Hacking the body: code, performance and corporeality

DOUGLAS THOMAS

University of Southern California, USA

Abstract

This article argues for a politics of code grounded in the performance of hacking as an activity that is able to deconstruct the longstanding binary of code and performance by centering on the relationship of each to the body. Through a reading of hacker responses to the state’s restriction on the export of cryptography, the article argues that interventions by hackers reveal to us the point at which the body becomes, literally, the limit of code, marking it as irreducibly transgressive, while rendering code impotent. It is this performance, it contends, that holds the most powerful possibility for hacktivism and resistance.

Key words

code • corporeality • encryption • hackers • performance

INTRODUCTION

Narratives of technology are routinely given to binary distinctions. Consider, for example, the impulse to construct narratives of technological innovation as either utopian or dystopian. Every celebration of the emergence of a new technology has been accompanied by a darker vision, usually spelling the end of civilization as we know it. Television, for example, was heralded as a technological breakthrough while at the same time, Congress convened in the 1950s to hold hearings to determine whether television caused juvenile delinquency (see Sturken and Thomas,
Later, throughout the 1990s, the rise and growth of the internet generated promises of world peace, the dissolution of international boundaries and the improvement of race relations, while at the same time triggering fears about user addiction and depression – and, most of all, about the figure of the (typically male, teenage) hacker.\(^1\)

Theoretical inquiry also reinforces these concerns: from Martin Heidegger’s 1954 essay, ‘The Question Concerning Technology’, which considers technology’s ‘greatest danger’ in science as well as its ‘saving power’ in art, to the debates over technological determinism and constructivism, scholarly considerations of technology seem to be continually reducible to pairs of terms and to the negotiation between them (Heidegger, 1993[1954]). One of the reasons for this has to do with the fact that technology always performs a function of mediation, generally between two poles and our discourse, even our theory, tends to recapitulate that tension.

Hence, throughout this inquiry we will position – or perhaps reposition – the idea of technology between two terms that often function in opposition to one another: code and performance. In doing so, this author wishes to suspend the meaning of technology between the two, to look at technology not as a product but as a process that generates meaning through the negotiation of these two conceptual and material elements. Of course, it in no way would it be fruitful to attenuate the force of this opposition. But there is some value in examining what the tension between the two entails – in examining the poles not as mutually exclusive options, but rather as anchoring points, or nodal points if you will, for the broader discussion of the meaning of technology itself.

Proceeding in this manner, this article will demonstrate that technology embodies aspects of both code and performance: we can gain a better of understanding of how technology functions by interrogating the ways in which technology both utilizes and problematizes each of these dynamic factors. In particular, it will examine the ways in which the development and uses of code, particularly in the cases of open source development and encryption, can be understood better as dynamics of code and performance within the context of hacking subcultures of development and subcultural resistance.

Most directly, much of the analysis here is located in the relationship between bodies and codes. While it is easy to understand the ways in which performance can be understood corporeally, code is a much different matter. As a system of abstraction, code tends to be removed from the body. Within the various contexts of code and performance, this article is an effort to rethink the boundaries between the natural and artificial, between the real and virtual and, ultimately, between code and performance. The body, it will be argued, is the site where the logics of such binary division can no
longer sustain themselves as separate and the point at which we can begin to understand the meaning of hacking as resistance.

**CODE**

Defining one’s terms is always a dangerous business, so let us begin by saying that this is done provisionally and somewhat advisedly. That said, the definitions proposed here, with respect to the facts and phenomenon of code and performance, are themselves designed to be useful rather than definitive. Let us begin with the crux of the matter at hand, with the lifeblood of technology and hacking in the late second half of the 20th century: code. By code, we mean to refer to a system of regulation, a regime, which is both structured and structuring, which is to say that at its base, the primary function of code is normative. This is a vital aspect of code, one upon which the subsequent arguments in this article will pivot and one that they will also push to its limits.

