The Ohio State Technology Law Journal

SEEKING THE SOURCE: CRIMINAL DEFENDANTS' CONSTITUTIONAL RIGHT TO SOURCE CODE

STEVEN M. BELLOVIN,[1] MATT BLAZE,[2] SUSAN LANDAU,[3] & BRIAN OWSLEY[4]

The right to a fair trial is fundamental to American jurisprudence. The Fifth Amendment of the Bill of Rights guarantees "due process," while the Sixth provides the accused with the right to be "confronted with the witnesses against him." But "time works changes, brings into existence new conditions and purposes." So it is with software. From the smartphones we access multiple times a day to more exotic tools—the software "genies" of Amazon Echo and Google Home—software is increasingly embedded in day-to-day life. It does glorious things, such as flying planes and creating CAT scans, but it also has problems: software errors.

Software has also found its way into trials. Software's errors have meant that defendants are often denied their fundamental rights. In this Article, we focus on "evidentiary software"—computer software used for producing evidence—that is routinely introduced in modern courtrooms. Whether from breathalyzers, computer forensic analysis, data taps, or even FitBits, computer code increasingly provides crucial

---

[1] Percy K. and Vida L.W. Hudson Professor of Computer Science and affiliate law faculty at Columbia University.
[2] Robert L. McDevitt, K.S.G., K.C.H.S. and Catherine H. McDevitt L.C.H.S. Professor of Law and Computer Science at Georgetown University.
[3] Bridge Professor in Cyber Security and Policy at The Fletcher School and the School of Engineering, Department of Computer Science, Tufts University.
[4] Assistant Professor of Law at University of North Texas Dallas College of Law. From 2005 until 2013, Brian Owsley served as a United States Magistrate Judge for the United States District Court for the Southern District of Texas.

trial evidence. Yet despite the central role software plays in convictions, computer code is often unavailable to examination by the defense. This may be for proprietary reasons—the vendor wishes to protect its confidential software—or it may result from a decision by the government to withhold the code for security reasons. Because computer software is far from infallible—software programs can create incorrect information, erase details, vary data depending on when and how they are accessed—or fail in a myriad of other ways—the only way that the accused can properly and fully defend himself is to have an ability to access the software that produced the evidence. Yet often the defendants are denied such critical access.

In this Article, we do an in-depth examination of the problem. Then, providing a variety of examples of software failure and discussing the limitations of technologists' ability to prove software programs correct, we suggest potential processes for disclosing software that enable fair trials while nonetheless preventing wide release of the code.

CONTENTS

I. Introduction

The right to a fair trial is fundamental to a democracy. In the United States, this right is guaranteed pursuant to the Fifth Amendment, which ensures "due process,"[5] and the Sixth Amendment, which provides the accused with the right "to be confronted with the witnesses against him."[6] These legal rights are a hallmark of the judicial process in the United States. Technology has transformed this, threatening the very

[5] U.S. CONST. amend. V.
[6] U.S. CONST. amend. VI.

right to a fair trial. The problem arises from the change in what constitutes a witness. At the time the Bill of Rights was written, witnesses were people, and it was well understood what it meant "to be confronted with the witness."[7] But times have changed.

Today we use complex software tools to measure certain types of activity, such as the level of blood alcohol a person has or how fast a car is being driven. We then use the results produced by these tools to provide evidence in court. The Federal Rules of Evidence enable defendants to question expert witnesses who present testimony on the validity of the evidence.[8] However, these protections do not extend to "questioning" computer code. Through three largely unrelated developments, these essential protections are in danger of being sharply curtailed, and therein lies a serious problem.

The first of the developments is the rise of "evidentiary software," that is, software whose output is itself evidence.[9] This has happened despite the prevalence of errors in large software systems, a concern well known to the computer science community but much less so to the public. The ubiquity of software errors is the second issue, since citizens may be convicted on the basis of what amounts to a witness bearing false testimony. The third arises from the increase of proprietary software; that and bureaucratic opacity have made it difficult for defendants to examine this software.[10] The result is a miscarriage of justice: conviction as a result of evidence that the defendants *are not able to cross-examine* and which, in fact, may be in error. Lest we lose the protections of the Fifth and Sixth Amendments, here we examine the confluence of these three issues.

---

[7] Interactive Constitution, *Sixth Amendment: Right to Speedy Trial by Jury, Witnesses, Counsel*, NAT'L CONST. CTR., https://constitutioncenter.org/interactive-constitution/amendment/amendment-vi [https://perma.cc/LAS7-2HPZ].
[8] FED. R. EVID. 702.
[9] While there are many examples, the simplest is the breathalyzer: complex software analyzes the breath sample for its alcohol content. *See, e.g.*, SEAN E. GOODISON ET AL., RAND CORPORATION, DIGITAL EVIDENCE AND THE U.S. CRIMINAL JUSTICE SYSTEM: IDENTIFYING TECHNOLOGY AND OTHER NEEDS TO MORE EFFECTIVELY ACQUIRE AND UTILIZE DIGITAL EVIDENCE (2015), https://www.ncjrs.gov/pdffiles1/nij/grants/248770.pdf.
[10] *See generally* Stephen W. Smith, *Policing Hoover's Ghost: The Privilege for Law Enforcement Techniques*, 54 AM. CRIM. L. REV. 233 (2017).

