The Ohio State Technology Law Journal

SPYWARE VS. SPYWARE: SOFTWARE CONFLICTS AND USER AUTONOMY

JAMES GRIMMELMANN*

CONTENTS

[*] Professor of Law, Cornell Tech and Cornell Law School. This Essay is a revised version of a Distinguished Lecture given for the Ohio State Technology Law Journal on September 20, 2019. My thanks to the participants there and in the Digital Life Seminar at Cornell Tech, and to Aislinn Black, Mary Anne Franks, Bryan Choi, Efthimos Parasidis, Guy Rub, Tom Dougherty, MC Forelle, Fred von Lohmann, Germán Ricardo Macías, Arvind Narayanan, Helen Nissenbaum, Frank Pasquale, C.E. Petit, and Christopher Thorpe. This essay may be freely reused under the terms of the Creative Commons Attribution 4.0 International license, https://creativecommons.org/licenses/by/4.0.

I. Introduction

This is the story of the time that Apple broke Zoom, and everybody was surprisingly okay with it. The short version is that Zoom provides one of the most widely used video-conferencing systems in the world. One reason for Zoom's popularity is its ease of use; one reason Zoom was easy to use was that it had a feature that let users join calls with a single click. On macOS, Zoom implemented this feature by running a custom web server on users' computers; the server would receive Zoom-specific requests and respond by launching Zoom and connecting to the call.[1] Security researchers realized that that web pages could use this feature to join users to Zoom calls without any further confirmation on their part, potentially enabling surveillance through their webcams and microphones.[2] The researchers released a proof-of-concept exploit in the form of a webpage that would immediately connect anyone who visited it to a Zoom video call with random strangers.[3] They also sketched out ways in which the Zoom server on users' computers could potentially be used to hijack those computers into running arbitrary code.[4]

After the story came to light, Apple's response was swift and unsparing. It pushed out a software update to macOS to delete the

[1] Jonathan Leitschuh, *Zoom Zero Day: 4+ Million Webcams & Maybe an RCE? Just Get Them to Visit Your Website!*, MEDIUM (July 8, 2019), https://medium.com/bugbountywriteup/zoom-zero-day-4-million-webcams-maybe-an-rce-just-get-them-to-visit-your-website-ac75c83f4ef5 [https://perma.cc/7W3Y-LDHY]. Using a custom local web server bypassed security checks ordinarily performed by browsers. *Id. See generally* Dan Goodin, *Zoom for Mac Made It Too Easy for Hackers to Access Webcams. Here's What to Do [Updated]*, ARS TECHNICA (July 9, 2019, 6:33 PM), https://arstechnica.com/informationtechnology/2019/07/zoom-makes-it-too-easy-for-hackers-to-access-webcams-heres-what-to-do/.

[2] Leitschuh, *supra* note 1.

[3] *Id.; see also* Matt Haughey (@mathowie), TWITTER (July 8, 2019, 8:39 PM), https://twitter.com/mathowie/status/1148391109824921600 [https://perma.cc/FP4F-EC27] ("This Zoom vulnerability is bananas. I tried one of the proof of concept links and got connected to three other randos also freaking out about it in real time.").

[4] Assetnote Team, Zoom Zero Day Followup: Getting the RCE, ASSETNOTE (July 17, 2019), https://blog.assetnote.io/bug-bounty/2019/07/17/rce-on-zoom/ [https://perma.cc/M528-7PX9]; Leitschuh, *supra* note 1.

Zoom server and prevent it from being reinstalled.[5] The update was remarkable, and not just because it removed functionality rather than adding it. Typical Apple updates to macOS show a pop-up notification that lets users choose whether and when to install an update. But Apple pushed out this update silently and automatically; users woke up to discover that the update had already been installed—if they discovered it at all. In other words, Apple deliberately broke an application feature on millions of users' computers without notice or specific consent. And then, six days later, Apple did it again.[6]

There is a lot that could be said about this episode; it illuminates everything from responsible disclosure practices[7] to corporate public relations to secure interface design for omnipresent cameras and microphones.[8] But I want to dwell on just how strange it is that one major technology company (AAPL, market capitalization $1.4 trillion[9]) deliberately broke a feature in another major technology company's (ZM, market capitalization $24 billion[10]) product for millions of users, and almost no one even blinked. We are living in a William Gibson future of megacorporations waging digital warfare on

[5] Dan Goodin, *Silent Mac Update Nukes Dangerous Webserver Installed by Zoom*, ARS TECHNICA (July 10, 2019, 7:50 PM), https://arstechnica.com/information-technology/2019/07/silent-mac-update-nukes-dangerous-webserver-installed-by-zoom/ [https://perma.cc/G2SV-P5DC]; Zack Whittaker, *Apple Has Pushed a Silent Mac Update to Remove Hidden Zoom Web Server*, TECHCRUNCH (July 10, 2019, 6:06 PM), https://techcrunch.com/2019/07/10/apple-silent-update-zoom-app/ [https://perma.cc/UD5J-8GEB].

[6] Dieter Bohn, *Apple Is Silently Updating Macs Again to Remove Insecure Software From Zoom's Partners*, VERGE (July 16, 2019, 1:20 PM), https://www.theverge.com/2019/7/16/20696529/apple-mac-silent-update-zoom-ringcentral-zhumu-vulnerabilty-patched [https://perma.cc/RS87-S6C8].

[7] *See* ALANA MAURUSHAT, DISCLOSURE OF SECURITY VULNERABILITIES: LEGAL AND ETHICAL ISSUES (2013); Kristin M. Bergman, *A Target to the Heart of the First Amendment: Government Endorsement of Responsible Disclosure as Unconstitutional*, 13 NW. J. TECH. & INTELL. PROP. 117 (2015).

[8] *See, e.g.*, Matthew Brocker & Stephen Checkoway, iSeeYou: Disabling the MacBook Webcam Indicator LED (Dec. 12, 2013) (unpublished manuscript), https://jscholarship.library.jhu.edu/handle/1774.2/36569 (demonstrating an attack to foil a security feature in which an indicator light was lit whenever a Mac's webcam was turned on).

[9] As of February 11, 2020. *Apple Market Cap*, YCHARTS, https://ycharts.com/companies/AAPL/market_cap (last visited Feb. 20, 2020).

[10] As of February 11, 2020. *Zoom Video Communications Market Cap*, YCHARTS, https://ycharts.com/companies/ZM/market_cap (last visited Feb. 20, 2020).

each other's software and everyone just accepts that this is how life is now.

Lest you think I am dwelling on an isolated and unrepresentative incident, here are some further examples of programs doing drive-bys on each other like warring street gangs:

- *Malware*: Antivirus software attempts to prevent malware from being installed on users' computers, and to remove that software if found. Malware tries to install itself and evade detection and removal, so of course its first order of business is often to turn off any antivirus protection.[11]

- *video game bots*: Some online game players use bots to play the game for them, leveling up their characters and obtaining resources.[12] Blizzard, which operates the popular game World of Warcraft (WoW), added a program called Warden to WoW, which detects bots and reports them to Blizzard so it can ban their users from connecting to Blizzard's servers.[13] One bot maker, MDY, modified its code to evade detection by Warden.[14] Others developed techniques to modify Warden itself and disable its surveillance without alerting Blizzard.[15]

[11] *See, e.g.*, *DoubleAgent: Taking Full Control Over Your Antivirus*, CYBELLUM (Mar. 22, 2017), https://cybellum.com/doubleagent-taking-full-control-antivirus/ [https://perma.cc/2BMR-VEY5]; *Malware Uses Certificates to Disable the Installation of Anti-Malware Solutions on Your Computer*, BITDEFENDER, https://www.bitdefender.com/consumer/support/answer/1921/ [https://perma.cc/6N27-E8QK].

[12] *See, e.g.*, GREG HOGLUND & GARY MCGRAW, EXPLOITING ONLINE GAMES: CHEATING MASSIVELY DISTRIBUTED SYSTEMS 19 (2007).

[13] *See, e.g.*, Andy Chalk, *World of Warcraft Bot Factory Gives Up After Massive Blizzard Banhammering*, PC GAMER (May 15, 2015), https://www.pcgamer.com/world-of-warcraft-bot-factory-gives-up-after-massive-blizzard-banhammering/ [https://perma.cc/V5RV-H7J4].

[14] MDY Indus. v. Blizzard Entm't, 629 F.3d 928, 936 (9th Cir. 2010).

[15] *Deceiving Blizzard Warden*, HACKMAG, https://hackmag.com/uncategorized/deceiving-blizzard-warden/ [https://perma.cc/MTA7-K3TS].

- *Ad blocking*: Some websites show ads.[16] In response, some users install adblockers in their browsers to block the ads on websites they visit. In reply, some websites detect when ads are being blocked and refuse to display content unless the adblockers are disabled. In surreply, some adblockers disguise from websites the fact that their ads are being blocked. Or, in reply, some websites modify their ads so that adblockers cannot detect them, and in surreply adblockers use more sophisticated techniques to recognize the mutated ads. In the words of Parker Higgins, it is "[i]ncreasingly obvious that any debate about adblockers is a thin veneer over questions of basic control of computers."[17]

- *Ad injection*: Or, maybe it is a browser plugin that shows the ads and the website that objects. Today the preferred term is "ad injectors"—defined as software "that modifies a page's content to insert or replace advertisements, irrespective of user consent"[18]—although readers of a certain age may remember the litigation over "popup ads."[19] Browser vendors have adopted increasingly stringent rules to restrict ad injectors.[20]

- *Browser tracking*: Websites use browser APIs, including placing cookies on users' computers, to gather information

[16] *See generally* Russell A. Miller, Liberation, Not Extortion: The Fate of Ad-Blocking in German and American Law (Aug. 15, 2017) (unpublished manuscript), https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3019254; Grant Storey et al., The Future of Ad Blocking: An Analytical Framework and New Techniques (May 24, 2017) (unpublished manuscript), https://arxiv.org/pdf/1705.08568.pdf.

[17] Parker Higgins (@xor), TWITTER (Sept. 7, 2015, 5:06 PM), https://twitter.com/xor/status/640994476480069632 [https://perma.cc/9FG2-AEBY].