This article takes as a grounding assumption the idea that, as Larry Lessig asserts, ‘code regulates’ and that in the case of the internet, the medium that occupies a great deal of his attention, we must understand ‘how the software and hardware that make cyberspace what it is regulate cyberspace as it is’ (Lessig, 1999: 6, emphasis added). In other words, the machinery and code that constitute the substance of cyberspace condition not only what it means, but also how it means. Accordingly, even the most basic functions of computer code, loops and branches, variable assignment and calculation (functions shared by all computer programming languages), describe the basic parameters of what is both possible as well as what is acceptable.

Lessig also argues that code, specifically computer code, shares its normative function with law. As law, in the general sense of the word, code functions abstractly, always moving from the particular to the general in an effort to define the boundaries of normalcy. The premise of law itself is grounded in the regulation of social function, abstracted from particular instances of behavior or events which are deemed unacceptable or injurious to the community.

Moreover, as a general concept, code goes well beyond the confines of Perl, C++ or the court system. Code is endemic to any system of writing. In broad terms, code is grounded in its own repetition, its iterability. Code only functions when it appears to repeat without difference, even as the situations, contexts and events to which it is applied vary in their specificity. Writing is transformed from the general structure of gestural codes and iteration into code at the moment when pen touches paper and ink marks the page – in the instance when ‘we simply do not know what our writing does’ (Kittler, 1995: 1). Writing becomes code when it produces a surface effect that appears so completely unrelated to the underlying processes of creation that those processes disappear altogether from the finished product.
But, code becomes wedded to technology at the moment that it begins to effect writing by making it permanent in some way, in those instances when, as Friedrich Kittler has noted, ‘modern media technologies in general have been explicitly contrived in order to evade all perception’ (1995: 1). In such circumstances, code is the means by which writing becomes alien to us, the means by which writing is transformed and ultimately, in this day and age, reduced to a measure of differences in voltage registered by silicon chips. It is a process that is repeated throughout the history of technology in what David Bolter and Richard Grusin (1999) have called the dual logic of remediation, the process by which technological sophistication has at base the goal of rendering itself invisible, even as its presence in the process of mediating communication increases. Think, for example, of the annoyance that occurs when we are reminded of the fact that we are speaking on the telephone as a result of line noise or more disturbingly about a ‘reality’ TV program such as Survivor: the excesses of technology which make the show possible stand in stark contrast to the abject denial of modern technology to the participants of the show, 16 castaways forced to live off the land.

Thus, code has a second, equally important quality: the ability to make itself disappear. Now, this insight should come as little surprise to us. The most effective systems of regulation and social control are presented to us always as pre-existing, natural or essential. Thus, we might supplement Lessig’s insight with the notion that the disappearance of code is an essential part of its functioning. That said, let us foreground this notion of code as an abstract system that resists any attempt at particularization and that, at every opportunity, attempts to erase the means and manner of its functioning, leaving only the effect of code (a kind of surface effect) which makes the underlying processes of appearance indecipherable.

Consider the relationship between source code and software in most retail computer products. The software bears no trace of the source code that produced it. Nowhere in Microsoft Word or Windows XP can you find the source code, the programming that gave rise to the software. The law is so deeply interconnected with this phenomenon that when such code does emerge the law (and law enforcement) is immediately invoked. So, for example, when portions of Microsoft’s source code for Windows 2000 and NT were posted anonymously on the internet in February of 2004, Microsoft’s response illustrated these assertions convincingly:

Microsoft continues to work closely with the U.S. Federal Bureau of Investigation and other law enforcement authorities on this matter. Microsoft source code is both copyrighted and protected as a trade secret. As such, it is illegal to post it, make it available to others, download it or use it. Microsoft will take all appropriate legal actions to protect its intellectual property. These actions include communicating both directly and indirectly with those who possess or seek to possess, post, download or share the illegally disclosed source
The notions of secrecy emerge as almost the very definition of the code. It is not only illegal to post or distribute it; it is illegal to possess it and use it. The juxtaposition is striking. In an industry where the goal is to have each and every computer running your software, the very success of that enterprise is dependent upon making the source, literally the code itself, invisible, inaccessible and, ultimately, bringing the full force of law to bear on those who seek to violate that principle. 4