Others have discussed concerns about proprietary code[11] and the inability of defendants in civil and criminal cases to access the full evidence against them. While our fundamental concern is about fairness and upholding the rights pursuant to the Fifth and Sixth Amendments, our focus is somewhat different. We are concerned that the evidence is inaccurate—and because the defendants cannot access the underlying code, there is no way for the defendants to discover the falsehood of the "evidence" against them. What this means is that, although case law has moved to enable a defendant to cross-examine one who bears witness against him, the failure of the courts to understand the vagaries of evidentiary software and the great chance of errors in the evidence means that the right often fails in practice.

We will start our explanation of these issues with a simple example: the Breathalyzer.[12] The scientific principle behind these devices is straightforward: when someone drinks, alcohol is not digested, but is simply absorbed into the bloodstream where it does not undergo any chemical changes. When alcohol-laden blood reaches the lungs, the alcohol diffuses into the lungs proportionally to the amount in the blood. Thus, alcohol in exhaled breath provides a way to measure the level of alcohol in the blood.[13] The rest is elementary chemistry.

---

[11] *See generally* Rebecca Wexler, *Life, Liberty, and Trade Secrets: Intellectual Property in the Criminal Justice System*, 70 STAN. L. REV. 1343 (2018).

[12] "Breathalyzer" is a live trademark, serial number 72028025. That said, the word is often used generically; *see, e.g.*, Kashmir Hill, *Imagine Being on Trial. With Exonerating Evidence Trapped on Your Phone.*, N.Y. TIMES (Nov. 22, 2019), https://www.nytimes.com/2019/11/22/business/law-enforcement-public-defender-technology-gap.html [https://perma.cc/4TK5-9L8K] ("Law enforcement agencies get a new investigative technique — fingerprinting, DNA analysis, breathalyzer tests — and those representing the accused struggle to play catch-up.").

[13] Certain situations can change the validity of the reading, including whether the person being tested is diabetic or has been fasting—this may produce acetone, which is measured as alcohol—or whether they have recently used a mouth freshener, which may contain alcohol. Dentures can also trap alcohol in the mouth. *See* MURDO BLACK, ALCOLIZER TECHNOLOGY WHITE PAPER SERIES, SUBSTANCES THAT CAN AFFECT A BREATH TEST 3, https://www.alcolizer.com/wp-content/uploads/2017/01/12660_WhitePaper_Substances-that-Can-Affect-A-Breath-test_LR.pdf.

The explanation that follows is taken from the Breathalyzer's[14] inventor, Robert Borkenstein, once a member of the Indiana State Police and later a professor of forensic studies at Indiana University. The device shuttles a person's breath down a tube, eliminating the first portion of the breath. The breath is then combined with a mixture of sulfuric acid, potassium dichromate, silver nitrate, and water. The silver nitrate acts as a catalyst to speed up the chemical reaction transforming alcohol, potassium dichromate, and sulfuric acid into potassium sulfate, chromium sulfate, acetic acid, and water. The potassium dichromate and potassium sulfate are the chemicals to observe. Potassium dichromate is reddish-orange, while potassium sulfate is green. A photocell measures the decrease in yellow light, thus revealing the amount of alcohol present in the sample.[15] These systems are called "fuel-cell" breathalyzers.

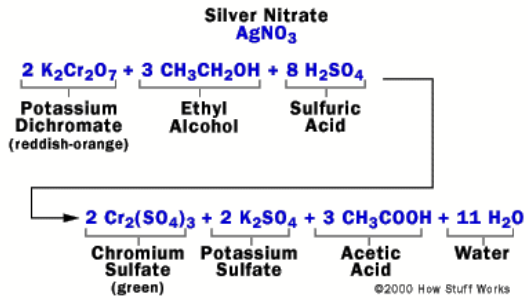

Diagram taken from *How Stuff Works*.[16]

Although this chemistry is simple, chemistry itself is messy; it is much easier if one can dispense with "wet" devices. The development of infrared instruments for testing breath alcohol levels did so. The principle is again straightforward:

---

[14] "Breathalyzer" refers to Borkenstein's invention. When used with a small "b," the word denotes devices that chemically ascertain breath-alcohol levels from a person's breath. *See* A.W. Jones, *Physiological Aspects of Breath-Alcohol Measurement*, 6 ALCOHOL, DRUGS & DRIVING 1, 2 (Apr.-June 1990).

[15] R.F. Borkenstein & H.W. Smith, *The Breathalyzer and Its Applications*, 2 MED., SCI. & L., 13, 14 (1961).

[16] Craig Freudenrich, *How Breathalyzers Work*, HOW STUFF WORKS (Oct. 20, 2000), https://electronics.howstuffworks.com/gadgets/automotive/breathalyzer.htm [https://perma.cc/4KWT-DS27].

