Software Security Activities that Support Incident Management in Secure DevOps

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

ABSTRACT

Many software services are currently created using DevOps, where developers and operations personnel are more tightly integrated. The DevOps paradigm enables shorter development cycles, but increased speed has raised concerns over whether security issues may be overlooked. However, perfect security is never achievable, and in addition to the proactive software security efforts, we also need a reactive effort to handle flaws and bugs that are not discovered before they are used in an attack. In this paper we explore how focus on incident management and collaboration with developers can contribute to improved software security.

CCS CONCEPTS

• **Security and privacy → Software security engineering**;

KEYWORDS

DevOps, Software Security, Incident Management

ACM Reference Format:
Martin Gilje Jaatun. 2018. Software Security Activities that Support Incident Management in Secure DevOps. In *ARES 2018: International Conference on Availability, Reliability and Security, August 27–30, 2018, Hamburg, Germany.* ACM, New York, NY, USA, 6 pages. https://doi.org/x.x/x.x

1 INTRODUCTION

DevOps is not a new development paradigm, but has reached increased prominence recently. In DevOps, the barriers between developers and operations are lowered, sometimes to the point that the slogan "you build it – you run it" becomes reality. For most DevOps shops operating in the Cloud, this is also coupled with *continuous deployment*, where new versions of software systems can be deployed several times a day. This mode of working is really taking agility to a new level, but some doubts remain: How can we ensure security if new versions get deployed without the customary battery of security tests? Our answer is that whereas 100% security is never achievable, much can be done *by paying proper attention to incident management.*

By proactively employing software security practices [16], developers can avoid many mistakes that cause vulnerabilities in software; but since developers are only human, it is never possible to guarantee that any piece of software in perfectly secure. It therefore behooves any service provider to also have a second line of defense, where any flaws and bugs that are not discovered before they are used in an attack can be handled.

The remainder of the paper is structured as follows: In Section 2 we present relevant background. We present an analysis of secure software activities that are relevant for incident response in Section 3, and we discuss these further in Section 4. We offer conclusions and directions for further work in Section 5.

2 BACKGROUND

Incident management and software development have not traditionally been seen as going hand in hand; in the following we will give some brief background of the field of software security before moving on to incident management and existing work on DevOps security.

2.1 What is Software Security?

Software security is not about implementing security mechanisms in software, but concerns how to develop (ordinary) software in such a manner that it cannot easily be attacked [10].

The Building Security In Maturity Model (BSIMM) [16] is a software security framework of four domains each broken down into three practices, where each practice consists of a collection of concrete and measurable software security activities. BSIMM is an attempt to address the difficulty of measuring software security directly; given two pieces of software solving the same problem, it is effectively impossible to say which is "more secure" [9]. Instead, McGraw and colleagues decided to measure *second-order effects*, i.e., identify which software development activities contribute positively to software security, and measure to what extent these are performed in a given organization. A large number (109 in the latest installment) of software development organizations have been measured against the BSIMM framework, and thus there are grounds to claim that the BSIMM report [16] documents the real software security activities performed by real software development organizations.

Based on the BSIMM framework, we have previously conducted an assisted self-assessment of 20 public sector development organizations [11], and since then we have performed a further 6 assisted self-assessments of software security activities in private sector development organizations. It may be dangerous to compare our results with the BSIMM numbers, both due to the small number of organizations involved in our studies, and because the official BSIMM measurement are likely to be more rigorous than we had the opportunity to be. The reliance on self-assessment leaves us

