Panel Summary: The Future of Software Regulation

Benjamin Edwards
University of New Mexico
Dept. of Computer Science
MSC01 1130
1 University of New Mexico
Albuquerque, NM 87131
bedwards@cs.unm.edu

Michael Locasto
University of Calgary
Dept. of Computer Science
602 ICT
2500 University Dr. NW
Calgary, Alberta T2N 1N4
locasto@ucalgary.ca

Jeremy Epstein
SRI International
1100 Wilson Blvd
Arlington, VA 22209
jeremy.epstein@sri.com

ABSTRACT

A panel at the New Security Paradigms Workshop (2014) discussed the topic of regulation and licensing of software developers and information security professionals. This included topics of the current state of certification, future possibilities, and challenges associated with new forms of regulation. This paper presents a brief background on the subject, three opinions presented by the panelists, and finally a summary of the discussion which occurred at the workshop, including input from both the panelists and the workshop attendees.

Categories and Subject Descriptors

H.1.1 [**Models and Principles**]: Systems and Information Theory—*Value of Information*; K.5.2 [**Legal Aspects of Computing**]: Government Issues—*Regulation*; K.7.3 [**The Computing Profession**]: Testing, Certification, and Licensing

General Terms

Security, Measurement, Legal Aspects

Keywords

regulation, licensing, ethics, software development, offensive security, credit score, public policy

1. INTRODUCTION

Increasingly there are calls for tighter regulation of the Internet [27], the introduction of "driver's licenses" for the Internet [9], and licensing of software developers. These calls are often made in response to significant cybersecurity events. In the last four years we have experienced numerous cyber-meltdowns: the 2011 summer of Lulz, failures of certificate authorities, failures in critical open source software [20], and the loss of massive amounts of personal and financial information [47]. In the wake of these events we expect a renewal of calls for schemes involving the licensing or regulation of the software development industry.

Certainly, regulation in meaningful forms would profoundly affect the construction and analysis of software, and that certain forms of regulation might have significant negative consequences for the relative ease with which bugs and broken systems can be studied. Strict regulation might mean a reduction in access to the ability to code and make the construction of software a much less open activity. We would agree that this is a poor outcome, but to conduct a fair assessment, such a negative consequence must be set against the perceived or anticipated increase in security that proponents claim might follow from various regulatory models. Moreover, there are many possible forms of regulation; we should withhold judgement until we understand the nature of this spectrum.

This panel sought to examine whether such regulation has any value for preventing or reducing the occurrence of software bugs that lead to compromised systems. If it is believed regulation can reduce such bugs, we considered exactly what forms of regulation are viable, and which should be avoided. Finally, we considered how best to cooperate with politicians, professional organizations, and developers to implement effective regulation.

2. PANEL FORMAT

The panel was presented as follows. The first author provided a brief introduction and motivation for the panel. Each of the three panelists then gave a brief statement (approximately 5 minutes), outlining a position concerning regulation of the software development industry.

Rather than specifically including two positions (FOR, AGAINST), the panel invited a more open discussion of the topic at hand. Participants at the workshop were prompted with a few questions.

1. Is regulation of the software industry inevitable?
2. If it is inevitable, on what timeline will regulations develop?
3. What are possible models for regulation?
4. What are some of the positive and negative outcomes for various modes of regulation?
5. Can regulation actually make software better, or do we have to settle for better protection of consumers?
6. What has prevented previous attempts at regulation to fail?

NSPW'14, September 15–18, 2014, Victoria, BC, Canada.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3062-6/14/09 ...$15.00.
http://dx.doi.org/10.1145/2683467.2683478.

7. Which organizations and individuals are most important to getting effective regulation implemented?

3. RELATED WORK

The idea of regulation in cyberspace is not unique. In this section we review proposals for regulation of cyberspace, the necessary information needed for effective regulation, as well as a review of regulation in other industries.

3.1 Software Industry

Data breaches often result in a flurry of calls for new regulations concerning data privacy. Following the ChoicePoint data breach in 2005 [41], in which Federal Trade Commission(FTC) forced the company to pay $15 million in civil penalties, discussions began about whether the software and data storage industry should be regulated or whether the free market simply needed more time to improve itself [61]. More recent breaches such as Target's loss of millions of consumer credit cards have also resulted in FTC investigations and lawsuits for companies responsible for data [47]. However, the effectiveness of these lawsuits has been questioned [51], and more robust laws have been suggested [52].

Bruce Schneier has written extensively on regulation of the software industry and advocates for a liability model [49]. This model was refined by Dan Geer and Poul-Henning Kamp to include three levels of liability. (0) In the case of damage caused by willful intent, the software company should be held responsible via a criminal code. (1) If the source code is provided with the software, then the provider is only liable for the cost of the software. In the case of open source software, the development cycle remains unchanged. (2) In all other cases, software companies are liable for any damage caused by their products [21].

Further refinement of who in the software development/ deployment chain should be liable has been discussed. Anderson et al. outline various options for software and systems liability, and ultimately recommend that network-connected equipment be 'secure by default' [2], with required vulnerability disclosure, and security patches kept separate from feature updates. A European Union working group further suggested that software vendors have upper and lower limited liability thresholds based on the potential harm or criticality of software [4]. Models for liability have examined the effectiveness of different liability policies, finding that their effectiveness can be highly variable depending on the probability of zero-day exploits, the cost of patch deployment, the nature of loss, and whether a software vendor holds a monopoly [3, 23].

The insurance industry is already taking note of the potential to protect against such liability, with the American International Group investigating how to assess cyber risks [54]. However, insurers have struggled to assess the risk associated with cyber insurance [57], and the cyber-insurance industry has failed to succeed despite theoretical work outlining its construction [5]. It is often difficult for insurers to assess the damage to a company's reputation that could result in lost sales, as was the case with the Target breach [60].