PERFORMANCE
Performance, however, is different. It is the opposite of code. In Peggy Phelan’s discussion of the ontology of performance, she writes:

Performance’s only life is in the present. Performance cannot be saved, recorded, documented or otherwise participate in the circulation of representations of representations: once it does so, it becomes something other than performance. (1992: 146)

Performance, at least in the sense that this article wants to use the word, deals with that which is essentially unrepeatable. Of course, both code and performance repeat (or in the strictest sense of the word, fail to), but what differentiates them is their direction. The goal of code is infinite repetition without difference, while the goal of performance is the production of an irreducible, unrepeatable event – a singularity. Of course, one without a tinge of the other would lapse into utter unintelligibility. In order to be in any way useful, code must have a certain ‘eventfulness’ about it; while in order to be understood, performance must engage in repetition even if that repetition is, as Judith Butler suggests, ‘parodic’ or ‘subversive’ (1989: 139).

Performance is literally a re-presenting or even re-presencing. It generates its authenticity not from repetition per se, but rather as repetition with difference. It is a calling of attention to itself which questions, subverts or even just calls attention to that which preceded it. It is everything that Plato abhorred about poetry, art and music. Performance is the copy that strives not to be the same, but to be different. A simulacrum, an image without resemblance. 5 This is not only what gives it force, but also ironically what gives it authenticity. Accordingly, while the value of code rests in its normativity, the value of performance is found in transgression.

In groups such as the Homebrew Computer Club, we find the origins of the ‘open source’ software movement, which is grounded in the belief that all software should be distributed along with the code that produced it, making replication and, more importantly, alteration possible. But it is more than a guiding principle: for many programmers it rises to the level of an ethic, even a moral imperative.
It is also the site of primary antagonism between open source culture and Microsoft. In 1976, Micro-Soft was a small start-up in Albuquerque, New Mexico, selling the computer language BASIC (a language released into the public domain by a pair of Dartmouth programmers) to a handful of enthusiasts who subsequently reproduced and distributed the language to club members. Alarmed that users were trading code, rather than paying for it, a young William Henry Gates III published ‘An Open Letter to Hobbyists’ in which he re-characterized the culture in dramatic terms: ‘As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share ... the thing you do is theft.’ That antagonism still animates the open source/Microsoft split that is very much alive today, marking open source as transgressive in terms of capital and, as importantly, in terms of the law. Open source is marked as a system of unregulated distribution and redistribution that undermines the normative function of code and, in so doing, demonstrates the ways in which law and capital are complicit with each other.

By way of contrast, open source is performative and transgressive. This performative, transgressive dimension manifests itself in what this article calls the ‘culture of code’. It is a space of repetition with difference and a space of questioning. It is the space in which the meanings, especially the cultural meanings of technology, are contested. It is the space that provides for acts of resistance within the very fabric of technology itself. These spaces or cultures have a very basic function in subverting the normative functioning of code. Accordingly, the traditional ‘hacker’ often has little patience for authority, prefers decentralization to centralized systems and uses words such as ‘freedom’, ‘art’, ‘beauty’ and ‘elegance’ to describe the act of programming. In turn, across the board, while the product of code, software, reflects a highly regulated structure, the production of code is better described as performative. Often, we find that while programmers produce code designed to be reproduced endlessly without difference, their own styles tend to be highly idiosyncratic, individualistic and particular. This is especially true of open source programmers such as Larry Wall, Richard Stallman and Eric Raymond (2001), self-described ‘evangelists’ who describe the software that they produce as ‘subversive’.

