Why Do Programmers Do What They Do? A Theory of Influences on Security Practices

Lavanya Sajwan, James Noble, Craig Anslow
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
sajwanlav@gmail.com {kjx,craig}@ecs.vuw.ac.nz

Robert Biddle
School of Computer Science
Carleton University
Ottawa, Canada
robert.biddle@carleton.ca

***Abstract*—Technologies are continually adapting to match ever-changing trends. As this occurs, new vulnerabilities are exploited by malicious attackers and can cause significant economic damage to companies. Programmers must continually expand their knowledge and skills to protect software. Programmers make mistakes, and this is why we must interpret how they implement and adopt security practices. This paper reports on a study to understand programmer adoption of security practices. We identified a theory of inter-related influences involving programmer culture, organizational factors, and industry trends. Understanding these decisions can help inform organizational culture and education to improve software security.**

I. Introduction

Software is now ubiquitous across many industries [6]. Programmers have to ensure that the security processes that they implement are resilient to attacks. Lack of attack prevention can cause leakage of sensitive information, significant economic damage and danger to massive numbers of users and employees. Consequently, this opens businesses, clients and end-users to exploitation by threat actors.

When security and privacy issues do occur in real-world scenarios, programmers often get blamed first as it is often faults in their implementation which introduce vulnerabilities [8]. Programmers do make mistakes, and lacking infrastructural safeguards [8], security is their own responsibility. Security mechanisms often have increased complexity, which makes them challenging to understand and use, and many programmers lack sufficient motivation and training [4].

In 2019 New Zealand (NZ) experienced a significant security breach within the Tū Ora Compass Health (Tū Ora) Primary Health Organisation (PHO) [16]. Tū Ora is one of the largest PHOs in the country, and it governs the greater Wellington region [16]. Personal information of up to one million New Zealanders were exposed, and the effects of this are ongoing [16]. Costs have been high in attempts to mitigate any effects to the public. Dedicated call centres have been set-up as well as dedicated mental health lines [16], [17].

Workshop on Usable Security and Privacy (USEC) 2021
7 May 2021, Auckland, New Zealand
ISBN 1-891562-73-8
https://dx.doi.org/10.14722/usec.2021.23012
www.ndss-symposium.org

This paper reports on a study to investigate how programmers implement and adopt security practices in the work they do, in order to develop an understanding of the influences involved. The study used Grounded Theory, a method which aims to establish a theory when there is none, where the method has been used for qualitative data analysis in software engineering [10], [11], [13]. Interviews were used to collect the data, and we then analysed and developed a theory on influences on security practices in the professional workplace. This study builds upon earlier research on programmer security practices [1], [18], [24].

Three key research questions we began with were:

RQ1: What security **challenges** do programmers face?
RQ2: What security **training** do programmers have?
RQ3: How do programmers **adopt** secure software practices?

Participants were recruited by posting on tech-related groups, mailing lists, and also using personal contacts. 15 semi-structured interviews were conducted on the topic of their security practices while programming. The participants were all programmers in NZ in varying stages of their careers to allow for a broad range of responses and a case study relevant to NZ. Our findings offer a new perspective on the psychology of programmers which we suggest might inform education and the workplace.

II. Related Work

This study aims to gather data in order to ultimately form a theory on the influences of programmer security practices in NZ. We begin by reviewing the on security and software development practices in NZ and more generally.

A study run by Aura Security showed a 10% increase in cyber-attacks on NZ businesses between the years of 2018-2019 [2]. In NZ, the rise is attributed to the digitisation of day-to-day way-of-life; as new technologies continually develop, there is an urgency to deploy products. There is a lot of pressure on programmers to finalise these products, and the deviation of attention to the finished output means that there are many new threats to security [5]. Malicious attackers are also becoming more sophisticated. Individual threat attackers now seem to have the same knowledge and resources as nation-backed threat actors [5]. Areas of improvement identified by the NZ government are as follows:

1) **Cyber security aware and active citizens:** Increased regular awareness campaigns and education opportunities for the public in regards to best personal security practices.
2) **Strong and capable cybersecurity workforce and ecosystem:** Increased promotion and support of the development of the cyber industry in NZ.
3) **Internationally active:** Detect and prevent any breaches as well as proactively maintaining international relationships regarding information security and participating in any rule reforms.
4) **Resilient and responsive NZ:** Supporting infrastructure, businesses, charity organisations, community organisations, individuals in improving security capabilities and resilience.
5) **Proactively tackle cybercrime:** Increasing support to impacted parties, preventing and encouraging reporting of any cybercrimes.

These five principles are planned to inform improvement over the period to 2023 [5]. It is expected that aspects of these will influence the security practices that programmers use in NZ industry. Without a focus on security while programming, there is a lack of confidence in the resulting security, resulting in the lack of use and waste of time, effort and money. Using unsecured programs poses significant adverse impacts to businesses and individuals.

Kirk and Tempero [14] looked at "developing and applying a range of software productivity techniques and tools to enhance the performance of the NZ software industry." Their survey aimed to understand the practices used by industry and in the findings can be used to make recommendations on best-use development practices for organisations. The key findings were:

1) Organisations and individuals **do not follow** standard agile process models.
2) NZ is generally more **implementation-focused** in software development. There is an emphasis on this over other aspects of the software development life-cycle such as security and testing.
3) Decision-making is a **collaborative effort** with individuals involved in different stages and traits of the development life-cycle.
4) While most New Zealanders state they are "agile" this is not supported as frequent contact with clients and stakeholders is not upheld. However, there is a **highly iterative aspect** to the work individuals do on projects which do maintain agile principles.
5) There is a **weakness in requirements gathering** which results in a widely noticed lack of clarity on scope details.
6) This point also relates to point 2, there is a noticed severe **lack of code quality** whether this is in design, reviewing and testing stages, or with general coding best practices.
7) Most **do not develop around** tools such as libraries - rather they use them as a support. This can be derived as not being "best-practice" and can be more time-consuming.