> Molecules absorb electromagnetic radiation at certain specific, unique wavelengths. Thus, it may be said that each molecule has its own 'infrared fingerprint.' Ethyl alcohol absorbs radiation at wavelengths of approximately 3.00, 3.39, 7.25, 9.18, 9.50, and 11.5 microns. No other compound absorbs radiation at all of those wave-lengths exclusively.[17]

Infrared and fuel-cell breathalyzers, which measure the current through the breath, are the most common types of breathalyzers.[18]

In order to accept the results of a breath-testing device, the judge and jury need to answer two questions: How does the device work? Does the device report alcohol breath levels accurately? The Breathalyzer was preceded by a simpler tool, the Drunkometer,[19] that operated on the same principles, and in 1955 Edwin Conrad, a senior attorney at the Federal Communications Commission, looked at how well evidence from various tools—radar guns, lie detectors, and Drunkometers—was being accepted by the courts.[20] He concluded that "scientific instrumentality of proof will not be accepted in evidence unless it has gained sufficient standing and scientific recognition, and has been demonstrated to be dependable."[21] Accepting the evidence required that it be:

> proved that qualified chemists made the test; the underlying theory and operation of the Drunkometer be explained in great detail; the accuracy of the machine in particular, and the accuracy of the method in general,

---

[17] AM. PROSECUTORS RSCH INST., BREATH TESTING FOR PROSECUTORS: TARGETING HARDCORE IMPAIRED DRIVERS 11 (2004), https://cdn.ymaws.com/mcaa-mn.org/resource/resmgr/files/tsrp/Resources/Breath_Testing_for_Prosecuto.pdf.
[18] *3 Types of Breathalyzers*, ELAWTALK, https://elawtalk.com/types-of-breathalyzers/ [https://perma.cc/AKT9-SY3L].
[19] Barron Lerner, *How Police Nab Drunk Drivers: From Drunkometer to Breathalyzer,* WBUR (Dec. 31, 2012), http://hereandnow.legacy.wbur.org/2012/12/31/breathalyzer-history [https://perma.cc/DAR6-LQLM].
[20] Edwin Conrad, *Push-Button Evidence*, 41 VA. L. REV. 217, 218 (1955).
[21] *Id.* at 219.

> were outlined by eminently qualified experts, and finally, the results of the test were interpreted medically.[22]

What constitutes such scientific "recognition"? The Federal Rules of Evidence Rule 702 on Testimony by Expert Witnesses states:

> A witness who is qualified as an expert by knowledge, skill, experience, training, or education may testify in the form of an opinion or otherwise if:
> (a) the expert's scientific, technical, or other specialized knowledge will help the trier of fact to understand the evidence or to determine a fact in issue;
> (b) the testimony is based on sufficient facts or data;
> (c) the testimony is the product of reliable principles and methods; and
> (d) the expert has reliably applied the principles and methods to the facts of the case.[23]

Pursuant to 1993 *Daubert v. Merill Dow Pharmaceuticals, Inc.*, the trial judge must determine "whether the testimony's underlying reasoning or methodology is scientifically valid and properly can be applied to the facts at issue."[24] In other words, the trial judge is the gatekeeper of scientific evidence determining what is admitted and what is excluded. It is important to note the role breathalyzers play in cases involving driving under the influence ("DUI") cases. Blood-alcohol concentration is what matters in American DUI cases, and thus, the breathalyzer measurements are critical.[25]

For breathalyzers, accuracy is determined by state and federal testing. The *Omnibus Transportation Employee Testing Act of 1991*[26] requires that pilots of various public transportation—aviation, mass transit, and

---

[22] *Id.* at 228.
[23] FED. R. EVID. 702.
[24] 509 U.S. 579, 580 (1993).
[25] Christopher Combs, *The Truth About Blood Alcohol Level & the Breathalyzer*, COMBS L. GRP. (June 3, 2018), https://www.combslawstl.com/blog/2018/06/03/the-truth-about-blood-alcohol-level-the-breathalyzer/ [https://perma.cc/BJ94-WZ3T].
[26] Department of Transportation and Related Agencies Act, Pub. L. No. 102-43, 105 Stat. 917, 953 (1992).

so on—be subject to alcohol testing programs, and the National Highway Traffic Safety Administration ("NHSTA") tests "evidential breath testers" on a regular basis[27] and publishes a "Conforming Products List of Evidential Breath Measurement Devices."[28] While states are not required to follow the federal list of approved devices, NHSTA provides funds to purchase approved devices, and thus many states adhere to the NHSTA list.[29] The devices' accuracy became trusted, and that trust lasted through several decades of cases.[30]

But there was always pressure against this trust. DUI cases form a peculiar subset of court cases; like jaywalking, cheating on income taxes, and failing to fully separate recyclable goods from trash, DUI is committed relatively frequently by people who are not considered criminals in the usual sense of the word. Yet the consequence of a DUI conviction—it may result in incarceration or the loss of a driver's license—can be high.[31] Accordingly, many drivers "lawyer up" when facing a potential DUI conviction. Legal attacks on breathalyzers' accuracy are thus not uncommon.[32]

