A Passion for Security: Intervening to Help Software Developers

Charles Weir
Security Lancaster
Lancaster University
United Kingdom
c.weir1@lancaster.ac.uk

Ingolf Becker
Security and Crime Science
University College London
United Kingdom
i.becker@ucl.ac.uk

Lynne Blair
Computing and Communications
Lancaster University
United Kingdom
l.blair@lancaster.ac.uk

***Abstract*—While the techniques to achieve secure, privacy-preserving software are now well understood, evidence shows that many software development teams do not use them: they lack the 'security maturity' to assess security needs and decide on appropriate tools and processes; and they lack the ability to negotiate with product management for the required resources. This paper describes a measuring approach to assess twelve aspects of this security maturity; its use to assess the impact of a lightweight package of workshops designed to increase security maturity; and a novel approach within that package to support developers in resource negotiation. Based on trials in eight organizations, involving over 80 developers, this paper demonstrates that (1) development teams can notably improve their security maturity even in the absence of security specialists; and (2) suitably guided, developers can find effective ways to promote security to product management. Empowering developers to make their own decisions and promote security in this way offers a powerful grassroots approach to improving the security of software worldwide.**

Keywords—Developer Centered Security; software security; software developer; intervention; Design Based Research

I. Introduction

Software security and privacy are now major issues: almost every day we hear that several more organisations' software systems have been compromised [26].

While there are many aspects to an organization's security and privacy, the design and implementation of the software used clearly has a significant impact on whether such breaches happen. Two industry trends contribute to this: the increasing use of microservices and Software as a Service (SaaS) components, and the DevOps movement both mean that security must be 'in the code' rather the responsibility of a separate operations team. So, it is vital that developers be effective at creating secure software.

Unfortunately, there is evidence that developers are not delivering this security. A recent report from Veracode concluded that *"more than 85 percent of all applications have at least one vulnerability in them; more than 13 percent of applications have at least one very high severity flaw"* [32]. A report from Microsoft found that 28% of Software as a Service applications were not supporting data encryption [23]. Clearly, industry practices are not yet sufficient to provide the software security and privacy we need. So, how can one support developers to deliver better security?

This research project addresses the objective of *defining a cost-effective intervention to support software development teams in creating secure products and services*. In earlier work the authors identified requirements and an approach using workshops to sensitize developers to the importance of security [36]. Based on this they derived a set of requirements for this package (Section III) and concluded the need for objective assessment of its impact.

The primary research question of this paper, therefore, is:

RQ 1 *What aspects of an intervention to a software development team are effective at improving security, and why?*

The paper describes the design of a security-improving intervention, its use in 8 different organizations, and the practical and theoretical conclusions. The analysis looked for and quantified improvements in 'assurance techniques': process improvements, understanding and skills that would generate better security in the longer term. The research makes the following contributions:

- An existence proof that a simple 'intervention package' structured as a facilitated series of workshops can improve the security of software developed by a team.
- Identification of the importance of representing security enhancements in terms of their business benefit, and the ability of developers to do so.

The rest of this paper is as follows. Section II discusses relevant past research; section III describes and explains the intervention package; section IV describes the research method; section V explores the outcomes from using the intervention with different groups; section VI discusses the results; and section VII provides a conclusion.

II. Background

Research related to interventions for secure software has taken a variety of approaches: ways to get developers to adopt process improvements; ways to get developers to adopt analysis tools; consultancy and training interventions; motivating developers to improve their processes; and motivating employees more generally to adopt secure practices. The following sections explore these approaches.

A. *Adoption of Security-Enhancing Activities*

One way to improve development security is to build a process around it using a 'Secure Development Lifecycle' (SDL), a prescriptive set of instructions to managers, developers and stakeholders on how to add security activities to the development process [39]. However, research suggests resistance from development teams to adopting a prescriptive methodology. For example Conradi and Dybå deduced in a survey that developers are skeptical about adopting the formal routines found in traditional quality systems [10].

Van der Linden et al. found from a task-based study and survey that developers tend to see only the activity of writing code to be security-relevant, suggesting a need for a stronger focus on the tasks and activities surrounding coding [20]. Caputo et al. concluded from three case studies a need for the alignment of security goals with business goals [8]. And an interview survey by Xie et al. suggests that developers make security errors from treating security as *"someone else's problem"* [42].

Such et al. defined a taxonomy of twenty assurance techniques from a survey of security specialists, finding wide variations in the perceived cost-effectiveness of each [30]. Weir et al. surveyed successful app developers, finding less than half using assurance techniques regularly [37].

This suggests a need for ways to encourage adoption. Indeed, recently Assal and Chiasson identified from a developer survey *"a need for new, lightweight best practices that take into account the realities and pressures of development"* [1].

B. *Encouraging the Adoption of Tools*

Witschey et al. concluded from an interview survey that more experienced and more inquisitive developers are more likely to adopt tools Key deterrents were the difficulty of trialling new tools and that developers are unlikely to notice a colleague using one [40]. Xiao et al. found from a similar survey that the main reasons for adoption were recommendation by trusted peers, including experts in discussion forums; or company policies mandating the use of such tools [41].

Bessey et al. describe the motivation and issues with adopting code checkers for large codebases; for example that a tool *"needs to deliver a true defect in its first three error messages"* to generate a sale [6].

C. *Consultancy and Training Interventions*

Türpe et al. explored the effect of a single penetration testing session and workshop on 37 members of a large geographically-dispersed project. The results were not encouraging; the main reason was that the workshop consultant highlighted problems without offering much in the way of solutions [31]. A study by Poller et al. followed an unsuccessful attempt *"to challenge and teach [the developers] about security issues of their product",* finding that pressure to add functionality meant that attention was not given to security issues, and that normal work procedures did not support security goals. The authors concluded that successful interventions would need *"to investigate the potential business value of security, thus making it a more tangible development goal"* [25].

