7.

Hacking

Finn Brunton

Hack. The word is a noun, a verb, an adjective. It is a professional title and a criminal indictment and a celebration and a pejorative. It is applied to developing software, exploiting software, collecting data, manipulating social networks, working for and against companies, for and against governments, making nice things and being the reason we can't have nice things. Obama dismissively referred to Edward Snowden as ‘some hacker’ - an outsider exfiltrating data by exploiting technology, without the moral role of the whistleblower - while his administration hosted ‘civic hackathons’ and promoted a ‘culture of hacking’. Facebook fought the threat of ‘hacking Facebook’ (manipulating metrics or collecting information on their users) while hiring ‘hackers’ (virtuosic, inventive coders) and celebrating ‘the Hacker Way’: ‘an approach to building that involves continuous improvement and iteration’ (Zuckerberg 2012). A hack can colloquially mean a brilliant, elegant, lateral solution to a programming challenge, or a crude, good-enough fix in the context of constant development - ‘move fast and break things,’ to take another Facebook credo. Hacking has been closely connected with the creation of the culture, technology, and philosophy of free and open source software, and with the secretive manipulation of national elections. How can these many meanings be reconciled?

In this chapter, I will assemble what I argue are the most significant meanings of this term through a vocabulary of *actions*. Using both accounts of self-professed hackers, and assembling the literature of the study of hacking, I argue that hacking is best understood as a distinct technological way of being in the world, which we can see most clearly in a set of practices around the making, breaking, and sharing of tools, machines, communities, and systems.

Following this structure of actions avoids the confusion of this overapplied term, side-stepping the merely pejorative or vacuously positive, and helps us see what it is that hacking consistently means. By identifying and grouping research into hacking by types of action we can see hacking outside computing and telecommunications: in biotechnology, law and policy, the creation and management of spaces and communities, even philosophy and cooking. The rest of this chapter will organize our approach to hacking around eight different components of the hacker vocabulary of action: getting and giving access; tinkering and reverse engineering; recursive tooling; commoning; making nonstandard things; performing virtuosity; defining and policing hackerdom; and social engineering. Not every hacker engages in every one or even most of these forms of action. Some were more prevalent at one time than another. Some blur a bit from one to the next but have unique, distinctive traits. Together, these constitute the action, the doing, that makes hacking what it is.

Getting and Giving Access

This is arguably the foundational act of hacking; the other possibility, which has to do with pranks, is part of the last of the eight actions. A thread that runs through popular, scholarly, and personal accounts of being a hacker is the act of *getting access:* to computers, to telecommunications systems, to knowledge, to source code, to tools, to the inner workings of machines and networks, to accounts with escalated privileges, to secrets and classified information. This thread is everywhere intertwined with *giving access:* with sharing, circulating, and further distributing one's access - providing passwords, how-to guides, commented code, phone numbers, files, specs, standards, and specialized screwdrivers and spatulas for getting inside the cases and containers of the technology.

Many seemingly disparate elements that are recognizably part of hacker culture come together in this fundamental action. The first point of Steven Levy's summary of the 'hacker ethic' as he described it in the early days of hacking at the Massachusetts Institute of Technology (MIT) is: '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!' (Levy 1984: 28). As with all of the entries in our vocabulary of hacker action, *getting and giving access* fuses abstract beliefs with practical goals and activities (see also Thomas 2006 on hackers and getting access to secrets). The hands-on imperative can explain the popularity of locksport - competitive lock-picking - in the hacker community. One of the liveliest corners of the biannual Hackers On Planet Earth conference is the area devoted to picking locks, where participants can get training and face challenges from padlocks to an entire payphone (personal observation). Locks are very sophisticated technological puzzles - riddles in manufactured form - that reward cleverness, persistence, logical problem-solving, and focus; they also happened to often prevent access to computers, telephone equipment, and other interesting gear. The same nascent hackers interested in picking locks in the 1970s were also interested in the pre-Internet Bulletin Board System (BBS), where 'discourses and texts about hacking were ubiquitous' (Coleman 2013: 30; see also Driscoll 2016). BBSs were likewise about getting and giving access: to information, including information about getting further information, with the fundamentals of hacking into remote systems, phone numbers for other more distant BBSs, typed-up *samizdat* textfiles of science fiction stories and conspiracy documents - and instructions in lockpicking. For the subculture of phone phreaks (Lapsley 2013) - another fecund space for what hacking would become - the practical benefit of free long-distance phone calls, which enabled ‘party line’ communities to form among far-flung proto-hackers, was secondary to the act of *getting access,* not just to the phone network itself but to knowledge about it: they knew AT&T's system even better than her own engineers.

