Light-touch Interventions to Improve Software Development Security

Charles Weir, Lynne Blair
Security Lancaster
Lancaster University
United Kingdom
{c.weir1, l.blair} @lancaster.ac.uk

Ingolf Becker, M. Angela Sasse
Department of Computer Science
University College London
United Kingdom
{i.becker, a.sasse} @ucl.ac.uk

James Noble
Engineering and Computer Science,
Victoria University of Wellington,
New Zealand
kjx @ecs.vuw.ac.nz

***Abstract*— Many software developers still have little interest in software security. To change this, we need 'interventions' to development teams to motivate and help them towards security improvement. An intervention costing less than two days' effort from a facilitator plus half a day of team effort can significantly improve that team's software security. This case study describes how this approach was used with one commercial team, and identifies its impact using Participative Action Research. With suitable improvements, the approach has the potential to help many other development teams.**

Keywords— Developer centered security; case study; software security; software developer; intervention; action research

I. INTRODUCTION

Software security and privacy are now major issues: the cost and potential threat to us all is increasing dramatically [1]. With over 100 billion lines of code created a year [1], the effectiveness of developers at creating secure software is vital[1].

Unfortunately, many if not most developers consider software security to be 'not their problem' [2]. Developers may expect security to be handled by a different team; consider it too expensive to incorporate without a significant drive from product management; or simply not know where to start.

In prior work [3], the authors identified through interviews with leading software security experts, a powerful 'cocktail' of eight, mostly well-known, techniques (including a 'top five'), to encourage and support software development teams to deliver secure code. This combination of techniques has the potential to improve secure software development.

However, prior work has established only the potential; for the research to have impact requires us to find ways to disseminate knowledge of this 'cocktail' to some of the hundreds of thousands of programmers who could benefit. In a recent paper [18] we suggested a variety of approaches to this problem; in this research we investigated one approach in detail: having a consultant lead the introduction of the techniques using a package of lightweight consultancy 'interventions'.

This paper describes four aspects of those interventions:

- The design of the intervention package and methodology used;
- The background of the software development project where it was used;
- The effect of the interventions on both developers and product security;
- and possible improvements we might make in future.

The contributions of this work are:

1. Research on using managerial and process-based approaches to developer security rather than on delivering better tools,
2. A low-cost but effective intervention method to help a team improve their development security, and
3. A detailed example of its implementation and the consequences.

The rest of the paper is as follows. **Section II** establishes the existing literature on the subject. **Section III** explains the research and analysis methods in detail. **Section IV** discusses the context and background to the development project. **Section V** discusses the results obtained. And **Section VI** compares the approach with existing practice and identifies future work.

II. EXISTING WORK

This section examines existing academic literature and related publications on the subject of helping and encouraging developers to improve their software security. We explore several areas in turn, examining the existing research available in each, and moving from the relatively mechanical to more sociological approaches.

A. *Encouraging the Adoption of Tools*

Much research has gone into the creation of tools to improve security, such as code analyzers [4]–[7]. Rather less, however has gone into getting them used. A survey by Johnson et al. analyzed 'Why Don't Software Developers Use Static Analysis Tools to Find Bugs' [8] and produced a set of recommendations for tool functionality; in particular the ability to avoid repeated false positives and support for 'quick fixes'. Jordan et al. [9]

[1] Throughout this paper we use 'secure' and 'security' to refer to privacy aspects of software development as well as security ones.

explored using emails to encourage developers to upgrade to more secure versions of components, and suggested a variety of other possible interventions, but provided no evidence of success.

Some studies of tool adoption have based their theory on the seminal work on getting new ideas adopted, Rogers' book 'Diffusion of Innovations' (DoI) [10]. Based on extensive research and examples, much of this work describes the adoption process and reasons for adoption or non-adoption, providing a language with which to describe the process. Only one chapter, 'The Change Agent', describes – though at an abstract level – how individuals and organizations may promote and stabilize change.

Witschey et al. [11] suggested that adoption of tools could be modelled by DoI theory [10], and used a survey of 40 developers recruited opportunistically to explore the model. They concluded that more experienced, and more inquisitive, developers are more likely to adopt tools, and that key deterrents were difficult of trialling new tools, and their invisibility – that developers are unlikely to notice a colleague using one.

Xiao et al. [11] reported a DoI-based study, interviewing 40 professional developers to explore the social factors that led to security tool adoption. They found the main reason for adoption was recommendation by trusted peers, including high-rated experts in discussion forums. Interestingly company policies mandating the use of such tools were very effective; all 13 who had security tools mandated did use them.

Research by Dodier-Lazaro et al. [12] surveyed expert users about their tool installation choices, finding that they were unable to sacrifice consistency and flexibility for security benefits. Whilst an article by Bessey et al. [13] describes the experience of Coverity in building and marketing a source code checking tool for large codebases. They observed that the tool needs to deliver a true defect in its first three error messages to generate a sale.

