Hunter Owens

March 2014

Open Source Origins: The Rise of a Hacker Politic

Section I: Introduction/Abstract

Issue #2054. Migrate to MIT License.[1] This seemingly easy to solve issue on Twitter's Bootstrap framework would end up sparking a 221 comment debate on Github,[2] a code repository site in 2012 and require every tracking down contributor to the project in order to allow a programmer to use Bootstrap with another popular piece of open source software, Drupal. It would also require rewriting certain contributors commit's out of the commit log. Bootstrap, at the time, was licensed under a license called the Apache Public License (APL for short), which was not compatible with the License of the GNU[3] Public License v2 (GPLv2 for short). The most interesting part of this debate, for the purpose this paper, was that each contributor to Twitter Bootstrap (which number well over 100 at this point) was required to sign off on the eventual change of license to the MIT License. Who writes open source software?

This paper will consider the nature of authorship, responsibility and copyright in the context of open source programming and how that led to the creation of a hacker politics. Responsibility is the other side of the coin to authorship, as it comes with a obligation to the to the work. For example, it exposes you to the legal liability of a libel, which is a certain type of responsibility. There is also the more abstract understanding of that this is a piece of work by the author, that is, the nature of ownership of a work. These are all deep, historical

[1]"Migrate to MIT License." *GitHub*, n.d. https://github.com/twbs/bootstrap/issues/2054.
[2] Twitter Bootstrap is a frontend programming framework. It may be the most popular piece of software on Github, a code repository site, having the most number of 'Stars'. More at info about Bootstrap is avaliable at https://github.com/twbs/bootstrap
[3] GNU: GNU isn't Unix. A recursive acronym.

conceptions to these ideas, but for interests of space, the modern nature of the responsibility of authorship will have to the be only one considered. Responsibility in authorship of code- rather than written work or say music, takes on a new dimension in open source because hundreds, if not thousands of people, depend directly on ones work working.

The nature of open source programming is one that been considered by several excellent works, and this paper will draw heavily on them. Notably, Chris Kelty's *Two Bits: The Cultural Significance of Free Software* and Eric Raymond's *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. However, each of the books is presented by a member of the open software community. Notably, *Two Bits* is licensed under the *Creative Commons Share-Alike* license and *The Cathedral and the Bazaar* is licensed under the *Open Publications License*.[4] Although the choice to use open-source styling licensing is not the indictment of these authors, both readily admit to want to the furthering the rhetoric of open source software. Furthermore, they both participate in a particular breed of the activist open source software community and advance its rhetoric in their books. They wish to prove that open source is better than not open source. [5] However, this paper does not have a point that open source is better than closed source. Rather, this paper attempts to talk about the consider the nature of open source in a historical sense and not make the comparison to other forms of licensing. Also, the nature of these open communities [namely, the governance by public listserv] provides a bevy of primary sources to consider, but they also come with a bias of being part of the community.

The natures in which is paper aims to track - namely authorship, the responsibility of authorship, and copyright come in three distinct phases. The first software projects, such as

[4] Raymond is far more involved than Ketly.
[5] Note to the Reader: I've developed several pieces of open source software and contributed to others. I frequently use the MIT license and have used other/contributed to projects with other licensings in the past.

the *vi* text editor, created and introduced the concept of a maintainer of a project. These projects existed prior to the creation of open source as a term. It would be anachronistic to use the word 'open source' in the modern context to talk about *vi*. The maintain holds certain amount of responsibility for the project, but may not be the author or original author. Then, the famous dispute over GOSMACS and Richard Stallman, the author of *emacs*, led to what Chris Kelty calls the 'Brilliant Hack', the first open source license. However, free software licenses had predated the GNU public license. This paper will push back on the idea of open source with its origins in the Free Software Foundation. However, it will spend a significant amount of time on how the nature of authorship changed with the advent of version control systems and software licenses. Finally, turning towards the nature of modern, post-acceptance, 'professional' open source software, the paper will consider what modern licensing and distributed version control systems have done to the nature of the three subjects considered. Open source projects have encouraged a loose meaning of copyright and responsibility in other to aid the hacker ethos. The whole process leads to the development of what we consider today to be hacker politics.