The more performative the act of coding (as is the case with open source software), the more likely it is that the process and the code itself will be rendered visible. Performance, like code, also disappears but in doing so it leaves a mark, not of its product (e.g. software) but of its process. Performance literally disappears into memory, but in so doing remains vivid and present. As such, a goal of performance is to disappear (as it must as an event bound in time and space) but in the process it renders itself opaque, lasting and permanent. Nowhere can this be seen more clearly than in the production of open source software. As the name suggests, open source
software requires that not only its distribution, but also all subsequent distributions, include the source code that generates the software. The resulting product is considered ‘free software’, not in terms of its price (one can charge for open source, although many do not) but in terms of its character. The user is ‘free’ to change, alter, develop and re-use aspects of the code as they see fit. It is a transmission not only of the code, but also of the culture that generated the code in the first place. It is this aspect, the quality of ‘freedom’, which cannot be documented in any way, but rather can be performed only through the process of distribution and redistribution. It is this process that disrupts the smooth flow of production from code to product by re-presenting (presencing) the code at every turn. It is this performance which is performing the cultural, transgressive and subversive aspects of the culture of code that is embodied within it.

PERFORMANCE AS THE DISRUPTION OF CODE

To say that the open source movement has disrupted traditional notions of software development and distribution would be an understatement. Not surprisingly, one of the venues in which the battle between code and performance is being fought is within the legal system itself. Brought to a head by issues of digital rights management, open source advocates’ demands for ‘free’ software and code has so fundamentally disrupted the software industry that Microsoft created a strategy designed to combat open source. In November 1998, two documents were leaked from Microsoft headquarters, the so-called ‘Halloween Documents’ (Open Source Initiative, 2005). Although many issues were identified as ‘threats’ to Microsoft (including the fact that Open Source Software (OSS) tends to be superior in quality), the primary issue was not code, but culture:

The ability of the OSS process to collect and harness the collective IQ of thousands of individuals across the internet is simply amazing. More importantly, OSS evangelization scales with the size of the internet much faster than our own evangelization efforts appear to scale. (Microsoft, quoted in Open Source Initiative, 2005)

Community, culture and evangelism are the motors driving both Microsoft and open source.

The Halloween documents also illustrate the failure of Microsoft’s standard marketing technique, ‘FUD’: Fear, Uncertainty, Doubt. This is a marketing technique deployed to scare consumers by questioning the permanence, reliability and compatibility of competitive products. They even included claims that Microsoft had implanted harmless ‘error’ or warning messages in their operating system, which would alert users to the fact that certain programs were not ‘Microsoft tested’ for compatibility. Nevertheless, open source was, by Microsoft’s own testing, immune to FUD tactics and in
order to combat the spread of open source software, Microsoft acknowledged the necessity of retreating back to code and law. The only effective way to combat Linux was through patent and copyright. There were certain ideas and concepts that Microsoft believed they could make legal claim to own and thereby invoke proprietary ownership of certain software.

CODE, PERFORMANCE AND THE BODY
Before moving to an extended example which addresses these phenomena more directly, let us consider a third aspect that technology affects: corporeality. In its most general sense, code regulates and defines not by attaching itself to particular bodies, but by abstracting notions that appear to apply to all bodies. There is and never can be a particular ‘coded’ body. To produce one would be to raise an objection already to the very abstracted, ideally universal conception of code. Therefore, the ‘coded body’ cannot exist without disrupting the very notion of code itself. In other words, to produce such a body, would mark de facto the singular coded body as itself different – as a reiteration and realization with at least one unique sign of substantive rearticulation. Therefore, the incarnate, coded body can exist only as a condition of its own impossibility. If it exists at all, it must exist in an abstract state, one which can never be realized or made flesh. Doing so, rendering the abstract material, would be to expose material or corporeal absence as the essential motor which drives the process of cultural, coded production.

Performance, on the other hand, is the body. It needs bodies to make itself real. It is the material and the flesh, the substance of being. It is the exception rendered visible which challenges not only the abstract category of code against which it is compared, but also the entire regime of comparison. In removing the essence, it removes the very possibility of appearance.

