Infiltrating Security into Development: Exploring the World's Largest Software Security Study

Charles Weir
Lancaster University
England
c.weir1@lancaster.ac.uk

Sammy Migues
Synopsys
USA
Samuel.Migues@synopsys.com

Mike Ware
Synopsys
USA
Michael.Ware@synopsys.com

Laurie Williams
North Carolina State University
USA
lawilli3@ncsu.edu

ABSTRACT

Recent years have seen rapid increases in cybercrime. The use of effective software security activities plays an important part in preventing the harm involved. Objective research on industry use of software security practices is needed to help development teams, academic researchers, and educators to focus their activities.

Since 2008, a team of researchers, including two of the authors, has been gathering objective data on the use of 121 software security activities. The Building Security In Maturity Model (BSIMM) study explores the activity use of 675,000 software developers, in companies including some of the world's largest and most security-focused.

Our analysis of the study data shows little consistent growth in security activity adoption industry-wide until 2015. Since then, the data shows a strong increasing trend, along with the adoption of new activities to support cloud-based deployment, an emphasis on component security, and a reduction in security professionals' policing role. Exploring patterns of adoption, activities related to detecting and responding to vulnerabilities are adopted marginally earlier than activities related to preventing vulnerabilities; and activities related to particular job roles tend to be used together. We also found that 12 developer security activities are adopted early, together, and notably more often than any others.

From these results, we offer recommendations for software and security engineers, and corresponding education and research suggestions for academia. These recommendations offer a strong contribution to improving security in development teams in the future.

CCS CONCEPTS

• **Security and privacy** → **Software security engineering**; • **Applied computing** → *Cross-organizational business processes*; • **Social and professional topics** → *Industry statistics.*

KEYWORDS

Software engineering, Software security, Developer centered security, Software security group, Secure software development lifecycle, SDLC, DevSecOps

ACM Reference Format:
Charles Weir, Sammy Migues, Mike Ware, and Laurie Williams. 2021. Infiltrating Security into Development: Exploring the World's Largest Software Security Study. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3468264.3473926

1 INTRODUCTION

According to the NIST National Vulnerability Database [34], reported vulnerabilities continue to rise through 2020 including a 600% rise in cybercrime in 2020 due to the covid-19 pandemic [10]. However, the growth in cybersecurity spending is expected to slow, and corporate boards are questioning the effectiveness of cybersecurity activities as implemented across enterprises globally [12]. As organizations seek to address mounting cybersecurity risk as efficiently as possible and to comply with regulations, a myriad of activities is available for improving software security. Organizations desire guidance on which of many possible software security activities to undertake first and how to structure adoption.

The goal of this paper is to aid software and security engineers, software engineering educators, and software engineering researchers in understanding opportunities for software security activity improvement, education, and research through an analysis of records of software security activity by many software development teams over a 12-year period.

Organizations prefer to adopt new practices based upon understanding their use in organizations similar to their own [28]. As a result, a good process to identify such opportunities is to base them on the practice of leading organizations that have focused on security, leveraging the trials and errors of teams as they 'infiltrate' security into their development practices. Accordingly, our first research question is:

***RQ1*: What have been the patterns of adoption and usage of software security activities by software development teams in security-focused companies?**

Software security is a fast-moving field. Accordingly, adoption trends over an extended period provide insight, leading to a second research question:

***RQ2*: What have been the trends in adoption of developer security activities industry-wide?**

1.1 Introducing the Study

Since 2008, the Building Security In Maturity Model (BSIMM) team has provided organizations with support in obtaining security guidance through an assessment process [27]. Each BSIMM assessment is a major undertaking involving approximately one consultant-month of work, including over a dozen detailed interviews with company experts and the creation of a report for the company. Each assessment evaluates which of 121 software security 'activities' have been adopted by the organization. These activities range from "Ensure host/network security basics in place" to the rarely-found "Build/use abuse cases in QA process" [1].

The model suits the needs of organizations with a focus on software security: every participating organization must have a Software Security Group. Named participants include Microsoft, Qualcomm, SAP, Visa, Citigroup, and PayPal. As Section 5.1 will discuss, at least 55 of the companies named are in the Forbes Global 2000 list of the world's largest public companies; and the list also includes many trailblazers in large company software security, including 70% of the members of SAFECode, an early initiative in this field.

Working with the companies in this list, two of the authors and a team of assessors have built a highly sensitive dataset of 322 objective assessments of the security practices of 211 companies throughout the world over a 12-year period, relating to the work of some 675,000 software developers. We are aware of no similar work of this magnitude in the field of development security. The BSIMM team has published 11 yearly reports containing a high-level descriptive analysis of that year's data, all publicly available to those willing to provide contact information; the latest is the BSIMM11 report from 2020 [27].

To address the research questions in this paper, we studied participants' software security development activities. This paper, therefore, explores findings from the analysis of the BSIMM dataset related only to the 43 security activities used by *software development teams*: groups of software engineers, IT staff, and Quality Assurance (QA) specialists. Effective security requires other organizational roles, of course, but their activities are out of scope for this paper. We studied both *adoption*, starting new activities, and continued *usage* of activities.

To explore RQ1, we started with two approaches: clustering algorithms looked for activities used or adopted together; and charts and graphs explored the most used, most adopted, and most discarded activities.

To look for further patterns in the data, we used a segmentation based on prior work [5, 40, 49] into *Prevention, Detection*, and *Response* activities. We used statistical hypothesis tests to explore if this segmentation contributed to adoption patterns within individual companies (RQ1) and within the industry as a whole (RQ2). Finally, we used graphical and statistical analysis to explore industry adoption further (RQ2).

1.2 Contributions

The main contributions of this paper are as follows:

(1) A longitudinal analysis of the trends in developer security activities in industry over a period of 12 years;
(2) The identification of a small set of software security activities that are adopted first, together, and most often by software development teams; and
(3) The observation that software security activities used together tend to be those supported by specific other job roles.

The rest of this paper is structured as follows. Section 2 discusses related work; Section 3 introduces the BSIMM; Section 4 describes our methodology; Section 5 describes the results we found; Section 6 explores these results, answers the research questions, and discusses threats to validity; and Section 7 summarizes the findings and suggests future work.

2 RELATED WORK

Software security is a major and pressing problem with current software systems. This section explores frameworks and related studies into how companies and development teams are addressing security.

2.1 Software Security Practice Frameworks

Several frameworks in addition to BSIMM provide guidance or enumeration of software security practices. The OWASP Software Maturity Model (SAMM or OpenSAMM) [36] is an open framework to help organizations assess, formulate, and implement, through a self-assessment model, a strategy for software security practice adoption that is tailored to the specific risks facing the organization. OpenSAMM is a prescriptive model in that it posits that an organization matures its cybersecurity efforts by progressing through the maturity levels.

The US National Institute of Standards and Technology (NIST) Cybersecurity Framework (CSF) [33] provides organizations with a structure to aid in understanding and improving cybersecurity risk. The framework is organized around five functions: identify, protect, detect, respond, and recover. In this paper, we utilize three of the five NIST functions in our analysis: identify, detect, and respond; but we replace the term 'identify' with its goal of 'prevent' to align with the frequently used model [5, 40] of *prevent, detect, respond.*

The US Department of Defense (DoD) Cybersecurity Maturity Model Certification (CMMC) [8] framework combines cybersecurity standards and best practices and maps 171 practices into five maturity levels that range from basic cyber hygiene to advanced optimization. For a given CMMC level, the associated practices are designed to reduce risk against a specific set of cyber threats.