---

[27] Highway Safety Programs; Model Specifications for Devices to Measure Breath Alcohol, 58 Fed. Reg. 48705 (Sept. 17, 1993).
[28] *See, e.g.*, Highway Safety Programs; Conforming Products List of Evidential Breath Alcohol Measurement Devices, 77 Fed. Reg. 35747 (June 14, 2012).
[29] U.S. GOV'T ACCOUNTABILITY OFF., GAO-08-477, TRAFFIC SAFETY: IMPROVED REPORTING AND PERFORMANCE MEASURES WOULD ENHANCE PERFORMANCE OF HIGH-VISIBILITY CAMPAIGNS (2008).
[30] As an example, in 2008 the New Jersey Supreme Court wrote that "[t]he accuracy and reliability of the breathalyzer itself has remained essentially unquestioned since our decision in *Romano v. Kimmelman*, 96 N.J. 66, 474 A.2d 1 (1984)." State v. Chun, 943 A.2d 114, 120 (N.J. 2008).
[31] With public transportation being minimal or even non-existent in many parts of the United States, loss of a driver's license can have a secondary, very severe penalty: loss of a job. *See* Alana Semuels, *No Driver's License, No Job*, ATLANTIC (June 15, 2016), https://www.theatlantic.com/business/archive/2016/06/no-drivers-license-no-job/486653/ [https://perma.cc/QEN6-C5E7]; Dana DiFillippo, *The Poverty Penalty: Should States Suspend Driver's Licenses for Scofflaws*, WHYY (Nov. 27, 2017), https://whyy.org/articles/poverty-penalty-states-suspend-drivers-licenses-court-scofflaws/ [https://perma.cc/2UKH-X5C3].
[32] *See generally* Ronald MacGregor, *Breathalyzers: Defects Then/Failures Still*, HG.ORG, https://www.hg.org/legal-articles/breathalyzers-defects-then-failures-still-30878 [https://perma.cc/L684-SUVE].

Modern devices all depend on software, but software unfortunately introduces new and potentially hidden failure modes into the technology. DUI convictions are per se violations based on breathalyzer results—a rating over .08 blood-alcohol level means the person is legally drunk[33]—making device accuracy critical. However, many of state testing labs were limited in their capabilities; the labs could only do "black box" testing, providing input to a breathalyzer and seeing the output, and not directly examining the software. Indeed, in many cases, even state lab testers lacked access to the code. In cases from Florida,[34] to Minnesota,[35] to New Jersey,[36] to Massachusetts, courts have said the denial of access to such source code[37] in criminal prosecution violates due process and the Confrontation Clause.[38] Thus, examinations of the software have started, and software in breathalyzers has been found to have errors.[39]

One important one was the 2008 New Jersey case of *State v. Chun*, a consolidation of twenty cases of individuals charged with DWI.[40] The counsel obtained the source code for the Alcotest 7110 MKIII-C breathalyzer.[41] The device works by using both infrared ("IR") and electrical chemical ("EC") oxidation technology to measure breath-alcohol levels.[42] Subjects are typically tested twice, for a total of four readings: IR_1 and EC_1 from the first breath and IR_2 and EC_2 from the

---

[33] *DUI Offense Basics*, FINDLAW, https://dui.findlaw.com/dui-charges/dui-offense-basics.html [https://perma.cc/F74G-G87S] (last updated Oct. 24, 2018).
[34] *See generally* State v. Allen, No. 08-CT-8840, slip op. at 2-3 (Fla. Cir. Ct. Apr. 24, 2009) (unpublished); State v. Atkins, No. 48-2008-CT-673-E, slip op. at 8-9 (Fla. Orange Cnty. Ct. June 20, 2008) (unpublished); State v. Lance Conley, No. 48-2012-CT-000017-A/A (Fla. Cir. Ct. Sept. 22, 2014).
[35] State v. Underdahl, 749 N.W.2d 117, 122-23 (Minn. Ct. App. 2008).
[36] *Chun*, 943 A.2d at 170.
[37] "Source code" is explained below. *See infra* Section II.
[38] Thomas E. Workman Jr., *Massachusetts Breath Testing for Alcohol: A Computer Science Perspective*, 8 J. HIGH TECH. L. 209, 232 (2008); *see also Chun*, 943 A.2d at 128-31.
[39] Stacy Cowley & Jessica Silver-Greenberg, *These Machines Can Put You in Jail. Don't Trust Them.*, N.Y. TIMES (Nov. 3, 2019) https://www.nytimes.com/2019/08/06/nyregion/crash-five-dead-long-island.html [https://perma.cc/K8PP-LV42].
[40] 943 A.2d. at 121.
[41] Evan Levow, *Summary of the Software House Finding for the Source Code of the Draeger Alcotest 7110 MKIII-C*, DWI.COM, https://www.dwi.com/new-jersey/state-v-chun/ [https://perma.cc/URH9-94KC].
[42] *Chun*, 943 A.2d. at 128.

second breath.[43] The measurements from the first breath are compared to the measurements from the second breath; if the measurements from the first breath are not within accepted levels of tolerance of the second breath, then the machine operator requires the subject to take a third breath; this will again provide two readings.[44]

The outside examiners noted that, "[w]hen the software takes a series of readings, it first averages the first two readings. Then, it averages the third reading with the average just computed. Then the fourth reading is averaged with the new average, and so on."[45] (Mathematically, that is not an average, as it weighs the first reading significantly more than the others, the second less than the first, but more than the others, etc.) The examiners were concerned that the error detection logic of the device needed a measurement error to register thirty-two times in a row for an error message to be displayed; otherwise the error was ignored.[46]