This fundamental action had an essential moral dimension (Coleman puts it in the context of the political history of liberalism [2013: 116-122]): from access, from the hands-on imperative, came knowledge and understanding - and with them, the responsibility of sharing knowledge with others in turn even if one faces legal consequences for doing so. One of the foundational objects of hacker inquiry, the Unix operating system (‘our Gilgamesh epic,’ Neal Stephenson called it in his essay on hackers and the design of operating systems [1999: 88]), was the property of Bell Labs. It circulated through generations of photocopies of an educational commentary on the source code: *Lions' Commentary on UNIX,* very likely the most copied book in the history of computer science - you would make one copy for yourself, and another for your friends (Lions 1977) (Unix and the work of giving access will come up again below, in ‘Commoning’, in the context of free, libre, and open source software.). The act, and the conviction, of giving and getting access has far-reaching consequences; it can be found even in very early and hacker-adjacent documents and projects, like the *Whole Earth Catalog* and the cyberculture movement chronicled by Fred Turner, which was premised on ‘access to tools’ (Turner 2008: 81). It appears in latter-day initiatives, like the One Laptop Per Child (OLPC) project, which sought to manufacture a cheap laptop whose design - built for tinkering and creating one's own software in an open-source framework - was explicitly meant to foster a new generation of hackers. Begun at MIT, like the word ‘hacking’ itself, it was a vision, combining hubris and altruism: that giving access was all you needed to start children on the path to hacking.

Tinkering and Reverse Engineering

The OLPC was a dream of giving access: not only to computing as such, or some suite of applications, or the Internet, but to a machine that invited *tinkering.* This is the second of the eight parts of the hacker vocabulary, and one that can be described more briefly than the first. It is two faces of the same phenomenon, a hands-on facet of getting access: the drive to take apart, to fiddle, to modify, to take pre-existing technologies and figure out how they work and how they can be made to work differently. It is a category of action we can partially observe in negative, through all the components developed and deployed by manufacturers and corporations to keep people out. The hacker drive to tinker and reverse engineer, particularly with electronics and digital technologies, is reflected in the prevalence of esoteric screws - pentalobe, hexaloblular - glued-down (rather than screwed-in) panels, holographic tape and other ‘tamper-evident’ details, warnings of ‘no user-serviceable parts inside’, nonstandard connectors and proprietary drivers, and encrypting the traffic between chips on a device. Whether used to protect a digital rights management (DRM) scheme to control the circulation of content, or to prevent competitors from cheaply duplicating a device, hackers often take these components as an affront and a challenge. What could be more interesting than out-thinking an entire company's worth of engineers and security specialists?

Bunnie Huang is one of the preeminent examples of this area of hacker activity: among other exploits, he famously hacked the Microsoft Xbox - figuring out and unlocking how it secured its internal communications - and has produced close analysis of the vulnerabilities of digital storage systems like microSD cards. ‘Without the right to tinker and explore,’ Huang wrote, ‘we risk becoming enslaved by technology; and the more we exercise the right to hack, the harder it will be to take that right away’ (2013: np). Huang makes clear throughout his work how intertwined these two activities are. Many hacker stories begin with tinkering, trying to fix some minor problem, or get a device or program to do what it wasn't exactly built to do, and in pursuit of that goal end up reverse engineering the whole of the object's operations; others involve a massive project of ‘undesigning’ and reverse engineering some elaborate apparatus so it can be playfully tinkered with. The tinkering can sometimes be for the sake of straightforward goals, like ‘overclocking’, making chips and computing architectures deliver faster and more powerful performance than their specifications suggest. But often it can be for more quixotic goals - complex for the sake of being complex, impressing other hackers who understand how demanding such a trick was to pull off. A classic example of the latter is getting the classic video game *Doom* running on some ridiculously inappropriate and unlikely platform: on a Kodak digital camera from the early 2000s, an ATM, a seatback in-flight entertainment system, the screen of an MP3 player, even a printer’s display.