Section I: Authorship and responsibility before the Beautiful Hack

What is the oldest piece of software sitting on your computer today? Well, if it's not *vi* [6] , at least *vi* is probably close. The popular text editor was created in 1976 by Bill Joy and Chuck Haley. The text editor was released with BSD-Unix and the source code was available for public inspection.[7] In 1979, responsibility of the editor was handed over to Mark Horton, who would become the maintainer of the project. A maintainer is the person in charge of making decisions about the project, adding in new code from forks, and all sorts of

[6] Vi originally was called Ex, and Vi was implemented as a hard link to the Ex for version 2.0. Vi fundamentally is Ex. Furthermore, in 1992, Vim (Vi Improved) was released. For the purposes of this paper, all Vi is distinct from Vim, but linked, while Ex and Vi are the same.
[7]Bill Joy. An Interview With Bill Joy, n.d. http://web.cecs.pdx.edu/~kirkenda/joy84.html.

administrativa. Notably, however, the do not (typically) hold copyright on the project except for the particular bits of code they add to source.

Vi is notable because it was packed with BSD[8]-Unix, one of the first widely distributed versions of the Unix operating system, however, BSD-Unix was Unix-like, not Unix. Unix was developed at AT&T's Bell Labs, which held copyright on the codebase, but distributed the source pretty widely. [9] However, as people wish to make modifications to Unix faced the problem of violating AT&T's copyright, which dictated that only AT&T was allowed to make modifications to the software. BSD-Unix was developed as a licensed offshoot of AT&T Unix, which meant it had different source code but the same external functionality for different machines.[10] At some level, it and its offshoots can be considered a fork of original Unix. However, one problem that was that the source needed to be license the distribution that allowed anybody to modify the source was needed. Thus, the BSD License was born. [11]The license contains 4 clauses, and is thus referred to as the original or 4-clause BSD license. It was leap forward in the family of software licenses, as it allowed for free distribution and modification of source code and binaries.

The four-clause license can be broken down into a key key parts. The first clause that is BSD licensed source code can not be distributed without it. The second is the same, but for binaries. The third and fourth clauses relate to advertising and promotion and endorsement by the people who have contributed to the project. Lastly, there is the as-is notice, which states the the software is provided as is and that the copyright holder (declared earlier) is not liable for anything that happens. This, at some level, is a abdication of the responsibility typically

[8] Berkeley Standard Distribution.
[9] At least as widely as Academic operating systems in the 1970's were.
[10] Some of this change is because AT&T Unix and BSD-Unix ran on different systems and would require different machine code.
[11]Copy can be found at: University of California Regents. "BSD 4-Clause License," n.d. https://spdx.org/licenses/BSD-4-Clause.

associated with professional, non-free (as in beer) software. BSD represents the first attempt at distributing software is a way that allows everybody involved in the project to contribute and provide software. This is a key part of the story of how hacker politics developed, as it was the first step in a move away from a very academic-corporate world of computing. The tinkerers are being let into the kitchen.

The BSD license however, was designed for closed source software. That is the list of contributors was expected to remain small and basically the members of UC Berkeley's Computer Systems Research Group. *Vi*'s creators/collaborators was a subset of this group and therefore, the BSD license was used to distribute *Vi*. Because Berkley decided to distribute *Vi*/BSD Unix as free (again, as in beer) software, an entire generation of hackers were raised on the software. Wide distribution of the software encourages dozens if not hundreds of users to submit bug reports back to the original team. For *Vi*, this meant that a maintainer had to be designated in order to manage this software, which may not be the creator of it as discussed earlier. Notably, although Berkley did occasionally distributed source code source for free, they were primarily in the market of distributing binaries.

BSD is a particularly niche creation of the academic-industrial complex of America. It is where computer science research was equally split among places like AT&T Bell Labs and IBM on one hand and on the other, universities such as MIT and Cal Berkeley. The License, therefore, as a few unique properties that would be problematic during the development of open source software, but were important in considering how to balance the open research agenda of the universities and the profit agenda of the corporations. BSD License software can actually cost money, which is impossible in the later generation of Licenses. Furthermore, code derived from BSD code can be redistributed as part of paid software, assuming that the authors and BSD code/license is still distributed within it. BSD, the first major, open license[12] was a product of two agendas. Many of the elements meant to project closed source and still allow the possibility of profit and introduction to the main AT&T unix project down the line. [13] *Vi* was just a small subset of this large project.