The court's special master deemed the device to be scientifically reliable,[47] but the court itself did not.[48] The New Jersey Supreme Court was not bothered by the way the device computed the so-called average, instead pointing out that what this did was give higher weight to the portion of the breath from the deepest air in the lung, the portion that most accurately reflects blood-alcohol levels.[49] But the court observed an error occurred when one of the original four readings fell outside accepted tolerance of the device.[50] In such a case, the subject is asked to do a third breath into the device,[51] and all six readings—two from each

---

[43] Jeffrey Lustick, *Getting to Know Washington's New DUI Breathalyzer: The German Made Dräger Alcotest 9510*, LUSTICK KAIMAN & MADRONE PLLC (Feb. 24, 2016), https://www.lustick.com/blog/getting-know-washingtons-new-dui-breathalyzer-german-made-drager-alcotest-9510/ [https://perma.cc/LN99-Q6QL].
[44] *Id.*
[45] Levow, *supra* note 41.
[46] *Id.*
[47] *Chun*, 943 A.2d. at 153; STEPHEN P. SMITH ET AL., IIT RESEARCH INSTITUTE, INDEPENDENT REVIEW OF THE CARNIVORE SYSTEM, FINAL REPORT xiii (2000), https://www.epic.org/privacy/carnivore/carniv_final.pdf.
[48] *Chun*, 943 A.2d. at 153.
[49] *Id.* at 156-57.
[50] *Id.* at 157-58.
[51] According to the court, as many as 5% of defendants might be tested on a third breath. *Id.* at 157.

breath—are included in its calculation (or at least, they should be). As the New Jersey Supreme Court discovered, this did not happen.

If the second EC reading was low—indicating the subject had low blood-alcohol levels—that evidence was simply ignored by the system. This was not because of malfeasance, but because of a programming error "due to [a] buffer overflow error"—an overwriting of the data.[52] Buffer overflow errors been known academically since 1981[53] (and in practice since 1988[54]). If the defense had an opportunity to examine the code, this error might have been found, in which case the evidence would have been dismissed.

That such an error should occur in a breathalyzer marketed in 2008 is both astonishing and highly disturbing. Avoiding such errors is always taught in first semester programming classes;[55] the existence of such an error demonstrates a serious lack of quality control by the vendor. An adversarial audit may have found the problem; without both source code and a suitable test environment (itself hard to construct without source code), there would be no way to find it.

The outside examiners found "thousands of programming errors" upon testing the Alcotest 7110 MKIII-C.[56] Drager, the manufacturer, claimed to have fixed the problems, but in 2019 the *New York Times* reported that the state failed to update the software in its devices.[57]

---

[52] *Id.* at 157-59.
[53] *See* C.A.R. Hoare, *The Emperor's Old Clothes*, 24 COMMC'NS ACM 75, 76-77 (1981).
[54] Jon A. Rochlis & Mark W. Eichin, *With Microscope and Tweezers: The Worm from MIT's Perspective*, 32 COMMC'NS ACM 689, 689 (1989).
[55] Buffer overflows are typically caused by lack of bounds-checking on what are known as "array indices." Improper indices are always wrong; using them can cause program crashes, erroneous results, or security problems. Consequently, all introductory classes and texts stress this point. *See, e.g.*, ROBERT SEDGEWICK & KEVIN WAYNE, INTRODUCTION TO PROGRAMMING IN JAVA: AN INTERDISCIPLINARY APPROACH §1.4 (2d ed., 2017) ("When programming with arrays, you must be careful. It is your responsibility to use legal indices when accessing an array element.").
[56] Cowley & Silver-Greenberg, *supra* note 39.
[57] *Id.*

Programming errors such as these are not limited to breathalyzers. A particularly striking example comes from Carnivore,[58] an FBI program for analyzing network traffic—this is called a packet sniffer—that was developed in the late 1990s and used on ISP's networks. Using filters to record traffic fitting some predetermined pattern, such as all email to or from a particular target, Carnivore would collect traffic pursuant to a pen/trap order or wiretap warrant.[59]

The software program was not properly designed, however, enabling incorrect collection of communications traffic. In 2000, an outside review committee discovered that, "Incorrectly configured, Carnivore can record any traffic it monitors."[60] The ability to record "any traffic" should not have been possible. In addition, the system permitted additional copies of intercepts to be made, even if they had not been minimized (a requirement for all lawful wiretaps); only FBI "procedures and professionalism" prevented that from occurring.[61]

One might imagine that the concerns the oversight committee raised were theoretical worries; surely the device would not actually exhibit such flaws in practice. In fact, serious problems occurred in the collection. An FBI anti-terrorism investigation by the team focusing on Osama bin Laden used Carnivore.[62] According to an internal memo, "[t]he FBI software not only picked up the E-mails under the electronic surveillance of the FBI's target XXX but also picked up E-mails on non-covered targets. The FBI technical person was apparently so upset that he destroyed all the E-mail take, including the take on XXX."[63]