The ISO/IEC 27001 [18] and NIST SP 800-53 [32] standards provide requirements for establishing, implementing, operating, monitoring, maintaining, and improving the security of a digital information management system. The ISO/IEC 27034 Application Security standard [17] offers definitions, concepts, principles and

[1]Figure 6 contains a list of activities with identifier codes, and detailed descriptions may be found in the BSIMM11 report [27].

non-prescriptive guidance to help organizations integrate security into the processes used for managing their applications. Finally, the OWASP Application Security Verification Standard (ASVS) [37] provides developers with a list of requirements for secure development.

None of these frameworks have databases of company results with which we could compare our study results.

2.2 Software Security Practice Studies

Cisco [6] surveyed 4,800 active IT, security, and privacy professionals from 25 countries about, first, their organization's use of 25 security practices spanning governance, strategy, spending, architecture, and operations; and, second, their program's level of success across 11 high-level security objectives. They found that having a proactive, best-of-breed technology refresh strategy allows an organization to keep up with business growth; and that having a well-integrated technology stack improves recruitment and retention of security talent. The report provides valuable industry data; however, the responses were self-reported, and the result is not longitudinal. Since the Cisco security software practices do not map cleanly to the BSIMM practices, the two studies cannot usefully be compared.

Such et al. [41] conducted a comprehensive review of the use of 25 assurance techniques from the ISO/IEC 27001 standard [18] using a large-scale stakeholder-supported study with 14 interviews and 115 respondents to an online survey. The responses identified the most cost-effective techniques to be architectural review, vulnerability scans, and penetration tests.

Further surveys of security activity by software developers included Venson et al.'s survey [44, 45] of 110 software professionals, which found that security practices had been applied thoroughly in the projects, but revealed high variability in secure software development effort across the participants' projects. Oyetoyan et al. [38] surveyed security practice usage, competence, and training needs in two organizations, finding that regardless of cost or benefit, skill drives the kind of activities that are performed, and that secure design may be the most important training need. And Jaatun et al. [19] used the BSIMM structure for a survey and interview study of software security activities used by 20 Norwegian companies, finding that those companies excelled at compliance and policy activities.

Other research teams studied historical data. Morrison [29] defined the Security Practices Evaluation Framework (SP-EF), a measurement framework for software development security activities, and evaluated the framework on historical data and industrial/open-source projects [30]. Kwon and Johnson [21] conducted an empirical analysis of data from 2,386 healthcare organizations to identify how different types of security investment affect subsequent security failures. They distinguished *proactive* activities that happen before a security incident from *reactive* activities that occur after an incident. Kwon and Johnson [22] also analyzed breach disclosure and detailed data on security investments in the healthcare sector. In both reports, they found that proactive security investments are associated with lower security failure rates and longer intervals before subsequent breaches than reactive investments.

Considering literature surveys, Venson et al. [46] conducted a mapping study to classify and analyze 54 papers in the literature related to the impact of security on software development costs. Perform Security Review, Apply Threat Modeling, and Perform Security Testing were the three most frequent activities related to cost, and the Common Criteria [15] was the most applied standard. Another systematic literature review by Wen [48] found the most studied software security practice to be verification, which includes design review, code review, and security testing.

Moving to the theory behind developer security, Barth et al. [2] challenged the conventional wisdom that proactive security is superior to reactive security, using a game-theoretic model. This showed that reactive security can be competitive with proactive security as long as the reactive defender learns from past attacks instead of myopically overreacting to the last attack. Maguire and Miller [23] contended that resistance to proactively implementing application-layer security may stem from the perceived expense and the idea that risk is a natural part of doing business.

Our work complements these studies by providing insights based on higher fidelity data over a longer timeframe.

3 THE BSIMM STUDY

In 2008, Gary McGraw, Sammy Migues, and Brian Chess created the BSIMM as a *descriptive* model to allow external assessment of organizations' state-of-practice in secure software development. The BSIMM framework is built around the assessment of the activities an organization may adopt in support of its software security initiative. The number of activities evolves with each version: BSIMM11, published in 2020, defines 121 activities. Activities are categorized into three 'maturity levels'; and 12 'practices' grouped into 4 'domains' [27]. All the activities are proactive [21]; the handling of security events and fixing of vulnerabilities are assumed to happen and are not considered activities.

Each assessment is carried out in cooperation with the organization's Software Security Group (SSG), whose role is to manage an organization-wide program to instill software security activities. For each assessment, security professionals, including two authors of this paper, conduct approximately 20 in-person interviews. These typically include the SSG leader and representative SSG members; plus samples of those whose roles involve implementing security, and of those whose roles are affected by the security processes. The interviewers create a 'scorecard' report of an organization's software security activities, including a comparison with other similar organizations. They also record in a database the company demographic data and which activities were practiced.

Annual BSIMM reports provide high-level findings based upon descriptive analysis. The latest such report is freely available to those willing to provide contact information [25]. Each report provides detailed descriptions of the activities; a grouping of the activities into 12 practices; adoption percentages for each activity; observations of industry trends noted during the collection; and a description of several different 'adoption approaches' for companies.

These reports contain the names of some of the companies assessed, which opens the possibility of re-identification of companies even in anonymized data, so the dataset cannot be contributed to the community. All the authors are subject to Non-Disclosure Agreements to have access to the data.

4 ANALYSIS METHODOLOGY

For academic analysis, we used an anonymized version of the BSIMM dataset in spreadsheet form. This contained three types of data:

(1) Demographic data for each company assessed, such as the industry verticals, number of developers, size of Security Services Group (SSG), and dates of each assessment;
(2) Descriptions of each software security activity; and
(3) For each assessment, the activities practiced by the company.

Figure 1 provides an overview of the analysis we performed on this data. To achieve independent insights, we used only the 'raw' data, ignoring the constructs used in the BSIMM reports such as the report number in which each assessment was reported, and the activity categorizations of 'maturity level', 'practice', and 'domain' .

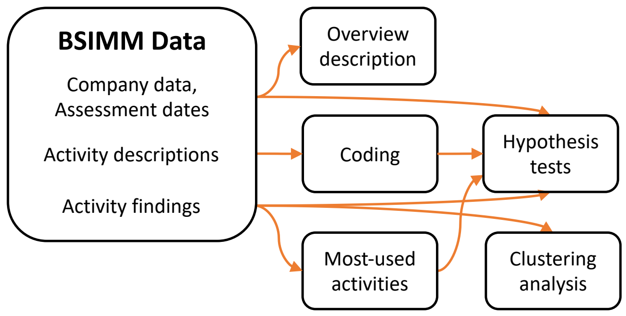


Figure 1: Overview of the Analysis Methodology

One difficulty we identified is that the field of software security activities changes fast. Since the goal of our study is to aid software engineers and academics to understand opportunities for practice improvement, we opted to limit the analysis related to RQ1 to the last five years to provide a timely view of current trends — specifically, since the start of 2015.