As a counterpoint Leveson advocates developing a science of computer security so that new regulations can be developed intelligently [26], using the development of high-pressure steam engines as an analogy. There have been many attempts at codifying the software development process, one of the earliest examples being the "Trusted Computer System Evaluation Criteria" [44]. Organizations, such as the Capability Maturity Model Integration Institute [11], have claimed to develop processes that reduce the number of bugs in software, as well as deliver it on time and under budget, with other organizations making similar claims [43, 7, 13]. Vinton Cerf, the past president of the ACM, has stated that a 'certified' or 'professional' designation is likely to occur in the near future [9]. A necessary component to professional certification is the need to teach new Computer Science majors the basics of cybersecurity [25]. However, Doctrow has argued that our limited understanding of how cyberspace will continue to shape society makes regulation difficult at this time [17].

An important component to designing effective regulations is knowing the risk of security vulnerability in a specific piece of software. Clark et. al found that most software enjoys a "honeymoon" period after release before the first vulnerability is found, and that legacy code reuse contributes to the number of vulnerabilities found and the rate at which they are discovered [10]. An examination of open source projects finds the change in the number and density of issues reported by source analysis correlates with future exploitable bugs, and that that the rate of exploitable bugs in software tends to drop 3 to 5 years after its initial release [18]. This suggests that the use of such static analysis tools might help regulators and developers alike identify at risk software. Other work indicates that files authored by independent groups were more likely to have vulnerabilities than those who had wide scrutiny [32]. However, when files were changed by numerous developers they were more likely to have vulnerabilities later. Having small groups responsible for maintaining code not only helps to identify liability but also lower the risk of vulnerabilities in this code.

3.2 Intellectual Property

The other area of cyberspace to be actively regulated is Intellectual Property. Proposals for how to manage intellectual property in an age where copying and distribution of media is increasingly cheap and easy are myriad. The Digital Millennium Copyright Act was pass in 1998 to prevent the circumvention of Digital Rights Management Software [33]. However, it has been argued that this particular law does not allow security researchers to locate flaws in current software [14]. Later attempts at protecting intellectual property such as the Stop Online Piracy Act(SOPA) and Protect IP Act(PIPA) were designed to help protect intellectual property. However they were widely regarded as written too broadly and likely to stifle legitimate free speech [29]. One view regards these broad copyright regulations as the ultimate stifler of innovation [28].

3.3 Other Industries

In different industries motivations for regulation are varied. Banking panics motivated regulation of the financial industry [8]. The codification of engineering as a profession came from a desire in the engineering community to more clearly define a leadership role in society [31]. The side benefit of this movement was the standardization of practices and knowledge engineers required. More general labor laws, like those protecting workers and enforced by the Occupational Safety and Health Act of 1970, arose after the realization that billions of dollars were being lost due to on the job deaths and injuries [30]. Medical practice has always fallen under particular legal scrutiny due to its obvious risks to public health. In the United States the practice of medicine (and who is able to do so) has been regulated since colonial times [46]. The potential vulnerability of individuals in the legal system has generally led to the self-regulation of the practice law with bar associations and courts working closely together to define who can and cannot practice law [16].

This diversity in origin of regulation in different professions has led to different methods of certification of individuals in the professions. In the United States, engineers first have to graduate from an accredited university, and pass an exam. After this individuals need to accumulate engineering experience over several years before being able to complete a Principles and Practice in Engineering exam, certifying them as a professional engineer [35]. This type of apprenticeship where work is required before certification can be achieved is still practiced in many technical professions in the United States such as electricians, iron workers, plasterers, plumbers and bricklayers [58].

In contrast, the ability to practice law or medicine usually only requires graduating from an accredited university with the appropriate degree and passing a qualifying exam. The details of the process vary from country to country and within states, and specialized programs exist to maximize the likelihood of passing legal or medical boards.

It is not clear however that professional certification benefits individuals in all cases. Hashimoto found no difference in outcomes when individuals represented themselves (*pro se*) vs having criminal defense attorney [22]. At the time of this writing there is no work indicating that unlicensed contractors have higher accident rates[1]. Licensed contractors are generally required to carry insurance, protecting homeowners against on the job accidents [39].

4. PANEL POSITION: DON'T REGULATE OFFENSIVE SECURITY[2]

From time to time the suggestion arises that software engineers or developers should be subject to a licensing or regulation scheme [9]. Such suggestions often occur in the aftermath of widely publicized failures of security controls or systems or in the wake of a significant data breach. Recent significant breaches include RSA [48], Target [47] and Heartbleed [20], and we are now seeing a renewal of calls for schemes involving licensing or regulation of developers.

Other professions are held up as models – e.g., if it works for civil engineers to be professionally certified, why can't it work for software engineers? We also consider the difficulty of translating certification frameworks into the real world. This position statement rejects the notion that people should be certified or that "offensive security" activities should be controlled [55]. Instead, the onus should be on showing software is secure, not on limiting activity and technology typically associated with offensive activities.

4.1 Why Does the Call for Regulation Arise?

Periodic calls for tighter regulation of the Internet include the introduction of "driver's licenses" for the Internet, and licensing of software developers raise the question of just what the cost and value of such regulation might be. Such calls are often made in response to significant cybersecurity events. It has been almost four years since Brian Snow predicted a significant security meltdown [50] (December 2010), and in that time, we have experienced any number of cyber-meltdowns, from the 2011 summer of Lulz to failures of certificate authorities to the Heartbleed bug.

Each of these events is no longer a small technical matter affecting a piece of software used by only a few individuals or companies. Most major events over the past few years have affected ordinary people with no special technical or Computer Science background. Calls from these people for a solution to the problem of computer trustworthiness will only grow louder.