[18] Kurt Thomas et al., *Ad Injection at Scale: Assessing Deceptive Advertisement Modifications*, *in* PROC. 2015 IEEE SYMPOSIUM ON SECURITY AND PRIVACY 151, 152 (2015).

[19] *E.g.*, 1-800-Contacts, Inc. v. WhenU.com, Inc, 414 F.3d 400 (2d Cir. 2005). For a more modern example, see Halperin v. Int'l Web. Serv., LLC, 70 F. Supp. 3d 893 (N.D. Ill. 2014).

[20] Nav Jagpal, *Out with Unwanted Ad Injectors*, GOOGLE SECURITY BLOG (Mar. 31, 2015), https://security.googleblog.com/2015/03/out-with-unwanted-ad-injectors.html [https://perma.cc/BDR9-PWS5].

about users and track them from page to page and site to site. In response, browsers allow users to block or delete cookies to prevent websites from recognizing them. In reply, websites have deployed ever more sophisticated techniques to fingerprint users' browsers based on other features, such as which fonts they have installed[21] and the characteristics of their computer's battery.[22] Also in reply, websites have deployed software techniques to circumvent browser-based cookie blocking, for example by simulating user input so that browsers think that users had consciously interacted with websites.[23] In surreply, browser makers have removed or restricted the APIs enabling these forms of tracking, and take increasingly strong measures against websites they identify as circumventing users' cookie settings.[24]

- *Email tracking*: Emails can include HTML that refers to resources on the web, which has been used for years by email senders to see who has opened an email—by including an image with a URL unique to a particular

[21] *E.g.*, Gunes Acar et al., *FPDetective: Dusting the Web for Fingerprinters*, *in* CCS '13: PROC. 2013 ACM SIGSAC CONF. ON COMP. & COMM. SECURITY 1129, 1130 (2013), https://dl.acm.org/doi/10.1145/2508859.2516674; Peter Eckersley, *How Unique Is Your Web Browser?*, *in* PRIVACY ENHANCING TECHNOLOGIES 1, 4 (Mikhail Atallah & Nicholas Hopper eds., 2011).

[22] Łukasz Olejnik et. al., *The Leaking Battery: A Privacy Analysis of the HTML5 Battery Status API*, *in* DATA PRIVACY MANAGEMENT, AND SECURITY ASSURANCE 254, 254 (Joaquin Garcia-Alfaro et al. eds., 2016).

[23] *See* In re Google Inc. Cookie Placement Consumer Privacy Litigation, 806 F.3d 125, 131-32 (3rd Cir. 2015); Jonathan Mayer, *Safari Trackers* (Feb. 17, 2012), http://webpolicy.org/2012/02/17/safari-trackers/.

[24] *E.g.*, Bill Buddington, *Apple's New WebKit Policy Takes a Hard Line for User Privacy*, ELECTRONIC FRONTIER FOUND.: DEEP LINKS (Aug. 20, 2019), https://www.eff.org/deeplinks/2019/08/apples-new-webkit-policy-takes-hard-line-user-privacy [https://perma.cc/MPJ5-5VG6]; Marissa Wood, *Today's Firefox Blocks Third-Party Tracking Cookies and Cryptomining by Default*, MOZILLA: BLOG (Sept. 3, 2019), https://blog.mozilla.org/blog/2019/09/03/todays-firefox-blocks-third-party-tracking-cookies-and-cryptomining-by-default/ [https://perma.cc/TPT9-ABY3].

user.[25] If that URL is loaded, the user has opened the email. So, of course, some email readers include options not to load remote resources unless the user specifically asks to.[26]

- *Jailbreaking*: Some operating systems make it difficult or impossible to install software not approved by the operating-system vendor.[27] Unsurprisingly, at the more restrictive end there is a market for programs that will allow the installation of other programs the operating-system vendor has attempted to prevent.[28] Some of these programs are used by device owners who want to "jailbreak" their devices to add new programs;[29] some are used by hackers to surveil users;[30] some are used by law enforcement to decrypt devices during investigations.[31] What happens when operating-system vendors discover that one of these programs is in use? They push out an update to the operating system to disable it.[32]

[25] *E.g.*, Mike Davidson, *Superhuman Is Spying on You*, MIKE INDUSTRIES (June 30, 2019), https://mikeindustries.com/blog/archive/2019/06/superhuman-is-spying-on-you [https://perma.cc/6QBH-LRVQ].

[26] John Gruber, *Superhuman and Email Privacy*, DARING FIREBALL (July 23, 2019), https://daringfireball.net/2019/07/superhuman_and_email_privacy [https://perma.cc/UQY7-LCQH].

[27] *See, e.g.*, *Safely Open Apps on Your Mac,* APPLE (Oct. 7, 2019), https://support.apple.com/en-us/HT202491 [https://perma.cc/B3DZ-EPQG].

[28] *E.g.*, Lily Hay Newman, *Unfixable iOS Device Exploit Is the Latest Apple Security Upheaval*, WIRED (Sept. 27, 2019, 3:18 PM), https://www.wired.com/story/ios-exploit-jailbreak-iphone-ipad/ [https://perma.cc/6TNH-UDED].

[29] *See, e.g.*, PANGU, http://en.pangu.io.

[30] *See, e.g.*, Ian Beer, *A Very Deep Dive into iOS Exploit Chains Found in the Wild*, PROJECT ZERO (Aug. 29, 2019), https://googleprojectzero.blogspot.com/2019/08/a-very-deep-dive-into-ios-exploit.html [https://perma.cc/9WY3-GL2G].

[31] *E.g.*, Andy Greenberg, *Cellebrite Says It Can Unlock Any iPhone for Cops*, WIRED (June 14, 2019, 6:05 PM), https://www.wired.com/story/cellebrite-ufed-ios-12-iphone-hack-android/ [https://perma.cc/D2QH-YQWJ].

[32] *E.g.*, Shaun Nichols, *Breaking News: Apple Un-Breaks Break on Jailbreak Break*, REGISTER (Aug. 26, 2019, 11:38 PM), *https://www.theregister.co.uk/2019/08/26/apple_fixes_ios124_jailbreak/* [https://perma.cc/TPP9-V9BN].

- *Browser certificates*: These contain public keys used by browsers to verify the identities of websites—thereby ensuring that users' communications with those websites are securely encrypted. ISPs in Kazakhstan required users to download and install a government-issued certificate, potentially allowing the government to eavesdrop on their communications with major websites like Facebook and Twitter.[33] In response, Mozilla, Apple, and Google disabled that certificate in their browsers, no matter how it was installed.[34]

- *DRM*: In the antediluvian pre-streaming days of digital music, many users would use "ripping" software to make MP3 copies of their CDs on their computers. Sony/BMG shipped a number of CDs which installed their own digital rights management (DRM) software on PCs in which they were inserted.[35] This software—XCP and MediaMax CD-3—prevented common ripping software from reading or making copies of Sony/BMG CDs.[36] It also modified users' computers in ways designed to make it harder to remove; XCP in particular took steps to conceal its presence on users' computers and created additional security vulnerabilities that other attackers could use to install their own software on users' computers. Security researchers compared this DRM software to "rootkits": forms of malware that actively resist attempts to uninstall them by

[33] *See* RAM SUNDARA RAMAN ET AL., KAZAKHSTAN'S HTTP INTERCEPTION (2019), https://censoredplanet.org/kazakhstan [https://perma.cc/72XV-7VAR].

[34] Catalin Cimpanu, *Apple, Google, and Mozilla Block Kazakhstan's HTTPS Intercepting Certificate*, ZDNET (Aug. 21, 2019, 10:00 PM), https://www.zdnet.com/article/apple-google-and-mozilla-block-kazakhstans-https-intercepting-certificate/; Sydney Li, *Browsers Take a Stand Against Kazakhstan's Invasive Internet Surveillance*, ELECTRONIC FRONTIER FOUND.: DEEP LINKS (Aug. 22, 2019), https://www.eff.org/deeplinks/2019/08/browsers-take-stand-against-kazakhstans-invasive-internet-surveillance [https://perma.cc/5PKT-9R89].

[35] *See generally* Deirdre K. Mulligan & Aaron K. Perzanowski, *The Magnificence of the Disaster: Reconstructing the Sony/BMG Rootkit Incident*, 22 BERK. TECH. L.J. 1157, 1158 (2007).

[36] Mark Russinovich, *Sony, Rootkits and Digital Rights Management Gone Too Far*, MARK'S BLOG (Oct. 31, 2005), https://blogs.technet.microsoft.com/markrussinovich/2005/10/31/sony-rootkits-and-digital-rights-management-gone-too-far/ [https://perma.cc/9HLZ-8UX9].

> hiding, disabling removal programs, and reinstalling themselves if partially removed.

I could go on, but you get the picture.

One way to make sense of these program-versus-program conflicts would be to proceed methodically through the bodies of law that could be (and have been) brought to bear on them. But their sheer number is stunning. There are statutory computer-misuse claims under the Computer Fraud and Abuse Act and its state analogs against programs that access users' computers without authorization.[37] There are property-tort claims for trespass to chattels against programs that harm users' computers.[38] There are contractual claims by users against programs that break their promises, and tortious interference claims against programs that keep other programs from working as promised.[39] There are copyright claims for modifying programs and content in unapproved ways;[40] there are trademark claims for passing off modifications as the original, and for misrepresenting the relationship between a program and its victim.[41] Section 1201 of the Digital Millennium Copyright Act[42] prohibits circumventing technological protections on copyrighted works (including music on CDs and games like World of Warcraft),[43] and section 1202,[44] which has been interpreted to prohibit stripping certain kinds of metadata from copyrighted works,[45] might also sometimes be in play. When a program justifies disabling another on the ground that it is harmful—as antivirus software does with malware—this justification may itself sometimes be actionable as trade libel, or as defamation of its

[37] 18 U.S.C. § 1030 (2008).

[38] The leading case on online trespass to chattels, *Intel Corp. v. Hamidi*, 71 P.3d 296 (Cal. 2003), held that the tort did not lie without "some actual injury," but that requirement will typically be satisfied when a defendant "impairs [the] functioning" of a program on the plaintiff's computer. *Id.* at 300.

[39] *E.g.*, Zango, Inc. v. Kaspersky Lab, Inc., 568 F. 3d 1169, 1171-72 (9th Cir. 2019).