To take a light-hearted example, consider hacker Natalie Silvanovich, who has done a series of in-depth projects to document and completely understand tamagotchis - yes, the keychain-sized ‘artificial pets’ that live on LCD screens, fed and pampered through a few buttons (Silvanovich nd). Silvanovich's reverse engineering efforts including using nitric acid, microscopes, ROM dumps, and painstaking analysis to access and decode the tamagotchi hardware and software to 'answer the "deeper questions" of Tamagotchi life.' Her project entails applying a full toolkit (literally and figuratively) of hacker training and techniques to a deliberately fun and silly goal - but one with serious implications, a part of the cultural continuity of hacking, from the phone phreaks mapping out Bell Telephone's network to the people who got Linux running on the Nintendo Switch handheld gaming console earlier this year (Julie Cohen has analyzed the question of a limited right to self-help this raises: 'freedom to tinker,' or 'the right to hack.' [2012: 219]). To be able to reverse engineer, and to open devices and systems to tinkering, is to expand the spaces where hackers can take action and where future hackers will emerge.

Recursive Tooling

Bunnie Huang has also worked as a manufacturer, focusing on producing 'open hardware' products which encourage their own user modification, reinvention, and development. One of the best examples of this kind of product, Huang's Novena laptop, leads us into the third part of the hacker vocabulary of actions: the reflective project of making the tools you need to make the tools you need for the creation, modification, and tinkering you want to do. Where most laptops are sealed and inaccessible to the user, Huang's is an open box of components: to tilt the screen, you have to expose all the internals. To get it to do anything you have to install parts and an operating system and figure out how to get the components to interoperate: to get to the stage where you could, for instance, compose an email, you would have to develop expertise and install the systems to get the box on the Internet with a working mail client; to get it on the Internet, you would have to get the operating system transmitting over an antenna or an Ethernet jack; to get the operating system working ... and so on. A recurring theme in hacker stories is a breakthrough that happened in the course of trying to develop better tools for some other purpose, whether an improved programming language, a versioning system to reconcile different parts of a project, or a whole operating system, in the case of Unix, created with an eye to making the creation of future tools faster and easier.

Like tinkering and reverse engineering, this element of the hacker approach has both smaller everyday and larger abstract aspects. Recursive tooling appears as jokes in the hacker lexicon around things like ‘yak-shaving’: ‘some stupid, fiddly little task that bears no obvious relationship to what you’re supposed to be working on, but yet a chain of twelve causal relations links what you're doing to the original meta-task’ (Brown 2000). You started out trying to update a dependency and ended up learning a new programming language. The legendary computer scientist Donald Knuth, for instance - adopted as a hacker patron saint - became frustrated at the inferior quality of the typesetting and design tools available for publishing his work in the 1970s. He ended up creating a complete, immensely complex layout system, TeX, which became the basis for LaTeX, the default standard for mathematical notation and publishing in the sciences to this day (Knuth 1986). In classic yak-shaving style, to get TeX to work, Knuth developed not only his own programming language for it, by an entirely new theory of how programming could work - and a custom digital font, still in wide use.

On a broader scale, recursive tooling appears as the political arrangement Christopher Kelty identified as the *recursive public:* ‘this kind of public includes the activities of making, maintaining, and modifying software and networks, as well as the more conventional discourse that is thereby enabled ... [a] series of technical and legal layers - from applications to protocols to the physical infrastructures of waves and wires - that are the subject of this making, maintaining, and modifying' (Kelty 2008: 29) He continues: '[G]eeks use technology as a kind of argument, for a specific kind of order: they argue about technology, but they also argue through it. ... They express ideas, but they also express infrastructures through which ideas can be expressed (and circulated) in new ways' (ibid). Hackers understand themselves as larger communities in terms of the tools that enable their communities, tools they themselves design, develop, and deploy. Arguments over a messaging or versioning system or the software of a mailing list can work on several levels at once: personal, political, technical, infrastructural. The hacker activity of recursive tooling also plays out as the hacker community of the recursive public.

Commoning

Of course, it also plays out in the question of whether and how to circulate and share those tools. Kelty (2008) was writing about the Free Software movement, as was Coleman (2013, 2014), earlier. It is a source of considerable public and scholarly interest: a new way of making things, social and technical at once, that Yochai Benkler terms 'commons-based peer production'. Benkler summarizes the idea of a commons: 'a particular institutional form of structuring the rights to access, use, and control resources ... [R]esources governed by commons may be used or disposed of by anyone among some (more or less well-defined) number of persons, under rules that may range from "anything goes" to quite crisply articulated formal rules that are effectively enforced' (Benkler 2006: 61). I want to identify 'commoning' as a particular form of action we've seen before in this chapter and will see again. In its most general, generic form, this is another facet of getting and giving access (Johns 2009: 463-496). Often what is being put into commons is the information necessary for tinkering or developing one's own tools - or information that is being leaked or exposed, as will be discussed below. Commoning is distinct, however, as the most conceptually rigorous version of this related set of actions. In a sense it is the most meta-level kind of recursive tooling, creating a legal, political, economic, and cultural environment as well as a set of technical tools for formalizing the getting and giving of access.