To categorize the activities, we applied dual coding to the detailed description for each. Since our research questions address only development teams, we coded each activity as 'carried out by developers' (a *developer activity*) or not. We then coded these developer activities into Prevention, Detection, and Response (see Section 2.1): Prevention activities slow the introduction of vulnerabilities during product design or implementation; Detection activities discover vulnerabilities that have been injected during design or implementation; and Response activities are used after the discovery by attackers and/or researchers of vulnerabilities in a deployed product.

Two authors first coded independently; then compared differences to identify discrepancies in interpretation; then re-coded all the activities; and finally agreed on the coding for the few remaining differences. We used the Cohen's Kappa calculation of Inter-Rater Reliability [13] to quantify agreement for each of the first two coding cycles.

4.1 Hypothesis Testing Methods

To achieve methodologically defensible statistical results, we wanted to avoid the poor research practice of 'data dredging'—cherry-picking interesting-looking results from many statistical tests on the same data—since this leads to conclusions that are unlikely to be replicable [16]. We, therefore, used two best practices: defining hypotheses only before accessing the data and correcting the analysis for multiple uses of the same data.

In a gold standard research project [7] we would make predictions, define the analysis, and write programs to do the analysis, all before we collected the data. However, in this case the data was already collected and available before the start of the analysis. To compensate, we randomized the raw data (without analyzing it) to create a dummy dataset. We then worked out the analysis approaches for the given hypotheses and implemented the approaches in software to work on the dummy dataset. This step was sufficient to verify that the implementation worked correctly. We then substituted the true dataset for the dummy one and recalculated the results.

To correct the analysis for multiple uses of the same data, we used the Bonferroni correction, which divides the threshold for statistical significance by the number of hypotheses tested [39].

Our first hypotheses were those to answer **RQ1**. We hypothesized that certain types of activities would tend to be adopted earlier than others:

H1: Development teams adopt Response activities before Detection activities.

H2: They adopt Detection activities before Prevention activities.

H3: Development teams adopt the most-used practices earlier than other practices

Our remaining hypotheses address **RQ2**. Software security is a fast-moving discipline, and we hypothesized that the data would show continuous industry progress:

H4: During the period 2008-2020, companies have increasingly adopted Prevention activities

H5: During the period 2008-2020, companies have increasingly adopted Detection activities

In each case, the corresponding null hypothesis would be that there is no evidence of which kind of activity comes first. That makes five tests on the data set, and accordingly, for the Bonferroni correction, we used a p-value for significance of: $0.05/5 = 0.01$.

In testing hypotheses H1 through H3 we found a problem: each BSIMM assessment records only which activities are present at the assessment time and has no record of when each activity had been adopted. So, a single assessment provides no information to help confirm or reject each hypothesis. We, therefore, looked only at the 70 companies that had undergone multiple assessments: for each pair of consecutive assessments and each type of activity (Prevention, Detection, Response) we calculated the proportion of the later assessment's activities that were newly adopted in the later assessment. A larger proportion means later adoption for that type of assessment.

The null hypothesis for H1 thus becomes: that the distributions of proportions (Detection, Response) are the same—i.e., the distribution of differences between the pairs of measurements is consistent with a zero-based distribution. We plotted this distribution of differences and used the Shapiro–Wilk test [43] to see if it was consistent with the Normal distribution. If so, we used a one-sided Students T-Test to assess the hypothesis; if not, we used the Wilcoxon rank-sum (Mann-Whitney) test [43]. We used the same approach to assess hypothesis H2 and (after identifying the most-used activities) H3.

Moving to RQ2, hypotheses H4 and H5 were first tested using linear regression. We used the Pearson R calculation to fit a line to the counts of each kind of activity found on each date. To check the preconditions for Pearson R, we plotted the data and used the Shapiro–Wilk test to test for a Normal distribution of the residuals. If (as we expected) the preconditions were not satisfied, we could then use techniques such as graphical plots and rolling averages to see what was happening in the data; and the Mann Whitney U test to look for significant increases in the population values between years.[2]

Even if a hypothesis is statistically significant, the effect might be small. We therefore also calculated, for each accepted hypothesis, a measurement of the effect involved.

4.2 Descriptive Analysis Methods

To explore the data further than our hypotheses could take us, we used descriptive statistics.

First, to give context for readers, we provided a descriptive overview of the companies assessed and the timing of assessments based on the demographic data. We created graphical summaries of the data most relevant to development teams: team sizes and industry verticals, for example. We also explored the number of assessments per year and the incidence of repeated assessments.

To start exploring RQ1, we then plotted the frequency of adoption of each activity, arranging the activities in order of that frequency to explore if there was any logical cut-off point to justify focusing on a particular subset.

Next we explored how activities clustered together. We considered it likely that the patterns of activity usage and activity adoption would differ. So, we addressed RQ1 with two further questions:

> ***RQ1.1*: Which activities tend to be <u>used</u> together?**

> ***RQ1.2*: Which activities tend to be <u>adopted</u> together?**

To answer these questions, we used Hierarchical (Agglomerative) Clustering [31]. For RQ1.1, in order to find clusters of activities that occurred *together*, even if not *frequently*, we used the following as the 'distance' measurement between two activities:

$$1 - \frac{N_{\mathit{assessments\ that\ found\ both\ activities\ in\ use}}}{N_{\mathit{assessments\ that\ found\ either\ activity\ in\ use}}}$$

For the clustering 'linkage' method we used 'complete', in which the distance between two clusters is the *maximum* distance between any item in one cluster and any item in the other.

For RQ1.2, we again considered companies that had undergone multiple assessments. We clustered the activities adopted in each repeated assessment, using the following as the distance measurement:

$$1 - \frac{N_{\mathit{repeat\ assessments\ that\ adopted\ both\ activities}}}{N_{\mathit{repeat\ assessments\ that\ adopted\ either\ activity}}}$$

Often clustering analysis aims to partition *every* item into clusters; in these analyses we wanted instead to find *any* activities that clustered together. We, therefore, looked for clusters where the distance between all the items was less than an appropriate cut-off point, chosen by increasing the cut-off point until the new clusters stopped showing meaningful relationships to each other.

Finally, to explore RQ2 beyond the simple trends suggested by hypotheses H4 and H5, we used graphical and statistical analysis to explore trends in the assessments over time. We considered that changes in the type of companies undertaking the assessments might be a driver of such change, so we also plotted how some key company statistics changed over time.

For all the analyses, we used Python statistical and graphical packages (Pandas, Statsmodels, Pyplot, Seaborn) hosted in Jupyter Notebooks [20]. Statistical results are quoted using APA conventions [1].

[2]Note that these last tests do not prove hypotheses, since the specific tests were defined after the first data analysis

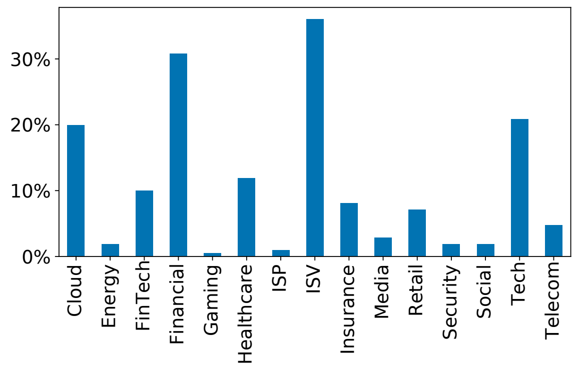


Figure 2: Industry Sectors

5 RESULTS

5.1 Participants and Assessments

The dataset describes a total of 322 assessments of 211 companies. In addition to those named in Section 1, participating companies included Nokia, Salesforce, Cisco, Goldman Sachs, Alibaba, and Verizon. A full listing of the companies who did not choose to remain anonymous can be found in the BSIMM reports (e.g. [27]). Of a sample of 107 companies named in two of the reports, we found 55 in the Forbes Global 2000 list of the world's largest 2000 companies in the world [11, 26, 27]. All had active Software Security Groups.