[40] *E.g.*, MDY Indus. v. Blizzard Entm't, 629 F.3d 928, 937 (9th Cir. 2010).

[41] *E.g.*, U-Haul Intern. v. whenU.com, Inc., 279 F. Supp. 2d 723, 727-29 (E.D. Va. 2003).

[42] 17 U.S.C. § 1201 (1998).

[43] *MDY Indus.*, 629 F.3d at 943-52.

[44] 17 U.S.C. § 1202 (1999).

[45] *See e.g.*, Murphy v. Millennium Radio Grp., 650 F.3d 295, 305 (2011).

developers.[46] If the makers of the two programs compete, actions by one against the other might violate the antitrust laws.[47] Any of the above in violation of a privacy policy, or terms of service, or other representation to users might be a deceptive trade practice in the view of the Federal Trade Commission and state attorneys general.[48] Cutting across almost all of the above there are some commonly arising defenses, such as Section 230(c)(2) of the Communications Decency Act, which protects "any action voluntarily taken in good faith to restrict access to or availability" of "objectionable" material.[49] And at the Constitutional level, restrictions on software functionality can raise First, Fifth, and Fourteenth Amendment issues.

The length of this list should give pause. If there is a principled way to resolve these software-versus-software conflicts, it needs a firmer foundation than a mess of doctrinal detail. If these bodies of law reach consistent results, we should seek the common thread that explains them all. If they reach inconsistent results, we should seek a coherent basis to harmonize them. Either way, we need a theory. So I would like to come at the problem the other way around: what kinds of principles might help sort out these cases?

Part II of this essay describes three seemingly appealing heuristics for resolving software conflicts—banning bad software, promoting user freedom, and enforcing contracts—each of which fails badly when confronted with common fact patterns. Part III argues that the missing element is user autonomy: only by connecting software's effects for users with their choices of what software to run and what contracts to agree to is it possible to make sense of software conflicts.

II. Software Conflicts

[46] *See, e.g.*, NEW.NET v. Lavasoft, 356 F. Supp. 2d 1071, 1113 (C.D. Cal. 2003).

[47] *See e.g.*, In re Apple iPod iTunes Antitrust Litigation, 796 F. Supp. 2d 1137, 1143 (N.D. Cal. 2011).

[48] 15 U.S.C. § 45 (2006).

[49] 47 U.S.C. § 230(c)(2) (1996).

Three theories of software conflicts are so straightforward, so widespread, and so intuitively appealing that they are often simply assumed. The first is that certain program behavior is intrinsically harmful and should be prohibited. Programs should not spy on users and delete their files. Call this theory "Bad Software Is Bad." The second is that users should be allowed to run whatever software they want. Call this theory "Software Freedom." And the third is that both users and software vendors should be held to the terms of whatever contracts they enter into. Call this "Click to Agree." Each theory captures an important insight about software but is incomplete on its own. Each theory gives good explanations in some easy cases but quickly runs into trouble in harder cases. Sometimes the theories agree, and sometimes they do not. We can understand much about software conflicts by studying the cases where one theory fails and another succeeds. We can understand even more by studying the cases where all three fall short.

a. Bad Software Is Bad

The first, and in some ways most intuitive, theory focuses on the technical characteristics of the software itself. Most programs are Good and do useful things for users. But some programs are Bad. Programs can be Bad because they harm users by invading their privacy and deleting their data, because they harm other people by pirating copyrighted works and making pornographic deepfakes, or because they harm other programs in all of the ways listed above. The legal system should intervene when Bad programs do Bad things, including when they disable Good programs. And, a little more subtly, the legal system should allow Good programs to disable Bad programs.

The underlying intuition here is sound. Some programs really are objectively Bad. Spousal spyware can put users in physical danger by enabling abusive partners to stalk them.[50] Ransomware that encrypts users' files until they send Bitcoin in exchange for a decryption key is also 100% downside: it does nothing good for its victims, ever.[51] Norton AntiVirus is Good;[52] NotPetya is Bad.[53] It makes perfect sense that the former should be allowed to block the latter, just as Bad Software Is Bad recommends. A theory that did the opposite and took the side of "the most devastating cyberattack in history" over antivirus software trying to stop it would be a non-starter.[54]

These are easy cases because the costs and benefits are so lopsided: one of the two programs is all cost and no benefit. To be sure, in the blasted *Fury Road* hellscape that is Internet security, there is no shortage of obvious villains, and thus no shortage of easy cases. But not all cases are so easy.

Compare a stereotypically Good program like Chrome Remote Desktop[55] with a stereotypically Bad program like FlawedAmmyy.[56] The former is thought of as a useful utility that lets system administrators upgrade employees' computers and provide tech support; the latter is thought of as a malicious "remote access Trojan"

[50] *E.g.*, Rahul Chatterjee et al., *The Spyware Used in Intimate Partner Violence*, *in* PROC. 2018 IEEE SYMPOSIUM ON SECURITY AND PRIVACY 441, 441 (2018), https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8418618.

[51] *See generally* MALWAREBYTES, CYBERCRIME TACTICS AND TECHNIQUES: RANSOMWARE RETROSPECTIVE (2019), https://resources.malwarebytes.com/files/2019/08/CTNT-2019-Ransomware_August_FINAL.pdf; GAVIN O'GORMAN & GEOFF MCDONALD, RANSOMWARE: A GROWING MENACE (2012), https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ransomware-a-growing-menace.pdf.

[52] *But see* Iulia Ion et al., *"...No One Can Hack My Mind": Comparing Expert and Non-Expert Security Practices*, *in* SOUPS 2015: PROC. ELEVENTH SYMPOSIUM ON USABLE PRIVACY AND SECURITY 327, 330-31 (2015), https://www.usenix.org/system/files/conference/soups2015/soups15-paper-ion.pdf (reporting that security experts are *less* likely to recommend anti-virus software than non-experts are).

[53] *See* Andy Greenberg, *The Untold Story of NotPetya, the Most Devastating Cyberattack in History*, WIRED (Aug. 22, 2018, 5:00 AM), https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/ [https://perma.cc/AF6Y-EZWE].

[54] *Id.*

[55] CHROME REMOTE DESKTOP, https://remotedesktop.google.com (last visited Jan. 20, 2020).

[56] *See* Proofpoint Staff, *Leaked Ammyy Admin Source Code Turned into Malware*, PROOFPOINT: BLOG (Mar. 7, 2018), https://www.proofpoint.com/us/threat-insight/post/leaked-ammyy-admin-source-code-turned-malware [https://perma.cc/PQ66-G3U4].

used by hackers to steal data and spy on users. But they have substantially identical functionality: they let someone use a computer over the Internet as though they were sitting at the keyboard and looking at its screen. The difference is that people we call heroes use Google Remote Desktop to do good and people we call villains use FlawedAmmyy to do evil.

It is not that there is no difference between good and evil online. It is just that the difference is not a purely technical one. Even in cases where the answer seems intuitively clear—surely Mozilla Firefox is Good and the Kazakhstani government's surveillance scheme is Bad—the clarity comes not from the functional characteristics of the software itself but from the context in which it is used. Firefox ships with over 150 certificates,[57] and it has a feature to install more.[58] The determination that the Kazakhstani government was up to no good with its ISP-supplied certificate rested on contextual knowledge about how it was likely to spy on users with the certificate, rather than anything inherent to the certificate itself.

Like certificates, many programs are dual use: they have both lawful and unlawful uses. Remote desktop tools themselves are a good example: they are used both by actual tech support and by tech-support scammers.[59] A program that deletes a remote desktop tool might be thwarting a crime, committing one, or both. Spyware often falls into this dual-use grey area: it is marketed as being for families wanting to keep in touch with each other, with a wink-wink nudge-nudge understanding that some actual uses will be less benign.[60]

[57] *See Mozilla Included CA Certificate List*, MOZILLA WIKI, https://wiki.mozilla.org/CA/Included_Certificates (last visited Jan. 20, 2020).

[58] *See Setting Up Certificate Authorities (CAs) in Firefox*, MOZILLA SUPPORT, https://support.mozilla.org/en-US/kb/setting-certificate-authorities-firefox [https://perma.cc/86MU-AVB4].

[59] *See* MICROSOFT, GLOBAL TECH SUPPORT SCAM RESEARCH 2, 4-5 (2018), https://news.microsoft.com/uploads/prod/sites/358/2018/10/Global-Results-Tech-Support-Scam-Research-2018.pdf.

[60] *See, e.g.*, Complaint at 2-4, In re Retina-X Studios, LLC, No. 1723118 (F.T.C. filed Oct. 7, 2019), https://www.ftc.gov/system/files/documents/cases/172_3118-retina-x_studios_complaint_updated.pdf.

Nor does it help to rely on institutional identity. Obscure lone wolves can produce Good software—some of the Internet's most essential infrastructure is written and maintained by individual volunteers.[61] On the other hand, major corporations can produce Bad software. The Sony/BMG rootkit is a prime example of software from a Fortune 500 company that intentionally introduced egregious security violations to users' computers. One of the world's leading antivirus makers, the Russian cybersecurity company Kaspersky, has been accused of using its antivirus software to exfiltrate classified documents from the U.S. government.[62]

In some cases, "Good" and "Bad" are themselves contested. For many copyright owners, it is obvious that DRM is Good and circumvention tools are Bad; many open-source advocates and copyright skeptics would say exactly the opposite. Some people think that it is fine to play World of Warcraft with bots; others vehemently disagree. Advertisers and adblockers have conflicting views about the legality and morality of viewing content without the accompanying ads. (The CEO of Turner Broadcasting once said that viewers make a "contract" with TV broadcasters to watch the ads and called skipping commercials "stealing."[63]) This is not to say that one cannot coherently invoke Bad Software Is Bad to condemn cheatbots or adblockers. It is just that in order to do so one must first take a position on fiercely debated normative and policy issues.

[61] *See* NADIA EGHBAL, ROADS AND BRIDGES: THE UNSEEEN LABOR BEHIND OUR DIGITAL INFRASTRUCTURE (2017), https://www.fordfoundation.org/media/2976/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure.pdf.