Considering book-based interventions, Yskout et al. tested if 'security patterns' might be an effective intervention to improve secure development in teams of student software developers. The results suggested a benefit but were statistically inconclusive [43]. A recent book by Bell et al. provides practical support for developers and tool recommendations [5].

D. *Motivating Change in Development Teams*

Looking at ways to motivate change in development teams, Dybå concluded from a quantitative survey that organizational factors were as least as important as technical ones: actions need to be aligned with business goals; and that employees take responsibility for the changes [12]. Beecham et al. conducted a literature review of 92 papers on programmer motivation in 2008, concluding that professional programmers are motivated most by problem solving, by working to benefit others and by technical challenges [4]. Hall et al. framed these as intrinsic motivators, relating them to self-determination theory [17].

Lopez et al. concluded that to encourage developer teams in doing security there is a need to *"raise developers' security awareness"* [21], such as by using discussions about security [22].

E. *Conclusions*

This previous work suggests a need for lightweight, cost-effective, enhancements to development practices to improve security. In particularly, it suggests raising a development team's security awareness, and aligning their security goals with business goals.

III. The Intervention

The purpose of the project was *defining a cost-effective intervention to support software development teams in creating secure products and services*. Prior work by the authors identified the following requirements for such an intervention:

- Motivate developers to drive their own security improvements [36];
- Encourage developers to adopt a subset of key assurance techniques, specifically Threat Assessment, Configuration Review, Automated Static Analysis, Source Code Review, and Penetration Testing [36];
- Work without security specialists, since few teams have access to them [37];
- Support developers currently using few or no Assurance Techniques, since few use them [37]; and
- Work with teams, as a majority of developers do so [29]

The authors had expected such an intervention to take the form of a website, a book or video [38]; or possibly a code analysis tool [24]. In practice, we found that excellent implementations already exist of such interventions, but the need for improved security remains. Instead, we determined that facilitated workshops with the teams would offer a good approach, and that two Assurance Techniques are suitable for such workshop sessions:

- Incentivization Session, and
- Threat Assessment

Based on early trials of a workshop-based intervention [36] we concluded two further requirements:

- To make the intervention scalable to many teams, non-researchers must lead the intervention, and
- A need for training to help developers promote security mitigations to product management

A. *Implementing the Incentivization Session*

As an alternative to traditional fear-based security motivation, we wanted an Incentivization Session that would help developers engage with security better and lose their fear of it. We used a facilitated game, the 'Agile App Security Game' [35], in which participants work in groups as product managers, selecting security-enhancing product improvements with varying costs and learning whether their choices deter attacks.

B. *Implementing Threat Assessment*

The Threat Assessment workshop was challenging to implement. Though valuable, normal Threat Modelling approaches [28] require considerable knowledge of possible technical threats and preferably support from a professional with a detailed understanding of both the industry sector and current cyber threats to it; neither were available.

Instead we used a lightweight method, using an ideation session [14] in which a facilitator writes down unfiltered suggestions from the group of possible threat actors and outcomes (this was later replaced by an approach involving participants creating post-it notes with that content). Following that, participants used colored dots to vote, separately, on the most likely and the most impactful threats.

C. *Facilitator Training*

We trained one or two facilitators from each organisation, and they then managed the intervention.

To motivate the developers, we encouraged the use of 'self-actualization' language and approaches rather than commands and formal processes [15], avoiding terms like *"you must"* or *"it's essential that"*. Though the researchers participated in the workshops, they provided only occasionally comments and stories as participants.

We encouraged the facilitator to discuss, when opportunity arose, the other key techniques identified in the requirements in Section III: Configuration Review, Automated Static Analysis, Source Code Review, and Penetration Testing.

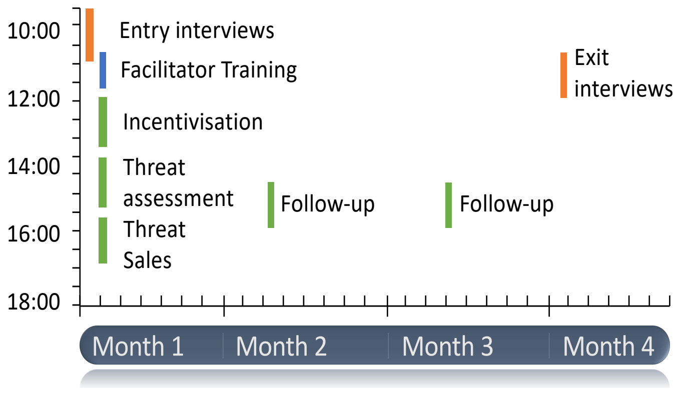

FIGURE 1: INTERVENTION TIMELINE

D. *Security Sales*

In this workshop participants split into groups, and each group addressed a different threat from the most important five or so identified in the Threat Assessment. We made the task approachable for developers by avoiding discussion of 'selling' or 'persuasion', two activities that do not form part of a normal development role. The instruction for the participants was to take one of the key threats and *"work out positive ways in which addressing that threat will benefit the organisation"*.

Each group discussed the threat they had chosen and wrote notes on a whiteboard or flipchart page. A representative from each group then presented their conclusions to the other participants.

E. *Intervention Schedule*

Figure 1 shows a typical schedule for the interventions, with different colors showing different sets of participants. The work with each group spanned 3-4 months, with only two days on site at the start and end. The involvement time was limited to four months in order to get the feedback from the exit interviews reasonably quickly. An online book, video, and materials [34] supported the package.

