# DevOps for Better Software Security in the Cloud

Author version

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

Daniela S. Cruzes
SINTEF Digital
Trondheim, Norway
danielac@sintef.no

Jesus Luna
Technische Universität Darmstadt
Darmstadt, Germany
jluna@cs.tu-darmstadt.de

## ABSTRACT

The DevOps paradigm means that development and operations for an organisation blend together. For security, this implies that information on detected attacks can be fed back to the development, enabling faster eradication of vulnerabilities in software. This is particularly important in cloud installations, where release cycles can be less than a day. This paper argues that DevOps can be employed for overall improved software security.

## CCS CONCEPTS

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

## KEYWORDS

DevOps, Software Security, Cloud Security, Security Metrics

**ACM Reference Format:**
Martin Gilje Jaatun, Daniela S. Cruzes, and Jesus Luna. 2017. DevOps for Better Software Security in the Cloud : Author version. In *Proceedings of ARES '17*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3098954.3103172

## 1 INTRODUCTION

Cloud computing brings with it many advantages, but *on-demand self service* [18] is perhaps the most significant for system developers; computer resources can be deployed and scaled up and down from web-accessible dashboards in a matter of seconds. This in turn has paved the way for DevOps: a new paradigm in developing and operating software systems, sometimes condensed to the phrase *"You build it, you run it"* [21].

Whereas practices that ease the development, deployment and operation of software are great, it cannot be denied that security breaches are happening all around us. Software systems have developed to the point that we use and depend upon them daily in the same way that we depend upon traditional infrastructures and utilities such as power, transportation and telecommunication. The value of sensitive information in computer systems is constantly increasing, and the same can be said for the corresponding threats, but measures to reduce the resulting vulnerability are not developed at the same pace. The consequences of this lack of investment in software security can be catastrophic.

The Heartbleed vulnerability has been called one of the Internet's biggest and most dangerous security threats. This vulnerability was present in running software for over two years, but only publicly discovered in April 2014. Almost two-thirds of all sites on the Internet were exposed. Reports say that the US National Security Agency (NSA) knew about the Heartbleed vulnerability from the start, and regularly used it to gather intelligence and access information on millions of users. The full extent of the consequences of this bug is as yet unknown, because it is impossible to trace the activities of the ones that took advantage of this vulnerability. Although the lack of software security is already a serious problem, it may become much worse in the future due to the increase in complexity, connectivity and extensibility of software systems.

A major problem in software security is that it is impossible to know all attacks that the system will be exposed to. Besides, uncovered vulnerabilities remain unresolved, often for many years, even in organizations that are particularly exposed to espionage. Still, far fewer vulnerabilities than normal software faults are typically reported [23]. These are reasons why there is a false sense that software security is not a big problem, and that related vulnerabilities are not prioritized over other software faults. However, considering the consistent rate of vulnerability reports and the immeasurable cost of insecure software, a systematic study of vulnerability characteristics and how they are created during the software development process is a subject of immediate need.

The fundamental way of solving the security problem is by building secure software, defending against exploitation from the earliest stages of development, with a consistent maintenance of a "security-push" throughout the whole development life-cycle. The tools of software engineering can, therefore, play an integral part in supporting security development activities, becoming powerful aids in reducing the number and severity of potential vulnerabilities. A major challenge is that available approaches for ensuring security during the software development are outdated for today's practices. Today's software development business requires high-speed software delivery from the development team. In order to provide fast delivery of products, organizations have made transformations from their conventional development


*ARES '17, August 29-September 01, 2017, Reggio Calabria, Italy*

ACM ISBN 978-1-4503-5257-4/17/08…$15.00
https://doi.org/10.1145/3098954.3103172

approach to agile development methods. These methods have a huge impact on how software is developed worldwide.

Back in day of the waterfall era, developers and operators lived in different worlds. Huge software projects took years to complete, and were then handed off to operations. With the advent of agile development, all this changed. With the focus on early release and short development cycles, software is being deployed at the sharp end much quicker than before, often with less testing than previously.

