Who Begat Who:
The Link between Hacking and Open Source

University of Plymouth
School of Computing, Communications & Electronics
BA/BSc (Hons) Digital Art & Technology
Who Begat Who: The Link between Hacking and Open Source
Chris Saunders
April 2007

Contents

Abstract

Since the early days of the MIT Railway Club when the foundations for what we know as hacking were being laid, there has been an understanding that information should be free to be exchanged with anyone who needs or wants it. Back in those early days, the “hackers” shared code amongst themselves; they even had a drawer near their console containing utilities available to everyone to use and improve upon. They saw this as an efficient use of everybody's time by reducing the time-wasting effort of different people developing their own version of the same program. So, even though the term Open Source Software (OSS) was not first used until the late 1990s, a good 20 years later, as hacking came into being did it not also give birth to the principles of OSS?

In this dissertation I intend to explore the relationship between hacking and OSS. From the MIT Model Railway Club, I will explore their developments and, in particular, the difference between old-school and contemporary hackers. Are they continuing the tradition or are they just hi-tech vandals? With their claims that the old-school had sold out with their having been funded by the military, are today’s hackers more in tune with the original ideals than their predecessors ever were? I will also examine the different viewpoints of Lawrence Lessig and McKenzie Wark and explore how their respecive positions reflect upon the principles of hacking and OSS. Through this, I hope to show that the development of hacking has had a direct impact on OSS and that they have become co-dependant.

Introduction

The term hacking has often been used ambiguously in relation to technology. What was initially a prank performed on each other by students at the Massachusetts Institute of Technology (MIT), has since come to mean both a clever, elaborate enhancement or a crude botch to an existing program as well as, with the advent of computer crime, unauthorized access to data via a computer. To be called a hacker can be taken as both a compliment and a derogatory insult. Yet despite this ambiguity the development of computing could, arguably, be attributed to this practice. As Eric S. Raymond explains in the preface to *The Cathedral and the Bazaar*, "[hackers] built the Internet; they built Unix; they built the World Wide Web" (2001:xii).

The beginnings of hacking can be traced back to the late 1950s/early 1960s with the MIT Tech Model Railway Club, or more specifically a subcommittee called the Signals and Power Subcommittee. The members of this group were more interested in "The System", essentially what went on under the track layout. This subcommittee were the people who devised ways of controlling different trains on different parts of the track using donated telephone equipment. They would often be found looking through junk yards for parts to improve The System. As Steven Levy puts it in *Hackers*, "Technology was their playground." (2001: 23) However, improving the system was just part of it; the way in which the improvement was performed was just as important. Levy explains:

> "While someone might call a clever connection between relays a 'mere hack', it would be understood that, to qualify as a hack, the feat must be imbued with innovation, style and

> technical virtuosity[...] the artistry with which one hacked was recognized to be considerable." (2001: 23)

Although the infancy of hacking can be seen with the ingenious enhancements performed on The System, Raymond argues that:

> "the beginnings of the hacker culture as we know it today can be conveniently dated to 1961, the year MIT acquired the first PDP-1. The Signals and Power Committee[...] adopted the machine as their favourite tech-toy and invented programming tools, slang and an entire surrounding culture" (2001: 4).

This culture, Levy refers to as "The Hacker Ethic" (2001: 39-49), has six basic premises which will be explained in greater detail later in this paper. However, of these, the one at the very core of the hacker ethic is the belief that all information should be free. Hackers freely swapped programs and helped each other to improve on the efficiency of each other's code. With computers in the 1950s and 1960s having almost no systems software, to reduce the time-wasting effort of developing different versions of the same tool, they had a drawer next to the system console containing utilities available to everyone. And yet, with this principle of absolute freedom it is perhaps somewhat ironic that a lot of the work produced by these hackers was funded by the United States Department of Defence which recognises the need for secrecy. Work included speech recognition which as Levy claims "would have directly increased the government's ability to mass-monitor phone conversations abroad and at home" (2001: 150). Even the forerunner to the Internet, ARPAnet was, as Douglas Thomas in *Hacker Culture* (2002) points out:

> "originally designed as a decentralized communication network intended to maintain command, control, and communication abilities in the event of nuclear war. In essence, these early hackers designed a secure communications system funded by the Department of Defense to ensure survivability in nuclear war" (2002: 14-15).

With this obvious contradiction between total freedom and total secrecy, is it no wonder that hacking has been dogged with internal conflicts ever since.

This dissertation will examine the links between hacking and open source software and suggest that the conflicts that have appeared to have dogged it have played a large part in its progress and acceptance to the wider world. First it will look at hacking's early beginnings. It will look in more detail at the six principles that make up the "hacker ethic" with some examples to help to explain them. It will look at the beginnings of the free software movement and what prompted Richard Stallman to initiate it and finally it will look at the development and rise of Linux and the part it played in the hacker movement. In the second chapter, it will look at some of the inner conflicts such as hacker versus cracker and whether today's hackers are different to those early hackers of the subcommittee or are they more in tune with the hacker ethic than their predecessors. It will examine free software versus open source software and in particular whether it is simply a different name for the same ideal or a completely different ideology. It will also look at the viewpoints of McKenzie Wark and Lawrence Lessig and whether the struggle between hackers and those who benefit from hacking is one of class as Wark suggests and if Lessig's "Creative Commons" offers a suitable solution. Finally, it will demonstrate

how hacking and open source software are inextricably linked, becoming co-dependant and that the very conflicts have in fact driven innovation and development.