Not much was asked specific to security, but finding number 6 links to poor practices around secure programming. The report had a limitation in which it did not make any recommendations at this stage, but it did mention that these findings can be used by organisations to obtain a view of the software practices in NZ. From here, organisations can make their own decisions on what to focus on to better their specific operations.

In a survey with software developers in North America [1], Assal and Chiasson investigated the human behaviours and motivations surrounding factors of software security. The authors specified a series of questions targeted toward software developers through an online survey. The research examined responses to support the professional development of programmers further, both in theory and practice. The results outlined the following common groups:

1) **Work Motivation:** Developers did not lack motivation in their job. They performed based on self-determination.
2) **Understanding of software security:** Developers had a sound understanding of software security. They grasped the importance of securing technical work and discussed various methods of doing so and specifying at what stages in the project life-cycle they should implement these based on best practices.
3) **Security Issues:** A majority of the participants believed their software could be compromised, despite being comfortable with the approaches to protecting the software. The majority has also experienced a security issue, whether that be a breach or vulnerable code.

The overarching theme was that the developers were not purposefully ignorant about maintaining security practices — the majority were proactive and willing to learn. However, it was the importance of functionality and lack of ongoing support from organisations which made working towards more secure software challenging.

Weir et al. [25] identified that security is more reliant on developers, but that the developers are not providing the security that is needed. The authors interviewed participants in order to obtain data so they can find ways to "help programmers themselves to improve security given existing constraints". The two major findings were:

1) Developer security is based on challenges in order to motivate better practice. These challenges are often fun adversary questioning usually to do with review and advisory. This emerged as the core theory as it was interwoven through most of the participant responses.
2) Six assurance techniques were identified in being the most helpful; threat assessment, stakeholder negotiation, configuration review, vulnerability scan, source code review and penetration testing. They all help provide software security.

Tahaei and Vaniea [23] conducted an extensive literature review of 49 publications on security studies with participants who were software developers. There were eight significant themes in the results shown by the authors. They were

"Organisations and Context", "Structuring Software Development", "Privacy and Data", "Third Party Updates", "Security Tool Adoption", "Application Programming Interfaces (APIs), "Programming Languages" and "Testing Assumptions". We were particularly interested in the theme "Organisations and Context", which focuses on how developers were influences by those factors. The authors discuss how dedicated security teams do see security as a priority, but their influence is typically small on the overall project.

III. Methodology

A. Grounded Theory

The research method of our study was based on Grounded Theory as first articulated by Glaser and Strauss [7]. This approach was originally created for taking a fresh look at situations with minimal focus from existing theory, with the intent to identify new theory grounded in the sampled data [11]. The emergent theory is expected to be explanatory, focusing on describing how the data informs answers to the research questions. As such, the theory should also be based on the collected responses and observations, rather than preconceived ideas. We chose this approach in hope of finding a new perspective on the issue of software development security practices. Throughout the study, we kept in mind the ACM SIGSOFT Grounded Theory recommendations: exploration of a broad area of study without deliberate framing, an iterative data analysis method, and ensuring findings have the support from the data itself [21].

Our University's Human Ethics Committee approved our study, and we then followed the steps advised for Grounded Theory. We contacted potential participants from the population of interest, asking their consent for a semi-structured interview. We then took an iterative approach conducting interviews, and analyzing the results to refine the questions for later interviews. This refinement of questions enabled delving deeper into the traits of the emerging theory. When interviews cease to reveal new findings, "saturation" is said to be achieved, and allowed articulation of our grounded theory.

Initially, we imagined getting key evidence about technical security practices of programmers. However, we identified as early as the third interview that technical security programming practices were not as influential as we expected, and the interview questions were adapted to match this gradual shift in focus. This led to the new perspective we found on the topic, as we describe in the following sections.

B. Data Collection

A recruitment post and a web-page was shared to NZ mailing lists and websites for software developers, and also personal contacts. We also used the "snowballing" approach, where participants helped with recruitment by reaching out to others in the industry and telling them about the study. We sampled participants which had varying job titles, years of experience, and were in a range of different industry sectors. 15 interviews were conducted with participants across NZ, with participants from 14 different organisations. The diversity within the participants allowed for connections to be made across a broader range of people.

Our initial questions were developed based on a small pilot study run with recent graduates from our University. This small pilot study was essential in providing some more clarity on the interview process, allowed practice in conducting interviews on the topic, and helped begin the continuous process of refining questions.

Due to 2020 being a disruptive year because of COVID-19, interviews were predominantly conducted over Zoom. Consent from the participants was obtained through an email response as per Human Ethics committee approval. Interviews ran for periods ranging between 30-90 minutes. A summary of the participants is shown in I. Four participants had greater than 15 years experience (expert), four had 10-15 years experience (senior), four had between 2-5 years experience (intermediate), and three less than 2 years experience (graduate).

TABLE I. Participants aliases and summary.