Corporate Unix was coming, especially after the success of the BSD unix project. BSD was released in 1977 and quickly become one of the most popular Unix distributions. System V was AT&T's answer to the project, and became the choice of large workstations and manufacturers, while BSD unix was typically the choice for desktop workstations. AT&T, however, regarded Unix as a trade secret. Ketly writes that "Unix was, by 1980, without a doubt, the most widely and deeply understood trade secret in computer history". [14] AT&T however, despite the popularity of Unix, viewed Unix as the expressive copyright of the company. This would bring about a long series of disputes and turf wars regarding who 'owned' Unix. Yet, AT&T distributed Unix with a reseller clause, which meant that most AT&T unixes were not actually unix, rather, they were based off of the AT&T reference distribution, leading more people to learn about the kernel and it's implementation. A terribly kept trade secret, indeed. These disputes would essentially be unsolved for nearly a decade, and it took the entire rewrite the kernel and the operating system itself under a 'true' open source license. [15] The 'Unix Wars' of the 80's were fundamental a battle over who owned this software and under what license is it allowed to be used and distributed. Unfortunately, there is no clear winner in adoption, rather Microsoft Windows would replace both BSD-Unix and AT&T System V over the course of the 1990's.

[12] Note this does not mean open source. BSD aims at causing the widest distribution of the Software, which was a major problem that plagued the original AT&T unixes.
[13] The lack of Copyleft in BSD License 4 clause one of the reasons the license is still contested over today.
[14] Kelty, Christopher M. *Two Bits : the Cultural Significance of Free Software*. Experimental Futures. Durham: Duke University Press, 2008. 129
[15] This is what eventually became Linux, or 'GNU/Linux' depending on your preferred terminology.

Section III: The Revolutionaries: GNU and the Creation of Modern Open Source

This paper discusses text editors, well, a lot. They do play a fundamental role in the role of the evolution of open source licensing, so their history should be documented. In 1976 Richard Stallman and Guy Steele authored the first version of Emacs[16], another text editor. However, Emacs was not designed to run on Unix machines, partly due to the usage of Lisp, a functional programming language. In 1981, James Gosling wrote Gosling Emacs, which relied on C and ran on Unix machines. He did this with the permission of Stallman, who held copyright on the original versions of Emacs. Gosling was distributing versions of the proprietary Gosling Emacs for free, and Stallman was fine with that. However, he did not provide the right to redistribute Gosling Emacs, so it was more of software by dictatorship rather than commune. Yet, in 1983 Gosling decided to sell his version of Emacs to Unipress, who then turned it into a product that needed to be paid for. In Stallman's mind, this was a form of software sabotage. So, Stallman decided that he would recreate a free version of Emacs for Unix. Thus, on September 27th 1983 (approximately 5 months after the April sale of Gosling Emacs to Unipress), he the famous Free Unix! email, announcing his intention to create the complete set of the GNU Unix distribution. [17] [18] The GNU project would create the GNU public license, the first truly open source license.

Richard Stallman is such a central character in the story of open source and licensing and hacking, that frankly, it is worth a bit of discussion to understand his motivations. This will lead into a discussion of what exactly is free software. First, let us consider the two tenets that inspired the GNU project. First, Stallman discusses how the 'Golden Rule' dictates that

[16] Emacs is so influential that there is a term "Emacs Pinky", referring to the repetitive strain injury caused by the editor's heavy reliance on the Alt-Meta-Control keys, which causes the pinkey to be strained using the modern IBM keyboard layout.
[17] Short for GNU: Gnu's not Unix.
[18] Found in Kelty, Christopher M. *Two Bits : the Cultural Significance of Free Software*. Experimental Futures. Durham: Duke University Press, 2008. 191-192

Stallman must share all software he uses. Therefore, he cannot sign a nondisclosure agreement or a software license agreement. Second, in order to continue using computers, he must produce a set of 'free software' so that everybody can participate. For Stallman, he writes that software needs to be both free as in freedom and free as in beer. So, Emacs becomes the first project which is part of the GNU software project. The GNU Manifesto[19] contains most of the points about how the public nature of the project requires this particular definition of freedom.