IV. EVALUATION METHODOLOGY

Trialling this intervention with professional developers required the involvement of the researchers both to train the facilitators and to support the workshops. We considered using Action Research [11], an accepted methodology used in many forms of academic social research including software engineering [27]. However, Action Research methodologies are designed to focus on the *clients*, following one or more clients through repeated research cycles. Here, by contrast, we were focused on the *intervention*, and the clients changed between research cycles. Accordingly, we used a different methodology, Design-Based Research (DBR). This supports different participants in each cycle of trials and focusses on designing an artifact.

DBR has its roots, and is used most, in education research. Its foundation lies in the 'Design Experiments' of Brown [7], and Collins [9] working with teachers as co-experimenters. It emphasizes the development of design theory in parallel with

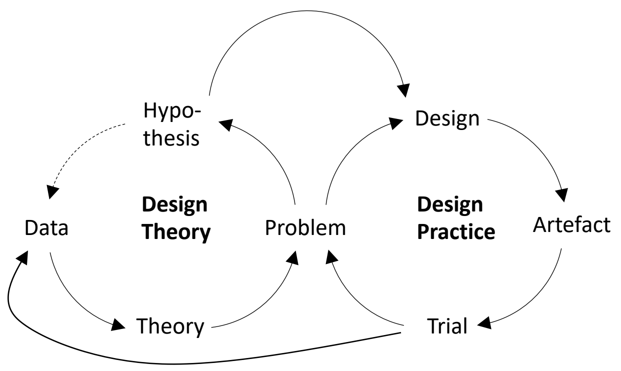

FIGURE 2: ACTIVITIES IN PRACTICAL DESIGN-BASED RESEARCH

the creation of teaching innovations. DBR is now an accepted research paradigm, used to develop improvements ranging from tools to curricula [18], with a recent guide book for practitioners [2].

The characteristics of Design-Based Research [33] are that it is: **pragmatic** in that DBR aims to solve current real-world problems, by creating and trialling interventions in parallel with the creation of theory; **grounded** in the practicalities of real-world trials in the *"buzzing, blooming confusion of real-life settings"* [3]; **interactive, iterative and flexible** with an iterative process involving multiple trials and experiments taking place as the theory develops; **integrative** in that DBR practitioners may integrate multiple methods, and vary these over time [33]; and **contextual** in that results depend on the context of the real-world trials.

A. *Practical Design-Based Research*

Figure 2, based on Ejersbo et al. [13], shows our understanding of the two parallel cycles of DBR research, creating theory and creating the artefact, with appropriate interactions between the cycles.

The practical aspects of carrying out DBR are defined by the 'integrative' nature of DBR: both design and assessment techniques must come from other research methodologies [33]. In this project we used the *techniques* of the Canonical Action Research method [11]—though not that method's overriding *paradigm*.

B. *Research Questions*

The DBR method requires separate research questions for the Design Practice cycle and the Design Theory cycle. Accordingly, we broke down the main research question RQ 1 into sub-questions. Our first Design Practice question was a measurement of the workshops' effectiveness:

RQ 2 *To what extent did the groups learn about and adopt different security-enhancing activities as a result of intervention?*

The second Practice question asks in what situations the intervention is likely to be most useful:

RQ 3 *How does the impact of the intervention vary with different company sizes, facilitation styles, security expertise, and kinds of participants?*

Turning to Design Theory questions, the suggestion from previous research, of the benefit of *"training to help developers promote security mitigations to product management"* was unproven, and so needed testing:

RQ 4 *Can having developers consider the positive benefits of security and privacy mitigations lead to security improvements in the development process?*

Colleagues had suggested that developers would require classroom training of techniques for risk assessment and 'sales'. For these workshops we had assumed this was not the case, though we had no a-priori justification for that:

RQ 5 *Can teams of developers produce both adequate risk and impact assessments and benefit analyses with minimal guidance?*

C. *Methodology Implementation*

We recruited groups in 8 different organizations, with whom we carried out the intervention. First, we interviewed a selection of the future participants to establish a baseline in terms of their current understanding, practice and plans. We then trained the facilitators, who led the intervention workshops, and supported the follow-up sessions. About three months from the start, we carried out 'Exit Interviews' with the same participants as before. Both Entry and Exit Interviews were semi-structured using open questions.

We recorded the audio of all the interviews and all the workshops, then transcribed the interviews. In an iterative process, two authors coded the interview transcripts using the tool NVivo. Differences in coding were discussed and resolved between us.

For the coding scheme, we identified a list of security-improving activities, including Assurance Techniques such as Pen Testing, and activities such as Contingency Planning, Training, and Stakeholder Negotiation; plus Adoption Levels for each (see Table 1). We were careful to distinguish changes due to the interventions from those due to other external factors; we did not code the latter.

From the coding, we determined the maximum Adoption Level coded for each Assurance Technique for each group, both 'before' and 'after' the intervention. Our 'Inter-Rater Reliability' calculations used Krippendorff's Alpha [16] to compare the *Adoption Levels calculated from the coding of each coder* rather than comparing the coding itself.

TABLE 1: ADOPTION LEVELS FOR EACH ASSURANCE TECHNIQUE

No mention	No reference to it in the interview
Aware	The team showed knowledge of it.
Planned	Existing plans to incorporate it.
Using	The team have used it.
Established	The team use it in each new project.