Away from the domain of software security, others have investigated the adoption of development tools in general software engineering. Murphy-Hill et al. [14] explored how developers learned of new tools using 18 interviews and a diary study of 76 programmers. They concluded that discovery required peer interactions – especially seeing the tools being used by colleagues – and that these were surprisingly infrequent.

B. *Encouraging the Adoption of Security-Enhancing Activities*

There has been rather less work on encouraging developers to use non-tool techniques. Prior to 2010, the main way of improving software security was seen to be the 'Secure Development Lifecycle' (SDL), a prescriptive set of instructions to managers, developers, stakeholders and testers [15]. The assumption appears to have been that developers, if instructed to use an SDL, would do so wholeheartedly – which did not happen in practice. Indeed Xiao's 2014 survey of 40 developers [11], found only 2 using them, and none of the three major SDLs appears to have been developed much since 2010.

On the adoption of specific techniques, only code reviews have received any academic attention. However, research on their effectiveness has been limited to general software improvement, rather than specifically as a security technique: Baum et al. [16], for example, reviewed a variety of earlier work, and interviewed 24 professionals in 19 German companies. They conclude that cultural issues, rather than practical ones, determined whether code reviews were used, and that reviews were best embedded in the development process from the beginning of a project.

C. *Consultancy and Training Interventions*

Several research teams have explored the impact of training and external involvement on teams' delivery of secure software.

A recent paper by Poller et al. [17] describes an ethnographic study over 13 months of a depressingly unsuccessful attempt to improve security practices long term in an agile development team of about 15 people. The study investigated the effect of security consultants whose task *'was not to advise the product group on how to change their organizational routines, but to challenge and teach them about security issues of their product'.* This proved insufficient, for two reasons. First, pressure to add functionality meant that attention was not given to security issues. Second, developers had trouble 'improving security' because their normal work procedures and ways of structuring their work did not support that kind of quality goal. The authors concluded that successful interventions would need "*to investigate the potential business value of security, thus making it a more tangible development goal*"; and that security is best promoted as a team, not individual, effort.

Türpe et al. [18] reported a similar ethnographic study exploring the effect of a penetration testing session and workshop on 37 members of a large geographically-dispersed project. The results were also not encouraging; the main reason suggested by the authors was that the workshop consultant highlighted problems without offering much in the way of solutions (a behavior this paper's authors have observed in other security practitioners).

Recent work by Ashenden and Lawrence [19] took a different approach. They used an Action Research method to investigate and improve the relationships between security professionals and business people in a single company, and found the approach effective in improving communication, though no evidence is yet available of longer-term impact.

D. *Using Formal Education Techniques*

Others have investigated the effect of programmer learning on security improvement. Yskout et al. [20] tested if 'security patterns' (such as described in Schumacher et al.'s book [21]) might be an effective intervention to improve secure development in teams of student software developers; the results suggested a benefit but were statistically inconclusive.

Acar et al. [22] concluded through a survey of nearly 300 successful app developers worldwide that they learned security using web search and from peers. They also used a practical experiment with over 50 Android developers to evaluate the effectiveness of the different ways of learning app security; this produced the surprising result that programmers using only digital books achieved better security than those using web

search. More recently Acar et al. [23] investigated web-based handbooks to see how well they supported such learning; finding that very few provided tutorials or exercises, or discussed wider aspects of security such as social engineering.

E. Motivating Change in Development Teams

To move from delivering insecure code to delivering secure code requires a change in thinking in the development teams. A variety of research has explored how to engender such a change.

Dybå [24] performed a wide-ranging quantitative survey of Software Process Improvement (SPI) in 120 organizations, and concluded that organizational factors were at least as important as technical ones. In particular, he identifies business orientation, the extent to which SPI goals and actions are aligned with explicit and implicit business goals and strategies, as one of the factors with the strongest influence on SPI success; together with employee participation, the extent to which employees use their knowledge and experience to decide, act, and take responsibility for SPI. Surprisingly, management commitment was not required. The paper also strongly recommends that, for SPI, the measurement systems be designed by the software developers themselves.

Beecham et al. [25] conducted a literature review of 92 papers on programmer motivation in 2008. Though virtually all the research cited is about motivation to do the job of programming rather than motivation to change behavior, the survey identifies that professional programmers tend to be motivated most by problem solving, by working to benefit others and by technical challenges. Fear of failure was not among the list of motivators, which suggests that merely frightening developers into security ('a terrible thing might happen') is unlikely to be an effective strategy to promote secure software. This is consistent with Xie et al.'s interviews of 15 professional programmers [2] to investigate why they believed they made security errors; they found a consistent tendency to treat security as 'someone else's problem'.

F. Limitations of Existing Literature

There has been virtually no academic investigation into ways to encourage developers to adopt successful security practices. Since governments and private companies are now investing considerable money into improving developer security, this seems an unfortunate omission. In this work we offer an experiment investigating one possible way to help developers.

