A Free Society Demands Free Software: Marcuse, Feenberg, and Computing in the 21st Century

Christopher Grimsley

22nd April, 2019
Dept. of Philosophy
University of Kentucky

Abstract:

In the opening pages of One Dimensional Man, Herbert Marcuse writes of "a comfortable, smooth, reasonable, democratic unfreedom" that characterizes one dimensional society in advanced industrial civilization. I will argue that as society transitioned into the information age, the same structures of control and domination were exported to the internet, software, and other new information technologies. The critical theory of technology shares important goals with what is known as the Free Software Movement, which recognizes the interrelatedness of capitalism and software development, and worries that capitalist ideologies are inextricably linked to the user experience of software designed by for profit corporations. This proprietary software has a kind of economic unfreedom written into its code, but suffers from the same problems that Marcuse identified about the consumer goods of the 20th century. The user experiences the unfreedom of proprietary software as a form of convenience. The Free Software Movement recognizes this problem, and in response advocates for a boycott of proprietary software; it urges people to choose software released under a license that allows users to read and modify the software's source code and share both the original and modified versions. Andrew Feenberg argues that technologies are not neutral, but that they can be transformed in such a way as to allow them to serve liberatory ends. I will argue that specific elements of proprietary software, namely the unavilability of its source code, permanently prevent this technology from being transformed, and as such it should be replaced by software that is unencumbered by the tendency toward domination. If we wish to build a free society, our only option is to use free software.

1 Dual-Potential Technologies: An Introduction to Critical Theory, Technology, and Computer Software

In the summer of 2006, a full year before the release of the first iPhone, I was an intern at the Maryland State Highway Administration. The SHA had just completed construction of a new, state of the art traffic monitoring facility, called the Statewide Operations Center, in Hanover, MD. Housed in this facility was the Coordinated Highway Action Response Team (CHART), the Maryland State Police, and a few other state organizations. I was given a tour of this new facility along with the other SHA interns. The facility has a large command center in a two-story tall room with one wall featuring large floor to ceiling monitor coverage, and dozens of connected computer workstations. During the tour, the facility administrator put a display on the massive monitor showing real time traffic data for Interstate 95. Each car that was traveling on I-95 showed up as a red circle overlayed on a map. An intern asked the administrator where the data was coming from. His answer was simple, "cell towers." Pressed for more information, he explained that every cell phone pings the nearest cell tower in order to stay connected to the network, but other nearby towers also hear these pings. When multiple towers all get the same ping from a cell phone, they can triangulate the position of the phone based on relative signal strength at each cell tower as well as the delay between receipt of pings between the closest tower and the farthest one. This data was being collected by the companies that owned the towers and forwarded to Maryland State Police, which shared the facility with CHART at the SOC. Again, this was in 2006, in the era of flip phones. GPS wasn't yet a standard cell phone feature, but with enough precision to monitor traffic on a major corridor like I-95, cell phone location data was already being tracked very closely.

Cell phones and cell towers represent an interesting case study. On the one hand, they can be used by regular people for the purpose of direct communication and organization; they allow for greater freedom, independence, mobility, and expressions of solidarity. At the same time, they present an opportunity for surveillance, location tracking, and other unprecedented levels of control for existing power systems. The main problem that

I wish to identify and discuss isn't the existence of dual-potential technologies like cell towers. The problem is that, despite this dual potential, the existing balance of power in society essentially guarantees that the counter-revolutionary elements of these technologies will prevail: the existence of a political, economic, and social system which allows for the concentration of these powerful technologies in the hands of the few, ensures that the possible uses of these potentially liberatory technologies will be restricted as much as possible to a narrow range of purposes, none of which support liberation. While critical theorists and philsophers of technology like Andrew Feenberg are right that there isn't necessarily anything inherent in a technology itself that forces it to be undemocratic (i.e. that technology can be transformed toward liberatory ends), because of the intense, entrenched centralization of certain technologies like cell phone towers, any democratization of such technologies would have to be the consequence, not the cause, of a broader trend toward democratization in society in general. Since these technologies are in a position where they cannot currently be used for transformative democratic ends, and because they are currently being leveraged against us by an undemocratic socioeconomic system, they are undemocratic, and should be boycotted. Absent a change in broader society, these technologies will remain undemocratic, and until such time that systemic changes allow for the preconditions necessary for the use of these technologies for liberatory ends, their use only contributes to the counter-revolutionary efforts of entrenched power systems. These technologies are currently major elements of the reactionary counter-revolution, and if we hope to escape the current forms of social domination, we must refuse to accept the use of any and all undemocratic forms of technology, even if, in principle, it can be transformed. Marcuse's Great Refusal demands that we refuse to accept the use of many technologies as they currently exist, despite the appearance in them of a liberatory potential.

In the opening pages of Herbert Marcuse's *One-Dimensional Man*, he writes of "a comfortable, smooth, reasonable, democratic unfreedom" that characterizes one dimensional society in advanced industrial civilization. One of the more insidious aspects of the dual-potential technologies described above is that their liberatory parts contribute to the

creation of this smooth sense of comfort, while the powers that wield these tools ensure that this side of the technology remains contained. The liberatory potential is allowed to shine through in such a way that it is guaranteed to be noticed, but also guaranteed not to be used effectively. It is allowed just enough expression to entice the use of the technology, while simultaneously being controlled such that the current balance of power is maintained. In Marcuse's words, the system "delivers the goods."[1]

In this essay, I will argue that as society transitioned into the information age, the same structures of control and domination that, under Marcuse's analysis, prevailed in advanced industrial civilization were exported to other new information technologies, particularly proprietary software. These technologies are not neutral; they embody the values of the system that created them. In order to track the relationship between the critical theory of technology and proprietary software, I will provide a brief review of the Marxian foundations of the critical theory of technology, identify ways in which the structures of domination identified by Marcuse have been adapted and put to use in the early 21st century, examine Andrew Feenberg's concept of the transformation of technologies through democracy, and argue that despite the existence of the potential to transform technologies, proprietary software is untransformably undemocratic, and must be abandoned in favor of free software.