So, Stallman finishes GNU Emacs. However, he had taken several pieces of code that were copyright James Gosling (with permission) in order to make it work. In GNU Emacs 15.34, there was a 'discovery'[20] of code copyright James Gosling. This led to much discussion on the Emacs user group over the nature of copyright et cetera. However, it also led to the main distributors of commercial emacs (Unipress and CCA) to consider suing Stallman and the Free Software Foundation (who took over the copyright to GNU Emacs in version 16. It was founded by Stallman). Gosling claimed innocence on the subject of whether he had given Stallman permission, to avoid scorn from either side (Unipress and Stallman) and avoid voiding the sale. In 1985, Stallman ends the debate by deciding to rewrite the display code entirely to avoid all sections that had any copyright owned by Gosling or others. This is one of another example of ex-post-facto removal of certain authors work in a software project.

Yet, the question had been raised on the newsgroups, did Stallman release GNU-Emacs into the public domain? Because of the crazy nature of software development and the Copyright Law of 1976, it is totally unclear who exactly held the copyright to Emacs. [21] So, in order to solve the problem of who exactly owns and controls emacs and to avoid

[19]Richard Stallman. "The GNU Manifesto." *Dr. Dobbs*, n.d. https://www.gnu.org/gnu/manifesto.html.
[20] Not really a discovery, because Stallman had always distributed source code along with the binaries.
[21] Remember, it was built as an extension to another editor at first.

similar incidents with GNU projects, he created the General Public License 1.0 in 1989. The GPL is a full copyleft license. It evolved out of each of the Free Software Foundation projects DISTRIB files. The DISTRIB files are files that dictated the rules of distributing the FSF projects. DISTRIB then turned into the license for each project. For example, each license then was renamed for each project and became the GNU GPL CC, GNU GPL EMACS, GNU GPL BISON, et cetera. In order to standardize this, each was relicensed under the GNU GPL 1.0 in 1990, which is the first full open-source license and a watershed moment in the history of open source.

The creation of the general GPL was the creation of a hacker politics. It meant that now there was a political nature to these projects, and more so than that, it exists orthogonal to the standard political systems. Stallman's band of Hackers have a political motus, that is, all software should be free (as in both beer and freedom). The beautiful hack of the GPL is in fact, that is subverts the modern copyright processes, being both not public domain and therefore uncontrollable but rather controlled and free. Stallman and the FSF will spend the next couple decades arguing over the nature of free software and repeatedly refusing to give the imprimatur of the FSF to projects and licenses that do not meet their very specific definition.

However, the GNU project would have been probably a historical curiosity expect for one man's hobby. Linus Torvalds posted to the Minix-Users[22] mailing list that he was working on a new implementation of the Unix kernel and that his project. Despite the openness of the Minix system, Torvalds wished for a system that was free, as in freedom. Thus Linux was born. The decision making process for adding collaborators in Linux was centralized in Torvalds, however. Torvalds, in fact, created a centralized system in himself (and eventually a

[22] Minix was an academic distribution of Unix owned by Prentice Hall, the publisher. Because the source code was distributed with all copies, it was a popular because students would change the source code.

few key maintainers) in order advance his goals on the project. However, Torvalds was open to having new features continually added to the Kernel to be made available to the distribution. It follows what is called the bazaar model of development. Torvalds would, after his initial license called for no commercial uses of the codebase[23], allowed the codebase to be relicensed under the GPL is order to have the FSF and Richard Stallman be able to use it to help complete the GNU project. Linux is functionally a near implementation of the Unix kernel, and Torvalds writes that if a GNU project kernel had been available at the time of the launch of Linux (1991), he would not have started the project rather, he would have used the GNU version. The fact that GNU's kernel was never completed[24] shows some nature of the incompleteness of the GNU project.

One odd sidenote of history that reflects the nature of this two monumental projects is the problemings with naming that occurred between most users of Linux and Torvalds on one side and Stallman/FSF/Debian on the other side. The first camp refers to the project as Linux, while the second uses 'GNU/Linux'. The debate however the naming convention is a proxy argument about the nature of the goals of the project, is it actually a reflection of what project is. For the 'GNU/Linux' camp, the project is a completion of an ideological project that culminates in an operating system that represents the values of free (as in freedom) software. This is reflected in the Debian[25] Social Contract [26] which notes that the software is free and talks about what licenses are available to be used in order to be considered free, and they take a 'Stallman-like' definition of it. People who state that it is 'GNU/Linux' are taking an ideological stance that is far to the more conservative definition of free software. They are part of the militant free software community, most likely. However, those who just use the term

[23] Aside from that, it was functionally the same as the GPL.
[24] Given essentially a decade to do so.
[25] The (un)official distro of the Free Software Foundation, but a distinct member of the GNU/Linux camp.
[26]"Debian Social Contract," n.d. http://www.debian.org/social_contract.