[62] Shane Harris & Gordon Lubold, *Russia Has Turned Kaspersky Software into Tool for Spying*, WALL ST. J. (Oct. 11, 2017, 1:44 PM), https://www.wsj.com/articles/russian-hackers-scanned-networks-world-wide-for-secret-u-s-data-1507743874; Raphael Satter, *Mysterious Operative Haunted Kaspersky Critics*, ASSOCIATED PRESS (Apr. 17, 2019), https://apnews.com/a3144f4ef5ab4588af7aba789e9892ed. *But see* Catalin Cimpanu, *EU: No Evidence of Kaspersky Spying Despite 'Confirmed Malicious' Classification*, ZDNET (Apr. 16, 2019, 6:31 PM), https://www.zdnet.com/article/eu-no-evidence-of-kaspersky-spying-despite-confirmed-malicious-classification/.

[63] *See* Ernest Miller, *Top Ten New Copyright Crimes*, LAWMEME (May 2, 2002, 1:05 PM), https://web.archive.org/web/20020604021107/http://research.yale.edu/lawmeme/modules.php?name=News&file=article&sid=198.

And even if the underlying principles are clear, in some cases their application will be muddled because *both* programs will be able to make out a claim to the same principle. Two pieces of 'antivirus' software may try to disable the other, each claiming that it is the real deal and the other is an impostor wearing a badly fitting antivirus mask.[64] Which is which? The principle that antivirus software is Good and viruses are Bad is not sufficient on its own. Some people think RegHunter is a harmful virus; others consider it useful antivirus software.[65] Up and down depend on where you're standing.

There is a running theme here. Bad Software Is Bad conflates the question of whether software is good or bad *in general* with whether it is good or bad *for a specific user*. Some users need remote tech support, others don't. Some users want to block ads, others don't. Surely, then, we have an issue of fact which users themselves are peculiarly fitted to determine. Our next theory does just that.

b. Software Freedom

Software Freedom, for our purposes, is "the freedom to run [any] program as you wish, for any purpose."[66] This is a straightforward

[64] *See* Enigma Software Grp. USA v. Malwarebytes Inc., 938 F.3d 1026, 1030 (9th Cir. 2019); Zango, Inc. v. Kaspersky Lab, Inc., 568 F. 3d 1169, 1171-72 (9th Cir. 2019). *See generally* Brett Stone-Gross et al., *The Underground Economy of Fake Antivirus Software*, *in* ECONOMICS OF INFORMATION SECURITY AND PRIVACY III 55 (Bruce Schneier ed., 2012).

[65] *See Enigma Software Grp.*, 938 F.3d at 1033, 1037.

[66] Free Software Foundation, *What Is Free Software?*, GNU OPERATING SYS., https://www.gnu.org/philosophy/free-sw.en.html [https://perma.cc/SV3M-U6NY]. I am borrowing the term "software freedom" and this definition from the Free Software Foundation's free software movement. *Id.* Although the idea of "free software" is most often deployed in the context of intellectual property restrictions on the use, modification, and distribution of software, that is principally because these restrictions have been the most salient. *See generally* COPYLEFT AND THE GNU GENERAL PUBLIC LICENSE: A COMPREHENSIVE TUTORIAL AND GUIDE (2018), https://www.copyleft.org/guide/comprehensive-gpl-guide.pdf (discussing the use of copyright licensing law to ensure software freedom). But philosophically, the ideas are broader and encompass any kind of restrictions on user freedom. *See generally* SAMIR CHOPRA & SCOTT DEXTER, DECODING LIBERATION: THE PROMISE OF FREE AND OPEN SOURCE SOFTWARE (2007). I refer to "software freedom" rather than "free software" to emphasize that it is freedom to use the software, not the freedom of the software itself, that

negative liberty: users should be free to use any software they want, even if it interferes with other software. Programs have no rights that users are bound to respect, and neither do software vendors or third parties. The principle can be rooted in freedom of speech and thought, in economic liberty, or in users' property rights over their own devices. But the upshot is the same: users can run any software they want, and the legal system will not attempt to stop them.

A general principle that the law does not meddle in the affairs of users is easy to articulate and easy to administer. Users who want a program can run it. Users who don't want a program can refrain from running it. Neither case presents a legal question. Similarly, if Program A disables Program B, it is no concern of the legal system. The fact that it did so presumptively reflects a choice by the user to run Program A knowing of its effects on Program B. Whatever Program B did, they valued it less than what Program A now offers. In fact, often they will have run Program A for the specific purpose of stopping Program B from doing something they didn't want. Every time a user runs CleanMyMac to remove an old and unwanted program that is just taking up hard drive space, Software Freedom correctly says, "Move along, nothing to see here." Virus, meet antivirus. Ad, meet adblocker.

The clearest doctrinal expression of this deregulatory impulse is in Section 230(c)(2), which prohibits imposing liability for "any action voluntarily taken in good faith to restrict access to or availability of material ... that the provider or user considers to be ... objectionable."[67] The courts have held that this protection applies, for example, to antivirus programs that flag other software for removal.[68] But numerous other doctrines have a similar effect, simply by making it hard to bring claims. For example, the creator of a disabled program typically lacks standing to raise a CFAA or trespass to chattels claim

matters. *See* Benjamin Mako Hill, *Freedom for Users, Not for Software*, *in* THE WEALTH OF THE COMMONS (David Bollier & Silke Helfrich eds., 2014), http://wealthofthecommons.org /essay/freedom-users-not-software.

[67] 47 U.S.C. § 230(c)(2)(A) (1996).

[68] Zango, Inc. v. Kaspersky Lab, Inc., 568 F. 3d 1169, 1177-78 (9th Cir. 2019). *But see Enigma Software Grp.*, 938 F.3d at 1035-37 (distinguishing *Zango* and holding that Section 230(c)(2) does not apply when the removal is for "anticompetitive reasons").

against the creator of the program that disabled it. It is the user's computer, not theirs, and it is the user who gets to decide what software runs on it. Software Freedom also argues against the use of copyright to restrict software modifications, against legally mandated software updates, and against anti-circumvention law.

It is illuminating to compare Software Freedom's prescriptions with Bad Software Is Bad's:

- Bad Software Is Bad brims with unjustified confidence that it can distinguish software suitable for all users from software suitable for none, and so it pays no attention to individual users' choices about software. Software Freedom, by contrast, eschews such distinctions and instead defers to users. The fact that a user chooses to run a program is sufficient evidence that the program is useful. Indeed, a user's choice to run a program is constitutive of the fact that the program is actually Good for them.

- Bad Software Is Bad is not capable of staying its hand in cases where users disagree on the right outcome, but Software Freedom is. It is not trademark infringement for a website to set cookies in a browser, but neither is it trademark infringement for the browser to clear them. Both "allow cookies" and "block cookies" are reasonable outcomes that reasonable users could choose. Software Freedom's deregulatory approach creates a technical space within which genuine user choice is possible.

- In some controversial cases, Software Freedom supports a program that does what its users want, whereas Bad Software Is Bad condemns it because it causes third-party harms. This is a straightforward clash of values; users' interest in running DRM-cracking software runs directly up against copyright owners' interest in using DRM software to limit access to copyrighted works. That is, Software Freedom's user focus has a specifically libertarian bent toward maximal freedom of action, whereas Bad Software Is Bad has a more communitarian bent. The proper balance between these two incomparable principles is a question of policy.

- Bad Software Is Bad can also be deployed paternalistically to prevent users from running software on the grounds that doing so will be bad for them. You may think you want to cheat at World of Warcraft, but think again: winners never cheat, and cheaters never win. Or, perhaps more persuasively, you may think you want to allow this person from "Windows Tech Support" full access to your computer, but think again. Software Freedom is an appealing principle if you think that users make good decisions. But if not, then Bad Software Is Bad can help protect them in a way that Software Freedom cannot.

- In many easy cases, both theories reach the same clearly right result, because users' goals and regulators' goals align. Antivirus software generally does what users want and is good for them. Software libertarianism and software paternalism converge.

So far, I have situated Software Freedom in contrast to Bad Software Is Bad: deregulation versus regulation, with the predictable tensions and tradeoffs involved. But just as the contestability of the definition of "Bad" software undermines the foundations of Bad Software Is Bad, there is also an instability in the foundations of Software Freedom.

One way to get at the issue is the observation that Software Freedom risks turning users' computers into free-fire zones: anything a program can get away with is permitted. One baleful consequence is that it encourages software vendors to engage in a technical arms race of escalating self-help. If Program A sends one of Program B's modules to the hospital, Program B may respond by sending one of Program A's to the morgue. This escalation is wasteful, as all unchecked arms races are. For example, Software Freedom makes no effort to end the cookie-blocking wars: blocking cookies looks like an exercise of user freedom, and so do the workarounds websites use to get around cookie blocking. The user who visits one of these websites has chosen to run a program—the website's workaround—that interferes with another program—the cookie blocker. So be it. No matter how irrational it may appear to run mutually incompossible programs in endlessly alternating succession, Software Freedom offers no basis to second-guess a user's decisions.

Software-versus-software arms races also cause collateral damage as uninvolved software gets caught in the crossfire. Cookie blockers interfere with websites that are just trying to add useful features, not track users.[69] The Sony/BMG rootkit was so tenacious about installing itself in a way CD-ripping software couldn't defeat that it created exploitable security vulnerabilities on users' computers.[70]

Nor, at the end of the day, will the "right" program always win the arms race. Users need good programs with guns to guard against bad programs with guns, but sometimes the bad programs have bigger guns. Software Freedom lets antivirus software delete malware—but it also lets malware delete antivirus software. By declining to step in, Software Freedom avoids thwarting user choices to run software—but by refusing to step in at all, it allows other software to thwart those choices. That outcome can be just as disempowering.

The distinction that antivirus software protects files while malware deletes them is not one Software Freedom on its own is capable of making. Users sometimes type `rm -r *` to delete files in bulk. Indeed, it is central to the idea of Software Freedom that it rejects Bad Software Is Bad's sharp distinction between "safe" and "dangerous" programs. Users sometimes work with dangerous programs, like amateur experimenters giving themselves fecal transplants.[71] Indeed,

[69] *See, e.g.*, *Blocking Third-Party Cookies Breaks the Save Button*, PLANAPPLE, https://planapple.uservoice.com/knowledgebase/articles/509652-blocking-third-party-cookies-breaks-the-save-butto [https://perma.cc/PTU2-GFLK].

[70] *See* Mulligan & Perzanowski, *supra* note 35, at 1158-60.