---

[58] Carnivore was later renamed DCS-1000, for Digital Collection Service 1000. *DCS-3000 is the FBI's New Carnivore*, WIRED (Apr. 27, 2006, 7:04 PM), https://www.wired.com/2006/04/dcs3000-is-the-/ [https://perma.cc/U6VQ-EUCH].
[59] SMITH ET AL., *supra* note 47.
[60] *Id.* at xii.
[61] *Id.* at 4.2-4.3.
[62] The unit was known as the "UBL" unit; they spelled the target's name "Usama bin Laden." Michele Zanini & Sean J.A. Edwards, *The Networking of Terror in the Information Age*, *in* NETWORKS AND NETWARS: THE FUTURE OF TERROR, CRIME, AND MILITANCY, 39 (1999), https://www.rand.org/content/dam/rand/pubs/monograph_reports/MR1382/MR1382.ch2.pdf.
[63] Memorandum from the FBI on FISA Mistakes to Spike Bowman (April 5, 2000) (on file with Electronic Privacy Information Center) https://www.epic.org/privacy/carnivore/fisa.html [https://perma.cc/6FND-CHYV]. Context is given in the EPIC press release. *See* Press Release, Electronic Privacy Information Center, FBI's Carnivore System Disrupted Anti-

Carnivore was collecting digital evidence, which makes the fact that the Carnivore software was improperly designed particularly disturbing. Breathalyzers test a physical object—a human breath. While it is not standard to preserve the actual breath sample, it is possible to require that such samples be preserved until court hearings have occurred; in fact, the states of Alaska and New Hampshire do exactly that.[64] However, the digital artifacts that Carnivore collected *were* the actual evidence itself. One aspect of the Carnivore evidence collection process—overcollection—had gone wrong, but so could other aspects of collection. This raises the possibility that the collected evidence as seen by the court could be incorrect or it could appear misleadingly out of context. This makes clear that the ability of the defendant to cross-examine the "witnesses"—in this case, the digital code that did the collection—is crucial.

Computer forensic analysis provides yet a third example of problems with evidentiary software. Apple forensics expert Jonathan Zdziarski was hired by the United States Army to examine phone evidence in the case of a brigadier general who had been accused of sexual harassment, assault, and making murder threats.[65]

Terror Investigation (May 28, 2002) (on file with author) https://www.epic.org/privacy/carnivore/5_02_release.html [https://perma.cc/QM3T-KYBF].

[64] Best v. Mun. of Anchorage, 712 P.2d 892, 898–99 (Alaska Ct. App. 1985); Mun. of Anchorage v. Serrano 649 P.2d 256 (Alaska Ct. App. 1982) (state must preserve a breath sample). Current New Hampshire law requires the collection and preservation of an additional blood sample, and that someone who is asked to perform a breath test shall be informed that they have the right, at their own expense, to have a blood sample drawn and preserved. *See* In re Op. of Justices (Eliminating Requirements for Additional Breath Samples), 2 A.3d 1102 (N.H. 2010) (New Hampshire Supreme Court rejected proposed amendment to eliminate the requirement to save two breath samples). Other states have a duty to preserve, but it is not constitutionally mandated. Arizona has a duty to preserve breath samples for independent testing pursuant to its implied consent law. *See* RSA 265-A:7 II and III; State v. Vannoy, 866 P.2d 874, 878 (Ariz. Ct. App. 1993) (discussing ARIZ. REV. STAT. ANN. § 28-691). Failure to preserve breath samples did not violate due process or equal protection. *See also* People v. Molina, 468 N.Y.S.2d 551, 558 (N.Y. Crim. Ct. 1983) ("basic fairness and the duty to make of the trial a search for truth informed by all relevant material requires that the police preserve for the defendant's use a separate breath sample for testing and analysis"); People v. Trombetta, 219 Cal. Rptr. 637 (Cal. Ct. App. 1985).

[65] SUSAN LANDAU, LISTENING IN: CYBERSECURITY IN AN INSECURE AGE 145 (2017) (ebook).

Zdziarski was brought into the case late, just before the trial was to start. In the course of examining the phone, Zdziarski determined that the computer forensic tools initially used in the investigation did not work correctly. Timeline was a crucial issue in the case, and so Zdziarski needed to look at timestamps for device erasures, backup restores, and file accesses. The commercial forensic tools had reported these inaccurately. Problems included incorrect times for app deletion, basing the time on when the phone was next booted up rather than when the app was actually removed.[66] This meant that the forensic tools were miscalculating when the app was actually used. Zdziarski's analysis changed the case, with the Army dropping the most serious charges.[67]

Breathalyzers, packet sniffers and wiretaps, and computer forensic tools are far from the only examples of evidentiary software. Indeed, the errors we have discussed present but the tip of an iceberg. Other instances include radar guns, tools for conducting DNA analysis, and network investigative techniques (the last is government malware used, pursuant to court order, to intrude onto a system in order to investigate and/or obtain evidence).[68] Moreover, with our drive willy-nilly into the Digital Revolution, we are moving into a world in which law enforcement will increasingly rely on evidentiary software. From automatic toll booths that register when a car enters and exits a highway to personal devices that record where and when an individual is, the percentage of cases that rely on evidentiary software is increasing. The fact that such software may incorrectly report its findings makes cross-examining source code crucial to the basic protections laid down in the Bill of Rights.