Linux are part of the community that does not reflect this values. Rather, they are what Eric S. Raymond (in his seminal work, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*) terms the "quieter, less confrontational and more market-friendly strain in the hacker culture. The pragmatists were loyal not so much to an ideology as to a group of engineering traditions founded on the early open-source efforts that predated the FSF." [27] Furthermore, the pragmatic tribe descended from the pre-commercial internet and Unix cultures that have been described in detail above. They do not have problems with commercial extensions to software, rather, they 'believe' in advancing the state of software engineering as a whole. It is not entirely trite to write that these are your *vi* users, while *emacs* is used by the FSF zealot types. [28]

A case study worth examining is the Debian social contract, mentioned earlier in these pages. Debian is a distribution of Linux and is one of the most popular distributions. Debian is entirely open source and is not supported by any corporation is the sense that RedHat[29] is supported by an entirely private company. On the Debian listserv in 1997, Ean Schuessler suggest that their is a social contract in which writers of Free Software participate in, and that that relationship must be quantified and discussed. This led to the drafting (by Bruce Perens, of Pixar at the time and leader of the Debian Project) and ratification of the Debian Social Contract. [30] The social contract stems from a desire to the define the responsibilities of building free software, and as Perens notes, the forbidding the commercial use of Debian would make the project itself non-free. Neither Zeolots nor pragmatics wish to restrict the paid distribution of free software, as long as there is a corresponding free version. In a later

[27]Raymond, Eric S. *The Cathedral and the Bazaar : Musings on Linux and Open Source by an Accidental Revolutionary*. Rev. Beijing ;Cambridge, Mass.: O'Reilly, 2001. 69.
[28] This is a hillarious, if false, dichotomy.
[29] Another important distro.
[30]Bruce Perens. "Copyright Question," n.d. https://lists.debian.org/debian-devel/1997/06/msg00053.html.

email, Perens writes that “I've included a _draft_ copy of the Debian "social contract" with this message, because it gives a set of guidelines that we are considering for the free software within Debian.” [31] The social contract aims to identify what exactly constitutes free software. It stems from the request made by another debian user that nobody unduly profit from their work. This is where Perens identifies a key nature of the of open source, which is that you cannot provide the software at cost without adding value. That way, if you contribute to open source, nobody will be able to substantially profit off of your work with adding some level of innovation. This is a key part of the social contract between developers and those of the project.

The Debian social contract was not just influential when it came to debian - it would becomes the basis for the Open Source Initiative’s Open Source Definition[32] - this is the closest thing to a universal working definition of what constitutes open source. The OSI will be somewhat of an adjudicator among the open source community their mission is to “to educate about and advocate for the benefits of open source and to build bridges among different constituencies in the open source community.” [33] It is not surprising that an organization meant to bridge the two divides in the community stems of out Debian which was divided between the two camps that Raymond defies. It is fundamental the origins of a practical, workable definition that will influence the creation of the politic.

For many years, these two archetypes will become the two primary types of hacker politics. It will be the quiet pragmatists versus zealots. Linus Torvalds vs Richard Stallman. The origins of the divide date back far into the history as has been shown in this paper, rather than the relatively recent origins that many in the community think of. The divide is actually

[31]Bruce Perens. “Copyright Question,” n.d. https://lists.debian.org/debian-devel/1997/06/msg00137.html.

[32]Open Source Initiative. “Open Source Definition,” n.d. http://opensource.org/docs/osd.

[33] Open Source Initiative. “About,” n.d. http://opensource.org.

the start of a formation of hacker politics and represents the primary divide in a hacker politic. What exactly is this hacker politic? That, while somewhat the question this paper, is a open divide inside the community. The divide over what licensing means is reflected about in the community divide, which in turn, is shown in the hacker politic.

Section IV: The Origins of a Hacker Politic and how it relates to Licensing

The importance of copyright and licensing to the development of a hacker politic cannot be understated. It is clear that the development of this politics is tied to the early relationship in licensing debates. The seminal work on this subject, Gabriela Coleman's *Hacker Politics and Publics* writes " hacker and geek politics are geared toward reordering the technologies and infrastructures that have become part of the fabric of everyday life." [34] The origins of this attitude is clearly in licensing and debates about it. The structure of the work is entwined with the ethic of the workers. Hackers developed their politics in learning lessons about copyright.