The rest of this paper is structured as follows: In Section 2 we present relevant background material. In Section 3 we introduce recent trends in software security, and then propose a metrics-based approach in Section 4. We discuss in Section 5, and offer conclusions in Section 6.

## 2 BACKGROUND

Software security is the property of being able to withstand attack [16]. It is tempting to believe that this is a property that is only important in security software, but in reality it is important in all software that is somehow exposed to external input [14]. The classic example here is Adobe's Acrobat Reader, which is a seemingly simple application for viewing Portable Document Format (PDF) files; certainly not security software by any stretch of the imagination. However, Acrobat Reader contained an error that allowed an attacker to craft a special PDF file which when opened on a victim's computer allowed execution of malware code that effectively gave the attacker full remote control of the victim's computer.

### 2.1 Cloud and DevOps

According to Bass, Weber and Zhu, "DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality." [3].

As illustrated in Figure 1, DevOps removes the wall that divides the development and operations teams in an organisation.

Cloud computing is an important enabler for DevOps, since it allows rapid deployment in a virtual infrastructure; a new version can be launched in a matter of hours [28].

### 2.2 DevOps Security in the Literature

DevOps seems to still be fairly high in the hype cycle, but much of the literature related to DevOps security seems to be blog entries and other non-validated channels. Zhu et al. [28] claim that DevOps enable "reliable" rapid deployment, but do not delve into whether this also covers "secure". In an interview with Hulme [11], Muntner takes issue with the notion that DevOps provides any security benefits in and of itself. Kim [15] argues that by integrating security testing into the daily operations of Dev, defects are found (and fixed) more quickly than before. However, Kim seems reluctant to leave potential security defects to post-deployment testing, stating "...it must be tested before the code is deployed". 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" [16].

Fitzgerald and Stol [10] advocate "continuous security" to make security a key concern in all development phases, but do not offer any tangible advice on how to accomplish this, other tan wishing for "a smart and lightweight approach" to identify software vulnerabilities. However, they do point out that security may have to be improved even when nothing else changes, e.g., if other events cause a loss of public confidence in a software solution.

## 3 SOFTWARE SECURITY TRENDS

### 3.1 Science of Security

Science of Security (SoS) is an area of research that seeks to apply a scientific approach to the study and design of secure and trustworthy information systems [9]. ScienceâĂŹs core goal is to develop fundamental laws that let us make accurate predictions. In software security, the only prediction we can usually make confidently is that a system will eventually fail when faced with sufficiently motivated attackers. However, the need and opportunity exists to develop a foundational science to guide the development and understand the security and robustness of the complex systems on which we depend. Suitable metrics, for example, would let designers evaluate alternative designs and determine which is more secure for a given deployment (cf., Section 4). Designers would also be able to reason about the minimum capabilities and effort an attacker needs to violate the security properties.

Security research is a long way from establishing a scientific approach based on the understanding of empirical evaluation and theoretical foundations as developed in other sciences, and even from software engineering in general. The area suffers from a lack of credible empirical evaluation, a split between industry practice and academic research, and a huge number of methods and method variants, with differences little understood and artificially magnified. There is little evidence of how to implement security practices in the software industry, much less in an agile context. In 2010, Alnatheer et al. [1] found 62 papers on the topics of agile and security, from these, only five were empirical. Empirical studies are a powerful approach to be used in security research. In particular, longitudinal extended case studies [7] and ethnographic methods [22] are helpful in generating rich and detailed accounts of software project teams, their interactions with project stakeholders, and their approaches for delivering products, as well as in-depth accounts of their experiences. By performing real-world type of empirical studies, results from research can also be faster introduced and validated in practice.