But who might be held responsible for such a breach? Certainly RSA bears the majority of the responsibility for their breach [48], but Adobe could conceivably be held liable for releasing software with this particular exploit. At a finer grained level, if the particular developer who introduced the flaw into the software could be identified, could she be held liable? What about the software architect who created the specification, the QA or tester who missed the bug, or the manager who failed to expose it during a code review?

For those outside the field of security or software engineering, a seemingly "natural" place to find a solution to "guarantee" trustworthiness is in the forced professionalization of software developers. Forced professionalization is but the tip of the legislative iceberg; self–interested parties sometimes call for the deep regulation of the Internet or the outright banning of new technology [28].

4.2 Calls for Regulation Are Just an Admission of Being out of Ideas

Various forms of certification and regulation exist for certain types of software development activities. Existing regulation, certification, and validation frameworks, guidelines, and standards seem to work within narrow circumstances. Recently, there has been a renewed call to somehow regulate the production of software (e.g., by licensing developers) as a way to reduce the number of flaws and vulnerabilities in important computing systems (e.g., cyber-physical or critical information or industrial control systems) [9]. In some cases, such calls are phrased as warnings, but increasingly senior and respected Computer Scientists are openly supporting such suggestions and efforts. One wonders whether such support is a tacit admission of defeat – after all, the security community has a long history of certification and validation practice, and such practices have not solved the problem of insecure software. Why should this time around be any better?

4.3 Regulation Should Not Focus on Offensive Security

One exceedingly dangerous outcome of such regulation crusades is the risk that such efforts will seek to regulate or prohibit the practice of offensive security. Certain forms of regulation might have significant negative consequences for the relative ease with which bugs and broken systems can be studied. In particular, there is a temptation to regulate tools of the trade, mostly exploits [19]. Using regulation or legal means to limit access to or the activity of "hackers" or "hacking" tools is a mistake. The panel position

[1] This is conspicuous as licensed contractors might have an interest making that information known.

[2] This section was authored by Michael Locasto

statement argues that the focus of regulation should be on certifying software correct, not in limiting offensive security techniques, software, or frameworks.

4.4 Regulating Tools and Research is a Bad Idea

One way to produce more secure software is to try to make developers better and thereby reduce the presence of flaws and vulnerabilities. Another method to reduce security incidents is to attempt to control "hacking" or offensive security tools and practices — after all, if we can "force" developers to be "better" through legal means, why can't we force blackhats to behave better by simply outlawing what they do? We consider such attempts misguided and the logic absurd.

I will highlight two recent developments that illustrate the nature of the threat to security research. The first is an article by Stockton and Golabek-Goldman that strongly recommends aggressive control of offensive security activities [55]:

> Third, the United States should amend the Computer Fraud and Abuse Act to strengthen its ability to prosecute researchers located both domestically and abroad who recklessly sell dangerous exploits targeting critical infrastructure to America's adversaries.

The second development centers around proposed modifications to the Wassenaar Arrangement to control "intrusion software" [38]. In such a proposal, we see the early fruits of our discipline and technical terminology being misunderstood by lawyers and policy makers. Bratus et al. [6] argue that as a community, we must establish definitions before they are established for us:

> Offensive security – or, in plain English, the practice of *exploitation* has greatly enhanced our understanding of what it means for computers to be trustworthy. Having grown from hacker conventions that fit into a single room into a distinct engineering discipline in all but the name, offensive computing has so far been content with a jargon and an informal "hacker curriculum." Now that it is unmistakably an industry, and an engineering specialization, it faces the challenge of defining itself as one, in a language that is understood beyond its own confines-most importantly, by makers of law and policy. [6]

4.5 Discussion Questions

1. **Threat to the Discipline:** Are regulation and developer certification a threat to Computer Science?

2. **Terminology:** Terminology is important, but many among us still don't use terms like "exploit" or "zero-day" correctly or consistently. What is the best approach to developing a sensible terminology that can be communicated to policy makers?

3. **Measuring Security:** Are calls to regulate development simply shallow and transparent ways to solve the hard problem of security measurement, and represent a dangerous shift in focus from software to people?

4. **LangSec Compliance** Should regulation schemes be aimed squarely at software artifacts and how they can show that a development process was followed that formally eliminates certain classes of bugs? The LangSec (`http://langsec.org`) philosophy is one example of this.

5. **Other Considerations:** Is secure software construction now desperately in need of professional licensing as the cure for its ills? Will that cure be worse than the disease? What are the possible models we can derive from trends toward guilds, professionalization, or regulation in other disciplines? How does this square with the National Academies saying that cybersecurity is an occupation, not a profession [45]?

4.6 Conclusion

The interaction of legislation and computer and information technology is an evolving morass of largely unintelligible and sometimes unfair legal consequences. The urge to regulate behavior is a natural "public" reaction to security meltdowns like RSA, Target, and Heartbleed. Given a public that is largely ignorant or uninformed of the actual complexities of software development and software security, the security and software engineering communities will continue to face this kind of pressure, and have a significant stake in educating their legal and political peers. Many of us are academics, and value a free and open exchange of ideas and information. Without defending this ground, we may find it already gone from beneath our feet.

The information security community has seen the elasticity applied to what is "lawful" by overzealous prosecutors and uninformed public officials. There are lessons for technical people to learn – namely, how to cope with the slow and uninformed grind of the legislative process and the justice system. Lawyers and legislators feel a deep need to apply legal solutions to a technical problem. The position of this statement is that our community has to fight very strongly against this urge.

5. PANEL POSITION: A CREDIT SCORE FOR DEVELOPERS, SOFTWARE, AND VENDORS[3]

Discussion of major cyber security events is no longer confined to tech blogs and security mailing lists. Target's leak of millions of customer credit card numbers [47], and the well marketed Heartbleed bug [20], made major national headlines. Following the Target breach, there were serious economic consequences for both the company and consumers [60]. To help avoid disasters like this, it has been suggested that software development should be subject to licensing or regulation [9]. It won't be long before it isn't just industry insiders calling for regulation, but the wider public who will be increasingly affected by future security events. Regulation of the use and creation of software is inevitable.