[71] *See* Denise Grady, *Fecal Transplant Is Linked to a Patient's Death, the F.D.A. Warns*, N.Y. TIMES (June 13, 2019), https://www.nytimes.com/2019/06/13/health/fecal-transplant-fda.html [https://perma.cc/9YNN-KC3J].

any sufficiently advanced exercise of Software Freedom is indistinguishable from a security hole. The ability to modify a system in deep and powerful ways is the hallmark both of freedom and of malware, just like the ability to pack a lot of chemical energy into a small volume is the hallmark both of batteries and of bombs. Android is both more extensible and more vulnerable than iOS.

Indeed, Software Freedom even has trouble with the distinction that the user has voluntarily installed antivirus software but not the virus; it is precisely the refusal to second-guess the user's actions that makes the pure form of Software Freedom so simple and administrable. The spear-phishing victim who clicks on an emailed link to open what they think is a website and turns out to be a spyware executable has voluntarily interacted with something, even if it turned out not to be what they expected. To distinguish the user who accidentally clicks on a malware link from a security researcher who deliberately runs malware on a sandboxed PC to study it, something more is needed.

I hinted at this when I said that one might question Software Freedom if one thinks that users make bad choices. But the issue is deeper and subtler than that. *Many things that happen on a user's computer are not the user's "choices" in any meaningful sense.* Whether or not a program's actions are the intended result of a deliberate exercise of user freedom is a question on which Software Freedom depends but cannot by itself answer, except by collapsing into the trivial claim that anything that happens on a computer is deliberate. (But when everything is deliberate, nothing is.) Software Freedom can distinguish between "allow cookies" and "block cookies," because they have different results. But it cannot distinguish between "allow cookies" and "attempt to block cookies but fail." The user's intent is not always reflected in what actually happens on a computer—otherwise, it would make no sense to make computer misuse torts and crimes turn on "authorization."[72] Sometimes users are deceived about what a program will do. Sometimes they change their minds but lack the technical

[72] *See* James Grimmelmann, *Consenting to Computer Use*, 84 GEO. WASH. L. REV. 1500, 1501 (2016).

skills to drag an icon off the desktop, let alone uninstall the program it represents. Sometimes they are confused about how programs will interact. And sometimes software makers lie about what their programs will do, or they install programs without even a semblance of user consent.

To be useful, a theory of software conflicts must be willing to say that some software really is unwanted (as Bad Software Is Bad does) and to say that some software really is wanted (as Software Freedom does). But it must also have a workable test for saying *which* software has a user's permission to run and which does not. A line must be drawn, and some principle besides "let the memory chips lie where they fall" is needed to draw it. Our next theory does just that.

c. Click to Agree

The standard approach to ascertaining user consent in the United States circa 2020 is contractual. Users agree to the terms of a contract, privacy policy, or other instrument when they provide a specific manifestation of assent to it after having been given notice and a reasonable opportunity to review its terms.[73] Contractual agreement provides three mechanisms for affecting the outcome in software-versus-software cases, along with a dog that does not bark in the night:

- A user can assent to software's actions: they agree that the software is allowed to do *X* and that they cannot sue the software maker when the software goes ahead and does *X*. Such terms can therefore be controlling if a user tries to object when software covered by such an agreement disables other software: having agreed to let us disable software, you cannot now object that we disabled it.

- A little more subtly, the terms can help protect software against being disabled. A user can promise not to decrypt, modify, disable, or even reverse engineer the software they

[73] *See, e.g.*, Meyer v. Uber Tech. Inc., 868 F.3d 66, 74-76 (2nd Cir. 2017).

are installing, giving the software maker rights against users who do.

- More subtly still, the terms can help provide a foundation for a suit directly against the maker of the other software doing the disabling. For example, the World of Warcraft license agreement played an indispensable role in Blizzard's suit against Glider.[74]

- In theory, terms could also create contractual obligations on the software vendor's part: for example, that it will not collect certain information from the user or damage her computer. In practice, consumer terms of service and end-user license agreements typically disclaim all such obligations as far as possible.

Call this standard approach Click to Agree: users and software vendors will be held to the terms of whatever contractual agreements they voluntarily enter into. There is a lot to be said for the standard approach. Most obviously, it is capable of drawing the basic required distinction between software installed with permission and software without. Typical antivirus software gets user consent to a license agreement; the typical virus does not. Moreover, the test is administrable. Although there is lingering judicial uncertainty around "clickwrap," "browsewrap," and other poorly defined terms, there is a clear pathway for a software maker to get legally sufficient user agreement: display an unambiguous call to action and require a specific click to agree.

Click to Agree also does a reasonable job at boiling software conflicts down into a coherent doctrinal framework: identify the relevant agreements, read them, and apply their terms. One common thread in the large range of doctrines implicated by software conflicts is that many of them can be resolved by looking to user consent.

[74] MDY Indus. v. Blizzard Entm't, 629 F.3d 928, 939-41 (9th Cir. 2010).

"Authorization" under the CFAA is consent.[75] So is permission of the owner under trespass to chattels. Copyright and trademark infringement claims are defeated by licenses from the owner. And so on.

Thus, Click to Agree solves the most obvious deficiencies with Bad Software Is Bad and with Software Freedom. Unlike the former, it respects different users' different choices, so it can distinguish users who want cookie blockers from users who do not. And unlike the latter—or at least the latter without a more fully fleshed out theory of user choice—it defends actual users' choices when they are under siege: malware installed without notice obviously fails.

Unfortunately, Click to Agree still falls short as a complete theory of software conflicts. It generates answers, but often those answers are wrong. The manifestations of assent on which these "agreements" rest are fundamentally fictional.[76] Terms of service, EULAs, privacy policies, and other such documents are unintelligible behemoths that no human would, should, or could read.[77] The PDF version of the macOS Mojave SLA is 15 pages and contains over 9,000 words.[78] They are notoriously dry, and studies consistently find that extremely few people accurately understand what is in them.[79] Whatever this is, it is not actual consent.[80]

[75] Grimmelmann, *supra* note 72, at 1501-02, 1521.

[76] *See generally* NANCY KIM, WRAP CONTRACTS (2013); MARGARET JANE RADIN, BOILERPLATE (2013).

[77] *See, e.g.*, Uri Benoliel & Shmuel I. Becher, *The Duty to Read the Unreadable*, 60 B.C. L. REV. 2255 (2019).

[78] APPLE INC., SOFTWARE LICENSE AGREEMENT FOR MACOS MOJAVE, https://www.apple.com/legal/sla/docs/macOS1014.pdf. The Central Pacific Railroad Photographic History Museum's User Agreement is a mind-boggling 35,833 words. *Central Pacific Railroad Photographic History Museum: User Agreement,* CPRR.ORG, http://cprr.org/Museum/legal.html [https://perma.cc/PT8V-BCJL].

[79] *E.g.*, Ewa Luger et al., *Consent for All: Revealing the Hidden Complexity of Terms and Conditions*, *in* CHI2013: CHANGING PERSPECTIVES CONFERENCE PROCEEDINGS 2687 (2013).

[80] *E.g.*, Neil M. Richards & Woodrow Hartzog, *The Pathologies of Digital Consent*, 96 WASH. U. L. REV. 1461 (2019).

In the standard bilateral context—user versus company—these concerns have been largely brushed aside. Most often, a user attempts to bring a class action lawsuit against a company for defective software in the face of an arbitration clause; most often, the court upholds the clause against a user who could have read it, even though they did not. Even if these agreements cannot be defended *as contracts*, there is nonetheless a defense of them *as policy*. This argument, frequently associated with Judge Easterbrook,[81] points to the economic efficiencies of mass-market contracting for software and other digital products. It frankly accepts that the user "choice" for any particular term is fictional, and then accepts the fiction as a useful way of achieving a reasonable policy outcome.[82]

But the fiction collapses in the software-versus-software context, where a user is caught in the crossfire between two dueling software makers. The policy goal of enabling mass-market digital contracting no longer does outcome-determinative work, and thus unmoored, the fiction of consent floats downstream like the barge Anna C, wreaking havoc as it goes.[83] The formal test for binding "agreement" bears so little relationship to actual user choice that it produces results that are essentially arbitrary.

For one thing, *both* dueling programs can meet the Click to Agree standard. Apple's macOS installer has a gold-standard explicit clickthrough. So does Zoom's installer. Apple reserves the right to install updates automatically; Zoom limits its liability to the maximum extent allowed by law. As far as Apple and Zoom are concerned,

[81] ProCD, Inc. v. Zeidenberg, 86 F. 3d 1447 (7th Cir. 1996) (Easterbrook, J.); Hill v. Gateway 2000, Inc., 105 F.3d 1147 (7th Cir. 1997) (Easterbrook, J.).

[82] Interestingly, survey research indicates that laypeople generally have the intuition that the "fine print" is legally binding even when it is not (for example, when a contract was induced by fraud), Meirav Furth-Matzkin & Roseanna Sommers, *Consumer Psychology and the Problem of Fine Print Fraud*, 72 STAN. L. REV. (forthcoming 2020), and that they regard even legally defective "consent" as actual consent, Roseanna Sommers, *Commonsense Consent*, 129 YALE L.J. (forthcoming 2020). Follow-up questions show that they are capable of distinguishing between these borderline cases and paradigm cases of full and unambiguous consent. In other words, actual users approach boilerplate terms of service with more nuance than either courts or their critics.

[83] *See generally* United States v. Carroll Towing Co., 158 F.2d 169 (2d Cir. 1947).

neither of them is responsible for any damage resulting from their struggle. User "choice" here is a choice for deregulation *à la* Software Freedom, which is to say no choice at all.

Apple and Zoom are both arguably Good software. But Bad software can meet the Click to Agree Standard, too. The FriendsGreeting virus, which emailed itself to everyone in an infected user's address book, protected its author by making users agree to a EULA.[84] This too was gold-standard consent: it used the standard Windows installer and the text of the EULA accurately described what it would do. Granted, sending an inane "greeting" to all of your contacts is something few rational email users would voluntarily do (that's what Facebook is for). But on the formalistic view online contracting law takes of consent, that is irrelevant. Clicking *is* agreement. If the FriendsGreeting EULA is valid, anything can be: Bitcoin ransomware, spousal spyware, or a virus that makes your iPod only play Jethro Tull. These ought to be easy cases, and yet they are not.