### 3.2 Agile Security

The goal of agile development is to work with the customer to explore their requirements and deliver a final software product incrementally [8, 19]. One can hypothesize that the

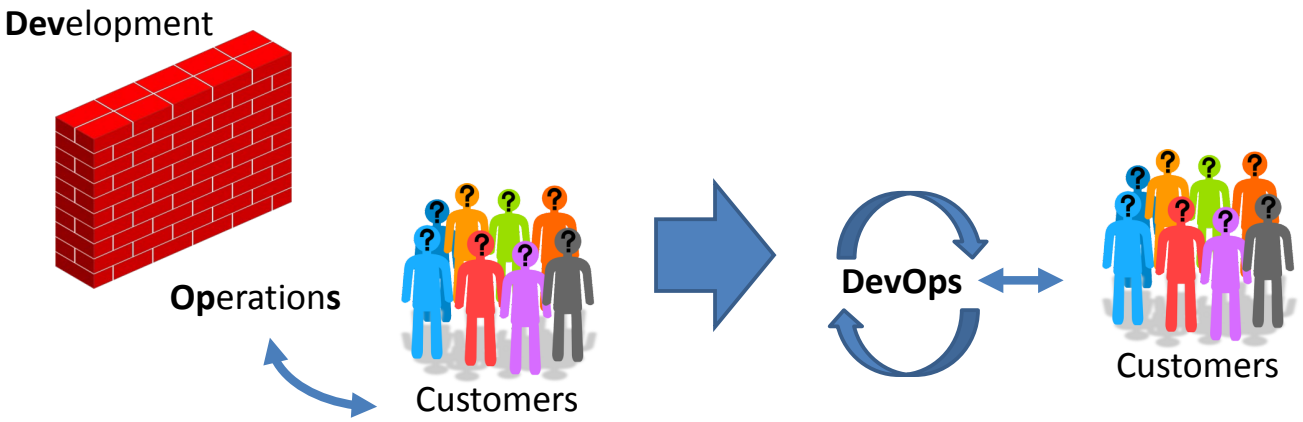


**Figure 1: Transition from traditional development to DevOps**

opportunity to quickly (within a few iterations) address a new security flaw will be an advantage of agile over traditional software engineering approaches. However, agile projects often focus on immediate features and functionality requested by the stakeholders over security requirements [2]. An important misconception is that addressing security delays the development process [27].

Trying to amalgamate security initiatives into an efficient well-built application can be an overwhelming task. With the possible exception of Microsoft SDL Agile[1], there are no security engineering practices developed specifically for agile processes. Traditional security engineering processes and models, such as Microsoft Security Development Lifecycle, Cigital Touchpoints, Common Criteria, measurement models such as the Building Security in Maturity Model (BSIMM)[17], and standards such as the ISO/IEC 27034, are all based on a traditional, prescriptive waterfall approach. However, agile development does not fit the sequential use of activities or the requirements for extensive documentation of these traditional security engineering processes.

Furthermore, the suitability of traditional security engineering processes has rarely been empirically evaluated in industrial agile development settings. Current research focuses on theoretically analyzing if a certain security practice can adhere to agile principles, concluding that some activities are suitable for agile development and some are not. Thus, there is little empirical evidence on how to implement security practices in agile software development. Besides, there is a need to better understand how to apply security practices across different industrial contexts.

## 4 CLOUD SECURITY METRICS AND DEVOPS

According to NIST 500-37 [2], a metric is a "standard of measurement that defines the conditions and the rules for performing the measurement and for understanding the result of a measurement". Metrics are commonly used to set the boundaries and margins of the quality levels that ICT systems are able to provide. The importance of security quantification for ICT systems trustworthiness is extensively recognised by academic research, practitioner and standards organizations such as ENISA [25], CIS [6] and NIST [26]. Despite the conspicuous advantages of this approach, measuring the level of security in a given piece of software is notoriously difficult [13], and instead it has been argued that the next-best thing is to measure second-order effects, i.e., measure the software security activities that are performed by the developers as part of the development process [17]. One such activity is the ability to make quick changes in the code if a vulnerability is discovered in operations. A closely related activity is ensuring that flaws or bugs that are discovered in operations are fed back to development. With the wall between development and operation torn down, these two activities are much easier to perform.

In the case of DevOps, a promising approach relies on the elicitation of security metrics during the Software Development Life Cycle (SDLC) in order to allow the continuous/agile evaluation of required vs. achieved security levels. This approach has proven its usefulness for improving the levels of security assurance and transparency in cloud computing systems [20] [24] [12]. From a high-level perspective, the proposed approach consists of the six incremental stages shown in Figure 2 and presented in the rest of this section.