In what follows, this article aims to make concrete the theses which have just been advanced. In so doing, it explores hacking as a context in which the tensions between code and performance provide a clear understanding of how technology both functions and fails to function as a system of cultural mediation between normativity and transgression, between repetition and the unrepeatable and between abstract incorporation and corporeal materiality. The case is made for a performative model of technology which understands and exploits the means of transgression that technology affords to us through the thematic of performance. The article then examines the various ways in which state control of encryption and hacker resistance to that control can help us to understand better the relationships among code, performance, the body and technology.
THE INTERNATIONAL TARIFF ON ARMS RESTRICTIONS: BODIES, CODES AND THE STATE

This illustration deals with encryption. Encryption, as the process of encoding and enciphering text, appears on its surface as an act of pure code: it appears to have nothing to do with the body. Recent algorithms, in particular the RSA algorithm (named after its inventors, Ron Rivest, Adi Shamir and Len Adleman) have reduced encryption almost entirely to the language of mathematics. As a system of public key encryption, RSA provides a mathematically secure system that creates problems which are unsolvable without a key. Such public key encryption, which has become the standard for most internet and government work, relies on mathematics no more complicated than basic prime number theory, algebra and modulus arithmetic. The principle of modern encryption is simple: one finds a mathematical problem that is simple to pose and impossible to solve without a key. Modern encryption utilizes even more sophisticated ‘one-way’ functions, allowing a code to have two different keys, one that locks and one that unlocks, neither of which can be derived from the other. The triumph of modern encryption is the removal of the body, rendering encryption an act of pure code.

Early efforts at encryption required a secret to be shared, passed from one body to another. It was predicated on both parties knowing the same secret, but our latest efforts remove even that problem. No one needs to know your code, your secret or your password, not even the computer. The history of encryption, however, tells a different story. In telling it, we begin not with mathematics or computers, or with the codebreakers of the Second World War or, even as some have suggested, with Julius Caesar scrambling messages to troop commanders in Rome.

The history of encryption begins with the last book of Sophocles’ Oedipus trilogy, Oedipus at Colonus, where three discrete elements merge within the first narrative of encryption. Oedipus, fearing that his grave would be defiled, extracts a promise from the King of Athens to keep the place of his burial (literally, his crypt) secret. In exchange, Oedipus offers Athens perpetual safety from its enemies so long as the secret is maintained (Sophocles, 2004: lines 1732–40). These three components comprise the base elements of the technology of encryption: a code, the body and the protection of the state.

It also marks the body as the disruptive element which continually threatens not only the normative structure of code, but the very security of the state itself, especially when that body, like Oedipus’s, is a transgressive one, marked by its violation of code and its transgression of the law.

Of course, the more things stay the same, the more they invite performative subversion. In some 20th-century contexts, for example, in contrast to the state, which utilizes encryption to keep secrets and guarantee
security, a new generation of self-proclaimed ‘cyberpunks’ has found encryption to be an invaluable tool for subverting a chief aim of government: surveillance. For cyberpunks, the killer application of encryption was not found in secrecy but rather in privacy, in the possibility of selectively revealing yourself to the world. Anonymity and privacy became the chief tools in the battle against an increasingly mechanized and panoptical security state. Once again, code had found a way to become transgressive. With the development of RSA and a free, open source application of it called PGP (Pretty Good Privacy), encryption was seen as a viable means for protecting the rights and freedoms for a digital generation.

But the state is never easily subverted and, as cyberpunks practised their performative acts of resistance, the government retaliated with legislation restricting encryption. In 1991, Congress attempted to pass legislation mandating that all forms of encryption contain a ‘back door’ allowing governmental access to coded civilian documents. It failed to pass into law, but was the impetus for Phil Zimmerman to develop and write PGP for public distribution. The government responded, in turn, by claiming patent infringement and later by classifying the RSA algorithm under the International Tariff on Arms Restrictions (ITAR). In so doing, for the first time, three lines of computer code become officially classified as a munition. Exporting it without official sanction from the US State Department was no small crime. If convicted of exporting RSA, PGP or any other encryption program, one could face a $1 million fine and 10 years in jail for each export violation. The extremity of US policy was comparable to only five other nations: China, France, Iraq, Iran and Russia.