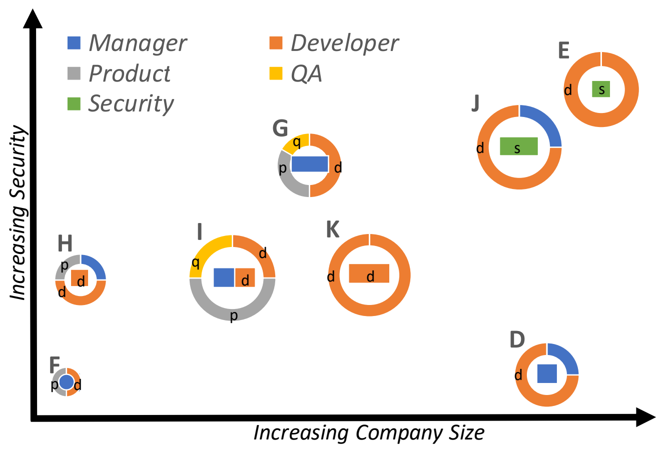

FIGURE 3: COMPOSITION OF THE PARTICIPATING GROUPS

This project was approved by the Lancaster University Faculty of Science and Technology Research Ethics committee.

V. RESULTS

Eight interventions were carried out with a total of 88 developers in eight different organizations, generating 21 hours of interview audio; and 47 hours of audio from training, workshop and follow-up sessions. The final code book contained 2859 references to 51 codes.

The Krippendorff's Alpha metric after the first round of coding was 0.18, indicating only slight agreement [16]. The main cause was that the interviewees had not been asked explicitly about their use of assurance techniques, in order to avoid bias in the responses. This allowed several kinds of discrepancy between the interpretations of the two coders.

Following a detailed discussion, the coders independently recoded the interviews; the resulting Krippendorff's Alpha was 0.46: moderate agreement. This is as good as can reasonably be expected, given the need for interpretation of the text by the coders. The remaining discrepancies were mainly due to omissions by one or other coder, so we used the union of both sets of codings for the later analysis.

A. *Summary of Participating Groups*

The participant groups were recruited opportunistically through industry contacts, university outreach and conference presentations. Groups are identified with a letter, starting with D (since three organizations had been involved in early trials); individual members we interviewed are identified using the team letter and a number: 'D1'. All of the developers interviewed were male, as were all managers and testers; three product managers were female: these numbers are consistent with industry norms [29].

Figure 3 visualizes the groups, plotting the organization sizes (from F's 20 to K's 15,000) against an estimate of their 'secure software capability maturity' [19] based on the groups' discussions during the workshops. Ring sizes show the number of participants (3 in F to 16 in K); ring centers show the facilitators; colors and lowercase letters show the job roles.

B. *Practical Results*

Table 2 shows the full list of Techniques derived from the coding. Horizontal lines divide them into three categories: Vulnerability Finding (top) to find specific vulnerabilities in created software; Process Improvements to create an environment to better support the creation of secure code; and Education (bottom) to teach participants and stakeholders.

Table 3 shows the anonymized details of the organizations involved, the groups we worked with, and the key outcomes from each intervention.

We encouraged facilitators to use their own facilitation styles, and saw a variety of such styles in the workshops. The Security Specialists, E1 J1 and J2 used a 'dominant' style, in which they and one or two others did most of the talking; in groups D, F, G and H there was a 'Listening' style, in which one facilitator controlled sessions, but encouraged wide participation. Finally, groups I and K had a 'Peer' style, setting only the workshop structure and letting the teams work independently

TABLE 2: ASSURANCE TECHNIQUES

Automated Pen. Testing	Using an automated tool to look for common vulnerabilities in a website or web service.
Automated Static Analysis	Using automated tools to look for common vulnerabilities in source or binary code.
Configuration Review	Choosing secure components and frameworks, and keeping them up to date
Code Review	Scheduled meetings or pair programming to analyse code for security defects
Penetration Testing	Having a security specialist look for vulnerabilities accessible via the web.
Threat Assessment	Design-level analysis of possible attackers, motives, and vulnerability locations.
Product Negotiation	Empowering product management to make security decisions.
Contingency Plan	The advance creation of a plan to handle security incidents.
Security Champion	Assigning a development team member, not usually a security expert, with a particular interest in security as the go-to person for security issues within the development team.
Standardisation	The creation of standard security configurations, ways of working, or 'Secure Development Lifecycles', plus auditing processes to validate these.
On-the-job Training	Mentoring or informal workshops, used regularly with the development team
Further Workshops	Using the entire package with other teams, the same team in a new project, or new members.

By assigning ordinal ratings to the engagement levels as shown in Table 1, we calculated an indication of the 'Impact' of the intervention—the extent to which the intervention affected the group's use of the technique. Of course, this 'Impact' calculation is merely an indication: a two-unit Impact (change in engagement) might be from No Mention to Planned, or from Planned to Established; these changes are not semantically equivalent.

Figure 4 thus provides an answer to RQ 2 *"To what extent did the groups learn about and adopt different security-enhancing activities as a result of intervention?"* The size of