ID	G	Role	Organisation	Years
P1	M	Enterprise Architect and Domain Lead	Government Agency	>15
P2	M	Principal Product Architect	Software Development Company	>15
P3	M	Senior Security Architect	Financial Technology	10-15
P4	M	Senior Software Engineer	Software Development Company	>15
P5	M	Software Developer	Financial Technology	<2
P6	F	Level 2 Security Analyst	Information Technology Services	2-5
P7	M	Site Reliability Engineer	Financial Technology	<2
P8	F	DevOps Engineer	Software Integration Services	<2
P9	M	Portfolio Architect	Government Agency	10-15
P10	F	Full-stack Software Developer	Utilities Company	2-5
P11	F	Cloud Engineer	Financial Technology	2-5
P12	F	Consultant (Cloud Engineering)	Consultancy Firm	2-5
P13	M	Development Manager and Technical Lead	Financial Technology	10-15
P14	M	Senior Software Engineer	Information Technology Services	10-15
P15	M	Business Rule Consultant	Government Agency	>15

C. Data Analysis

We transcribed each interview and analysed the text using the Qualitative Data Analysis software tool Nvivo[1] We augmented the interview text with memos describing our thoughts on the content. We then conducted selective coding [7]. Key points from each transcript were paired with a simplified summary of the points [11]. The constant comparison method was used, and this is where codes were compared to others from within the same interview and also with other interviews. These comparisons continued to occur as more interviews were conducted and were further narrowed to become categories for the emergent theory. Examples of the codes from the study are shown in Table II. The different columns show the constant comparison method which occurs to match raw data to codes which then are narrowed to concepts and ultimately categories.

Theoretical saturation is the point in a Grounded Theory study where no new significant information is being added

[1]NVivo qualitative data analysis software; QSR International Pty Ltd. Version 12, 2018.

TABLE II. SAMPLE OF SELECTIVE CODING

Raw Data	Code	Concept	Category
...languages used by the company	Already in-use	Established Process	Organisation
...yeah we just use kanban across the company	Already in-use	Project Management	Organisation
...cloud is the newer process	Cloud is getting popular	Improving Process	Trends

to inform the emergent theory. This started to occur during interviews from P10 to P15, after which we stopped conducting interviews. Theoretical memos were written throughout this coding process to articulate relationships between concepts and categories.

The emergent theory we identified consists of three main categories: Cultural Factors, Organisational Factors, and Industry Trends (see Figure 1). In the following sections we present each of these in turn. We then identify the relationships between them, and finally offer a discussion and our conclusions.

IV. CULTURAL FACTORS

This category involves the effect of culture we see in groups of individuals. We specifically mean the small-scale culture local to the workplace, with the characteristics that programmers have relating to security: their knowledge sharing customs but also their attitudes, biases, and experience.

A. Knowledge Sharing

Knowledge sharing is the deliberate exchange of information [9], and it was highly regarded among the participants. Participants identified communication between team-mates as being a fundamental way to learn better and newer security practices. These opportunities are taken within teams and occur when pair-programming, or when merely asking questions.

Often vital knowledge sharing moments happened in passing. When programmers were stuck with using a library, framework or even learning a new language nuance, they appreciated the ability to turn to the person next to them and get help. *"I like that people come to me when they need help. I think that is the best way for everyone to learn, even me!" - P3 "...we've got a really flat structure, if I needed to I could pretty much directly ping someone..." - P10*

Participant *P8* declared that the open communication streams between people across business and experiences made it easier to obtain answers. Walking between desks to other teams and also sending anyone email regardless of hierarchy helped facilitate these exchanges. These exchanges were often more attainable in smaller organisations where individuals were more familiar with each other *P5, P8, P9."It's a lot easier when there are only seven people in my company." - P8*

This is harder to maintain in larger organisations. *"You don't do technical inductions which is quite irritating because you do have to know people and you do have to find people who know stuff" - P11 "I think bigger companies have very secure processes, and it's very need-to-know and there is a very clear divide, and no one is afraid to say that it's above your clearance. In smaller companies the lines are a little bit blurred on clearance...I think [at] the size of that is fine because there is communication...I think security is far more approachable in a smaller company; it seems far more reasonable and accessible." - P8*

B. Biases and attitudes

Participants had strong feelings about how security practices aligned with their work, but different people had different feelings. Those who were strictly developers *[P4, P5, P7]* typically found security concerns a deterrent to completing their work on time. *"I believe that tasks kind of [get] pushed back a month or two purely because security teams get quite busy on an ad-hoc basis and hiding something, removing something, [adding something], can understandably be a bigger thing, even if it is one manager asking for access...Now we are aware of the lead times that those kind of security tasks can take and so we do plan for those, but I'll also probably say a lot of those times we don't plan enough" - P7*

However, others with broader responsibilities had a more positive attitude towards managing security. *"...It is going to take time, my point was we can manage it nicely without confusing people" - P1 "...when a change happens that they don't like, there have been people who have straight up left and quit their jobs...their empire is DESTROYED!" - P11 "For me, I know that it is important so I'm willing to compromise on that on a personal level, but I can get where that someone that it is not their thing, they can see it as a hindrance" - P12*

One participant *P11* acknowledged the personal relationships within the workplace as contributing to biases. In particular, They referred to *"tech bro-culture"*. They gave the example of a vendor-client relationship that they had witnessed where the vendor accepted any requests into production because of the close friendship with the client. The client had not given any comments and had caused the build pipeline to fail many times, and it was still was not working as the vendor continued to approve changes. This *"tech bro-culture"* also makes people have a view that asking questions makes you seem lesser, an outsider, and this discouraged necessary communication.

C. Experience

Experience pertains to the level of expertise an individual has in the industry. This was not only measured by the number of years that participants had worked, but also whether they had worked in a range of organisations and projects. One participant *P11* is an example of this. They only have 2-5 years of experience in the industry but have worked in multiple government and private organisations during this time. They regularly advise the senior members of their team. *"...Which is a strange dynamic because they're both seniors...Yeah, I'm a lot younger, I'm the only woman and I have a lot more experience than they do in the specific stuff that we're working in right now. And a lot of where my skill-set comes in is like picking up things up quickly because I have moved around a lot, I have done a whole bunch of random stuff" - P11.*

Almost all the participants identified significant differences in how less experienced and more experienced individuals deal with security practices. Often less experienced team members are wanting and willing to learn, but still are lacking in the ability to identify critical threats and risks which a more experienced team member is more versed at doing. *"New team*

members, I find, that they're kinda charged-up, ready-to-go, and that they want to prove themselves. I do find the more experienced people are a little more humble and they're a little bit more set-back... it's a bit more of a different energy-vibe; you see the new people come in, so ready to learn and they want to do security and then you get the people who have been there for ages like oh yeah that's easy and just do it." - P6

V. Organisational Factors

This category involves the influence of organisation structure and practices to security. The elements within this category are the parts of an organisation which affected programmer choices: in particular, project management techniques, and security training techniques.