The Production of Production

During the late 1950s and early 1960s, when the Signals and Power Subcommittee of the MIT Tech Model Railway Club were constructing interesting ways of controlling the model trains and, later, writing programs on the TX-0[1], they had no idea that what they were doing would be so influential on the fledgling computer industry. What they did, they did for enjoyment. Not just in the end result but in the method and style of the enhancement. As Levy puts it "the artistry with which one hacked was recognized to be considerable" (2001: 23). During this time, a set of unwritten, silently agreed upon concepts were forming which would help to form the basis for the practice of hacking.

1. Access to computers – and anything which might teach you something about the way the world works – should be unlimited and total. Always yield to the Hands-On Imperative.

The hackers believed that by taking things apart and seeing how they work, you learned about them. They hated anything or anybody that stopped them from following this, which was particularly true if a hacker wanted to fix or improve something, as Levy points out "Rules which prevent you from taking matters[...] into your own hands are too ridiculous to even consider abiding by" (2001: 40). This is perhaps why, as Levy further documents "the Model Railroad Club start[ed][...] something called the Midnight Requisitioning Committee" (2001: 40) where hackers would wait until night time to enter previously restricted places to obtain parts. "Lock hacking" became an

[1] The TX-0 was the first transistor-run computer and was used to control a larger computer. When the larger computer was decommissioned, the TX-0 was 'loaned' to one of the labs at MIT. It pre-dates the PDP-1 by about 4 years.

important skill, even to the point of some hackers taking correspondence courses to qualify as locksmiths to allow them to buy blank keys from which they would make master keys (2001: 102-103). At one point when a high security safe was purchased, as Levy writes, “they went to a super-ultra-techno surplus yard in Taunton, found a similar[...] safe, took it back[...] and opened it up with acetylene torches to find out how the locks and tumblers worked” (2001: 104). A locked door, to a hacker, was an unnecessary restriction. As hackers believed information should be allowed to move freely and unhindered around a computer system, so should people have “access to files or tools[...] to find out and improve the way the world works[...] when a hacker needed something to help him create, explore, or fix, he did not bother with ridiculous concepts as property rights” (2001: 102).

2. All information should be free

As seen above, hackers had a fundamental belief that access to information should be unhindered. They saw the freedom of information exchange as essential for greater overall creativity (2001: 40). The TX-0 came with almost no software so if the hackers wanted it to perform a particular function, they had to write a program to cater for it including system programs to help with the act of programming. “Tools to Make Tools” (2001: 41) were kept within easy access of the systems console. Theses tools consisted of the best versions of the programs freely available to all to use as well as improve upon which the hackers saw as the logical way to reduce the time wasting effort of everyone writing their own version. But it also improved on the efficiency of the

programs themselves. The ultimate goal was "feature-full programs, bummed[2] to the minimum, debugged to perfection" (2001: 41). This belief of information being free can largely be attributed to the flow of data around a computer. If data or a device is unavailable for access then a program would crash. "In the hacker viewpoint, any system could benefit from that easy flow of information." (2001: 41)

3. Mistrust Authority – Promote Decentralization

As far as the hackers could see, the best way to promote the free exchange of information was to have an open system, or in other words, no restrictions between the hacker and the information or piece of equipment he needed to perform his hack (2001: 41). As they saw it, bureaucracies were flawed systems in that they inhibit the hacker by the implementation of rules. At the time, nothing epitomised bureaucracy as much as IBM. With processing performed on their machines in batches, the strict white shirt/black tie dress code and the restricted areas of access, this was in direct conflict with the informality of the hackers. However, it was the attitude of IBM computers being the only "real" computers that really annoyed the hackers. People who believed that were batch-processed people in the eyes of the hacker. As Levy points out:

> "Those people could never understand the obvious superiority of a decentralized system, with no one giving orders[...] [if] they discovered a flaw in the system, they could embark on ambitious surgery. No need to get a

[2] 'Bumming code' was a term coined to reduce the instructions of a program to get the same results. With memory measured in kilobytes back then, reducing the amount of code would have increased the efficiency of the program.

> requisition form. Just a need to get something done."
>
> (2001: 42)

4. Hackers should be judged by their hacking, not bogus criteria such as degrees, age, race, or positions.

To be accepted by the hackers meant that it was your ability to hack and to create new worthy programs that you were judged on than some arbitrary qualification. This is perhaps best epitomised with the acceptance into the hackers ranks of a twelve year old called Peter Deutsch. Deutsch was particularly strong in mathematics and had been writing programs on paper after reading a discarded manual. As his father was a MIT professor, he would often explore the labs and soon found his way to the TX-0 where he would stand behind so called Officially Sanctioned Users offering advice. As Deutsch's comments were invariably found to be correct, he was soon accepted as an equal by the other hackers.

5. You can create art and beauty on a computer

Although there have been many artistic projects that have relied upon the abilities of a computer with a controlling program, it was the aesthetics of the code that really captured the hackers. As Levy puts it "The code of the program held a beauty of its own." (2001: 43) With memory space at a premium on the TX-0, the hackers appreciated any techniques that allowed a program to do the same (or more) in fewer instructions. The smaller a program was, the more room there was in the memory for other programs. "Program bumming" was a very competitive contest with a lot of code trimming performed by people who could view the problem from a different,

previously unthought-of perspective.