If regulation is inevitable, what should it look like? The unique nature of software development will likely require a unique solution. New attacks are developed daily, while old attacks are still viable in new software. The number of vulnerabilities reported in the National Vulnerabilities

[3] This section was authored by Benjamin Edwards

Database has increased from 102 a month in early 2002 to 506 a month in the summer of 2014 [37]. The constant arms race between security professionals and attackers will result in new languages, platforms, and techniques leading to a dynamic and complex security landscape. In such an environment, classic models of certifying professionals will be insufficient, and insurance agencies will struggle to establish premiums when risk is unknown.

Here some possible models for the regulation of software development are outline. I will focus on one particular model, which develops a credit score for developers, products and software vendors. This model is flexible enough to address the dynamic nature of the software development industry, while providing valuable feedback to companies, developers, and users. However, like all regulation there are many challenges, both technical and social, to bringing this model to fruition, and I will detail some obvious objections.

5.1 Regulatory Models

In this section I introduce models which represent a spectrum of ways the software development industry might be regulated. While some are outlandish, they provide a lens with which to view possibilities for regulation. Next, I briefly summarize a more realistic model for regulation of software development advocated by security experts, detailing some possible flaws and roadblocks.

5.1.1 The Regulatory Spectrum

While somewhat hyperbolic in nature these regulatory models provide a spectrum of ways to regulate the software development industry. One could certainly imagine intermediary models, or borrowing some features of one for another.

Stallman's Paradise: Under this model all software must be free and all binaries distributed with source code. Patents cease to exist. Developers are not licensed, as this interferes with the free exchange of ideas and code. All hardware is distributed with a full and transparent specification. The burden of verification rests on the users of software or their appointed agents. Most people must therefore be code-literate. This regulation scenario is the most feasible to "enforce" in some way, but it is also most likely to lead to the collapse of profitability of the software market.

Draconian Ground-up Digital Signing: A hybrid technical and policy solution mandated by legislation, the production of each software bit must be carefully tracked and accounted for. Computing infrastructure must be constructed to support near 100% attribution of functionality and information flow. Government mandates the forced registration, licensing, and repeated testing of software developers. Developers likely lose their license according to some schedule of offenses or bug rates. Hobbyist software is either banned or of quasi-legal status. Significant legal tests as to whether code is free speech arise; even if it is, the use of such speech in commercial products deemed important to critical infrastructure may be outlawed, taxed, or subject to significant fees.

Developer Licensing: An alternative to government-run registration and licensing is government support for professional societies or unions that enforce their own professional codes of conduct and licensing. One likely expression of this form of regulation is to produce a guild model or apprentice-like model for developers. Various forms of "computer systems failure" insurance may or may not be involved or mandated. General access to such "approved" software development is largely unavailable or involves significant wait times.

US Department of Software Development: An obvious approach to managing the above models would be to create a government agency who would be entrusted enforcing the laws. As appealing as such an agency might be to politicians, such a blunt and limited solution has significant shortcomings. Its task is hopelessly complex and might be easily circumvented without other legal and economic support. The agency itself presents a rather attractive target and if it chose to involve itself in the certification of software (like Apple does), would rapidly be overwhelmed without a mechanism to discourage or penalize the creation and submission of software for review. Such an agency might have to contend with copyright law: how could the government make itself party to all code, which is a written expression of ideas protected by copyright? Probably most damning, such an agency would probably do little to improve software security or quality. Registration without review provides a false sense of security, and registration with review means an explosion of government employees or a deep backlog of cases akin to the US Patent process. Having the agency sign software to indicate registration of the software means nothing in terms of quality or absence of bugs.

Even licensing of developers will likely have little impact on software security. CGI Federal, the lead contractor at Healthcare.gov, has been awarded the highest possible Capability Maturity Model Integration(CMMI) level for development certification. CMMI purports to improve the software development process by helping companies deliver software which meets requirements, stays on budget, and is delivered on time [11]. Despite this, on the date of the rollout of healthcare.gov only 30-40% of the system had been built [56]. CMMI appraisal seems to have little value except to the CMMI corporation.

5.1.2 Software Liability

Bruce Schneier has written extensively on regulation of the software industry and advocates for a liability model [49]. By forcing software companies to be liable for defects in their products, Schneier believes that software quality will increase. Moreover, this will encourage the insurance industry to provide security insurance. This model was refined by Dan Geer and Poul-Henning Kamp to include three levels liability as outlined in subsection 3.1.

Other refinements for software liability have been suggested [5, 2, 3, 23, 4]. These works suggest various types of liability, applied to various parties within the software deployment/development chain. The insurance industry is already taking note of the potential to protect against such liability, with the American International Group investigating how to asses cyber risks [54]. However, insurers have struggled asses the risk associated with cyber insurance [57], and the cyber-insurance industry has failed to succeed despite theoretical work outlining its construction [5]. It is often difficult for insurers to assess the damage to a companies reputation that could result in lost sales, as was the case with the Target breach[60]. Moreover, as we noted in the previous section, certifications don't seem to decrease the probability of a dangerous vulnerability being released within a piece of software. Even if best practices are followed, new types of attacks, such as the MD5 collision attack utilized in the FLAME malware [53], would be nearly impossible to predict.

There are some broad estimates of the cost of bugs. A 2002 NIST report estimates that only one third of the cost of bugs can be eliminated by improved testing infrastructure. If we do a back of the envelope calculation it could mean a liability of roughly $200 billion for US companies today[4].