A. Project Management Techniques

Project management techniques get set by the choices of more senior people in an organisation [13], [15]. Participants indicated these project management techniques as affecting how they approached security in their work. The three project management techniques mentioned specifically were "waterfall", Agile (Kanban and Scrum), and DevOps.

Only one participant *P10* said they used a waterfall methodology, while other participants *[P3, P4, P5, P7, P8, P11, P12, P13, P15]* referred to using either Agile or DevOps. When managed correctly, the techniques benefited participants a lot. It was regarded as more comfortable getting the security teams involved throughout the process in a more DevOps approach, and with an Agile approach each deliverable was checked at the end of each sprint, but strict management styles did not get enforced. *"It is a major problem with the waterfall type projects.... it's slowly going out, but very slowly, and they've left a lot of things behind, or rather forgotten to update a bunch of processes to match this stuff... Most places that I've worked, we do Agile, we do DevOps, but [also] not really. It's a whole thing." - P11*

A traditional waterfall-type management leaks through however, and some participants felt that there is an emphasis on security at the beginning and end of projects rather than throughout. This was done as a means to reduce *"tech-debt" - P7*, but does the opposite when programmers struggle to fix multiple issues at the end. *"I think that it takes the load off developers so developers can just focus on building [a] good application. I think the set-up right now [by having a separate security team] is pretty good right now. I just say that it is annoying putting the request through and them being slow" - P10*

Participants with more senior roles *[P3, P4]* did not like the wave of DevOps management; they identified it as being too time-consuming or just a fad. The feelings to do with security were different as one was a developer *P4* and one was a security architect *P3*. The latter's critique also stemmed from a hatred of buzz-words when they mentioned DevSecOps, a more security-specific iteration of DevOps project management. *"It's just annoying and gets in the way of my work" - P4 "...I don't like it. No one actually follows anything anyway, and an ongoing security approach should be there already..." - P3*

The project management techniques also influenced how teams interacted with each other. With a DevOps approach, the support from the security team was continuous, and talking between teams occurred more often. Often a security person was fully involved in the entire project; consequently, this increased collaboration meant that security issues were being identified earlier, not just near deadlines or after completion. *"The cross-functional teams, is how I've kind of been told to refer to them, they're really good... Anyway the point was, yeah, a lot of it is around your organisational structure and the way you form your teams. It's a major problem with like waterfall-type projects where you do like dev, and then you do test and then you do security at the end, but that's, it's slowly going out, but I mean like very slowly, and they've left a lot of things behind, or rather forgotten to update a bunch of processes to match this sort of stuff." - P11*

This does not often occur due to a lack of resources such as skilled employees and money as different business units charge for time. *"Ideally that [would] be awesome, but you have to pay all those people." - P12*

B. Security Training Techniques

Security training techniques and methods are chosen internally by the organisations. Most of the participants *[P1, P2, P3, P6, P7, P9, P10, P11, P12, P13, P14, P15]* stated that they had not obtained much prior exposure to security education. This means the training that the organisations provide is vital in educating developers. *"I don't have any formal education, but I have worked in security roles in two other organisations" - P2 "The closest thing really is [learning about] concurrency [at uni] for the most part." - P5 "I had one security paper when I was studying" - P6 "No there were no security specific papers [courses], if anything, it might be a handful of lectures at uni" - P7*

Organisation training methods were quite traditional. They were more policy and protocol-related, and often employees have to do readings, watch videos or listen to talks in the office by either internal teams or external businesses. The training techniques were more informative to-do of what to watch out for rather than learning any practical mitigation techniques. *"There was some basic training done for certification... watching stupid videos like Kevin Mitnick" - P4*

Often organisations did not expect employees to know the policies and protocols, but are expected to know the technical programming-side or should pick those up themselves. *"It's kind of expected, we don't do any formal coding security training, we've got other general security training about data and procedures, but nothing like technical related... I've done, I've had some, listened to some talks throughout about general security risks. They talk about you know, the different areas that our company gets kind of attacked. More, of just of a FYI." - P7 "People assume that if you're coming into these roles you kind of understand this stuff. But I've seen it go so wrong." - P11*

Personal training involves employees searching for non-work organisation security training. This is encouraged in some organisations, and a budget is put aside for employees. Participants stated that they could ask to do their own training

at work using Linkedin videos or Pluralsight *[P10, P11, P12, P13]. "We get given a budget to go pick out what kind of training we want to do." - P6*

While the subject of the solo-learning videos is more technically focused, they are still theory-based. This is why most participants identified that they learn best from talking to other people in their teams as they can learn more technical skills while applying existing knowledge. *"I'm actually really lucky I work with some great guys. I got paired up with another guy and we sat through and scheduled out six weeks." - P6 "I just ask and someone will help me out and they'll teach me... Yeah, I learn lots when we're working on a chunk of code together [pair programming]" - P10 "I think the best training you can get is from your peers..." - P14*

There were two unique points brought up by participants from the sample group which show a gradual change in how organisations aim to educate programmers on security. Participant *P10* recalled the use of technical workshops with an external company being a great way to learn how to program more securely to protect applications. This method was a more active approach than what had been described by others. They stated it was similar to following live-coding during university lectures. *"I think at work actually, they also made us do a workshop on SQL injection attacks and stuff and they made us do a bunch of things and we actually had to follow along on our own computers and that's really the only way you learn is by doing." - P10*