TABLE 3: ORGANIZATIONS, GROUPS AND KEY OUTCOMES

	Organisation	Team	Threat Sales Outcomes	Main Outcomes
D	A project team within a university, funded by a government grant to promote business innovation by developing proof of concept (PoC) applications.	Aware of the importance of software security but had little practical knowledge; worked on several different projects at a time.	Identified that the threat and risk assessment itself was a valuable asset to their clients	Identified need to support their clients in identifying security issues when the clients came to implement applications based on the PoCs.
E	A government department delivering software for sensitive government applications. The group worked on a high-confidentiality product.	Less experienced than average for the industry, though the session leader E1 is an experienced security specialist	Realized that while every security enhancement was essential, the ordering of their implementation could be altered to suit the client.	Improved understanding in the team of the importance of Threat Assessment and Product Negotiation.
F	A small surveying company delivering a Geographical Information System product and related services.	A previous developer had implemented some security aspects; the current team had little knowledge.	'Lined up' security improvements to be incorporated in the enhancements when new clients wanted them	Fixed several security issues in customer handling and created a list of issues which were later used as the basis for a new customer engagement to fix the following year.
G	A web applications developer delivering a wide variety of applications for clients.	The two leads G1 and G2 were expert in software security but were expected to provide costly security enhancements 'free'.	Found an impressively simple way to discuss security cost-benefit with a client: gold level hosting, silver and bronze security.	Incorporated new way to discuss security with customers. G1 later expanded it to five options to include other aspects of security.
H	A small company selling a range of Internet of Things (IoT) devices and their associated infrastructure.	The group justifiably consider themselves good at software security;	Identified that their security story was a major Unique Selling Point against competitors.	Following the workshop, they plan further training.
I	A well-established company providing the infrastructure for a commodity trading. Planning move from perimeter security to cloud-based services.	The company has considerable internal expertise in security, especially I2. However, the developers were less experienced.	Subsequently included security requirements in discussions with new clients	Following the initial workshops, they re-ran both Threat Assessment and Threat Sales workshops to gain a more complete idea of the threats and impact on customers. They also ran workshops with further development teams.
J	A well-established large company providing web interfaces for retailers. The particular group involved had the responsibility of creating tools and services to support deployment	The group was a team of about a dozen creating deployment tools and included security specialists J1 and J2.	Devised several functionality and process improvements for their (internal) customers	While some of the participants may well have learned from the workshops there were no detected improvements in understanding and technique use.
K	A well-established company with a few hundred employees creating tools for developers.	The group has a strong emphasis on agile development processes, and team interaction. All the participants were developers.	Each of four subgroups delivered a convincing sales pitch for a security improvement.	Analysis showed an increase in awareness in the teams of some of the assurance techniques.

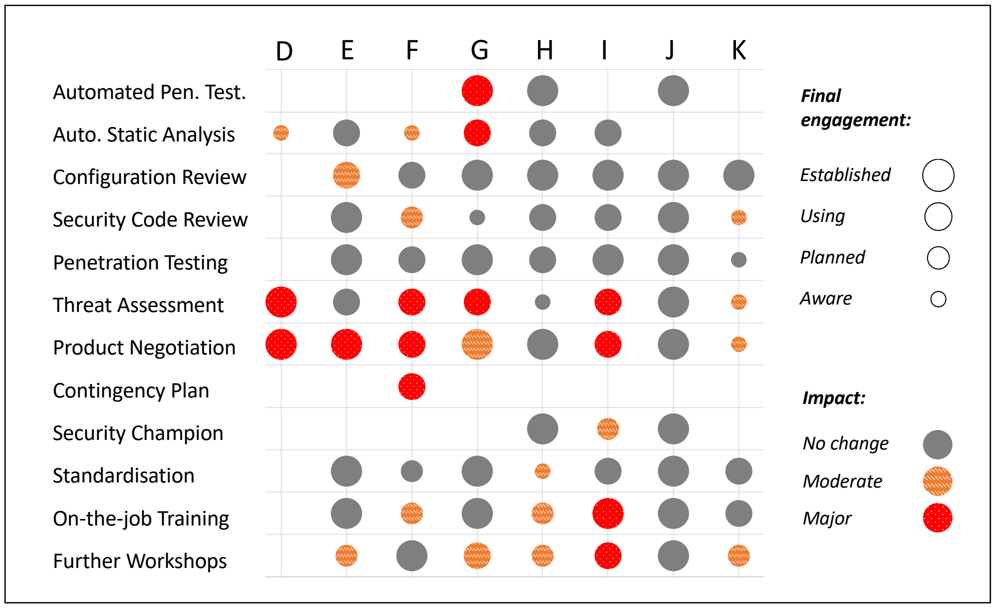

FIGURE 4: CHANGES IN ASSURANCE TECHNIQUE USAGE FOR ALL GROUPS

each bubble indicates the final engagement level after the intervention; the color shows the change attributed to the intervention: amber for a change of 1 to 2 levels; red for 3 to 4 levels.

As the figure shows, the use of Threat Assessment and Product Negotiation had dramatically improved in a majority of groups; use of Penetration Testing and Use of Checklists were not affected at all. Group J showed little change as a result of the intervention; all the others did see at least some changes. Groups I, J chose to carry out further workshops independently from the researchers, and D, E, F, G and I all showed major improvements in their use of Threat Assurance and Product Negotiation.

C. *Technique Adoption by Different Categorizations of Group*

Turning to RQ 3 *"How does the impact of the intervention vary with different company sizes, facilitation styles, security expertise, and kinds of participants?"*, Table 4 calculates average impact values for different categorizations of the groups. The deeper shadings show the higher values in each categorization; the red-green colors distinguish different categorizations. The shading in the first column delineates the types of assurance technique (Vulnerability Finding, Process Improvements, and Training), while the figures on the bottom line show the average increment over all assurance techniques for each category.

D. *Positive Benefits of Security and Privacy*

To address the theory-based research question RQ 4 *"Can having developers consider the positive benefits of security and privacy mitigations lead to security improvements in the development process?"*, we looked for cases where the Threat Sales activity lead to 'security improvements in the development process'.

Group D identified in the Threat Sales discussion that the threat and risk assessment itself was a valuable asset to their clients as part of their proof-of-concept developments. They now incorporate a security discussion in their 'handover document' for every project they do:

> *Now, after the workshop I think it was, we redesigned our handover template, which is where we now have a specific section for security [in every release] (D4)*

Group F realized that they could 'line up' security improvements to be incorporated in the enhancements when new clients wanted them:

> *Yes, .. we have picked up some new contracts, and ... they will require us to implement pretty much everything that we had listed... (F1)*

Group G identified the 'Gold, Silver, Bronze' approach to selling security enhancement costs to their clients.