Commoning, then, is a verb that takes us beyond giving or sharing. Dumping a bunch of digital media into some online repository is not commoning, as such. Commoning is making use of things like the GNU Public License (GPL), 'copyleft' provisions, Creative Commons licenses, and other open source frameworks. (If you are reading this digitally, the screen you read it on very likely includes some of these frameworks somewhere in its software). Commoning is, more tangentially, the creation of open data, open access, open publishing, open hardware, and open standards. Commoning is engagement in ongoing debate about what one means by these very terms: 'open' or 'free' as in whether you have to pay, or whether you can do anything you want? As in transparency? As in having to participate in the commons in turn, with what you produce? 'Open' as a canny business decision, or as a philosophical commitment to a specific understanding of knowledge and society? Android, the mobile phone operating system initially produced by Google, exemplifies the former; Richard Stallman, founder of the Free Software Movement, who exemplifies the latter, described putting together a collection of free software necessities '[s]o that I can continue to use computers without dishonor ... I refuse to break solidarity with other users' (Stallman 1985). Rational, righteous, or both?

As with the other components of hacker action, commoning can cover many degrees of action for an array of goals. One can engage in commoning by posting a picture under a Creative Commons license, by making a contribution to a free/libre/open source project, by running Linux or teaching others to use it, by designing and manufacturing an open source piece of hardware which can become the basis for other devices (like the famous Arduino platform), by sharing exfiltrated data with the public for a specific end, by engaging in what Aaron Swartz called ‘guerilla open access,’ moving large bodies of knowledge into the commons even if the project is unsanctioned, or illegal: ‘We need to take information, wherever it is stored, make our copies and share them with the world’ (Swartz 2008). (Swartz faced disproportionate legal penalties for his guerilla open access downloading of a massive set of academic journal articles, leading to his suicide in 2013.)

This final aspect of commoning has taken on a new significance in the last decade and a half as more and more social and political institutions have moved their operations online. There were prior cases of hacking for disclosure - to share concealed information - but the formal role the hacker occupies as whistleblower has changed: ‘the politically engaged geek family continues to grow - in size and significance,’ wrote Coleman in her study of Anonymous (Coleman 2014: 382). Edward Snowden’s release of NSA materials to journalists, the creation of the WikiLeaks model by Julian Assange and his collaborators for online publication, the attacks on Sony’s movie division and the Ashley Madison site by unknown hacker teams - both of which involved dumping massive caches of documents online for the public to comb through - suggest the scale of this transformation in what it can mean to be a ‘hacker’. As Benkler explains in his study of the release of emails related to vulnerabilities in the Diebold company's voting machines, hacking as commoning creates its own infrastructure of sharing, including 'the initial observations of the whistle-blower or the hacker; the materials made available on a "see for yourself" and "come analyze this and share your insights" model; the distribution by students; and the fallback option when their server was shut down of replication around the network' (Benkler 2006: 262).

As even this brief list suggests, the act of commoning can take place in a mix legally sanctioned frameworks (themselves often the product of hackers and lawyers and journalists formalizing hacker commitments), or as appeals to a higher moral authority: Stallman's 'solidarity,' Swartz's call for informational 'justice'. Finally, like many of the actions outlined here, commoning is reciprocal, to do with both giving and getting - one puts things into the commons, but also draws on them: the other part of Benkler's 'commons-based peer production'. This brings us back to tools, hardware, and software. You need components and data offering the kind of privileges that commoned objects do in order to make many hacker things. What kind of making - what kind of production, what kind of labor - needs that level of access?

Making Nonstandard Things

McKenzie Wark identified what he called 'the hacker class' as a way to talk about two things. The first was a larger question he identified as 'the nature of information itself as something inimical to property and necessarily existing only as something shared' (2017: 306). It is an issue that should feel familiar to us now as a part of the toolkit of hacker action, a question that builds on his earlier *Hacker Manifesto* (Wark 2004). The second was to have a way to talk about the people engaged in '(non-)labor practices that make *nonstandard things,*' 'new things' (2017: 9).

There are many aspects of this idea, including the very hacker-ish question of the blurring between labor and play, experiment, art, science, and performance (see the next action), with implications for how we discuss issues from compensation, to economics, to unionization in the tech industry (see also Scholz [ed.] 2012, Liu 2004, Neff 2015).