The other unique point was corporate "Hackathons" as a new way of learning. These are events which the organisation runs for its employees. Two participants *[P13, P14]* talked positively about it as being fun and casual while also prompting people to learn concepts. *"You can't enforce it... when you do that you that you've lost people. Hackathons are optional and part of [good] culture". - P14*

The Hackathon topics would be centred around the organisation field (e.g. Fintech) in order to translate learnings from these sessions to their current work. *"We have our own in-house, we call it Hackathon, you know, programme and base-line programme where anybody [can join], it's a kind of mandatory training programme where everybody [will go through] - [they] will be given a kind of a training or walk through by one of our security team, team member[s] actually, to how it works, how does it work in reality... People will be given some level of platform information, that what [does] this Hackathon [mean] and it's a game, you know, game! Where can you show me where is [the] problem. Can you fix [the] problem? [Do] you think there is a problem here? Do you think [with this] given website, will you be able to hack some data from my, from the machine, you know? So it's all kind of doing, rather than just doing, a Powerpoint programme, a presentation, [it's] more getting your hands dirty and getting things done." - P13 "[I'm] fortunate enough to work in a company where there is a Hackathon every month somewhere in the world right. And, and it's amazing" - P14*

VI. Industry Trends

This category is about the influence of industry trends on programmers. Within this category are several elements that affect the security-related decisions which programmers make: trust in practices, industry-standards, and evolving technologies.

A. Trust in Practices

Trust in practices involves the faith that people put in technologies. There are two categories that emerged: enterprise and open-source. Enterprise solutions are paid licenses that provide ongoing support, while open-source is free and built by a community. This can mean that open-source products do not provide long-term support. Those who worked in financial tech and consultancy firms *[P3, P6, P7, P9, P11, P12, P13]* favoured enterprise tools and libraries. They felt these tools provided them with more security in protecting their assets.

Those participants did acknowledge that while enterprise was favoured over open-source due to more trust, there were times where open-source was needed when coming across a new problem without any enterprise solution *P12*. The participants also used open-source when they wanted to adapt anything to best suit their practices without breaching any legal agreements with enterprise solutions. *"We do prefer enterprise, but to avoid breaking any SLAs [service level agreements] sometimes we have to look towards other open-source alternatives. We don't prefer it, but have to do it." - P12*

Every participant stated that while the security team has to vet new software, they do not check libraries. This, in turn, gives programmers free rein over choice. The majority did not check the security of open-source libraries *[P1, P2, P3, P4, P5, P7, P8, P9, P10, P11, P12, P13, P5]*. One participant *P10* also stated that only the functionality of the library gets checked after use. Therefore, there is a trust in libraries being secure to use within an application, but as a community of people building open-source libraries, there are no guarantees. *"I honestly, like, I don't know, like, I'm really allowed to like code, and use the library and play around with it." - P10 "It's this really strange thing that I've seen a few times now around in places. They provisionally accept a lot of different pieces of open-source technology which is kind of scary because it's just based on someone vouching for this piece of software." - P11*

Only one participant said they checked open-source libraries before using them *P14*. They stated that open-source libraries are great to use because they have so many different people working on them at once, but it was naive not to check them for any discrepancies or issues, especially since so many are readily available which can make the choosing process overwhelming. This checking burdens programmers, and their ethics and a lack-of doing this can provide entryways to threats. *"... there are a certain giveaways which you can look in the code-base and footprints to actually see if the tools are leaning towards the good-side or bad-side... there are a few giveaways like the test coverage of a tool, linting in the tool. How many commits do you actually do? Do you write any footprint doc for it? How do you add a feature? Is it commented? Types, annotations." - P14*

The participants in government organisations or within smaller start-ups had to outsource a lot of the security work *[P1, P5, P8, P9, P15]*. Organisations do not have the resources to dedicate the time into this aspect of programming, and

consequently, they also do not robustly test any outputs of the vendors against security, only the functionality. They rely on a trust on vendors. *"I think it's mostly a resource limitation because we don't have that many developers... Yeah that's a no from me." - P5*

Participants stated they have a trusting relationship with their vendors. This relationship was acknowledged as not the best practice, and one participant *P11* noted that vendors and client both lie, so it was better to ask as many questions as possible. *"There is a lot of like - I'm not quite sure what the right word would be, but basically they try and front very differently. Like if you're a vendor you try and act like you're very, very competent and understand everything and are a pro of what you do because that's what you're getting paid for right? ... I've been in meetings where people have blatantly lied. Like vendors have blatantly lied about their experience, how their piece of technology works and like it happens and it's very hard to control it especially when the other-side of the table, often the client who you're dealing with as like a consultant isn't often very technical. So you'll try to explain something to them and they do not understand it, they don't get it... They will sometimes get it, but often they pretend that they understand and will just kind of put a stamp and move on or business will say budget and say no".*

Another participant mentioned that a data breach due to a vendor designed product was the catalyst for change within their organisation for hiring an internal software team as they did not trust future contractors. *"Around the time we started doing everything in-house." - P10*

B. Industry Standards

Industry recommendations and standards such as the OWASP (Open Web Application Security Project) Foundation, SOC (System and Organisation Controls) and ISO 9001 affected much of the processes of how programmers work with security [12], [19]. There are frequent changes in recommendations, and ongoing work involved in legal compliance and best-practice adherence.