> *To make that process a lot simpler for our sales team, [G1] did a lot of the leg work and setting up a Gold, Silver and Bronze package to say "right, answer these 10 questions", and then ... this is the package that you need'. (G6)*

Group I subsequently included security requirements in discussions with new clients:

> *So, we are giving the Product Owners some more insight into why you would do this stuff, and where the value is. (I1)*

While we do not have evidence that the Threat Sales activity generated value in every case, the experience of Groups D and F, in particular, indicate that the activity of getting developers to consider the positive benefits of security

TABLE 4: IMPACT AVERAGED BY GROUP ATTRIBUTES

	All	Org. size			Facilitation			Sec. maturity			Prod.mgr		Facilitator		
		Large	Med.	Small	Dom.	Listen.	Peer	High	Med	Low	Yes	No	Mgr.	Sec.	Dev.
Count in each category	8	3	3	2	2	4	2	2	4	2	4	4	4	2	2
Automated Pen Testing	0.5		1.3			1.			1.		1.		1.		
Auto. Static Analysis	0.6	0.3	1.	0.5		1.3			0.8	1.	1.	0.3	1.3		
Configuration Review	0.1	0.3			0.5			0.5				0.3		0.5	
Code Review	0.4		0.3	1.		0.5	0.5		0.3	1.	0.5	0.3	0.5		0.5
Penetration Testing															
Threat Assessment	1.6	1.	2.3	1.5		2.3	2.		1.8	3.	2.3	1.	3.		0.5
Product Negotiation	2.1	2.7	2.	1.5	2.	2.3	2.	2.	1.5	3.5	2.	2.3	3.	2.	0.5
Contingency Plan	0.4			1.5		0.8				1.5	0.8		0.8		
Security Champion	0.3		0.7				1.		0.5		0.5		0.5		
Standardisation	0.1			0.5		0.3			0.3		0.3				0.5
On-the-job Training	1.		1.3	2.		1.	2.		1.5	1.	2.		1.5		1.
Further Workshops	1.3	0.3	2.3	1.	0.5	1.	2.5	0.5	2.3		1.8	0.8	1.3	0.5	2.
Average over all	0.7	0.4	0.9	0.8	0.3	0.9	0.8	0.3	0.8	0.9	1.	0.4	1.1	0.3	0.4

can help get resources allocated to security improvements. We conclude, therefore, that the answer to RQ 4 is yes.

E. *Skills Not Associated with Developers*

Considering the second theory-based research question RQ 5 "*Can teams of developers produce both adequate risk and impact assessments and benefit analyses with minimal guidance?*", we found that, surprisingly, none of the teams had any trouble with **risk assessment**. Even Group D, who are producing proof of concept apps for companies and are not therefore domain experts for their products, had little difficulty:

> *We've identified huge risks that they need to consider before they ever get anywhere near an actual working product. (Participant, Group D)*

Team E learned and took away the prioritization process:

> *We had a follow-on session afterwards where we took everything away, ... and sat down and thought "what do we need to do next". (E3)*

For Group F, F1 produced a table of risks and impacts based on their discussion. Group G had no problem with risk assessments, since G1 and G2 were familiar with the likelihood of attacks on the websites they managed. Group H simply had their most expert members (H1, H3) identify the most likely threats by placing asterisks on the flipchart. Group I did similar. Group J had J1 and J2 (facilitators and also security experts) do the assessment. Group K successfully used a post-it approach, with very little facilitation, for the risk assessment; with separate dot-voting to identify the most likely and the most impactful threats.

In summary, all the groups found effective ways to assign risk to each of the threats they identified.

The **benefit analysis** outcomes from Threat Sales workshop are shown in column 4 of Table 3. They were also surprisingly satisfactory, given the lack of specific training given the group.

It seems reasonable to conclude that the developers in the groups had the necessary skills and insights required, and thus that the answer to RQ 5 is affirmative.

VI. DISCUSSION

As Section IV explains, Design-Based Research (DBR) has been used mostly in the field of education research. While the creation of an intervention in the field of Developer-centred Security is arguably a form of education, we are not aware of other researchers using DBR in this field. In this research, as Section V shows, DBR has provided an effective basis for trialling, evaluating, and deducing theory from the use of an intervention.

A. *Outcomes from Workshops*

The workshops concentrated on two aspects of security: using Threat Assessment to help participants focus their security effort on the appropriate threats; and using Threat Sales to present security requirements as positive opportunities to product management. It was therefore encouraging that the results in Figure 4 show that all the groups completed the intervention with an understanding of Threat Assessment and Stakeholder Negotiation, and, in a large majority of cases, incorporated those techniques into their ways of working. For half the groups involved this represented a large improvement over their previous practice.

Given the purpose of the new intervention package was to encourage others to use the package and lead sessions, it was encouraging that two companies did so; it was disappointing that a larger number had not got around to it after several months, even if they expressed intentions of doing so.

B. *Variation of Results with Different Situations*

We highlight some points of interest from the categorizations of impact in Table 4:

The intervention was adopted most by medium sized companies: such companies will have latent security expertise but no formal security function, making this intervention particularly useful for them.

Sessions facilitated by managers appear more effective than those facilitated by developers or security specialists: This may reflect better training in facilitation-related skills given to managers; it may also reflect greater power amongst managers to introduce new techniques.

The presence or absence of a product manager in the group had negligible effect on the adoption of Stakeholder Negotiation: This was a surprise. The author had expected a product manager would encourage emphasis and therefore improvements in this, but the results do not show that effect. The presence of a product manager did, however, encourage the incorporation of other assurance techniques.

Peer-based learning appears as effective as more active forms of facilitation: This offers the possibility of modifying the intervention workshops to be entirely peer-to-peer learning; making fewer demands on future facilitators.

Of course, given the sample size of 8 groups, these results are merely indicative and not statistically valid as indicators.

