Security Lancaster | Lancaster University

Developer Cyber Essentials: Trialling Interventions to Improve Development Security

Summary Report – April 2018

Management Summary

Cyber security is a big and increasing problem. Yet 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.

A series of interventions costing less than two days' total effort from a facilitator can significantly improve the software security of the products developed by teams in a wide range of companies.

This report introduces this series of interventions, explains how they were derived from previous research in industry, and evaluates using Participative Action Research their effectiveness with software development teams in three widely-varied organisations. The interventions are:

- A game-based Incentivisation Session, in which participants work in groups to prioritise security enhancements;
- A lightweight Threat Modelling Session, based on ideation of possible issues and agents; and
- Repeated follow-up sessions, involving discussion, threat modelling or risk analysis according to the needs of the participants.

The interventions proved very effective with the teams in two companies, that were less experienced and knowledgeable about software security. They were only marginally effective with more experienced teams.

We conclude that this intervention process provides a cost-effective and powerful means to improve the effectiveness of the majority of development teams who still lack the vital skills and motivation to deliver secure software.

Charles Weir (Security Lancaster, Lancaster University)
Dr Lynne Blair (SCC, Lancaster University)
Prof. James Noble (ECS, Victoria University, Wellington, NZ)
Ingolf Becker (University College London)
Prof. Angela Sasse (University College London)

1 Introduction

Software security and privacy are now major issues: the cost and potential threat to us all is increasing dramatically [5]. With over 100 billion lines of code created a year [5], 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' [13]. 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 [10,11], the authors identified through two sets of interviews with leading software security experts, a range of eight techniques (see Section 2.2) for people to encourage and support software development teams to deliver secure code. These techniques have the potential to improve secure software development.

In the project described in this report we investigated having a consultant lead the introduction of the techniques using lightweight consultancy 'interventions' in three very different organisations.

The contributions of this short report are:

1. A low-cost but effective intervention method to help teams improve their development security, and
2. Exploration of the intervention method's effectiveness with a range of types of development team

The rest of this report is as follows. **Section 2** explains the research and analysis methods in detail. **Section 3** introduces the companies and development teams involved. **Section 4** discusses the results obtained and compares the approach with existing practice, and **Section 5** summarises the conclusions.

2 Methodology

2.1 Research Method

Initially we considered an ethnographic research methodology, but it was inappropriate for a situation where an observing researcher was also a key participant. Instead we adopted a method used in many forms of academic social research: Action Research [12]. This approach emphasises participation and action; it aims to understand a situation in a practical context, and to improve it by changing the situation.

Specifically, we used Participatory Action Research [1], with the lead author working, as 'intervener', directly with the participants. In each of three companies we first interviewed a selection of the participants to establish a baseline in terms of their current understanding 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, as far as possible, participants as before.

The audio recordings of the interviews and most of the workshops – a total of 19 hours of audio – were transcribed and qualitatively analysed. In coding, we were looking for aspects of security improvement implied by statements from the speakers. Specifically, we were looking for evidence of improvements in the security of developed software.

---

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

This research was approved by the Lancaster University Faculty of Science and Technology ethics committee.

2.2 Implementing the Techniques as Practical Interventions.

The purpose of this work was to find ways best to improve developer security using the eight interventions derived from previous work [10,11], which are as follows:

Incentivisation Session	A workshop or discussion to help developers to understand the impact of security issues.
Threat Modelling	Working as a team to identify actors and potential threats; following this up with risk assessment and mitigation decisions.
On-the-Job Training	brownbag sessions sharing security knowledge; also mentoring and having developers more expert at security join the team.
Continuous Reminder	Regular activities to keep up the teams' awareness of the need for security.
Component Choice	Choosing secure components, and keeping them up-to-date; adding tools to the toolchain to support this.
Automated Static Analysis	Using code analysis tools to identify certain categories of security error.
Penetration Testing	Having security testers identify flaws, based on a knowledge of common defects and using specialist web based tools.
Code Review	Introducing scheduled meetings to analyse code for security defects; having other programmers or security experts review code for problems.

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 to some extent by a team lead or manager; the remaining four are more technical and must be implemented by developers themselves. As outside consultants, therefore, we concentrated on the process interventions, and used opportunities within the consultancy to promote and improve the technical interventions.