III. METHODOLOGY

A. Research Method

How were we best to structure this work as academic research? Given that the researchers themselves directly influence the behavior of the research participants – the researchers *are* the intervention – an ethnographic research approach was inappropriate. Instead an accepted methodology, used in many forms of academic social research, is Action Research [26]. This is an approach to research in communities that emphasizes participation and action; Action Research aims at understanding a situation in a practical context and aims at improving it by changing the situation.

Specifically, we used Participatory Action Research [27], with the lead author working, as 'intervener', directly with the participants. We had a Pragmatic approach, since the intention was to primarily to trial the impact of the interventions. Given that this was to some extent a pilot project, we had only a single feedback cycle [28].

The key research question was: '*What effect did the interventions have?*' To measure an effect, we needed a baseline with no intervention. A-B testing, requiring a different team working in parallel, was not practical. Instead, we used a longitudinal approach, deducing a baseline ('no intervention situation') from the initial situation plus a knowledge of the original plans by the team leaders to improve security over the same timescale.

First, we interviewed a selection of the future participants to establish a baseline in terms of their current understanding, plans and practice related to secure software development. We then carried out a series of intervention workshops with members of the development teams, led by the intervener. Finally, a suitable time after the final intervention workshop, we re-interviewed the same participants as before.

The audio recordings of the interviews and most of the workshops – a total of 7 hours of audio – were transcribed and qualitatively analyzed. In an iterative process, two of the authors coded all transcripts. Initially both authors used open coding [37] on the first two hours of material, then agreed on a coding scheme based on that and the research questions. Differences in coding were discussed and resolved between us. The final code book consisted of 5 families of codes, making a total of 41 codes, applied to 2661 references in total.

In coding, we were looking for aspects of security improvement – including in learning and attitude – implied by statements from the speakers. Specifically, we were looking for signs of new knowledge in the team, new activities related to security, and evidence of improvements in the security of developed software; we were also looking for problems they had encountered and how they had overcome them; we also recorded evidence of baseline security activities and awareness, present or planned, before the start of the interventions.

This research was approved by the Lancaster University FST ethics committee.

B. Implementing the Techniques as Practical Interventions.

The purpose of this work was to use the eight interventions introduced in Section I, which are as follows [3]:

1.	**Incentivization Session**	A workshop or discussion to help developers to understand the impact of security issues.
2.	**Threat Modelling**	Working as a team to identify actors and potential threats; following this up with risk assessment and mitigation decisions.
3.	**On-the-Job Training**	informal sessions sharing security knowledge; also mentoring and having developers more expert at security join the team.

4.	**Continuous Reminder**	Regular activities to keep up the team's awareness of the need for security.
5.	**Component Choice**	Choosing secure components, and keeping them up-to-date; adding component analysis tools to the toolchain.
6.	**Automated Static Analysis**	Using code analysis tools to identify certain categories of security error.
7.	**Code Review**	Introducing scheduled meetings to analyze code for security defects; having other programmers or security experts review code for problems.
8.	**Penetration Testing**	Having external specialist security testers identify flaws, usually using web based tools.

However, each intervention had a variety of forms, suitable for different development budgets, team sizes, and team cultures. What forms would be suitable for us, outside interveners, with limited knowledge of the development team?

Looking at the list of interventions, we observed that four – the first four – are to do with process and can be implemented for limited cost by a team lead or manager; the next three require commitment by the developers; and the last, **Penetration Testing**, is expensive [29]. As outside consultants, therefore, we concentrated on the process interventions, and used opportunities within the consultancy to consider the remaining interventions.

Perhaps the biggest challenge was to find a suitable way to provide the **Incentivization Session**. From prior work [3], we were aware of three forms of such sessions: a two-day training course by a security and training professional aware of the kinds of attack currently happening on the developers' system; a long personal interview with each one of the developers involved; and a full penetration test of the developers' system to identify problems. None of these three approaches was suitable for a lightweight intervention.

Fortunately, while we were working on this challenge we received an enquiry from a colleague. He wanted to use a game, the 'Agile Security Game' [30], invented by the lead author. This was based on the 'Mumba' role-playing game, invented by Frey [31] to help elicit participants' prior experience of real-life security attacks. The 'Agile Security Game' variant, however, was designed for a conference workshop simply to educate developers about security. The colleague wanted to use the game to help motivate development teams towards security. After some discussion we realized that this game would indeed work well as an **Incentivization Session**, and that there was considerable possible benefit for the teams from using the full intervention package.

Threat Modelling, too, was also challenging to implement. Much of the literature [32], [33] describes a heavyweight process taking a while to set up and requiring considerable knowledge of possible technical threats, preferably with support from a professional with a detailed understanding of both the industry sector and current cyber threats to it. But such a process would be expensive in time and commitment, and the required professional knowledge was not available to support us.

