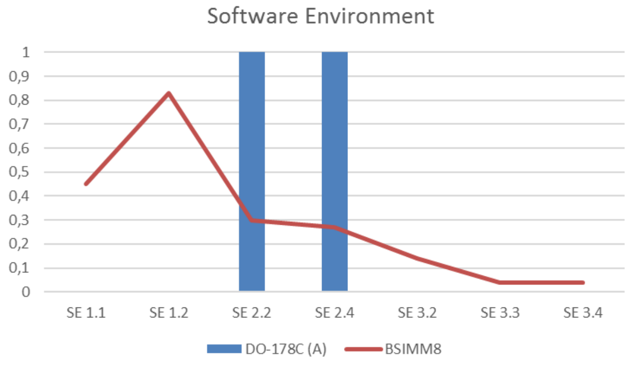

Fig. 5: DO-178C Level A Software Environment (SE) activities compared with the BSIMM compliance percentage [2].

V. Conclusions

DO-178C is a successful standard with a proven history. It has many benefits, amongst other including greater upfront requirements clarity, fewer coding iterations, fewer bugs found during module testing, fewer defects found during integration and fewer in-the-field defects [6]. Our own experience with developing software in accordance to Level D also indicates that it will catch many bugs, including potential security loopholes that otherwise could have been exploited.

Nevertheless, the standards also have weaknesses. First, as pointed out in [7], while a system may always have implementation defects or bugs, the security of many systems is often in practice breached due to "design flaws". In contrast to bugs, design flaws are much more subtle than bugs and harder to detect by traditional verification and validation methods, such than the ones prescribed by DO-178C. Second, the standard, and the full set of airworthiness standards in general, are not adapted to today's world in terms of support rapid patching and updates of software to fix bugs and defects found in the field. Finally, as discussed in Section III, it is difficult to reuse previously developed software that has not been previously certified.

An interesting observation from the results presented in Section III is that, according to the BSIMM measuring yardstick, Level A certified software will not be "more secure" than Level B or C, unless you are certain that the "independent reviewer" does a really good job. However, DO-178C does not prescribe any required competence of the reviewers, which contrasts with the 109 firms evaluated in the BSIMM study that all agree that the success of their initiative hinges on having an internal group devoted to software security [2].

Based on the results presented in this paper, we recommend that the following aspects are considered in a future update of the DO-178C standard:

- Make the Software Planning Process objective "*The software life cycle(s), including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria, is defined*" mandatory also for assurance Level D. This objective corresponds to one of the 12 BSIMM activities that "everybody does" and is hence considered best practices for any organization who is concerned about software security.
- The standard should include an objective that requires the organization to compile and maintain a list of "most important bugs" to be used as input to the verification activities. Such a list could be initially generated from public sources of known vulnerabilities but should eventually be maintained and updated based on real data gathered from code review, testing, and actual incidents [2]. This will effectively counter the most common security threats that exist today.
- Organizations need to be able to feed software defects found in the field back to the development process in an easy manner.

REFERENCES

[1] DO-178C, Software Considerations in Airborne Systems and Equipment Certification, RTCA, December 13, 2011

[2] G. McGraw, S. Migues, J. West: BSIMM: Building Security In Maturity Model. Version 8, September 2017 https://www.bsimm.com/

[3] K. Sampigethaya, R. Poovendran, S. Shetty, T. Davis, and C. Royalty, “Future e-enabled aircraft communications and security: The next 20 years and beyond,” Proceedings of the IEEE, vol. 99, no. 11, pp. 2040–2055, 2011.

[4] H. Chivers and J. Hird. 2013. Security Blind Spots in the ATM Safety Culture. In Proceedings of the 2013 International Conference on Availability, Reliability and Security (ARES '13). IEEE Computer Society, Washington, DC, USA, 774-779. DOI: https://doi.org/10.1109/ARES.2013.103

[5] L. Rierson. Developing safety-critical software. A practical guide for aviation software and DO-178C compliance. CRC Press, 2012

[6] V. Hilderman. Understanding DO-178C Software Certification: Benefits Versus Costs. Proceedings of the 2014 IEEE International Symposium on Software Reliability Engineering Workshops.

[7] Avoiding the Top 10 Software Security Design Flaws. IEEE Center for Secure Design, 2014. Available at https://www.computer.org/cms/CYBSI/docs/Top-10-Flaws.pdf

[8] M. Paulitsch , R. Reiger, L. Strigini and R. Bloomfield. Evidence-Based Security in Aerospace. From Safety to Security and Back Again. In IEEE 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW), 2012.

[9] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou and Y. Halgand. A survey of approches combining safety and security for industrial control systems. In Reliability Engineering and System Safety 139 (2015) 156–178. DOI: http://dx.doi.org/10.1016/j.ress.2015.02.008

[10] P. E. Black, L. Badger, B. Guttman, E. Fong "Dramatically Reducing Software Vulnerabilities." NISTIR 8151 (2016) https://doi.org/10.6028/NIST.IR.8151

[11] C. E. Landwehr "A building code for building code: putting what we know works to work." Proceedings of the 29th Annual Computer Security Applications Conference. ACM, 2013.

[12] M.G. Jaatun: "Hunting for Aardvarks: Can Software Security be Measured?", in Multidisciplinary Research and Practice for Information Systems, LNCS 7465, 2012, pp. 85-92

[13] DO-326, Airworthiness Security Process Specification, RTCA, August 6, 2014

[14] DO-356, Airworthiness Security Methods And Considerations, September 23, 2014

[15] C. Taylor, J. Alves-Foss and B. Rinker. "Merging Safety and Assurance: The Process of Dual Certification for Software." Proc. Software Technology Conference, April 2002.

[16] N. Nostro, A. Bondavalli, and N. Silva. “Adding Security Concerns to Safety Critical Certification.” Proc. - IEEE 25th Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2014, pp. 521–526, 2014.

[17] G. Stoneburner, “Toward a Unified Security / Safety Model,” Appl. Phys., pp. 96–97, 2006.

[18] C. Schmittner, Z. Ma, and E. Schoitsch. “Combined Safety and Security Development Lifecylce.” Proceeding - 2015 IEEE Int. Conf. Ind. Informatics, INDIN 2015, pp. 1408–1415, 2015.