The Free Coding Manifesto

Adam Lucas Young

ayoung235@gmail.com

This manifesto is dedicated to Bantu Steve Biko 1946-1977.

Abstract: Working-class coders and vulnerability researchers the world over are subject to prior restraints on their speech imposed by the institutions they work for. The restraints are in the form of non-disclosure agreements (NDAs) and employment contracts that are typically enforced using a process called pre-publication censorship. Industrial pre-publication censorship chills contributions of source code to society and chills the publication of vulnerabilities found in code that has been given to society. This has a harmful effect on the depth, breadth, and information assurance of society's foundation of code. Restrictions on the human spirit call for new liberties to be defined and upheld. This manifesto defines Freedom A and Freedom B as follows. Freedom A: you have the freedom to write code and give it to society under conditions of your choosing. Freedom B: you have the freedom to write and publish, under conditions of your choosing, a critique or documentation of code that has been given to society. *Free coding* is defined as Freedom A and Freedom B. Obstructions to free coding are identified and measures are presented to uphold free coding. The measures presented include a proposed corporate policy that balances institutional equities with personal liberty, a software license term tailored after Freedom B, and an experimental free coding software license. Utilitarian, philosophical, and theological foundations of free coding are given. Obstructions to free coding form a subset of the problem of knowledge hoarding. I present my interpretations of the Book of Genesis, namely, the Original Command and the Original Paradox. I believe that these interpretations reveal the root of the problem of knowledge hoarding.

Contents

Published through the viXra.org e-Print archive Nov. 2, 2017.
viXra citation number: viXra:1711.0117. This version contains updates as of Feb. 15, 2018.
This manifesto was submitted to the Cornell University arXiv e-Print service on Oct. 29, 2017 under the category "Computers and Society," was assigned identifier submit/2052823, and was rejected by Cornell University on Nov. 15, 2017 by the moderators "who determined it to be on a topic not covered by arXiv." For the latest version go to www.coderfreedom.org.

Chapter 1

Utilitarian Foundation of Free Coding

Employment agreements and institutional policies and procedures, such as industrial pre-publication censorship, are intended to solve a tangible problem faced by the institution: preventing the exposure of critical business models, processes, data, patentable inventions, copyrighted material, and trade secrets. However, such controls, when overreaching, negatively impact the individual liberty of employed coders and vulnerability researchers. Overreaching censorship has a chilling effect on contributions of code to free/libre and open source software projects, thereby diminishing the depth and breadth of society's foundation of code. Similarly, it has a chilling effect on publishing vulnerabilities found in free/libre and open source software, thereby diminishing the information assurance of society's foundation of code and exacerbating vulnerability hoarding. This chapter addresses this problem by presenting methods to help balance industrial censorship controls with personal liberty. It shows the practical benefits of free coding to institutions and individuals alike, thereby establishing a utilitarian foundation of free coding.

1.1 Introduction

There is no question that for an institution to succeed it must be forward thinking, innovative, and stay ahead of the pack. It must succeed in a never-ending process of acquiring and retaining talent. The back-end systems that receive information, process it, and provide services to end-users are critical to the success of many modern businesses. Should such wholly internal business processes and logic become public, a competitor may easily be able to replicate the business model. Institutions therefore have a vested interest in preventing the disclosure of their internal processes and business logic.

NDAs and employment contracts are typically leveraged to prevent the disclosure of sensitive information. Employees that resign walk off with some level

Published through the viXra.org e-Print archive Nov. 2, 2017.
viXra citation number: viXra:1711.0117. This version contains updates as of Feb. 15, 2018.
This manifesto was submitted to the Cornell University arXiv e-Print service on Oct. 29, 2017 under the category "Computers and Society," was assigned identifier submit/2052823, and was rejected by Cornell University on Nov. 15, 2017 by the moderators "who determined it to be on a topic not covered by arXiv." For the latest version go to www.coderfreedom.org.

of proprietary knowledge. NDAs and employment contracts are designed to prevent word-of-mouth, hard copy, and soft copy disclosure of sensitive business practices, processes, data, and logic. Such agreements are commonly enforced using a process called pre-publication censorship. This is a process in which employees are required to submit to a pre-publication censor within their firm works that they want to publish, works produced during or outside of business hours. The process reviews the work and notifies the employee as to whether or not the institution is willing to allow publication. The institution may decide that the work falls under the category of work-for-hire and thereby refuse to authorize publication. An employee that refutes a decision risks losing his or her job. An employee that skirts the pre-publication censorship process risks losing his or her job.

The problem at hand is that of overreaching institutional censorship policy. An example of overreaching policy is as follows: institutional policy that imposes pre-publication censorship of *all* journal articles, code, books, research papers, reports, and presentations produced by employees during and outside of work hours wherein review and approval is required prior to public dissemination. This **completely** violates the *essence* of the First Amendment to the Constitution of the United States of America as it pertains to abridging the freedom of speech.[1] I define the *free coding teeter-totter* problem to be as follows: balancing the institution's perceived need to censor employees with the human right to free speech as it pertains to code.[2] Central to this chapter is the question of how to balance the free coding teeter-totter.

A natural question to ask is just how pervasive overreaching pre-publication censorship policy is. In a world, under the many-worlds interpretation, wherein all institutional censorship policy is public, this question may be answered by surveying existing institutional censorship policies and presenting an analysis of them. However, in this world, institutional censorship policies are, to a great extent, kept secret. Progress on the social front is hampered by *secret institutional censorship policy*. If such policy upholds the highest ethical standards then why not publish it? Doing so would allow a young coder to know the coding restrictions that she will be subject to *before* taking the job. Doing so would allow a student to know how pervasive coding restrictions are *before* majoring in Computer Science. I would at once justify the breadth of this problem using concrete data derived from published institutional policies but not enough data is available. It is this lack of data, resulting from excessive secrecy of institutional censorship policy, that has forced me to speak publicly about my own

[1]The First Amendment applies to laws that Congress passes. I don't believe that the founders of America ever anticipated *mass* pre-publication censorship of the working-class by corporations.

[2]This is analogous to the *Congressional teeter-totter* [56].

dehumanizing censorship experiences in order to convince the skeptic that this problem is real and harmful.

I am a working-class victim of industrial pre-publication censorship. The following is an account of the harm that industrial pre-publication censorship, and the associated culture that condones it, has inflicted upon my soul and the souls of my colleagues:

1. **Scientists called "wild dogs":** In a prior role I was once at a company lunch wherein it was openly stated that "research scientists are like a pack of wild dogs, roaming and frothing at the mouth." This was said in direct reference to the disposition of research scientists to publish. The speaker was apparently unaware of my contributions to science at the time.

2. **Creative literary works will be reviewed:** While in a leadership role I was once tasked with managing the expectations of an employee, who I shall call Bob, who was having particular moral difficulty with the pre-publication censorship process. I was chosen to assist for being the most familiar with the process, having gone through it numerous times myself. In an obvious "statement" and in compliance with the overreaching policy, Bob submitted a creative literary work of his through the censorship process and resigned shortly thereafter.

3. **Give infosec talk = get fired:** I also had a colleague, who I shall call Dave, who was headed to a conference to give an infosec talk. On his way to the conference he was informed that if he went to the conference he would be fired. Likewise, Dave was not long for that position thereafter.

4. **My family was threatened:** But the singularly defining event, the gravity of which exceeded all events before it and all events after it, happened to me at a chance encounter at work. In a clear reference to my wife Elisa, my daughter Mia, my son Evan, and I, using an unmistakable metaphor that I will never forget, a man at work told me that all four of us were going down. He said it was the only direction we were headed in. Then he asked me why it was so. Based on multiple conversations with this person I have every reason to believe that this threat was in direct reference to my active research in Cryptovirology.

Every single one of these examples stem from a culture of vulnerability hoarding and exploit weaponization. So, whereas the problem of pre-publication censorship might not be evident to some, it is prominent to me. I believe that this problem will become orders of magnitude worse for posterity in every sector of industry. The damage pre-publication censorship causes doesn't get better by sweeping such experiences under the rug. It gets worse.

It has been said that there can be no 'heresy' without 'orthodoxy' (p. 4 of [32]). It is an unfortunate reality that the orthodox view of sharing these experiences is that it "airs dirty laundry." This view stems from the doctrine of *corporate personhood* coupled with the principle of double-effect.[3] The word 'heresy' originally meant 'an act of choosing'—leading to 'the choice of philosophical principles' (p. 5 of [32]). The word 'heresy' lacked the negative connotation that modern society assigns to it. The heretical view of sharing these experiences is that it reveals souls that have been harmed by censorship; that the pre-publication censorship process is sacrilegious.[4] Chapter 3 provides the context for this point.

In the foregoing I have identified not one but *two* distinct problems relating to industrial pre-publication censorship. These problems compound one another, forming a double-whammy involving censorship: overreaching institutional pre-publication censorship policy plus secrecy of said policy to boot. This situation is similar to a double-whammy involving surveillance. As an outcome of its rulings, the secret U.S. FISA court that authorizes surveillance produces an ever growing body of secret law. Since secret law is not subject to public scrutiny, the public has limited assurance that the secret laws uphold the Fourth Amendment. There is one confirmed FISA court case that involved a violation of the Fourth Amendment [56]. It is possible that one or more secret laws run afoul of the Fourth Amendment. The secrecy of those laws prevent the public from knowing the depth and breadth of the problem and prevent the public from leveraging their voting power to effect change. To summarize:

1. **censorship double-whammy**: overreaching institutional pre-publication censorship policy plus secrecy of said policy to boot.

2. **surveillance double-whammy**: possible unconstitutional or objectionable surveillance law plus secrecy of said law to boot.

Not all firms impose undue restrictions on their coders and vulnerability researchers. Some do, but have a very strong habit of "looking the other way" when publications escape review, ultimately giving rise to a perilous sense of freedom. Based on my experience there appears to be an increasing trend in industry towards overreaching pre-publication censorship policy, employment contracts, and NDAs. For example, I cannot recall, at the start of my career, companies requiring visitors to sign an NDA in order to set foot within corporate buildings. Requiring mere visitors to sign an NDA seems to be a growing trend.

[3] The applicability of the principle of double-effect here is covered in Chapter 2.

[4] "I would not show regard for any man, or temper my speech for anyone's sake; For I do not know how to temper my speech—My Maker would soon carry me off!" (Job 32:21-22 of [18]).

This rest of this chapter takes a pragmatic view of free coding, appealing to the utilitarian. Chapter 2 explores rational arguments in support of free coding, leveraging principles from Western philosophy. Overreaching restrictions on free coding form a subset of the problem of knowledge hoarding. Chapter 3 presents a theological foundation of the problem of knowledge hoarding. Chapter 4 presents a highly experimental software license aimed at comprehensively upholding free coding. Acknowledgments are given in Chapter 5.

1.2 Free Coding

I now define what I call Freedom A and Freedom B:

Freedom A:

You have the freedom to write code and give it to society
under conditions of your choosing.

Freedom B:

You have the freedom to write and publish,
under conditions of your choosing,
a critique or documentation of code that has been given to society.

I define *free coding* to be Freedom A and Freedom B. Free coding and free software are different concepts. Free software is embodied by Freedoms 0, 1, 2, and 3 [44].[5] A program is free software if the program's users have the four essential freedoms:

Freedom 0: The freedom to run the program as you wish, for any purpose.

Freedom 1: The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.

Freedom 2: The freedom to redistribute copies so you can help your neighbor.

Freedom 3: The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

The phrase "give it to society" in Freedom A does not necessarily mean making the code free software. The freedom to "give it to society" means the

[5] www.gnu.org/philosophy/free-sw.en.html

freedom to provide it to the people in accordance with the goodness of your heart. Example: Freedoms 1 and 2 are granted but not Freedom 3. In this case it is not as good of a gift as it could be.

The following example illustrates Freedom A and Freedom B in conjunction with Freedoms 0, 1, 2, and 3.

Alice the cryptographer writes a free encryption program and gives the source code of it to society (Freedom A). She is protecting people from harm. Bob the free software freelancer redistributes copies of it as a service to others for a fee (Freedom 2). Carol the vulnerability researcher receives a copy from Bob and studies the source code (Freedom 1), finds a vulnerability in it, and publishes a description of the vulnerability (Freedom B). Dave the coder reads the vulnerability description, fixes the code (Freedom 1), and gives the fixed encryption program to society (Freedom A and Freedom 3). Eddie the journalist receives the fixed encryption program and uses it to encrypt his controversial article (Freedom 0).

Notice how free coding and free software go hand in hand. If Alice could not give her program to society, Bob would not have it to redistribute. If Bob could not redistribute it, Carol would not receive a copy. If Carol received a copy, she could find the vulnerability in it. But, if she could not publish a description of the vulnerability she found, Dave would be oblivious to the vulnerability and would not fix the code. If Dave learned of the vulnerability, but could not fix the code and distribute the fixed program, then Eddie would not have a safe and trustworthy encryption program to use. Regardless of whether one considers code to be property or speech, the following is irrefutable: in some countries, Eddie could be tortured and murdered as a result.

Freedom A helps protect people from harm. The example provided is giving encryption code to society. Freedom A therefore embodies, within the realm of code, the freedom to protect people from harm, the freedom to give, and freedom of speech.

Freedom B also helps protect people from harm. The example provided involves publishing a description of a vulnerability in code that has been given to society. Freedom B therefore embodies, within the realm of code, the freedom to protect people from harm, the freedom to give, and freedom of speech.

The security benefits of Freedoms A, B, 0, 1, 2, and 3 are summarized in Table 1.1. Free coding supports the spiritual nature of coders and vulnerability researchers to contribute to society and make the world a better place. It is a way for them to leverage their unique gifts to help their neighbors.

Upholding Freedom...	Security Impacts
A	Removes obstructions to giving source code society, thereby increasing the number of programs having public source code, thereby increasing the opportunities that vulnerabilities and logic that mistreats the user will be found in programs that people rely on. Removes obstructions to giving strong security and crypto source code to society.
B	Removes obstructions to publishing descriptions of vulnerabilities in public source code, thereby prompting the public source code to be fixed. Removes obstructions to publishing descriptions of logic that mistreats the user in public source code, thereby prompting the public source code to be fixed. Removes obstructions to publishing: implementation-level vulnerabilities, design/architectural vulnerabilities, foundational/theoretical vulnerabilities. Removes obstructions to publishing easy-to-understand APIs and documentation that facilitate secure coding.
0	Removes obstructions to using security programs, crypto programs, and trustworthy programs in general.
1	Removes obstructions to analyzing software to see if the code mistreats the user (Chp. 4 of [44]). Example transgressions: spyware, censorship, backdoors, time-bombs, stealing credentials, surreptitious software installation, contrived performance while under test, concealed user options (from www.gnu.org/proprietary). Removes obstructions to fixing an exploitable vulnerability once it becomes known. Removes obstructions to fixing logic that mistreats the user once it becomes known.
2	Removes obstructions to redistributing secure and trustworthy source code for widespread use.
3	Removes obstructions to fixing vulnerabilities, fixing security issues, and removing logic in public source code that mistreats the user, wherein the derivative source code is publicly distributed.

Table 1.1: Security benefits of Freedoms A, B, 0, 1, 2, and 3

1.3 Obstructions to Free Coding

Overreaching industrial pre-publication censorship is one obstruction to free coding. Other obstructions include international regulations that treat crypto code as munitions and that treat proof-of-concept exploit code as munitions. Laws also restrict free coding.[6] Such obstructions threaten the depth, breadth, and information assurance of society's foundation of code.

In Olmstead v. United States (1928) Justice Louis Brandeis published a famous dissent against the government's position that wire-tapping does not run afoul of the Fourth Amendment since wire-tapping does not extend into the home. He wrote,[7]

"The makers of our Constitution undertook to secure conditions favorable to the pursuit of happiness. They recognized the significance of man's spiritual nature, of his feelings and of his intellect. They knew that only a part of the pain, pleasure and satisfactions of life are to be found in material things. They sought to protect Americans in their beliefs, their thoughts, their emotions and their sensations. They conferred, as against the Government, the right to be let alone—the most comprehensive of rights, and the right most valued by civilized men."

To emphasize the above: *the right to be let alone is the most comprehensive of rights and the right most valued by civilized men.* How then can industrial pre-publication censorship be allowed to extend into the home of Americans and censor all of their publications?

The obstructions are not limited to policies, procedures, regulations, and laws. The belief that code is property as opposed to speech causes Alice to think twice before contributing knowledge to society: her career progression may be at risk, her reputation as a loyal worker-bee may be at risk, she may be subject to civil liability [29], and she may be accused of violating international agreements [5]. This creates both fear and apprehension of contributing code to society and publishing vulnerabilities that are found in code that has been given to society. The issue is not only the submission of a work for pre-publication censorship review and having it be rejected; the issue is also believing so thoroughly that code is property *that one does not even try.* The tangible obstructions to free coding have unequivocally given rise to *psychological obstructions* to free coding. Industrial pre-publication censorship of *all* publications appears to be on track to becoming an uncontested social norm.

[6]The state secret privilege has been applied to descriptions of software vulnerabilities.

[7]Olmstead v. United States, 277 U.S. 438 (1928).

1.4 Balancing the Free Coding Teeter-Totter

I argue that censoring *all* publications made by employees is overkill. There is simply no good reason for such a blanket policy. From an information security perspective it is essentially equivalent to the following, "block all outbound network traffic by default and create proxy servers for new protocols when they arise." Whereas this approach may certainly be suitable for securing the Internet-facing perimeter of an enterprise, it is certainly not suitable for application to people.

1.4.1 Free Coding Policy

A different approach is to have institutions define a list of types of code that are subject to pre-publication censorship. Such a definition would appear within the institution's policy. For example, an on-line gaming company might list: gaming code, gaming engines, graphics drivers, sound drivers, peripheral device drivers, and networking code in support of on-line gaming. This restricts censorship to certain types of code and gives coders the freedom to contribute all other types of code to society, free from censorship. An institution can make the list be "the empty list," thereby showing zero tolerance for censorship. An institution can also grant employees the freedom to publish, without any prior restraint, all descriptions of all vulnerabilities they find in all code that has been given to society. This explicitly authorizes and endorses the publication of software vulnerabilities in code that has been explicitly given to society. This approach is taken in the institutional policy defined below. Use of this policy would help balance the free coding teeter-totter.

The policy statement in the text box below is version 1.0 of the "free coding policy." When adopted by an institution, the institution replaces [Institution-Name] with its own name. Corresponding changes in employment contracts and "intellectual property" agreements may be needed to adequately support the policy.

The policy adheres to the principle of voluntary co-operation (p. 169 of [43]). The institution chooses to utilize the policy. The institution defines the categories listed in policy statement (1). An institution that does not list a single category fully eliminates pre-publication censorship of code. An institution that lists a few categories implements a limited form of pre-publication censorship. An institution that lists "all software" in policy statement (1) maintains complete pre-publication censorship of source code. However, even in this extreme case, policy statement (3) serves to uphold at least part of the goals of free coding. The policy, by design, accommodates a wide-range of business risk tolerance levels.

Any adoption of the policy by an institution that previously censored all publications would help uphold free coding. Even if policy statement (1) has a laundry list of types of code that is censored, there are at least some types of code that will not be censored at all. Progress does not have to be all or nothing.

> Free Coding Policy:
>
> 1. All staff seeking to publish source code that falls into any of the following categories must submit the source code for publication approval prior to publicly disseminating it:
>
> ---
> ---
> ---
>
> This list of categories is subject to change at any time.
>
> 2. All staff seeking to publish source code that falls outside of the categories listed in (1) above are authorized to publish it under a license of their choosing without review and without approval and [InstitutionName] will not assert copyright ownership over the source code, provided that all of the following conditions are met: the source code does not violate any copyrights held by [InstitutionName], the source code does not infringe on any patents held by [InstitutionName], the source code does not utilize any trade secrets of [InstitutionName], the source code was produced outside of work hours, and the source code was not produced using resources owned by our organization.
>
> 3. All staff are authorized to publish descriptions of all vulnerabilities they find in all code that has been given to society. [InstitutionName] will not assert copyright ownership over descriptions of vulnerabilities in code that has been given to society.
>
> 4. To show support for free coding, all staff are authorized to publish the following statement and only the following statement in regards to this policy: "[InstitutionName] has adopted version 1.0 of the free coding policy." All staff members are prohibited from publishing the list of categories from (1) above.

Policy statement (1) is effectively a "disallowed list" in the sense that it is types of code that are censored. All types of code that are not on this list are not censored, subject to certain conditions. Alternatively, policy statement (1) could have been defined to be an "allowed list." With an allowed list an institution defines the types of code that are not censored, subject to certain conditions. Everything else would be censored.