However, in this case the researchers' own experience was valuable. As technical lead for a major mobile money project, the lead author had faced this problem in a commercial project. With the help of Alec Muffett, a consultant security expert, his development team had developed a lightweight brainstorming process to identify threats and potential attackers [34]. While this may have lacked the rigor of the secure-development-process-based threat modelling approaches used by some large companies, it had certainly served to deliver a product which was successful as far as its security needs were concerned. Accordingly, therefore we determined to use the same approach here.

On-the-Job Training was not required as an external intervention in this case; the technical leads were enthusiastic about improving security, and were already considering using a variety of different approaches, including working through the OWASP Top Ten [35].

The final process-based intervention was **Continuous Reminder**. For this we agreed to a monthly meeting, by video conferencing; its main purpose was to act as a regular 'nudge' of the importance of security.

To introduce the more technical interventions, we used an ad hoc approach. The facilitator mentioned and discussed each of these interventions with the developers during the **Threat Modelling**, the mitigation discussions, and subsequent sessions, using comments from the developers as cues.

C. Intervention Attitude

We know from literature that developers dislike formal processes [24]; we know also that developers, like most other people, tend to dislike being told what to do and will react against it [33, Ch. 2], So at no point did the facilitator interact with the development teams using terms like *"you must"* or *"it's essential that"*. On the other hand, we also know from personal experience that developers are very happy to solve problems that they agree to be important. Therefore, throughout the workshops and game, we allowed the developers themselves to drive the solutions; as facilitators we provided only guidance.

D. Scope of this Paper

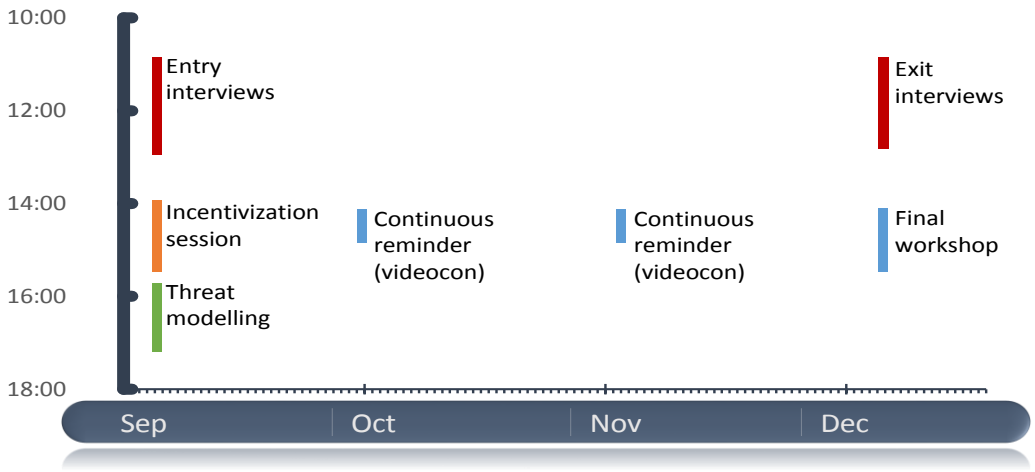
This paper describes our activities with a single organization; the work is part of a much larger ongoing project, involving three organizations in the first year and further organizations thereafter.

IV. This Project

This section discusses the company, project, and development team involved. To preserve confidentiality, we have changed all names, and the exact functionality of the product.

AyCo is a company employing around 50 people in the UK. Set up about 10 years ago, it has a single product which is sold both web based as 'software as a service', and as an installable system for clients' own sites. The product, 'Ambassador', is a web-based accounting and planning package suitable for managing the working of very large organizations. AyCo's customers include several household names.

Figure 1: Project Calendar



A. *Interview Participants*

To help identify the effects of the interventions, we interviewed four team members both before and after the process. They were chosen to provide a cross section representing both experienced and less experienced developers in each team. They are shown in Table 1.

We have included quotations in the remainder of the paper, in *italics*. Where the speaker can be identified, we have cited the appropriate ID. In the recordings of group sessions, however, it was rarely possible to identify individual speakers, and quotations are cited accordingly, e.g. *'Developer, Threat Modelling.'* We have edited the quotations to protect confidentiality and indicate context: square brackets show additions and replacements; ellipses show removals.

ID	Experience	Role	Team
A1	17 years	Architect & Lead	Demigods
A2	2 years	Developer	Demigods
A3	14 years	Senior developer	Superheroes
A4	3 years	Developer	Superheroes

Table 1: Interviewee Roles

V. Results

A. *Intervention Time Requirements*

Figure 1 shows the timeline needed for the interventions. **On-the-job Training** is not shown as it was instigated by the team leads, not the external intervener. As will be seen, despite the long-elapsed time, the total effort required from the intervener was relatively short: a total of two days, of which at least four hours were research interviews and not part of the intervention itself. So, the effort spent for the total intervention was less than one working day. Adding another day for preparation – scheduling, preparing materials for the workshops, etc. – the total time spent by the intervener on the interventions was less than two working days.