[1] https://www.microsoft.com/en-us/SDL/Discover/sdlagile.aspx

[2] http://csrc.nist.gov

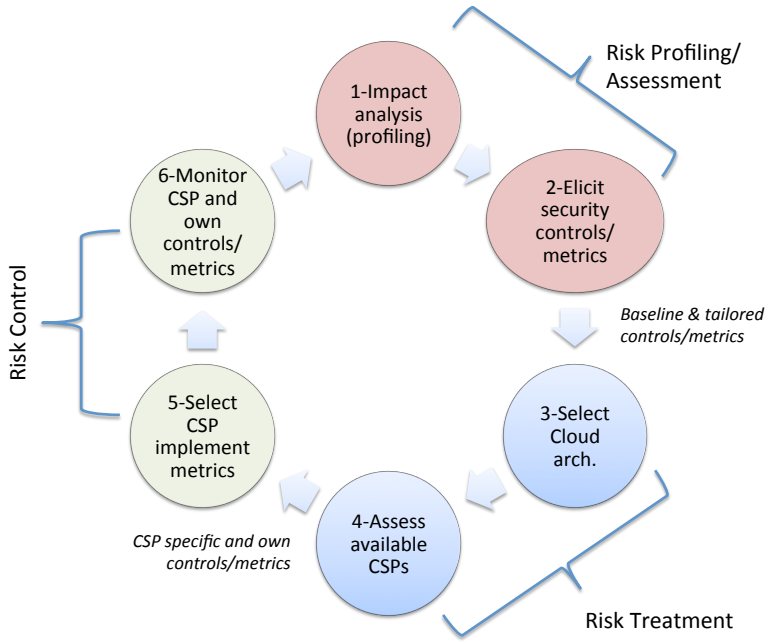


**Figure 2: DevOps and the cloud security metrics life-cycle**

The key elements for the successful adoption of a secure DevOps approach particularly suited for the cloud are the developer organization's understanding of the *(a)* cloud-specific characteristics influencing the software development, *(b)* the architectural components for each cloud service type and deployment model, *(c)* along with each cloud actor's precise role in orchestrating a secure ecosystem. The cloud customer's confidence in accepting the risk from using cloud services depends on how much trust they place in the entities orchestrating (and developing) the cloud ecosystem.

The risk management process ensures that issues are identified and mitigated early in the development process and followed by periodic reviews aligned to the agile DevOps vision. As cloud customers and the other cloud actors involved in securely orchestrating a cloud ecosystem have varying degrees of control over cloud-based IT resources (including software components), they need to share the responsibility of implementing and monitoring the security requirements. Cloud actors also need to assess the correct implementation and continuously monitor all identified security controls which will become part of the cloud service to develop. The approach shown in Figure 2 is a cyclically executed process composed of a set of coordinated activities for overseeing and controlling risks during the DevOps life-cycle. This set of activities consists of the following tasks:

- Risk Assessment.
- Risk Treatment.
- Risk Control.

These tasks collectively target the enhancement of DevOps security in cloud systems, which goes beyond the capabilities offered by widely used security control frameworks and methodologies introduced in Section 2.

A risk-based approach to DevOps for cloud services is an holistic activity that should be integrated into every aspect of the developer organization, from planning and system development life cycle processes (Steps 1 – 2 in Figure 2) to security controls/metrics allocation (Steps 3 – 5). The resulting set of security controls (baseline, tailored controls, controls inherited from providers and under customer's direct implementation and management) lead gradually to the creation of the security metrics in Step 5. Elicited metrics can be monitored during the life-cycle of the cloud service in order to enable the agile (and secure) iterations as required by DevOps.

## 5 DISCUSSION

DevOps is closely tied to rapid release cycles. Frequent releases can lead to big problems if operations and development are siloed; new releases will be deployed, and first-line support may find themselves in a situation where they have no way of knowing whether an issue reported by users is a bug or a new feature.

Ben Othmane et al. [4] 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 [5]. 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, and updating something which is only sold as a service is clearly less arduous.