6. Computers can change your life for the better

The hackers believed that the world could benefit from the power of the computer. It enriched their lives so why would it not do the same for everyone else? However back then, computers were viewed differently. Possibly a good example of this is one given by Levy in which a hacker called Bob Wagner, in a Numerical Analysis class, was asked to do homework using an old electromechanical calculator. Rather than doing so, he wrote a program that emulated the calculator, did his homework with it and handed it in. His professor promptly failed him not believing that a computer could be used for such a task. Levy writes,"Wagner didn't even bother to explain. How could he convey to his teacher that the computer was making realities out of what were once incredible possibilities?" (2001: 46-47)

An employee of MIT who embraced the above principles and was to become a significant figure in the hacker world was Richard Stallman. Stallman had started working in the Artificial Intelligence (AI) Lab of MIT in 1971 and soon adopted the ideals stated above, especially the freely sharing of information. As he is quoted in *Free Software, Free Society* (2002):

> "Whenever people from another university or a company wanted to port and use a program, we gladly let them. If you saw someone using an unfamiliar and interesting program, you could always ask to see the source code, so that you could read it, change it, or cannibalize parts of it to make a new program." (2002: 15)

As he puts it "Sharing of software[...] is as old as computers, just as sharing of

recipes is as old as cooking." (2002: 15) Because of this, when he identified a problem and needed the source code to fix it, he did not hesitate to ask for it.

The problem Stallman wanted to solve was, and has always been a frustrating, occupational hazard for people working with computers, the issue of printer jamming and in particular the reporting of the jam. As Williams documents in *Free as in Freedom* (2002), years before with a printer connected to MIT's PDP-11, Stallman had overcome a similar problem by writing some code that made the PDP-11 check the status of the printer and report it back to the lab's central computer, the PDP-10. He then wrote a command that made the PDP-10 broadcast a message to every user waiting on a print job that the printer was jammed. Because the message went to all parties with a vested interest in the state of the printer, there was a good chance that someone would fix the problem (2002: 3). In the late 1970s when a new, faster laser printer donated by Xerox started suffering with jams, Stallman thought nothing of implementing the same fix. However, this printer came with pre-compiled software and so he was unable to install his work-around. He contacted Xerox and requested copies of the source code but they refused. A former employee of theirs who had worked on the same laser printer was now working at another university and Stallman saw this as a way of getting hold of the source code, following the sharing, gift culture of the hacker. But, this former employee had been forced to sign a Nondisclosure Agreement (NDA) by Xerox. Although something of a standard procedure now within the realms of software development, this was something new back then and was something Stallman had not expected from a fellow hacker.

With the AI Lab's decision to scrap ITS[3] in favour of Digital's own, proprietary operating system and the poaching of fellow hackers by the company Symbolics, Stallman saw this as privately owned software intruding "on a culture he had long considered sacrosanct." (2002: 10) He kept refusing to sign NDAs despite his fellow hackers being a little more flexible morally. He became more and more isolated even to the point of calling himself "the last true hacker" (2002: 12). He saw the refusal of a request for source code as:

> "a betrayal of the scientific mission that has nurtured software development since the end of World War II, it was a violation of the Golden Rule, the base line moral dictate to do unto others as you would have them do unto you." (2002: 12)

Stallman consequently resigned from MIT and turned his attention to developing a free suite of software products. Following on from his announcement on the Usenet newsgroup net.unix-wizards where he stated that he was going to develop a Unix-compatible software system called GNU (Gnu's No Unix) and give it away for free to everyone, Stallman set about developing tools for the new system with Emacs, a text editor, being the first. Although he had asked for help in terms of money and assistance in his original post, it was not until he had released GNU Emacs that he had something to show and people started to take notice. From then on, money and requests for source code on tape began to materialise. To help with the business side of the project, Stallman enlisted some like minded hackers and formed the Free Software Foundation (FSF).

[3] The Incompatible Time-sharing System (ITS) was an operating system developed during the 1960s by the MIT hackers in direct response to an attempt by the Lab administrators to install time-sharing systems such as the Compatible Time-sharing System and Multics.

During the development of Emacs, Stallman encountered software copyright. The US Copyright Act of 1976 let programmers “take inspiration from [other programs][...] but to make a direct copy or nonsatirical derivative, they first had to secure permission from the original creator.” (2002: 123) Initially against such a concept he later modified his opinion when he saw “verbatim copying permitted” (2002: 123) included in email messages. Using this as inspiration, Stallman developed the GNU Public License (GPL). Under this license, users have the permission to run, copy, modify and even redistribute the software as long as they do not add any restrictions of their own. In other words, if a program is built from code covered by the GPL, then it must also be covered by the GPL. Of the GPL, Williams writes:

> “As hacks go, the GPL stands as one of Stallman’s best. It created a system of communal ownership within the normally proprietary confines of copyright law. More importantly, it demonstrated the intellectual similarity between legal code and software code. Implicit within the GPL’s preamble was a profound message: instead of viewing copyright law with suspicion, hackers should view it as yet another system begging to be hacked.” (2002: 127)

Stallman’s original intention was to create a free Unix-compatible system, however, it was not until the early 1990s that this was achieved and it wasn’t by Stallman. Rather it was by a 21 year-old computer student called Linus Torvalds from Sweden who initiated, arguably, one of the most significant hacker projects to date.