In fact, they are difficult cases for all three theories. They are difficult for Bad Software Is Bad, which cannot by itself explain why software that is harmful to many users is nonetheless appropriate for others who want to run it. They are difficult for Software Freedom, which cannot by itself explain why software some users want to run is nonetheless inappropriate for others. And they are difficult for Click to Agree, which cannot by itself explain which users truly want to run software when they click a button, and which do not. It is time to take a step back and ask why all three theories get into similar trouble.

III. User Autonomy

[84] Ed Felten, *Virus with a EULA,* FREEDOM TO TINKER (Nov. 15, 2002), https://freedom-to-tinker.com/2002/11/15/virus-eula/ [https://perma.cc/TF9E-36BN]; Robert Lemos, *Greeting Card Virus Licensed to Spread*, CNET NEWS.COM (Nov. 13, 2002, 4:00 AM), https://web.archive.org/web/20030207164353/http://news.com.com/2100-1001-965570.htm; David Mikkelson, *FriendsGreeting.com Virus*, SNOPES, https://www.snopes.com/fact-check/friendgreetingscom/ [https://perma.cc/R3N5-4A8W] (last updated Jan. 27, 2008).

A theory of software law is also a theory of users.[85] To say that a program should be allowed is to say that users should be allowed to run it; to prohibit a program is to prohibit them from running it. So to understand a theory of software law, we must understand how that theory thinks about users.

a. Imagined Users

Bad Software Is Bad's imagined user is a passive consumer. Their welfare matters: users should be able to enjoy Good software and be protected from Bad software. But they can be limited to ordering off an approved menu of welfare-enhancing Good software. It is fine to treat users identically and make choices for them, because if they are allowed to choose software for themselves, some of them will make bad choices. In other words, a user's freedom can be limited, even quite sharply, to limit harm to them and to others.

Software Freedom's imagined user is a romantic hacker.[86] They are fully informed about all the software installed on their computer and all the software they are considering installing. They understand the interactions among programs well enough that even if they cannot predict what the programs will do under all conditions, they can at least make knowing choices in view of the possible consequences. In other words, they are a technically skilled user who makes rational decisions about how best to achieve their goals using software.

There is something to both of these theories. Some users, some of the time, are passive consumers. They want to watch Netflix and play Call of Duty; they want one-click video calls; they want computers that work without effort or frustration. And some users, some of the time,

[85] *Cf.* Paul Ohm, *The Myth of the Superuser: Fear, Risk, and Harm Online*, 41 U.C. DAVIS L. REV. 1327 (2008); Julie E. Cohen, *The Place of the User in Copyright Law*, 74 FORDHAM L. REV. 347 (2007).

[86] This term, of course, is a reference to the figure of the romantic author in copyright scholarship. *See, e.g.*, JAMES BOYLE, SHAMANS, SOFTWARE, AND SPLEENS (1997); Peter Jaszi, *Toward a Theory of Copyright: The Metamorphoses of 'Authorship'*, 1991 DUKE L.J. 455; Martha Woodmansee, *The Genius and the Copyright: Economic and Legal Conditions of the Emergence of the 'Author'*, 17 EIGHTEENTH-CENTURY STUD. 425 (1984).

are active tinkerers. They want to build their own PCs from parts, play Civilization in an Excel spreadsheet, stream music seamlessly between "incompatible" devices, and even occasionally to write their own printer drivers. Many users are a bit of both, depending on what they want to do at any given moment and what tools are available. Bad Software Is Bad and Software Freedom are heuristics for these two latent tendencies in every user; they are dueling canons of construction. Bad Software Is Bad is maximally majoritarian; Software Freedom is maximally individualistic. Neither of them is complete without the other, or without a theory to decide which of them applies, to explain when a user is trying to customize their computer and when they just want the factory default to work.[87]

Click to Agree tries to fill that void. Its underlying theory of the user is a close cousin of the theory behind Software Freedom. Both of them defer to individual user choices in the name of promoting user autonomy. Both of them see users as romantic individuals who make informed and rational choices. There is a difference, however, and it is a telling one. The idealized user of Software Freedom exercises autonomy through *actions*; they install specific software to achieve their goals. But the idealized user of Click to Agree exercises autonomy through *agreements*; they enter into bargains to achieve their goals. The user of Software Freedom is *technically* sophisticated; they understand software and its consequences. The user of Click to Agree is *legally* sophisticated; they understand contract law and its consequences. The user of Software Freedom is maximally free in the moment: they always have the option to rip out their current software and replace it with something different and better. But the user of Click to Agree plays a longer game: to gain the things they value most, they are willing to bind themself to do certain things and refrain from doing others. The difference between these two theories is the difference between two libertarian ideals: the freedom to use one's property and the freedom to enter into enforceable contracts.

[87] There is another relevant line, discussed above: the choice between individual freedom and third-party protection. *See supra* Part II.B. For now, let us regard this line as externally given or determined. We are interested in how to determine, within the category of software users are *allowed* to run, which software they *want* to run.

The difference is perhaps most clearly visible in their attitudes towards DRM. The ideal user of Software Freedom can rationally choose to install DRM-protected media and then install DRM-breaking software to decrypt it. But the ideal user of Click to Agree who installs DRM-protected media and clicks to agree to its accompanying license is no longer free to install and run the DRM breaker. They have rationally traded away their freedom to do so in order to obtain the media in the first place. Having made that choice, they are committed to it.

To sum up, these three theories are theories of the user in three different senses. Bad Software Is Bad is a theory of public policy: it describes what software would be best (for society) for users to have. Software Freedom is a theory of individual rights: it describes what users should have the freedom to do. And Click to Agree is a theory of consent: it describes what users have and have not agreed to.

All three theories fail on their own terms: they provide implausible answers to questions within their domains. Bad Software Is Bad prohibits users from choosing to run software that it would be good policy to let them run. Software Freedom allows users the freedom to run software that makes their computers unusable. And Click to Agree often claims that users have made choices about software that they did not in fact make and would not willingly make.

These theories all fail for the same reason: *they are all incomplete on their own*. Questions of software policy, user freedom, and user consent cannot be separated. There is no way to coherently describe what is good for users without taking account of what they want and what they choose, no way to describe what they want without taking account of what is good for them and what they choose, and no way to describe what they choose without taking account of what is good for them and what they want. To make sense of software conflicts, we need a genuine theory of user *autonomy*.[88]

[88] *See* ANGELA DALY, PRIVATE POWER, ONLINE INFORMATION FLOWS, AND EU LAW: MIND THE GAP (2016); Batya Friedman & Helen Nissenbaum, *Software Agents and User Autonomy*, *in*

b. A Theory of User Autonomy

What might a theory of user autonomy look like? Perhaps something like this:

Software helps users achieve important life goals. These include finding a job, finding a spouse, making deals, making art, learning about the world, learning about themselves, participating in democracy, participating in fan culture, being amused, being inspired, connecting with old friends, and making new ones. Anything that matters to people, no matter how big or how small, they now do in part with software. A program is better when it tends to advance these goals, and worse when it tends to thwart them. Some uses of software—such as search—can promote a wide range of individually chosen goals. Other uses of software—such as intensive surveillance—start off in significant tension with user goals.

Software can be helpful (or harmful) both directly and indirectly. Some software, like an email program or a drawing app, directly helps users achieve a goal (building meaningful relationships with others, or being creatively expressive). A program that plays a loud annoying noise is directly harmful. But much important software is helpful or harmful on a meta level. An operating system is helpful because it allows users to run other software like an email program or a drawing app, and a virus is harmful because it deletes emails and drawings.

Users by definition delegate particular tasks to software. I have elsewhere described this delegation in fiduciary terms: programs can act as agents and advisors for their users.[89] As such, users require both *access* to software and *loyalty* from their software. Access promotes

AGENTS '97: PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS 466 (1997); James Grimmelmann, *Speech Engines*, 98 MINN. L. REV. 868, 911-12 (2014); *see also* GERALD DWORKIN, THE THEORY AND PRACTICE OF AUTONOMY (1988); Sarah Buss & Andrea Westlund, *Personal Autonomy*, STAN. ENCYC. OF PHIL. (FEB. 15, 2018), https://plato.stanford.edu/entries/personal-autonomy/.

[89] Grimmelmann, *supra* note 88, at 904-06.

software in general; loyalty distinguishes good software from bad. The tradeoffs are important: access-promoting measures can be bad for users if they undercut loyalty, and vice-versa. In particular, users frequently delegate to software the task of keeping them safe from other software.

Users' goals are diverse and highly individualistic. Different users want different things—both big (professional versus personal) and small (DC versus Marvel). So we must distinguish between programs that are better for users *in general* and programs that are better suited *for a specific user*. Of course, the two are related. On the one hand, a program's overall general benefits are the aggregation of its benefits for specific users. On the other, if we know nothing more about a specific user, a program's overall benefits provide strong evidence of the best default assumption as to whether the program is good for them.

Different users' goals can conflict. Resolving these conflicts is a central task of politics and policy. Wherever the limits are drawn, users' autonomy is appropriately limited by other users'. This conflict cannot be evaded by saying that one particular person is "the user" of a program and their autonomy takes priority; any such claim requires a detailed articulation of why running the program falls within that person's sphere of autonomy and no one else's. For example, a program that runs only on a user's own physical computer and does not communicate with other computers is less likely to have consequential effects for others. Thus, personal property ownership can sometimes be a plausible proxy for whose autonomy is at stake.[90]

Users have different levels of technical skill. Even when pursuing the same primary goal, one user may prefer to rely on a program that does more of the work, and another to do more of the work herself. Some professional photographers do detailed retouching work in Adobe Photoshop; some amateurs click the "enhance" button in Apple

[90] *See generally* AARON PERZANOWSKI & JASON SCHULTZ, THE END OF OWNERSHIP: PERSONAL PROPERTY IN THE DIGITAL ECONOMY (2016).

Photos. Different software is better suited for users with different skills. This is an important axis for software conflicts in two ways. First, users can be more or less capable of understanding and managing the operations of complex programs with potentially serious and irreversible effects. More skilled users may not be able to achieve all of their goals as well if they are denied access to complex and dangerous programs—but less skilled users may be thwarted in their own goals if they are forced or nudged to use such programs rather than simpler alternatives. Second, users especially vary in their skill and interest in the domain of software conflicts itself: computer security. Some users benefit from manually managing multi-program interactions; some users benefit from delegating the details to software.