Prioritsation of defects is challenging. Highly critical security defects can be defined as blocking defects, and as such are easier to deal with - they have to be fixed as quickly as possible. Level 2 or lower security defects are more tricky; they have to compete with all other feature requests and defects, and risk getting pushed back at every junction. On approach to deal with this might be to set a time limit for all lower-priority security defects that are discovered. This will acknowledge that there may be more important issues for the developers to fix right at the moment, but will take into account that the likelihood of a security defect being exploited will increase as time goes by, and eventually it will become a number 1 priority.

It could be argued that software security education of developers is more important in agile development than in traditional waterfall. However, exactly how much is needed is a matter for debate. We certainly don't think that it will be possible to teach every developer to be a software security expert, but we could aspire to teach every developer enough to enable them to identify areas where they would benefit from the advice of an expert [13]. It has been claimed that if we can only make developers *think* about security, the number of security defects are reduced by 50%.

Tools are important, and anything that can be automated should be automated. This is particularly true in DevOps, where 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.

The Building Security in Maturity Model (BSIMM) [17] states the necessity of having a Software Security Group in any software development organisation, but based on industry reports it seems even more important to have security champions who are part of the development teams. An external SSG who performs short-term helicopter-style incursions will inevitably be perceived as an outside agent hindering progress. Another issue is the size of the development organisation; many of the BSIMM organisations have hundreds and thousands of developers, and it is not immediately clear if their experiences are applicable to small European firms with tens of developers.

The main notion behind approaches like BSIMM is the existence of quantitative/qualitative security metrics that can be related to the SDLC. In Section 4 we have presented a promising approach for eliciting and monitoring in a continuous way the security of cloud services, just as required by DevOps. The usage of cloud security metrics for similar purposes (e.g., security certification[3], and continuous security monitoring[4]) is still on its early days, and thus more validation is necessary to make it suitable for industrial adoption. Particular efforts are needed for developing cloud-adapted risk management methodologies suitable for DevOps, along with relevant control frameworks and security metrics beyond those available nowadays to practitioners.

Some argue that security is just another quality issue, but a long history shows us that saying it does not make it so. More concretely, since many security defects are non-obvious, they can lie dormant for years before discovery, as was the case of the Heartbleed[5] bug.

At the end of the day, there is currently much hypothesising and many opinions regarding DevOps and security, but few if any empirical studies that measure a before-and-after effect.

## 6 CONCLUSIONS

DevOps represents both a challenge and an opportunity for software security. It is clear that the ability to make quick changes will shrink the window of opportunity for attackers once a security vulnerability has been discovered, but at the same time it must be acknowledged that rapid releases are not always conducive to thorough testing schemes. We have proposed a metrics-based approach to improve DevOps security, but further empirical research is necessary to establish how DevOps and software security can be reconciled.

## ACKNOWLEDGMENT

The research reported in this paper has been supported by the Norwegian Research Council through the SoS-Agile project.

## REFERENCES

[1] Ahmed Alnatheer, Andrew M. Gravell, and David Argles. 2010. Agile Security Issues: An Empirical Study. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '10)*. ACM, New York, NY, USA, Article 58, 1 pages. https://doi.org/10.1145/1852786.1852860

[2] S. Bartsch. 2011. Practitioners' Perspectives on Security in Agile Development. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*. 479–484. https://doi.org/10.1109/ARES.2011.82

[3] Len Bass, Ingo Weber, and Liming Zhu. 2015. *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional.

[4] L. ben Othmane, P. Angin, H. Weffers, and B. Bhargava. 2014. Extending the Agile Development Approach to Develop Acceptably Secure Software. *IEEE Transactions on Dependable and Secure Computing* (2014).

[5] 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.