Yet that capability, so necessary for protecting a defendant's rights, is disappearing. Even as we rely on computer software to bear witness, we are simultaneously moving to a criminal justice system in which "law

---

[66] Jonathan Zdziarski, *An Example of Forensic Science at Its Worst: US v. Brig. Gen. Jeffrey Sinclair*, ZDZIARSKI'S BLOG OF THINGS (Aug. 24, 2014), https://www.zdziarski.com/blog/?p=3717 [https://perma.cc/PR75-FW6A].

[67] LANDAU, *supra* note 65.

[68] *See, e.g.*, Jennifer Stisa Granic, *Challenging Government Hacking: What's at Stake*, ACLU (Nov. 2, 2017, 10:00 AM), https://www.aclu.org/blog/privacy-technology/internet-privacy/challenging-government-hacking-whats-stake [https://perma.cc/T6HC-FZ27].

enforcement privilege"[69] frequently prevents access to the details of investigative techniques and procedures.

Given the guarantees of the Fifth and Sixth Amendments, it may be difficult to understand how prosecutors are able to withhold details of the tools used in evidence gathering. This approach is a relatively new phenomenon.[70] Recently, United States Magistrate Judge Stephen Smith presented a compelling history of the law enforcement privilege's genesis.[71] The change began with the 1950 espionage trial of Judith Coplon, a Department of Justice employee in the Foreign Agents Registration Division.[72] Coplon was also a Soviet spy. The FBI had become suspicious of her and fed her a false document.[73] Coplon was arrested as she was handing over the fake document—and twenty-eight classified memoranda—to her accomplice.[74]

The twenty-eight other memoranda were a problem for FBI Director J. Edgar Hoover, for they showed evidence that the FBI was illegally wiretapping.[75] In 1937, ruling on the basis of the Federal Communications Act,[76] the Supreme Court deemed it illegal to "intercept" and "divulge" wired communications.[77] Initially, the Department of Justice interpreted the decision to mean interception and divulgence of content *outside the federal government* was unlawful.[78] In 1940 President Franklin Roosevelt authorized the practice against those suspected of subversive activities against the United States government, urging Attorney General Jackson to limit the investigations

---

[69] Smith, *supra* note 10, at 233.
[70] *Id.*
[71] *See generally id.*
[72] *See generally* United States v. Coplon, 185 F.2d 629 (2d Cir. 1950).
[73] *Id.* at 632-35.
[74] ATHAN G. THEOHARIS & JOHN STUART COX, THE BOSS: J. EDGAR HOOVER AND THE GREAT AMERICAN INQUISITION 256 (1988).
[75] Smith, *supra* note 10, at 237.
[76] 47 U.S.C. § 605.
[77] Nardone v. United States, 302 U.S. 379, 382 (1937).
[78] SELECT COMM. TO STUDY GOVERNMENTAL OPERATIONS WITH RESPECT TO INTELLIGENCE ACTIVITIES, S. REP. NO. 94-755, 94th Cong., 2d Sess., Book III, at 278 n.25 (letter from Attorney General Robert Jackson to Rep. Hatton Summers, March 19, 1941).

to a minimum and "limit them insofar as possible to aliens."[79] Nonetheless, the original restriction of the court still stood. The FBI had been wiretapping Coplon; although Hoover sought to conceal that fact during Coplon's trials, her defense attorney was able to get it into the record.[80] As a result, Coplon's convictions were overturned.[81] The more significant result from this was Hoover's response, which was effectively to hide wiretaps from view.[82]

The wily FBI director did not stop there. As Stephen Smith has detailed, Hoover publicly campaigned to establish an "evidentiary privilege covering law-enforcement techniques and procedures."[83] Evidentiary privilege was a new idea in 1950. However, as we can see from very public law-enforcement efforts to keep the techniques of investigative technologies hidden,[84] it is an oft-used practice.

The FBI director's efforts are not the only ones pushing in favor of evidentiary privilege. Judge Smith details how a small exemption for law-enforcement records in the Freedom of Information Act became a Mack Truck-sized hole that has permitted the withholding of evidentiary information by multiple courts.[85]

Sometimes legal problems arise from a single issue, as it did in the Coplon trial, where the withholding of evidence resulted in an unfair trial and Coplon's release. Sometimes legal problems develop from a

---

[79] *Id.* at 279 (Franklin D. Roosevelt, Confidential Memorandum for the Attorney General, May 21, 1940).
[80] *See* Smith, *supra* note 10, at 234.
[81] Coplon was tried in the District of Columbia on unauthorized possession of classified documents and in the Southern District of New York on espionage. The illegal wiretapping formed the basis for an appeal of Coplon's second trial, which won her a reversal. Smith, *supra* note 11, at 234-37.
[82] Wiretap evidence including requests for taps, etc. would be stored separately from the main files of a case, with only a single copy of the wiretap stored. This was held in a secure area at FBI headquarters. THEOHARIS & COX, *supra* note 74, at 259-60.
[83] Smith, *supra* note 10, at 243.
[84] Stephanie K. Pell & Christopher Soghoian, *Your Secret Stingray's No Secret Anymore: The Vanishing Government Monopoly over Cell Phone Surveillance and Its Impact on National Security and Consumer Privacy*, 28 HARV. J.L. & TECH. 1, 33-34 (2014).
[85] Smith, *supra* note 10, at 248-50.

confluence of issues. That is the situation here with the combination of evidentiary software, evidentiary privilege, and software error.