It is important for insurance companies to appropriately determine what fraction of this $186.5 billion should be covered by which companies. Over price insurance and this may stifle innovation, preventing small companies from being able to to enter the market without devoting a significant portion of capital towards insurance. Under price insurance and the market will collapse. Insurance companies need an objective way to measure the quality of developers, software and company to help appropriately price cyber insurance.

5.2 Credit Score for Developers

In this section I outline a measure of software quality. This model proposes the creation of a developer ratings bureau which is able to provide scores to developers, software products, and vendors. Similar to the way credit bureaus in the United States provide FICO scores to individual consumers rating their creditworthiness, a private or public (or some combination thereof) organization would provide scores to developers based on their past development history.

5.2.1 Registration and Tracking of Developers

In such a scheme, developers, projects or vendors register with a bureau. After a vendor releases a new product to market two pieces of information would be reported to the bureau (1) information on the amount of code produced by each registered developer and (2) bugs and vulnerabilities linked back to specific developers through commit histories in version control systems. The more bug free code a developer writes, the higher their score. Bugs and vulnerabilities that are found in released products negatively affect the score. The score could be weighted by the severity of the bug, or its ability to cause financial (or even physical damage). Like a FICO score, a developers score could increase with time and the more bug free code written, and past mistakes would hold less weight as time goes on. The bureau could further aggregate scores for individual pieces of software, or the entire software companies. The constant stream of bugs and vulnerabilities would likely provide a rich dataset for the creation a score, and modern version control systems provide a link between bugs and developers.

While simple in structure, this solves some of the problems in the liability scheme. Rather than providing a theoretical set of standards by which the company operates, as CMMI does, this provides a measurable, historical basis by which to rate developers, projects, and software companies. As new attacks merge and best practices evolve, developers would be required to improve their knowledge or suffer the risk of writing buggy code and doing damage to their scores.

Scores could provide insurance companies with a more concrete basis on which to assess risk and price their products accordingly. Because the ratings bureau is not directly involved in the development process, there are no questions of copyright ownership in the case of registered software.

Mechanisms would need to be put in place to avoid having companies attempt to inflate the scores of their products or developers. For example, such a system would create an incentive for companies to under report bugs in order to boost their, and their products' score. Bug reports and tracking systems would need to be required to be public, with the bureau cross referencing the public reports, with the private reporting of the company. Random audits examining commit histories of products could also provide spot checks of a software companies honesty.

Further consideration would have to be given to the open source community, and hobbyist developers. The most tenable solution would be to borrow from the Geer/Kamp model, and provision that if software is provided with the source code, registration and reporting with the bureau is optional. In all other cases it would be required. Open source developers and projects may decide to voluntarily register however, as this would provide a method for developers to build their scores outside of the context of industry.

The bureau itself could be a profitable venture. Software companies could buy reports from the bureau on potential new hires to evaluate their abilities. Companies seeking to contract with a software vendor would be able to purchase reports on the company and its products to assess not only who could provide the lowest bid for a product, but who would be most likely to produce a working product.

Despite these advantages, there would likely be issues with implementing such a bureau. Deciding the criticality of bugs and the actual construction of the score is an open question. Moreover, is there even enough variance in the ability of developers make a score meaningful? If critical security bugs have a large, lasting negative affect on a developer's score, it might dissuade individuals from working on critical infrastructure products. A single mistake could have a devastating impact on a developers score and therefor career. While bug reports and commit histories make it theoretically possible to link bugs back to specific developers, in practice this could be a complex undertaking.

5.3 Questions and Conclusions

Here I have given a brief overview of a few schemes for regulating the software development industry. These are prototypes, and will require significant effort to implement. However, I believe the unique nature of software development will require a unique solution, and developer scores may provide a simple basis for which to begin to improve the software development.

5.4 Questions

1. What are advantages and disadvantages in the schemes above that are not described?
2. Are there other schemes that should reasonably be considered?
3. Are any of these schemes actually tenable?
4. What would be the best ownership for the bureau, public, private or some sort of partnership?

[4] The NIST reported bugs cost US companies $59.5 billion in 2002. In 2014 there are roughly 5 times as many bugs, if two thirds of these are addressable through testing, this works out to $\frac{2}{3} * 5 * \$59.5 \approx \$200$

6. PANEL POSITION: SOFTWARE REGULATION MUST CONSIDER PUBLIC POLICY[5]

Issues of software development regulation to improve security have been bouncing around for decades. Many of the early efforts in security standardization (e.g., Trusted Computer System Evaluation Criteria [44], aka Orange Book) include elements of programmer regulation.

The focus of regulation in recent years has divided into several categories:

- Regulate the programmers. Testing, certification, licensure, etc. are common methods, including licensures such as $(\mathrm{ISC})^2$ CSSLP [42].

- Regulate the employers. Companies must obtain some form of corporate certification based on processes. Variations on the Software Engineering Institute's Capability Maturity Model (SEI-CMM) [11, 34, 36] and ISO 9001 have been proposed, as well as approaches such as OpenSAMM [43] and BSIMM [7] which focus on measurement but do not formally define requirements for organizations.

- Regulate the products. Regardless of who developed the products (primarily but not exclusively software), they must go through a certification process, such as Common Criteria [15] or PCI DSS [13].

The three are not mutually exclusive-for example, a product could be built by licensed employees employed by certified employers following a process, with the resulting systems being certified against a standard. Other fields, such as the medical profession, use a combination of all three: doctors, nurses, and other professionals are individually licensed and work for employers that go through a regulation process (e.g., hospital accreditation), with the "systems" (health care) being subject to monitoring and regulation for quality, cost, and other factors.

In the security field, methods of each of the three forms of regulation are, at this point, rudimentary. Software products are developed, almost exclusively, by developers with no independently verifiable qualifications, by companies that follow no independently verifiable processes, with the resulting systems having no verifiable quality characteristics.