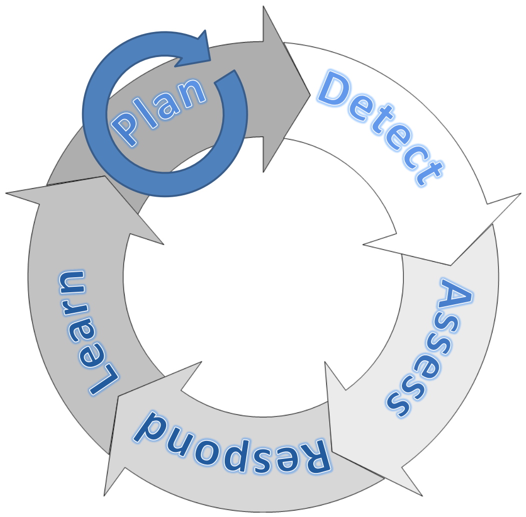

Figure 1: The ISO/IEC 27035 Incident Management Cycle

open to optimistic bias on the responders' side [21], since a natural reflex when doing any kind of assessment is to strive for the "best possible" score. However, we have found that even when accepting these limitations, self-assessments using the BSIMM framework have provided a useful starting point for working with software security in development organizations.

2.2 Computer Security Incident Management

Computer security incident management is unfortunately not an area where a lot of empirical research is being published [23], but even so it is a well-established field documented by international standards. The ISO/IEC 27035 standard [7] divides the incident handling process into five phases, as illustrated in Figure 1. The *Plan and prepare* phase (Plan) is an ongoing, continuous improvement process where plans are made, defenses are mounted, and troops are trained. The *Detection and reporting* phase (Detect) can be short, but crucial - this is where an incident is detected, by automatic or manual means. The *Assessment and decision* phase (Assess) tries to determine what happened, surveys the damage and determines how the incident should be handled. The *Responses* phase (Respond) is where the actual handling takes place, including post-incident mopping-up and restoring systems to regular service. Finally, in the *Lessons Learned* phase (Learn), the responders and the rest of the organization take a step back to learn from the incident, noting what worked well and areas for improvement. This naturally feeds back into the Plan phase; updating plans, training activities and countermeasures as appropriate.

Incident response is the natural habitat of system administrators; in larger organizations there may even be permanent incident response teams (IRTs) ready to deploy at a moment's notice. Traditionally, IRTs have not needed to worry about developers, since systems were either made by someone else, or produced locally with 6-month release cycles. However, with DevOps this has changed: When incidents happen, you need to determine if a bug or a flaw was the cause – if so, said bug or flaw needs to be fixed post haste. DevOps and continuous deployment enables such rapid fixing, but it also means that developers need to be clued into the incident response process.

Nearly a decade ago, Grobauer and Schreck [5] published a paper outlining some research needs related to incident management in the cloud. Unfortunately, there has been very limited research in this space since then, and much of that which has been published seems to focus rather narrowly on forensics issues [18], and fails to take into account the peculiarities of the cloud, such as the potentially long provider chains [12, 13]. Furthermore, the existing literature on incident handling does not consider interaction with software developers.

2.3 DevOps Security in the Literature

Much of the literature related to DevOps security seems to be blog entries and other non-validated channels. In one such blog [6], the idea that DevOps provides any security benefits in and of itself is challenged. Kim [14] argues that by integrating security testing into the daily operations of Dev, defects are found (and fixed) more quickly than before, but states "...it must be tested before the code is deployed", indicating a reluctance to rely on catching security bugs post deployment. This might seem contradictory, since a problem with agile development in general seems to be that developers "do not have time to think about security", and thus might not be qualified to identify which components that need extra testing. Furthermore, it is a tenet that "you cannot test yourself to security" [15].