The nature of writing code as a part of regular practice becomes that each author is producing copyrighted material every day, as a part of the regular task without thinking about it. This is fundamentally different from the way in which artists and writers perceive copyright, which is that the production of the thing is the value of it. For example, a writer realizes that in writing a book, their copyright of it is what is valuable. [35] However, for people who write code, the function of the code itself (ie, running it) is what is valuable and the derived copyright, while may be useful if one wants to sell the code, is not always ideal. There is value derived from the production function of producing and then running the code, but the copyright is sometimes incidental.

Examples of this become clear as one looks at how licensing lies at the first true

[34]Coleman, Gabriela. "Hacker Politics and Publics." *Public Culture* 23, no. 3 (2011): 511–516.
[35] This is obviously in a modern context of authorship and copyright.

intersection between hacker culture and copyright. However, it is important to consider the historical origins

Where does the privacy and free speech issues come from. Coleman looks towards how "[Hackers] but are deeply entangled in various distinct institutional and cultural webs and economic processes." [36] The idea of a hacker culture as a sub culture is not new, both worth looking at for at it inflicts the licensing process. Everything is done as a hack, it is not majority culture, rather each thing aims to create a sub-process for subverting (in a good way) the system. It is a mark of pride to figure out how to subvert the system and work around it, because of these economic and political processes.

One area in which the technocratic nature of the hacker politic becomes clear is when projects rewrite code in order to change licenses or such. The nature of the rewrite process, is actually, quite hilarious. In one example between BSD-Unix and AT&T Unix, a programmer sat reading out the lines of particular set of the program while another person started writing them down. It was a little bizarre, but for the purposes of the project this code had now been written by the programer who was typing, who could then promptly slap an Open-Source license on it. Yet, this would appear to be copyright violation. However, in the programmers minds, due to the nature of the source code being accessible and open and therefore having permission to look at it and use it, they can perform this quasi ritual in order to comply with the law. However, the more common situation is what is called a refactoring. The bootstrap project, mentioned early, is a good example of this. During the relicensing issue, a few contributors were unable to be contacted/refused to relicense their code under the MIT license, so they did what is called a License refactoring. [37]Refactoring refers to the process of modifying code in order to change it as part of a modular system. A license refactoring is a

[36] Coleman, Gabriela. "Hacker Politics and Publics." *Public Culture* 23, no. 3 (2011): 511–516.
[37]"Migrate to MIT License." *GitHub*, n.d. https://github.com/twbs/bootstrap/issues/2054.

reimplementation with the same behavior but a new, novel implementation. The bootstrap team followed this process, literally expunging the contributors code from the git version control history and having contributors who were okay with the MIT license. This process of expunging code is but an interest idea, the idea of rewriting history to confirm with the idea of licensing. The idea that history can be re-written in order to comply with a license is somewhat astounding.

The idea of version control, refactor and how that relates to licensing shows how the Hacker politic develops as a modification ethic, rather than a rewrite ethic. Even the zealots of the FSF do not, at least most of the time, advocate an overthrow of the entire system of governance and copyright. Rather, they attempt incremental improvements. Coleman makes the point that the "Since the commitments of hackers and geeks are not entirely of their own making, the liberally rooted political messages they herald should be familiar to most readers." [38] Rather, this it is a highly individualistic point of view, coming from the idea that the production of each person, when willfully and legally combined, will produce something better. It is certainly a results based dogma.

Concerns over DRM and other such means of copyright control and limits on ownership are directly related to the licensing debate. While in the 1980's AT&T was trying to lock down ownership of Unix while many others had to actually have access to it in order to make it work because of technical limitations, this created the origins of a free software and free culture community. However, the debate becomes more contentious when talking about culture products, not software project. There are truly two camps in the community at this point, those who believe all culture should be and those who believe non-DRM goods can be sold. This divide, is however, superseded when talking about governmental efforts to limit

[38]Coleman, Gabriela. "Hacker Politics and Publics." *Public Culture* 23, no. 3 (2011): 511–516.

copyright. The hacker community frequently argues against content-industry supported laws such as the Digital Millennium Copyright Act. The logic of why at some level is, well, the community does not like restrictions. However, laws like this make listening to music or watching videos un-free in their minds. DRM is an impinging on their freedom to do what they want with what they own.