2 Technology as Material and Intellectual Power: Marx and the Foundations of the Critical Theory of Technology

The invention and successful introduction of new technology can be the source of power, but also its product. For this reason it seems clear that technology can never be neutral; the very existence of any technology is necessarily the result of an existing social reality and it is a reflection of the material conditions in that reality. Once a given technology comes into existence, it exerts force back into the same social reality which gave rise to it. A feedback loop exists between society and technology. Particular social conditions give rise to particular technologies which then affect social conditions. The ability to

[1]Herbert Marcuse, *One-Dimensional Man: Studies in the Ideology of Advanced Industrial Society* (Boston: Beacon Press, 1964):xliv

dictate the future course of technology is effectively the power to determine the social future, and those who already possess power of one kind have an incentive to introduce new technologies which will solidify or enhance their power, but they are still essentially subject to various social and historical whims which could favor one technology at one time, and another at a different time. In order to discover the seeds of a potential Marxian response to the problem of advanced technology, one may look to the often quoted passage from The German Ideology, "in every epoch the ideas of the ruling class are the ruling ideas, that is, the class that is the ruling material power of society is at the same time its ruling intellectual power." [2] In many ways, the creators of influential new technologies are already part of both the ruling material power and the ruling intellectual power. The fact that their wealth stems from their innovations (ideas) shows that they represent an intellectual power, but when those ideas are put into practice, they become material power. Because of this flexibility, technological power is situated somewhere between material power and intellectual power; it has the capacity to function as both.

It is obvious that Marx was aware of the fact that the bourgeoisie made effective use of technology, but he believed that the power which exists within technology is not an independent power but rather it is subsumed under the power of class. Technology, for Marx, is merely a tool used by the Bourgeoisie, which could be put to equally effective use by the proletariat under different circumstances. Marx does acknowledge the relationship between technology and social change however, "the bourgeoisie cannot exist without constantly revolutionising the instruments of production, and thereby the relations of production, and with them the whole relations of society." [3]

It seems possible that a Marxian response to the assertion that technology is not neutral could emerge from this view. Perhaps it is the case that new developments in technology in the 21st century are simply new instruments of production which the bourgeoisie have developed and which have changed the relations of production in a typically Marxian way; innovations in advanced computer technology, according to this

[2]Karl Marx, The German Ideology in Karl Marx: Selected Writings, ed. Lawrence Simon, (Indianapolis: Hackett, 1994), 129.

[3]Karl Marx and Freidrich Engels, The Communist Manifesto, 161.

view, function just the same as innovations in 19th century industrial machinery. This counterargument would say that it is precisely because this new technology, e.g. a social networking website, is under the control of the ultra-wealthy that technology is demonstrated to be neutral, because it is serving to exploit the "labor" of the site's users, whose participation generates advertising revenue for the site's owners, but that revenue could easily be channeled away from the owners and to the "workers." The response to this counter-argument lies in Marx's discussion in the German Ideology of ruling material and intellectual powers: technology exists as both a material condition and as an idea.

Support for this response can be found in the work of Marcuse. Technology is not simply a method by which the existing power structure is maintained or reinforced but which could be otherwise if only material conditions were different. If the users of the social networking site in the above example were to share collectively in the advertising revenue their activity generated, it is unclear how this would demonstrate the use of technology for liberation rather than domination. Marcuse provides insight into the problem in *One-Dimensional Man*, "the technological a priori is a political a priori inasmuch as the transformation of nature involves that of man, and inasmuch as the 'man-made creations' issue from and re-enter a societal ensemble." [4] There is a connection between the technological, political, and social worlds. Anything created in any of these worlds is subject to the influence of the context within which it was created. Marcuse continues, noting that "a closer relationship seems to prevail between scientific thought and its application, between the universe of scientific discourse and that of ordinary discourse and behavior – a relationship in which both move under the same logic and rationality of domination." [5]

Rationality finds its expression in the intellectual and material (and thus technological) creations of the society from which it emerged. If the prevailing ideology of a society is one of domination, that tendency toward domination will manifest itself in the technology created by that society. Similarly, a society focused on liberation will

[4]Herbert Marcuse, *One-Dimensional Man: Studies in the Ideology of Advanced Industrial Society* (Boston: Beacon Press, 1964), 154.

[5]Herbert Marcuse, *One-Dimensional Man*, 155.

create technology with the same aim. Technologies can be reshaped as changing social conditions create the demand for a more efficient means to accomplish the shared goals determined by the ruling ideas. Under the Marxian conception, when the ruling ideas aim toward a particular goal, so does the technology.