As reported by Williams, Torvalds was in the audience of a talk given by Stallman in 1990 and whilst he did not exactly agree with the socio-political agenda, he did appreciate the underlying message; “no programmer writes error-free code. By sharing software, hackers put a program’s improvement ahead of individual motivations such as greed or ego protection.” (2002: 136) Unlike Stallman, Torvalds had started out on home-built computers until moving up to Unix programming on a MicroVAX whilst at the university. Whereas Stallman’s main obstacle had been bureaucracy, Torvalds’ barriers had been geography and the Scandinavian winter. Making his way across the campus of the University of Helsinki to logon to the Unix system quickly made him look for a method to logon from the warmth of his room. He came across a system called Minix which had been developed by a Dutch professor called Andrew Tanenbaum that ran on a 386 clone but lacked a few features, in particular a terminal emulator which would enable Torvalds to access the university’s MicroVAX. He set about rewriting Minix, adding new features as he did. By the summer of 1991, as Williams recounts, he posted the following on the comp.os.minix Usenet news group:

> “Hello everybody out there using minix - I’m doing a (free) operating system (just a hobby, won’t be big and professional like gnu for 386 (486) AT clones). This has been brewing since April, and is starting to get ready. I’d like any feedback on things people like/dislike in minx, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).” (2002: 137)

With that simple post, Torvalds had started a revolution.

He got a few responses, one of which was from Ari Lemke at his own university who offered a sub-directory on the university's ftp server and who also, as reported by Feller and Fitzgerald in *Understanding Open Source Software Development* (2002) "persuaded [Torvalds] to adopt the name Linux" (2002: 33). Within a short space of time, version 0.01 was released and as Raymond writes "by late 1993 Linux could compete on stability and reliability with many commercial Unixes and hosted vastly more software." (2001: 16) But what was interesting about the development of Linux was that it did not follow what was largely accepted to be the recognised path of operating system development. Raymond explains, "everyone believed that any software as complex as an operating system had to be developed in a carefully coordinated way by a relatively small, tightly-knit group of people." (2001: 16) But Linux was different. Raymond further writes "From nearly the beginning, it was rather casually hacked on by huge numbers of volunteers coordinating only through the Internet." (2001:16) This is further backed up by Thomas as he explains "Something about the project touched a nerve among certain segment of the computer community, and, as a result, Linux development took on a life of its own." (2002: 83) Thomas also suggests that what helped to capture the hackers' imagination about Linux was that:

> "it was, in essence, the perfect hacker project, requiring the user to learn about his or her machine at just about every turn. If you didn't like the way Linux handled something, you could change it; indeed it was your responsibility to change it and to distribute your improvement to anyone who might be interested." (2002: 84)

But with all this distributed effort, how was Linux able to be so stable in such a

short space of time? Raymond offers this in way of explanation; “Quality was maintained not by rigid standards or autocracy but by the naively simple strategy of releasing every week and getting feedback from hundreds of users within days” (2001: 16). As Feller and Fitzgerald suggests Linux has become “the largest collaborative project in the history of the world.” (2002: 34)

By staying true to the six basic principles of the hacker ethic, Linux flew in the face of tradition when it came to its development. While proprietary software development was shrouded in secrecy and support more than often was at a price, Linux was just the opposite, no secrets and no support. If you needed a driver for a piece of hardware, you wrote it yourself, with the proviso that it was available to the rest of the Linux community. Linux, or more to the point hacking, had changed the way that large IT projects were run. No closed-source project could match the resources that Linux had at its disposal. As Raymond explains, “The hacker culture, defying repeated predictions of its demise, was just beginning to remake the commercial-software world in its own image.” (2001: 16)

This chapter has looked at the six basic ideals of the hacker ethic, the reasons behind the formation of the Free Software Foundation and the creation of Linux. The next chapter will examine the inner conflicts that have helped to shape and progress hacking and the free/open source software movement.

Free but not as in Beer

The previous chapter examined the six basic premises of the hacker movement, how and why Richard Stallman formed the Free Software Foundation and the development of Linux. Just as important to the development and advancement of the hacker and open source causes have been the inner conflicts that each have suffered. This chapter will look at some of them and propose that, whilst they might seem damaging taken out of context, they have actually been just as instrumental in their advancement as any hacker-developed technology.

Mention hackers today and invariably an image of a disenfranchised, teenager typing furiously on a keyboard at the dead of night trying to break into some corporate system is conjured up. Where this image first manifested is open to debate. Certainly the film *Wargrames* (1983) played a large part in it. The main protagonists in the film are WOPR, a computer created to control the USA's intercontinental nuclear missiles, and a game-obsessed teenage boy called David Lightman. WOPR replays different scenarios of World War Three in an attempt to discover the best strategy to win it. Meanwhile, whilst trying to find a computer games company, Lightman manages to connect to WOPR and starts playing "the game" with it. After disconnecting, WOPR continues playing and soon is unable to distinguish the game from reality and puts the world on the brink of a nuclear war. Lightman, along with the systems designer, teaches the computer to play noughts-and-crosses from which WOPR learns that nuclear war is futile, that the only way to win is not to play and disaster is adverted. Whilst Lightman is seen as the hero of the day having averted disaster, the film still manages to pose technology and hacking as a

danger. Nuclear missiles controlled by a central computer that was broken into by a teenager who then managed to put the world on red alert certainly does nothing to dispel the myth of the modern hacker mentioned earlier.

And yet, if we use the film as an accurate portrayal of modern day hacking and Lightman as an atypical hacker, was his actions not aligned with those of the earlier hackers of the 1960s? For example, he might have used a modem and a telephone line to gain access to the computer, but that was no different to the hackers obtaining blank keys from which to fashion masters in their aim to gain access to an office or piece of equipment they needed. Looking at modern day hackers, it is easy to draw parallels with them and the previous generations of hackers. Both developed their own language; whilst old school hackers "bummed" code, modern hackers are replacing characters so that "elite" becomes "3133+" and "hackers" becomes "hAck3Rz"[4] as offered by Thomas (2002: 56). More importantly, they both believe all information should be free. Whilst this meant code sharing and becoming locksmiths for the old school hackers, modern day hackers are more than likely to be sharing the security flaws of proprietary software or the secrets of faceless multinationals or oppressive regimes. Such as the actions performed by the Cult of the Dead Cow (cDc). The cDc have been helping a group of computer scientists and human rights activists called the Hong Kong Blondes in China to use "hacking as a strategy for intervention" (2002:97). As Thomas further writes:

> "technological networking, the use of secure encryption

[4] Hackers are playing with language as they do with technology by relacing letters with numbers and characters. The numbers 0, 1 and 3 for the letters O, L and E respectively and the plus sign (+) for the letter T.