However, what is not often understood is the response to the argument of well how can people make money off of non-drm material, or just freely published materials. This is where the licensing debate comes into play. Licenses and open source suggest, as Perens notes in his article about debian, that nobody should be able to 'steal and profit' somebodies work in open source. By applying this logic to the content debates, you get the hacker positions, which is that leaving it open doesn't necessarily mean that you cannot make money off of it. The history of the licensing debates helps inform our sense of a hacker politic.

Section V: Conclusion

The focus of this paper has to form a notion of a hacker politics that goes far deeper than the popular perception would suggest. Rather, it is the long standing evolution of a subculture on a subculture. Rather it is the nature of the debates that forged the community that influence it today. The whole process started with the Unix debacle, where a key piece of technology had unclear ownership. For those who were involved in the process of making Unix and specifically, BSD-Unix, the need for a new legal structure became clear. Thus begat the open source license, a software specific way of handling copyright after the Copyright Act of 1976 here in the United States. [39]

Open source licensing provides a framework in which hackers would begin to view the political system. The first big debates over opensource, between the Richard Stallman and the

[39] It is important to note though that those who create

Free Software Foundation (Raymond's 'Zealots') one side and Linus Torvalds and others who had a more pragmatic streak on the other. Mind you, the lines in this debate are quite fluid, but it important to note that there has always been a long history of arguing over what is the meaning of open source. The pragmatics see it as a tool, while the zealots see it as religion.

As open source continues to grow, it is important to understand what structures and episodes led to the development of norms. For example, nowadays the Apache and now Nginx web servers are widely used and almost nobody uses a closed source or commercial web server. However, the story is the opposite of what happened with Unix, which is that eventually, the MS-DOS world would take over because of the lack of structure in the unix world, and it would take a new project, Linux to bring it back into the forefront. The appeal of free software and its widespread adoption was never guaranteed, in fact, for many years, it looked completely crazy and would not even come close to being successful. It would take the model imposed by the creation of strong post-BSD (3-Clause) open source licenses that would eventually help free software come about and participate in the world of computing.

To understand the hacker politics that has grown up with this world, is becomes ideal to

understand what exactly led to the creation and interaction with politics, and it was the integral place where people discussed relationships with the law. In order for a hacker politics to be built around such issues as SOPA, PIPA and the DMCA it was necessary for the fights over licensing to occur. Licensing is the lens in which the Hacker community expresses its relationship to the law with.

Works Cited:

Primary Sources:

Bill Joy. An Interview With Bill Joy, n.d. http://web.cecs.pdx.edu/~kirkenda/joy84.html.

Bruce Perens. "Copyright Question," n.d. https://lists.debian.org/debian-devel/1997/06/msg00053.html.

———. "Copyright Question," n.d. https://lists.debian.org/debian-devel/1997/06/msg00137.html.

"Debian Social Contract," n.d. http://www.debian.org/social_contract.

"Migrate to MIT License." GitHub, n.d. https://github.com/twbs/bootstrap/issues/2054.

Open Source Initiative. "Open Source Definition," n.d. http://opensource.org/docs/osd.

Richard Stallman. "The BSD License Problem," n.d. http://www.gnu.org/philosophy/bsd.html.

———. "The GNU Manifesto." Dr. Dobbs, n.d. https://www.gnu.org/gnu/manifesto.html.

University of California Regents. "BSD 4-Clause License," n.d. https://spdx.org/licenses/BSD-4-Clause.

Secondary Sources:

Coleman, Gabriela. "Hacker Politics and Publics." Public Culture 23, no. 3 (2011): 511–516.

Kelty, Christopher M. Two Bits : The Cultural Significance of Free Software. Experimental Futures. Durham: Duke University Press, 2008.

National Conference on Open Source Software Mumbai, India), M. Sasikumar, R. Aparna, and Centre for Development of Advanced Computing. Open Source beyond GNU/Linux : Selected Readings. New Delhi: Allied Publishers.

Raymond, Eric S. The Cathedral and the Bazaar : Musings on Linux and Open Source by an Accidental Revolutionary. Rev. Beijing ;Cambridge, Mass.: O'Reilly, 2001.

Salus, Peter H. A Quarter Century of UNIX. Reading, Mass.: Addison-Wesley Pub. Co.