Ur Rahman and Williams [24] studied 66 Internet "artifacts" (blogs etc.), and then performed interviews with nine DevOps organizations. Roughly half (32) of the artifacts provide input on security, and Ur Rahman and Williams list the following DevOps activities as contributing to improved security:

- Automated monitoring (mentioned in 13 artifacts)
- Automated deployment pipeline (8)
- Automated deployment (3)
- Automated testing of software changes (3)
- Software delivery in small increments (3)

On the other hand, the artifacts indicate the following DevOps activities as detrimental to software security:

- Use of immature automated deployment tools (2)
- Use of unsuitable metrics (2)
- Insufficient monitoring of collaboration (1)

Mohan and ben Othmane [17] attempted to perform a mapping study on security in DevOps, but found very few primary studies; mostly trade conference presentations and blogs (similar to the artifacts above). Yasar and Kontostathis [25] propose focusing on security requirements, threat modeling, environment configuration, static analysis, code review, penetration testing, environment testing, and finally a manual security review. They claim that quick incident response is an implicit benefit of such an approach, but do not offer any empirical evidence to support the claim. Incident response in mentioned several times by de Feijter [3] as a factor in DevOps maturity.

3 INCIDENT MANAGEMENT AND SOFTWARE DEVELOPMENT

There are several possible approaches when seeking to marry software security with incident response; we could ask:

- What software security activities can support incident response in secure DevOps?
- What are the most frequent IR-related software security activities ?

The first of these questions could be answered indirectly by analyzing the 113 BSIMM activity descriptions [16], and the answer to the second question would then follow directly from the BSIMM statistics.

Below we enumerate the BSIMM activities that, based on this author's assessment, are directly relevant to incident response in a DevOps setting.

SM2.3 *Create or grow a satellite*
The presence of security-minded developers across the organization will aid in resolving quick fixes in case of incidents. The term "satellite" is the one used in the BSIMM; some organizations have formalized this as a security champion for some or all development teams.

CP2.1 *Identify PII data inventory*
Due to GDPR [4], it will be paramount to know what kind of Personal Identifiable Information (PII) is handled by the system under attack, and where in the system it is stored or handled.

T3.5 *Establish SSG office hours*
In case of an incident, the handlers will benefit from having an available point of contact for software security issues.

AM2.7 *Build an internal forum to discuss attacks*
In order to learn from attacks, they need to be discussed – both to improve handling and to update software so that the same attack cannot succeed twice.

SR3.1 *Control open source risk*
If an attack is due to an open source vulnerability, you need to know which components use the library in question.

SE1.1 *Use application input monitoring*
Monitoring the input to your application can help detecting an attack as it happens.

SE3.3 *Use application behavior monitoring and diagnostics*
Monitoring the behavior of your application can help detecting an attack as it happens.

CMVM1.1 *Create or interface with incident response*
To have any hope of being able to make software changes quickly enough when an attack is manifest, there must be an interface between developers and incident response.

CMVM1.2 *Identify software defects found in operations and feed them back to development*
This has the dual effect of learning from incidents and improving the development lifecycle.

CMVM2.1 *Have emergency codebase response*
When a software related attack occurs, it is important to be able to make quick changes in order to stop the attack and prevent the same type of attack from occurring again.

Table 1: Mapping BSIMM Activities to ISO/IEC 27035 Phases

Plan	Detect	Assess	Respond	Learn
SM2.3	SE1.1	(SM2.3)	CMVM2.1	CMVM1.2
CP2.1	SE3.3	(CP2.1)	(SM2.3)	AM2.7
T3.5		(T3.5)	(T3.5)	
SR3.1			(SR3.1)	
CMVM1.1			(CMVM1.1)	
CMVM2.3			(CMVM2.3)	
CMVM3.3				

CMVM2.3 *Develop an operations inventory of applications*
This extends SR3.1 by creating a complete overview of which libraries and/or components are used in which applications.

CMVM3.3 *Simulate software crises*
This implies running preparedness exercises involving both incident responders and developers.

These software security activities can be mapped to the ISO/IEC 27035 phases as illustrated in Table 1. At first glance, it might seem that the majority of the activities relate to the Plan phase, with little or no activities related to the last three phases. However, many of the activities in the Plan phase are directly relevant for other phases; e.g., SM2.3 creates a virtual team of software security savvy developers who can be called on in the Assess and Respond phases. This is indicated in Table 1 by putting activities in parenthesis in the relevant phases.