These are two different types of controls. They are both sensitive to the accuracy level of the list. I am of the opinion that an allowed list that defines only those types of software that are not censored would be insufficiently broad in practice. A disallowed list is a logical solution in this case, from the viewpoint of supporting free coding.

Policy statement (4) is what I call a *policy quine*, a new policy concept that I named in honor of the philosopher Willard Van Orman Quine. I define a policy quine to be as follows: a policy statement that authorizes one or more persons to publish (e.g., duplicate) some or all of the policy statement itself and possibly surrounding policy statements as well. In this case it authorizes the publication of a specified line of text. The purpose of this statement is to raise

public awareness of institutions that openly support free coding in the hope that other institutions will follow suit.

There is significant competition for coders and vulnerability researchers. A firm that adopts free coding principles early on could have a hiring advantage over firms that do not. The risk of losing technical talent to firms that openly support free coding could drive adoption and produce "ethical pressure" across the industry to support free coding. Ultimately, supporting free coding may become a social norm that coders and vulnerability researchers come to expect.

This policy is intended to allow institutions to balance business needs with personal liberty. I encourage its adoption to strengthen national security, improve the security of the enterprise, and improve personal liberty and privacy.

1.4.2 The Business Benefits of Free Coding

The following are benefits that adopting the free coding policy may have on the business:

1. **Improved IT Security**: Supporting free coding may increase the amount and types of free/libre and open source software since it removes unnecessary obstructions to giving code to society. This type of software is heavily used by institutions directly by their own developers and also indirectly in third party hardware and software products such as appliances, operating systems, servers, and applications. Public code is subject to the widest degree of scrutiny possible and thereby affords every opportunity for vulnerabilities to be found and fixed. Supporting free coding also opens the door for the publication of vulnerabilities that are discovered. This directly improves the information assurance of free/libre and open source software. The net effect is this: an improvement in the security posture of the enterprise.

2. **Improved Free/Libre and Open Source Software**: The free coding policy paves the way for improving existing free/libre and open source software as well as the introduction of new free/libre and open source software code bases. This broadens and improves the set of software that companies can leverage in their own products and services. The net effect is this: it may reduce the cost of software development since more code will be available as free/libre and open source software.

3. **Advantage in Hiring and Retaining Talent**: The free coding policy encourages institutions to be open about supporting free coding. Many software developers and vulnerability researchers value their freedom. An institution that publicizes that it supports free coding may have a hiring

advantage over institutions that do not and may be better able to retain talent.

4. **Improved Respect for Humanity**: The free coding policy supports the individual liberty of employees. It recognizes the fact that many yearn to give back to their communities, with some better positioned to contribute code and vulnerability discoveries as opposed to money. By supporting free coding, the moral character of institutions will be improved in a key area, paving the way for the individual to give to society.

Supporting free coding has the potential to strengthen the IT security posture of the enterprise, reduce costs in software development, gain an advantage in attracting and retaining staff, and improve avenues in which staff can give back to their communities. The benefits also extent to improving national security and respecting the privacy and liberty of individuals.

1.4.3 Supporting Free Coding via Licensing

Copyleft (Chp. 29 of [44]) is a software licensing technique that has been used to uphold the freedom to use public knowledge, a technique that leverages the power of copyright. I believe that the power of copyright can also be used to uphold the freedom to give knowledge to the public. This is applying the power of copyright *in the other direction*. In particular, below I leverage copyright in a license term that upholds Freedom B tailored to the case of vulnerability publication.

This is an important point that is worth repeating. The license term below should not be confused with provisions of the GPL in any capacity. The GPL is about upholding the freedom to *use knowledge* that has been given to society and making sure that it can be used in downstream works. The license term below is about the freedom to *give knowledge* to society. These are two completely different things: using public knowledge vs. giving knowledge to the public. It is inaccurate to say that the below license term "is an extension to the GPL" since it does not uphold the freedom to use public knowledge. It is inaccurate to say that the below license term "is an extension of copyleft" for the same reason. It is my hope that the principles of *free coding* will not be confused with the principles of *free software*. They are different.

By conditioning the modification, redistribution, and distribution of derivatives of a program on the adoption of Freedom A and Freedom B, the legal power of copyright can be leveraged to uphold free coding. Below is a license term that supports Freedom B restricted to the case that the code critique is the description of a vulnerability.

"The phrase "freedom to publish" means the freedom to publish without approval, without restraint, and without prior restraint. If you are an organization then as a condition of modifying, redistributing, or distributing derivatives of the covered work you must: (1) grant all of your staff the freedom to publish, under conditions of their choosing, descriptions of all vulnerabilities they find in all code that his been given to society, and (2) waive all copyright claims to all descriptions, produced by your staff, of all vulnerabilities in all code that has been given to society."

The above term can be used as a starting point to upholding Freedom B in software licenses. Care should be taken to leverage an appropriate term that forces the above term to take effect in works produced downstream. The above term is *experimental*. Only legal experts should develop it and include it in a mature software license.

An important case to consider is as follows. A company releases software under a software license that contains the above term and then a vulnerability researcher who is an employee at the same company later discovers a vulnerability in the software. Per the license term, the employee can publish the vulnerability immediately.

A first reaction to this scenario may be that immediate publication of the vulnerability could harm the company. The perceived harm can be broken down into two categories: infosec risk and reputational risk. One might conclude from these two types of risk that the above license term should be changed to, e.g., mandate a 90-day responsible disclosure period prior to any publication of the vulnerability.

In some cases any delay in publishing the vulnerability could cause harm to people. A responsible disclosure delay is considered a best practice but who is to say that it will end up being the right approach when considering overall loss of life and limb? No one knows the perfect responsible disclosure delay interval and so it is a pure judgment call whether the vulnerability researcher should decide when to publish the vulnerability vs. whether the institution should decide. The right answer depends upon the philosophical (or theological) lens through which one views the question.

The reputational risk stems from the doctrine of corporate personhood.[8] *Corporate personhood* is a personification of the corporation, one that is backed by the law. A classic example is granting corporations the right to own property. Corporate personhood leads to the belief that a corporation can be "harmed" in the same sense as an individual. In what follows corporate personhood is personified as "the corporate person." Despite the fact that the software was given to society, there may be a perception that the reputation of the institution

[8] en.wikipedia.org/wiki/Corporate_personhood

is tied to the information assurance of the software. Perhaps the company is the entity that is making the most updates to it and so this perception persists over time.

Worry over reputational damage has one foot rooted in pride and the other rooted in the pursuit of silver and gold. Humility is a virtue and pride is a vice. The corporate person should be held to the highest moral standard. Therefore, the corporate person should be every bit as humble as the individual. The reproof of vulnerability publication should be taken in stride by the corporate person. The freedom of the vulnerability researcher to protect his neighbor from harm, the freedom of the vulnerability researcher to give, and the freedom of speech of the vulnerability researcher should not be restricted in any way, shape, or form for the sake of the corporate person's vices. The software was *given* to society and there should be no strings attached.

If one were to relax this responsible disclosure requirement and have some entity choose whether or not a given vulnerability must undergo a responsible disclosure delay who should that entity be? The vulnerability researcher, the institution, our an outside third party in this case? If it is not the vulnerability researcher who decides then coder freedom is taken away; in particular, the vulnerability researcher suffers a restriction on Freedom B. Put another way, the vulnerability researcher has lost the freedom to warn, in a timely fashion, his neighbor about a vulnerability that exists in society's foundation of code.

I am of the view that, year after year, the corporate person increasingly regards employees as untrusted cogs that need to be controlled for the machine to run smoothly. Changing the above license term to force a 90-day responsible disclosure period is a natural corollary to this belief. I would rather live in a world in which wage workers are viewed as human beings with hearts, minds, and souls. It is from this view that the above license term is proposed. Let developers contribute code to society on these terms, terms that value freedom over exploitation and control.

1.4.4 Contract Patching

The Software Freedom Conservancy has observed that employment agreements that free/libre and open source software (FLOSS) developers sign can affect whether and how developers contribute to FLOSS projects. ContractPatch is an initiative of the Software Freedom Conservancy to give developers the language that they need to defend their freedom to contribute to FLOSS projects. The initiative is aimed at providing developers with negotiation tactics that can be used during the hiring process, providing language for a prospective employment agreement, and giving general information on legal rights relating to contracts.[9]

[9]See: https://sfconservancy.org/contractpatch

1.5 Conclusion

The foundation has been laid by the Free Software Foundation to uphold the freedom to use public knowledge. But, there are significant obstructions to giving knowledge to the public in the first place. These are two separate but important issues. They impact one another as shown in the software life-cycle example involving Alice, Bob, Carol, Dave, and Eddie.

Institutions have equities they seek to manage. People have a spiritual desire to help their neighbors. These are the opposing sides of the free coding teeter-totter. I encourage vigorous debate on this issue, to let the *marketplace of ideas* find a way forward [27, 26]. It is in the interests of the institution and the individual alike to balance this teeter-totter, as it will allow both to benefit from technological advances and free will.

Chapter 2

Philosophical Foundation of Free Coding

Prior to the 1980s source code was, for the most part, universally shared gratis. Then NDAs and industrial pre-publication censorship were applied to hoard source code. In the 1990s software bugs transitioned from being widely perceived as defects to being widely perceived as vulnerabilities that can be exploited. Today, the CIA and NSA hoard software vulnerabilities, placing the privacy and safety of individual Americans at risk. So, now even software vulnerabilities are being hoarded. What is happening to humanity? Where are we *going*? One can only wonder what's next. There will be something next. Insert the next harmful hoarding practice here: $\mathcal{X}$. New restrictions on the human spirit call for new liberties to be defined and upheld. Chapter 1 introduced *free coding* to uphold the freedom of coders and vulnerability researchers to give knowledge to society. This chapter leverages rational arguments and principles from Western philosophy to establish a philosophical foundation of free coding. I present a new philosophy that I call *coder consciousness* that is aimed at inspiring coders and vulnerability researchers to embrace the spiritual nature of coding.

2.1 Introduction

People help others in a variety of ways. Some give by teaching. Others give by volunteering their time and energy. Some give blood. Others give money. Some give an encore on the street corner. And some leverage their intellectual capacities to write code and give it to society.

This manifesto is about the freedom to give knowledge to society, to give within the realm of code. I define Freedom A and Freedom B as follows.

Published through the viXra.org e-Print archive Nov. 2, 2017.
viXra citation number: viXra:1711.0117. This version contains updates as of Feb. 15, 2018.
This manifesto was submitted to the Cornell University arXiv e-Print service on Oct. 29, 2017 under the category "Computers and Society," was assigned identifier submit/2052823, and was rejected by Cornell University on Nov. 15, 2017 by the moderators "who determined it to be on a topic not covered by arXiv." For the latest version go to www.coderfreedom.org.

Freedom A:

You have the freedom to write code and give it to society
under conditions of your choosing.

Freedom B:

You have the freedom to write and publish,
under conditions of your choosing,
a critique or documentation of code that has been given to society.

Note that these definitions include the freedom to "write" the code and the freedom to "write" the critique. It may seem that this is not necessary. This is not the case. A Restricted Person under South Africa's banning decree was forbidden to write anything, even a diary or a postcard. A Restricted Person was forbidden to speak to or associate with more than one person at a time other than immediate family,[1] was forbidden to travel, was forbidden to communicate publicly, and was forbidden to be quoted in any publication. Donald Woods and Steve Biko were Restricted Persons. Donald noted 44 people restricted under the rule of apartheid (Prologue of [55]).

Freedom A is important for security for multiple reasons. It is a freedom that is needed to give security code such as an encryption program to society. This protects people from harm. Source code that is published offers every opportunity for vulnerability researchers to find exploitable vulnerabilities in the code. Source code that is kept secret hampers the ability of vulnerability researchers to find exploitable vulnerabilities in it. Source code that is secret also hampers the ability of coders to determine whether or not the code constitutes malware. There are many programs having secret source code that spy on their users or mistreat them in some way.[2] Many such programs are not even formally designated as malware by antivirus programs. Simply put, published source code is important in achieving information assurance and trust. Freedom A therefore embodies these liberties: freedom of speech, freedom to give, and freedom to protect people from harm.

Freedom B is important for security for multiple reasons. It embodies the freedom of a vulnerability researcher to publish a description of a vulnerability that he or she finds in code that has been given to society. This protects people from harm. There are multiple manifestations of such vulnerabilities, ranging

[1] Steve Biko could attend no gathering of any kind except a *bona fide* Church service (p. 161 of [4]).

[2] See www.gnu.org/proprietary

from classic buffer overflow bugs, to arbitrarily complex design flaws, to deep mathematical failures in the security foundation of code. Such critiques are the expressions of the vulnerability researcher in every respect. Freedom B therefore embodies these liberties: freedom of speech, freedom to give, and freedom to protect people from harm.

I define *free coding* to be Freedoms A and B. Freedoms A and B are entirely separate and distinct from Freedoms 0, 1, 2, and 3 from the Free Software Movement (Chp. 1 of [44]). Freedoms A and B address the freedom to give knowledge to society. Freedoms $0-3$ address the freedom to use knowledge that has been given to society.[3] Therefore, the notion of free coding should not be confused with the notion of free software.

Freedoms A and B are human rights. They allow Alice to contribute to public knowledge and thereby help her neighbor Bob. Freedom A and Freedom B are critically important in building society's foundation of code: the volume of code, the quality of the code, and the information assurance of the code.

A summary of this chapter is as follows:

1. Free coding is defined, obstructions that exist to free coding are identified, and the harmful effects that incorrect terminology has on free coding are covered.

2. A philosophical interpretation of the industrial practice of pre-publication censorship is presented.

3. A Western philosophical foundation of free coding is given that leverages arguments in support of free coding.

4. To explore all sides of the issue, arguments against free coding are presented.

5. Western philosophy is applied to characterize the problem of knowledge hoarding.

6. A new philosophy called *coder consciousness* is presented. It is aimed at encouraging coders and vulnerability researchers to embrace the spiritual nature of coding.

I dedicate this manifesto to Bantu Steve Biko. I first learned about Biko from the song "Biko" by Peter Gabriel that I heard when I was a child. I played it on the record player over and over. Being a young white boy living in suburban Connecticut with all the privilege in the world, I had no conception of this great

[3]The battle for freedom to use public knowledge is addressed in (p. 189 of [54]).

man. But something in the music caused Biko to subconsciously take root in my soul. Bantu Steve Biko fought apartheid, fought the systematic subjugation of native South Africans, and fought the disintegration and distortion of South African culture and history. Prior to the arrival of the Anglo–Boer colonists, South Africans had a *sacred* tradition of sharing (p. 96 of [4]).

I dedicate this manifesto to Steve Biko for the following reasons: First and foremost, because he was a visionary philosopher on social oppression, social resistance, and social change. When he was on trial in South Africa it was quite literally his *ideas* that were on trial; Secondly, because he endured the worst-possible restrictions on giving knowledge to society, being forbidden to write, speak publicly, and meet with more than one person at a time except with immediate family or to attend a *bona fide* Church service. Thirdly, because he sacrificed his life at the violent hands of the oppressor to free his people; and lastly, to uphold the parting guidance of Donald Woods in his book "Biko" (p. 376 of [55]):

"Help to finish the work of Steve Biko. Help to smash the remaining links of the chains he broke, and let the sound of this work echo around the world so that chains may be broken wherever they hold in bondage the bodies and minds of men."

2.2 Free Expression and the Manifesto

The concept of a manifesto has a negative connotation in Western society. It is associated with a form of radicalism, surfacing uncomfortable or controversial issues that many would rather not think about. Yet the Merriam-Webster dictionary contains the following rather disarming definition: a written statement declaring publicly the intentions, motives, or views of its issuer.

2.2.1 The Positive Manifesto

The U.S. Declaration of Independence is a manifesto. The dissent of a U.S. judge is a manifesto. The GNU Manifesto set the stage for free software in 1984 (p. 31 of [43]). Without it, the data centers of the world would be filled with nonfree operating systems having secret source code together with laws that make reverse-engineering them illegal as opposed to the GNU/Linux operating system that is free software. It is because of free software that we are able to study the source code of the GNU/Linux operating system and look for malicious logic at the source code level. There are those who look down upon the GPLv2, declaring it virus-like since it forces all new software with which it is intertwined to be shared with society. That is its precise intention because it is code that

has been given to society distributed with the condition that modifications be available for use by society. The GPLv3 goes even further to assure the freedom to use public knowledge. Some companies live in perpetual fear that one of their developers will merge their proprietary code with free software released solely under the GPL, thereby placing their proprietary code in peril of having to be shared.

Yet even the harshest of critics of the GPL can surely see the value in the GNU/Linux operating system being free software. Without it many users and organizations would be compelled to place their trust in nonfree operating systems that have secret source code, operating systems that would inevitably undermine the privacy and safety of their users. Simply put, the GNU Manifesto set in motion a monumental defense against tyranny.

2.2.2 Free Speech Justifications for this Manifesto

Free coding embodies freedom of speech, freedom to give, and freedom to protect people from harm. In the sections that follow rational arguments in support of free coding are given. Yet it is also instructive to convey the justifications for freely expressing this manifesto itself. The following are arguments in support of free speech outlined by James Fieser [13], namely, democratic government, the search for truth, and personal autonomy.

Democratic Government: The industrial practice of pre-publication censorship is addressed herein that chills contributions of code to society and that chills the publication of critiques of code that has been given to society. Pre-publication censorship has a harmful effect on the body of software that people can build upon, use, and trust. It has a harmful effect on the information assurance provided by software that people rely on every day for their privacy and safety. Free speech is a cornerstone of democracy. It is a way for the people to inform their legislators of injustice and unethical practices, thereby allowing lawmakers to pass laws that protect the human rights of citizens, reining in industry when it oppresses the people. A goal of this manifesto is to inform democratic debate on practices that inhibit free coding. This type of speech is required for the proper functioning of a democracy.

Search for truth: Free speech is essential to allow society to search for truth. This is distinct from the role that free speech plays in democracy. Silencing new concepts and ideas prevents society from advancing on scientific and social fronts. As observed by John Stuart Mill (p. 21 of [26]):

"But the peculiar evil of silencing the expression of an opinion is, that it is robbing the human race; posterity as well as the existing generation; those

who dissent from the opinion, still more than those who hold it. If the opinion is right, they are deprived of the opportunity of exchanging error for truth: if wrong, they lose, what is almost as great a benefit, the clearer perception and livelier impression of truth, produced by its collision with error."

The Western philosopher Hannah Arendt constructs "the meaning of life" out of a sincere and open-ended commitment to becoming human by virtue of the decisions we make, the skillful use of language in the public domain, and unique contributions to the creation of a collective identity (p. 33 of [36]). In this sense, the search for truth is one facet of the meaning of life. A goal of this manifesto is to define the issue of free coding, bring it to the fore, and encourage debate and progress on it.

Personal autonomy: Free speech is a human right. As thinking, caring, and creative members of the human race we yearn to convey through speech and expression the thoughts and visions in our minds, the feelings in our hearts, and the music in our souls. In the words of H. W. Beecher, "Liberty is the soul's right to breathe, and when it cannot take a long breath, laws are girdled too tight. Without liberty man is in a syncope." (p. 70 of [3]).

The Universal Declaration of Human Rights adopted by the United Nations in 1948 states the following in Article 19:

"Everyone has the right to freedom of opinion and expression; this right includes freedom to hold opinions without interference and to seek, receive and impart information and ideas through any media and regardless of frontiers."

Two additional free-speech justifications for this manifesto are the dead dogma argument ([26], p. 28-29,35 of [50]) and *Liberty of Thought* from John Stuart Mill.

Dead Dogma: Unless a new idea or belief is fully, frequently, and fearlessly discussed, one is in danger of holding it as a dead dogma, providing only an unthinking and formulaic response. In not speaking about nor challenging restraints on free coding, coders are likely to: accept restraints on free coding as a simple fact of life, not understand both sides of the free coding issue, not feel passionate about arguments in support of free coding, and lean towards believing restraints on free coding are justified simply because no one talks about free coding. Without opponents of commonly held views we will be less alive as rational thinkers.

Liberty of Thought: John Stuart Mill believed that there is essentially no difference between the freedom to think vs. the freedom to speak and write. He wrote, "This one branch is the Liberty of Thought: from which it is impossible to

separate the cognate liberty of speaking and of writing" (p. 18 of [26]). One who is willing to accept this as true may rightfully equate the severity of restraining speech to the severity of restraining thought itself.