3 Charting the Implications of Technology: Marcuse's Contributions to the Philosophy of Technology

Marcuse's *One-Dimensional Man* can be credited with revealing the interrelatedness of the topics of technology, domination, and liberation. This can be tied to the rationalization of society as discussed by previous thinkers such as Marx, Weber, Horkheimer, and Adorno. Under Marcuse's analysis, the rationalization of society is simultaneously the driver of progress and a force of domination. In late industrial society, rationalization is technological. New technologies give us the capacity to realize the potential for a better world but also enable the perpetuation of the existing systems of domination. Technology is not neutral - it embodies the values of the society within which it was created - but it has the capacity for both liberation and domination. As technological progress continues, there is less need for physical labor, but at the same time, the main systems of social control are based on class distinctions, so there is a conflict between the technological capacity for the end of work and the social need for control through labor. In essence, we don't have a problem of production, we have a problem of distribution; that is to say that all of the wealth is held in the hands of the few even though there is more than enough to go around. Importantly, this inequality is technologically mediated in the sense that technology enables the continuation of social stratification in important ways.

Marcuse writes of the end of dialectical thinking. Through dialectical thinking, he argues, the contradictions in society are confronted. One-dimensional society prevents this confrontation by filling in the gaps left by the contradictions with pacification strategies (goods, services, entertainment etc.). People living in this society see their needs being met and do not feel the need to oppose the system which is meeting those needs. To resist this system is irrational because resistance is harder than compliance. The con-

tradictions of late capitalism are thus never confronted, the potential for a better world is never recognized and mass movements of resistance are cut off before they can even be formed. Under one-dimensional society, domination continues because those subject to it identify with the forces that are carrying it out. Marcuse writes that "this assimilation indicates not the disappearance of classes, but the extent to which the needs and satisfactions that serve the preservation of the Establishment are shared by the underlying population" [6]. This process is made possible through the use of the technologies of late capitalism, "the social controls of technological reality [extend] liberty while intensifying domination" [7].

Within the context of 21st century software, this is nowhere more obvious than in the use of proprietary operating systems like Microsoft Windows and the proprietary software that runs on those operating systems. When a consumer purchases a license to use a proprietary operating system like Microsoft Windows, they do so because they have work that they wish to accomplish, and the use of the operating system makes it easier to accomplish this goal. The use of programs like Microsoft Word, Excel, Powerpoint, Outlook and others are far better than the previously available alternatives which existed before personal computers became available for the average consumer. It is obvious that Microsoft Word is a more efficient way to type a document than the use of a typewriter. In this way, liberty has been extended. As I will argue, there are serious problems with the use of these proprietary programs which serve to intensify domination under late capitalism.

Marcuse argues that part of the problem of one-dimensionality is the equalization of class distinctions: one of the ways in which the system smooths over the conflict and contradiction at the heart of late industrial society is to present the illusion that class conflict does not exist – that not only do the workers and the ruling class have the same overriding interests, but that they have fundamentally similar life experiences because they have access to the same commodities, which in late capitalism constitute "experience." If my boss and I watch the same TV shows, drive similar cars, listen to

[6] Herbert Marcuse, *One-Dimensional Man*, 8.
[7] Herbert Marcuse, *One-Dimensional Man*, 72.

the same music, eat at the same restaurants and so on, there is an appearance that class distinctions don't exist. While my boss may have more than I do, what there is more of is still the same stuff [8]. The illusion of the absence of these distinctions contributes to the perpetuation of the problem. Marcuse writes, "all liberation depends on the consciousness of servitude, and the emergence of this consciousness is always hampered by the predominance of needs and satisfactions which, to a great extent, have become the individual's own" [9]. If the problem is to be found in the 21st century, certainly we can find an example of it in cell phones. It would appear that the use of a smart phone is so essential to participation in contemporary culture that ownership and use of one is almost a necessity. They are used both for work and for entertainment. Importantly, advanced smart phone technology is available to rich and poor alike. iPhones are used by celebrities, politicians, police officers, bureaucrats, construction workers, nurses, and stay-at-home parents. Just as Marcuse wrote about access to commodities in the 1960's, cell phones serve to erase class distinctions in the sense that they are easily related to by nearly all sectors of society. The crucial difference between the technologies of the 1960's which Marcuse wrote about in *One-Dimensional Man* and the technologies of the 21st century is that newer technologies tend to require the use of software. I will argue that there are crucial aspects of software which distinguish it from the other forms of technology that Marcuse considered in his analysis.

4 Transforming Technology: Andrew Feenberg's Critical Theory of Technology

Cognizant both of the developments in critical theory from its roots in Marxist thought through the work of Marcuse, and of Heidegger's many contributions to the philosophy of technology, Andrew Feenberg is perhaps best situated to advance a well developed critical theory of technology. As a former student of Marcuse, Feenberg is acutely aware of the intricacies of Marcuse's work surrounding technological rationality, his critique and expansion of Weber, and his articulation of the problems associated with one-dimensionality.

[8] Herbert Marcuse, *One-Dimensional Man*, 8.
[9] Herbert Marcuse, *One-Dimensional Man*, 7.