The biggest challenge was to find a suitable way to provide the **Incentivisation Session**. For this, we used a game, the 'Agile Security Game' [9], developed by the lead author. This was based on the 'Mumba' role-playing game invented by Frey et al. [4], to help elicit participants' prior experience of real-life security attacks. The 'Agile Security Game' variant, however, was designed simply to educate developers about security. In it, participants act as product managers, selecting security-enhancing product improvements with varying costs and learning whether their choices deter attacks.

Threat Modelling, too, was also challenging to implement. But as technical lead for a major mobile money project, the lead author had faced this problem in a commercial project. With the help of a consultant security expert [14], his development team had developed a lightweight brainstorming process to identify threats and potential attackers [6]. We determined to use the same approach for these interventions.

Given our emphasis on a 'lightweight' approach, **On-the-Job Training** was not practical as an external intervention. However, all the companies already had some form of such training taking place.

The final process-based intervention was **Continuous Reminder**. For this we agreed to a monthly meeting, by Skype video conferencing 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 the remaining interventions with the developers during the **Threat Modelling**, the mitigation discussions, and subsequent **Continuous Reminder** sessions, using suggestions from the developers as cues.

2.3 Intervention Attitude

We know from literature that developers dislike formal processes [2]; 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 take action 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.

3 Participating Companies

This section introduces the three different companies, with the projects and development teams involved. To preserve confidentiality, we have changed all names, and the exact functionality of the products involved.

3.1 Company A

Company A is a small-to-medium company employing around 50 people in the UK. Set up about 10 years ago, it has a single product which is sold both as 'software as a service', and as an installable system for clients' own sites. This product manages sensitive data, and is used by some very large organisations, including several that are household names.

The company development teams show some of the enthusiasm and characteristics of a start-up. We observed a culture of technological improvement, and a willingness to embrace change.

3.2 Company B

Company B is a tiny non-profit start-up, run on a part-time basis by two professionals: an educationalist and a software project manager. Other staff also assist on a part-time basis. The company purpose is to provide work experience for promising young people who would otherwise be unable to get initial jobs in IT.

3.3 Company C

Company C is a well-known and long-established multi-national organisation, providing information services mainly via the Internet to a range of companies and individuals. The team members involved were testers, managers and programmers of the membership system. This is mature software, but the company has a policy of continuous architectural improvement. All the team were competent and experienced professionals, but in contrast to company A we noted more emphasis on inter-departmental politics.

4 Results

4.1 Intervention Time Requirements

While the interventions took place over three to four months for each company, the total effort required from the intervener was relatively short: excluding the time for travel and research interviews the total effort spent for the interventions 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 for each company. The time spent by the development teams was typically 3-4 hours by the whole team, and 2-3 more hours by the more senior members. Thus the time requirements for this set of interventions was as follows:

Total Cost	
Intervention facilitator:	15 man hours
Development team:	20 - 70 man hours depending on team size.

The cost of the interventions, therefore, is relatively small, and is within the scope of a wide range of organisations.

4.2 Outcomes Attributable to the Interventions

This section identifies the concrete outcomes attributable to the interventions. Further details will be presented in sections 4.3 and 4.4.

4.2.1 Outcomes for Company A

We identified at least two significant improvements in Company A's product and process security as a result of the interventions. Beforehasnd, 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 components' security checker to their build cycle, and embarked on a programme of updating and replacing components according to their security vulnerabilities.
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.

4.2.2 Outcomes for Company B

Company B, coming from a lower baseline in terms of security experience, had more potential improvements in process and in product security. As a result of the first **Threat Modelling** process we identified two changes. They changed the planned design of the website not to use local databases for storing form data; and they introduced improved security for development workstations and code repositories, keeping them upgraded and password-secured, against the threat of malicious code msodifications or access to personal data. Later they made further improvements: the student developers improved their security hygiene; and in a subsequent project they took particular care with web service access keys.

4.2.3 Outcomes for Company C

It was difficult to identify any concrete outcomes from the interventions for Company C. The primary reason for this is that their security knowledge and practice as a team were already good – rather better than they may have realised. While some security improvements were made, we believe these were the result of a wider awareness of security needs within the organisation rather than specifically because of our interventions.