2.3 The Problem: Restraints on Free Coding

A major hurdle to achieving free coding is industrial pre-publication censorship. In the subsections that follow the dimensions of this problem are presented. Steps that an institution can take to better manage the needs of the business with respect to personal liberty are covered in Chapter 1.

2.3.1 Problem 1: Pre-Publication Censorship

When copyright law is in effect Freedom A may be restrained by not allowing one to copy source code covered by copyright when the privilege of copying has not been granted. Employment law and agency principles affect whether or not a work is legally "made for hire." This can affect who the copyright holder is. These are legal restraints.

When we talk about free coding we are talking about liberty, not the law or the Constitution. These can be unethical. The Constitution did not grant women the right to vote until the 19^{th} Amendment in 1920. The Fugitive Slave Act of 1793 guaranteed the right of a slaveholder to recover an escaped slave. The title of the Act was "An Act respecting fugitives from justice, and persons escaping from the service of their masters."[4] Jim Crow laws segregated African Americans from whites until 1965.[5] It is appalling when the voting majority supports such things as Jim Crow laws. The majority can be woefully unjust. Therefore, we owe it to ourselves as thinking, feeling, and conscionable members of the human race to ever question what is and ever contemplate what may be so that our spirits may transcend.

This manifesto addresses the following obstruction to free coding:

Problem: Overreaching institutional policy and agreements that impose undue restrictions on the speech of the institution's staff members. An example is an institutional policy that imposes pre-publication censorship of all journal articles, code, books, research papers, reports, and presentations produced by staff during and outside of work hours. Review and approval is required prior to public dissemination. This is an obstruction to free coding.

Approval to publish a given work is often required from the author's manager and possibly others, sometimes going significantly far up the management

[4] en.wikipedia.org/wiki/Fugitive_Slave_Act_of_1793

[5] en.wikipedia.org/wiki/Jim_Crow_laws

chain. Pre-publication review is work. It is work that is assigned in 180 degrees the opposite direction than work is normally assigned in companies. An author creates work and assigns it to his manager to complete. Pre-publication censorship is therefore an administrative burden of the first order; and whether or not that is what it seems from above that is what it is from below.

Industrial pre-publication censorship strips the proletariat of his clothes, makes him effuse his soul upon the operating table, invites opinion and critique from his managers where none is wanted, poking here at a spelling error, questioning there a line of thought, with the occasional congratulatory remark being offered, ever reminding the proletariat that his work is indeed being judged by those who wield power over him. It is a process that inextricably links the proletariat's ability to feed his children on the morrow with his feeble attempts at making the world a better place by speaking. The following is from Henry David Thoreau in his chapter entitled *Economy* [46]:

> *"The mass of men lead lives of quiet desperation. What is called resignation is confirmed desperation. From the desperate city you go into the desperate country, and have to console yourself with the bravery of minks and muskrats."*

A company may seek to justify the practice of pre-publication censorship on the basis that other companies institute pre-publication censorship as well. This is an instance of *the tyranny of the majority.* From *On Liberty* by John Stuart Mill (p. 7 of [26]):

> *"This view of things, recommending itself equally to the intelligence of thinkers and to the inclination of those important classes in European society to whose real or supposed interests democracy is adverse, has had no difficulty in establishing itself; and in political speculations 'the tyranny of the majority' is now generally included among the evils against which society requires to be on its guard."*

Among coders, cryptographers are particularly sensitized to obstructions to sharing code. The ciphers they design and code have been treated no less than weapons of war banned from export by international arms control regulations.[6]

2.3.2 Problem 2: Industrial Censorship De-Anonymization

Brilliant minds have published works anonymously under pen names, otherwise known as pseudonyms, for years. It is a standard approach to publishing controversial ideas and creative works in the face of an intolerant or prejudiced majority. Example authors include Steve Biko as Frank Talk and Mary Ann Evans as George Eliot. Also, there are real-world scenarios in which a coder

[6]See *Bernstein v. United States.*

may need to exercise free coding anonymously. This section concludes with two such examples.

Exercising the right to contribute written works to society anonymously is particularly challenging when faced with a pre-publication censor. To explain why sufficiently, an understanding of the controls that are leveraged in institutions is needed.

Some institutions have each employee sign an "intellectual property" agreement and/or employment contract. Contained therein may be language that mandates the pre-publication censorship of all publications produced by the employee during and outside of work hours. The institution may have a policy that requires that all employees submit all written works intended for publication through a specific pre-publication censorship procedure. The procedure may be defined in a procedure document and/or ticketing system. The chain of approvers may be well-defined a priori but in some cases it is somewhat ad hoc. Finally, some institutions go as far as to return to the requesting author a hard copy of the fully processed publication request, the final state being "approved" or "rejected." The purpose of the policy and corresponding procedure is to enforce the "intellectual property" agreement/employment contract.

The requesting author is typically required to provide the full written work to be published to the institution and reveal him or herself as an author. Anonymous does *not* mean the censor plus Alice knows that Alice is the author. Anonymous means that *only* Alice knows Alice is the author. Anonymous means *anonymous*. Her anonymity is compromised by even the smallest shred of information that can correlate her to the published work.

Below is a hypothetical exchange between Alice and the censor of ABC Corp, her employer:

Alice: I am requesting approval to publish a paper anonymously. I plan to use a pseudonym.

Censor: Okay, well what is it about?

Alice: I can't really say without de-anonymizing the paper

Censor: Did you write this yourself or do you have co-authors?

Alice: If I tell you the number of authors (i.e., number of pseudonyms that will be used) that could completely de-anonymize the work.

Censor: Fine. But you must tell us if all co-authors work for this company.

Alice: There exists a co-author not at this company.

Censor: Thank you. Please tell us if the paper relates to or could impact our business in any way.

Alice: The paper does not mention ABC Corp anywhere.

Censor: You didn't answer the question completely. Could it impact our business in any way?

Alice: I cannot see into the future. It advances a certain area of knowledge in a non-trivial way. So, here is one speculation: it could change the way some people at ABC Corp view the underlying problem.

Censor: Can you please clarify?

Alice: No. That would risk de-anonymizing the work. In fact, you've already forced me to reveal that there is more than one author.

Every person should have the right to publish his or her ideas under a pseudonym. No person should have to break that anonymity for their employer. Alice is justifiably concerned that by revealing even the smallest bit of information about the publication to her employer that her authorship could be inferred by her employer. She may have no control over how well the institution will safeguard her identity as an author.

The ability to exercise free coding anonymously is important for the privacy and safety of coders in some contexts. The following are examples that illustrate this. Suppose that Alice lives in a censorship state and that she wants to contribute code to an anti-censorship protocol. If the authorities were to find out, they might arrest her or place her on a watch list. The same applies if she discovers and publishes a critical vulnerability in it, thereby hardening it against attack.

2.3.3 Problem 3: The Euphemism of "Business Ethics"

The phrase "business ethics" came into common usage in the U.S. in the 1970s [16]. The phrase originally denoted the study of everyday moral and ethical norms *to* business. However, over time its meaning expanded to denote the moral and ethical norms *in* business that uphold some human rights but not others. In the office, the phrase "business ethics" has strayed from its original meaning. The proletariat sees this phrase placed upon a pedestal with a plaque that reads "highest ethical standards," declaring the importance of knowing one's customer (to avoid allegations of money laundering), adhering to embargoes against certain foreign countries, prohibiting hiring children under 18 to do dangerous work, and prohibiting the use of prison labor, bonded labor, military labor, slave labor, and indentured labor [8], while on the other hand not only failing to mention the human right to freedom of speech but in fact instituting policies and procedures that take away the freedom of speech of employees. The

term "business ethics" conspicuously omits the issue of the human right to free speech.

Calling things by their right name is critically important. Confucius believed that this is the single most important way to achieve peace and order. The Chinese word *Li* cannot be rendered by an English word. Interpretations include: sense of propriety, the order of things, ritual, good manners, and an ideal social order with everything in its place (p. 13 of [58]). An understanding of *Li* helps convey a corollary of Confucius' teachings: that everything should be called by its right name. Confucius believed that when the ceremonies are improper, things become disorderly, and when the terminology employed is incorrect, then things are out of place (p. 98 of [58]).

When Confucius wrote the Spring and Autumn, a goal of his was to restore the social order by using crisp distinctions in terminology. When the Baron of Wu usurped the title of "King," Confucius wrote down "Baron Wu" thinking he had denigrated him (p. 17 of [58]). By using certain words to condemn or show approval of practices of his time, Confucius hoped that if a king ever opened the book in the future and adopted the principles it taught, unruly princes and thieves of power would restrain themselves out of shame (p. 96 of [58]).

At one point in which Confucius was potentially faced with the opportunity to govern he was asked, "If the ruler of Wei should put you in power, how would you begin?" Confucius answered by saying, "I would begin with establishing a correct usage of terminology." (p. 85 of [58]).

The Free Software Foundation (FSF) has pointed out the negative impact of improper terminology on user freedom. An example from the FSF will go a long way to show the impact of terminology on user freedom.

When we hear the acronym DRM expanded into "digital rights management" it causes us to think of the "rights" of the rights holder of the work it covers.[7] It can just as easily be expanded into "digital restrictions management" that instead causes us to think of the restrictions that it imposes on users (p. 88 of [44]). The phrase "digital rights management" therefore sends a negative message from a user's perspective, fostering sympathy for the rights holder while marginalizing the end user. The FSF has provided astute guidance on recognizing and countering this type of indoctrination.[8] This example shows that words and phrases are loaded with meaning and can lead one to overlook one side of a particular issue. The FSF has not only identified and exposed misleading terminology that negatively impacts user freedom, but has also offered replacement terminology in an effort to fix the problem.

Words and phrases can also be hurtful with pervasive and long lasting ef-

[7] It is more accurate to say "privileges" instead of "rights."

[8] www.gnu.org/philosophy/words-to-avoid.en.html

fects. In directly responding to a judge, Steve Biko stated: "...the term black is normally in association also with negative aspects, in other words you speak of the black market, you speak of the black sheep of the family, you speak of—you know, anything which is supposed to be bad is also considered to be black" (Chp. 3 of [55]). The information security industry is remarkably insensitive to phrases that use the word black:

1. Blacklist: It is standard practice in information security to use a whitelist to define elements that are allowed and a blacklist to define elements that are not allowed. For example, a firewall can be configured to use a "whitelist" of IP addresses. A packet is not allowed in unless it's source IP address is in the whitelist. Whitelisted IPs are privileged. Alternatively, a firewall can be configured to use a "blacklist" of IP addresses. A packet is allowed in unless it's source IP is in the blacklist, in which case the packet is blocked from entering. Blacklisted IPs are bad.

2. Black market: The term black market is used to describe ethically questionable marketplaces in which vulnerabilities are sold, sometimes to ill-intentioned buyers.

3. Blackhat: The word blackhat is often used to refer to a computer attacker that breaks into a computer network or computer with malicious intent. It is also used to refer to an exploitive computer security researcher. This contrasts with a "whitehat" that is a "good guy" security analyst that seeks vulnerabilities for the purpose of having them fixed.

These terms enter into infosec conversations the world over on a daily basis. The use of a color to differentiate these things is hurtful and unnecessary. Biko had it right in 1976. The IT Security industry should be more mindful of this issue. I encourage the use of alternative terminology. The following may be considered: "allowed list," "disallowed list," "underground vulnerability market," "goodhat," "neutralhat," and "badhat."

It is my hope that these examples, namely, "business ethics," "digital rights management," "blacklisting," "vulnerability black market," and "blackhat" convey the importance of calling things by their right name. Getting back to the issue of the phrase "business ethics," I posit the question: is there a more suitable phrase and under what circumstances should it be used? I argue that, in the case of a company that institutes pre-publication censorship of all publications produced by all of its staff during *and* outside of work hours, a more suitable name is "bourgeoisie ethics."

2.3.4 Problem 4: The Negative Message of Censorship Policy

Institutional policy that mandates pre-publication censorship of all publications produced by the institution's members and blanket agreements that lay claim to the same sends a negative message: that the freedom to contribute to public knowledge pales in importance to protecting "intellectual property." This is evidenced by the office that censors the speech of the institution's members. Such measures feign service to every echelon, the nethermost notwithstanding, fronting such hazards as these should an article bypass review: diminished profits, increased liabilities, forfeiture of privileges, brand damage, and drawing unwanted attention. It is through such policy and agreements that institutions inculcate upon the proletariat the belief that contributing to public knowledge is perilous. In one fell swoop such measures trample the coding liberty of the individual within and the corresponding humanitarian benefits to society without.

The worst case is not the forfeiture of coding liberty by Alice, the coder, who, having little alternative, must make ends meet; for she may yet vie for her freedom. Rather, the worst case is a working-class so hopelessly institutionalized that it ceases to believe that freedom is achievable. At length the very taste of coding liberty may be all but forgotten.

2.4 A Philosophical Interpretation of Industrial Pre-Publication Censorship

According to the Constitution, people have rights, corporations do not. The U.S. Constitution does not mention corporations. But in 1823 the Supreme Court decided *Society for the Propagation of the Gospel in Foreign Parts v. Town of Pawlet*. In writing for the court, Justice Joseph Story explicitly granted to corporations the same property protections that individuals have.[9] In 1930 Chief Justice Marshall wrote that, "The great object of an incorporation is to bestow the character and properties of individuality on a collective and changing body of men."[10]

Corporate personhood is a personification of the corporation, one that is backed by the law. In what follows corporate personhood is regarded as a person who I shall call the "corporate person." The issue at hand is the property rights of the "corporate person" vs. the free speech rights of the individual. The below timeline shows the initial transition of code from being perceived as speech to being perceived as property.

[9] en.wikipedia.org/wiki/Corporate_personhood

[10] Providence Bank v. Billings, 29 U.S. 514 (1830).

1970s: In the 1970s software was shared openly across communities (p. 15 of [43]). Computers used to be large, expensive, and few and far between. Software was garnish freely given by most computer companies to make computers more appetizing (p. 99 of [54]). As the price of computers dropped and operating system and application software showed promise of being profitable, code began to be treated as property instead of speech.

1980s: Non-Disclosure Agreements (NDA) covering software started to appear in the 1980s ending the era of source code sharing. NDAs were fashioned to cover software to restrict the sharing of code. Stallman recounts his first encounter with an NDA on a visit to Carnegie Mellon University. He was requesting the source code to a Xerox laser printer and found to his surprise that the person with it had promised not to share it with anyone due to an NDA (p. 8 of [54]). He also recounts when the AI lab purchased the PDP-10 in 1982 that, for the first time, came with an associated NDA (p. 16 of [43]). This implied that you had to promise not to help your neighbor to even get a copy of the OS.

Industrial pre-publication censorship of code operates in tandem with NDAs to further control the dissemination of code. It was the opportunity to generate profit that prompted companies to start treating code as property instead of speech. This historical context establishes the practice of the "corporate person" to effectively treat code as property. The switch in society from sharing code to hoarding code opened the door to censorship of code. I now argue that the *principle of double-effect* comes into play in relation to censoring code.

The principle of double-effect was put forth by the philosopher, theologian, and jurist Thomas Aquinas. The principle of double-effect explains the permissibility of an act that inflicts significant harm as a side-effect of promoting some form of good. For example, it explains why some rationalize that it is okay for Alice to hit Bob so hard that he dies if it is the only way for Alice to save herself from Bob's attack (p. 224 of [51]).

The practice of industrial pre-publication censorship can be rationalized using the principle of double-effect from the perspective of the "corporate person." The "corporate person" believes that it is okay to apply pre-publication censorship to a member of the corporation on the premise that it is the only way to "protect" the property of the "corporate person."

Therefore, it was the invention of corporate-owned property followed by the invention that code is property instead of speech that set the right of the "corporate person" to own property on a collision course with the human right to freedom of speech.

2.5 Western Philosophical Foundation of Free Coding

The goal of this section is to present a philosophical foundation for free coding. For security benefits of free coding that may appeal to the utilitarian, see Table 1.1.

At the outset, it may seem that but one obstruction is at issue: the practice of pre-publication censorship. In fact, many will have obstructions of their own: monoliths that eclipse Freedom A and Freedom B. Being wholly distinct, they mire the way forward for the logician, the capitalist, and the pious. This section appeals to the logician and the capitalist and Chapter 3 appeals to the pious.

It was noted that Kantian ethics; or, the Golden Rule, demonstrates the destructive nature of restricting a user's ability to use a program (p. 36 of [43]). Kantian philosophy can also be used to rationalize the harm caused by obstructing Freedom A and Freedom B.

In a toast to Kant, I now fashion the modern business practice of pre-publication censorship into a maxim. The maxim should not be too narrow since pre-publication censorship chills many types of speech: from publishing articles and books, to code contributions to society, to publishing vulnerabilities found in code given to society, to matters that may affect the reputation of the institution,[11] to anything at all that may be deemed "intellectual property." Such public discourse may be perceived as impacting the "equities" of the institution. In the case of a cyber arms producer, vulnerabilities in code are their bread and butter. So, the term equities is fitting.

A maxim has three components: (1) the action, or type of action, (2) the end or purpose to be achieved by the action, or the motive, and (3) the conditions under which it is to be done.[12] The proposed maxim for industrial pre-publication censorship is as follows: "Apply pre-publication censorship to the working-class to protect the equities of our institution, administering it to all within but not those without so that our workers can choose to live free of censorship." The action is: apply pre-publication censorship to the working-class. The purpose to be achieved is: to "protect" the equities of our institution. The conditions are: administering it to all within but not those without so that our workers can choose to live free of censorship. The maxim begs for reprieve, to give it a fighting chance, to take in the full measure of it's effect on the individual, the institution, and society.

Kantian philosophy has us take a maxim, apply it universally, and then ask

[11] Is there consideration given to the reputational harm that industrial pre-publication censorship policy has on employees?

[12] en.wikipedia.org/wiki/Maxim_(philosophy)

the question, "Is it still conceivable in this world?" For if it is, then we should properly embrace it and carry on; or, if it should prove inconceivable, why then we should shine the brightest light upon it for all to see.

For concreteness, let us suppose that ABC Corporation applies this maxim and that Alice, a working-class coder, is one of their number. Per Kantian philosophy we now apply this maxim universally; every corporation is ABC Corporation. We see then that there is nowhere for Alice to turn. The maxim is upended since Alice cannot skirt censorship. A contradiction in conception has therefore been reached. Holding the maxim is irrational. Working-class coders and vulnerability researchers are, in effect, living in a total and complete censorship state.

This universal state is entirely conceivable and possible. I have *proven* logically that the business practice of pre-publication censorship is *destructive*. But there are still more states to consider. Does paring back the number of infringing institutions pardon ABC Corporation? If none other than XYZ Corporation supports free coding, is the innocence of all corporations tied to the existence of XYZ? Quite to the contrary, ABC Corporation is in every way responsible for its measure of harm to society.

The point here is that the argument of ABC Corporation, "you agreed to these work conditions so if you want you can work some place else" does not in any way exonerate the unethical censorship practice at ABC Corporation. Just before the 1920s it was believed, though not yet scientifically proven, that asbestos was hurting people in factories that produced insulation. The fibers could be seen floating in the air. Asbestos companies conducted their own investigations and in some cases tried to suppress the inconvenient results [38]. Let us instead suppose that ABC Corporation consists of factories that make insulation out of asbestos with asbestos floating in the air. Does the argument, "you agreed to these work conditions so if you want you can work some place else" apply in that situation? There may be a factory with clean air that Alice can go work at now, but in the future perhaps no such factory will exist; and even if there is such a factory and Alice goes to it, *what about all the workers she leaves behind?* The existence of an insulation factory that has clean air does not justify the prolonged exposure of employees at ABC Corporation to asbestos. The fact that Alice originally "agreed" to work in the factory does not make the asbestos in the air any less harmful. There are wounds inflicted upon the body and there are wounds inflicted upon the soul. It so happens that in both of these cases, the workplace is tainted so as to take your breath away.

So, in the case of overreaching censorship policy the following has been shown: the argument that the overreaching censorship policy is ethical since Alice is free to work some place else is fallacious. It *distracts* the listener away from the harmful work environment at Alice's company by suggesting that a

company without a harmful work environment might exist.

The above Kantian universalization argument was applied to an institution instead of an individual. Kant argued that a person should only act on maxims that are universalizable (p. 118 of [51]). Some may therefore question the applicability of a Kantian universalization argument to an institution instead of a person.

In regards to corporate personhood, if one is willing to bestow the character and properties of individuality to an institution and the rights thereunto, then one should be equally determined to hold the corporate person (i.e., institution) accountable to the highest moral standards in every case and without exception. Put another way, if we accept as valid corporate personhood, then the corporate person (i.e., institution) must be held accountable to Kantian universalization arguments.