Feenberg takes up these arguments and expands them, developing a strong and nuanced argument for the need for democracy, not just in politics and at work, but in technology as well. If democracy is important in the workplace as socialists claim, we must not ignore the problems presented by undemocratic technologies. Reshaping our economies toward a more democratic structure may not be sufficient to ensure a democratic society if we ignore undemocratic technologies. Feenberg argues that we must take note of the context surrounding the development of new technologies; we must be aware of the history of the technologies that we use, because embedded in those histories may be undemocratic tendencies that, left unchecked, pose a problem for democratic societies. It is not enough to make political life more inclusive or to shift the control of the means of production to the workers, according to Feenberg, "fundamental change requires a democratic transformation of technology."[10]

Feenberg is critical of the notion of technological determinism. Many forms of technology appear to take their current form as a result of necessity, and this appearance of necessity closes off avenues to potential change in the future. When we consider, for example, how we came to have the types of computer software that we have, we may feel like it evolved through a process similar to natural selection where the best software features survived and those features which detracted from the user's experience were removed. This illusion of a smooth progression from a new technological idea to the refined technology which we enjoy today ignores the fact that each new generation of the technology was changed as a result of choices made by the designers of the technology, and those designers are subject to the social context within which they do their work. If software companies only ever hire a certain type of person to do their software engineering, those software engineers will bring their biases and blindspots with them to their work. If all software engineers, or a large enough marjority of them anyway, are non-disabled upper middle class white males, then software will be designed in such a way as to ignore the needs of individuals outside of that group. When we then look at a finished software product and assume that it works well because it was shaped as it was by necessity, those

[10]Andrew Feenberg, *Transforming Technology: A Critical Theory Revisited*. Oxford University Press (2002):4

existing biases are reinforced. Technological artifacts have a social, cultural, and political history which cannot be ignored if we seek to democratically transform technology.

The ideology of the broader society is reproduced in that society's technology. The use of such ideology-laden technology reinforces the ruling ideas, that is, the ideas of the ruling class.Feenberg argues that in order to democratize technology, we must allow for greater democratic participation in design decisions during technological development. [11] Feenberg explains that a socity has a "horizon," a term which refers to "culturally general assumptions that form the unquestioned background of every aspect of social life." [12] This horizon then gets embedded into the "technical code," which responds to the horizon during the design of technologies, "quite down-to-earth technical parameters such as the choice and processing of materials are socially specified by the [technical] code." [13] According to Feenberg, even though our technology does not take its current form out of necessity, it still appears to; in other words, technical necessity is an illusion. This illusion is convincing because we interact with technological objects on a daily basis, but only in their current forms - though these objects carry their histories with them, those histories are difficult to access.

The decision to require seatbelts, airbags and anti-lock break systems in newly manufactured cars was contingent, not necessary. From the perspective of a user of these technologies, however, the contingency of the social relations which gave rise to the inclusion of these safety features fades into the background. Every time a person sees the "airbag" label on the airbag compartment of every vehicle, every time their car chimes to remind them to put on their seatbelt, every time the ABS light becomes illuminated, the notion that such a development was one of technological necessity is reinforced. In reality, social conditions could have been different, and they could have been such that no federal laws mandated these life-saving devices. This is, perhaps, a difficult example because in this case, the technology in question is overwhelmingly a good thing - the

[11] Andrew Feenberg, "Democratic Rationalization: Technology, Power, and Freedom" in Scharff, R.C. and V. Dusek. *Philosophy of Technology: The Technological Condition: An Anthology.* Wiley (2013): 717

[12] Andrew Feenberg, "Democratic Rationalization" 711

[13] Andrew Feenberg, "Democratic Rationalization" 714

federal requirement of the factory installation of these safety devices has saved countless lives. If we take a different example of the appearance of technological necessity, the problem of the illusury nature of technological necessity becomes more obvious. Consider for example the assumption that large scale industrial production must necessarily result in the reduction of air quality through pollution. This may currently appear as necessary as it once appeared that driving a car was inherently dangerous for the driver - that even a minor accident could result in major injuries. Of course factories produce smog; that's just the cost of doing business. The problem of the illusion of technological necessity is a problem of imposing artificial limitations of the horizon of human possibility. There is no reason to categorically rule out the creation of factories that do not diminish the air quality of the surrounding regions. Technology is developed through a process of interpersonal communication; it is contingent upon the social relationships that give rise to it, and the form it takes is never necessary - technology can be transformed so that it is the product of a more inclusive design process, and so that it is safer and more democratic. That is, technology, as it is contingent, can be shaped in such a way as to remove the built-in impediments to a more democratic society. Technology can be transformed.

Feenberg argues that viewing the course of technological development as arising from necessity results in a fallacious approach the problem of harmful technology, seeking to contain rather than transform dangerous technologies. In the car safety example above, society could have attempted to solve the problem of unsafe cars by banning cars entirely rather than requiring the inclusion of safety devices. This approach, Feenberg argues, is quite common:

> most proposals for the reform of technology seek only to place a boundary around it, not to transform it. We are told, for example, that the harm we do the environment can be reduced by returning to a more natural way of life, without cars, trash compactors, and nuclear energy. The high- tech medicalization of childbirth and dying are criticized for penetrating "too far" into zones where nature should be allowed to take its course. Reproduc-

> tive technologies are under constant attack on religious grounds. Genetic engineering is the ultimate biohazard. In all these cases critics urge us to reject certain technologies, and then ask us to accept the price of preserving tradi- tional or natural ways. [14]

According to Feenberg, the correct response is not the containment of technology, and it is not the abolition or prohibition of certain technologies - these approaches will never work. The only option moving forward is to accept that democracy must be allowed to grow into social realms that have so far shut it out: democracy must be allowed to spread to the realm of technology. In what follows, I would like to combine the contributions to the philosophy of technology from Marx, Marcuse, and Feenberg in order to analyze a problem that I see in the way that we think about computers and software. I will argue that the process of transforming technology as suggested by Feenberg has certain limitations; in some cases, a technology cannot be transformed and must be abandoned. This can occur when the development process of a particular technology is so openly undemocratic that safeguards are built-in to the technology to permanently cut off even the possibility of democratic transformation. This can occur when technology designers take explicit steps to hide from public view the way that their technology works so that to outsiders it appears only as a black box. In these cases, the technology cannot be transformed, because it is impossible to determine the ways in which transformative modifications would be carried out. In order to argue this point, I will use the example of proprietary software, which is often designed in such a way as to prevent democratic transformation. I will argue that in these cases, the only way to ensure democracy in technology is to abandon the existing undemocratic technological infrastructure and build a new, alternative, democratic technological infrastructure.