However, I would like to highlight a different aspect of the fifth of the eight hacker actions: a culture of craft, with a specific aesthetic - one that connects the previous set actions with the one that follows below. While their work may have widespread effects, appearing in templates, libraries, dependencies, and other widely reproduced, standardized formats and components, a hallmark of hacker production is 'nonstandard things': bespoke tools and products, modified versions, idiosyncratic designs, one-off fixes and solutions whether elegant or crude. As hacker Rodney Folz put it, 'We do things that don't scale. It's in our blood' (Folz 2015). That last thing, the ugly but effective and expedient fix to an immediate problem, even has its own hacker jargon: a 'kludge' - 'an improvised, spontaneous, seat-of-the-pants way of getting something done,' as Lisa Nakamura put it, which was sometimes also called a 'hack' in the early days of the term (Nakamura 2006: 318). As the specialized terminology suggests, these kinds of fixes are commonplace, in the spirit of tinkering and developing one's own tools: sometimes the goal might be logical, clean-slate perfection (a recurring hacker temptation) but more often is just to get something working well enough for now.

This nonstandard making can be seen in many aspects of hacker labor but I'll mention two that I believe to be particularly pertinent. The first is in comments and documentation for code. These written materials are intended to provide guidance to the person who is using, reviewing, or modifying the code. They can include text explaining a program and its commands - like the pages one receives for a ‘man’ (for ‘manual’) request in the Unix or Linux command line - and text written into the program itself, bracketed out so the computer won't try to interpret or execute it, and the human can read it. One would assume that such technical documentation would be dry, impersonal, the expression of a standardized approach to a standardized product - akin to the owner's and mechanic's manuals printed for cars, for instance, a canonical standardized assembly-line machine. But hacker documentation is a textual culture of its own, delightfully personal, sometimes sardonic, frustrated, or gnomic, filled with in-jokes and reflections on the work itself. Sometimes documentation and comments are agreeably ragged: admitting the code could be better, noting an unfinished feature, an experiment that never panned out, a kludgy fix that the programmer will come back to one day. Sometimes they reflect pride in craft - including the warning not to mess with part of a design that you probably don't understand. (Coleman has written extensively about the culture of commented code [2013: 100-120].) Famously, *Lions' Commentary on UNIX* included a comment on a very weird mechanism on line 2238: ‘You are not expected to understand this.’ Though meant as ‘this won't be on the test,’ it was often interpreted and playfully riffed on in other code as a challenge -- ‘don't even try.’ The page of text you get for ‘man rsh,’ the manual for a Unix program, includes this explanatory line for a command: ‘this is arguably wrong, but currently hard to fix for reasons too complicated to explain here.’ These kinds of notes reflect code that is made by people in personal, inventive, nonstandard, crafty ways - one stitch at a time - and code made with the expectation it will be tinkered with, studied, and further modified.

The second, which I will touch on only briefly, is *how* hacker production often happens: using heavily modified, nonstandard systems and environments. This is closely related to the activity of recursive tooling, making the tools with which to make the tools. A classic hacker rite of passage is not just installing an open source operating system, but installing that system and then modifying it until it breaks, and then fixing it again. From choosing window managers to customizing text editors to remapping keyboards, nonstandard objects are made with nonstandard tools: a minor but telling detail of hacker life is developing and sharing one's own ‘dotfiles’, the configuration files normally hidden from the user, with which you can specify your own preferences for how work is done. Over time, each hacker's production environment will become unique, engineered for their particular, nonstandard practices.

Performing Virtuosity

One consequence of making nonstandard objects is that there are individual creators and craftspeople and groups to be celebrated (or castigated) - rather than anonymous systems where interchangeable human crank out interchangeable parts. The questions of attribution and authorship in hacker production are much more complex that we can cover here, but through them runs another distinct form of hacker action: the performance of virtuosity - in some cases for functional ends, as with a brilliant fix, but often as an end in itself, to be appreciated by other hackers who can understand what you've accomplished. This goes back to the tricks pulled off by phone phreaks, like routing calls from relay to relay, across the phone network, around the world. This served no functional purpose - in the sense of getting free long distance, for instance - but was instead a trick that demonstrated extraordinary technical competence: the legendary phone phreak Captain Crunch would set it up to route a call from one handset around the world to another handset in the same room, putting his voice on a planet-size time delay (Rosenbaum 1977).