This rationalizes the application of Kantian universalization to ABC Corporation and therefore the finding that ABC Corporation is in violation of the human right to free speech. This type of rationalization may become increasingly important as corporate personhood expands to cover ever more civil rights. Corporate personhood has been resurfacing in American law for a long time [48]. In Citizens United v. FEC in 2010 the majority ruled that corporations, by virtue of being associations of individuals, have free speech rights under the First Amendment. It is not inconceivable that corporate personhood might one day extend to cover the Fifth Amendment.

2.6 Devil's Advocate: Reasons Against Free Coding

To present a balanced view it is necessary to play devil's advocate and present reasons for restricting free coding. These reasons are broken down into general reasons, institutional reasons, and "cyber defense" reasons.

2.6.1 General Reasons Against Free Coding

Malware source code presents an interesting challenge to free coding since it is code that is designed to cause harm. In playing devil's advocate it is therefore instructive to consider the freedom to publish malware source code.

In a secure ransomware attack, a cryptovirus, cryptotrojan, or cryptoworm hybrid encrypts the victim's data using the attacker's public key that is embedded in the malware, thereby denying the victim access to his or her own data. Assuming there are no backups, only the attacker can recover the victim's data since only the attacker has the corresponding private decryption key. This is called cryptoviral extortion. Moti Yung and I invented cryptoviral extortion, i.e., the secure ransomware attack, and presented it at the 1996 IEEE Security

& Privacy conference [57]. This same paper noted the possibility of extorting electronic money as ransom payment from malware attacks. This allows the attacker to be paid remotely and anonymously. This was long before Bitcoin existed. Ransomware is a scourge that extorted approximately 1 billion dollars from victims in 2016 [25]. Ransomware is one attack of many in an area of research known as Cryptovirology.

Arguably, one of the most extreme cases of writing code and giving it to society is writing ransomware and giving the ransomware code to society. A case in point is a coder that writes and publishes a cryptographically sound cryptoworm that successfully evades detection. One might infer from this that one should censor contributions of code to society.

I disagree. Controlling code contributions to society via censorship is a slippery slope. Soon after the publication of the cryptoviral extortion attack in 1996 some people insisted that no victim would ever pay the ransom. Others insisted that the attack had no utility beyond simply deleting the hard drive. Still others could not mentally get past the new terminology of cryptovirology itself and dismissed the warnings forthwith. A skeptic was encountered who did not believe that the protocol constituted a real threat, dismissing it as being only "theoretical." This dismissal prompted the further explanation that the original IEEE S&P paper on cryptoviral extortion also detailed a real world experiment that proved that the attack works. The experiment, conducted in 1995, demonstrated the first cryptoviral extortion attack. It was carried out on a Macintosh SE/30 computer and it used RSA and the Tiny Encryption Algorithm. It was only after explaining this real-world experiment that the skeptic believed. It is a fact that some people simply do not take a malware threat seriously unless you can show them the code of the malware. period.

On the whole, industry was not motivated to develop and deploy defenses against cryptoviral extortion until attacks that demanded Bitcoin were well underway. In short, the threat of cryptoviral extortion was largely ignored for two decades. There is no substitute for the awakening that demonstrably effective attack code provides.

Another challenge to free coding involves the publication of software exploit code. A vulnerability researcher who finds a vulnerability will often research the degree to which the vulnerability can be exploited. Whether or not a vulnerability is exploitable can be confirmed by attempting to write functional exploit code. For example, exploit code may demonstrate the ability of the attacker to abuse the vulnerability to execute arbitrary code of the attacker's choosing on the victim's machine. Functional exploit code can be weaponized to deliver malware to the target system.

A software vulnerability can be devastating. The Shadow Brokers blew the whistle on the NSA by publishing MS Windows exploits that the NSA hoarded

[49]. Once this exploit code was exposed an adversary leveraged it to create and release the WannaCry cryptoworm (ransomware). Functional exploit code can place lives in danger, e.g., when transportation systems are vulnerable. One might conclude from this that code contributions to society should be censored since exploit code can cause harm.

Again I disagree. The social norm in this case is for the vulnerability researcher to engage in responsible disclosure, giving the author of the software a 90-day window with which to craft a fix and release it prior to publishing the vulnerability.

Finally there is the conceivable apocalyptic scenario in which a pillar of modern cryptography topples over. A cryptanalyst could conceivably discover a probabilistic polynomial time algorithm that solves the integer factorization problem. This would break the RSA cipher overnight. If the cryptanalyst published the code that breaks RSA, this would expose an inconceivable number of computer systems to risk: on-line shopping systems, on-line banking systems, secure e-mail servers, VPN servers, software update systems, and so on.

Once again I argue that even this provides an insufficient basis for censoring contributions of code to society. Responsible disclosure should again be considered by the researcher, though a window longer than 90 days may be advisable.

In addition to publishing code there is the issue of publishing critiques of code. A coder could publish a critique of source code that carries out the cryptoviral extortion attack that explains how to make the attack more effective. I argue that this should not be censored for the same reason that publishing source code for ransomware should not be censored. Similarly, being free to publish descriptions of vulnerabilities and descriptions of software exploits is important to convey the gravity of the associated risk and enable countermeasures to be devised and put into place.

In the foregoing, risks associated with publishing code and publishing vulnerabilities were considered. In all cases I find censorship unjustifiable.

2.6.2 Institutional Reasons Against Free Coding

Industrial pre-publication censorship is a catch-all that aims to prevent the release of information that, if disclosed, could negatively impact the institution. This information includes sensitive business practices, processes, data, patentable ideas, trade secrets, and information that falls under copyright. The issue is when the censorship is overreaching.

For example, consider an on-line game company that houses internal servers for a medieval on-line role playing game. Such a company would have a substantial code base. An employee of this company that contributes code to a competing commercial on-line role playing game presents an obvious and logi-

cal conflict-of-interest. In this case, institutional pre-publication censorship is a logical approach to preventing such conflicts of interest.

However, there is no need for the censorship policy to be overbroad. It should apply only to those types of code bases that are deemed to present a conflict of interest. For an approach to balancing pre-publication censorship with free coding, see Section 1.4.

The publication of software vulnerabilities presents a potential conflict of interest for institutions that produce cyber arms. This case is addressed in the next subsection.

2.6.3 "Cyber Defense" Reasons Against Free Coding

Intelligence agencies, the military, and law-enforcement leverage security vulnerabilities to gather intelligence, win wars, and enforce the law. In their eyes, allowing a member of their community to publish a secret vulnerability that is "strategically" exploitable risks "aiding the enemy" or closing off access to target systems should the vulnerability be fixed in them.

Insofar as these three missions are concerned, contributing code to society and publishing critiques or documentation of code that has been given to society may be construed as negatively impacting their missions depending upon the circumstances. For example, publishing source code that implements a secure cipher hampers wire-tapping; publishing polished documentation that makes a crypto library easier for programmers to use hampers wire-tapping; publishing an exploitable vulnerability may cause it to be fixed, thereby preventing "strategic" exploitation.

Security systems can be proactive or reactive. A proactive security measure places locks on all the doors to a house. A reactive security measure places video cameras at all the doors to identify would-be robbers that might break in. We can spend great time and effort improving our ability to spy on one another, or we can spend great time and effort improving the locks on our doors. I prefer the latter.

Making sure that there are no obstructions to free coding is a proactive security measure. Having the codebase for society be free of vulnerabilities is a matter of national security.

2.7 The Philosophy of Coder Consciousness

The writer expresses himself through words. The musician expresses herself through music. The painter expresses himself on canvas. The dancer expresses herself through motion and the coder expresses herself through code. Yet, the industrial coder is made to feel that her contributions of knowledge to society

are a liability, and, by virtue of wanting to contribute knowledge to society, that she herself is a liability by extension.[13] It is at this stage that the chilling effects of pre-publication censorship are complete. The coder has reached the docile state of silence and knowledge is *self* restrained. The coder, by way of institutionalization, is led to believe that she is producing machinery as opposed to expressing herself through code. Given enough time, she becomes a cog in the machine itself.

I endeavor to raise awareness of this issue, as it has far reaching consequences, from restraining the amount of source code given to society to diminishing the information assurance of source code that has been given to society. Software exploits facilitate the oppression of all manner of people in all manner of situations: due to the color of their skin, their religious beliefs, their political beliefs, and their walk of life, e.g., journalists and activists. Restraints on free coding call for a new philosophy to counter them. I call this new philosophy "coder consciousness."

Coder consciousness is inspired by the concept of Black Consciousness (Chp. 11 of [4]).[14] Steve Biko was the father of the Black Consciousness movement.[15] Coder consciousness takes into account multiple obstacles to free coding. Three of these categories of obstacles were defined by Biko. They are as follows: the emptiness of the past, the dependence of the oppressed on their oppressors, and traditional complexes (Chp. 11 of [4]). However, I refer to the 'traditional complexes' category as 'corporate personhood' to reflect a particular complex that results from corporate personhood. The fourth obstacle, the use of incorrect terminology, was observed by Confucius who sought to restore a rationalized feudal order at a time when the feudal system of the Chou Dynasty was falling apart (p. 6 of [58]).

Obstacles to free coding:

1. **The Emptiness of the Past**: For the young coder entering the workforce today, the beginning of time was the 1980s when NDAs were first leveraged to restrict the sharing of code and treat code as property instead of speech. Industrial pre-publication censorship further restricted the flow of code. The young coder is unconscious of the fact that before the 1980s code was freely shared gratis. In South Africa, through the history books of the Anglo–Boer colonists and their associated emphasis on the european way of life, the colonists de-emphasized, to the point of forgetfulness and distortion, the history of Black South Africans. The modern coder

[13] This mirrors Biko's liability argument relating to oppressors (p. 68 of [4]).

[14] Chp. 11 is a paper that was later banned (p. 56 of [55]), written by Biko when he was a student.

[15] From the Preface of [4] by Archbishop Desmond-Tutu.

suffers this very same phenomenon. *Without knowing that code used to be universally shared gratis and vulnerabilities used to be perceived as defects as opposed to a means of exploitation, coders are at a disadvantage in conceptualizing a world where code and descriptions of vulnerabilities are speech.*

2. **Dependence on the Oppressors**: A coder that is dependent upon wage income must obey the policies of his employer or risk being fired. The more institutions there are that obstruct free coding, the harder it is for the coder to find an employer that supports free coding. *Dependence causes fear. Once steeped in fear, speech is chilled.*

3. **Corporate Personhood**: The desire of the corporate person to control contributions of knowledge to society, rooted in claims of "property" ownership, is at odds with the desire of the individual to exercise free speech in order to contribute knowledge to society. *Industrial policies and procedures commonly obstruct free coding.*

4. **Incorrect Usage of Terminology**: The continued use of improper terms perpetuate the causes against free coding and cause people to overlook obstructions to free coding. Policies that characterize "business ethics" as being the highest ethical standards while at the same time disregarding the human right to free speech conceal and soften injustices against the individual. *Words shape the way coders view the world. Incorrect terminology distorts that view and puts a pleasant face on policies that violate human rights.*

Plato's *allegory of the cave* addresses people who are doomed to perceive a false reality (p. 17 of [36], p. 48 of [12]). It is an allegory that applies equally well to the young coder today. This new cave is man-made in every respect, the construction of which began in the 1980s. The prisoners are young coders born chained in the cave, forced to gaze at the shadow puppets on the wall in front of them and not turn around. The puppet masters correspond to the system that institutionalizes the pursuit of silver and gold, and the shadow puppets on the wall are the prisoner's reality that condones software and vulnerability hoarding. To achieve coder consciousness is to break free from the cave and reach the sunlight. The benefit of teaching the history of *freedom*, as it relates to code, is that it instills within the coder a sense of what has been, giving hope to the prospect of reconstructing a world of sharing code and exposing vulnerabilities instead of a world where software and vulnerabilities are hoarded.

Free coding oppression is one of many forms of oppression. The work of Johann Fichte provides a framework for understanding social oppression in its

various forms. Johann Fichte leveraged *thesis, anti-thesis, and synthesis* as a formula for the explanation of change (p. 46, note 37 of [53]). They are the cardinal points around which social revolution revolves. Biko used this triadic formula to characterize the white liberal view of oppression in South Africa: the thesis being institutionalized racial segregation, i.e., apartheid, the anti-thesis being non-racialism, and the synthesis being weakly defined, involving the formation of non-racial groups. Biko argued that this is an incorrect characterization of the problem. According to Biko, the correct characterization follows from Black Consciousness (p. 90 of [4]), that the thesis is strong white racism. Without correctly characterizing the social problem, one risks leveraging a hopelessly weak approach to solving it.

It is instructive to apply Fichte's framework to the problem of free coding oppression. Inspired by Biko's model, it helps show both extremes and proposes a method to go from a state of oppression to a state of freedom. My triadic characterization of the problem of knowledge hoarding, as it pertains to free coding, is as follows:

Thesis: Knowledge hoarding

Anti-thesis: Knowledge sharing

Synthesis:

1. **Conscientisation**: For the coder and vulnerability researcher: conscientise your peers to coder consciousness.[16] This will help them achieve coder consciousness. Conscientisation is not just a nice idea. It is a real, tangible process that people can follow to cause free coding to be upheld. In the past the world did not hoard software and the world did not hoard vulnerabilities. Help people envision a world in which there is no such hoarding. If people cannot envision it then it will not happen.

2. **Responsible Disclosure**: For the person that is free to share an instance of knowledge with society but is afraid that it might cause physical harm: indeed, there are situations in which the sharing of knowledge can cause physical harm in the short term. This is a manifestation of the *harm principle* put forth by John Stuart Mill. Responsible disclosure prior to publication may be advisable. But in the long run it should be possible to share the knowledge.

3. **Free Coding License**: For the person that is writing code and giving it to society: consider using a license that upholds free coding. An exam-

[16] Biko used the term conscientise (p. 114 of [4]).

ple license term that upholds Freedom B fashioned around vulnerability publication was given in Section 1.4.

4. **Institutional Policy Publication**: For the person that is in a position to establish institutional policies and procedures: publish the pre-publication censorship policy of the institution. Presumably it upholds the highest ethical standards. Doing so will allow coders to evaluate the restrictions on their freedom before they decide to join the firm. Industry-wide surveys will allow students to evaluate the breadth of restrictions on free coding in industry before deciding to become coders.

5. **Free Coding Policy**: For the person that is in a position to establish institutional policies and procedures: consider using the free coding policy from Section 1.4. The free coding policy lets society know that the firm at least acknowledges the importance of coding liberty.

Coder consciousness is not a matter of pride. It takes into account that final day of judgment in which one stands *alone*.

Coder consciousness is: awareness of being gifted as a coder with the responsibility to use the gift to help one's neighbor, exalting the freedom to give knowledge to society, exalting the freedom to use public knowledge, and knowing in one's heart that code is speech that reflects the very essence of one's soul.

Being a coder is a gift that can be leveraged to help others. This is why obstructions to free coding need to be overcome. Vulnerable code leads to exploitation, oppression, torture, and murder; people living in fear, lost sheep. God does not like it when his sheep are mistreated.

In a letter to Harrison Blake, Henry David Thoreau wrote (p. 419 of [47]):

"These are the regions of the Known and the Unknown. What is the use of going right over the old track again? There is an adder in the path which your own feet have worn. You must make tracks into the Unknown. That is what you have your board and clothes for."

2.8 Conclusion

The farmers at present wear but different shoes, the work of the foot having been replaced by the work of the mind, their harvest effortlessly reproduced. Is their yield tangible still? To perceive this as an awakening is folly. For it is now the farmer himself who is farmed, having likewise been planted row upon row, oriented and pruned using implements that glisten with exactness. The tangible crop is renewed.

Can the life long magistrate fairly conclude for the farmer that his plow is too dull for his tillage? From what finely sharpened experience or enlightening book comes such revelation? To say that code is not speech is like the dwarf saying to the elf that his runes are but gibberish; and so, according to elvish ears; to bypass the apprentice, to seek only the spell that is cast; and to not, in the fray of dancing candlelight against advancing shadow, marvel at the iridescent glyphs that lift clear off the page.

To set upon the road to coding liberty is to loose the wayward girdle, to put to rout the walls that divide, and to join the symphony of the giving.

Chapter 3

On Lady Wisdom and Knowledge Hoarding

Free coding embodies the freedom to give knowledge to society. In particular, it consists of Freedom A and Freedom B. Freedom A is the following: you have the freedom to write code and give it to society under conditions of your choosing. Freedom B: you have the freedom to write and publish, under conditions of your choosing, a critique or documentation of code that has been given to society. These freedoms are restricted by practices such as industrial pre-publication censorship. Restrictions on free coding form a subset of the larger problem of knowledge hoarding. In this chapter I present my findings on the problem of knowledge hoarding and establish a theological foundation of free coding. Writing this chapter involved extensive research and deep spiritual introspection, filled one day with tears and the next with dreams of the most awe-inspiring kind. In particular, I convey my interpretations of Genesis 2 and 3, namely, the Original Command and the Original Paradox. This leads to what I believe is *the root* of the problem of knowledge hoarding. My conclusion: that humankind was explicitly created by God and Lady Wisdom, that Knowledge is sacred, and that only by raising Knowledge to the stature of Life can humankind solve the problem of Knowledge hoarding.

3.1 Introduction

The problem of knowledge hoarding is getting steadily worse for society. Source code has transitioned from being regarded as speech to being regarded as the property of people. Software vulnerabilities have transitioned from being regarded as defects to being regarded as the property of people. The types of knowledge that people and institutions believe that they own is expanding.

A key question then, is this: what is the root of the problem of knowledge hoarding? There are multiple approaches to try to find the root of the problem, for example: Eastern philosophy, Western philosophy, religious dogma, and

Published through the viXra.org e-Print archive Nov. 2, 2017.
viXra citation number: viXra:1711.0117. This version contains updates as of Feb. 15, 2018.
This manifesto was submitted to the Cornell University arXiv e-Print service on Oct. 29, 2017 under the category "Computers and Society," was assigned identifier submit/2052823, and was rejected by Cornell University on Nov. 15, 2017 by the moderators "who determined it to be on a topic not covered by arXiv." For the latest version go to www.coderfreedom.org.

religious mysticism. These approaches are by no means mutually exclusive. Anselm of Canterbury approached issues of faith using reason. He believed that metaphysical arguments from Latin and Greek classics could provide a deeper understanding of God's nature (p. 189 of [36]). This manifesto leverages all of these approaches to varying extent.

When one goes to Church or to Temple, one is exposed to a combination of pure Scripture and dogma. Someone who is raised going to Church may not think to ever question the doctrines of his or her denomination. Western society teaches one to delegate the interpretation of the Scriptures to "religious experts." The division of labor in modern society is taken to such an extreme that one is no longer encouraged to individually seek Wisdom in Scripture as for silver and gold. So, if the answer is not to be found in existing dogma then the answer is not likely addressed by religious establishments.

Hope may lie in the mystical tradition: in Judaism there are Kabbalists and Hasidim. In Christianity there were the Gnostics. The mystics bring a creative element to religion, appealing to dreams and visions to understand the metaphysical realm. Throughout history mystics have collided head-on with ecclesiastical institutions. Does the path of religious mysticism hold the key to understanding the root of the problem of knowledge hoarding?

With an understanding of the root of the problem of knowledge hoarding society can begin to chart a course out of the storm. In the sections that follow I present what I believe to be the root of the problem of knowledge hoarding. It is my deepest hope that this problem will be treated with the urgency it deserves. The freedom of future generations depends on it.

3.2 Religion and Knowledge Hoarding

The unification of church and state as well as the unification of church and school has been a significant force of oppression throughout human history. Secularization has occurred, causing religion to lose both cultural and social significance. Today, when someone starts preaching many tune out altogether.

The point here is not to pass judgment on the separation of church and state, of church and school, and of church and workplace; but, to simply acknowledge it for what it is. Westerners are wont to place religion upon a pedestal, somewhere distant and out of sight, to be addressed only when it is deemed safe to do so. This phenomenon has existed for generations and it has affected the relationship between the Westerner and her religion. Charles Taylor asks (p. 25 of [45]):

"Why was it virtually impossible not to believe in God in, say, 1500 in our Western society, while in 2000 many of us find this not only easy, but even inescapable?"