About 15 people, from both development teams, attended the incentivization and threat modelling workshops, which were sequential, each taking 1 hour 15 minutes, with a break between. Roughly six people – the two architects and several senior developers – attended the follow-up sessions, which were an hour each. Some ten attended the exit workshop of 45 minutes. Thus, the total time investment from the development team was in the region of 60 man hours, which is about half a day's effort for a 15-person team.

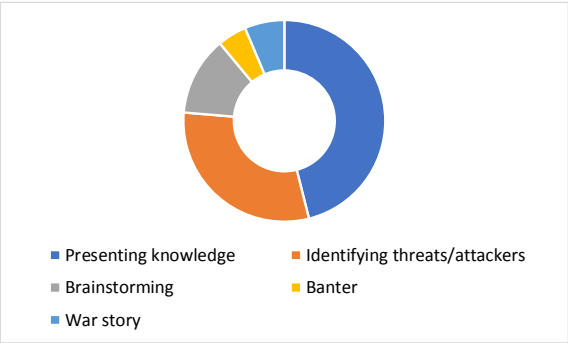
Total Cost	
Intervention facilitator:	15 man hours
Development team:	60 man hours

B. *Discussion Topics*

Figure 2 summarizes the kinds of discourse in the discussions.

We were surprised to find about half of the discussion involved participants, including the facilitator, presenting knowledge in different ways in response to the issues raised. A proportion of this was what we call 'war stories' – complete anecdotes relevant, or partially relevant to the discussion. A relatively small amount of the total time (which included the

Figure 2: Workshop Topics by Words Used



game) was spent on brainstorming; but the other major proportion was, as one would expect, spent identifying threats and attackers. A sign of good esprit-de-corps among the team was the level of 'banter' – shared jokes and friendly insults – between themselves.

The participants clearly enjoyed and learned from the incentivization game workshop [30]:

> *The game was fun, I did enjoy the game. And it was proving as per usual, that... whatever you do, you are going to lose somewhere (A3)*

> *Actually, it was very useful… [showing] you cannot always know the right answer. (A2)*

The Threat Modelling workshop generated some ways of thinking and conclusions that were unexpected for the participants.

> *I never really thought about 'who would', so much, until you put up 'why would somebody and who would they be' (A4)*

In the first follow-up session (not recorded), we discussed prioritization of the mitigations arising from the Threat Modelling session. The main learning point for the participants was that the decisions were mostly for the product manager; not for the 'security expert'.

The second follow-up session was a discussion about their progress so far implementing mitigations, plus a technical discussion about industry experience with encryption as a mitigation.

> *What [are] the advantages of database encryption, and what would that give us, compared to application level encryption… and risks with encryption itself? (Architect)*

Finally, we carried out the exit interviews to help identify the outcomes, as follows.

C. Outcomes Attributable to the Interventions

We can identify two significant improvements in the Ambassador product and process security as a result of the interventions. Beforehand, the developers had been thinking of security improvements as line by line improvements in the code they themselves had written. Afterwards, they understood that their most effective security improvements were likely to be elsewhere. Specifically, they made two changes:

1. They introduced a component security checker to their build cycle, and embarked on a program of updating and replacing components according to their security vulnerabilities.

   > *We [have built] the OWASP dependency checker into our build process, … and established a process for how we deal with new vulnerabilities in existing libraries, or adding new libraries or upgrading libraries. (A1)*

2. They identified their own existing customers as competitors with each other, and therefore potential 'attackers', and identified that the permissions functionality was therefore a major privacy issue; making fixes in this area was likely to give security wins:

   > *I have a … task to check user permissions, and check that a user has access to that specific entity or a set of those entities (A2)*

D. Learning Attributable to the Interventions

We looked for evidence of learning by workshop participants. From the transcripts, we identified three insights:

1. The importance of using secure and up-to-date components.

   > *[Learning about updating components] tends to stick in your memory. I would never trust a third-party library with anything that is going to the user (A4)*

2. Security issues related to business functionality can be as important as technical aspects of security.

   > *I find it a little concerning that there are so many attacks that we traditionally haven't mitigated against. … Stuff like social engineering. (Participant, Threat Modelling)*

3. Decisions on "what security fixes should we implement?" are for product management, not technical developers.

   > *I guess, one challenge, as always, is playing what we, as architects, believe are the most pressing security concerns, against what customers are asking for in terms of dealing with security concerns. (A1)*

E. Outcomes Not Attributable to the Interventions

As we discussed in Section IV, the team leads were already interested in software security. We anticipate, therefore, that they would have made some improvements even without the external interventions we provided. From the initial interviews with A1 and A3 we identified two such improvements that would likely have occurred without the interventions:

First, they had already identified the OWASP Top 10 [35] as being important, and understood the need for regular reinforcement of the security message. So, although one direct outcome of the Threat Modelling session was agreement to study a new OWASP threat each month – an intervention that fits nicely as a Continuous Reminder – this might have happened without our intervention.