This is the purest expression of the making of nonstandard things, the least inflected by the quotidian demands of industries, managers, and end-users, and a crucial action in the hacker lexicon: the performance of technical virtuosity more or less for its own sake. Such accomplishments don't just garner prestige; they reflect a larger community that can admire the extreme difficulty (often self-imposed), and deep insight into the technologies and tools that they reflect. Hackers dub themselves and one another as ‘wizards,’ ‘Jedi,’ and ‘ninjas,’ all groups whose membership is limited by very demanding thresholds of dedication, training, and skill. (Much has been made in geek discussion of how the Jedi from the *Star Wars* universe mark their progress in training by making their own lightsabers from scratch - a cultural fantasy of recursive tooling if ever there was one [Brunton 2013: 18].) Understanding the implications of this virtuosity is likewise limited to those in the know.

In fact, one of the most extreme expressions of the hacker performance of virtuosity produces the least impressive result: a program that outputs the string ‘Hello, world’ or various other traditional phrases, like ‘Just another Perl hacker,’. (The comma is traditional.) To write a command that will return this result is the most basic, introductory act of many computer language lessons. The goal of hacker virtuosity is to produce it using the most mind-smashingly opaque, complex, counterintuitive means, which other hackers will delight in picking apart and trying to understand, for events like the International Obfuscated C Code Contest. There are numerous contests for different languages, as well as ‘esoteric’ languages designed to be challenges in themselves, like Brainfuck, Grass - whose code, built entirely from ‘W,’ ‘w,’ and ‘v,’ looks like grass - and Malbolge, named for the eighth circle of Hell. (The related phenomenon of the ‘demoscene’ seeks to produce visual and sonic performances out of deliberately constrained programming tools, sometimes pulling off astonishingly rich displays out of only a few lines of exquisitely composed code; part of the pleasure of demoscene events is understanding the ingenuity with which the effect was produced.) As Nick Monfort put it, obfuscated code ‘darkens the usually “clear box” of source code into something that is difficult to trace through and puzzle out, but by doing this, it makes code more enticing, inviting the attention and close reading of programmers’ (Montfort 2009: 198; see also Mateas & Montfort, 2005). It speaks to the aesthetic and craft pleasures of hacking expressed as its own set of actions and productions.

Policing and Defining Hackerdom

Of course, part of the activity around those demoscene competitions is to separate those who really understand and appreciate the technological feats from those who don't, or who fail to appreciate them on the appropriate level - a process of policing and defining the ‘elite’ and the varieties of non-elite ‘lamers’. The seventh of the eight actions and the most self-referential will also be the briefest to describe, because it is the least technically particular: a recurring activity in the hacking community is discussion and debate over the meaning of ‘hacker’ itself - what are the criteria, who really gets to be one, what you should be doing to qualify, and who has been excluded.

There are many debates and internal conversations to this question that lie beyond the brief scope of our work here. A few brief examples will suffice. Eric S. Raymond, a notable open source software developer (and author of the landmark open source development manifesto, *The Cathedral and the Bazaar*) maintains a lengthy document for those who have written to him seeking to develop ‘wizardly hacker’ expertise: ‘How to Become a Hacker’ (Raymond nd). ‘Hackers build things,’ he writes, and provides a blend of mindsets (‘No problem should ever have to be solved twice’), particular skills and tools (‘Get one of the open-source Unixes and learn to use and run it’), and social capital (‘Help test and debug open-source software’). This document, periodically expanded and refined since 1996, perfectly exemplifies a particular hacker type - outside the formal recommendations, Raymond advocates for other practices to find one's way more easily in the hacker scene, like reading science fiction and cultivating a fondness for puns. Raymond subsequently disgraced himself with increasingly bizarre, conspiratorial, racist and misogynist personal positions which in retrospect colored this document. It implied another requirement to be a hacker in some circles, unstated but seemingly evident: to be an abrasive but thin-skinned, competent but deeply insecure, white guy who likes arguing on the Internet, and assumes other hackers must be more or less like him. The ‘flamewar’ culture of name-calling, abuse, and insults, and the casual sexism and racism, which result from this culture have killed many an open-source project, or reduced it to only the most high-blood-pressure personalities - who are not necessarily the best developers.