Further, it is common practice that even when one (or more) of these characteristics are present, they are not accurately represented. Companies routinely obtain SEI-CMM certification for one part of the company, and advertise as if the entire company followed those practices. Similarly, a single version of a product may be certified as meeting the Common Criteria, but other versions of the product, or even other products made by the same company, may be implied to meet those requirements.

However, even if these problems are solved, effective regulation may never come to fruition, because any scheme developed by engineers and scientists that addresses the requirements that doesn't address public policy will simply sit on the shelf.

6.1 Public Policy and Software

Adopting any form of regulation will require regulatory and/or legislative actions. However, the vast majority of legislators and regulators do not have technical backgrounds, and will not understand the deep implications of regulation. In addition, they will be influenced by stakeholders, including:

- Software vendors who want to minimize their development and sales costs, and hence will oppose any move that requires licensed professionals (who can presumably demand higher salaries).

- Plaintiff's lawyers (e.g., for software purchasers) who will want to ensure they can pass the liability to the software vendors.

- Organizations that earn fees from issuing licenses, whether to individuals or organizations. This could include professional organizations like ACM and IEEE, quasi-professional organizations like $(\mathrm{ISC})^2$ and SANS, as well as organizations that provide product assessments (e.g., ICSA Labs).

Dan Geer argues [21] that regulation is inevitable. Laws are pending in some states to make licensure mandatory for developers of certain types of systems [24]. One of the key risks of regulation is that those writing the regulations won't understand what they are regulating. Combined with the above-named stakeholders goals, the risk is that regulations (especially licensing of software developers) will set a floor rather than a ceiling - to avoid killing the software industry, it will establish a low minimum, and there will be little motivation to go above those minimums.

As an example, some years ago I was named to a legislative commission on securing voting systems. I argued that they were vulnerable to many forms of attacks, to which a state legislator responded that I was wrong - since the source code is not available for the voting systems, they can't be attacked. This level of technical naiveté is unfortunately common in legislators, just as naiveté about the political process is common among computer scientists.

So how can any progress be made? Computer scientists, and security experts in particular, must not simply pontificate about what regulation and licensing should entail (i.e., the technical requirements). Rather, we need to spend at least as much time and effort working with elected officials to educate them about the limits of licensing and regulation as well as the benefits. And we need to learn from them what would motivate them to make the necessary legal and policy changes with the goal of improving security of the software we all rely on.

6.2 Questions

Addressing issues of regulation must include the following:

1. How will the "public interest" in regulation be defined, balancing security against other criteria such as cost and performance?

2. What organization(s) will be responsible for defining the regulations, and how will they ensure that the resulting regulation maximizes the public interest?

[5] This section was authored by Jeremy Epstein

3. How will any resulting standards be kept up to date? As both technology and attack methods evolve, standards will either be high level (and hence subject to interpretation) or low level (and hence obsolete before they are even adopted).

4. Who/what (individuals, organizations, or software products) will be grandfathered / grandmothered in by any new regulations?

5. What will happen with existing orphaned software (as defined by Geer, software for which there is no active development or maintenance)? Will someone who voluntarily picks it up become responsible for upgrading it to current standards?

6. Should "in house" software (i.e., developed by an organization for its own use) be subject to regulation, or only software developed by others? What if the in house software is developed under contract as custom software?

7. What methods are effective for educating the general public as to the values of software regulation?

8. What are the interactions with existing financial regulations (e.g., SEC regulations that require recognition of risks)?

7. PANEL SUMMARY

In this section we summarise the discussion that took place at the workshop. To the best of our ability, we report the opinions and conclusions that were expressed. The discussion focused primarily on the certification of security professionals and imposing personal liability on software developers.

7.1 Certification

The current state of certification, specifically Certified Information Systems Security Professional (CISSP) [12], has done little to reduce security incidents or the number of vulnerabilities present in software. While certifications such as CISSP can create a lower bound on knowledge, in practice certification does not guarantee real world experience or competence. The main motivation for current certification systems is the desire to satisfy customers that deliverables were produced by skilled individuals. The process to obtain the certification is difficult even for experienced security professionals, but a lack of certification may hinder the ability of professionals to access crucial data.

Given the failings of CISSP, it is not clear whether good alternatives exist. One possible reason for the failure of programs like CISSP to improve the security landscape is that, unlike other fields, there is no agreed upon body of knowledge which could be used as a foundation for certification. Other efforts to systematize software development knowledge, such as the Software Engineering Body of Knowledge(SWEBOK) [1], lack foundational information which might be used to develop a reasonable security professional certification. It may be the case that no such body of knowledge could be gathered given the universal nature of a computer and the emergent behavior of software. However, some persistent threats have been identified. For example, there has been little change in the top ten most critical web application security risks as ranked by the Open Web Application Security Project between 2010 and 2013 [40, 59]. These persistent threats, given the proper organization, could begin to form the basis of a type of certification.

As the field of computer security grows and adapts it may be necessary for certifications to be equally flexible. One possibility is to restrict the scope of certifications and offer different types of certifications for different subfields such as security operations, cryptography, web security or systems integration. Specifically, systems which require certifications in other engineering fields, e.g. medical devices and aviation, could be extended to software. In this way, software could be certified rather than the individuals who wrote it, developing a type of building codes for software. Formal methods may be able to help verify a particular piece of software is valid given the developers current knowledge.

These are promising suggestions, but there exists some barriers to their implementation. If new certifications are required in the United States for instance[6], this could create a sudden dearth of security professionals, and current professionals with other types of certifications would need to be grandfathered in. Such certifications would likely lengthen schedules and increase the cost of software development. This may drive development overseas where no certification is required. Even if better certifications are developed it is unlikely that such certifications will actually improve software quality. The political process of implementing such regulation could will be an onerous, but necessary task as mentioned in section 6. Possibly the easiest targets for such required certification may be at the state level; however, this could fragment the regulation landscape and delay a consensus of what is an acceptable certification.