Taylor offers these reasons why God's presence was seemingly undeniable in the past: (1) natural events such as floods and plagues were seen as acts of God but are now a dead metaphor as modern legal language attests, (2) the various associations of life including guilds, boroughs, parishes, and so on were interwoven with ritual and worship, and (3) people lived in an "enchanted" world but now live in disenchantment (p. 25 of [45]). He argues that postmodernity and scientism fuel skepticism and tend to obscure our appreciation for the larger "constitutive good" (p. 49 of [36]).

This accounts for secularization in society as well as secularization in the workplace. But, the story does not end with mere secularization. *Corporate personhood* has evolved well past a company having the rights of an individual, being regarded as a person. It is now being openly stated that a company has a soul [6]. As a Christian I find this cavalier use of the word soul highly offensive.

Part and parcel to the phenomenon of secularization is the tendency of the Westerner to delegate the interpretation of the Scriptures. Steve Biko observed a tendency by Christians to make interpretation of religion a specialist job (p. 58 of [4]). This limits those who vigorously search for Wisdom within the bible.

Steve Biko also pointed out the critical relationship between religion and the oppression of his people. In addressing this, he wrote (p. 59 of [4]):

"Christianity can never hope to remain abstract and removed from the people's environmental problems. In order to be applicable to people, it must have meaning for them in their given situation. If they are an oppressed people, it must have something to say about their oppression."

This leads to the following question: is there a Judeo-Christian foundation for knowledge hoarding and if so what is it? Having an understanding of the root of the problem is critical in addressing it. Henry David Thoreau wrote, "There are a thousand hacking at the branches of evil to one who is striking at the root" (p. 78 of [46]). If this manifesto only addressed the utilitarian aspects of free coding, e.g., relating freedoms to security (see Table 1.1) then the manifesto would only be hacking at the branches. To address this problem fully it is necessary to identify the root of the problem. Hacking at the branches of evil is futile over the long term.

The path forward dives head first into religion, leveraging Western philosophical concepts along the way. In some cases Western philosophy is leveraged to reason about parables in the Bible. In other cases it is used to show the chasm between rational thought and pure faith.

The word 'philosopher' is from the Greek words meaning 'love of Wisdom' (p. 3 of [51]). René Descartes proposed the method of *skeptical doubt*: that in order to identify foundational truths, a skeptical attitude towards everything

must be taken.[1] Put another way, all presumed knowledge must be brought into question (pp. 40-41 of [36]). The modus operandi of the philosopher is to question everything. Everything is questionable by everyone. Everything is subject to re-interpretation. Everything is subject to being understood better. In seeking a better understanding there are no boundaries that the lover of Wisdom mustn't cross.

So, for the moment, let us dispense with the Western bias towards believing that only religious experts are qualified to interpret the Scriptures. Let us dive straight into the Old Testament with the problem of "knowledge hoarding" at the forefront of our minds and seek connections throughout. Let us heed Lao-tse from the East, empty our cups, and flow like water, for water does not tangle on anything on its way down, and in so doing, free ourselves from *every* prior church doctrine. Let us bring with us the rational thought from the West that has developed over the centuries and use it to challenge existing dogma in our return to the age of enchantment, and, in so doing, attempt to discern God's position on knowledge hoarding.

"For the sages, the knowledge of God came through the tradition of the ancestors; observation of the cosmos, society, and human experience; reflection on and remembrance of sapiential teachings and experience; rational analysis that was in part related to the understanding that there were connections between elements in nature; and the activation of the imagination by means of key metaphors"

—Leo G. Perdue (p. 65 of [35],pp. 73-76 of [34])

3.3 Lady Wisdom

3.3.1 Lady Wisdom Throughout the Ages

Goddess Asherah was the first female deity known to have been worshiped by the Children of Israel. For about six centuries after the arrival of the Israelite tribes in Canaan, down to the destruction of Jerusalem in 586 B.C. the Hebrews worshiped Asherah (p. 34 of [33]). She was the chief goddess of the Canaanite pantheon and her full name was "Lady Asherah of the Sea." (pp. 36-37 of [33]). In Ugaritic mythology she figured prominently as the wife of El, the chief God (p. 37 of [33]). Worship of Asherah was introduced into the Jerusalem Temple by King Rehoboam, the son of Solomon, in or about 928 BC. Her statue was removed and re-introduced multiple times into the Temple over an extended period of time (p. 50 of [33]).

Gods are rarely invented or discovered and instead are often taken over by one

[1] Also called *Cartesian doubt*.

group from another. Raphael Patai posits the question: were Asherah, Astarte, and others Hebrew goddesses or merely foreign abominations as characterized by the prophets? But, then he goes on to state: "There can be no doubt that the goddess to whom the Hebrews clung with such tenacity down to the days of Joshiah, and to whom they returned with such remorse following the destruction of the Jerusalem Temple was, whatever the prophets had to say about her, no foreign seductress, but a Hebrew goddess" (p. 32 of [33]).

Patai further asserts that the Hebrew Goddess underwent an astounding metamorphosis into a manifestation of God's presence, the Shekhina, a feminine name just as God's is masculine (p. 32 of [33]). Shekhina is a Hebrew abstract noun meaning "the act of dwelling" and in actual usage means that aspect of the deity which can be apprehended by the senses (p. 99 of [33]). The Shekhina is identically the "Matronit," the Matron, Lady, or Queen (p. 32, 252 of [33]).

Over time the Shekhina gained strength and in the 13th century A.D. developed in Kabbalism as a distinct Hebrew Goddess who often confronted and sometimes even opposed God (p. 32 of [33]). Mysticism daringly spoke of Shekhina in most deeply revered sources as the spouse of God (p. 20 of [33]).

Raphael Patai affirms that the Hebrew Goddess is none other than Asherah; regarding the Hebrew Goddess and Israel: "Is the Hebrew Goddess dead, or does she merely slumber, soon to awaken rejuvenated by her rest and reclaim the hearts of her sons and lovers?...And it was there—less than 400 years ago!—that her Rachel and Leah forms revealed to the pious and the learned the divine meaning of earthly love, the last of her great motherly-wifely acts, and that her identity with the ancient Biblical goddess Asherah was recognized in a remarkable flash of intuitive insight." (p. 33 of [33]).

Lady Wisdom also features prominently in Jewish and Christian religion, appearing in the Book of Proverbs that is in both the Torah and the Christian Bible. A subset of the major themes of this book include divine creation and providence, the importance of the acquisition of Wisdom, and the power of the word with a particular emphasis on elegance of expression and persuasion (p. 48 of [35]). Proverbs 3 tells of the happiness that comes to the person who finds Wisdom. It also portrays Wisdom in the "guise" of an ancient Near Eastern goddess of life and covers Wisdom's role in creation (p. 49 of [35]). Leo Perdue writes, "Finally, Wisdom, associated with God, is often a personified and eventually hypostatized attribute seen in the activities of Woman Wisdom, originally a goddess in Israelite religion prior to the development of monotheism, and then a personified metaphor." (p. 30 of [35]).

Familiarity with the Book of Proverbs is needed to place the Book of Genesis in perspective since Proverbs describes the relationship between God and Lady Wisdom before the creation of humankind. Lady Wisdom said, "The Lord possessed me in the beginning of his way, before his works of old" (Proverbs

8:22 KJV). Regarding the tree of knowledge in the Garden of Eden, Raphael Patai writes,

"It is consonant with this terrible aspect of the Shekhina-Matronit that her old Talmudic role of death bringer is also remembered and revived in the Zohar, which repeatedly asserts that the words of the Book of Proverbs (5:5) "Her feet go down to death" refer to the Shekhina, symbolically represented by the forbidden tree which for Adam was a "tree of death"." (p. 150 of [33])

Proverbs centers around Lady Wisdom. This therefore implies that the Shekhina-Matronit is the forbidden tree who is Lady Wisdom. Proverbs directly equates Lady Wisdom with a tree of life: "She is a tree of life to those who hold her fast and blessed are those who look for her" (Proverbs 3:18 [2]). Wisdom as a tree is also supported by Proverbs 8: "My fruits are better than refined gold and my produce than choice silver" (Proverbs 8:19 [2]). Leo Perdue ties together the tree of life, the tree of knowledge, Asherah, and Goddess Wisdom in the following:

"In the second strophe (v. 18), Wisdom is personified as a "tree of life" (see Gen. 2:9; 3:22, 24), a major symbol of fertility goddesses in ancient Near Eastern religions... The tree of life in Gen. 2:9; 2:16-17; 3:11; 3:22-24 (...) is associated on occasion in the Hebrew Bible with wooden poles or sacred trees (...) a common symbol for the fertility goddess Asherah, worshiped in Israel and Canaan. In this proverbial poem, however, the tree of life and the tree of knowledge of good and evil (an expression for Wisdom) are merged into Goddess Wisdom." (p. 50 of [35]).

The foregoing draws on the Bible and the work of Raphael Patai and Leo Perdue. The findings imply that goddess Asherah, the Hebrew Goddess, the Shekhina, the Matronit, the tree of knowledge, and Lady Wisdom are one and the same.

According to Jewish Mysticism, the Shekhina, in Her motherly love, went into exile with Israel following the destruction of the Jerusalem Temple and that, accordingly, the Shekhina and God are separated. There exist multiple rites that the mystics believe will help unify the Shekhina with God. In particular, the unification of wife and husband is believed to be a way to bring God and his Shekhina closer together. The implication of this belief is scary, since conversely, it implies that the sins of the Children of Israel further separate God from His Shekhina, *that our sins have a negative effect on heaven.* The belief that the union of wife and husband helps bring together God and His Shekhina is incredibly heart-warming.

3.3.2 Lady Wisdom Owns Knowledge

Wisdom is not a "what." She is a "who," and not just any "who." She is the divine Lady who was by God's side at the beginning. She was there when God created heaven and earth. Wisdom said, "I was fashioning with Him; He was rejoicing in me everyday, and I have been rejoicing before Him always." (Proverbs 8:30 [2]). "I, Wisdom, have created cunning and I own knowledge and reason" (Proverbs 8:12 [2]). Her arms are open to all people. She said, "I love those who love me, and those who seek me will find me" (Proverbs 8:17 ISV).

Lady Wisdom *owns* knowledge. Institutions do not own knowledge. Men have wasted no time in taking ownership of everything imaginable: from slaves, to wives, to parcels of land, to source code, to exploitable software vulnerabilities. So, if it helps, one might feel at ease regarding Wisdom as the owner of the tree of knowledge. Or, one might accept the characterization that the tree of life and tree of knowledge are merged into Goddess Wisdom (p. 50 of [35]).

3.4 Established Doctrines on the Garden of Eden

In the midst of the Garden of Eden stood the tree of life and tree of knowledge of good and evil. Adam and Eve were permitted to eat of the fruits of all trees in the Garden of Eden except for the fruit of the tree of knowledge, the forbidden fruit. Adam was told that if he eats the forbidden fruit he will die the same day (Genesis 2:17 KJV). Eve had a dialog with the serpent and decided to take fruit from the tree of knowledge. Against God's will, Adam and Eve ate the forbidden fruit. After doing so Adam and Eve were cast out of the Garden of Eden and a flaming sword was placed to guard the way of the tree of life. To eat of the tree of life is to live forever (Genesis 3:22 KJV).

Original Sin is the doctrine that humanity is in a state of sin resulting from Adam and Eve's disobedience of God by eating of the tree of knowledge. The doctrine places a strong emphasis on collective guilt. In its most extreme form it characterizes total depravity of humankind.

On a lighter note, Saint John Chrysostom (349–407 AD) interprets the banishment of Adam and Eve as a gracious act of God. Chrysostom believed that had Adam and Eve stayed in the Garden of Eden and eaten of the tree of life and become immortal then they would have been stuck in disobedience and arrested development forever (p. 81 of [40]). Saint John Chrysostom wrote:

"So when partaking of this tree he became liable to death and subject in the future to the needs of the body, and entry of sin had its beginnings as the result of which death also was fittingly provided by the Lord, no longer did he allow Adam in the garden but bade him leave there, showing us that his sole motive in doing this was his love for him." (Homily 18 part (8) [7]).

Saint John Chrysostom therefore attributes the banishment of Adam and Eve to love alone. This casts a positive light on Adam and Eve.

Another positive interpretation of the Genesis is due to the medieval Christian mystic Hildegard von Bingen (1098–1179 AD). Hildegard believed that people are born with Original Wisdom and that it manifests in children like a folded tent. The tent expands as a child grows and finds her home in Wisdom's tent (p. 103 of [40]).

Creation Spirituality is an ancient tradition, named and conveyed with vigor by Matthew Fox beginning in the 1970s. Fox believes that we are all sons and daughters of the Divine and are original blessings.[2] Danielle Shroyer defines Original Blessing as this: we are steadfastly held in a relationship with God and that He calls us beloved and good before we are anything else (p. xi of [40]). Fox explains how Hildegard von Bingen's "Original Blessing" theology was ultimately overshadowed by the Original Sin doctrine (p. xxiii of [19], see also [14]).[3]

Danielle Shroyer presented compelling analogies regarding the harmful effect of setting up our relationship with God in negative terms, an effect caused by the doctrine of Original Sin. She wrote: "Have you ever been to a wedding ceremony where the minister began by describing all the ways the two people are completely incompatible and dissimilar?" (p. 34 of [40]). One of her more jocular observations is as follows: "I've lost count of how many times I've read that original sin is an effective bonding agent between humans, like we're all members of the same classroom detention." (p. 44 of [40]). The harmful chasm between God and us that results from the Original Sin doctrine survives to this day.

Why has the doctrine of Original Sin developed such a strong foothold? Perhaps one of the reasons is that it facilitates the pursuit of silver and gold. South Africans had no notion of hell prior to the arrival of the Anglo–Boer colonists (p. 44 of [4]). The version of Christianity pushed upon South Africans scared mother and father, invoking visions of eternal flames, the gnashing of teeth, and the grinding of bone (p. 45 of [4]). Biko explains how Westerners leveraged this fear to establish themselves as the perennial teachers and twist South Africans away from their traditions that included a sacred belief in sharing. Soon after the arrival of the colonists the people were divided into two groups, the converted and the pagans (p. 56 of [4]). The Anglo–Boer culture had all the trappings of colonialist culture, honed for conquest. Where possible, they conquered by persuasion, using a highly exclusive religion that denounced all other Gods (p. 41 of [4]).

[2] www.matthewfox.org/what-is-creation-spirituality

[3] Matthew Fox coined the term "Original Blessing" (p. 215 of [40]).

The following doctrines were covered: Original Sin, Original Wisdom, and Original Blessing. However, none of these focus on God's original command to Adam not to eat of the tree of knowledge. This is the subject of the next section.

3.5 The Original Command

I propose the following interpretation of God's Original Command to Adam: one should not hoard knowledge.[4] I now lay the groundwork for this.

The Good News Bible describes the tree of knowledge as "the tree that gives knowledge of what is good and what is bad" (Genesis 2:9 GNT). In a footnote the Good News Bible expands on the definition of this tree: "knowledge of what is good and what is bad; or knowledge of everything." This may be an instance of a merism,[5] such as someone who has "searched high and low," i.e., everywhere. I am of the opinion that the tree of knowledge is the tree of knowledge of everything. In the case of Adam and Eve, hoarding knowledge is the literal act of taking and eating fruit from the tree of knowledge. Did the fruit from the tree of knowledge contain seeds and if so what became of them?

In the Westminster Commentaries, S. R. Driver gives the following references to the tree of life in Proverbs: 3:18, 11:30, 13:12, and 15:4 (p. 39 of [11]). "The fruit of the righteous is a tree of life and the souls of the evil will be removed" (Proverbs 11:30 [2]). "A man who begins to help is better than he that props up with hope, and the tree of life brings hope" (Proverbs 13:12 [2]). "The tree of life is the healing of the tongue, and he that eats its fruit will be filled by it" (Proverbs 15:4 [2]).

One can imagine the water of Life flowing up from the ground, through Lady Wisdom's arms,[6] ever forking off in ever new directions, into long narrowing tunnels that open to the Light,[7] the water of Life flowing through each soul along it's journey to the Light, the final stretch traversed alone,[8] giving rise to

[4]The "Original Command" is not to be confused with the first commandment. "Yeshua said to him, "You shall love THE LORD JEHOVAH your God from all your heart and from all your soul and from all your power and from all your mind." (Matt 22:37 [2]). "This is the great and the first commandment." (Matt 22:38 [2]). For comparison: "This is the greatest and the most important commandment." (Matt 22:38 GNT). "Original" in "Original Command" is meant to signify the origin, i.e., the early part of Genesis.

[5]en.wikipedia.org/wiki/Tree_of_the_knowledge_of_good_and_evil

[6]"The Lord possessed me in the beginning of his way, before his works of old." (Proverbs 8:22 KJV)

[7]The view along the true meridian fixes upon the Northern star.

[8]The near-death experience of Joe Hyams: "Twice during that time my fever reached 106°, and the doctors told Elke they had lost me. Of those moments, I recall only floating in a cocoon of warmth down a narrow tunnel where I would be free of pain. I could hear Elke's voice from a distance pleading with me not to die." (p. 70 of [20]).

a new sacred fruit, with the whole tree reaching up to the Light, ascending on toward Heaven. "The Way of Life is an ascent to the understanding One, to turn away from Sheol beneath" (Proverbs 15:24 [2]). In this sense, the water of Life forms the tree of Life within the tree of Knowledge, Lady Wisdom; the water of Life flowing in unison with the tree of Knowledge as it grows.[9] According to the Gospel of Thomas, Jesus said, "Split the wood—I am there; lift the stone and you will find me there." (Reading 11 of [15]). A theological interpretation of the Book of Proverbs characterizes Wisdom as providing the link between the Creator and His creation, mediating between heaven and earth (p. 57 of [35]). One may think of her roots in the ground and her branches reaching up to heaven as providing this link.

One can imagine the roots of Lady Wisdom running deep into the ground. As a result of Adam eating the forbidden fruit, God cursed the ground (Genesis 3:17). With this view, a soul that chooses to spurn Lady Wisdom, instead of rising up and forming a branch, travels downward through a root to Sheol. "And her legs marry to death; her walk causes men to recline in Sheol" (Proverbs 5:5 [2]). "And those who sin against me harm their soul, and all who hate me are the friends of death" (Proverbs 8:36 [2]). Under this view of Lady Wisdom, a soul has a choice: flow in the water of Life up to Lady Wisdom's crown and on to heaven or through her roots down to Sheol.

The setting: in the center of the Garden of Eden there is the tree of Life and tree of Knowledge (Genesis 2:9); Wisdom *is* a tree of Life (Proverbs 3:18 [2]). The most *cunning* of all creatures is present, the serpent (Genesis 3:1), commonly depicted wrapped around the tree of Knowledge; Wisdom created cunning (Proverbs 8:12 [2]).[10] Jesus told his disciples to be wise as serpents (Matthew 10:16 KJV).[11] Eve *reasons* about disobeying God by eating the fruit from the tree of Knowledge. Wisdom owns reason (Proverbs 8:12 [2]). Adam and Eve ultimately choose to eat the fruit of *Knowledge*. Wisdom owns Knowledge (Proverbs 8:12 [2]). Lady Wisdom is the utterly divine Woman who is stunningly present in the Garden of Eden, I aver.[12] I believe that humankind was not fully formed until receiving both the breath of Life from God (Genesis 2:7) and the

[9] Water features prominently in Zen. Zen practitioners define ki to be an energy or inner strength that can be directed from the 'one point' called the *tai-den* through visualization to places outside the body, like a valve through which water flows. Aikidoists believe the center for ki, the tai-den, is about 1.5 inches below the navel (pp. 63-64 of [20]). Also, Heraclitus believed that a person's soul is made out of water and that the soul will live in the divine spark only if the person led a certain type of virtuous life (p. 174 of [36]).

[10] A serpent is simply a head propelled by a tail, a physiology that emphasizes pure thought.

[11] "And Moses made a serpent of brass, and put it upon a pole, and it came to pass, that if a serpent had bitten any man, when he beheld the serpent of brass, he lived." (Numbers 21:9 KJV).

[12] For a monumental course correction see Mark 10:2-9.

fruit of Knowledge from Lady Wisdom; God and Wisdom, tree of Life and tree of Knowledge, Adam and Eve.

According to Raphael Patai, the Zohar holds that Adam was the offspring of God and the Shekhina. This is hinted at in the passage that states that when Adam "emerged into the world, the sun and the moon saw him and their light faded, because the apple of the heel darkened their light. Why? Because he came from the work of the Supernal Sun and Moon," i.e., from God and the Shekhina (pp. 162-163 of [33]).