Is it reasonable to expect that software security activities should support the Assess phase? Certainly, most developers will not have a primary focus on security, but if the organization has a satellite of security champions, these represent a natural sounding board when the Incident Response Team (IRT) stumbles upon something that doesn't smell right. Actually, an incident may not even reach the IRT before the satellite is involved if the incident is first discovered by the the customer support function due to some functional deviation; we could envision the support function first conferring with a security champion before deciding that the incident indeed is a security incident.

In Figure 2 we illustrate to what extent the BSIMM software security activities we identified as being relevant for the ISO/IEC 27035 "Plan" phase have been adopted in the three segments we introduced[1] in section 2.1; the BSIMM8 organizations ("BSIMM8"), the 20 public sector organizations [11] ("Pub20"), and the 6 industry organizations ("Industry").

There are some surprising differences, but the seemingly large adoption of T3.5 (SSG Office hours) among the local industry may be attributable to the fact that those that say they have an SSG typically only have a single employee in this (part-time) role, and state "I am always here". It is perhaps less surprising that the predominantly US BSIMM organizations are less worried about PII than their European counterparts, but this is a situation that will

[1]Remember that the BSIMM8 figures are taken from McGraw et al. [16], whereas the figures from the other two segments originate from our own research.

be interesting to monitor as US vendors realize they have to worry about GDPR if they plan to offer their services in the European market.

One important statistic is CMVM3.3 (simulate SW crisis), which shows that only a negligible percentage of the BSIMM organizations involve the developers in their emergency preparedness exercises (if they have any). The numbers are slightly higher for the European organizations, but our interactions with these organizations seem to confirm that this is an issue that most of them have not been been aware of.

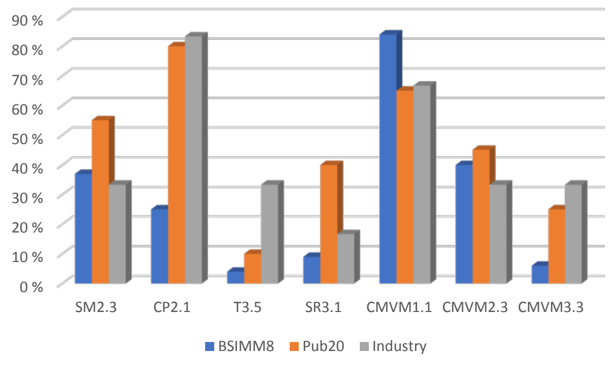

Figure 2: Activities Related to the Incident Response Plan Phase in Different Segments

There are only two BSIMM activities directly related to the "Detect" phase, as shown in Figure 3. Interestingly, we can see here that whereas the three segments are roughly on par with respect to SE1.1 (monitor input), our local samples have a dramatically higher adoption rate of SE3.3 (behavior monitoring). There is no obvious explanation to why this should be the case, but the small sample size of the industry organizations may skew the results. A possible explanation for the high adoption rate in the public sector organizations may be that they generally deal more with Personally Identifiable Information (PII), and thus are more compelled to monitor behavior.

The only primary activity directly related to the Respond phase is CMVM2.1 (emergency codebase response), and as can be seen from Figure 4, the BSIMM organizations are about the same as the local industrial sample (~80%), with the public sector organizations showing only slightly lower adoption (70%).

Almost all (~80%) the BSIMM8 organizations perform CMVM1.2 (feed defects back to development), as do the 6 industry organizations; it is actually surprising that only 50% of the public sector organizations claim to do this. Although we did not follow this up directly at the time, a possible explanation might be that those organizations to a greater extent outsource the operation of their systems. Less than 10% of the BSIMM8 organizations do AM2.7 (internal forum for attacks), whereas about half of the organizations from the other two segments do. Unless there is a large discrepancy between what the official BSIMM considers an "internal forum" and how the Norwegian organizations interpreted the concept, it is difficult to understand this large difference.