Indeed, the two main issues highlighted from the **Threat Modelling** – control of physical access to workstations and the relationship with the company's security department – were outside the control of any of the participants in the workshops. This hadn't been identified before; but we have no evidence that matters have changed.

4.3 Interventions Adopted

Table 1 lists all interventions from section 2.2, and the extent to which each was adopted by the three companies.

Table 1: Summary of Interventions Eventually Adopted

	Company A	Company B	Company C
Incentivisation Session	Provided by Intervener	Provided by Intervener	Provided by Intervener
Threat Modelling	Led by Intervener	Led by Intervener (2 sessions)	Led by Intervener
On-the-Job Training	Instigated study of each OWASP Top 10 vulnerability per release.	Introduced as educational benefit following intervention.	Already in place – e.g. C5 had worked in the Security Team.
Continuous Reminder	Provided by Intervener	Provided by Intervener	Stream of 'incidents' to handle.
Component Choice	Introduced due to Intervention	Introduced (at a basic level) due to intervention.	Already in place.
Automated Static Analysis			Already in place.
Penetration Testing	Already in place		Already in place.
Code Review	Already in place (peer		Already in place

As shown, there were some further activities prompted by the interventions, but by no means all.

While the last two (**Penetration Testing** and **Code Review**) might have been inappropriate [10] for cash- and discipline-strapped Company B, we might have hoped to consider **Automated Static Analysis** for both Companies A and B; in fact the possibility was not discussed. One possible approach to encourage considering these forgotten interventions was some form of checklist.

Another improvement suggested by participants from both Company B and Company C was an initial presentation explaining the aims and process of the interventions.

4.4 Improvements on Existing Practice

Current practice in interventions is often based on Penestration Testing. Aside from the high cost [7], this approach can prove ineffective in the longer term [8].

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, and the software security knowledge required is an overview rather than in-depth. It also requires a smaller amount of effort from the interveners (section 4.1) and achieved identifiable impact in terms of security improvements (section 4.2).

The research approach, Participative Action Research, provided a useful framework for evaluating the interventions, and the coding techniques provided a measure of objectivity to the results.

We suggest that the techniques described in this paper, supported by the improvements of an initial presentation and a checklist of interventions as discussed in section 4.3, offer significant potential as the basis for a lightweight programme to improve the security performance of non-security-expert development teams.

5 Conclusion

This report summarises the effect of introducing a light-touch facilitation-based set of 'interventions' to development teams in three very different organisations: a mature SME, a start-up focussed on providing work experience, and a multinational information service provider. The intervention required limited effort from the facilitator, and relatively little from the development teams. The authors demonstrated, via an Action Research method, evidence of security enhancements introduced by the less experienced teams, while more security-expert teams only benefitted marginally.

Lightweight, facilitation-based, interventions of the kind used here offer the potential to help any software development teams with limited current security skills to improve their software security. Widescale adoption of the process will empower developers, and play a much-needed role in improving software security for all end users.

6 References

[1] Baskerville, R.L. Investigating Information Systems with Action Research. *Communications of the AIS 2*, 3es (1999), 4.

[2] 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.

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

[4] Frey, S., Rashid, A., Anthonysamy, P., Pinto-Albuquerque, M., and Naqvi, S.A. The Good, the Bad and the Ugly: A Study of Security Decisions in a Cyber-Physical Systems Game. *IEEE Transactions on Software Engineering*, (2017).

[5] Morgan, S. 2017 CyberVentures Cybercrime Report. 2017, 14. https://cybersecurityventures.com/2015-wp/wp-content/uploads/2017/10/2017-Cybercrime-Report.pdf.

[6] Penrillian. Penrillian's Secure Development Process. 2014. http://www.penrillian.com/sites/default/files/documents/Secure_Development_Process.pdf.

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

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

[9] Weir, C. and Lancaster, S. The Agile App Security Game The Agile App Security Game – Leader's Instructions. 2017. https://www.securedevelopment.org/app/download/11233441072/TheAgileAppSecurityGame.zip.

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

[11] Weir, C., Rashid, A., and Noble, J. I'd Like to Have an Argument, Please: Using Dialectic for Effective App Security. *Proceedings 2nd European Workshop on Usable Security*, Internet Society (2017).

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

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

[14] Wikipedia: Alec Muffet. https://en.wikipedia.org/wiki/Alec_Muffett.