[14] Andrew Feenberg, *Transforming Technology*, 8

5 Unfree Technology is Not Always Obvious: The Importance of Access to Source Code

In this section I will describe the technical reasons why I believe that proprietary software is permanently untransformable. This will require a technical discussion of the nature of software design and development as well as a brief exploration of how computer programs work.

Computation was born with the creation of the algorithm. An algorithm, plainly put, is a step-by-step process that, when followed correctly, produces a correct answer to a specific problem. An example of an algorithm is a primality test, a test to check if a number is prime. Here's how such an algorithm could work:

1. choose an integer greater than 2, call it n.
2. if n is even, stop. The number is not prime.
3. Divide n by x where x=3, if the quotient has no remainder, stop. The number is not prime.
4. increment x by 2 and repeat step 3, incrementing x by 2 after each iteration until you have reached n.
5. If you follow step 4 until you reach n without finding a quotient that has no remainder, n is prime.

This process, if followed correctly, is guaranteed to produce a yes/no answer about whether a number is prime. The process can be completed by a computer or by a human, and if it is done by a human, the human does not need to understand what a prime number is in order to come to a correct conclusion. Algorithms are powerful because they come to definite conclusions about difficult questions in a finite number of steps. Algorithms are the basis of modern computing; they are the force behind computer programs, cell phone apps, websites, and much more. To a computer user, however, the existence of the algorithm is not obvious. The primality test above, for example, could be running on the

back end of a primality test app for a cell phone, but the user interface of the app would probably only have one entry field where the user would enter the number to be tested and one button to tell the app to start the primality test. The user benefits from the existence of the algorithm, but does not see it in action. In most cases, to the user, an application that makes use of an algorithm is a black box: the user supplies a number, and the application supplies an answer of yes or no.

Suppose in the situation above, someone using the primality test app wanted to know exactly what was going on behind the scenes. This is possible if the user is able to read the source code of the application. Source code is the human-readable computer program in some programming language that describes the algorithm(s) that the software uses. There are many reasons why someone would want to read source code. They may suspect that a program is using more hardware resources than it should, so they would like to check the efficiency of the algorithm. The primality test algorithm described above, for example, is extremely inefficient and would take more computing resources than most users would consider reasonable in order to do a primality test on some arbitrarily large number. Resources devoted to one computing task are unavailable to be allocated to other tasks, so users have an interest in making sure their programs are running efficiently.

Of course, there are much bigger problems that could be revealed by reading the source code of an application. What if my primality test app does two things: first, it checks the primality of a number and correctly displays it on your phone's screen, and second it searches through the files on your phone, and if it finds any pictures, it emails them to me. Users of software should certainly be allowed to know in advance if the software they are running on their devices is doing something malicious like this. One way to safeguard against this sort of thing is to read the source code to check that the program is only doing what it should be doing.

The problem that most users of software have with regard to source code is not that they don't know how to read it, but that it is simply not available to read even if they wanted to. When software is written, the source code is human readable, but in most cases, before it is put into production, it must be converted into a format that is machine

readable. The process of converting human-readable source code into machine-readable machine code is called compiling. When software users install software, they typically do so with what are called binary files or "binaries." This is a compiled, machine-readable program that will run on a device like a phone or a PC, but cannot be read by a person. In the case described above of the malicious primality testing app, a user has no easy way to discover the malicious activity of the program they are running. This situation may sound far fetched, but it is a relatively common occurrence. In 2017 it was discovered that a flashlight app in the Google Play store was secretly stealing online banking credentials from unsuspecting users.[15]

One may begin to wonder how, if the source code is not available, anyone could ever discover the malicious code that is concealed in binaries. The answer is that security researchers must engage in a difficult and labor intensive process of reverse engineering to discover malicious programs. It is safe to say that the number of malicious programs that are discovered is smaller than the number of malicious programs that exist. Without access to the source code of programs, it is often difficult to discover hidden malicious program components.

It is not necessary for a program to be malicious in order for it to harm users. The fact that the source code is hidden is enough to raise serious doubts about whether the algorithms that lie behind certain software are working in the best interests of program users, or if they are secretly serving some other purpose. Take as a further example, Google News. When someone does a Google search of a newsworthy item, the search results contain links to news stories at the very top of the page. How does Google decide which news articles to feature at the top of the results? Only Google is able to answer, and so far, the company refuses to release any details about the proprietary algorithms that make these determinations [16]. This could be a problem for several reasons. It may be the case that news organizations are paying Google for access to the top spot in the search results, Google may have hidden political motives that motivate the placement

[15]Dell Cameron. Flashlight Apps Snuck Malware Into Google's Play Store, Targeting Bank Accounts. 2017. url: https://gizmodo.com/flashlight-apps-snuck-malware-into-googles-play-store-1820611768

[16]Owen Williams, "Google News is broken," 2019. url: https://char.gd/blog/2019/google-news-is-broken

of certain stories, or Google may be engaging in anti-competitive business practices by featuring articles that could hurt competing businesses. Simply put, we do not know how the results are generated, we have no way of knowing, and because we do not have access to the source code, we will never know unless Google decides to tell us.