> techniques, and exploiting bugs and holes in network operating systems are allowing the Blondes to effectively communicate and coordinate political action around human rights violations" (2002:97).

Exploiting bugs and holes is a common theme for the cDc. In 1998, they released a program called Back Orifice which is a parody of Microsoft's Back Office. With the ability to monitor network traffic and keystrokes including the ability to display cached passwords, the intent of the release was to highlight serious security flaws within Windows. Microsoft, the modern day hacker equivalent of IBM, rather than fix the problem, chose to deny the risk. In a marketing bulletin, it claimed that "[Back Orifice] does not expose or exploit any security issue with the Windows platform" (2002:98). A claim backed up by a member of cDc when they described them as "fundamental design flaws[...] Microsoft calls them Features" (2002:99).

Whilst parallels between old school and modern day hackers can be easily drawn, it is the differences between them that are more significant. Old school used computers in institutions such as universities, modern day are more than likely to own their own computer. Old school hackers formed companies such as Apple, Intel, Sun, even Microsoft was formed by a pair of old school hackers, modern day hackers are more likely to form groups or collectives to further their aims. However, perhaps the most significant difference is the one of secrecy. During the 1950s, the US Department of Defence financed universities in the study of computer science. Whilst the hackers were sharing code and ideas and building the foundations of a hacker ethic, what they built and produced belonged to the Department of Defence. As Thomas puts it "the

ultimate irony – work produced in the climate of absolute freedom would be deployed by the military in absolute secrecy" (2002:14). Whilst old school hackers freely shared information on projects such as speech recognition or ARPAnet, this sharing all took place within the confines of the computer lab. This practice of freedom and secrecy was further emphasized when the hackers left the universities to form companies and continued in the same vein to develop and produce proprietary software. Modern day hackers appear to view secrecy differently. Thomas writes:

> "Without military funding and without corporate secrets to protect, these hackers took up the spirit of the original hacker ethic, but from the point of view that fully contextualized it in terms of politics, economics, and cultural attitudes. These hackers held to the tenets of freedom and abhorred the notion of secrecy" (2002:15).

In fact, chances are the companies that the modern hacker tries to infiltrate are being "protected" by tools created by old school hackers.

Today's hackers appear to follow the first two premises of the hacker ethic more closely than the hackers who reputedly founded it. Whilst the old school hackers freely shared their information and skills, this was all within their labs. Contemporary hackers appear to be more open with their sharing; exposing bugs in proprietary software and secrets of large corporates and opressive regimes, distributing the source code to their projects such as Linux or Mozilla. All of this they make available to the world and not just fellow hackers. A good example of staying true to the ideal that all information should be free.

Perhaps nothing portrays conflicts better than the word “free” when used in the context of free software. Whilst it certainly has the meaning of “without cost” (dictionary.com), this is not the meaning Richard Stallman was striving for when he derived the phrase. Rather he saw free not as in price but as in limiting the controls of others, free as in free speech or free society. With regard to cost, Stallman in fact has always advocated the selling of free software. In the collection of his essays called *Free Software, Free Society* (2002), he writes that “collections of free software sold on CD-ROMs are important for the community, and selling them is an important way to raise funds for free software development.” (2002: 18) With the ambiguity of the word “free”, is there no wonder that people have tried to clarify matters by looking for an alternative.

Eric Raymond had been an active participant within the Free Software Foundation until, as Williams quotes him in *Free as in Freedom*, Stallman “kicked up a fuss about my making unauthorized modifications when I was cleaning up the Emacs LISP libraries” (2002: 156). This micro-management style forced Raymond, as it would do others, to leave but despite that, he stayed active within the free software community and would often attend conferences. During one such conference in 1996, Stallman and Linus Torvalds were invited to be keynote speakers. During the conference, Torvalds’ style, one of hacker and slacker, proved to be popular. Raymond saw that conference as “a pivotal moment” (2002: 157). Before it, Stallman was seen as the ideological leader of the hacker culture and no-one dissented from his view until Torvalds. For example, Torvalds admitted that he liked Microsoft’s PowerPoint which was no doubt quite courageous to admit at a

conference attended by hackers but he saw it as common sense. If a program was not inferior why discard it because it is proprietary to make a point. Williams writes, “Being a hacker wasn’t about suffering, it was about getting the job done.” (2002: 157). This pragmatic approach was in direct contrast to Stallman’s near evangelical stance and was a signifcant difference between old and modern day hackers.

Shortly before the conference, the Free Software Foundation had a mass exodus of staff with many citing Stallman’s style of management and his refusal to compromise on software design. This and the delays in Stallman’s operating system only confirmed to Raymond that Torvalds had hacked a new model of software project management and had found a way round Brooks’ Law[5]. Following the Torvalds method of releasing source code often and as early as possible, Raymond began work on “fetchmail”, a popmail utility. He soon received bug reports and enhancement suggestions and rather than being something of a mess, he found the resultant software to be quite robust. As Williams explains, “using the Internet as his ‘petri dish’ and the harsh scrutiny of the hacker community as a form of natural selection, Torvalds had created an evolutionary model free of central planning.” (2002: 158) Writing up his observations into a paper, Raymond compared and contrasted the management styles of the GNU project with that of Linux and in so doing, he was comparing Stallman and Torvalds. Where Stallman would micro control every aspect of a development project, Torvalds would not enforce control but keep ideas flowing and only intervene when necessary.