C. *Skills Not Associated with Developers*

Section V.E's answer to RQ 5 was that, surprisingly, developers found it easy both to assess the impact and likelihood of successful threat activities; and to think up ways of 'selling' security improvements to Product Management.

While we have no way of validating their results, we believe that their assessments will be sufficient for the purpose:

- With risk assessment, the consequence of getting a risk assessment wrong is much less than the consequence of not doing it at all.
- With sales, where Product Managers were present, they engaged very well with the process and found it valuable. This suggests that others may also find the results useful.

We conclude that there is no need in future interventions to provide more sophisticated training in either risk assessment or sales; most teams will be able to carry out both workshops without it.

D. *Threats to Validity*

We have considered the following types of validity:

Internal Validity: *Were the measured improvements caused by the workshops?* Since the majority of improvements were in areas directly addressed by the workshops (Figure 4), it is reasonable to assume a causal relationship here.

Conclusion Validity: *Can we derive convincing theory from the results?* The Krippendorff's Alpha of 0.46 indicates only moderate agreement between developers, suggesting that the measurements depend more than we would like on the skill and bias of the coders: it was difficult to identify initial adoption of Assurance Techniques since the Entry interviews did not ask explicitly about techniques. The sample size of 8 workshops is insufficient for a test of statistical significance; therefore, the results here should be taken as an indication rather than definite proof of the workshops' effectiveness.

Construct Validity: *Does the experimental model reflect reality?* The research measured the self-reported activities and knowledge; clearly it would be better to measure the resulting process or the security of the artefacts produced. Unfortunately, the latter is practically impossible in engagements of this kind, so this remains a limitation of this kind of study. A further limitation is the granularity of the adoption measurements used. Automatic Static Analysis, for example, could range from a simple pattern-based solution to a complex tool integrated into the CI system. Future studies might differentiate types of adoption.

External Validity: *How far can we generalize the results?* Participant companies were self-selected; the variety of the teams involved suggests that the workshops will work in many situations, but this research provides an indication rather than proof.

E. *Future Work*

The package used in these trials has a practical limitation: it requires time input from a researcher to train the facilitators. This severely restricts its scalability to a wider audience of development teams, and hence the academic impact it can have.

However, the success of the workshops as peer-to-peer exercises, where the facilitator only provided instructions, opens the possibility of a new version of the intervention that needs no direct training, and therefore can scale without limit.

VII. Conclusions

Recall the research question for this work: RQ 1 "*What aspects of an intervention to a software development team are effective at improving security, and why?*"

The trials showed that the intervention led to improvements in security process or understanding with all the groups who used it except the most security expert one. All three workshops were effective at helping improving security; developers proved adept even at risk assessment and creating positive representations of security improvements (Section VI.C).

The intervention had most impact where the workshops were facilitated by managers (Section VI.B), and were adopted most by medium sized companies; those that will have latent security expertise but no formal security function.

The findings from this project promise a new version of the package that can scale without limit and pave the way to the creation and trial of such a new package (Section VI.E). The lead author is currently working on a project to do exactly that.

VIII. Bibliography

[1] Assal, H. and Chiasson, S. Security in the Software Development Lifecycle. *Symposium on Usable Privacy and Security - SOUPS*, USENIX Association (2018), 281–296.

[2] Bakker, A. *Design Research in Education: A Practical Guide for Early Career Researchers*. Routledge, Abingdon, 2018.

[3] Barab, S. and Squire, K. Design-Based Research: Putting a Stake in the Ground. *Journal of the Learning Sciences 13*, (2004).

[4] Beecham, S., Baddoo, N., and Hall, T. Motivation in Software Engineering: A Systematic Literature Review. *Information and Software Technology 50*, 9 (2008), 860–878.

[5] Bell, L., Brunton-Spall, M., Smith, R., and Bird, J. *Agile Application Security: Enabling Security in a Continuous Delivery Pipeline*. O'Reilly, Sebastopol, CA, 2017.

[6] Bessey, A., Engler, D., Block, K., et al. A Few Billion Lines of Code Later. *Communications of the ACM 53*, 2 (2010), 66–75.

[7] Brown, A.L. Design Experiments : Theoretical and Methodological Challenges in Creating Complex Interventions in Classroom Settings. *Journal of the Learning Sciences 2*, 2 (1992), 141–178.

[8] Caputo, D.D., Pfleeger, S.L., Sasse, M.A., Ammann, P., Offutt, J., and Deng, L. Barriers to Usable Security? Three Organizational Case Studies. *IEEE Security and Privacy 14*, 5 (2016), 22–32.

[9] Collins, A. Toward a Design Science of Education. In *New Directions in Educational Technology*. Springer, 1992, 15–22.

[10] Conradi, R. and Dybå, T. An Empirical Study on the Utility of Formal Routines to Transfer Knowledge and Experience. *ACM SIGSOFT Software Engineering Notes 26*, 5 (2001), 268–276.

[11] Davison, R.M., Martinsons, M.G., and Kock, N. Principles of Canonical Action Research. *Information Systems Journal 14*, 1 (2004), 65–86.

[12] Dybå, T. An Empirical Investigation of the Key Factors for Success in Software Process Improvement. *IEEE Transactions on Software Engineering 31*, 5 (2005), 410–424.

[13] Ejersbo, L.R., Engelhardt, R., Frølunde, L., Hanghøj, T., Magnussen, R., and Misfeldt, M. Balancing Product Design and Theoretical Insights. In *The Handbook of Design Research Methods in Education*. Routledge, 2008, 149–164.

[14] Fisher, R., Ury, W.L., and Patton, B. *Getting to Yes: Negotiating Agreement Without Giving In*. Penguin, 2011.