User choice is an important component of user autonomy, for three overlapping reasons. First, it is constitutive of autonomy that one's actions should be the result of one's own choices. It might be better for a user to stop using Facebook, but this is generally not a decision that their operating system should make for them. Second, an individual user's choices are strong evidence, often the best available, about their goals. The choice to install *Minecraft* rather than *Fortnite*, or vice versa, is not easily gainsaid on the basis that the user "really" prefers the other. And third, the aggregation of individual users' choices is relevant evidence about users' goals in general. Facebook, *Minecraft*, and *Fortnite* are all popular for a reason.

The allowable scope of user choice is appropriately circumscribed by any limits on user goals. This is an analytically distinct issue from the next one: accurately determining what a particular user has chosen within that allowable scope. A rule that users may not remove DRM from copyrighted works does not require the fiction that users "choose" not to remove it.

User choice is a question of fact: what actions by a program are within the scope of the user's consent?[91] Regardless of whether one thinks that for moral purposes consent is attitudinal or expressive,[92] for legal and policy purposes we should work with an authorization account.[93] I prefer Tom Dougherty's reliable evidence principle: "An action A falls within the scope of the consent that X gives to Y if and only if X is giving consent through behaviour B, and Y's reliable evidence sufficiently supports the interpretation that X is intending to communicate authorization for A in virtue of B."[94] This evidence includes any evidence about X's specific situation and communicative intentions, about consent-givers' situations and intentions in general, and any relevant evidence X could acquire through due diligence.[95]

Evaluated against this standard, *clickthrough agreements and other standard formal indicia of digital consent are typically* prima facie *valid*. The widespread use of such agreements establishes a general communicative norm that clicking constitutes agreement to whatever widely used terms are contained in them, even if the clicker has not inspected the terms and is unaware of the details.[96] The basic argument

[91] This is most usefully phrased as a question of the *scope* of the user's consent, rather than the presence or absence of consent. *See generally* Tom Dougherty, The Scope of Consent (unpublished book), https://sites.google.com/site/tomdoughertyphilosophy/.

[92] *See generally* PETER K. WESTEN, THE LOGIC OF CONSENT (2004).

[93] Dougherty, *supra* note 91, at 92–102; *see also* RUTH R. FADEN & TOM L. BEAUCHAMP, A HISTORY AND THEORY OF INFORMED CONSENT (1986) (discussing consent as authorization).

[94] Dougherty, *supra* note 91, at 114. As Dougherty notes, Y may still sometimes act blamelessly even if X is not intending to give consent through B. *Id.* at 114. n.82. This is an important qualification, because misleading evidence can sometimes suffice to shield Y from legal consequences. *See* Joseph Millum & Danielle Bromwich, *Understanding, Communication, and Consent*, 5 ERGO 45 (2018) (discussing scope of communicated consent as a process of communication).

[95] Dougherty, *supra* note 91, at 108–34 (discussing evidence available to consent-receiver).

[96] *But see* Robin Bradley Kar & Margaret Jane Radin, *Pseudo-Contract and Shared Meaning Analysis*, 132 HARV. L. REV. 1135 (2019). Kar and Radin argue that the communicative norms of cooperative contract formation show that the parties to boilerplate clickthrough agreements lack essential "shared meaning" as to many terms in those agreements. But after more than two decades of widespread usage of clickthrough agreements, it is equally plausible to say that they have their own communicative norms, precisely because there are such widespread practices of presenting them and clicking to agree, sight unseen. My best understanding of the *communicative* content of clicking to agree is that it expresses assent to any terms in the proffered instrument that are of a type users generally would expect to be in an instrument of

that voluntary agreements are generally autonomy-enhancing for users is sound.

That said*, the validity of a formal indication of digital consent is defeasible*. One possible reason is that a program's behavior falls outside of the range of typical behavior users are accustomed to from similar programs using similar agreements. Thus, even though FriendsGreetings used a standard clickthrough installer and described its behavior in the agreement, it probably obtained invalid consent. Its spamming behavior was sufficiently unusual that it did not fall within the scope of user expectations, and this gap would have been apparent to a reasonable software vendor. The general communicative norm typically relied on for clickthrough consent was not applicable.[97]

Other standard cases of invalid consent include incapacity, fraud, and duress.[98] Where the reliable evidence would tell a reasonable software vendor that the "consenting" user is a minor, the clickthrough is presumptively invalid. This is obviously the case for many games targeted at children; many app stores have revised their in-app purchase options to create a clickthrough process that more reliably targets an adult owner of the account rather than the minor user.[99] Similarly, where the vendor has elsewhere misrepresented what its program does, or its interface deliberately conceals aspects of its functionality that would be material to a reasonable user, the clickthrough is presumptively invalid—even if the clickthrough itself is scrupulously accurate. The FTC gets this right in some of its enforcement actions against spyware: formal "consent" is ineffective

this sort. One who clicks to agree may accepts arbitration and give the company a nonexclusive copyright license in any user uploads, for example—but not accept unusual and oppressive terms, like giving a company custody of one's firstborn.

[97] *See also* Nathaniel Good et al., *Stopping Spyware at the Gate: A User Study of Privacy, Notice and Spyware*, *in* SOUPS 2005: PROCEEDINGS OF THE SYMP. ON USABLE PRIVACY AND SECURITY 43 (2005).

[98] *See* WESTEN, *supra* note 92.

[99] *See, e.g.*, *Prevent In-App Purchases from the App Store*, APPLE (Nov. 23, 2019), https://support.apple.com/en-au/HT204396.

when a program conceals from users what it will do to them.[100] And where a vendor has reliable evidence that the clickthrough has not been voluntarily given—which may be the case for many spousal spyware apps—here too user consent is presumptively invalid.[101] Note that in all such cases, other programs to protect users from these abusive consent processes are more likely to be acting consistently with user choice.

General user goals are interpretively relevant to understanding the scope of a specific user's consent to specific program behavior. Intrusive surveillance tools, for example, are significantly harmful to many users who run them, and helpful to very few. This provides strong probative evidence that users have not factually given substantive consent, even though they may have gone through a formal mechanism purporting to indicate that they have. Formal consent mechanisms have evidentiary, cautionary, and channeling functions to make indications of consent more reliable, but they must never be mistaken for consent itself.[102] No legal formality can provide conclusive evidence on its own; fraud in the factum can never be completely discounted.

Where one program interferes with another, the user goals that both programs serve or hinder are interpretively relevant. Bitcoin ransomware is nearly universally harmful, and thus nearly universally unwanted. (There are exceptions, like anti-malware researchers, whose goals are rare and unusual: to understand how the ransomware works so they can develop countermeasures.) This is so because the programs the ransomware interferes with are significantly useful for a wide range of important user goals. And in comparison with a normal file encryption utility, which can protect a user's greatly valued privacy, it does not advance any user goals to be able to decrypt one's files only on payment of a Bitcoin-denominated ransom.

[100] *See, e.g.*, Press Release, F.T.C., FTC Brings First Case Against Developers of "Stalking" Apps, FTC (Oct. 22, 2019), https://www.ftc.gov/news-events/press-releases/2019/10/ftc-brings-first-case-against-developers-stalking-apps.

[101] *See, e.g.*, *id.*

[102] *See* Lon L. Fuller, *Consideration and Form*, 41 COLUM. L. REV. 799 (1941).

Different consent processes produce different levels of evidence about user choices. The clickthrough is widely used for software installed on users' computers by businesses, and for online services that require the creation of user accounts. But it is not the only option, and hardly the only one in widespread use. There are lighter-weight processes, like free-software licenses that grant unilateral permission subject to restrictions and do not require any specific act of user assent,[103] and websites that assume user permission to set cookies and run in-browser scripts from the simple act of navigating to them. There are also heavier-weight processes, like additional clickthrough screens and checkboxes for important terms, interfaces that ask "Are you sure?" before taking significant actions, cooling-off periods, and even (gasp) requirements that users who want to agree must confirm their intentions in person, in the physical world.[104]

Many expressions of user authorization are implicit: they are established by the actions a user takes, rather than by an explicit act of purely legal agreement. This is a normal feature of software—most uses of which would be utterly impossible without implicit prospective blanket consent. Good user interfaces are carefully designed so that users can predict the consequences of their actions; the "save" button saves and the "delete" button deletes. When we say that a user has "chosen" one of these consequences, we are really making a complex contextual judgment that the consequence is within the scope of the authorization the user gave through a long sequence of program actions—click here, scroll there, type this, click again, etc. The EU "cookie directive"[105] (which is responsible for websites showing pop-ups asking for user consent to use cookies) is silly because in the Year of Our Lord Two Thousand and Twenty cookies are everywhere and

[103] *See, e.g.*, Free Software Foundation, *GNU General Public License 3.0,* GNU OPERATING SYS. (June 29, 2007), https://www.gnu.org/licenses/gpl-3.0.en.html.

[104] On the interplay between technical design, notice, consent, and legal consequences, see generally WOODROW HARTZOG, PRIVACY'S BLUEPRINT: THE BATTLE TO CONTROL THE DESIGN OF NEW TECHNOLOGIES (2018).

[105] Council Directive 09/136, 2009 O.J. (L 337) 11.

have been for nearly 25 years. Consent to set cookies is implied from visiting a website using a browser.

The converse is also true. *Denials of authorization can also be implicit*. When a browser or browser plugin blocks cookies, that is an implicit denial of permission to set cookies, and a website that circumvents this denial to track the user is acting without user consent, regardless of what its cookie directive popup says.[106] Anti-adblocking popups, which prompt the user to disable their adblocker before the website will display its content, get this right: they ask the user to take a specific action—disabling their adblocker—that meaningfully betokens consent.