[5] Fred Brooks had written a book called The Mythical Man-Month and in it he states that by adding more developers to a project will actually result in further delays instead of reducing development time.

In the audience of a conference where Raymond delivered his paper on his experience with fetchmail was a team from Netscape who quickly embraced this model of software development. This earned Raymond something of a celebrity status and eager to keep the ball rolling, he visited Netscape's offices in California to give interviews and offer advice to their executives. Whilst there, he attended a meeting to discuss how other companies might follow Netscape's lead. The initial topic covered was how to name this particular approach to software development. Despite Stallman and other hackers best efforts, when hearing "free" most executives immediately associated it with "zero cost" and would dismiss it out of hand. It was during this meeting that the term "open source" was first used. Although not accepted as an "official" term until later, it was quickly adopted – but not by Richard Stallman. Stallman views open source as a completely separate movement to that of free software. As he states in *Free Software, Free Society* (2002):

> "The fundamental difference between the two movements is in their values, their ways of looking at the world. For the Open Source movement, the issue of whether software should be open source is a practical question not an ethical one[...] For the Open Source movement, non-free software is a suboptimal solution. For the Free Software movement, non-free software is a social problem and free software is the solution." (2002: 55)

Whilst this evangelical approach is admirable, Stallman's inflexible attitude has done little in advancing the cause as a whole. Williams recounts quite an ugly tirade at John Ousterhout, the creator of the scripting language Tcl,

where "Stallman called Ousterhout a 'parasite' on the free software community for marketing a proprietary version" (2002: 166). Openly castigating fellow hackers and his micro management style had not helped his cause. Certainly it would be hard to believe that government agencies in countries like Brazil and the UK and large corporates would not be seriously examining open source software were it not for the user-friendly attitude and sensible message as delivered by Raymond and Torvalds. But perhaps Stallman sees open source as more of a personal attack than a genuine vehicle to advance the free software cause. Williams quotes Raymond:

> "Richard's rejection of the term open source and his deliberate creation of an ideological fissure in my view comes from an odd mix of idealism and territoriality.[...] It's more that he so personally associates himself with the free software idea that he sees any threat to that as a threat to himself." (2002: 168)

An opinion that Stallman himself appears to enforce in *Free Sofware, Free Society* where he writes

> "We acknowledge that they [the Open Source movement] have contributed to our community, but we created this community, and we want people to know this. We want people to associate our achievements with our values and our philosophy not with theirs." (2002: 55-56)

Whilst Stallman's contribution to free/open source software is without question, his infexible attitude has arguably done little in advancing its acceptance by the IT community, and the rest of the world, as a whole. Indeed,

his micro management style appears to be indirect conflict with the third "rule" of the hacker ethic that he holds so dear – mistrust authority, promote decentralization. This certainly could not be said for Torvalds' approach. By opening up the source code of Linux to the hacker community and effectively decentralizing its development, Torvalds produced a stable, robust operating system in a fraction of the time that Stallman had been developing his version. This is clearly another difference between old and new hackers and one which also appears to be yet another instance of new hackers being more closely aligned with the hacker ethic than their predecessors.

Whatever your opinion on which terminology best fits the hacker/cracker and free software/open source software camps, arguably what lies at the core of each is the concept of property. A concept that is endorsed by McKenzie Wark in A *Hacker Manifesto* (2004) where he writes; "Only one intellectual conflict has any real bearing on the[...] issue for hackers: the property question. Whose property is knowledge?" (2004: 069) Wark equates hacking to the farming of land in the past; both result in a product created from other resources and consequently, both produce a class division. Whereas farmers worked on land owned by the then ruling classes, the lords and barons, hackers work on projects whose intellectual property rights (IPR) are owned by a class Wark has named "the vectoralist class" (2004: 020). Just as the lords became wealthy from the toil of their farmers, so do the vectoralist class by owning the IPR of the work produced by the hackers. As Wark writes, "Patents and copyrights all end up in the hands, not of their creators, but of[...] vectoralist class" (2004: 021). If access to previous hacks is restricted by copyright, how can hackers produce more work? By the very nature, a hacker

takes an existing object and builds upon it, improves it to produce yet another new piece of work. “Hacking is the production of production” (2004: 158) as Wark puts it. If the vectoralist class is restricting access to previous hacks, then surely it is restricting the creative advancement of society as a whole. Wark suggests that “free and unlimited hacking of the new produces not just ‘the’ future, but an infinite possible array of futures” (2004: 078).