7.2 Liability

Given the problems with certification, it may be prudent to impose personal liability on software developers in the same way doctors and lawyers are personally liable for their practice. However, personal liability in the software development industry has its own barriers. Medical malpractice often stems from the death or disabling of a single patient. In the security field, a single mistake could be much more far reaching, touching tens of millions of systems. Frequently developers have less control over their work than in other fields, with management imposing timelines in which no reasonably secure software could be made. For liability to be viable, developers would have to be given the power to postpone deployment of questionable software. Moreover, with such a rapidly changing field it is unclear whether developers should be liable for unanticipated flaws and new attacks. Holding developers liable for previously unknown attacks seems unreasonable. Software often integrates many different libraries which may not have been authored by a single individual, or may have no known author in the case of open source software. Flaws may emerge not from the software itself but from the integration of different pieces of software. Holding system integrators liable in the same way as the original developer would be necessary. This could make the assignment of liability following an incident difficult. However, courts frequently assign liability in other fields, and it is likely that after a significant body of law is developed the same could be done in the software development field. In this way, software would become one more portion of a failing system to which liability could be assigned.

[6] This is likely the best course of action.

8. CONCLUSIONS

While the panel itself provided a fruitful discussion, there seems to be little agreement on the best course of action to improve software quality through regulation. Two things are readily apparent however 1) as more people are touched by security incidents, calls for regulation will increase, and 2) it is crucial that as a community we need to act rather than let non-professionals impose regulation. This should either be done by working with policy makers directly to develop reasonable policy or through non-governmental professional action.

9. REFERENCES

[1] Abran, A., Bourque, P., Dupuis, R., and Moore, J. W. *Guide to the software engineering body of knowledge-SWEBOK*. IEEE Press, 2001.

[2] Anderson, R., Böhme, R., Clayton, R., and Moore, T. Security economics and the internal market. *Study commissioned by ENISA* (2008).

[3] August, T., and Tunca, T. I. Who should be responsible for software security? a comparative analysis of liability policies in network environments. *Management Science 57*, 5 (2011), 934–959.

[4] Böhme, R., Rath, M., Schneider, R., and Telang, R. Economics of security: Facing the challenges. *Study commissioned by ENISA* (2011).

[5] Böhme, R., and Schwartz, G. Modeling cyber-insurance: Towards a unifying framework. In *WEIS* (2010).

[6] Bratus, S., Arce, I., Locasto, M. E., and Zanero, S. Why Offensive Security Needs Engineering Textbooks: Or, How to Avoid a Replay of âĂIJCrypto WarsâĂİ in Security Research. *;login 39*, 4 (August 2014).

[7] BSIMM. Building security in maturity model. `http://www.bsimm.com/facts/`, Aug. 2014.

[8] Calomiris, C. W., and Gorton, G. The origins of banking panics: models, facts, and bank regulation. In *Financial markets and financial crises*. University of Chicago Press, 1991, pp. 109–174.

[9] Cerf, V. G. 'but officer, i was only programming at 100 lines per hour!'. *Communications of the ACM 56*, 7 (2013).

[10] Clark, S., Frei, S., Blaze, M., and Smith, J. Familiarity breeds contempt: The honeymoon effect and the role of legacy code in zero-day vulnerabilities. In *Proc. of ACSAC 2010* (2010), ACM, pp. 251–260.

[11] CMMI. Cmmi institute. `http://whatis.cmmiinstitute.com/what-is-cmmi`, Aug. 2014.

[12] Consortium, I. S. S. C. Cissp - certified information systems security professional. `https://www.isc2.org/cissp/default.aspx`.

[13] Council, P. S. S. Pci ssc data security standards overview. `https://www.pcisecuritystandards.org/security_standards/`, Aug. 2014.

[14] Craver, S., Wu, M., Liu, B., Stubblefield, A., Swartzlander, B., Wallach, D. S., Dean, D., and Felten, E. W. Reading between the lines: Lessons from the sdmi challenge. In *USENIX Security Symposium* (2001).

[15] Criteria, C. Common criteria for information technology security. `https://www.commoncriteriaportal.org`.

[16] Denckla, D. A. Nonlawyers and the unauthorized practice of law: An overview of the legal and ethical paramaters. *Fordham L. Rev. 67* (1998), 2581.

[17] Doctorow, C. Why it is not possible to regulate robots. `http://www.theguardian.com/technology/blog/2014/apr/02/why-it-is-not-possible-to-regulate-robots`, April 2014.

[18] Edwards, N., and Chen, L. An historical examination of open source releases and their vulnerabilities. In *Proc. of CCS 2012* (2012), ACM, pp. 183–194.

[19] Egelman, S., Herley, C., and van Oorschot, P. C. Markets for zero-day exploits: Ethics and implications. In *Proc. NSPW 2013* (New York, NY, USA, 2013), NSPW '13, ACM, pp. 41–46.

[20] Evans, P. Heartbleed bug: Rcmp asked revenue canada to delay news of sin thefts. `http://www.cbc.ca/news/business/1.2609192`, April 2014.

[21] Geer, D. Cybersecurity as realpolitik. `http://geer.tinho.net/geer.blackhat.6viii14.txt`, Aug. 2014.

[22] Hashimoto, E. Defending the right to self representation: an empirical look at the pro se felony defendant. *North Carolina Law Review 85*, 2 (2007), 423–487.

[23] Kim, B. C., Chen, P.-Y., and Mukhopadhyay, T. The effect of liability and patch release on software security: The monopoly case. *Production and Operations Management 20*, 4 (2011), 603–617.