Figure 2 shows the 15 industry sectors represented: 44% of the companies operated in more than one sector, so the total adds up to more than 100%. As shown, Independent Software Vendors, Financial, Tech, and Cloud are strongly represented. Other sectors, such as Energy, Gaming, and Internet Service Providers have low representation. Participants include many early adopters of security processes, including 9 of the 16 members of SAFECode [35], a global nonprofit organization set up in 2007 to promote scalable and effective software security programs.

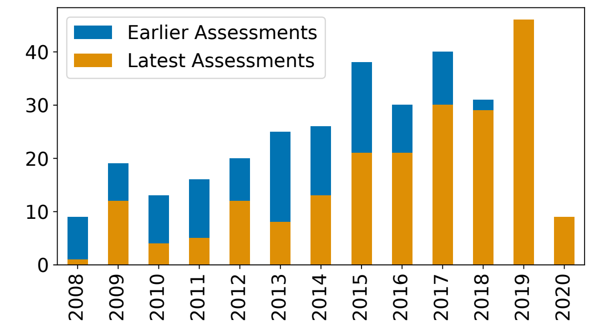


Figure 3: Number of Assessments Carried Out Each Year

In terms of geography, 79% of the organizations are based in America, 17% in the EU or UK, and 4% in Asia-Pacific, reflecting that the assessors are based in the USA.

Table 1 characterizes the distributions of four relevant aspects of the companies assessed: the number of developers, the number of applications produced, and the size of the SSG. For each aspect, the table shows rounded values for the lowest decile, the median, and the highest decile. The large developer numbers emphasize that these are mostly big and therefore high-profile companies, likely to be targeted by attackers or damaged by adverse press coverage; we conclude that their decisions on software security approaches are likely to be worth emulating. The SSG numbers are relatively small for all the companies; remember that the SSG provides the 'evangelist' role for software security and does not include those operating Security Operations Centers, for example.

Table 1: Participant Attributes

Feature	Low 10%	Median	Top 10%
Dev. team size	100	800	7500
Number of Apps	5	175	3000
SSG team size	1	6	35

Summing the number of developers involved in each organization showed that the assessments have covered approximately 675,000 developers.

Figure 3 shows the number of assessments carried out each year. The latest date for the dataset is April 2020, so there are relatively few assessments for 2020. The chart indicates how many assessments are repeated; blue bars show assessments that were superseded later.

One third (70) of the 211 companies involved have received more than one assessment so far, as shown in Table 2. The interval between assessments for a company varied widely, from under a year to ten years, with a median of 2.5 years.

Table 2: Number of Assessments per Company

# Assessments:	1	2	3	4	5
Companies:	141	42	18	7	3

5.2 Coding Developer Software Security Activities

Two authors independently coded the 121 activities, as described in Section 4. The first round of coding by the two authors gave a Kappa coefficient of 0.51, a surprisingly low level of agreement. On discussion, the authors discovered several reasons for the discrepancies, and resolved them as follows:

- First, should work done by the SSG with developers be considered developer activities? With increasing DevOps and cloud usage, developers are increasingly carrying out activities that previously would be considered SSG-only. Although this trend is documented in the BSIMM11 report [27], the activity descriptions for some practices do not distinguish who carries each out. We, therefore, agreed to not include these shared activities in the set of developer activities.
- Activities detecting design problems (such as "Do design review for high-risk apps" could be categorized either as Detection (finding 'flaws') or Prevention (preventing the implementation of flawed code). We chose 'Detection'.
- Activities that are a large-scale response to bugs, such as "Fix all instances of SW bugs found in ops", could be considered Prevention (stopping future defect reports) or Response (to a defect report). We agreed 'Response'.

The second round of independent coding led to a more acceptable Kappa coefficient of 0.89. We resolved the remaining discrepancies in coding by discussing each activity and coming to a consensus. Figure 4 shows the final coding. For consistency, tables and illustrations throughout this paper will use the same colors for Prevention, Detection, and Response activities.

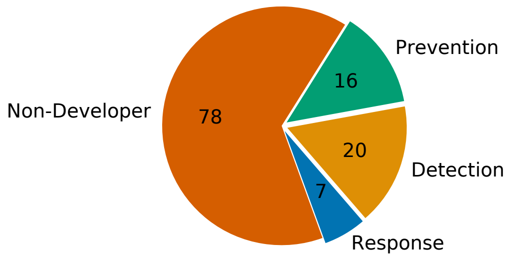


Figure 4: Final Coding Categories

5.3 Descriptive Analysis

As discussed in Section 4.2, Figure 5 considers assessments since 2015. The x-axis shows the 43 developer activities sorted in order of increasing usage; the y-axis shows the proportion of assessments that found each activity. The figure shows a marked jump between a large number of little-used activities and a smaller number of much-used activities. By taking a cut-off point at 50%, as shown by the dashed amber line, we get a set of 12 'most-used activities', which can be found listed at the top of Figure 6.

To determine the prevalence of these most-used activities, we explored to what extent they are found together. We found that 92% of all the assessments found at least half (6) of the most-used activities. We also explored to what extent these most-used activities dominated activity usage. We found that in most assessments (88%), these 12 activities made up more than half the total activities adopted.

To address RQ1.1, we then used the clustering approach as described in Section 4.2. We found the results shown in Figure 6, which shows all 43 developer activities. The code in parentheses after each description is the activity identifier to enable readers to find the full

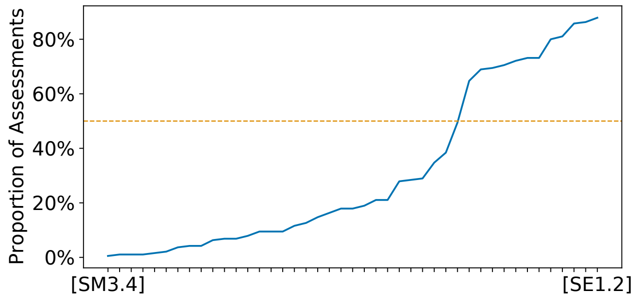


Figure 5: Activities in Order of Increasing Usage

description and related analysis in the BSIMM model [27], followed by P, D or R to indicate the activity type: Prevention, Detection, or Response. The text color further highlights this activity type. The following percentage is the proportion of all assessments in the last five years that found that activity.

As shown, the clustering algorithm found first a tightly-clustered group of 12 items, with other items more distant from each other. The items in this group are identical to the most-used practices discussed above. Figure 6 highlights the 'interesting' clusters (see Section 4.2). The cut-off point at 26% clusters 28 of the 43 activities analyzed.

Turning to RQ1.2 and the analysis of activities *adopted* together, the clustering analysis found fewer clusters, with only three clusters closer than the same cut-off point as above (26%). Table 3 shows the results.