Descartes will have us believe that we are because we think, "I think, therefore I am" (p. 66 of [51]). Nietzsche will have us believe that we think because we are, "I am, therefore I think" (aphorism 276 of [31], p. 8 of [23]).[13] God is not in this picture and neither is Lady Wisdom. These two aphorisms can be viewed as secularizations of Genesis 2 where Adam received the breath of Life and Genesis 3 where Adam and Eve ate the fruit of Knowledge. In regards to these aphorisms of Descartes and Nietzsche, I beg to differ.

Although it may seem that Adam and Eve were capable of reasoning *prior* to eating the forbidden fruit, it is also quite possible that the Garden of Eden was a timeless place, that this is a parable after all.

"I received the breath of Life therefore I am; I ate the fruit of Knowledge therefore I think." —Adam Young

Rather than tying "I am" to "I think" and vice-versa, denying our two Creators, the above aphorism accounts for our formation in the Garden of Eden; we are because of God and we think because of Lady Wisdom.

One might even go as far as to call *cogito, ergo sum; sum, ergo cogito* a form of idolatry. In speaking of the revolution against God, Alan Watts wrote (pp. 11-12 of [52]):

"When the throne of the Absolute is left vacant, the relative usurps it and commits the real idolatry, the real indignity against God—the absolutizing of a concept, a conventional abstraction."

In an appeal to our rational nature, it is instructive to apply Kantian universalization to Eve's reasoning, i.e., the maxim she adhered to when eating the forbidden fruit. In order to determine whether or not a given maxim is rational, Immanuel Kant has us take the maxim, apply it universally, and then ask the question, "Is it still conceivable in this world?" If it is, then it should be regarded as rational. Otherwise it can only be seen as irrational.

Consider the following characterization of the maxim that Eve adhered to when eating the forbidden fruit: eat the forbidden fruit since the tree of knowledge is pleasant to the eyes and the fruit is good for food and is desired to

[13]The latin form of both of these aphorisms is: *cogito, ergo sum; sum, ergo cogito.*

make one wise (Genesis 3), and do so despite God's explicit command not to. By applying Kantian universalization to this heretical, or child-like maxim, it follows that the tree of knowledge would be devoid of fruit; all of humankind would disobey God and eat fruit from the tree of knowledge. The crown of the tree of knowledge, Lady Wisdom, would be fruitless; the link between heaven and earth severed.

At the opposite extreme is the following sapiential maxim regarding Lady Wisdom: "If thou seekest Her as silver, and searchest for Her as for hid treasures; Then shalt thou understand the fear of the Lord, and find the knowledge of God" (Proverbs 2:4-5 KJV). By applying Kantian universalization to this holy maxim it follows that all of humankind will seek Wisdom as for hid treasures and will find the knowledge of God. The crown of the tree of knowledge, Lady Wisdom, would be in heaven; heaven and earth joined as one.

By universalizing Eve's maxim and Wisdom's maxim two diametrically opposed end states can be seen. Universalizing Eve's maxim characterizes Adam and Eve's act as knowledge hoarding. I conclude that God's Original Command was this: one should not hoard knowledge. Hoarding knowledge severs our link to the understanding One.

3.6 The Original Paradox

I believe that the story of our creation is one of symmetry. God possessed Wisdom in the beginning of his way (Proverbs 8:22 KJV). God created Adam and from the rib of Adam, God created Eve. Lady Wisdom created cunning (Proverbs 8:12 [2]) and the serpent is the most cunning of all creatures.[14] The God $\rightarrow$ Adam, Eve connection embodies Life. The Wisdom $\rightarrow$ Serpent connection embodies Knowledge (Wisdom, thought, reason, crafty, cunning, subtle). The turning point in creation was when Eve collided with the serpent.

In reference to the tree in the midst of the garden, Eve informed the serpent that God said "neither shall ye touch it" (Genesis 3:3 KJV). The ban on touching the fruit was not addressed previously in the Book of Genesis. Multiple commentators have observed this extension to the divine prohibition (p. 11 of [1]). With the view that Eve falsified the word of God, Adam and Eve were in a state of sin prior to eating the forbidden fruit.[15]

What I am calling the Original Paradox is based on the following assumptions:

1. **Adam and Eve were in sin before eating the forbidden fruit**: This

[14] "Now the serpent was more crafty than any of the wild animals the Lord God had made" (Genesis 3:1 NIV).

[15] As opposed to "sin" one might prefer the view that they possessed free will but were naive.

is evidenced by what Eve said to the serpent, the falsification of the word of God in saying "neither shall ye touch it" (Genesis 3:3 KJV).

2. **The fruit of knowledge heals the tongue**: This is supported by the following. Regarding Wisdom, "She is a tree of life to those who hold her fast and blessed are those who look for her" (Proverbs 3:18 [2]). "The tree of life is the healing of the tongue, and he that eats its fruit will be filled by it" (Proverbs 15:4 [2]).

It is possible that upon uttering "neither shall ye touch it" Eve realized that she blasphemed. This may therefore have provided motivation for Adam and Eve to heal their tongues. Under this interpretation Adam and Eve were trapped in what I call the Original Paradox:

1. If Adam and Eve obey God and do not eat of the tree of knowledge then their tongues may utter blasphemous words for centuries or perhaps all eternity.

2. If Adam and Eve disobey God and eat of the tree of knowledge then their tongues will receive healing in the short term but at the penalty of death in the long run.

It is a paradox since sinning is unavoidable. The only path that causes Adam and Eve to eventually cease sinning against God completely is to disobey the Original Command. The Original Paradox is resolved by disobeying the Original Command.

The principle of double-effect was put forth by the theologian, philosopher, and jurist Thomas Aquinas. The principle of double-effect explains the permissibility of an act that causes significant harm as a side-effect of promoting some form of good. For instance, it explains why some rationalize the acceptability of Alice hitting Bob so hard that he dies if it is the only way for Alice to save herself from Bob's assault (p. 224 of [17]). The decision of Adam and Eve to heal their tongues by eating the forbidden fruit can be viewed as an instance of the principle of double-effect: they disobey God by eating the forbidden fruit but in so doing hope to sin less against God in the long run.

Upon eating the fruit from the tree of knowledge, the God $\rightarrow$ Adam, Eve connection that embodies Life united with the Wisdom $\rightarrow$ Serpent connection embodies Knowledge; Life and Knowledge combined to make humankind a full and complete image of God. "And the Lord God said, Behold, the man is become as one of ***us***" (Genesis 3:22 KJV). Under this interpretation, Adam and Eve did what the Book of Proverbs encourages all of us to do: seek Wisdom as for silver and gold.

3.7 Aspiring to be Trees of Life

If we presume that the fruit of the tree of knowledge had seeds and we focus on the Original Command then the command implies that one should not eat seed bearing fruit from the tree of knowledge. How many seeds of knowledge did Adam and Eve eat from the tree of knowledge? Knowledge expands broadly upon itself in the same way as Life. Perhaps the seeds of the tree of knowledge are within us and our pursuit of Wisdom is the way to keep the knowledge growing as God had originally intended; that for the seeds of knowledge to grow within us, we must necessarily carry on the same *process* as the tree: *experience* birth–*experience* life–*experience* reproduction–*experience* death–*experience* birth. The realization of new knowledge is an inherently inward process, one that we can hardly describe, perhaps born of the seeds of knowledge. In this respect, preventing Adam and Eve from becoming immortal by eating of the tree of life was the only way to rectify the imbalance caused by knowledge hoarding. Only with an ever expanding and replenishing population, whereupon each child has a fresh outlook on the world, being at once limber of mind, body, and spirit, can the sacred seeds that Adam and Eve consumed grow and achieve their intended fullness. Age brings with it a certain rigidity of mind and body. With an ever expanding populace outside of Eden death may be regarded as a practical necessity. In this view, the punishment was a restoration of balance.

Adamah in Hebrew is soil, earth. As a result of the breath of Life, the human of dust becomes a human being, literally an earthling (p. 75 of [40]); from Adamah comes Adam. Ben-Adam (son of Adam) is any human being. The metaphor of man as tree is supported by both the manner of Adam's creation and the manner of his demise. Like a tree, Adam was formed of the dust of the ground and unto dust shall he return. In this epistemological view, God is salvaging the growth of the sacred seeds through the progeny of man, having them grow in man himself.

It is all too easy to regard fruit as food and overlook the fact that fruit is alive. A powerful symbolism then, in the Genesis, is that *the fruit of knowledge is alive*. This paves the way for embracing the unification of Life with Knowledge.[16] Eating the fruit prevents a new tree from growing in the ground.

"To hoard Knowledge is to take a Life; To take a Life is to hoard Knowledge."
—Adam Young

[16] "He set before them Knowledge, and allotted to them *the law of life*." (Sirach 17:11 [10])

3.8 The Root of the Problem of Knowledge Hoarding

The foregoing laid the foundation to identify the root of the problem of knowledge hoarding, as I see it. There are those who believe that the serpent in the Garden of Eden was possessed by satan. Under this belief, one may interpret the root of the problem as being satan that coaxes us to hoard fruit from the tree of knowledge, i.e., satan encourages us to hoard knowledge.

It is only after disentangling ourselves from the preoccupation that the serpent is effectively satan that an entirely different possibility can be seen. That possibility is this: male chauvinism going back millennia has contributed greatly to the problem of knowledge hoarding.

Relative to God, Lady Wisdom plays a diminutive role in many religious circles. She is reduced to a metaphor of divine Wisdom (p. 48 of [35]). She is personified as a tree of life, a major symbol of a fertility goddess (p. 50 of [35]); passages in the Bible demote Her from Lady Wisdom to a sacred tree or pole setup near an altar (2 Kings 13.6, 17.16; Deuteronomy 16.21).[17] She is fashioned as an intellectual love for men, "For a largely male audience, this erotic language is used to depict Woman Wisdom here and in chapters 8–9 as an intellectual love that attracts young men and sages to disciplined study and moral living." (p. 50 of [35]). It is said "Blessed be He" but not "Blessed be She."

A number of scholars have pointed out that many early Christians, most of whom spoke a dialect of Aramaic, viewed Jesus primarily as an embodiment of Holy Wisdom (p. 419 of [42]). This view is supported during His baptism where the synoptic Gospels characterize the "holy breath" (ruha d'qudsha) making Her home in Him. Ruha (Aramaic) is a noun in the feminine gender and ruach (the Hebrew form) is also feminine. Jim Stacey concludes that Jesus was a spokesman and representative of the Divine Feminine more than any other entity and that Christianity has missed this truth for 1,800 years (p. 420 of [42]).

> *"Sophia, El Shaddai, Shekinah, and Holy Wisdom are names of The Divine Feminine that have been hidden, lost, and purposely left out by the Roman Catholic Church and covered with its masculine mask"*
>
> —Jim Stacey (p. 421 of [42])

The devout do not question the divine nature of the breath of Life. But why has there not been equal emphasis placed on the fruit of Knowledge? The chauvinistic view of Lady Wisdom has caused the sacred nature of Knowledge to be de-emphasized. This has opened the door for society to treat Knowledge as our property instead of Her *sacred* property.

I have put forth two possibilities for the root of the problem of knowledge hoarding: that satan, commonly regarded as a *male* embodiment of pure evil, is

[17] Esoteric Theological Seminary, www.northernway.org/hgoddess.html

to blame since he coaxes us to hoard knowledge. This amounts to saying that only a powerful twisted man could possibly be the root of the problem. The second: that we have failed to hold the utterly divine Woman, Lady Wisdom, up high. From this perspective, the beginnings of knowledge hoarding has its root in the marginalization of Lady Wisdom.

The Original Command, so defined, tells us not to do something, namely, not to hoard knowledge. This is the prohibition of an act; of all things that can be done, this command prohibits one of them. An interesting counterpoint to this is the following question. Is there anything in the Scriptures that encourages us to make sure that knowledge can be freely shared? Lady Wisdom addresses exactly this positive act. "Blessed is the man who will listen to me and will keep watch upon my gates all day and guards the posts of my gates. Because my goings forth are the goings forth of Life, and so the will of Lord Jehovah goes forth." (Proverbs 8:34-35 [2]). The book of Proverbs also addresses the actions of the intelligent. "Fools inherit madness and the intelligent distribute knowledge." (Proverbs 14:18 [2]).

By elevating Lady Wisdom we raise Knowledge to its rightful place alongside Life. By treating Knowledge as sacred and by guarding the posts of Wisdom's gates, knowledge hoarding may one day become a thing of the past.

3.9 Theological Foundation of Free Coding

The conclusion from Section 3.8 is that the root of the problem of knowledge hoarding is the marginalization of Lady Wisdom and not treating knowledge as sacred. This, combined with the Original Command and the Original Paradox, forms a Judeo-Christian foundation for the problem of knowledge hoarding. Since obstructions to free coding form a subset of the problem of knowledge hoarding, it also forms a Judeo-Christian foundation of free coding. It is rooted in the Old Testament. An argument in support of free coding based on the New Testament will now be given.

The Constitution of the United States did not originally grant corporations rights. People had rights, corporations did not. This state of affairs changed over time in American history. *Corporate personhood* is a legal personification of the corporation. Chief Justice Marshall wrote in 1930 that, "The great object of an incorporation is to bestow the character and properties of individuality on a collective and changing body of men."[18] In what follows the notion of corporate personhood is leveraged along with the Golden Rule, also called the Law of Reciprocity, to show the harmful effect of knowledge hoarding relative to free coding. The Golden Rule is written in the first book of the New Testament.

[18]Providence Bank v. Billings, 29 U.S. 514 (1830).

"Therefore all things whatsoever ye would that men should do to you, do ye even so to them: for this is the law and the prophets." (Matthew 7:12 KJV)

In the below "you" should be read as "your institution."

If you would like other institutions to give their members the freedom to report vulnerabilities they find to society unobstructed so that you might be protected, so then you should give your members the freedom to report vulnerabilities they find to society unobstructed so that other institutions might be protected.

If you would like other institutions to give their members the freedom to contribute code to society unobstructed so that you might benefit from the code, so then you should give your members the freedom to contribute code to society unobstructed so that other institutions might benefit from the code.

An institution that side-steps giving their staff the freedom to give knowledge to society moves the world towards dystopia: vulnerabilities would flourish, causing rampant exploitation, and duplicitous coding efforts would abound.

3.10 The Instantaneous Embrace

Those who strongly believe in freedom of speech, the free software movement, or full disclosure of vulnerabilities may find themselves instantaneously embracing Freedom A and Freedom B; no essay, explanation, or manifesto is needed. It is an ineffable view, a causeway to free coding traveled by the kindred spirit. To this end, I have managed a parable that will resonate with some yet irk others, appearing oracular to the former while pretentious to the latter. But there is no helping it; and then there is the matter of its meter and form. I chose poetry, for I seek the assembly of both heart and mind. Free Coding:

As if etched in stone in the deepest, darkest mountain cavern,
on rough-hewn obelisk in chamber high;
for them not to seek, but find;
unseen but for our inner light,
unheard but for the whisper of our innermost voice,
to touch, sending forth the mighty hand
that grips our hearts, holds us fast, drains our eyes,
and at once know them to be good and true.

3.11 Triadic Characterization

Johann Fichte leveraged *thesis, anti-thesis, and synthesis* as a formula for the explanation of change (p. 46, note 37 of [53]). In Section 2.7 a triadic char-

acterization of the problem of knowledge hoarding was presented. The thesis is knowledge hoarding. The anti-thesis is knowledge sharing. The synthesis in Section 2.7 is a series of logical and Western philosophical measures. The below are additional measures having a religious foundation. The additional points for the synthesis are these:

1. **Heed the Original Command**: Insofar as possible heed the Original Command. There are situations in which it may seem that publishing knowledge may harm people, e.g., publishing a severe IT security vulnerability that affects millions of unpatched computer systems. In such cases responsible disclosure to the author of the software three months prior to publication of the vulnerability may be desirable (responsible disclosure). When faced with the choice to hoard knowledge vs. share it, and the righteous path is not clear, seek guidance from the understanding One.

2. **Guard the posts of Wisdom's Gates**: Challenge, resist, and diminish restrictions on giving Knowledge to society. "Blessed is the man who will listen to me and will keep watch upon my gates all day and guards the posts of my gates. Because my goings forth are the goings forth of Life, and so the will of Lord Jehovah goes forth" (Proverbs 8:34-35 [2]).

3.12 Conclusion

I conclude that the root of the problem of knowledge hoarding is the marginalization of Lady Wisdom, the reduction of Her to a metaphor, and treating Knowledge as our property instead of Her sacred property. I believe that to solve the problem of knowledge hoarding we must, as a society, heed the Original Command to refrain from hoarding knowledge and beyond this, guard the posts of Wisdom's gates so that knowledge may go forth, the goings forth of God.

It may be difficult to move past the belief that Lady Wisdom is a metaphor, to accept that She is real. For those who don't believe, I offer these parting words: To hoard knowledge is to violate His Original Command and hoard Her sacred property. Therefore, any act that hoards knowledge is in direct defiance of God and Lady Wisdom, blessed be They, and should not be taken lightly.

Chapter 4

The Experimental Free Coding License

Humankind has come to understand, owed to no small sacrifice of whistleblowers, that the U.S. Government favors the hoarding of vulnerabilities over having them remedied. This is evidenced by scandals this year that include the Vault-7 trove of secret software exploits hoarded by the CIA, as exposed by Wikileaks, and the MS Windows exploits hoarded by the NSA, as exposed by The Shadow Brokers. To diminish government surveillance run amok and defend human rights, I endeavor to exalt the industrial coder and vulnerability researcher commonly girded by pre-publication censorship. Overreaching pre-publication censorship chills contributions of source code to society and the publication of software vulnerabilities. I adopt a positive reinforcement approach to solving this problem by presenting a new software license that grants the privilege to modify, redistribute, and distribute derivatives of the covered work to institutions that: (1) affirmatively support the contribution of code to society under conditions of the author's choosing without any prior restraints, and (2) affirmatively support publishing, under conditions of the author's choosing, critiques or documentation of code that has been given to society without any prior restraints. Institutions that do not affirmatively remove obstructions to contributing code to society and that do not affirmatively remove obstructions to publishing critiques or documentation of code that has been given to society are not given these privileges. I call this a free coding license. A *highly* experimental free coding license is presented that *might* be enforceable based on copyright law. A mature free coding license has the potential to increase free/libre and open source software contributions, diminish secret vulnerability stockpiles, and amplify freedom.

Published through the viXra.org e-Print archive Nov. 2, 2017.
viXra citation number: viXra:1711.0117. This version contains updates as of Feb. 15, 2018.
This manifesto was submitted to the Cornell University arXiv e-Print service on Oct. 29, 2017 under the category "Computers and Society," was assigned identifier submit/2052823, and was rejected by Cornell University on Nov. 15, 2017 by the moderators "who determined it to be on a topic not covered by arXiv." For the latest version go to www.coderfreedom.org.

4.1 Introduction

Today software controls many aspects of human life, from the communications systems we rely on to command-and-control systems in our automobiles. The information assurance they provide is critical to the privacy and safety of individuals. Yet, we live in an era in which private industry and governments hoard software vulnerabilities for the purpose of achieving, in the parlance of the intel community, "total information superiority," the ability to hack any device, listen to any conversation, and pwn any command and control system.[1]

The contribution of free/libre and open source code to society helps reduce vulnerabilities since vulnerabilities are more readily identified in source code as opposed to in compiled code. The publication of descriptions of vulnerabilities also helps reduce the number of vulnerabilities in society's foundation of code. Yet, industrial pre-publication censorship chills such contributions and therefore has a harmful effect on privacy and safety. Industrial pre-publication censorship allows secret vulnerability stockpiles to burgeon in support of government surveillance run amok.

In what follows the power of software licensing, as rooted in copyright, is explored to mitigate the negative effects of overreaching industrial pre-publication censorship. Free coding, which consists of Freedom A and Freedom B, was introduced in Chapter 1. A philosophical foundation for free coding was given in Chapter 2 and a theological foundation for free coding was covered in Chapter 3. To review, Freedom A is: You have the freedom to write code and give it to society under conditions of your choosing. Freedom B is: You have the freedom to write and publish, under conditions of your choosing, a critique or documentation of code that has been given to society. The free coding license is intended to uphold free coding by, in the case that the licensee is an institution, conditioning the privilege of modifying, redistributing, and distributing derivatives of the covered work on instituting organizational policy that upholds free coding without any prior restraints. Text within the license effectively reiterates Freedom A and Freedom B. These are copyright distribution terms that inject liberty-preserving policy into institutions.