Lawrence Lessig in his book *Free Culture* (2004) describes how there is a war currently taking place against piracy. He explains the idea behind the definition of piracy as:

> “Creative work has value; whenever I use, or take, or build upon the creative work of others, I am taking from them something of value. Whenever I take something of value from someone else, I should have their permission. The taking of something of value from someone else without permission is wrong.” (2004: 18)

But as he further explains, copyright law at its conception only had the creativity of an author to concern itself with i.e. the right to copy a piece of work. Now, however, its role is less about supporting creativity and more about protecting companies against competition - a notion Wark’s vectoralist class would no doubt fully agree with. Copyright has become a kind of property that can be owned, sold and protected against theft, a concept that Wark describes as “a legal hack” (2004: 020). If your neighbour has a blue car and you take it then that is obviously stealing but if you take their idea of owning a blue car and buy your own blue car, is that still stealing? Lessig makes the same point by quoting Thomas Jefferson; “He who receives an idea from me, receives

instruction himself without lessening mine; as he who lights his taper at mine, receives light without darkening me." (2004: 84) And Wark agrees:

> "the nature of information means that the possession by one of information need not deprive another of it.[...] While exclusivity of property may be necessary with land, it makes no sense whatsoever in science, art philosophy, cinema or music." (2004: 080)

The most obvious response would seem to be to remove patents and copyrights from anything intangible. Whilst this is unlikely to happen, perhaps the answer is to change their basic concepts. Lessig suggests that,

> "What's needed is a way to say something in the middle – neither 'all rights reserved' nor 'no rights reserved' but 'some rights reserved' – and thus a way to respect copyrights but enable creators to free content as they see fit." (2004: 277)

This proposal of Lessig's is the basic idea behind Creative Commons, a flexible system of copyright that promotes a creative use of the works as well as offering a layer of protection to the creator. A Creative Commons licence consisting of a legal licence, a human-readable description and machine-readable tags, allows the author to grant freedoms to anyone. For example, they can restrict use of their work to only non-commercial purposes, grant any use as long as the same freedom is granted to other users or just simply grant any use as long as credit is given to the original author. By creating a set of simple to understand, free licences, Creative Commons are effectively making a large range of content available to be built upon. Whilst being admirable and granting hackers the freedom to build upon existing works, will this help in the

hackers' struggle with the vectoralist class? Obviously by staying true to the Creative Commons licence imposed, a new hack would itself be available to be hacked and potentially prohibit vectoralists from profiting from the hacker's work. But, perhaps all this will do is just redefine the term pirate.

In this chapter we have looked at some of the conflicts that have dogged the hacker community; the similarities and differences between old and new hackers, the conception and rise of the Open Source movement in contrast with the Free Software Foundation and the class struggle between hackers and the owners of their IPR. The next and final chapter will summarise some of the main points and suggest that these conflicts have helped to shape and progress the hacker and free/open source communities.

Conclusion

From its beginnings in the 1950s and 1960s at MIT, the world of hacking has been embroiled with conflict. Promoting an open, sharing, gift culture is hard to reconcile when the principle funders of the hackers' work is possibly one of the most secretive organisations on the planet, the US Department of Defence. Taking correspondent courses to become locksmiths in order to buy master keys to gain access to other people's property is surely no less illicit than using a modem and a computer to expose corporate wrong doing and yet, one is recalled with misty eyed nostalgia and the other is criminalised. Whilst the computer industry was still in its infancy in the early days of hacking, the original hackers were arguably making the rules as they went along, creating the laws for a new world order, an order which prohibits the same levels of freedom for the modern day hacker that they themselves enjoyed. Whether or not you subscribe to the belief that all of the new breed of hacker is bent on criminal intent, it is hard to ignore that they are, without doubt, in the public consciousness more than the MIT hackers.

But what of the free/open source movement? With the honourable intentions of protecting access to source code, Richard Stallman started a movement to fight against what he saw as the encroachment of intellectual property rights into the hacker community. However, his evangelical stance to the point of fundamentalism with his berating of fellow hackers who did not conform to his ideal, and his micro-management style of project leadership lead to dissent within the movement. This paved the way for a more user-friendly approach in the shape of Linus Torvalds and Eric Raymond. Torvalds' style of project management on the development of Linux was one of macro management. He

would release changes and bug fixes to the hacker community as a whole for them to work on, relying on the idea that the style of the hack was just as important as the end result and with the hackers' need to earn kudos amongst their peers as a form of quality control. By opening up the project, he effectively released control of it and assumed the role of facilitator to keep the ideas flowing around the project team. This style of development was adopted by commercial software companies such as Netscape and quickly led to the adoption of the term Open Source. Whilst the Open Source movement and Free Software Foundation seem to stand and advocate the same thing, Stallman vehemently disagrees. He views free software as a solution to a social problem whereas open source is simply one of practicalities. Whilst not everyone can afford proprietary software, the only way that free software can achieve mass adoption is when it is also used within the workplace. If by calling it open source instead of free helps to achieve that goal, then Stallman's belief that free software solves a social problem surely will be realised.

But what of the ownership of the hackers' work? To a hacker, a hacked product is as available to be hacked further as the original item. But with software "protected" by copyright law and IPR owned by faceless corporations, this is only driving the hacker further and further down the path of criminality. To a certain extent, the GNU Public License (GPL) and Creative Commons certainly appear to alleviate that problem. By allowing the author of the original work to grant re-distribution and modification rights to it and any subsequent works based on it, appears to give hackers the opportunities for further hacks upon it. However, this does not stop the vectoralists from still laying claim to the work – the systems created by the early hackers being

claimed by the military who funded them is a good example. Certainly it stops them from restricting access, assuming that they honour the legalities, but it does not change ownership. A hacker working for a vectoralist is as unlikely to own a piece of work released under Creative Commons as a hacker working on a proprietary product; it will still be their employer who will claim ownership.