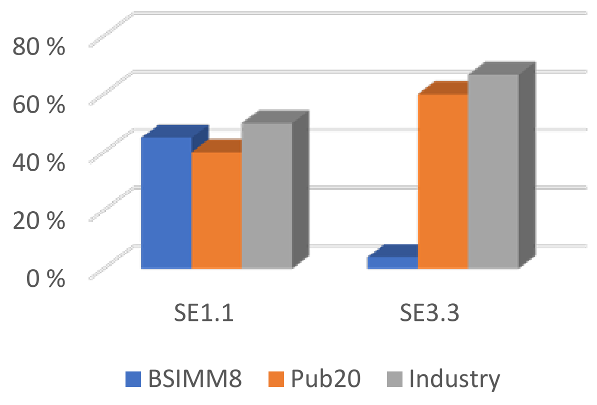

Figure 3: Activities Related to the Detect Phase in Different Segments

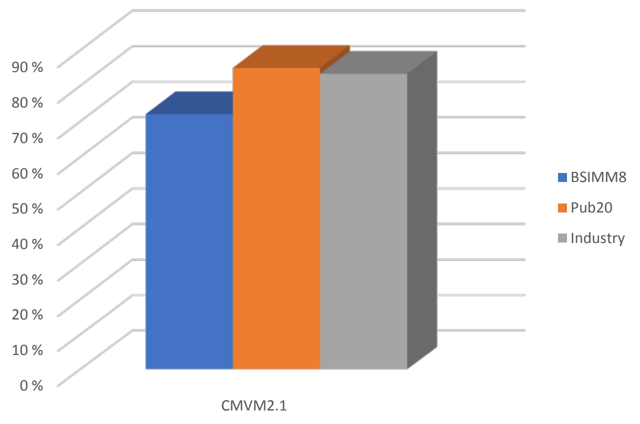

Figure 4: Activities Related to the Respond Phase in Different Segments

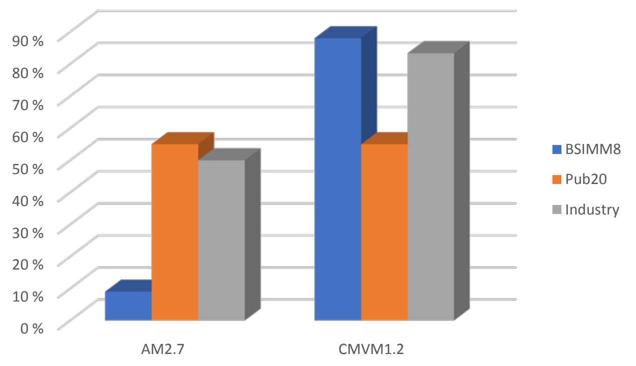

Figure 5: Activities Related to the Learn Phase in Different Segments

4 DISCUSSION

The BSIMM framework was initially created in 2008, and at that time neither Cloud Computing nor DevOps were generally known terms. Even though new software security activities have been added to the framework over the years, it is reasonable to ask: Do we need additional software activities to handle DevOps incidents? Companies may also do other software security relevant activities that support incident management, but that are not currently part of BSIMM, and that therefore do not show up on the graph. In a similar vein, it is clear that the activities in Table 1 will also help incident response in organizations that do not use the DevOps paradigm. The 2017 State of DevOps Report [20] claims that DevOps high performers recover from downtime 96 times faster than lower performing organizations, which suggests that studying the incident response practices (and related software security practices) of the high performers could yield interesting results.

A clear limitation of this study is represented by the informal nature of the selection criteria for choosing the activities in Table 1. One could argue that *all* activities that contribute to better security also contribute to better incident management, if for nothing else then by reducing the number of incidents that occur in the first place. There are also general things, such as management support, that will be vital to the success of any security initiative, not only incident response. Furthermore, we could envisage extending some activities, such as SM1.1 (publish process), to more explicitly cover incident response aspects. We hope to address this in our further work, by empirically identifying new and existing software security activities that stand out as important contributors to incident response.