In an effort to fuse law and code, the US State Department’s efforts created a climate in which the export and distribution of code (and in some cases products) was heavily regulated, based strictly on a system of classification and protection. In the new classification, RSA deployed the tropes of ‘national security’ as well as a number of other social and cultural threats, including organized crime, drug trafficking, terrorism and pedophilia. In response, hackers began a movement to display the code wherever possible. The event which actually led to the compression of the RSA algorithm into three lines of inscrutable code, was later refined by Adam Back into the following:

```
print pack'C*\',split/\D+/,'echo '16iH*\vU@{$/
 = $z;[(pop,pop,unpack'H*', < >
,))\ExMsKsN0|N*\1K[d2%Sa2/d0 < X+d*IMLa^*|N%0|dsXx++|MLN/
```

With these two lines of Perl code, the guts of the RSA algorithm became at once completely obfuscated (even the most wily Perl hacker would have
difficulty deciphering Back’s code), while at the same time rendering it on a different level completely accessible. Reduced to two lines of code, it could be reproduced and disseminated not as code per se, but rather as a statement about the reproduction and dissemination of code. In other words, Back’s code transformed the RSA algorithm from code into performance. In its more verbose version, three lines long, the code first appeared on a T-Shirt (see Figure 1).

The significance of this transformation is that it both rendered the code visible (and hence performative and disruptive) while at the same time completely neutralizing it. The T-shirt could not encrypt anything, its only value was in the visibility of the code, not in the function of the code itself.

The code emerged later on a webpage, ITAR Civil Disobedience (alternatively titled the International Arms Trafficker Training Page), which billed itself as an act of civil disobedience and in so doing, allowed anyone to export banned crypto anonymously. Additionally, there were boxes to check for visitors to add their name to a list of ‘known arms traffickers’ or to send a letter to the president declaring their act of civil disobedience (see Figure 2).
Ultimately, however, the performative acts of resistance would be taken to what can be argued is their logical extreme: to the fusion of bodies and code, which marks the body as irreducibly transgressive. In this same vein, the cyberpunk movement itself clearly recognized and powerfully deployed corporeality as a means to resist through the medium of tattooing. One of the first cyberpunks to acquire an RSA tattoo did so as an effort to raise public awareness about issues of encryption. Thus, in this instance and in others like it, the literal embodiment of code announced its own performance not as an act of encryption, but as an act of resistance. Therefore it was the ultimate statement of a ‘culture of code’, allowing the body to speak the language of code, rending the code both transgressive and impotent (see Figure 3). It also had the effect of rendering a number of

![Figure 3 RSA Encryption Code in tattoo form, photograph by Richard White](image-url)
publications complicit in arms trafficking, as pictures of the image appeared in Japanese, French and UK publications as well as the New York Times.

In doing so, the performative act of creating a coded body, defies precisely what the law allows. In other words, by forfeiting function and making the code appear where it cannot, these marked bodies became transgressive and disruptive in a way that law simply cannot reconcile.

A second celebrated RSA tattoo, that of the cyberpunk known as ‘CancerOmega’, is even more explicitly subversive, bearing the warning ‘This Man is Classified as a Munition’ (see Figure 4).

By marking the source and origin of the law’s code (the classification of code as a munition), CancerOmega has exposed something fundamentally wrong with the notion of law and code. This is code pushed to its limit. And if we are to conclude that the statement ‘This man is classified as a munition’ is at once technically correct as a matter of code, law and regulation, we are forced at the same moment to confront the absurdity of such a statement through performance. The law, when confronted with the coded body, loses its force as law: it ceases to function as an abstract principle of normative regulation and, instead, becomes a parody of itself.

In addition, speculative newsgroup and listserv discussions appeared regarding other ways that such performances might further subvert the law. When foreign students, particularly those studying computer science, learned of the export regulation attached to the RSA algorithm, students considered tattooing their bodies with the code to prevent them from being deported when their student visas expired. In a post to alt.callahans, a poster suggested the following:

[A]n illegal immigrant might get an RSA-in-Perl tattoo in order to prevent deportation, since it’s illegal to export and US law won’t allow the government to subject people to involuntary surgery. (Steve Brinich, alt.callahans, 28 August 1998)

Similar discussions proliferated on lists ranging from a cyberpunk newsgroup to rec.arts.bodyart to any number of Perl newsgroups and
mailing lists. One exchange (on comp.org.eff.talk) highlighted still more performative acts. One poster’s suggestion for an RSA tattoo included additional subversions:

You know, my wife’s a tattoo artist and I’ve been seriously toying with the idea of getting the RSA algorithm tattooed somewhere on my body. Might be interesting to call a press conference and then try to board an international flight. What are they going to do – tell me I can no longer leave the country, as I’m now classified as a ‘munition’ and thus prohibited from export? (Richard White, comp.org.eff.talk, 3 July 1995)

Such a suggestion drew the following response:

Might I suggest the best place for this would be on your posterior [sic]? That way, when you had to ‘present evidence’, you would get to moon the court and law enforcement officials [sic]!! What better statement could one make about the whole situation?!!? (Farrell McGovern, comp.org.eff.talk, 3 July 1995)

Here, again we return to the argument that the performative act of transforming codes into bodies (and also bodies into code) potentially disrupts the entire system of regulation and normativity that allows code itself to function. As has been demonstrated, such disruption is generated by making the code appear and that disruption is, in the strictest sense of the word, a performative act.

CONCLUSIONS: CODE AND/AS PERFORMANCE
This article has endeavored to chart an understanding of technology that is opened up through performance. That opening is not merely one that mediates technology, but is the space in which resistance and transgression both become possible. As this author hopes to have argued convincingly, performance is no more a-technological than technology is non-performative. As a consequence, the politics of technology have a great deal at stake in their performative dimension insofar as the relationship between performance and technology can be best described perhaps as one of intervention. Importantly, while that intervention does not necessarily require one to write code (although it in no way excludes that capability either), it does require one to be able to negotiate code, to translate code and ultimately be able to redeploy it in ways which challenge its normative functioning.

What this literal hacking of the body demonstrates is perhaps the most deeply held and most important element of the hacker ethos: that resistance is always about finding alternative and profound means to transgress. What characterizes hacktivism as a unique form of resistance has more to do with process than it does with effects or outcomes. Hacktivism is, at base, a
deconstructive endeavor, which seeks not to undermine any single idea or expression, but instead which seeks to undermine the very foundations and logics upon which such expressions are built.

Acknowledgement
The author would like to thank Ann Chisholm for her comments on earlier drafts of this essay.

Notes
1 For a discussion of the primarily male, teenage, suburban image of the hacker as ‘boy culture’ see Thomas (2002) especially pp. xii–xvii, 75–6, 158, 161, 206–212. For a particularly interesting examination of how the term ‘hacker’ has changed to include the notion of these threats, see Nissenbaum (2004).
2 This theme animates much of Lessig’s more recent work as well in The Future of Ideas: The Fate of the Commons in a Connected World (2003) and more recently in Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity (2004). Both can been seen as efforts to put this fundamental premise into action, especially in relation to reforming copyright and its effect of creativity.
3 Ironically, these are also themselves moments of injury of violence. On this point see Benjamin (1975) and Jacques Derrida (1992).
4 In many ways this can be seen as the origins of hacking as political action. See, for example, Thomas (2002). Also of note is the fundamental transition that took place in the late 1990s, where hackers engaged in direct political intervention by making code appear rather than through acts of disruption or defacement. Although this author would never claim that these two are mutually exclusive, the spirit of this distinction is an important one, which requires one to pay careful attention to the development of the subculture itself.
5 For an in-depth discussion of the relationship between simulacra and resemblance, see Deleuze (1990).

References


DOUGLAS THOMAS is Associate Professor in the Annenberg School for Communication at the University of Southern California. He is the author of *Hacker Culture* (Minnesota University Press, 2002), co-editor (with Marita Sturken and Sandra Ball-Rokeach) of *Technological Visions: the Hopes and Fears that Shape New Technologies* (Temple University Press, 2004), and (with Brian D. Loader) *Cybercrime: Law Enforcement, Security and Surveillance in the Information Age* (Routledge, 2000).

Address: Annenberg School for Communication, University of Southern California, Los Angeles, CA 90089–0281, USA. [email: douglast@usc.edu]