> *We also decided to introduce a knowledge improvement process, that wasn't too costly, and didn't distract too much from what we were trying to do. And basically, we decided that we would take the OWASP Top Ten, starting at No.1, and every release, we would focus on that, in some capacity. (Architect, follow-up session 2)*

Second, they made the interesting experiment of assigning one of the team as 'saboteur', with the task of deliberately inserting security defects for others to find. In practice, whilst it kept a focus on security, this technique proved unpleasant for

Table 2: Blockers and Motivators

Blocker	Motivator
The difficulty of incorporating a component analyser tool in the toolchain, especially given they were using a number of components that were not supported by that tool. *The time-consuming bit was trying to get [the tool] to recognize more obscure libraries that we use… we probably spent a man-week's work of time, manually Googling. (A1)*	The enthusiasm on the part of the open-source tool implementers for incorporating AyCo's suggestions. *We have been able to feed into the tool a bit as well, which has been nice... They have been very responsive in terms of issues that we have raised, and suggestions for improvements (A1)*
The significant work involved in upgrading a range of components, and modifying the code to support the upgraded APIs. *We haven't necessarily got to as much of [the component upgrading] as we would have liked to. (A3)*	Representing the upgrades as explicit stories for scheduling: *Hopefully, the architect guys… [will] try and feed some of those stories in. (A3)* And the satisfaction of seeing 'red lights' turn green as the components were updated. *You've got lights that you can turn green - it becomes relatively straight forward to go through turning them green, one after another until they are all green (A1)*
More generally, the additional work involved in implementing and prioritising security enhancements. *[Only] a certain amount of our road map time is given to architecture. And we have ended up diverting most of that time to addressing security vulnerabilities in one way or another, since you first came. And there is still more to do. The downside of that is, obviously, that we don't address other architectural concerns like performance, or code quality. (A1)*	The benefits of security as a feature, whether tick box support for the audits of potential customers, or actual unique selling propositions when compared with others. *I think it has come at just the right time for us, because … the world is moving forward in terms of expectations around security… [and] we are getting more customers for whom security is a bigger concern (A1)*
The difficulty of learning from existing security sources. *I still find reading the OWASP stuff difficult. (A3)*	Learning as a group. *We've adopted this idea of focussing on a particular one of the OWASP Top Ten each release. I think that went pretty well in the first release. (A1)*

the team member assigned, so is unlikely to be continued, nor tried as an intervention elsewhere.

> *The saboteur didn't enjoy being a saboteur ... So, it is on hold at the moment until we can figure out how to make it work, or whether it is a good idea in the long term. (Participants, follow-up session 2)*

F. Problems, and How They Were Overcome

Analyzing the workshops in more detail, we identified several problems encountered in carrying out the security enhancements. We have termed these 'blockers', and against them we have identified successful 'motivators' that permitted team members to overcome them. Table 2 shows the four most important such blockers and their corresponding motivators.

G. General Feedback on the Process

We had some overall feedback during the second follow-up session, suggesting that the developers' own ratings of the intervention impact were lukewarm.

> *I did a quick survey this morning to find out how people thought the process had gone. … And most people thought, it was about average, in terms of effectiveness. So, on a scale of one to five, it was a little under 3, but not terrible. (Team lead)*

In view of the time cost, however, the leaders felt the involvement had been successful.

> *It doesn't feel like it has distracted significantly from the ongoing development process, so that is worth balancing against how much effort (Team lead 2)*

H. Opportunities for Improvement

While our primary goal from the intervention was to improve the security of code delivered by the team, a secondary goal was to empower participants to deliver secure code in future. Accordingly, in the exit interviews we were interested to what extent they had understood the process as well as learning about security and its application to the Ambassador product. To avoid leading questions, we phrased this as an open question:

> *Let's imagine there's a team starting a similar project now… What would you recommend that's the same as we did, and how would you recommend improving it?*

The resulting answers and discussion around them showed the understanding of specific threats they had encountered, of

prioritization, and an increased understanding of **Component Choice**.

> *[I'd] address the most important threats, and first implement those things, that would help us to address … attacks that would be more harmful (A2)*
>
> *The OWASP stuff... Libraries, definitely, don't use anything old, always. (A3)*
>
> *More security focus… [for example] if it was a site where I was taking user details, I am aware now of EU regulations… it has to be stored securely. (A4)*

Only A1, the lead architect appreciated the importance of the interventions as a process to achieve that.