Table 3: Activities Adopted Together

Use orchestration for containers/virtualized env (SE3.5 P)
Use application containers (SE2.5 P)
Do cloud host/network/etc. security basics (SE2.6 P)
Use automated tools along with manual review (CR1.4 D)
Results go to defect mgmt systems (PT1.2 R)
Use automated tools with tailored rules (CR2.6 D)
Security requirements and features drive tests (ST1.3 D)

Table 4 shows the top five activities adopted, the top five dropped, and the proportion of activities involved in each case.

Table 4: Top Activity Changes in Repeated Assessments

Adopted activities:	
31%	Have ops/deployment inventory of applications (CMVM2.3 R)
29%	Identify open source (SR2.4 P)
23%	Results go to defect mgmt systems (PT1.2 R)
21%	Compliance constraints become sw req'ts (SR1.3 P)
20%	Mandatory code review for all projects (CR1.5 D)
Dropped Activities:	
19%	Publish installation guides for security (SE2.2 P)
17%	Security requirements and features drive tests (ST1.3 D)
16%	Use secure coding standards (SR3.3 P)
15%	Pen testers get all app info and use it (PT2.2 D)
13%	Mandatory code review for all projects (CR1.5 D)

5.4 Hypothesis Tests

Recall that Section 4.1 defined five a-priori hypotheses and specified our approach for testing them and, for accepted hypotheses, to calculate the effects involved.

Figures 7 address H1 through H3 using Kernel Density Estimation (KDE) plots of the differences between the adoption counts for the pairs of kinds of activity. The ticks above the x-axis are observations; the plotted shape shows the density of these observations. In all three plots, the Shapiro-Wilk test indicates non-normal distributions, so we used the Wilcoxon signed-rank test rather than a paired T-Test to assess the hypotheses. Using that test, we reject H1 and accept H2 and H3.

Table 5 shows the effect sizes corresponding to H2 and H3. As shown, a Prevention activity is 9% more likely to be adopted in a later assessment than is a Detection one: a small effect. However, a most-used activity (one of the Top 12) is 29% more likely to be adopted before any of the others: a reasonably large effect.

Table 5: Distributions of Proportions of Activities Adopted

	Prevention - Detection	All The Rest - Most Used
Mean	9%	29%
St. Dev.	26%	33%

Moving on to hypotheses H4 and H5, Figure 8 shows a scatter plot and attempted linear regression for trends in each type of activity found in each assessment. The counts are represented as percentages of the total number of developer software security activities of each kind. The figure shows a *decrease* in Detection activity use and no change in Prevention activity use. However, the Shapiro-Wilk test on the residuals gave $p < 0.001$ in both cases, indicating that the residuals fail to follow a normal distribution and that the linear regression is meaningless.

Returning to descriptive analysis, we created two-dimensional KDE plots of the same data, adding the one-year rolling mean of the values. Figure 9 shows the results for Prevention and Detection activities. The results are instructive, showing a wide range of 'security maturity levels' among participating companies which varies year by year, particularly in the early years (prior to 2015). We conclude that the data do not support hypotheses H4 and H5.

Figure 9 shows a jittery decrease in the mean activity counts to about 2015, and then a gradual but consistent increase over the five years starting 2015. Indeed, compared to a baseline of 2015, Prevention activities show an increase of around 100% and Detection activities 30%. Mann Whitney U tests comparing the counts of each type of activity in 2015 with the counts in the final year gave $p < .001$ for Prevention and $p = .001$ for Detection activities. These p-values suggest that the increases are unlikely to be the results of statistical fluctuations.

Figure 10 explores whether these observations reflect a change in the nature of organizations being assessed, showing one-year moving averages of four key attributes: the proportion of companies in finance (Financial or FinTech); the proportion of ISVs; the number of developers; and the size of the SSG team.

6 DISCUSSION

6.1 Adoption and Usage by Individual Companies

Exploring the findings related to RQ1, consider first activities *used* together (RQ1.1). Figure 5 shows a clearly separated set of 12 activities, which make up the majority of activity use. Figure 6 shows they are predominantly Detection (6) and Response (4) activities,