Participants cited SOC reports and ISO compliance *[P2, P3, P4, P6, P7]* as being the key frameworks that get followed when coding. ISO publishes standards that have requirements (clauses) that organisations need to follow. SOC has some more adaptability where an organisation can meet the criteria in any way. Ultimately, they build trust and are mandates so that programs can be used externally by clients and other parties. *"It's so it can be used by clients or it's not allowed." - P3*

The clauses and criteria enforce transparent coding practices upon programmers such as encryption of their work and separation between applications, *"zero-trust architecture" - P1*. Programmers also follow the best-practices outlined by OWASP. This includes not only the actual code but also documentation. *"With the various compliance regimes that we're under, so there's ISO27001, PCI and some stuff for the government we have to demonstrate that we are following the processes." - P2*

There is pressure to keep on par with other similar people and companies in the industry. This is to still stay relevant in the area of expertise and especially for vendor firms to look appealing for their clients. *"We haven't [stuck] ourselves in any particular way. Whatever industry is responding [to] and whatever the new features and challenges are coming, we adopt it; we adopt as early as possible." - P13*

C. Evolving Technologies

Participants emphasised the emerging and evolving technologies in the industry as being a motivator in continually adapting security practices. Cloud technology was frequently bought up in the interviews. The very first participant suggested that in following interviews, we should focus on cloud in order to match the newer ways of working rather than just the old. This emergence of the cloud involves server and data migration to services like AWS (Amazon Web Services) and Azure (Microsoft). AWS and Azure are prevalent in the industry as they are the two cloud services approved by the government. The migration has been slower in NZ compared to the rest of the world due to data legally having to be stored on-shore within NZ. *"Any organisation private or public, they are putting extra [effort] and going out of the box to [move] onto cloud, and the cloud is totally different compared to the on-premises legacy infrastructure." - P1*

VII. Relationships between Categories

This section describes the relationships between each of the categories: trends inform organisations, organisations impact culture and then vice versa with culture influencing people in organisations (see Figure 1).

A. Trends Inform Organisations

Emergent trends in the industry heavily informed organisational practices. This relationship exhibits when standards (§VI-B) and newer technologies (§VI-C) arise. There is a rush for organisations to seem relevant, up-to-date or in compliance with the law.

When new technologies become widespread in the industry, organisations adopt them in their technology stack, as seen in the rapid rise of React and other JavaScript frameworks into widespread practice *[P3, P5, P7, P10, P13]*. *"Nowadays React is getting more popular, you know? We as a company have to grow as well and we can't [be] sitting on our past legacy code... We have to offer and we have to adapt those new, we call it as futuristic technology... It has a capacity to adapt new security principles - measures... We see a long lasting future rather than continuing with our legacy process. In terms of productivity, it is more convenient, it is more user-friendly, it is more responsive, it is more secure compared to our legacy. We can't avoid it... We have to keep progressing " - P13*

They can also become mandated by the organisation as with Jenkins for automation of component pipelines *P11*. The choice of that is because of new industry practice, not explicitly stated by compliance standards or by legislation, but the desire to match and be on-par with other companies.

Project management techniques are informed by industry trends as well. The phasing out of the traditional waterfall approach is due to the emergence of Agile, and now a movement towards DevOps. A consequence of the latter

methodology is DevSecOps as another trend. These changes to match the trends seem to be premature and confusing for many programmers. How do deal with these new management styles does not get explained well – but programmers have learned to accept this. *"DevSecOps is a new thing that's getting popular in Wellington." - P3*

There is more of a shift and understanding that there needs to be more technical programming support provided in workplaces (see V-B, with some budget put aside for this by companies). There is not a not strong trend yet, but as established companies *[P2, P10, P14]* have begun to do live-coding workshops and internal "Hackathons", this may further inform other organisations of the benefits of such security training techniques.

B. Organisations Impact Culture

Organisations impact the culture of employees, affecting the elements we identified earlier: knowledge sharing, biases and attitudes and experience.

Organisational structure impacts the ability to share knowledge. This is most clearly affected by the project management techniques in use. With a more DevOps approach, there is talking going on throughout the project life-cycle, which involves different teams. This is similarly identified in the Agile management styles as best practice, pulling in security team members to assess at the end of each sprint. Participants identified these two methodologies as being extremely helpful. *"I love DevOps, and what it kind of means to me is that we have these cross-functional teams and we make it that you're not throwing over a dead cat to Ops..." - P11*

These two approaches, as opposed to waterfall, also promote communication within teams as the constant assessing makes internal teams discuss whether their solutions are best-practice, and they share ways of changing. *"It's much easier to scale it back rather than catch it, what is it called? DevSecOps." - P12*

Biases and attitudes are impacted and shaped by an organisation as well. An organisation which fosters security involvement throughout a project, and also promotes ongoing security education for programmers, will be an organisation that finds people more willing to change and more optimistic about having their code going through multiple iterations of security. *P10* stated that they were happily making the waterfall approach work because they were unwilling to change despite the difficulties they bought up in being blocked by the security team for long periods. *"Developers can be focused on just building good applications. I think the set-up right now [by keeping teams different] is pretty good, I just say that it is just annoying trying to put requests through and them being slow, but that happens everywhere." - P10 P4* also was observed to have a distaste in dealing with security when programming as they do not get provided with ongoing training opportunities. *"They are aware of the lacking and they're looking to fix it." - P11*

Experience gets impacted by one aspect of an organisation, security training techniques. Theoretical security training, whether it be about policy and protocols, or lecture-style talks and LinkedIn videos, are a way of exposing employees to past and current threats to an organisation. As cases and how they can be managed is explained to employees (e.g. SQL injections), they can learn ways of mitigation and increase their knowledge. Increase in knowledge, however, does not increase people's experiences: *"I think new people kind of go into it kind of full-steam ahead, I think. I think it's something we have going around when we are in formal education and it's something we are vaguely aware of. But more experienced teams - more experienced members, definitely have those, like you asked, do you have something you've learned from, that you know stung you, and experienced people have those. And I think, new people, not that we are unaware of it, it hasn't happened to us and it doesn't seem quite as real. And obviously we are trying to mitigate it, but I think until you get to the point where you have that moment, it's not going to shake you to your core the way it really maybe should." - P8*

C. Culture Influences Organisations

While organisations impact culture, culture also influences organisations [3]. The critical distinction between the two is that impact is forcing change, while influence is more about persuasive norms. Culture does this to both project management techniques and security training techniques.