> *I think the workshop approach was really good fun, and really interesting. I think brainstorming threats and vulnerabilities and assets was really helpful. (A1)*

That suggests a need in future interventions for some more explanation of the process and reasons for it. That might be appropriate as a short presentation by the facilitator at the start, and even shorter 'milestone' presentations before each later session. A participant also suggested introducing a checklist.

> *I think maybe some sort of tick sheet in terms of 'have you got these things in place' to take away, that might be a good addition (A1).*

It might be demotivating to do this at the start, risking being perceived as telling participants 'what to do' [36]. Instead we plan to introduce such a checklist after the initial workshops, thus enhancing the perceived effectiveness as discussed in section F above.

In terms of improvements to the research methodology, we have identified two key limitations of our approach in this project, and plan to redress them in future work as follows:

Need for evidence whether the change is long-term	Repeat exit interviews one year after intervention, so see if enhancements have been retained.
Need for quantitative evidence	Apply a scoring system for the team's security activities and knowledge before and after the interventions.

VI. Conclusions

A. Improvements on Existing Practice

Current practice in interventions is often based on Penetration Testing. Aside from the high cost [29], this approach can prove ineffective in the longer term [18].

The approach described in this paper is significantly less costly, in that the skills required are the much more readily-available ones associated with facilitation. It also requires a smaller amount of effort from the interveners.

While it is early to know the long-term impact, it is probable that that the team will preserve their component security evaluation and upgrade process, and that the understanding will remain of the business functionality being a part of the security story. The motivation of the team towards security has also remained throughout the three months of the pilot; given the team's increased understanding of the business impact of security issues there is at least a reasonable possibility that it will remain long-term.

This intervention approach, therefore, offers inexpensive and impactful security improvements for development teams like those at AyCo.

B. Conclusion and Future Work

In this case study, the authors used a light-touch facilitation-based 'intervention' to two productive and successful development teams. The intervention required limited effort from the facilitator, and relatively little from the development teams. The authors demonstrated, via Participative Action Research and detailed coding, evidence of improved understanding of several important lessons in software security; also, two significant security enhancements implemented for the product as a result.

A single case study is insufficient to draw conclusions about the general effectiveness of this kind of intervention. Further work in this project includes trialing the interventions with other kinds of software development teams and comparing and differentiating the consequences. Further future work suggested by the section 'Opportunities for Improvement' above is to empower the participants by improving the process, and even to have the researchers train facilitators among the participants rather than providing the facilitation themselves.

Lightweight, facilitation-based, interventions of the kind used here offer the potential to help many software development teams improve their software security. Wider-scale adoption of the process will empower developers and play a much-needed role in improve software security for all end users.

References

[1] S. Morgan, "2017 CyberVentures Cybercrime Report," 2017. [Online]. Available: https://cybersecurityventures.com/2015-wp/wp-content/uploads/2017/10/2017-Cybercrime-Report.pdf.

[2] J. Xie, H. R. Lipford, and B. Chu, "Why Do Programmers Make Security Errors?," *Proc. - 2011 IEEE Symp. Vis. Lang. Hum. Centric Comput. VL/HCC 2011*, pp. 161–164, 2011.

[3] C. Weir, A. Rashid, and J. Noble, "Developer Essentials: Top Five Interventions to Support Secure Software Development," Lancaster University, 2017.

[4] Y. R. Smeets, "Improving the Adoption of Dynamic Web Security Vulnerability Scanners," Radboud University, NL, 2015.

[5] J. Xie, B. Chu, H. R. Lipford, and J. T. Melton, "ASIDE: IDE Support for Web Application Security," *Proc. 27th Annu. Comput. Secur. Appl. Conf. - ACSAC '11*, p. 267, 2011.

[6] I. Pribik and A. Felfernig, "Towards Persuasive Technology for Software Development Environments: An Empirical Study," in *International Conference on Persuasive Technology*, 2012, pp. 227–238.

[7] D. Nguyen, Y. Acar, and M. Backes, "Developers Are Users Too: Helping Developers Write Privacy Preserving and Secure (Android) Code," 2016. [Online]. Available: https://www.usenix.org/sites/default/files/soups16poster22-nguyen.pdf.

[8] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why Don't Software Developers Use Static Analysis Tools to Find Bugs?," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 672–681.

[9] T. B. Jordan, B. Johnson, J. Witschey, and E. Murphy-Hill, "Designing Interventions to Persuade Software Developers to Adopt Security Tools," in *Proceedings of the 2014 ACM Workshop on Security Information Workers - SIW '14*, 2014, pp. 35–38.

[10] E. M. Rogers, *Diffusion of Innovations*. Simon and Schuster, 2010.

[11] S. Xiao, J. Witschey, and E. Murphy-Hill, "Social Influences on Secure Development Tool Adoption: Why Security Tools Spread," *Proc. ACM Conf. Comput. Support. Coop. Work. CSCW*, pp. 1095–1106, 2014.