It is likely that incident handlers at some point need additional information from developers in the Assess phase - but that would not necessarily be considered a software security activity in itself. The key BSIMM activity is of course CMVM1.1 (interface with incident response); once this is established it should be easy to answer the question "Who you gonna call?" [8]. A vast majority of the BSIMM organizations [16] perform this activity, as do about 60 % of the local organizations we have surveyed. For larger organizations with a significant Software Security Group (SSG), that will be the natural point of contact; but none of our local organizations are large enough to have a full-time SSG, and another approach is thus needed. It is tempting to assign this responsibility to the security champions in the satellite, but that could be counter-productive in the long run. Being a security champion should be a bonus, not something that generates extra work and inhospitable hours. A better option would be to create a rota of *all* developers, ensuring that there is always someone to call if the need arises. However, for this to work, it is imperative to train the tier-1 responders to a minimum level in order to avoid spurious calls to the developers.

Just as smaller software companies are unlikely to have a full-time SSG, they are unlikely to have a full-time IRT. This implies that a virtual IRT must be created, usually starting with a core operations team. This team would naturally be augmented with the software security champions during working hours; this would be how CMVM1.1 would be achieved in practice in a small organization.

Traditional incident response management has been focused on operation of servers running commodity operating systems with some standard server applications, such as web servers, mail servers and database servers. In the recent years before the cloud era, such systems were bought, rather than built. This implied that if a software vulnerability was found, the incident response team needed to depend on the software vendor to supply a patch or update; generally there were no or few "user-serviceable parts inside".

In DevOps, there is no difference between deploying a security patch and and any other change - both can happen several times a day. This implies that the organization needs to have higher focus on incident response, also by developers. The need for a patch can arrive at any time, and the developers should expect this.

Ben Othmane et al. [1] list added cost as a disadvantage of the agile security engineering scheme they propose. However, one could argue similarly that fixing syntax errors in the code introduces extra work and associated costs, but no one would propose that such errors should not be fixed in order to save money. Furthermore, it is necessary to consider the total lifecycle cost of a software product, and it is actually cheaper to fix errors at the design or coding stage than when they are discovered after deployment [2]. However, the dramatic differences that Boehm refer to may be less noticeable when we are talking about DevOps; the big difference presumably comes when SW is shrink-wrapped and shipped, whereas updating something which is only sold as a service is clearly less arduous.

On the other hand, whereas incident response is something that traditionally would have been an externality [22] for a software development organization that sells shrink-wrapped software, it becomes a vital business function for a DevOps shop offering software as a service. Thus, the cost of of incident response is less of an issue in terms of determining whether or not you need to do it, but doing it in the most cost-effective way will of course always be important.

We have argued that software security education of developers is more important in agile development than in traditional waterfall, stopping short of teaching every developer to be a software security expert, but aspiring to teach every developer enough to enable them to identify areas where they would benefit from the advice of an expert [9]. This education would need to be extended to also cover incident response with respect to the developers, but as mentioned it cuts both ways; the front line incident handlers need to know something about development as well. One obvious approach to this is to include the developers in general incident response exercises. With a properly defined incident response scenario, such an exercise would provide training to both Dev and Ops, but would also possibly help in identifying additional activities that are currently not part of the organization's SSDL.

DevOps success is tightly coupled to the available toolchain, and the rapid deployments are only possible due to tightly configured deployment scripts. Communications between Dev and Ops is also vital in any situation where they are not actually the same persons. It therefore becomes important to establish who should know what. This implies that the tools used for incident response need to be integrated with the tools already used by the developers, and vice versa.