In contrast to this, a wave of new groups, events (including hackathons and workshops) and publications are explicitly trying to renegotiate who gets to be called a hacker and what a hacker is assumed to be. The !!Con, for example, tries to foster not only a more diverse array of hackers, but also a different culture of hacker engagement - one that is less language and platform focused, and instead emphasizes ‘the joy, excitement, and surprise of programming,’ an ethos in many ways much closer to the historical roots of hacking as a vocation than the intensely monetized, product-first, overworked-at-a-big-company approach that characterizes the hacker in contexts like Silicon Valley today. Sumana Harihareswara describes !!Con's breadth of technical and engineering talks as an assumption: ‘every attendee has the capability of being curious about everything’ (Harihareswara 2016). Or, as Raymond phrased the first requirement for being a hacker almost twenty years earlier, you must believe that ‘the world is full of fascinating problems waiting to be solved’ (Raymond nd).

Social Engineering

This final entry in the hacker vocabulary of actions echoes the earliest days of hacking, but has a new contemporary resonance. With it, we close the loop and conclude this chapter: from the earliest pranks and collegiate ‘hacks’ to the discussion around ‘hacking the election’ in the United States and other countries over the last two years - and future mutations of the term ‘hacking’ and the actions that constitute it.

The earliest ‘hacks’ identified with that verb were often in the service of sophisticated, complex pranks: the technological ingenuity and access needed to surreptitiously put a car on top of the Great Dome of MIT, or inflate a weather balloon at the fifty-yard-line in the middle of a football game. Pulling off these pranks often involved not only material engineering, but ‘social engineering’: a security-focused version of confidence trickery (Peterson 2003). Larry Wall, the legendary developer of the Perl programming language, said great programmers possess laziness, impatience, and hubris: they are wildly ambitious, but have no patience for what Eric Raymond's guide to hacking terms ‘boredom and drudgery.’ This means that - along with automating away repetitive tasks - hackers are always in search of the optimal, efficient shortcut around the seemingly intractable problem. If getting access to a closed building or a phone network needs a code, why not fast-talk someone with the password into giving it to you in a few minutes instead of various time-consuming and perhaps unsuccessful technical approaches? One under-recognized component of the hacker toolkit evolved from this: the accumulated lore of social engineering, from interpersonal activities like cold-calling and looking over someone’s shoulder as they type in a password (‘shoulder surfing’), to going through a company's trash (‘trashing’) in search of useful intrusion information, to now-commonplace phishing emails that fool the recipient into logging into an account and thereby giving up a password.

It was a phishing attack that originally gained access to the email account belonging to John Podesta, the chairman of Hillary Clinton's 2016 presidential campaign. The emails, subsequently published on WikiLeaks, played a part (their exact consequences still debated) in the failure of the Clinton campaign and the election of Donald Trump, along with the likewise uncertain effect of social network manipulation through bots and the circulation of false stories and images. This has been widely referred to in the American media as ‘hacking’ the election, though, arguably, the only part that resembles the history of hacking to this point was the acquisition and leak of the Podesta emails. But it raises a useful question for us: the popular meaning of ‘hacking’ continues to evolve. Social engineering has always been a part of hacking, back to the phone phreaks getting access to Ma Bell's network and proto-hackers finagling parts and computing system time from their universities and institutions. Etymologically, many of the earliest projects of ‘hackers’ getting and giving access were in the service of stunts and pranks. Can this be plausibly explained as a single, coherent thread in the history of hacking that began to blur into the space of trolling, doxxing, Rita Raley's ‘tactical media,’ and prankish weirdness that has become a vector for political disruption - ‘social engineering’ on a much larger scale? (Raley 2009; see also Phillips 2015, Coleman 2014) At what point does the expansion of the concept of hacking become meaningless?

I hope this chapter has demonstrated that the answer to these contemporary questions lies not in an abstract definition or redefinition, but in studying the particulars of actions that the people involved think of as ‘hacking’. Will a distinct, new category of action be added to this collection? Will one of these forms of activity drop away as a salient part of the spectrum of hacking? Getting and giving access; tinkering and reverse engineering; recursive tooling; commoning; making nonstandard things; performing virtuosity; policing and defining hackerdom; social engineering: the eight forms of action described here will change in their subjects and implications, but their continuity throughout the history and transformations of hacking so far argues for their persistence in the future, as components in a technologically specific way of living and working.