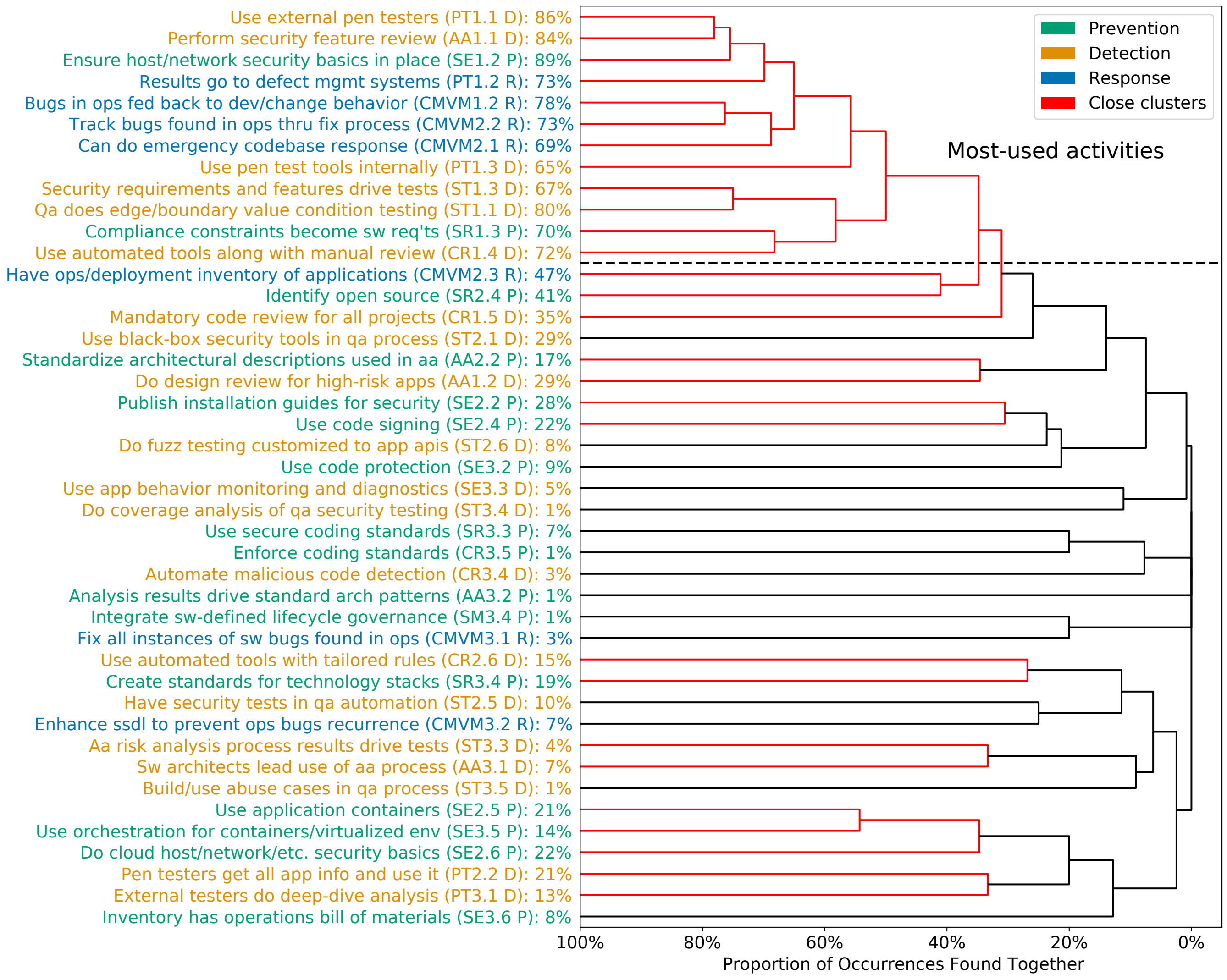


Figure 6: Activities Used Together

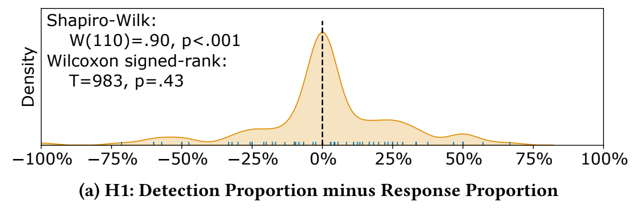
with only 2 Prevention activities. The clustering analysis in Section 5.3 shows that these most-used activities also tend to be used *together*: 88% of assessments found at least half these activities in simultaneous use. The acceptance of Hypothesis H2 in Section 5.4 shows that these most-used activities also tend to be adopted *first*, before any others. To summarize, these 12 are adopted *most often, together, and first*, as revealed by three forms of analysis.

From Figure 6, we also see that clusters of other activities tend to be those carried out by particular roles. 'Ops inventory of applications' and 'Identify open source' are both inventory activities; 'Standardize architectural descriptions' and 'Design review for high risk apps' are both software architect activities; 'Installation guides for security' and 'Code signing' are both activities of a release control team. The remaining clusters are, respectively, activities for DevOps engineering, software architects, developers tasked with cloud implementations, and penetration testers. This observation suggests that adoption of the less popular software security activities tends to be associated with particular job roles, independent of work elsewhere in the organization. This data suggests, therefore, that software security activity use tends to be driven from each different job role, as much as centrally from an SSG team.

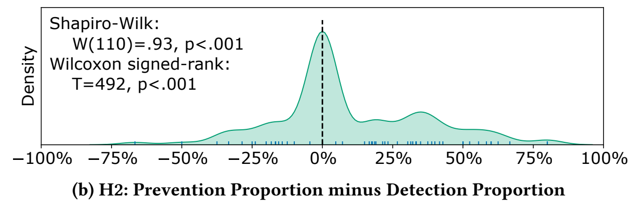
Considering changes in activity use (RQ1.2), the rejection of Hypothesis H1 suggests that there is little value to distinguishing Detection and Response activities with respect to adoption. The acceptance of Hypothesis H2 shows that Prevention activities tend to be adopted later, but we note also that this effect is not large.

Table 3 shows little clustering in the adoption of security activities. The first cluster reflects the novelty of cloud hosting and containerization. Neither the second nor third cluster seems explicable, suggesting that, aside from responding to new cloud requirements, the adoption of techniques together is relatively random.

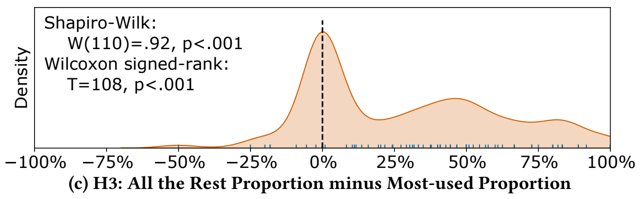
Table 4 shows that the two most commonly-adopted activities address the problem of insecure components: inventorying applications and identifying open source. The activities most often *dropped* show evidence of a 'shift left', the transfer of security responsibilities earlier in the development life cycle, to developers rather than



(a) H1: Detection Proportion minus Response Proportion



(b) H2: Prevention Proportion minus Detection Proportion



(c) H3: All the Rest Proportion minus Most-used Proportion

Figure 7: Differences between Proportions of Types of Activities Adopted. ***Positive differences represent later adoption of the first activity type.***

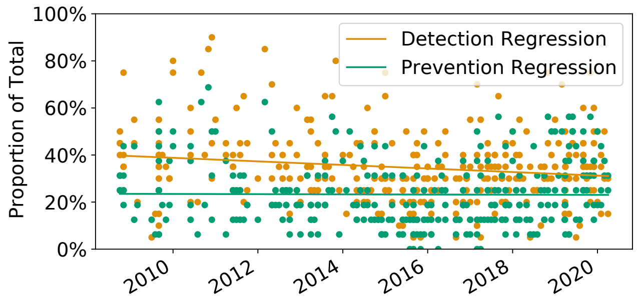


Figure 8: Simple Linear Regression on Activity Use

operations and security specialists: three of the five most often dropped (SE2.2, SR3.3, and CR1.5) are activities enforced by SSG on developers. As developers increasingly have direct responsibility for, and expertise at, security, such activities are becoming less appropriate. These observations seem to reflect industry trends (RQ2), rather than offering an insight into how companies are likely to adopt security activities in the future.

We conclude that the answer to RQ1 is:

***RQ1*: What have been the patterns of adoption and usage of software security activities by software development teams in security-focused companies?**

The 12 most-used activities in Figure 6 dominate activity usage, and tend to be adopted first. Prevention activities tend to be adopted slightly later than Detection and Response activities, and activities related to specific job roles tend to be used together.

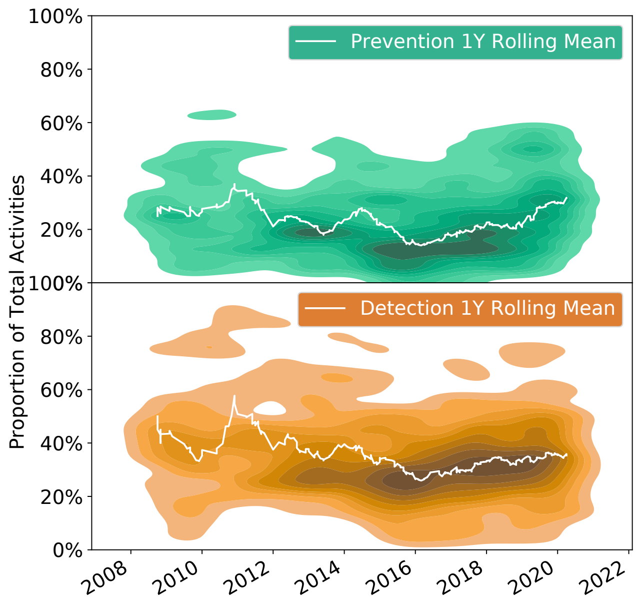


Figure 9: Changing Prevention & Detection Activities

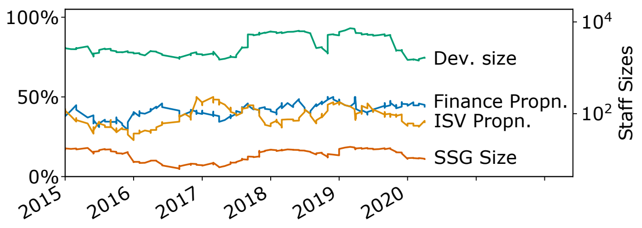


Figure 10: Changing Company Attributes over Time

6.2 Further Industry Trends

Figure 9 shows an interesting pattern of activity adoption. Prior to 2015, the number of security activities adopted by organizations varied widely year by year and demonstrated a general downward trend. We suggest that this phenomenon reflects that the early set BSIMM adopters come from a wide variety of organisations with inconsistent security focus though all were interested enough in security adoption to engage with the BSIMM team. Since 2015, activity adoption demonstrates an upward trend. As Figure 10 shows, the nature of the companies involved may have remained more consistent. Additionally, industry has a more consistent focus on regulation and must respond to the unceasing and increasing levels of cyber threat.