[12] S. Dodier-Lazaro, I. Becker, J. Krinke, and M. A. Sasse, "'No Good Reason to Remove Features': Expert Users Value Useful Apps over Secure Ones," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10292 LNCS, no. c, 2017, pp. 25–44.

[13] A. Bessey, D. Engler, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C. Henri-Gros, A. Kamsky, and S. McPeak, "A Few Billion Lines of Code Later," *Commun. ACM*, vol. 53, no. 2, pp. 66–75, 2010.

[14] E. Murphy-Hill, D. Y. Lee, G. C. Murphy, and J. McGrenere, "How Do Users Discover New Tools in Software Development and Beyond?," *Comput. Support. Coop. Work*, vol. 24, no. 5, pp. 389–422, 2015.

[15] B. De Win, R. Scandariato, K. Buyens, J. Grégoire, and W. Joosen, "On the Secure Software Development Process: CLASP, SDL and Touchpoints Compared," *Inf. Softw. Technol.*, vol. 51, no. 7, pp. 1152–1171, Jul. 2009.

[16] T. Baum, O. Liskin, K. Niklas, and K. Schneider, "Factors Influencing Code Review Processes in Industry," in *FSE2016*, 2016, pp. 85–96.

[17] A. Poller, L. Kocksch, S. Türpe, F. A. Epp, and K. Kinder-Kurlanda, "Can Security Become a Routine? A Study of Organizational Change in an Agile Software Development Group," in *Proc. CSCW'17*, 2017, pp. 2489–2503.

[18] S. Türpe, L. Kocksch, and A. Poller, "Penetration Tests a Turning Point in Security Practices? Organizational Challenges and Implications in a Software Development Team," in *WSIW at Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*, 2016.

[19] D. Ashenden and D. Lawrence, "Security Dialogues : Building Better Relationships," *IEEE Secur. Priv. Mag.*, vol. 14, no. 3, pp. 82–87, 2016.

[20] K. Yskout, R. Scandariato, and W. Joosen, "Do Security Patterns Really Help Designers?," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, pp. 292–302.

[21] M. Schumacher, E. Fernandez-buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, 2005.

[22] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, "You Get Where You're Looking For: The Impact of Information Sources on Code Security," in *IEEE Symposium on Security and Privacy*, 2016, pp. 289–305.

[23] Y. Acar, C. Stransky, D. Wermke, C. Weir, M. L. Mazurek, and S. Fahl, "Developers Need Support, Too: A Survey of Security Advice for Software Developers," *IEEE Secur. Dev. Conf.*, pp. 22–26, 2017.

[24] T. Dybå, "An Empirical Investigation of the Key Factors for Success in Software Process Improvement," *IEEE Trans. Softw. Eng.*, vol. 31, no. 5, pp. 410–424, 2005.

[25] S. Beecham, N. Baddoo, and T. Hall, "Motivation in Software Engineering: A Systematic Literature Review," *Inf. Softw. Technol.*, vol. 50, no. 9, pp. 860–878, 2008.

[26] W. F. Whyte, *Participatory Action Research*. Sage Publications, Inc, 1991.

[27] R. L. Baskerville, "Investigating Information Systems with Action Research," *Commun. AIS*, vol. 2, no. 3es, p. 4, 1999.

[28] K. Petersen, C. Gencel, N. Asghari, D. Baca, and S. Betz, "Action Research as a Model for Industry-Academia Collaboration in the Software Engineering Context," in *Proceedings of the 2014 International Workshop on Long-Term Industrial Collaboration on Software Engineering*, 2014, pp. 55–62.

[29] J. M. Such, A. Gouglidis, W. Knowles, G. Misra, and A. Rashid, "The Economics of Assurance Activities," 2015.

[30] C. Weir, "The Agile App Security Game – Leader's Instructions," 2017. [Online]. Available: https://www.securedevelopment.org/app/download/11233441072/TheAgileAppSecurityGame.zip. [Accessed: 28-Mar-2018].

[31] S. Frey, "Mumba Role Playing Game: The Rulebook." 2016.

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

[33] Microsoft, "Microsoft Secure Development Lifecycle." [Online]. Available: https://www.microsoft.com/en-us/sdl/. [Accessed: 29-Mar-2018].

[34] Penrillian, "Penrillian's Secure Development Process," 2014. [Online]. Available: http://www.penrillian.com/sites/default/files/documents/Secure_Development_Process.pdf. [Accessed: 21-Jul-2016].

[35] OWASP, "Top Ten Project," 2017. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project. [Accessed: 31-Oct-2017].

[36] R. Fisher, A. Sharp, and J. Richardson, *Getting It Done: How to Lead When You're Not in Charge*. HarperPerennial, 1999.

[37] B. G. Glaser and A. L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago: Aldine Transaction, 1973.

[38] K. Schwaber, *Agile Project Management with Scrum*. Microsoft press, 2004.