The below example from Chapter 1 illustrates Freedom A and Freedom B in conjunction with Freedoms 0, 1, 2, and 3 as defined by the Free Software Foundation. This shows how free coding and free software go hand-in-hand.

Alice writes a free encryption program and gives the source code of it to society (Freedom A). She is a cryptographer that is protecting people from harm. Bob who is a free software freelancer redistributes copies of it as a service to others for a fee (Freedom 2). Carol the vulnerability researcher receives a copy

[1] pwn is not misspelled.

of the program from Bob and analyzes the source code (Freedom 1), finds a vulnerability in it, and publishes a description of the vulnerability (Freedom B). Dave the coder studies the vulnerability description, fixes the code (Freedom 1), and gives the resulting encryption program to society (Freedom A and Freedom 3). Eddie the journalist receives the fixed encryption program and uses it to encrypt a controversial article that he is writing (Freedom 0).

There is a battle for freedom to use public knowledge (p. 189 of [54]). The Free Software Foundation (FSF) has made great strides in this battle, progress made possible in part due to a clever use of copyright law known as copyleft. Copyleft is embodied in the free software license known as the GNU General Public License (GPL). Copyleft is also embodied in the GNU Free Documentation License (FDL) that covers documentation.

A free software license aims to give people the freedom to use public knowledge. The GNU GPL has the following condition: when distributing the program or a derivative of it as object code or an executable you make the corresponding source code available to society and you release all of your derivatives of the program under the GPL.

A free coding license aims to give people the freedom to contribute to public knowledge. A free coding license has the following condition: when distributing the program or a derivative of it you affirmatively grant Freedoms A and B without any prior restraints to all of the members of your organization and you release all of your derivatives of the program under the free coding license.

Free coding and free software are different concepts that should not be confused with one another. Free coding is about the freedom to give knowledge to society, and in particular Freedom A and Freedom B. Free software is about the freedom to use knowledge that has been given to society, and in particular Freedoms 0, 1, 2 and 3. Similarly, a free coding license should not be confused with a free software license. A free coding license upholds the notion of free coding. A free software license upholds the notion of free software.

I state up-front that the experimental license that I present herein is not ready to be used operationally in any capacity. It is in need of significant legal expertise and perhaps further research. It is best considered as a highly experimental wedge, rooted in copyright, for opening the door to free coding. I call it a wedge because it is very strongly worded. One might wonder why it is so. I believe that restrictions on free coding, and more generally restrictions on giving knowledge to society, will get orders of magnitude worse for posterity. So, the experimental license should not be criticized under the presumption that I think it will be used within my lifetime. It should be regarded as a possible technique to be leveraged in the event that restrictions on free coding reach a pitch that is intolerable for the soul in bondage. The time to research such solutions is now, not when the enormity of the restrictions are upon us and we reach the

brain-washed state of unquestioning.

In Chapter 1 the utilitarian aspects of free coding were presented and a software license term that upholds Freedom B fashioned around vulnerability publication was given. It was included in Chapter 1 since it is a novel pragmatic approach to reducing vulnerabilities, one that companies might agree to (with the obvious exception of cyber arms manufacturers). The present chapter can be viewed as a radical expansion of such a license term.

In Chapter 2 the triadic notion of *thesis, anti-thesis, and synthesis* from Johann Fichte was reviewed. This is a formula for the explanation of change (p. 46, note 37 of [53]). The problem of knowledge hoarding, in relation to restrictions on free coding, was then characterized using this triadic approach. The synthesis embodies an approach for upholding free coding in society. This chapter forms an addition to the synthesis presented therein.

The first two chapters thereby present rational measures for upholding free coding. They are designed to be fair to both the company and the individual alike. Only time will tell if these common sense measures will be upheld by industry. If they are ignored, then a more compelling approach may be needed. By applying free coding distribution terms to knowledge that is given to society, such stubbornness could potentially be overcome. It is due to this very possible turn of events, namely, industry disregarding the fair compromise presented in Chapters 1 and 2 that this copyright-based wedge for upholding free coding is being presented.

Put another way, I am well aware that at this point in time no company is likely to agree to the terms of the license presented herein. But after awareness of free coding is achieved and coders and vulnerability researchers feel in their hearts that they deserve *affirmative* support for free coding, they may decide to use a license that contains free coding distribution terms frequently. A mature free coding license gives coders a new choice, the ability to condition the use of the knowledge that they are giving to society on affirmatively upholding the freedom to give knowledge to society.

A summary of this chapter is as follows:

1. The first software license that conditions the modification and distribution of the covered work on enacting institutional policy P is presented.

2. As an application, a policy P that affirmatively authorizes all members of the institution to exercise Freedom A and Freedom B without any prior restraints is given. The license with this embedded policy is the free coding license.

3. Since institutional policy does not carry with it the force of law, techniques are given that make P self-enforcing. In particular these techniques are:

(1) a policy quine that authorizes all members of the institution to publish P and the fact that the institution is bound to P, and (2) a policy statement that authorizes all members of the institution to publish all accounts of violations of P to hold the institution publicly accountable for adhering to policy P.

4. The potential of a free coding license to amplify freedom is shown.

4.2 Background

Free software is embodied by Freedoms 0, 1, 2, and 3 [44].[2] A program is free software if the program's users have the four essential freedoms:

Freedom 0: The freedom to run the program as you wish, for any purpose.

Freedom 1: The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.

Freedom 2: The freedom to redistribute copies so you can help your neighbor.

Freedom 3: The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

Perils in limiting the freedom to run programs were outlined by the FSF. The FSF asserts that Freedom 0 must be complete (p. 181 of [44]).

The principle of voluntary co-operation and the principle of decentralization are foundational principles of the GNU GPL. The principle of voluntary co-operation was described as follows[3] (p. 169 of [43]):

"Remember, never force anyone to co-operate with any other person, but make sure that everybody's allowed to co-operate, everyone has the freedom to do so, if he or she wishes."

The principle of voluntary co-operation and the principle of decentralization evolved from Stallman's work on Emacs and the corresponding social contract that existed within the Emacs Commune. The social contract called for all changes to be published and sent to Stallman. He later felt that it was wrong to require people to publish all changes and require them to be sent to one privileged developer (p. 126 of [54]).

Bradley M. Kuhn suggested the use of a quine to fill a gap in the GNU GPLv2 wherein a derivative work could be run as a network server but with the

[2] www.gnu.org/philosophy/free-sw.en.html

[3] From a speech given by Richard Stallman at New York University on May 29, 2001.

source code withheld. A quine in computing is a non-empty program that does nothing but output its own source code when run. A server that is made to support a network API call that functions as a quine, returning the source code of the server, solves this problem. This led to the creation of the Affero General Public License.[4]

The name quine is in honor of the philosopher Willard Van Orman Quine. The following is Quine's paradox [37]:

"Yields a falsehood when appended to its own quotation" yields a falsehood when appended to its own quotation.

The concept of a quine has also been used in security to detect the presence of malware [17].

4.3 Censorship and Vulnerability Hoarding

An "intellectual property" agreement executed between a company and it's employee may restrict the employee from distributing source code and descriptions of vulnerabilities when the work can be considered within the scope of employment. Employment contracts exist that assert copyright ownership over all written works produced by the employee during and outside of work hours. When a scope claim can be justified, the copyright is held by the company on the basis of that the work constitutes a work-for-hire even when the employee is the sole author of the work.

Such agreements and employment contracts that claim ownership in this fashion are often enforced via institutional policies and procedures. Institutional policies exist that mandate that all written works that an employee plans on publishing be subjected to pre-publication review and approval. A corresponding procedure will define the manner in which works are to be submitted and will define the department that carries out the pre-publication censorship. Therefore, from a functional perspective, institutional agreements, contracts, policies, and procedures are a form of censorship that can negatively impact free coding.

Vulnerabilities are treated very much like trade secrets today. They are harvested by public and private entities and are sold in marketplaces shrouded in secrecy. Private sale of vulnerabilities to the Federal government under NDA can hamper the government's ability to disclose them to the author of the affected software for remediation [39].

One avenue that vulnerability researchers have at their disposal to profit from vulnerabilities they find is to sell them to the affected software company. This effectively occurs in bug bounty programs [30]. The reporting and sale of

[4] en.wikipedia.org/wiki/Affero_General_Public_License

vulnerabilities by vulnerability researchers to commercial entities that authored the affected software has traditionally been fraught with challenges on both sides.

The National Telecommunications and Information Administration's (NTIA) Awareness and Adoption working group conducted a survey on vulnerability researchers and vendors to assess vulnerability disclosure and handling procedures [28]. The study found that vulnerability researchers prefer open and reliable lines of communication with affected vendors over being paid. Poor responsiveness on the part of affected vendors to vulnerability reports and failure to meet remediation deadlines imposed by researchers has prompted researchers to remedy the situation by publicly disclosing their findings.

A decade ago researchers grew tired of turning bugs over to Microsoft for free and had ongoing concerns of being sued [29]. This was part of the No More Free Bugs movement. The Wassenaar Arrangement is a multilateral export control regime for conventional arms and dual-use goods and technologies. It includes export controls for technology related to "intrusion software" that can defeat "protective measures". There is a policy of presumptive denial for items that have zero-day exploit capabilities. Legal concerns grew among bug-hunters over draft changes to the Wassenaar Arrangement last year [28]. The changes not only omitted key exceptions, but further appear to prohibit sharing of vulnerability research without a license [5].

Vendors are at times skeptical of the technical claims made by vulnerability researchers that respond to bug bounty programs. A software company typically wants verification of the claims being made by the vulnerability researcher [21] and this may be hard to achieve in a direct researcher-to-vendor transaction where trust may run thin.

As a result of the aforementioned issues a whole new industry of intermediaries has sprouted up in the last couple years to "fairly" settle vulnerability trades. Example intermediaries include Synack and HackerOne[5] [29]. Such an intermediary is a trusted arbiter between vulnerability researchers and software companies. They provide vetted vulnerability reports to affected software companies, handle payments, and establish a reputation system for vulnerability researchers. This provides assurance to buyers that what is on the label is what is in the can.

The existence of a system for "fair trade" of vulnerabilities conditions us to believe that software vulnerabilities are property that can be legitimately be transferred. Those who purchase vulnerabilities can secretly hoard them.

Notable examples of vulnerability hoarding include the following. Wikileaks published a trove of vulnerabilities hoarded by the Central Intelligence Agency.

[5] en.wikipedia.org/wiki/HackerOne

This was from the Wikileaks cache of documents dubbed Vault-7. This included a vulnerability in Cisco devices exploited via Telnet that grants an unauthenticated remote adversary the ability to execute code with elevated privileges on the targeted device [41]. A group that called itself The Shadow Brokers posted data online in 2016 indicating that it penetrated the NSA's Equation Group. The trove contained vulnerabilities in Cisco and Fortinet products. Cisco confirmed that two vulnerabilities in the disclosure can be used to breach its firewalls [9]. Later in the same year the same group published a collection of MS Windows exploits [22].

4.4 The Free Coding License

The concept of a policy quine was introduced in subsection 1.4.1 of this manifesto. Here it is used to achieve policy transparency: the policy quine authorizes all members of the organization to publish the policy itself along with a statement that it is effect in the organization, naming the organization explicitly. The perpetuation language and text thereafter is adapted from the GNU GPLv2. The experimental free coding license is as follows:

Free Coding License: "Organization" is defined by taking Your chief legal officer (CLO), going successively up his or her management chains until the roots of the management hierarchies are reached, and then including every person that reports directly and indirectly into said roots. So, for example, a CLO in a company that is structured as a separate legal entity but that has a management chain extending up into a larger company necessarily includes the larger company as part of the Organization. Example organizations include but are not limited to: agencies, bureaus, branches of government, branches of the military, corporations, companies, firms, non-profit organizations, not-for-profit organizations, educational institutions, civil societies, charities, partnerships, co-operatives, associations, clubs, trade organizations, and a business. The following is a policy:

*"**Scope:** This Policy covers all members of our organization including but not limited to all: full-time employees, part-time employees, contractors, consultants, interns, temporary staff, resident visitors, visiting regulators, resident regulators, visiting auditors, resident auditors, visiting inspectors, resident inspectors, and board members ("Members"). **Purpose 1**: To authorize all members of our organization to publish all critiques and documentation of all code that has been given to society without delay, without approval, without restraint, without prior restraint, and without following involuntary procedures and to authorize all of*

our members to hold our organization publicly accountable for violations of this policy. These policy statements affirmatively uphold the human right of our members to publish critiques and documentation of all code that has been given to society, to include but not limited to vulnerabilities, thereby helping society. Purpose 2: To authorize all members of our organization to create new hardware and software code projects and contribute hardware and software code to society without delay, without approval, without restraint, without prior restraint, and without following involuntary procedures and to authorize all of our members to hold our organization publicly accountable for violations of this policy. These policy statements affirmatively uphold the human right of our members to contribute code to society thereby helping society. ***Policy Statements:*** *Every member of our organization is authorized to publish to the world without delay, without approval, without restraint, without prior restraint, and without following involuntary procedures a critique or documentation of any code that has been given to society wherein said member is the copyright holder of said critique or documentation ("AuthorizationT1"). Every member of our organization is authorized to present in any conference or forum without delay, without approval, without restraint, without prior restraint, and without following involuntary procedures a critique or documentation of any kind of all code that has been given to society wherein said member is the copyright holder of said critique ("AuthorizationT2"). Every member of our organization is authorized to create new software and hardware code projects without delay, without approval, without restraint, without prior restraint, and without following involuntary procedures wherein the author is the sole copyright holder of the code for said new software or hardware code project ("AuthorizationT3"). Every member of our organization is authorized to write and publish any code and give it to society without delay, without approval, without restraint, without prior restraint, and without following involuntary procedures wherein said member is the copyright holder of said code ("AuthorizationT4"). Every member of our organization is authorized to write and publish to the world without delay, without approval, without restraint, without prior restraint, and without following involuntary procedures an account of all violations of this Policy wherein said account may include but is not limited to all: person(s) involved, written and oral statements made, times and dates, descriptions of events and the violation(s) and all relevant documentation without exception ("AuthorizationT5"). An attestation is this Policy along with a statement that this Policy is permanently in effect in our organization, naming our organization explicitly. Said attestation may optionally include the name of the person making the attestation, his or her job title, and a date in which this Policy is in effect ("Attestation"). Every member of our organization is authorized to publish to the world without delay, without approval, without restraint, without prior restraint, and without following*

involuntary procedures said Attestation wherein said Attestation is entered into the Public Domain ("AuthorizationT6"). Authorization is comprised of AuthorizationT1 and AuthorizationT2 and AuthorizationT3 and AuthorizationT4 and AuthorizationT5 and AuthorizationT6 ("Authorization"). This policy is permanent. If any instrument to include but not limited to a policy, contract, or agreement, issued or executed by our organization diminishes, inhibits, or conflicts with this policy in any way or contradicts this policy in any way then said instrument is void. The full text of this policy shall be explicitly given to all members of our organization during our regularly planned policy training ("Training"). Under no circumstances is a member of our organization permitted to discourage another member of our organization from exercising the liberties and allowances afforded by this Policy. Doing so is a violation of this Policy. Under no circumstances is a member of our organization permitted to retaliate against another member of our organization for exercising a liberty or allowance afforded by this Policy. Doing so is a violation of this Policy. The authoritative policy of an organization is the most authoritative policy that is in place. For commercial companies the authoritative policy is the Human Resources policy ("Authoritative Policy"). This policy shall be included within the Authoritative Policy of our organization ("Incorporation"). This policy shall be made freely available to all members of our organization at all times ("Availability"). If our organization is ever purchased or acquired then as a condition of said purchase or acquisition the purchasing or acquiring entities must fully adopt this Policy as an Authoritative Policy ("PerpetuationA"). If our organization is ever merged with another organization then as a condition of said merger the merged whole must fully adopt this Policy as an Authoritative Policy ("PerpetuationM"). If our organization ever spins-off another organization then as a condition of separation said spin-off organization must fully adopt this Policy as an Authoritative Policy ("PerpetuationS"). ***Effective Date:*** *This Policy is effective immediately.* ***Responsibilities:*** *Human Resources, or an equivalent department if there is no Human Resources department, is responsible for executing the Incorporation, Availability, and Training policy statements. The Chief Legal Officer is responsible for executing the PerpetuationA, PerpetuationM, and PerpetuationS policy statements."* ("Policy").

If You are an Organization and You do something that copyright prohibits, e.g. modify the covered work, redistribute copies of the covered work, or distribute derivatives of the covered work then You are only permitted to do so if You have said Policy permanently in effect verbatim as an Authoritative Policy. You do not have to adopt said Policy, but if You are an organization and You do not have said Policy permanently in effect verbatim as an Authoritative Policy, then You do not have a license to modify, redistribute, or distribute a derivative

of the covered work and a distribution without a license will violate my [name of person distributing the software under this license] copyright.

Perpetuation: You may copy and distribute verbatim copies of the covered work as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the covered work a copy of this License along with the covered work. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the covered work or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License. These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the covered work, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the covered work, the distribution of the whole must be on the terms of this License, whose permissions and restrictions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

If conditions are imposed on You (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse You from the conditions of this License.

If You cannot grant said Authorization to every member of Your organization so as to satisfy simultaneously Your obligations under this License and any other pertinent obligations, then as a consequence You may not modify the covered work, nor redistribute the covered work, nor distribute a derivative of the covered work.

An organization that lacks the requisite policy can institute it, become a licensee, and then modify, redistribute, and distribute derivatives of the covered work. To my knowledge this is the first software license that contains a stand-alone institutional policy.

The license establishes the following: a condition of the privilege to modify, redistribute, and distribute derivatives of the covered work is that the licensee distribute all derivatives under the same license and that the licensee enact policy that: (1) upholds Freedom A and Freedom B, and (2) authorizes all members of the organization to publish the policy and the fact that the organization is bound to it.

The GNU GPLv2 establishes the following: a condition of the privilege to modify, redistribute, and distribute derivatives of the covered work is that the licensee distribute all derivatives under the same license and that the licensee give a copy of the derived source code to all that request it if the corresponding object code or executable is also distributed (p. 125 of [54]).

Voluntary Co-operation: The free coding license adheres to the principle of voluntary co-operation. The public conveyance of information in accordance with Authorizations T1-T6 is voluntary. The path is clear to do so.

Decentralization: The free coding license adheres to the principle of decentralization. There is no requirement that licensees report anything to the licensor or anyone else.

It may be possible to construct a version of this license in which AuthorizationT1 through AuthorizationT6 and PerpetuationA, PerpetuationM, and PerpetuationS are *distribution terms* instead of being in the embedded policy. This could conceivably strengthen the license by making these terms tied directly to the power of copyright.

4.5 Results: Security Contributions

Software vulnerabilities pose a serious risk to the privacy and safety of people. The following subsections shed light on the extent to which a free coding license may help reduce the size of vulnerability stockpiles.

4.5.1 Reducing Vulnerability Hoarding

What is not achieved: All organizations can be partitioned into 2 disjoint sets as follows. Place all organizations that identify and stockpile vulnerabilities into a set labeled "hoarders." Place the remaining organizations into a set labeled "non-hoarders." The license does not legally discourage hoarders from obtaining a copy of the source code and analyzing it for vulnerabilities. It is not clear how to leverage copyright to achieve this. Were this possible it would be a block-and-tackle approach.

What is desired to be achieved: All organizations can be partitioned into 2 disjoint sets as follows. Place all organizations that need to modify, redistribute, or distribute derivatives of the covered work and that have policies and practices that restrict free coding into a set labeled "dependent obstructors." Place all remaining organizations into a set labeled "other." Assuming the experimental license is legally sound, organizations in the dependent obstructors set must

affirmatively remove their obstructions to free coding or they will not be granted the privilege to modify, redistribute, or distribute derivatives of the covered work.

Let a secret stockpile denote a stockpile that contains secret vulnerabilities. When a vulnerability becomes public it is summarily removed from the secret stockpile. Secret stockpiles are reduced as follows. Suppose that as a direct result of removing obstructions to free coding a vulnerability researcher discovers a vulnerability in free or open source code and publishes it. There are two cases:

Case 1: The vulnerability exists in one or more secret stockpiles. The vulnerability is published so those secret stockpiles shrink.

Case 2: The vulnerability does not exist in any secret stockpiles. There is one less vulnerability in the world to add to secret stockpiles.

The experimental free coding license does not prevent vulnerability hoarders from analyzing the covered work for vulnerabilities. Rather, it diminishes obstructions that silence vulnerability researchers, empowering them to lessen secret vulnerability stockpiles from afar via full disclosure.