One challenge with using BSIMM as a blueprint for adding support for incident response in DevOps is that unlike OpenSAMM [19], BSIMM is descriptive, not prescriptive. This means that the BSIMM report [16] documents the adoption rate of IR-relevant software security activities among a specific (non-random) set of development organizations, but it does not make value judgments of the sort "you should do activity X". This is more explicitly done in OpenSAMM, but the fixed structure of only two (cumulative) activities per OpenSAMM maturity level makes us question whether it would ever be possible to get all the relevant activities fit the framework. In other words, if there are any IR-relevant software security activities missing from BSIMM they could be added, but in the case of OpenSAMM one would always need to maintain a "shadow" list of software security activities that come in addition to the ones used for calculating the maturity level.

5 CONCLUSION

Failing fast is a virtue in agile development, but when transported to a DevOps situation, all failures become public. This could be seen as a potential public relations nightmare, and many companies are thus wary of making the leap to a full continuous deployment DevOps shop. We have in this paper argued that proper attention to incident management, and ensuring that the developers are included as part of the incident management lifecycle, can reduce this risk to manageable proportions, allowing more companies to reap the benefits of DevOps without sacrificing security.

We will continue to work with our partner software development organizations to establish the optimal integration between software developers and incident responders, in particular through running and evaluating exercises. We will want to study more high-performing DevOps organizations to learn the core software security activities that set them apart from the rest, eventually producing a version of Table 1 which is empirically validated.

ACKNOWLEDGMENT

This work was supported by the SoS-Agile: Science of Security in Agile Software Development project, funded by the Research Council of Norway (grant number 247678). Thanks to Daniela Cruzes and Karin Bernsmed for useful discussions on the interplay between software security and incident management.

REFERENCES

[1] Lotfi ben Othmane, Pelin Angin, Harold Weffers, and Bharat Bhargava. 2014. Extending the Agile Development Approach to Develop Acceptably Secure Software. *IEEE Transactions on Dependable and Secure Computing* (2014).

[2] Barry Boehm and Victor R Basili. 2005. Software defect reduction top 10 list. In *Foundations of empirical software engineering: the legacy of Victor R. Basili*. Vol. 426.

[3] Rico de Feijter. 2017. *Towards the adoption of DevOps in software product organizations: A Maturity model approach*. Master's thesis.

[4] EU. 2016. Position of the Council at first reading with a view to the adoption of a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *LEGISLATIVE ACTS AND OTHER INSTRUMENTS* 16, 5419 (2016). http://data.consilium.europa.eu/doc/document/ST-5419-2016-INIT/en/pdf

[5] Bernd Grobauer and Thomas Schreck. 2010. Towards Incident Handling in the Cloud :. In *Proceedings of the 2010 ACM Workshop on Cloud Computing Security*. ACM, Chicago, Illinois, USA, 77–85. https://doi.org/10.1145/1866835.1866850

[6] George V. Hulme. 2015. The Myth of DevOps as a Catalyst to improve Security? (2015). http://devops.com/2015/07/16/the-myth-of-devops-as-a-catalyst-to-improve-security/

[7] ISO 2011. ISO/IEC 27035:2011 Information technology - Security techniques - Information security incident management. (2011).

[8] Ivan Reitman (dir.), Dan Aykroyd (Writ.), Harold Ramis (Writ.), Bill Murray (Perf.), Dan Aykroyd (Perf.), and Sigourney Weaver (Perf.). 1984. Ghostbusters. Motion picture. (1984). Columbia Pictures.

[9] Martin Gilje Jaatun. 2012. Hunting for Aardvarks: Can Software Security Be Measured? In *Multidisciplinary Research and Practice for Information Systems*, Gerald Quirchmayr, Josef Basl, Ilsun You, Lida Xu, and Edgar Weippl (Eds.). Lecture Notes in Computer Science, Vol. 7465. Springer Berlin Heidelberg, 85–92. https://doi.org/10.1007/978-3-642-32498-7_7

[10] Martin Gilje Jaatun. 2017. Secure Software Engineering is not About Security Features. *International Journal of Secure Software Engineering* 8, 2 (2017), iv.