Feenberg and others argue that technology can be transformed to serve democratic ends. Unless we have access to the source code of these programs, there can never be a transformation. Absent access to the source code, the only options are to accept the technology as it currently exists, or to refuse to use it entirely. I am arguing that closed-source, proprietary software is inherently, untranformably, undemocratic. Without access to the source code, there is no way in principle to transform existing proprietary software for liberatory ends; we do not have the technological means of making the necessary changes. Software is different in this way from many of the technologies that critical theorists have explored in the past. Even the technology that lies behind the most destructive force known to humanity, nuclear weapons, can be put to work for peaceful purposes in the form of nuclear power, but pre-compiled, closed source, proprietary software is unsalvageable. For those who care about preserving freedom, the only only option is to refuse to use proprietary software.

In short, access to source code is a necessary precondition for the use of software for liberatory ends. Proprietary software was designed and created within the context of exploited labor under capitalism, it embodies the values of the system under which it was created. While it is capable of being transformed, this can only be the case if the source code is available. Since, in most cases, the source code for proprietary software is not available, it is in most cases impossible to transform proprietary software into a tool of liberation. A free society must run on free software.

6 A Free World Demands Free Software: The Difference Between Free and Proprietary Software

I have written above of dual-potential technologies. Another obvious example of this phenomenon is the Internet. One the one hand, the Internet is the most powerful surveil-

lance tool ever invented, while on the other, it has allowed for the organization and communication of technologically-minded groups of hackers who wish to resist widespread state-sponsored domination through the use of technology. Technologies tend to embody the values of the society that produced them, but in the 21st century, this characteristic causes some strange paradoxes. The values of the society that created the Internet are overwhelmingly capitalistic, neo-colonialistic, and authoritarian, but within this society exists various sub-groups, brought together by the Internet, who recognize the possibility of a better world, and who have the desire, knowledge, and the technological means to build some core elements of the technologies that might exist in a better world. Creating them is a radical form of direct action, and many thousands of computer experts have been participating in it since the late 1960's. In this way, there can simultaneously exist two alternative technologies, both produced under similar material conditions, and both embodying the values of the systems under which they were produced, but one of which has been carefully insulated from the counter-revolutionary values of the broader society, and thus has greater liberatory potential. An example of one such alternative is the GNU/Linux operating system.

I would like to paint a picture of two distinct modes of software development, both of which produce effective and useful software, but only one of which embodies the values of freedom and democracy. This is the difference between free and proprietary software. According to the Free Software Foundation, an organization which has always been at the heart of the free software movement, there are four essential freedoms that are guaranteed to the users of free software:

> A program is free software if the program's users have the four essential freedoms: The freedom to run the program as you wish, for any purpose (freedom 0). The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this. The freedom to redistribute copies so you can help others (freedom 2). The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance

> to benefit from your changes. Access to the source code is a precondition for this.[17]

These four freedoms are absent in proprietary software. The absence of freedom 0 results in a situation called "vendor lock-in." In cases where computer users depend on some software for the completion of a given task, the prevention of the user from applying that software in certain ways allows the author of the software to control the actions of the user. This could be leveraged so that the user is required to purchase access to software features that are already present but are locked by default. It could also be used to prevent users from using the software to express certain opinions when those opinions conflict with the goals of the software developer. There are real life examples of this happening: the company that makes the Ham Radio Deluxe software remotely disabled their program for users that posted negative reviews of the software online [18]. The absence of freedom 1 contributes to the types of software vulnerabilities discussed above; if software users are unable to read the source code of a program, they are unable to verify that the program is doing what its authors claim it is doing and only what its authors claim it is doing. The absence of freedoms 2 and 3 represent a prohibition on sharing. Prohibitions of this sort are unthinkable when it comes to traditional technologies; it should go without saying that I can lend my car to my neighbor if I choose to do so, but this freedom is denied for users of proprietary software. A prohibition on sharing is inconsistent with democratic ideals.

7 A Better World is Possible: Concluding Remarks

Even if one rejects the idea that technology is neutral, it makes sense to view most technology as capable of being reshaped and repurposed - to remake the values embodied within the technology, or to recreate the same technology in a way that embodies different values. I am arguing for a view that says there exists another class of technology which

[17] Richard Stallman. Free Software Foundation. "What is Free Software?". url: https://www.gnu.org/philosophy/free-sw.en.html

[18] Tim Cushing. Software Company Shows How Not To Handle Negative Review. 2016. url: https://www.techdirt.com/articles/20161220/12411836320/software-company-shows-how-not-to-handle-negative-review.shtml

resists such transformations. Some technologies resist the reshaping of the values that they embody, and this resistance is so entrenched that the only option left to those who wish to build a free world is to reject those technologies outright and rebuild them from the ground up. The absence of the four freedoms in proprietary software forecloses on even the possibility of transformative change. There already exists a project which aims not to transform proprietary software, but to replace it; this project is the free software movement. The rejection of proprietary software in favor of free software represents the first steps along this path. Marcuse's conception of the Great Refusal means refusing the "go along to get along" attitude. This may mean that one who rejects structures of domination in favor of free alternatives will face significant hardship, but Marcuse argues that freedom is worth the cost. The same is true in the case of free software. It is not easy to switch from Microsoft Windows or MacOS to GNU/Linux or OpenBSD, but such changes are necessary in order to build a more free world. Choosing to switch to free software is choosing freedom.