Biases and attitudes influenced project management techniques. As clear from our participants, interpretation varied for waterfall, Agile or DevOps. This related to various attitudes, biases, and experience. These consequently make it hard to involve security as an ongoing component of a project life-cycle. It is even difficult to make changes to organisational management regarding security as some people do not want any disruption in the current norm. *"I have worked with so many of these people who refuse to change, refuse to update, refuse to do things better, refuse to do anything new." - P11*

Biases and attitudes, paired with experience, do affect the way organisations implement security training techniques. Based on the participants within the sample group, the more experienced members did not participate in as frequent training sessions compared to the lesser experienced individuals. This provides the bias to organisations that experienced members will not benefit from the training, and some places did not provide it ongoing.

Attitudes in regards to security training techniques were also a significant influence to organisations with a participant noting that if fun events like Hackathons are enforced, people do not want to participate. *"Hackathons are, here's the thing, when you define a structure for people sometimes, sometimes people don't perform. If you define Hackathon [from] Tuesday till Friday and you're busy, then you've lost it. It needs to [fit with] the culture of the team. I'm fortunate enough to work in an organisation where it's a part of the culture. And Hackathons are optional. You can be the part or you can leave the part... It should be part of the culture, which means it should be part of your day-to-day chatters [that] you have , it should be encouraged by not only your colleagues, by, by your chain or command of people. It should be something which should be part of your daily discussions" - P14*

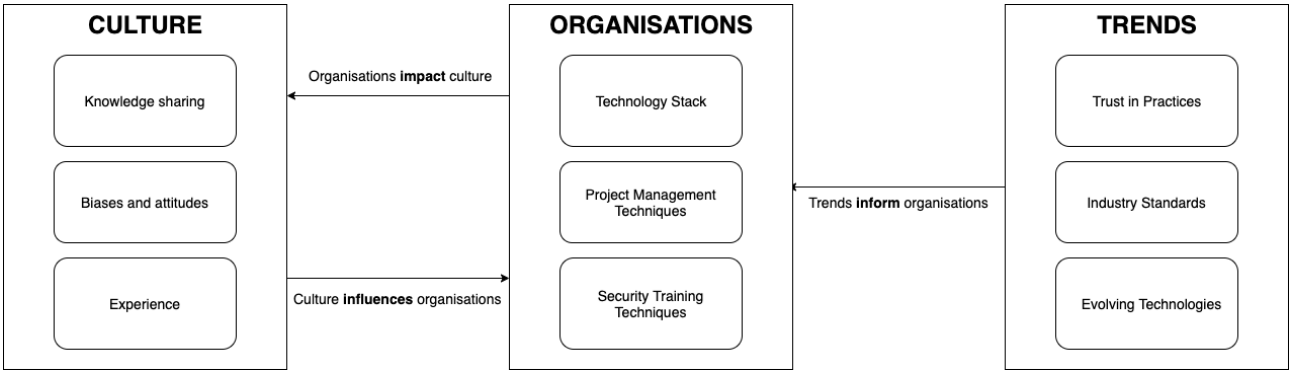

Fig. 1. A Theory of Influences on Security Practices. Three broad categories: Culture, Organisations, Trends, and the relationships between the categories.

VIII. Discussion

Our study set out to better understand programmer security practices. Using Grounded Theory, we identified how practices arose because of a set of influences. We found three broad categories of influences, each with several elements. Moreover, we found rich interactions between these influences. Our overall theory is described in Figure 1.

Organisational factors was found to be the core category as it was the central category which occurred frequently and was closely related to the other two categories and attributes [11]. A prior iteration of the diagram had shown one additional category, "Teams". However, as the constant comparison process occurred, the factors of teams seemed to be more-so attributed to organisations. This meant that this category was discarded and the factors merged into those of organisations. Another discarded category involved the influence of stakeholders. At the beginning of the interview process, clients and vendors seemed to have significant influence on the security practices of programmers, but as the analysis process developed, it was not as strongly referred to in the data. Even when prompted, the idea seemed irrelevant to many. Participants typically disclosed that they were not aware that security decisions were dependent on any particular stakeholders, and that set practices were already stipulated.

By understanding how the theory categories each relate to each other, organisations might implement changes in how they react to trends and how they deal with and motivate changes in culture. In particular, there needs to be reform in how security is taught in organisations. There should be more support for programmers in terms of technical education, instead of just general policy and practices. Technical education might be best done with frequent, optional internal Hackathons. Our participants described working on cases related to the nature of the organisation, with a mock scenario which to build up the experiences of employees in a safe environment. In an investigation done by researchers Pe-Than et al. [20] it was found that after completing a large scale corporate software Hackathons, participant's attitudes (based on five teams) had also changed to become more positive and confident about learning new skills by themselves.

There were several limitations with our study data. While we sought diversity in our sample, it was only 15 participants, and more work is necessary to explore generality. There was also a gender imbalance in the participants. One third were female while the rest male. It might be argued that this is indicative of the actual population. As women in the software development and security industry typically have different experiences to the majority, it would have been interesting to have more diverse data [22]. While it was interesting to obtain a diverse range of experiences within the sample group of participants, for future work, the investigation might benefit from reducing the scope of participants. By choosing one type of role, one sector or similar years of experience, the theory might include better understanding of the diverse perspectives that are involved in creating software.

Another difficulty that limited the theory when data collecting was the reticence to answer questions. For some participants, non-disclosure agreements (NDAs) were so strict that they could not answer questions such as, "what language do you use to program with at work?" This made it challenging to draw appropriate conclusions during the middle of this study. This could have also introduced slight biases in the collection as observational inferences were made based on the non-responses.