References

Benkler, Yochai. 2006. *The Wealth of Networks: How Social Production Transforms Markets and Freedom.* Yale University Press

Brown, Jeremy. 2000. ‘Yak-Shaving.’ Available at: http://www.mit.edu/~xela/yakshaving.html

Brunton, Finn. 2013. *Spam: A Shadow History of the Internet.* MIT Press

Cohen, Julie. 2012. *Configuring the Networked Self: Law, Code, and the Play of Everyday Practice.* Yale University Press

Coleman, Gabriella. 2013. *Coding Freedom: The Ethics and Aesthetics of Hacking.* Princeton University Press

Coleman, Gabriella. 2014. *Hacker, Hoaxer, Whistleblower, Spy: The Many Faces of Anonymous.* Verso

Driscoll, Kevin. 2016. 'Social Media's Dial-up Roots.' *IEEE Spectrum* 53, no. 11 (November 2016): 54–60

Folz, Rodney. 2015. 'Selling Out and the Death of Hacker Culture.' Available at: https://medium.com/@folz/selling-out-and-the-death-of-hacker-culture-fec1f101b138

Harihareswara, Sumana. 2016. 'Towards a !!Con Aesthetic.' *The Recompiler.* Available at https://recompilermag.com/issues/extras/toward-a-bangbangcon-aesthetic/

Huang, Bunnie. 2013. *Hacking the Xbox: An Introduction to Reverse Engineering.* No Starch

Johns, Adrian. 2009. *Piracy: The Intellectual Property Wars from Gutenberg to Gates.* University of Chicago Press

Lapsley, Phil. 2013. *Exploding the Phone: The Untold Story of the Teenagers and Outlaws Who Hacked Ma Bell.* Grove Press

Levy, Steven. 1984. *Hackers: Heroes of the Computer Revolution*.
Anchor/Doubleday

Lions, John. 1977. *Lions' Commentary on Unix 6th Edition with Source Code.* Peer to Peer Communications

Liu, Alan. 2004. *The Laws of Cool: Knowledge Work and the Culture of Information.* University of Chicago Press

Kelty, Chris. 2008. *Two Bits: The Cultural Significance of Free Software.* Duke University Press

Knuth, Donald. 1986. *Computers & Typesetting, Volume A: The TeXbook.* Addison-Wesley

Mateas, Michael and Nick Montfort. 2005 'A Box, Darkly: Obfuscation, Weird Languages, and Code Aesthetics.' In *Proceedings of the 6th Digital Arts and Culture Conference,* IT University of Copenhagen, 1-3 Dec 2005

Montfort, Nick. 2009. 'Obfuscated Code.' In *Software Studies: A Lexicon* (Matthew Fuller, ed.). MIT Press

Nakamura, Lisa. 2006. 'Cybertyping and the Work of Race in the Age of Digital Reproduction.' In *New Media, Old Media: A History and Theory Reader* (Chun and Keenan, eds.). Routledge

Neff, Gina. 2015. *Venture Labor: Work and the Burden of Risk in Innovative Industries.* MIT Press

Peterson, T.F. 2003. *Nightwork: A History of Hacks and Pranks at MIT.* MIT Press

Phillips, Whitney. 2015. *This Is Why We Can't Have Nice Things: Mapping the Relationship between Online Trolling and Mainstream Culture.* MIT Press

Raley, Rita. 2009. *Tactical Media.* University of Minnesota Press

Raymond, Eric S. No date (ongoing). 'How To Become A Hacker.' Available at: http://www.catb.org/~esr/faqs/hacker-howto.html

Rosenbaum, Ron. 'Secrets of the Little Blue Box.' *Esquire Magazine,* October 1971. Available at: http://classic.esquire.com/secrets-of-the-blue-box/

Scholz, Trebor (ed.). 2012. *Digital Labor: The Internet as Playground and Factory.* Routledge

Silvanovich, Natalie. No date. 'Many Tamagotchis Were Harmed in Making This Presentation.' Available at: http://natashenka.ca

Stallman, Richard. 1985. ‘The GNU Manifesto.’ Available at: https://www.gnu.org/gnu/manifesto.en.html

Stephenson, Neal. 1999. *In the Beginning Was the Command Line.* Avon Books

Swartz, Aaron. 2008. ‘Guerilla Open Access Manifesto.’ Available at: https://archive.org/stream/GuerillaOpenAccessManifesto/Goamjuly2008_djvu.txt

Thomas, Douglas. 2006. *Hacker Culture.* University of Minnesota

Turner, Fred. 2003. *From Counterculture to Cyberculture: Stewart Brand, the Whole Earth Network, and the Rise of Digital Utopianism.* University of Chicago Press

Wark, McKenzie. 2004. *A Hacker Manifesto.* Harvard University Press

Wark, McKenzie. 2017. *General Intellects: Twenty-One Thinkers for the Twenty-First Century.* Verso

Zuckerberg, Mark. 2012. ‘Mark Zuckerberg's Letter to Investors: “The Hacker Way.”’ *Wired*. [online] Available at: https://www.wired.com/2012/02/zuck-letter/