[15] Gagné, M. and Deci, E.L. Self-Determination Theory and Work Motivation. *Journal of Organizational Behavior 26*, 4 (2005), 331–362.

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

[17] Hall, T., Sharp, H., Beecham, S., Baddoo, N., and Robinson, H. What Do We Know about Developer Motivation? *IEEE Software 25*, 4 (2008), 92–94.

[18] Hoadley, C., Baumgartner, E., Bell, P., et al. Design-Based Research: An Emerging Paradigm for Educational Inquiry. *Educational Researcher 32*, 1 (2002), 5–8.

[19] ISO/IEC. 21827:2008 - Systems Security Engineering - Capability Maturity Model. 2008, 144.

[20] van der Linden, D., Anthonysamy, P., Nuseibeh, B., et al. Schrödinger's Security: Opening the Box on App Developers' Security Rationale. *International Conference on Software Engineering - ICSE*, IEEE (2020).

[21] Lopez, T., Sharp, H., Tun, T., Bandara, A., Levine, M., and Nuseibeh, B. Hopefully We Are Mostly Secure: Views on Secure Code in Professional Practice. *Workshop on Cooperative and Human Aspects of Software Engineering - CHASE*, IEEE (2019), 61–68.

[22] Lopez, T., Sharp, H., Tun, T., Bandara, A., Levine, M., and Nuseibeh, B. Talking about Security with Professional Developers. *Workshop on Conducting Empirical Studies in Industry - CESSER-IP*, IEEE Computer Society (2019).

[23] Microsoft. Microsoft Security Intelligence Report, Volume 23. 2018. https://info.microsoft.com/rs/157-gqe-382/images/en-us_cntnt-ebook-sir-volume-23_march2018.pdf.

[24] Nguyen, D.C., Wermke, D., Backes, M., Weir, C., and Fahl, S. A Stitch in Time: Supporting Android Developers in Writing Secure Code. *Conference on Computer and Communications Security - CCS*, ACM (2017).

[25] Poller, A., Kocksch, L., Türpe, S., Epp, F.A., and Kinder-Kurlanda, K. Can Security Become a Routine? A Study of Organizational Change in an Agile Software Development Group. *Conference on Computer Supported Cooperative Work - CSCW*, ACM (2017), 2489–2503.

[26] RiskBased Security. Mid Year Data Breach Report. 2019, 1–14. https://pages.riskbasedsecurity.com/hubfs/Reports/2019/2019 MidYear Data Breach QuickView Report.pdf.

[27] Santos, P. and Travassos, G. Action Research Use in Software Engineering: An Initial Survey. *Symposium on Empirical Software Engineering and Measurement - ESEM*, IEEE (2009), 414–417.

[28] Shostack, A. *Threat Modeling: Designing for Security*. John Wiley & Sons, 2014.

[29] Stack Overflow. Annual Developer Survey 2016. https://insights.stackoverflow.com/survey/2016.

[30] Such, J.M., Gouglidis, A., Knowles, W., Misra, G., and Rashid, A. Information Assurance Techniques: Perceived Cost Effectiveness. *Computers and Security 60*, (2016), 117–133.

[31] Türpe, S., Kocksch, L., and Poller, A. Penetration Tests a Turning Point in Security Practices? Organizational Challenges and Implications in a Software Development Team. *Workshop on Security Information Workers - SIW*, USENIX Association (2016).

[32] Veracode. State of Software Security Report Volume 9. 2018. https://info.veracode.com/report-state-of-software-security-volume-9.html.

[33] Wang, F. and Hannafin, M.J. Design-Based Research and Technology-Enhanced Learning Environments. *Educational Technology Research and Development 53*, 4 (2005), 5–23.

[34] Weir, Charles; Knight, Jack; Ford, N. Developer Security Essentials. https://www.securedevelopment.org.

[35] Weir, C. The Agile App Security Game: Leader's Instructions. 2018. https://www.securedevelopment.org/resources/agile-security-game/.

[36] Weir, C., Becker, I., Noble, J., et al. Interventions for Long-Term Software Security: Creating a Lightweight Program of Assurance Techniques for Developers. *Software - Practice and Experience*, October (2019), 275–298.

[37] Weir, C., Hermann, B., and Fahl, S. From Needs to Actions to Secure Apps? The Effect of Requirements and Developer Practices on App Security. *29th USENIX Security Symposium (USENIX Security 20)*, (2020).

[38] Weir, C., Rashid, A., and Noble, J. Reaching the Masses: A New Subdiscipline of App Programmer Education. *Symposium on the Foundations of Software Engineering Proceedings: Visions and Reflections - FSE*, ACM (2016).

[39] De Win, B., Scandariato, R., Buyens, K., Grégoire, J., and Joosen, W. On the Secure Software Development Process: CLASP, SDL and Touchpoints Compared. *Information and Software Technology 51*, 7 (2009), 1152–1171.

[40] Witschey, J., Xiao, S., and Murphy-Hill, E. Technical and Personal Factors Influencing Developers' Adoption of Security Tools. *Workshop on Security Information Workers - SIW*, (2014), 23–26.

[41] Xiao, S., Witschey, J., and Murphy-Hill, E. Social Influences on Secure Development Tool Adoption: Why Security Tools Spread. *Conference on Computer Supported Cooperative Work - CSCW*, ACM (2014), 1095–1106.

[42] Xie, J., Lipford, H.R., and Chu, B. Why Do Programmers Make Security Errors? *IEEE Symposium on Visual Languages and Human Centric Computing*, (2011), 161–164.

[43] Yskout, K., Scandariato, R., and Joosen, W. Do Security Patterns Really Help Designers? *International Conference on Software Engineering - ICSE*, IEEE (2015), 292–302.