IX. Conclusions

This paper reported on a study to investigate programmer security attitudes and behaviour. Following a Grounded Theory research process, we developed the "Theory of Influences on Security Practices." This suggests an answer to our original overall question: Why Do Programmers Do What They Do?

We had started with three research questions about challenges, training, and adoption. In the process of the study, however, we realized there was a system of influences at work that informed all these issues in a complex way. Cultural factors affected knowledge sharing, biases and attitudes. Organisational factors included project management techniques and security training techniques. Industry trends affected trust in various practices, and evolving technologies also affected programmer practices. This emergent theory was drawn from newly gathered information, which is relevant to the topic and appears to be adaptable in the longer term. Overall, the findings suggest a need for more robust educational programmes in the workplace, a need for companies to reevaluate how they respond to trends, how they structure their organisations, and what they do to improve their security culture.

ACKNOWLEDGMENTS

This research was supported by a Victoria University of Wellington, University Research Fund (URF) Grant, #216549, User-Centred Secure Programming.

REFERENCES

[1] H. Assal and S. Chiasson, "'Think secure from the beginning': A Survey with Software Developers," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, 2019, p. 1–13.

[2] Aura Information Security, "Cyber security market research report," [Online]. Available: https://www.kordia.co.nz/aura-cyber-security-market-research-2019, 2019, [Accessed May 25 2020].

[3] R. Biddle, A. Meier, M. Kropp, and C. Anslow, "Myagile: Sociological and cultural effects of agile on teams and their members," in *IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2018, pp. 73–76.

[4] L. F. Cranor and S. Garfinkel, *Security and Usability*. O'Reilly Media, 2005.

[5] Department of the Prime Minister and Cabinet, "New Zealand's cyber security strategy 2019," [Online]. Available: https://dpmc.govt.nz/sites/default/files/2019-07/Cyber%20Security%20Strategy.pdf, 2019, [Accessed May 25 2020].

[6] N. Forsgren and M. Kersten, "Devops metrics," *Communications of the ACM*, vol. 61, pp. 44–48, 03 2018.

[7] B. G. Glaser and A. L. Strauss, *Discovery of Grounded Theory : Strategies for Qualitative Research*. Chicago, USA: Aldine Publishing, 1967.

[8] M. Green and M. Smith, "Developers are not the enemy!: The need for usable security APIs," *IEEE Security Privacy*, vol. 14, no. 5, pp. 40–46, 2016.

[9] B. Hendrix, "Knowledge sharing: Definition & process," Available: https://study.com/academy/lesson/knowledge-sharing-definition-process.html, 2017, [Accessed October 20 2020].

[10] R. Hoda and J. Noble, "Becoming agile: A grounded theory of agile transitions in practice," in *Proceedings of the 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, p. 141–151.

[11] R. Hoda, J. Noble, and S. Marshall, "Grounded theory for geeks," in *Proceedings of the 18th Conference on Pattern Languages of Programs (PLoP)*. ACM, 2011.

[12] ISO.org, "ISO/IEC 27001 Information Security Management," [Online]. Available: https://www.iso.org/isoiec-27001-information-security.html, [Accessed October 06 2020].

[13] B. Julian, J. Noble, and C. Anslow, "Agile practices in practice: Towards a theory of agile adoption and process evolution," in *Agile Processes in Software Engineering and Extreme Programming (XP)*. Springer, 2019, pp. 3–18.

[14] D. Kirk and E. Tempero, "Software development practices in new zealand," in *Asia-Pacific Software Engineering Conference (APSEC)*, 2012, pp. 386–395.

[15] M. Kropp, A. Meier, C. Anslow, and R. Biddle, "Satisfaction, practices, and influences in agile software development," in *International Conference on Evaluation and Assessment in Software Engineering (EASE)*. ACM, 2018, p. 112–121.

[16] Ministry of Health, "Cyber security incident," [Online]. Available: https://www.health.govt.nz/our-work/emergency-management/cyber-security-incident, [Accessed May 31 2020].

[17] ——, "Update on tū ora cyber security incident at 8 october 2019," [Online]. Available: https://www.health.govt.nz/system/files/documents/pages/health_report_8_october_20191935_redacted.pdf, [Accessed May 31 2020].

[18] N. Newton, C. Anslow, and A. Drechsler, "Information security in agile software development projects: A critical success factor perspective," in *Proceedings of the 27th European Conference on Information Systems (ECIS)*. AIS, 2019.

[19] OWASP, "OWASP Secure Coding Practices," [Online]. Available: https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf, [Accessed May 25 2020].

[20] E. P. P. Pe-Than, A. Nolte, A. Filippova, C. Bird, S. Scallen, and J. Herbsleb, "Corporate hackathons, how and why? a multiple case study of motivation, projects proposal and selection, goal setting, coordination, and outcomes," *Human Computer Interaction*, pp. 1–33, 2020.

[21] P. Ralph, "ACM SIGSOFT Empirical Standards Released," *SIGSOFT Softw. Eng. Notes*, vol. 46, no. 1, p. 19, Feb. 2021.

[22] S. Sobieraj and N. C. Krämer, "The Impacts of Gender and Subject on Experience of Competence and Autonomy in STEM," *Frontiers in Psychology*, vol. 10, 2019.

[23] M. Tahaei and K. Vaniea, "A Survey on Developer-Centred Security," in *European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2019, pp. 129–138.

[24] C. Weir, I. Becker, J. Noble, L. Blair, M. A. Sasse, and A. Rashid, "Interventions for long-term software security: Creating a lightweight program of assurance techniques for developers," *Software: Practice and Experience*, vol. 50, no. 3, pp. 275–298, 2020.

[25] C. Weir, J. Noble, and A. Rashid, "Challenging Software Developers: Dialectic as a Foundation for Security Assurance Techniques," *Journal of Cybersecurity*, 04 2020.