While the free software movement is well on its way, what remains - it is important to note that the depth and breadth of this project cannot be understated - is to abolish proprietary software entirely. I believe that making this change on a wide scale requires as a precondition the transformation of lower level societal values which will reveal the imperative of the switch to free software. Once those societal changes take place, free software will already be available, because this project is already in motion. Building the alternative is not enough; what must happen is a social change that makes the the superiority of that alternative obvious. The structure of late capitalistic society, the tendency towards domination, and the drive for the conquest of new markets serve as the driving forces behind the creation of proprietary software. In order to remove the motivation for the creation of unfree and undemocratic software, we must remove the tendency toward the construction of unfree and undemocratic power structures in society. This is the place of critical theory. A better world is possible, and free software is an important part of that vision.

Bibliography

algotransparency.org. *Daily YouTube Recommendations*. 2019. URL: https://algotransparency.org/index.html (visited on 04/09/2019).

Andersson, Hilary. *Social media is 'deliberately' addictive*. 2018. URL: https://www.bbc.com/news/technology-44640959 (visited on 04/09/2019).

Bambenek, Cadence. *Ex-Googler slams designers for making apps addictive like 'slot machines'*. 2016. URL: https://www.businessinsider.de/ex-googler-slams-designers-for-making-apps-addictive-like-slot-machines-2016-5?r=US&IR=T (visited on 04/09/2019).

Berson, Scott. *Nurse thought they gave patient something to 'relax.' It was execution drug.* 2018. URL: https://www.charlotteobserver.com/news/nation-world/national/article222422170.html (visited on 04/09/2019).

Bright, Peter. *Even when told not to, Windows 10 just can't stop talking to Microsoft.* 2015. URL: https://arstechnica.com/information-technology/2015/08/even-when-told-not-to-windows-10-just-cant-stop-talking-to-microsoft/ (visited on 04/09/2019).

Busby, Mattha. *Social media copies gambling methods 'to create psychological cravings'.* 2018. URL: https://www.theguardian.com/technology/2018/may/08/social-media-copies-gambling-methods-to-create-psychological-cravings (visited on 04/09/2019).

Cameron, Dell. *Flashlight Apps Snuck Malware Into Google's Play Store, Targeting Bank Accounts*. 2017. URL: https://gizmodo.com/flashlight-apps-snuck-malware-into-googles-play-store-1820611768 (visited on 04/13/2019).

Chaslot, Guillaume. *YouTube will stop recommending flat earth conspiracy theory videos.* 2019. URL: https://threader.app/thread/1094359564559044610 (visited on 04/09/2019).

Claburn, Thomas. *From hard drive to over-heard drive: Boffins convert spinning rust into eavesdropping mic*. 2019. URL: https://www.theregister.co.uk/2019/03/07/hard_drive_eavesdropping/ (visited on 04/09/2019).

Cushing, Tim. *Software Company Shows How Not To Handle Negative Review*. 2016. URL: https://www.techdirt.com/articles/20161220/12411836320/software-company-shows-how-not-to-handle-negative-%20review.shtml (visited on 04/09/2019).

Das, Kushal. *Tracking my phone's silent connections*. 2019. URL: https://kushaldas.in/posts/tracking-my-phone-s-silent-connections.html (visited on 04/09/2019).

Feenberg, Andrew. "Democratic Rationalization: Technology Power and Freedom". In: *Philosophy of Technology: The Technological Condition: An Anthology*. Ed. by R.C. Scharff and V. Dusek. 2nd ed. Wiley, 2013, pp. 706–719.

— *Technosystem : the social life of reason*. 2017. ISBN: 9780674982109.

— "Ten Paradoxes of Technology". In: *Techné: Research in Philosophy and Technology* 14.1 (2010), pp. 3–15.

— *Transforming Technology: A Critical Theory Revisited*. Oxford University Press, 2002. ISBN: 9780195146158. URL: https://books.google.com/books?id=X6bmCwAAQBAJ.

Fuchs, Christian. *Digital Labour and Karl Marx*. Routledge, 2013. ISBN: 0415716152,9780415716154.

Gerrard, Ysabel and Tarleton Gillespie. *When Algorithms Think You Want to Die*. 2019. URL: https://www.wired.com/story/when-algorithms-think-you-want-to-die/ (visited on 04/09/2019).

Hawkins, Andrew J. *Deadly Boeing crashes raise questions about airplane automation*. 2019. URL: https://www.theverge.com/2019/3/15/18267365/boeing-737-max-8-crash-autopilot-automation (visited on 04/09/2019).

Hruska, Joel. *Microsoft finally reveals exactly what telemetry Windows 10 collects about your PC*. 2019. URL: https://www.extremetech.com/computing/247311-microsoft-finally-reveals-exactly-telemetry-windows-10-collects-pc (visited on 04/09/2019).

Jong, Michiel de et al. *Terms of Service; Didn't Read*. 2019. URL: https://tosdr.org/ (visited on 04/09/2019).

Marcuse, Herbert. *An Essay on Liberation*. Beacon Press, 1971. ISBN: 9780807096871. URL: https://books.google.com/books?id=RY8DAQAAQBAJ.

— *Counterrevolution and Revolt*. Beacon Press, 2010. ISBN: 9780807096567. URL: https://books.google.com/books?id=tY0DAQAAQBAJ.