[24] Laplante, P. A. Licensing professional software engineers: seize the opportunity. *Communications of the ACM 57*, 7 (2014), 38–40.

[25] Ledin Jr, G. The growing harm of not teaching malware. *Communications of the ACM 54*, 2 (2011), 32–34.

[26] Leveson, N. G. High-pressure steam engines and computer software. In *Proc. of ICSE 1992* (1992), ACM, pp. 2–14.

[27] Markoff, J. Taking the mystery out of web anonymity. *The New York Times* (July 2010).

[28] Masnick, M. Former copyright boss: New technology should be presumed illegal until congress says otherwise. `https://www.techdirt.com/blog/innovation/articles/20120927/00320920527/`, September 2012.

[29] McSherry, C. Sopa: Hollywood finally gets a chance to break the internet. `https://www.eff.org/deeplinks/2011/10/sopa-hollywood-finally-gets-chance-break-internet`, October 2011.

[30] Meeds, L. Legislative history of osha, a. *Gonz. L. Rev. 9* (1973), 327.

[31] Meiksins, P. The "revolt of the engineers" reconsidered. *Technology and Culture 29*, 2 (1988), pp. 219–246.

[32] Meneely, A., and Williams, L. Secure open source collaboration: an empirical study of linus' law. In *Proc. of ACSAC 2009* (2009), ACM, pp. 453–462.

[33] of America, U. S. Digital millenium copyright act. http://thomas.loc.gov/cgi-bin/toGPO/http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=105_cong_public_laws&docid=f:publ304.105.pdf, 1998.

[34] of Energy, D. Cybersecurity capability maturity model (c2m2). http://energy.gov/sites/prod/files/2014/03/f13/C2M2-v1-1_cor.pdf.

[35] of Examiners for Engineering, N. C., and Surveying. Continuing professional competency guidelines, Oct. 2010.

[36] of Standards, N. I., and Technology. Systems security engineering capability maturity model (sse-cmm), Apr. 1999.

[37] of Standards, N. I., and Technology. National vulnerability database. https://nvd.nist.gov/, Apr. 2014.

[38] Omanovic, E. International agreement reached controlling export of mass and intrusive surveillance technology. https://www.privacyinternational.org/blog/international-agreement-reached-controlling-export-of-mass-and-intrusive-surveillance, Dec. 2013.

[39] Orleans, L. U. N. Insurance requirements for contractors. http://finance.loyno.edu/risk/insurance-requirements-contractors, 2014.

[40] OWASP, T. Top 10–2010–the ten most critical web application security risks. *The Open Web Application Security Project* (2010).

[41] Pantesco, J. Ftc imposes record fine on choicepoint in data-loss case. http://jurist.law.pitt.edu/paperchase/2006/01/ftc-imposes-record-fine-on-choicepoint.php, Jan. 2006.

[42] Proffesional, C. S. S. L. https://www.isc2.org/csslp/Default.aspx.

[43] Project, O. W. A. S. Software assurance maturity model. https://www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model, Aug. 2014.

[44] Qiu, L., Zhang, Y., Wang, F., Kyung, M., and Mahajan, H. R. Trusted computer system evaluation criteria. In *National Computer Security Center* (1985), Citeseer.

[45] Ragan, S. Cybersecurity should be seen as an occupation, not a profession, report says. http://www.csoonline.com/article/2134002/security-awareness/cybersecurity-should-be-seen-as-an-occupation--not-a-profession--report-says.html, Sept. 2013.

[46] Richards, E. P. Police power and the regulation of medical practice: A historical review and guide for medical licensing board regulation of physicians in erisa-qualified managed care organizations, the. *Annals Health L. 8* (1999), 201.

[47] Risen, T. Ftc investigates target data breach. http://www.usnews.com/news/articles/2014/03/26/ftc-investigates-target-data-breach, March 2014.

[48] Rivner, U. Anatomy of an attack. http://blogs.rsa.com/anatomy-of-an-attack/, Apr. 2011.

[49] Schneier, B. Liability and security. https://www.schneier.com/crypto-gram-0204.html#6, Apr. 2002.

[50] Schneier, B. Brian snow sows cyber fears. https://www.schneier.com/blog/archives/2010/12/brian_snow_sows.html, Dec. 2010.

[51] Solove, D. J., and Hartzog, W. The ftc and the new common law of privacy. *Available at SSRN* (2013).

[52] Solove, D. J., and Hoofnagle, C. J. A model pregime of privacy protection (version 3.0). *Available at SSRN* (2013).

[53] Sotirov, A. Analyzing the md5 collision in flame. *Presentation at SummerCon, slides available at http://www. trailofbits. com/resources/flame-md5. pdf* (2012).

[54] Stanley, M. K. Aig: Cyber threats a top concern for industry execs. http://www.lifehealthpro.com/2013/02/07/aig-cyber-threats-a-top-concern-for-industry-execs, February 2013.

[55] Stockton, P., and Golabek-Goldman, M. Curbing the Market for Cyber Weapons. *Yale Law and Policy Review* (December 2013).

[56] Thibodeau, P. The firm behind healthcare.gov had topnotch credentials – and it didn't help. *ComputerWorld* (Oct. 2013).

[57] Thomas, L., and Finkle, J. Insurers struggle to get a grip on the burgeoning cyber risk market. *Reuters* (July 2014).

[58] Unwin, L., and Wellington, J. Reconstructing the work-based route: lessons from the modern apprenticeship. *The Vocational Aspect of Education 47*, 4 (1995), 337–352.

[59] Williams, J., and Wichers, D. Owasp top 10-2013 rcl-the ten most critical web application security risks. *The open wep application security project* (2013).

[60] Wolff, J. The $10 million deductible. *Slate* (June 2014).

[61] Zetter, K. The fight over cyber oversight. http://archive.wired.com/politics/security/news/2005/02/66632r, February 2005.