This variation in user-consent mechanisms would be unnecessary unless there were a variation in the severity of program actions. *More unusual and more potentially dangerous program actions require more specific user consent; less unusual and less potentially dangerous program actions require less specific user consent*. Match-three games and online banking apps need different levels of user consent. The same consent process that suffices to run an antivirus program is insufficient to run a virus. The question of whether a user has consented to a specific program action thus depends both on the action and on the consent mechanism, both on what users in general want, and on what this specific user has chosen.[107]

In particular, *the appropriate level of required consent typically increases with increased complexity and power*. This facilitates user sorting based on technical skill and confidence. Command-line tools, programming development environments, and bulk-erase disk utilities are appropriate for experienced users and dangerous in the hands of inexperienced ones. Certifying oneself as an experienced user by installing them is sometimes an implicit consent mechanism: it may require a degree of skill and knowledge simply to make them work.

[106] *See* Mayer, *supra* note 23.

[107] For examples of programs that are "consensually" installed but may fail this test, see Nathaniel Good et al., *User Choices and Regret: Understanding Users' Decision Process about Consensually Acquired Spyware*, 2 I/S: J. L. POL'Y FOR INFO. SOC'Y 283 (2006).

But once one has, there will be fewer checks to make sure that one really intends the result. (This is part of what makes scams that convince users to install and use such programs and operate them for the scammer's benefit so insidious.) Jonathan Zittrain proposed dividing computers into a "green" portion with strong technical safeguards and a "red" portion where users could engage in riskier tinkering.[108] The red/green distinction by itself is too simple to capture all of the nuances of software conflicts, but the core idea is sound: red programs and green programs need different consent processes.

c. Freedom to Tinker

This is, I admit, a complicated theory of users. But any less complicated theory will not work. We need all three heuristics, because each sees something the others do not. Bad Software Is Bad understands that software in general can be helpful or harmful; Software Freedom understands that users have individual needs and goals; Click to Agree understands that they make choices and commitments. Each heuristic informs the others.

The picture of user autonomy that emerges is an appealing one. There are nice hints of it in another popular slogan about software users: the "freedom to tinker," which Ed Felten defined as "your freedom to understand, discuss, repair, and modify the technological devices you own."[109] One way to understand the freedom to tinker is as a negative-liberty synonym for Software Freedom: it prizes users' freedom over third-party restrictions. But it also provides a positive vision of user

[108] JONATHAN ZITTRAIN, THE FUTURE OF THE INTERNET—AND HOW TO STOP IT 155 (2008). Granular app permissions are a modern expression of this idea: they put speed bumps in the way of more powerful apps, so that it is easier to run a safer program. *See* Apple Developer Documentation, Protected Resources, https://developer.apple.com/documentation/bundleresources/information_property_list/protected_resources.

[109] Edward Felten, *The New Freedom to Tinker Movement*, FREEDOM TO TINKER (Mar. 21, 2013), https://freedom-to-tinker.com/blog/felten/the-new-freedom-to-tinker-movement/. *See generally* Pamela Samuelson, *Freedom to Tinker*, 17 THEORETICAL INQUIRIES IN L. 563 (2016) (giving an eight-part taxonomy of the freedom to tinker, including both autonomy and liberty interests).

autonomy.[110] First, it is explicitly justified as a user-autonomy project. Tinkering is essential for users to achieve important goals in a self-chosen way. Second, it is attractively compatible with user diversity. "Tinkering" is at a relatively advanced level of Maslow's hierarchy of computer needs. Not all users will tinker, but all of them should be free to. And third, the reasons commonly cited for promoting tinkering prominently include learning, research, and improvement of one's own skills.[111] The freedom to tinker is a way of helping novice users become more advanced ones. There is a feedback loop here: use improves competence, but competence is important to safe use. In this respect, the freedom to tinker embraces the intentional self-development at the heart of a rich theory of autonomy. So if one is looking for a slogan to describe the vision of the theory of the user I have sketched, one could do worse than "Freedom to Tinker."

d. Apple and Zoom

Return now to where we started, with Apple and Zoom. What do these various theories have to say about the situation?

Bad Software Is Bad is of at least two minds. Operating systems are useful; so is videoconferencing software. macOS has helped millions of users accomplish something they want to do; so has Zoom. Single-click calling is a useful convenience. So is having a computer that is secure against hackers and spies. Zoom balanced those equities in favor of single-click calling; Apple balanced them in favor of security. Neither is obviously wrong, but the fact that Apple's upgrade secured users' computers against *other* malware probably cuts in favor of Apple. (Note that this take does not really respond to users who genuinely preferred one-click calling and were disappointed to lose it.)

[110] In fairness, much of this positive normative vision is implicit in arguments for software freedom. One advantage of the "freedom to tinker" framing is that it makes this vision more explicit.

[111] *See* Andrea M. Matwyshyn, *Generation C: Childhood, Code, and Creativity*, 87 NOTRE DAME L. REV. 1979 (2012); Samuelson, *supra* note 109.

Software Freedom, on the other hand, probably favors Zoom. Apple downgraded users' computers and locked down macOS to remove features. Some authors call this kind of remote control "tethering,"[112] but this was more like a choke chain: Apple yanked it and cut off a feature that many users found useful. Users who preferred the convenience of one-click calling should have been allowed to keep using it. (Note that this take does not really respond to users who don't care about the extra click but do care about not having their computers hijacked.)

As for Click to Agree, Apple's Software License Agreement for macOS 10.14 Mojave states:

> Q. Automatic Updates. The Apple Software will periodically check with Apple for updates to the Apple Software. If an update is available, the update may automatically download and install onto your computer and, if applicable, your peripheral devices. By using the Apple Software, you agree that Apple may download and install automatic updates onto your computer and your peripheral devices.[113]

The conclusion here is straightforward: Apple had clear and unambiguous permission from macOS users to install its Zoom-server-disabling update. But by this reasoning, Apple could have uninstalled Zoom entirely, server and all, or uninstalled it even if it had no security vulnerabilities at all, just because it competes with Apple's own FaceTime—which suggests that Click to Agree is not really sensitive to the factors that make this a difficult case in the first place.

By contrast, here is a sketch of the facts that are relevant to the Apple-Zoom incident if we take the Freedom to Tinker view of user

[112] *E.g.*, Chris Jay Hoofnagle, Aniket Kesari, & Aaron Perzanowski, *The Tethered Economy*, 87 GEO. WASH. L. REV. 783 (2019); ZITTRAIN, *supra* note 108, at 101–10.

[113] APPLE INC., *supra* note 79. By way of comparison, the *Zoom Terms of Service*, ZOOM (May 30, 2019), https://zoom.us/terms [https://perma.cc/B98C-EV3K], are drafted to restrict how users use the Zoom service, and have little to say about what users will do with the Zoom software.

autonomy—including a few details I have sneakily held back until now. First, one-click calling is a useful convenience. Second, one-click calling into a call you didn't plan to join is an unsettling prospect. Third, macOS applications do not typically install web servers; this is a very unusual feature. Fourth, users really, really, really don't like having their computers hijacked. Fifth, Zoom did not describe the server or disclose the risks it presented, even in very general terms. Sixth—and this is new—macOS users can disable silent updates by unchecking a box in a system preference tab. Seventh—and this is also new—it is not hard to run a web server on a macOS computer. Apple sells macOS Server for $19.99,[114] or you can install one of your own. (I am running one as I type this.)

Put all of this together and the portrait that emerges of Apple's view of macOS users looks something like this: Most users want to run applications like Zoom. They can. Users like the convenience of one-click calling, but the security risks from running a secret web server are serious, and most users would give up the convenience if they were aware of the tradeoff. Most users were unaware that Zoom installed a secret insecure server, and so were unlikely to remove it on their own. Many users who are prompted to install an update delay it, sometimes for years. Some users run web servers on their Macs, but extremely few users (if any) use the Zoom server for anything besides Zoom. Some users are very concerned about closely inspecting any software updates. So: a silent background update patches most users' computers in the way they would want if they were fully informed and given an explicit choice, without breaking any features explicitly relied on by users who run servers, and while giving users who don't want forced downgrades a way to opt out of this one.

All in all, this is a nuanced and respectful approach toward users. There is always a tradeoff between beneficence and respect, and Apple's Zoom update handles that tradeoff in a thoughtful and defensible way. The only things missing are that Apple could explain

[114] *See macOS Server*, APPLE, https://www.apple.com/macos/server/ (last visited Feb. 5, 2020).

this reasoning publicly in more detail, that it could give users more information up front about the option to disable automatic macOS security updates, and that it could provide users the ability to roll back unwanted updates after the fact.

This approach shows how Freedom to Tinker both draws on and goes beyond the other three theories of software conflicts. Bad Software Is Bad explains why Zoom's secret server is a danger that most users should be protected from. Software Freedom gives users who like the server the choice to keep it. Click to Agree provides a legal shield for the update. But Freedom to Tinker goes further in its willingness to look at users as actual people: people with differing goals and abilities, people who sometimes make choices and sometimes don't, people who both rely on software to help them achieve their goals and need to be protected from software.

This is the real point of the Freedom to Tinker user-autonomy theory of software conflicts. It is not something new and radical. Rather, it is a restatement—explicitly and in one place—of many widely shared assumptions and commitments about users and software. The everyday practices of the computer security and human-computing interface communities already reflect something like this understanding of users and the goals of responsible software. Something like Apple's disabling of Zoom's server—or any of a dozen other examples given above—is intelligible only against this ethical backdrop.

IV. Conclusion

I have focused on software conflicts for three reasons. First, they are surprisingly common.[115] Looking at them together as a category yields insights that looking at them individually does not. Second, they call into question common legal heuristics for dealing with software.[116] These heuristics are problematic even in less controversial settings, but there it is easier to brush aside their shortcomings. Software conflicts

[115] *See supra* Part I.
[116] *See supra* Part II.

provide a setting where these failings are harder to ignore, because the heuristics give such obviously wrong answers. And third, software conflicts direct attention where it belongs: to user autonomy.[117]

Software-versus-software conflicts cannot be coherently resolved without a good theory of user autonomy. Any theory of user autonomy hinges on a good theory of user consent. And any theory of user consent hinges on a factually and normatively rich understanding of what users do with computers and what they are trying to do. A legal realist might say that an informal and generally subconscious lay theory of user autonomy is actually doing most of the work in software-conflict cases already, so legal theorists should make that theory explicit, test its claims, and fill in its details.

Consent is complex because life is complex. Software is complex because life is complex. Software law cannot escape being complex as well, because it cannot escape paying attention to the nuances of what users consent to.

[117] *See supra* Part III.