— *One-dimensional Man: Studies in the Ideology of Advanced Industrial Society*. Beacon Press, 1991. ISBN: 9780807014172. URL: https://books.google.com/books?id=XwC0xZU5z7kC.

Marx, K. and L.H. Simon. *Marx: Selected Writings*. Hackett Classics. Hackett Publishing Company, Incorporated, 1994. ISBN: 9781603847230. URL: https://books.google.com/books?id=vctgDwAAQBAJ.

Marx, Karl and Friedrich Engels. *The Communist Manifesto*. Rethinking the Western Tradition. Yale University Press, 2012. ISBN: 9780300163209. URL: https://books.google.com/books?id=Q1NjXzqRAkAC.

Microsoft. *Microsoft Privacy Statement*. 2019. URL: https://privacy.microsoft.com/en-gb/privacystatement (visited on 04/09/2019).

Notopoulos, Katie. *Facebook Showed Me My Data Is Everywhere And I Have Absolutely No Control Over It*. 2019. URL: https://www.buzzfeednews.com/article/katienotopoulos/facebook-advertisers-data-brokers-car-dealerships (visited on 04/11/2019).

Perez, Sarah. *US iPhone users spent, on average, $79 on apps last year, up 36% from 2017*. 2019. URL: https://techcrunch.com/2019/02/11/us-iphone-users-spent-79-last-year-up-36-from-2017/ (visited on 04/09/2019).

Rao, Harish. *Boeing: AI driven transformation*. 2019. URL: https://www.boeing.com/features/innovation-quarterly/feb2017/feature-leadership-rao.page (visited on 04/09/2019).

Sample, Ian. *Study blames YouTube for rise in number of Flat Earthers*. 2019. URL: https://www.theguardian.com/science/2019/feb/17/study-blames-youtube-for-rise-in-number-of-flat-earthers (visited on 04/09/2019).

Savage, Charlie. *Disputed N.S.A. Phone Program Is Shut Down, Aide Says*. 2019. URL: https://www.nytimes.com/2019/03/04/us/politics/nsa-phone-records-program-shut-down.html (visited on 04/09/2019).

Schneier, Bruce. *Cryptography Is Harder Than It Looks*. 2016. URL: https://www.schneier.com/essays/archives/2016/03/cryptography_is_hard.html (visited on 04/09/2019).

— *Don't Listen to Google and Facebook: The Public-Private Surveillance Partnership Is Still Going Strong and real corporate security is still impossible*. 2014. URL: https://www.schneier.com/essays/archives/2014/03/dont_listen_to_googl.html (visited on 04/09/2019).

— *Metadata = Surveillance - Schneier on Security*. 2014. URL: https://www.schneier.com/essays/archives/2014/03/metadata_surveillanc.html (visited on 04/09/2019).

— *On Surveillance in the Workplace*. 2019. URL: https://www.schneier.com/blog/archives/2019/03/on_surveillance.html (visited on 04/09/2019).

— *Stop Trying to Fix the User*. 2016. URL: https://www.schneier.com/essays/archives/2016/09/stop_trying_to_fix_t.html (visited on 04/09/2019).

— *Want to Evade NSA Spying? Don't Connect to the Internet*. 2013. URL: https://www.schneier.com/essays/archives/2013/10/want_to_evade_nsa_sp.html (visited on 04/09/2019).

— *Your Life, Under Constant Surveillance*. 2013. URL: https://www.schneier.com/essays/archives/2013/10/your_life_under_cons.html (visited on 04/09/2019).

Stallman, Richard. *What is Free Software*. URL: https://www.gnu.org/philosophy/free-sw.en.html (visited on 04/13/2019).

SwitchingSocial. *switching.social – Ethical alternatives to popular sites and apps*. 2019. URL: https://switching.social/ (visited on 04/09/2019).

Tabuchi, Hiroko and David Gelles. *Boeing Will Stop Charging to Install Safety Feature After Deadly Crashes*. 2019. URL: https://www.nytimes.com/2019/03/21/business/boeing-safety-features-charge.html (visited on 04/09/2019).

Telford, Taylor. *Anti-vaxxers are spreading conspiracy theories on Facebook, and the company is struggling to stop them*. 2019. URL: https://www.washingtonpost.com/business/2019/02/13/anti-vaxxers-are-spreading-conspiracy-theories-facebook-company-is-struggling-stop-them/ (visited on 04/09/2019).

Temperton, James. *I tried to keep my unborn child secret from Facebook and Google*. 2019. URL: https://www.wired.co.uk/article/the-internet-hates-secrets (visited on 04/09/2019).

Tozzi, C. and J. Zittrain. *For Fun and Profit: A History of the Free and Open Source Software Revolution*. History of Computing. MIT Press, 2017. ISBN: 9780262341189. URL: https://books.google.com/books?id=ICEwDwAAQBAJ.

Whittaker, Zack. *Facebook won't let you opt out of its phone number 'look up' setting*. 2019. URL: https://techcrunch.com/2019/03/03/facebook-phone-number-look-up/ (visited on 04/09/2019).

— *Stop saying, 'We take your privacy and security seriously'*. 2019. URL: https://techcrunch.com/2019/02/17/we-take-your-privacy-and-security-seriously/ (visited on 04/09/2019).

Williams, Chris. *How one developer just broke Node, Babel and thousands of projects in 11 lines of Javascript*. 2016. URL: https://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/ (visited on 04/09/2019).

Williams, Owen. *Google News is broken*. 2019. URL: https://char.gd/blog/2019/google-news-is-broken (visited on 04/09/2019).