4.5.2 Increasing the Amount of Free/Libre and Open Source Code

Authors of proprietary code often distribute their programs in compiled form, withhold the corresponding source code, and deny Freedom 1 to users. Such a proprietary program is hidden from developers and vulnerability researchers that do not know how to reverse-engineer executable code. It is significantly more difficult to analyze compiled code for vulnerabilities than it is to analyze the corresponding source code for vulnerabilities. It follows that the problem of software hoarding exacerbates the problem of vulnerability hoarding.

It has been argued that withholding source code has a security benefit. The argument is that since fewer people can access the program logic the program is more secure operationally. This fallacy is well-known among security professionals. So much so that it has a name: "security by obscurity." It presumes that no determined reverse-engineer will dissect the machine code and find the vulnerabilities therein. Also, time and time again compiled proprietary code conceals design-level vulnerabilities that would be more evident were the source code public.

A free or open source program does not obstruct vulnerability analysis but a proprietary program with the source code withheld does obstruct vulnerability analysis. This is concretely illustrated as follows. Let P_1 be the public source code of a free or open source program in language L_1. Let P_2 be the secret source code of a comparable proprietary program in language L_2. Suppose that

compiled programs in L_2 are hard to decompile. This is commonly the case, e.g., in programming languages such as ANSI C and C++. Note that,

1. Everyone can analyze P_1. People with access to P_2 can analyze P_2 but everyone else is obstructed from analyzing P_2.

2. Let T be a publicly available vulnerability analysis tool that works only on source code. Everyone can apply T to P_1. People with access to P_2 can apply T to P_2 but everyone else is obstructed from applying T to P_2.

This observation shows that making source code publicly available directly facilitates vulnerability discovery. The free coding license encourages the adoption of institutional policy that affirmatively supports contributing free/libre and open source code to society without prior restraint. This may increase the amount, quality, and use of free/libre and open source code. This may therefore diminish the chances that vulnerabilities will go unnoticed.

4.6 Policy Analysis

One can imagine a variety attempts to side-step the conditions of the free coding license. The following are two conceivable maneuvers along with arguments as to why they should fail:

1. Policy P1: You may contribute code to any free or open source project provided that you give the company at least 2 weeks notice of the contribution prior to making the contribution.

2. Policy P2: You may contribute code to any free or open source project provided that you give the company notice of doing so on or before the day of the contribution.

Policies P1 and P2 are reminiscent of the TSU Notifications that the U.S. government required of all Americans who contributed strong crypto code to society. They are mandatory reporting requirements and there is no approval involved.

Policy P1 is a violation since it imposes a delay on the individual. The text of the license explicitly prohibits delays. A delay in releasing the fix of a zero-day vulnerability may place the privacy and safety of people at risk.

Policy P2 is a violation since it foists an involuntary procedure upon the individual. The involuntary procedure is the compulsory reporting of the code contribution. The text of the license explicitly prohibits forcing individuals to follow an involuntary procedure.

4.7 Spreading Freedom

The concept of a free coding license can be extended to cover other types of knowledge as well. For example, it can be extended to cover documentation. This is analogous to the GNU GPL and the GNU Free Documentation License. The provisions of the free coding license, after a fashion, may bolster the freedom to give to public knowledge in all manner of expression that falls under copyright.

4.7.1 Policy Spider Plant

Some may recall the days when computers were large, very expensive, and rare, when software was garnish given away by most hardware companies to make computers more appetizing (p. 99 of [54]). Few institutions had policy that forbade sharing of code. But as operating systems and applications showed promise of being profitable, institutional policy soon followed to restrict the flow of code and treat it as valuable property as opposed to speech.

The idea that software is property that should be controlled spread from one institution to another. Conceptually, the institutional policy that restricted the flow of code can be thought of as having been copied from one place to another by people. Like a living thing it therefore "spread" among institutions, restricting the freedom of those it covered.

The GNU GPL provides a remarkable defense of sharing. The GNU GPL has the following condition: when the program is combined with non-free code, the resulting whole must fall under the GNU GPL. Were this requirement not present there would be a gaping whole in the GPL; such combinations would allow proprietary software to overwhelm free software. As a result of this novel defense, some referred to the GPL as being virus-like. Stallman likens it to a spider plant that when actively cut and relocated starts to spread (p. 22 of [54]).

The GNU General Public License therefore has the effect of spreading like a spider plant. The free coding license exhibits this effect as well due to language adopted from the GPLv2.[6] But, the free coding license terms also have a meta spreading effect: the institutional policy that they install also spreads like a spider plant. In an acquisition the embedded policy requires that the acquiring company adopt free coding policies. In a merger the embedded policy requires that the merged whole adopt free coding policies. In a spin-off the embedded policy requires that the spin-off adopt free coding policies.

[6] But this is an area that can likely be expanded and improved in the experimental free coding license.

4.7.2 Eastern Philosophical Inspiration

The experimental free coding license was inspired by Eastern philosophy and in particular Zen martial arts. Observe that it uses policy to counter policy. In particular, it uses public freedom preserving policy to counter the "spread" of secret freedom restricting policy.

"You and your opponent are one. There is a coexisting relationship between you. You coexist with your opponent and become his complement, absorbing his attack and using his force to overcome him." — Bruce Lee (p. 59 of [20]).

4.7.3 Freedom Amplification

Consider the case in which the free coding license terms are merged with a free software license. It *may* be possible to do so in such a way that Freedoms 0-3 are preserved completely for all *individuals*. However, the free coding license terms limit Freedoms 1, 2, and 3 in a specific way with respect to institutions. The privileges of modifying, redistributing, and distributing derivatives of the program are not granted to institutions devoid of institutional policy that affirmatively upholds free coding.

It is not clear how to enforce restrictions on Freedom 0 using copyright law. It has been argued that were such restrictions possible it would not be a good thing. Provisions such as "no military use" might appear, as well as programs banned for use in meat processing, programs limited to Kosher food, etc. [44]. One might perceive this ability to impose restrictions as being a Pandora's Box. But, given that it is not clear how to enforce such a restriction on Freedom 0 it seemed to be a non-issue.

Curtailing Freedoms 1-3 is an important issue with significant implications. This chapter presents a software licensing hack, rooted in copyright, that curtails Freedoms 1, 2, and 3 in return for arbitrary institutional policy. Assuming the license is sound, this has opened Pandora's Box.[7] This extends the "dual-use" nature of copyright law. It follows that, like it or not, software licenses can now appear, supported by copyright law, that impose arbitrary policies that cater to special interests.

Freedoms 1, 2, and 3 are being curtailed for the sake of affirmatively removing obstructions to giving knowledge to society (knowledge as it relates to free coding). This is a manifestation of positive feedback. So, the free coding license establishes a trade-off between Freedoms 1-3 and positive feedback in the form of giving knowledge to society.

[7] Hesiod, an ancient Greek poet, wrote *Works & Days* that includes the myth of Prometheus and Pandora. Pandora's box was actually a large jar. When Pandora opened the jar all manner of evil spirits entered into the world. A healing spirit named Hope remained inside.

The following analogy may help convey the potential of the license. Guitarists are familiar with the phenomenon known as acoustic feedback, also called the Larsen effect. Acoustic feedback is a positive feedback process. It occurs when a sound loop is formed between an audio output and it's corresponding input. With electric guitars, the feedback is the sound that travels from guitar's loudspeaker into the guitar's pickup. The signal received by the pickup is amplified and sent to the loudspeaker. The sound from the loudspeaker travels to the pickup and is amplified again, and so on.[8] The sound can become so loud as to become deafening.

There is currently a veritable legion of coders whose voices are quelled. By removing the prior restraints on their freedom to publish code, documentation of code, and critiques of code, they would be free to amplify one another to ever greater heights. In principle, like the Larsen effect, the free coding license feedback loop could cause free/libre and open source software contributions to crescendo.

4.7.4 Policy-Man-Computer Symbiosis

The free coding license is code that contains as a payload institutional policy that preserves certain speech freedoms of the institution's members. The free coding license, with the help of the human that executes it, installs the policy within the human's institution. The policy itself is code. It is a quine that, with the help of the human that executes it, forks off a child that is published to the world. The child quine tells the world of great things: that the people in the institution have regained their coding liberties, that the quine's parent is alive and well within the institution's governance structure. The free coding license is therefore mobile code containing mobile free speech preserving policy code that counters the spread of mobile free speech restricting policy code.

Man-computer symbiosis is the observation that man relies on computers and computers rely on man, that they will become increasingly dependent upon one another [24]. Put another way, man controls computers and computers control man. Alice writes a program and a computer executes it. The program may control Bob as a result.

Policy-man symbiosis can analogously be defined: policy controls man and man controls policy. Alice writes a policy and a human executes it. The policy may control Bob as a result.

The approach taken herein to give Alice freedom of speech utilizes mobile policy in the form of a software license and institutional policy. When such policies supplant freedom restricting policies liberties are preserved. An institutional policy that affirmatively gives Alice the freedom to write code and give it

[8] en.wikipedia.org/wiki/Audio_feedback

to society affects what computers execute. This is a cross-over effect from policy-man symbiosis to man-computer symbiosis. Due to the interaction between the two systems, the larger system may be aptly referred to as policy-man-computer symbiosis.

4.8 Conclusion

A highly experimental software license was presented that, in the case of a licensee that is an institution, conditions the privilege of modifying, redistributing, and distributing derivatives of the covered work on affirmatively upholding free coding in institutional policy. To enforce compliance, the policy allows the institution's members to publish the policy itself, publish that the company is bound to it, and publish accounts of policy violations to hold the institution publicly accountable for it's commitment to free coding. The policy installed by the experimental free coding license is therefore self-enforcing. It is my hope that the security, software, and legal communities will apply constructive criticism to the experimental license and that a mature solution will result. I am *not* claiming to have achieved a working license. My goal is to start the conversation. Should such a mature license ever be produced, copyright holders will have the choice to adapt their works to use it, a switch that once thrown will help defend the freedom of individuals to give knowledge to society.

Chapter 5

Acknowledgments

I wholeheartedly thank Moti Yung for extensive input and reviews of this manifesto covering everything from content to structure. Moti provided helpful information security, utilitarian, and Hebrew perspectives. At times I felt myself utterly *consumed* by this manifesto and Moti was ever the stable voice of reason, keeping me tethered to this world.

I thank Bruce Schneier for input, reviews, moral support, and leveraging his connections to provide further reviews and constructive criticism. Bruce assisted in the earliest stage of this effort when it was aimed at creating an "anti-stockpile license." This was the precursor to the free coding license in Chapter 4. Bruce has been an ardent supporter of this work.

I am grateful to Eben Moglen who convinced me of the critical importance of conveying to business entities, in a clear and concise fashion, the business benefits of supporting free coding. This was in a spirited discussion I had with Eben regarding an earlier draft. Eben also made me realize the critical importance of characterizing the free coding license as being *highly experimental* in its current form.

I extend my gratitude to Jonathan Zittrain who helped identify weaknesses in my earliest attempts at producing an anti-stockpile license. Jonathan taught me valuable lessons in copyright and software licensing and was a supporter of anti-stockpile licensing since the earliest stage of this work.

I thank Jon Leonard for technical feedback and engaging discussions on everything from Christianity to free software. It was Jon Leonard who originally taught me about the label *corporate personhood*, a doctrine that I knew well by nature but not by name. Jon also helped me cut down on the rhetoric (indeed there used to be more!).

I also thank Michael Makarius for reviewing and providing feedback on earlier versions of Chapters 1 and 4. Michael provided a healthy dose of skepticism that reinforced the need to segregate the utilitarian measures presented in Chapter

Published through the viXra.org e-Print archive Nov. 2, 2017.
viXra citation number: viXra:1711.0117. This version contains updates as of Feb. 15, 2018.
This manifesto was submitted to the Cornell University arXiv e-Print service on Oct. 29, 2017 under the category "Computers and Society," was assigned identifier submit/2052823, and was rejected by Cornell University on Nov. 15, 2017 by the moderators "who determined it to be on a topic not covered by arXiv." For the latest version go to www.coderfreedom.org.

1 away from the highly experimental free coding license in Chapter 4.

Adam Greene provided helpful feedback on Chapter 1 that led to a revision. Being a pentester, his input was invaluable. Adam also opened my eyes to certain aspects of the New Testament.

A hearty thanks goes to Nitin Mitra for his review of Chapter 1. In particular, Nitin provided valuable feedback and perspectives on the free coding license term presented in subsection 1.4.3.

Finally, I thank God and Lady Wisdom. Though She originally kept Her distance from me in my dreams while I wrote this manifesto, finally, one night, She stood directly before me in a dream, wearing a flowing white dress laced with silver and gold, with Her arms forward and out slightly, palms upward, and head so high I couldn't see it. Her face was trained upon heaven above. She allowed me to reach up on my tippy-toes and place my son's silver necklace around Her neck.

Bibliography

[1] R. Alter. *Genesis*. W. W. Norton & Company Ltd., 1996.

[2] The Original Aramaic New Testament in Plain English with Psalms & Proverbs. Translated by Glenn David Bauscher, 8th Edition. LuLu Publishing, 2013.

[3] H. W. Beecher. *Life Thoughts: Gathered from the Extemporaneous Discourses of Henry Ward Beecher*. Phillips, Sampson and Company, 1858.

[4] S. Biko. *I Write What I Like*. Heinemann, 1978.

[5] N. Cardozo and E. Galperin. What is the U.S. doing about Wassenaar, and Why do we need to Fight it? www.eff.org/deeplinks/2015/05/we-must-fight-proposed-us-wassenaar-implementation. May 28, 2015.

[6] S. G. Carmichael. Microsoft's CEO on Rediscovering the Company's Soul. HBR IdeaCast, Harvard Business Review, September 28, 2017 from hbr.org/ideacast/2017/09/microsofts-ceo-on-rediscovering-the-companys-soul, 2017.

[7] J. Chrysostom. *Homilies on Genesis*. The Catholic University of America Press, 2001.

[8] Coca-Cola. Human Rights Policy. human-rights-policy-pdf-english.pdf downloaded from www.coca-colacompany.com/our-company/human-rights-policy, 2014.

[9] K. Conger. Cisco and Fortinet say vulnerabilities disclosed in 'NSA hack' are legit. *TechCrunch*, Aug 2016.

[10] J. Corley. *Sirach*. Liturgical Press, 2013.

[11] S. R. Driver. *The Book of Genesis*. Methuen & Co. Ltd., 1911.

[12] M. Eckert. Cinematic Spelunking Inside Plato's Cave. *Glimpse Journal*, 9, 2012.

Published through the viXra.org e-Print archive Nov. 2, 2017.
viXra citation number: viXra:1711.0117. This version contains updates as of Feb. 15, 2018.
This manifesto was submitted to the Cornell University arXiv e-Print service on Oct. 29, 2017 under the category "Computers and Society," was assigned identifier submit/2052823, and was rejected by Cornell University on Nov. 15, 2017 by the moderators "who determined it to be on a topic not covered by arXiv." For the latest version go to www.coderfreedom.org.

[13] J. Fieser. CENSORSHIP. From the book in progress "Moral Issues that Divide Us", 2008.

[14] M. Fox. *Original Blessing.* Bear & Company, 1983.

[15] M. Fox. *Christian Mystics.* New World Library, 2011.

[16] R. T. D. George. A History of Business Ethics. *Markkula Center for Applied Ethics*, Nov 2015.

[17] V. Gratzer and D. Naccache. Alien vs. Quine. *IEEE Security & Privacy*, 5(2):26–31, 2007.

[18] M. Greenberg, J. C. Greenfield, and N. M. Sarna. *The Book of Job—A New Translation According to the Traditional Hebrew Text.* The Jewish Publication Society of America, 1980.

[19] B. Hozeski. *Hildegard von Bingen's Mystical Visions.* Bear & Company, 1986. Foreword by Matthew Fox.

[20] J. Hyams. *Zen in the Martial Arts.* G. P. Putnam's Sons, 1979.

[21] B. Krebs. Got $90,000? A Windows 0-Day Could Be Yours. krebsonsecurity.com/2016/05/got-90000-a-windows-0-day-could-be-yours. May 16, 2016.

[22] R. Lawler. 'Shadow Brokers' dump of NSA tools includes new Windows exploits. *www.engadget.com*, Apr 2017.

[23] B. Lee. *Striking Thoughts.* Tuttle Publishing, 2000.

[24] J. C. R. Licklider. Man-Computer Symbiosis. *IRE Transactions on Human Factors in Electronics*, HFE-1:4–11, 1960.

[25] L. Mathews. 70% of Executives Caved in to Ransomware Demands. Forbes, Dec 2016.

[26] J. S. Mill. *On Liberty.* John W. Parker and Son, 1859.

[27] J. Milton. *Areopagitica: A Speech of Mr. John Milton for the Liberty of Unlicensed Printing to the Parliament of England.* 1644.

[28] M. Mimoso. Bug Hunters Prefer Communication over Compensation. threatpost.com/bug-hunters-prefer-communication-over-compensation/122529. December 15, 2016.

[29] M. Mimoso. Four New Normals for 2017. threatpost.com/four-new-normals-for-2017/122584. December 28, 2016.

[30] Netscape. Netscape announces "Netscape Bugs Bounty" with release of Netscape Navigator 2.0 beta. Press Release October 10, 1995.

[31] F. Nietzsche. Die fröhliche Wissenschaft, 1882.

[32] J. O'Grady. *Early Christian Heresies*. Barnes & Noble, Inc., 1985.

[33] R. Patai. *The Hebrew Goddess*. Wayne State University Press, 1990. Third Enlarged Edition.

[34] L. G. Perdue. *In Search of Wisdom*. Westminster John Knox Press, 1993. Chapter 5: Wisdom in the Book of Job.

[35] L. G. Perdue. *Wisdom Literature—A Theological History*. Westminster John Knox Press, 2007.

[36] K. Perry. *Philosophy*. Zephyros Press, 2015.

[37] W. V. Quine. *The Ways of Paradox and Other Essays*. Harvard University Press, 1976.

[38] S. Reiny. Living in the Town Asbestos Built. *Distillations—Chemical Heritage Foundation*, Summer 2015.

[39] A. Schwartz and R. Knake. Government's Role in Vulnerability Disclosure Creating a Permanent and Accountable Vulnerability Equities Process, June 2016. Harvard Kennedy School—Belfer Center for Science and International Affairs.

[40] D. Shroyer. *Original Blessing—Putting Sin in its Rightful Place*. Fortress Press, 2016.

[41] T. Spring. Cisco Warns of Critical Vulnerability Revealed in 'Vault 7' Data Dump. *Threat Post*, Mar 2017.

[42] J. Stacey. *Liberating Jesus from Christianity: Healing from the Fear and Shame of Religious Dogma*. Page Publishing, Inc., 2015.

[43] R. M. Stallman. *Free Software Free Society: Selected Essays of Richard M. Stallman*. GNU Press, 2002.

[44] R. M. Stallman. *Free Software Free Society: Selected Essays of Richard M. Stallman*. GNU Press, 3rd edition, 2015.

[45] C. Taylor. *A Secular Age*. Harvard University Press, 2007.

[46] H. D. Thoreau. *Walden; or, Life in the Woods*. Ticknor and Fields, 1854.

[47] H. D. Thoreau. *Familiar Letters. Edited With Introd. and Notes by F. B. Sanborn*. Houghton, Mifflin & Co., 1894.

[48] C. Torres-Spelliscy. The History of Corporate Personhood. *Brennan Center for Justice, NYU School of Law*, April 2014.

[49] D. Volz and E. Auchard. More disruptions feared from cyber attack; Microsoft slams government secrecy. May 15, 2017.

[50] N. Warburton. *Free Speech—A Very Short Introduction*. Oxford University Press, 2009.

[51] N. Warburton. *A Little History of Philosophy*. Yale University Press, 2011.

[52] A. Watts. *The Way of Zen*. Pantheon Books, 1957.

[53] R. R. Williams. *Recognition: Fichte and Hegel on the Other*. SUNY Press, 1992.

[54] S. Williams and R. M. Stallman. *Free as in Freedom (2.0)*. GNU Press, 2010.

[55] D. Woods. *Biko*. Henry Holt & Company, 1978.

[56] R. Wyden. Senate address on H.R. 5949 - FISA Amendments Act Reauthorization Act of 2012. Dec. 27, 2012.

[57] A. Young and M. Yung. Cryptovirology: Extortion-Based Security Threats and Countermeasures. In *IEEE Symp. on Security and Privacy*, pages 129–140, 1996.

[58] L. Yutang and Confucius. *The Wisdom of Confucius*. Carlton House, 1938.