[11] Martin Gilje Jaatun, Daniela S. Cruzes, Karin Bernsmed, Inger Anne Tøndel, and Lillian Røstad. 2015. Software Security Maturity in Public Organisations. In *Information Security*, Javier Lopez and Chris J. Mitchell (Eds.). Lecture Notes in Computer Science, Vol. 9290. Springer International Publishing, 120–138. https://doi.org/10.1007/978-3-319-23318-5_7

[12] Martin Gilje Jaatun, Siani Pearson, Frédéric Gittler, Ronald Leenes, and Maartje Niezen. 2016. Enhancing Accountability in the Cloud. *International Journal of Information Management* (2016). https://doi.org/10.1016/j.ijinfomgt.2016.03.004

[13] Martin Gilje Jaatun and Inger Anne Tøndel. 2015. How Much Cloud Can You Handle?. In *Availability, Reliability and Security (ARES), 2015 10th International Conference on*. 467–473. https://doi.org/10.1109/ARES.2015.38

[14] Gene Kim. 2012. Top 11 Things You Need to Know About DevOps. (2012). https://www.thinkhdi.com/~/media/HDICorp/Files/White-Papers/whtppr-1112-devops-kim.pdf

[15] Gary McGraw. 2006. *Software Security: Building Security In*. Addison-Wesley.

[16] Gary McGraw, Sammy Migues, and Jacob West. 2017. Building Security In Maturity Model (BSIMM 8). (2017). http://bsimm.com.

[17] Vaishnavi Mohan and Lotfi ben Othmane. 2016. SecDevOps: Is It a Marketing Buzzword? - Mapping Research on Security in DevOps. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*. 542–547. https://doi.org/10.1109/ARES.2016.92

[18] Victor Ion Munteanu, Andrew Edmonds, Thomas M. Bohnert, and Teodor-Florin Fortis. 2014. Cloud Incident Management, Challenges, Research Directions, and Architectural Approach. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC '14)*. IEEE Computer Society, Washington, DC, USA, 786–791. https://doi.org/10.1109/UCC.2014.128

[19] OpenSAMM. [n. d.]. Software Assurance Maturity Model (SAMM): A guide to building security into software development. ([n. d.]). http://www.opensamm.org/.

[20] Puppet and DORA. 2017. 2017 State of DevOps Report. (2017). https://puppet.com/resources/whitepaper/state-of-devops-report.

[21] Hyeun-Suk Rhee, Young U. Ryu, and Cheong-Tag Kim. 2012. Unrealistic optimism on information security management. *Computers & Security* 31, 2 (2012), 221 – 232. https://doi.org/10.1016/j.cose.2011.12.001

[22] Bruce Schneier. 2007. Information Security and Externalities. Schneier on Security Blog. (2007). https://www.schneier.com/blog/archives/2007/01/information_sec_1.html

[23] Inger Anne Tøndel, Maria B. Line, and Martin Gilje Jaatun. 2014. Information security incident management: Current practice as reported in the literature. *Computers & Security* 45, 0 (2014), 42 – 57. https://doi.org/10.1016/j.cose.2014.05.003

[24] Akond Ashfaque Ur Rahman and Laurie Williams. 2016. Software Security in DevOps: Synthesizing Practitioners' Perceptions and Practices. In *Proceedings of the International Workshop on Continuous Software Evolution and Delivery (CSED '16)*. ACM, New York, NY, USA, 70–76. https://doi.org/10.1145/2896941.2896946

[25] Hasan Yasar and Kiriakos Kontostathis. 2016. Where to Integrate Security Practices on DevOps Platform. *International Journal of Secure Software Engineering* 7, 4 (2016), 39–50. https://doi.org/10.4018/IJSSE.2016100103