We, therefore, answer RQ2 as follows.

***RQ2*: What have been the trends in adoption of developer security activities industry-wide?**

Security activities adoption demonstrated little consistent pattern until 2015. Since then, industry companies have increased their use of security activities, especially Prevention activities. Looking at changes within companies in the last 5 years, we also find a focus on components, increased cloud-related security, and a reduction of the security professionals' policing role.

6.3 Implications

Based on the discussion in Section 6.1, we can reasonably deduce that the most-used activities are likely to be the most cost-effective and valuable for companies to adopt, and that they provide a basis for the later adoption of more advanced activities.

Our results also showed that Prevention activities are adopted later than Detection and Response ones. From the point of view of a company adopting them, the commercial justification for Response activities is clear: vulnerabilities have been detected so there is an economic justification for stopping similar problems in the future. Detection activities allow post-hoc justification: practitioners are motivated to discover and mitigate vulnerabilities that have been injected into their product. Prevention activities are the hardest to justify commercially due to their unknown future economic impact. It is only when a company has a number of Detection and Response activities to provide the cost-benefit figures that practitioners can start justifying Prevention activities. That observation may explain why Prevention activities tend to be adopted later, even if academic research [2, 21, 22] proves them more cost-effective.

Figure 9, however, also shows a much higher proportional increase of Prevention activities since 2015 than of Detection ones. This observation suggests that the changing industry environment now provides a commercial justification for Prevention activities that was absent before.

Returning to the original goal of this paper, to aid software and security engineers, software engineering educators, and software engineering researchers, we can make several observations from the answers to the research questions in Sections 6.1 and 6.2:

- Focusing on the 12 most-used activities is likely to give maximum impact for both practitioners, educators, and researchers.
- Commercial justification of activities is important. Successes in Response and Detection activities offer a way to commercially justify Prevention activities, though this has been gradually becoming less necessary since 2015.
- An effective way to encourage the adoption of more sophisticated activities is to work with specific roles around the development teams: inventory specialists, software architects, release control, DevOps developers, cloud configuration specialists, and penetration testers.
- Existing programs are likely to need to change to support the use of components, cloud-based computing, and developer-centered security.

The corresponding challenge for **software and security engineers** is, therefore, to find ways to commercially justify and adopt the 12 most-used activities, and to work with the roles listed above to adopt the more advanced activities. Options to help them include developer workshops [47] and improving security advocacy skills [14]; books are also available to support developers in adopting the more advanced security activities [3, 24].

Examples of future work for **software engineering researchers** include tools to support statistics gathering; studies of how particular groups relate to the development teams with respect to software security activity adoption; and exploration of the reasons for the synergy between the most-used activities. A good starting point might be a literature survey on developer-centered security [42].

For **software engineering educators**, the challenge is to update their training to provide a good grounding in the most-used activities and in the measurement of security effectiveness. They may also need to upgrade software security training to support the use of components, cloud-based computing, and developer-centered security. To support them, there is a good general resource in the the Cybersecurity Body of Knowledge [4].

6.4 Threats to Validity

There are several validity threats [9] to this study. Any conclusions must be considered with the following issues in mind:

Conclusion validity: can we trust the conclusions? While we can be reasonably sure that the statistical findings in Section 5.4 would be likely to be replicated in any similar analysis, the sensitivity of the data means that the calculations cannot be independently verified. In addition, the deductions from the clustering and about the increase in activities since 2015 are indicative rather than statistically proven.

Construct validity: how valid is the theory? The care and effort expended on making the assessments make it likely that the results recorded are reasonably accurate. However, though the categorization of activities into Prevention, Detection, and Response is justified by literature (Section 2.1) and the double coding and explanations in the text make it likely that the categorization is repeatable, allocation of activities to specific categories proved to need further clarification (Section 5.2). We conclude that the triad may have limitations in perfectly categorizing security activities.

External validity: how far can the conclusions be extended? Our calculations around later adoption cover only the companies with more than one assessment which may bias the sample. Similarly, all the data is from companies prepared to invest both money and considerable effort in the assessment process. It seems likely that the results will be typical of similar companies (medium to large, with an SSG), but we have no justification to extend conclusions to other kinds of organizations.

7 CONCLUSION AND FUTURE WORK

This paper explores the most comprehensive dataset of developer software security activity known to us. The longitudinal study of many large, security-focused companies identifies 12 most-used activities; that the more sophisticated activities tend to be adopted in cooperation with specific non-programmer roles; and a trend towards activities to handle components, cloud-based computing and developer-centered security.

Future work will include exploring this dataset for evidence of trends in assurance activities not carried out by developers, and possibly exploring new trends as later assessments are carried out

Section 6.3 offers implications for software and security engineers, for software engineering educators, and for software engineering researchers. Acting on these implications provides an objectively-justified means to maximize the software security impact of each of these roles, and a strong potential contribution to improve security outcomes in the future.

REFERENCES

[1] American Psychological Association. 2019. *Publication Manual of the American Psychological Association: The Official Guide to APA Style* (seventh ed ed.). https://apastyle.apa.org/products/publication-manual-7th-edition

[2] Adam Barth, Benjamin I. P. Rubinstein, Mukund Sundararajan, John C. Mitchell, Dawn Song, and Peter L. Bartlett. 2010. A Learning-Based Approach to Reactive Security. In *Financial Cryptography and Data Security*, Radu Sion (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 192–206.

[3] Laura Bell, Michael Brunton-Spall, Rich Smith, and Jim Bird. 2017. *Agile Application Security: Enabling Security in a Continuous Delivery Pipeline.* O'Reilly, Sebastopol, CA.

[4] Bristol University. [n. d.]. The CyBOK Project. ([n. d.]). https://www.cybok.org/

[5] CDC. September 23, 2019. Preventing, Detecting, and Responding to Epidemics: CDC's Achievements. (September 23, 2019). https://www.cdc.gov/globalhealth/security/ghsareport/2018/prevent-detect-respond.html

[6] Cisco. 2021. The 2021 Security Outcomes Study. (2021). https://www.cisco.com/c/en/us/products/security/security-outcomes-study.html

[7] CONSORT. 2010. Checklist of Information to Include When Reporting a Randomized Trial. (2010), 11–12 pages. http://www.consort-statement.org/consort-2010 [Online; accessed 2019-09-10].

[8] DoD. 2020. Cybersecurity Maturity Model Certification (CMMC). (2020). https://www.acq.osd.mil/cmmc/docs/CMMC_ModelMain_V1.02_20200318.pdf

[9] Robert Feldt and Ana Magazinius. 2010. Validity Threats in Empirical Software Engineering Research: An Initial Survey.. In *SEKE*. 374–379.