[6] Boyle K., *et.al.* 2010. *The CIS security metrics*. Technical Report TR-28. Center for Internet Security. http://benchmarks.cisecurity.org/

[7] Michael Burawoy. 1998. The Extended Case Method. *Sociological Theory* 16, 1 (1998), 4–33. https://doi.org/10.1111/0735-2751.00040

[3] https://cloudsecurityalliance.org/star/

[4] http://www.specs-project.eu/

[5] http://heartbleed.com/

[8] Claudia de O. Melo, Daniela S. Cruzes, Fabio Kon, and Reidar Conradi. 2013. Interpretative case studies on agile team productivity and management. *Information and Software Technology* 55, 2 (2013), 412 – 427. https://doi.org/10.1016/j.infsof.2012.09.004 Special Section: Component-Based Software Engineering (CBSE), 2011.

[9] D. Evans and S. Stolfo. 2011. Guest Editors' Introduction: The Science of Security. *IEEE Security & Privacy* 9, 3 (May 2011), 16–17. https://doi.org/10.1109/MSP.2011.50

[10] Brian Fitzgerald and Klaas-Jan Stol. 2015. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* (2015), –. https://doi.org/10.1016/j.jss.2015.06.063

[11] 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/

[12] J. Luna, A. Taha, R. Trapero, and N. Suri. 2015. Quantitative Reasoning about Cloud Security Using Service Level Agreements. *In Trans. on Cloud Computing* 99 (2015).

[13] 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

[14] Martin Gilje Jaatun and Per Håkon Meland. 2011. Special Issue from the 5th International Workshop on Secure Software Engineering. *International Journal of Secure Software Engineering* 2, 4 (2011), i–ii. http://www.igi-global.com/Files/Ancillary/1947-3036_2_4_Preface.pdf

[15] 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

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

[17] Gary McGraw, Sammy Migues, and Jacob West. 2015. Building Security In Maturity Model (BSIMM 6). (2015). http://bsimm.com.

[18] Peter Mell and Tim Grance. 2011. The NIST definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg. (2011).

[19] C. Melo, D. S. Cruzes, F. Kon, and R. Conradi. 2011. Agile Team Perceptions of Productivity Factors. In *Agile Conference (AGILE), 2011*. 57–66. https://doi.org/10.1109/AGILE.2011.35

[20] Jolanda Modic, Rubén Trapero, Ahmed Taha, Jesus Luna, Miha Stopar, and Neeraj Suri. 2016. Novel Efficient Techniques for Real-Time Cloud Security Assessment. *Elsevier Journal on Computer & Security* (Sep. 2016), 1–18. https://doi.org/10.1016/j.cose.2016.06.003

[21] Charlene O'Hanlon. 2006. A Conversation with Werner Vogels. *Queue* 4, 4, Article 14 (May 2006), 9 pages. https://doi.org/10.1145/1142055.1142065

[22] Carol Passos, Daniela S. Cruzes, Tore Dybå, and Manoel Mendonça. 2012. Challenges of Applying Ethnography to Study Software Practices. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '12)*. ACM, New York, NY, USA, 9–18. https://doi.org/10.1145/2372251.2372255

[23] Yonghee Shin and Laurie Williams. 2011. Can traditional fault prediction models be used for vulnerability prediction? *Empirical Software Engineering* 18, 1 (2011), 25–59. https://doi.org/10.1007/s10664-011-9190-8

[24] Rubén Trapero, Jesus Luna, and Neeraj Suri. 2016. Quantifiably Trusting the Cloud: Putting Metrics to Work. *IEEE Security & Privacy* 14, 3 (2016), 73–77. https://doi.org/10.1109/MSP.2016.65

[25] Trimintzios, P. 2011. Measurement Frameworks and Metrics for Resilient Networks and Services. Discussion Draft. European Network and Information Security Agency. (2011).

[26] W. Jansen. 2010. *Directions in security metrics research*. Technical Report TR-7564. National Institute for Standards and Technology.

[27] Carol Woody. 2013. Agile Security - Review of Current Research and Pilot Usage. SEI White Paper. (2013).

[28] L. Zhu, D. Xu, A. B. Tran, X. Xu, L. Bass, I. Weber, and S. Dwarakanathan. 2015. Achieving Reliable High-Frequency Releases in Cloud Environments. *IEEE Software* 32, 2 (Mar 2015), 73–80. https://doi.org/10.1109/MS.2015.23