It is easy to see why the hackers' struggle with their employers can be viewed as being one of class. But as in all class wars, the only way to end it is in the form of a revolution and in this case that revolution is Linux. Torvalds rewrote the book on software development projects. By not following the common belief that complex software should be developed by small groups of developers and opening up the source code to everyone, he disproved the theory that more man power did not necessarily improve productivity. With a world wide team of hackers working at his disposal and following the six basic principles of the hacker ethic, Torvalds created an operating system so stable that not only is it used by government agencies and multinational corporations, but 30% of Internet servers run it (www.oss-institute.org). If anything epitomises the rise of hacking and open source and their impact on the computing industry and the world in general, then Linux is surely it.

The conflicts and contradictions within both hacking and free/open source software are evident. But without the differences, neither would have progressed nor be accepted in the wider world as they are today. Hopefully this will lead to even more discussion and productive antagonism.

Bibliography

Fitzgerald, Joseph Feller & Brian (2002) *Understanding Open Source Software Development,* London: Pearson Education Limited

Gay, Joshua (ed) (2002) *Free Software, Free Society,* Sebastopol: O'Reilly

Graham, Paul (2004) *Hackers & Painters Big Ideas from the Computer Age,* Sebastopol: O'Reilly

Himanen, Pekka (2001) *The Hacker Ethic,* New York: Random House

Lessig, Lawrence (2004) *Free Culture,* New York: The Penguin Press

Levy, Stephen (2001) *Hackers Heroes of the Computer Revolution,* London: Penguin

Raymond, Eric (2001) *The Cathedral and The Bazaar,* Sebastopol: O'Reilly

Stone, Chris DiBona, Sam Ockman & Mark (ed) (1999) *Open Sources: Voices from the Open Source Revolution,* Sebastopol: O'Reilly

Thomas, Douglas (2002) *Hacker Culture,* Minneapolis: University of Minnesota Press

Wark, McKenzie (2004) *A Hacker Manifesto,* London: Harvard University Press

Williams, Sam (2002) *Free as in Freedom,* Sebastopol: O'Reilly

Filmography

WarGames John Badham (1983) USA

Hackers Iain Softley (1995) USA

Sneakers Phil Alden Robinson (1992) USA

Great Artists Steal [videocassette] 1996 Channel 4 60 mins Off-air recording.

The Triumph of the Nerds, Channel 4, 28th April

Impressing Their Friends [videocassette] 1996 Channel 4 60 mins Off-air

recording. *The Triumph of the Nerds*, Channel 4, 14th April

Riding the Bear [videocassette] 1996 Channel 4 60 mins Off-air recording. *The Triumph of the Nerds*, Channel 4, 21st April

Netography

Creative Commons URL: http://creativecommons.org/ [date accessed 2 February 2007]

Stevenson, John (2000) *hacker\culture* URL: http://www.tranquileye.com/hackerculture/home.html [date accessed 20 December 2006]

Mage, Virtual (2002) *Hacker Culture and its Misportrayal by Media and Government* URL: http://www.adequacy.org/stories/2002.5.28.183725.295.html [date accessed 20 December 2006]

Mozilla URL: http://www.mozilla.org [date accessed 31 March 2007]

Open Source Initiative URL: http://www.opensource.org/ [date accessed 30 January 2007]

Stone, Chris DiBona, Sam Ockman & Mark (ed) (1999) *Open Sources: Voices from the Open Source Revolution* URL: http://www.oreilly.com/catalog/opensources/book/toc.html [date accessed 29 December 2006]

Riemens, Patrice (2002) *Some thoughts on the idea of 'hacker culture' URL*: http://cryptome.org/hacker-idea.htm [date accessed 11 December 2006]

Stallman, Richard (2002) *Stallman on Hacking* URL: http://www.stallman.org/articles/on-hacking.html [date accessed 19 December 2006]

Raymond, Eric (2003) *The Art of Unix Programming* URL: http://www.faqs.org/docs/artu/ [date accessed 2 January 2007]

The Free Software Foundation URL: http://www.fsf.org/ [date accessed 30 December 2006]

Wikipeida: Hacker Culture URL: http://en.wikipedia.org/wiki/Hacker_culture [date accessed 29 December 2006]

Lowgren, Jonas (2000) *Hacker culture(s)* URL: http://webzone.k3.mah.se/k3jolo/HackerCultures/index.htm [date accessed 29 December 2006]

Graham, Paul (2003) *Hackers and Painters* URL: http://www.paulgraham.com/hp.html [date accessed 12 December 2006]

Raymond, Eric (2006) *How To Become A Hacker* URL: http://catb.org/~esr/faqs/hacker-howto.html [date accessed 6 January 2007]

Thieme, Richard (2001) *Hacking Culture and the Hunger for Knowledge* [online] URL: http://www.thiemeworks.com/write/archives/HackerGenerations.htm [date accessed: 3 February 2007]

Mentor, The (1986) *The Conscience of a Hacker* URL: http://www.phrack.org/archives/7/P07-03 [date accessed 6 January 2007]

Modern Language Association "*free*" *Dictionary.com Unabridged* URL: http://dictionary.reference.com/browse/free [date accessed 11 February 2007]

Kingstone, Steve (2005) *Brazil adopts open-source software* URL: http://news.bbc.co.uk/1/hi/business/4602325.stm [date accessed 18 February 2007]

Marson, Ingrid (2005) *UK government promotes open source in schools* URL: http://news.zdnet.co.uk/software/0,1000000121,39196487,00.htm [date accessed 18 February 2007]

Open Source Software Institute URL: http://www.oss-institute.org/index.php?option=com_content&task=blogcategory&id=119&Itemid=127 [date accessed 26 February 2007]

Conferences

FAVE 2005, Trinity Community & Arts Centre, Bristol 20th August 2005

FAVE 2006 Limehouse Town Hall, London 25th November 2006