[10] Jason Firch and PurpleSec. 2021. 10 Cyber Security Trends You Can't Ignore in 2021. (2021). https://purplesec.us/cyber-security-trends-2021/

[11] Forbes. 2020. *Global 2000 - The World's Largest Public Companies 2020.* Technical Report. https://www.forbes.com/global2000/

[12] Gartner. 12 February 2020. The Urgency to Treat Cybersecurity as a Business Decision. ID G00466055 (12 February 2020).

[13] Kilem L Gwet. 2014. *Handbook of Inter-Rater Reliability: The Definitive Guide to Measuring the Extent of Agreement Among Raters.* Advanced Analytics LLC.

[14] Julie M Haney and Wayne G Lutters. 2017. Skills and Characteristics of Successful Cybersecurity Advocates. In *Workshop on Security Information Workers - SIW*. USENIX Association.

[15] Debra S Herrmann. 2002. *Using the Common Criteria for IT security evaluation*. CRC Press.

[16] John P A Ioannidis. 2005. Why Most Published Research Findings Are False. *PLOS Medicine* 2, 8 (2005), 0696–0701. https://doi.org/10.1371/journal.pmed.0020124

[17] ISO. 2011. ISO/IEC 27034 Application Security Standard. (2011). https://www.iso27001security.com/html/27034.html

[18] ISO/IEC. 2013. ISO/IEC 27001: Information Security Management Report. (2013). https://www.iso.org/isoiec-27001-information-security.html

[19] 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.). Springer International Publishing, Cham, 120–138.

[20] Thomas Kluyver, Benjamin Ragan-kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. 2016. Jupyter Notebooks: A Publishing Format for Reproducible Computational Workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, 87–90. https://doi.org/10.3233/978-1-61499-649-1-87

[21] Juhee Kwon and M. Eric Johnson. 2011. An Organizational Learning Perspective on Proactive vs. Reactive Investment in Information Security. In *Tenth Workshop on Economics of Information Security (WEIS 2011), Fairfax, VA, USA, June 14-15, 2011. (WEIS Workshop Proceedings)*.

[22] Juhee Kwon and M. Eric Johnson. 2014. Proactive Versus Reactive Security Investments in the Healthcare Sector. *MIS Quarterly* 38, 2 (2014), 451–A3. https://www.jstor.org/stable/26634934

[23] John Maguire and H. Miller. 2010. Web-Application Security: From Reactive to Proactive. *IT Professional* 12 (09 2010), 7 – 9. https://doi.org/10.1109/MITP.2010.117

[24] Gary McGraw. 2006. *Software Security: Building Security In*. Vol. 1. Addison-Wesley Professional.

[25] Gary McGraw and Brian Chess. 2009. The Building Security in Maturity Model (BSIMM). USENIX Association, Montreal, Quebec.

[26] Gary McGraw, Brian Chess, and Sammy Miques. 2011. *Building Security in Maturity Model 5*. Technical Report May. pp. 1–61 pages.

[27] Sammy Migues, John Steven, and Mike Ware. 2020. *BSIMM11 Report*. Technical Report. Synopsys. https://www.bsimm.com/download.html

[28] Geoffrey A Moore. 2009. *Crossing the Chasm: Marketing and Selling Disruptive Products to Mainstream Customers*. Harper Collins.

[29] Patrick Morrison. 2015. A Security Practices Evaluation Framework. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2 (ICSE '15)*. IEEE Press, 935–938.

[30] Patrick Morrison, Benjamin H. Smith, and Laurie Williams. 2017. Surveying Security Practice Adherence in Software Development. In *Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp (HoTSoS)*. Association for Computing Machinery, New York, NY, USA, 85–94. https://doi.org/10.1145/3055305.3055312

[31] Fionn Murtagh and Pedro Contreras. 2012. Algorithms for Hierarchical Clustering: An Overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2, 1 (2012), 86–97. https://doi.org/10.1002/widm.53

[32] NIST. 2020. NIST Special Publication 800-53, Revision 5, Security and Privacy Controls forInformation Systems and Organizations. (2020). https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf

[33] NIST. 2021. NIST Cybersecurity Framework. (2021). https://csrc.nist.gov/projects/risk-management/about-rmf

[34] NIST. 2021. Statistics Results. (2021). https://nvd.nist.gov/vuln/search/statistics

[35] SAFECode organization. 2021. SAFECode. (2021). https://safecode.org/

[36] OWASP. 2017. *Open Software Assurance Maturity Model (Open-SAMM)*. Technical Report. 1–96 pages. http://www.opensamm.org/

[37] OWASP. 2020. OWASP Application Security Verification Standard (ASVS). (2020). https://owasp.org/www-project-application-security-verification-standard/

[38] Tosin Daniel Oyetoyan, Martin Jaatun, and Daniela Cruzes. 2019. *Exploring Security in Software Architecture and Design*. Chapter Measuring Developers' Software Security Skills, Usage, and Training Needs, 260–286. https://doi.org/10.4018/978-1-5225-6313-6.ch011

[39] Deborah Rumsey. 2009. *Statistics II for Dummies*. Wiley, Indianapolis.

[40] James LaPiedra SANS Institute. 2002. The Information Security Process: Prevention, Detection and Response. (2002). https://www.giac.org/paper/gsec/501/information-security-process-prevention-detection-response/101197

[41] Jose M. Such, Antonios Gouglidis, William Knowles, Gaurav Misra, and Awais Rashid. 2016. Information Assurance Techniques: Perceived Cost Effectiveness. *Computers and Security* 60 (2016), 117–133. https://doi.org/10.1016/j.cose.2016.03.009

[42] Mohammad Tahaei and Kami Vaniea. 2019. A Survey on Developer-Centred Security. In *European Workshop on User-Centered Security - EuroUSec*. 14. https://doi.org/10.1109/EuroSPW.2019.00021

[43] Graham Upton and Ian Cook. 2014. *A Dictionary of Statistics* (3rd ed.). Oxford University Press. https://doi.org/10.1093/acref/9780199679188.001.0001

[44] Elaine Venson. 2020. The Effects of Required Security on Software Development Effort. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings (ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 166–169. https://doi.org/10.1145/3377812.3381393

[45] E. Venson, R. Alfayez, M. M. F. Gomes, R. M. C. Figueiredo, and B. Boehm. 2019. The Impact of Software Security Practices on Development Effort: An Initial Survey. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–12. https://doi.org/10.1109/ESEM.2019.8870153

[46] Elaine Venson, Xiaomeng Guo, Zidi Yan, and Barry Boehm. 2019. Costing Secure Software Development: A Systematic Mapping Study. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES '19)*. Association for Computing Machinery, New York, NY, USA, Article 9, 11 pages. https://doi.org/10.1145/3339252.3339263

[47] Charles Weir, Ingolf Becker, and Lynne Blair. 2021. A Passion for Security: Intervening to Help Software Developers. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE. https://eprints.lancs.ac.uk/id/eprint/151678/1/WeirSEIP2021Signed.pdf

[48] S. Wen. 2017. Software Security in Open Source Development: A Systematic Literature Review. In *2017 21st Conference of Open Innovations Association (FRUCT)*. 364–373. https://doi.org/10.23919/FRUCT.2017.8250205

[49] L. Williams, G. McGraw, and S. Migues. 2018. Engineering Security Vulnerability Prevention, Detection, and Response. *IEEE Software* 35, 5 (2018), 76–80. https://doi.org/10.1109/MS.2018.290110854