The issue of software error occupies a large part of computer science research and practice,[86] including security (a high percentage of security problems are due to buggy code[87]) and concerns about fairness and accuracy in machine learning. Here, we have focused narrowly on how the combination of evidentiary software, evidentiary privilege, and software error poses a threat to the fundamental right of criminal defendants to a fair trial. The good news is that, unlike the larger problem of software error, this problem posed by errors in evidentiary software appears relatively solvable. While our proposed solution challenges the present handling of the situation, our recommendation is technically feasible and is without high financial costs.[88]

We begin in Section II with an explanation of the nature of software and thus of coding errors, ending with an explanation of modern techniques used to find and correct these problems. In Section III we discuss the three Constitutional concerns that are raised by software errors: *Brady* violations (the prosecution must provide exculpatory evidence to a defendant),[89] the implications of the Confrontation Clause, and the Due Process clause. We conclude in Section IV with our recommendations.

II. The Root of the Problem

The essence of our argument is this: the nature of software, and hence of computer programming, is such that certain errors are more likely to

---

[86] Buggy code has long been a focus for academics and practitioners alike. "First, one must perform perfectly. The computer resembles the magic of legend in this respect, too. If one character, one pause, of the incantation is not strictly in proper form, the magic doesn't work. Human beings are not accustomed to being perfect, and few areas of human activity demand it. Adjusting to the requirement for perfection is, I think, the most difficult part of learning to program." FREDERICK P. BROOKS, JR., THE MYTHICAL MAN-MONTH 8 (1975).

[87] *See* WILLIAM R. CHESWICK & STEVEN M. BELLOVIN, FIREWALLS AND INTERNET SECURITY: REPELLING THE WILY HACKER 7 (1994).

[88] It is in fact not clear that there will be a net increase in costs, if early audits reduce the need for future litigation.

[89] Brady v. Maryland, 373 U.S. 83, 87 (1963).

be found by adversarial testing. We start by giving a brief overview of the nature and limitations of software development.

The software design process typically starts with the creation of "specifications," a detailed description of what the system should do.[90] It is not enough to say, for example, "measure breath alcohol level." What are the expected ranges? (A real breath sample will never consist of pure ethanol.) How much breath must be sampled? How close should it come to the results from an actual blood test? What sort of operational errors should be detected? What should the system do if it encounters anomalous situations? Should there be internet connectivity, to relay results immediately? Should there be a camera and a GPS, for better evidentiary value? If so, what if the device is used in a cellular dead spot, or if the camera or GPS are not working? How many samples from how many drivers must the device store?

Errors can creep in at any point. Unexpectedly high readings might erroneously show up as low because the device was never intended to handle such values.[91] The specification may have omitted important details, e.g., what to do if the battery voltage is below a certain level. Programmers use the specifications to guide writing the actual programs. Testers then use the same specification documents to guide their tests. An error in the specifications, then, can turn into erroneous code; such errors will not be detected by the testers, since their job is to ensure that the program matches the specifications.

Before we explain how coding errors can occur, we provide a brief note on terminology. As is widely known, computers "understand" only 0s and 1s. That is, the actual computer hardware is only capable of understanding "binary": a way to represent numbers or characters using only the values 0 and 1, commonly understood as an electrical circuit

---

[90] *See, e.g.,* Barry W. Boehm, *Verifying and Validating Software Requirements and Design Specifications*, IEEE SOFTWARE, Jan. 1984, at 76.

[91] Extreme values can occur legitimately. For example, in a recent New York criminal case, the defendant was described as having "the second highest level of marijuana that they've ever seen in a living specimen." Arielle Dollinger, *The Stolen Car Was Going 154 M.P.H. Five People Ended up Dead.*, N.Y. TIMES (Aug. 6, 2019), https://www.nytimes.com/2019/08/06/nyregion/crash-five-dead-long-island.html [https://perma.cc/3H9S-3E3S].

being off or on. This is the reason that the actual programs run by a computer chip are often referred to as "binaries" or "executables." Programmers, though, almost never directly code actual binary programs. Instead, they write in "source code," a (more-or-less) human-readable language.[92] Specialized programs known as "compilers" translate the source code to executables. While it is possible, with proper tools and considerable effort, for a human to understand an executable—indeed, analysis of computer viruses and other malware relies on this ability[93]—analysis is far easier and far more complete if the original source code is available.

A. Computer Programming

It is a truism that computers do only what they are told to do. The process of "telling" a computer what to do is called programming, and while programming is no longer the rare, arcane art that it once was, nevertheless many people have no exposure to it. Then, we explain programming errors that cause bugs as well as how programmers test to eliminate such bugs. Finally, we address the necessity of adversarial audits to safeguard defendants' rights.

Consider the following program fragment.[94]

```
int a, b;                   /* Reserve two storage locations */

scanf("%d", &a);            /* Read the first value */
scanf("%d", &b);            /* Read the second */
printf("%d\n", a + b);      /* Print their sum */
```

It reads into two values and prints their sum. There are several things worth noting about these few lines of code.

---

[92] Some examples of source code are given. *See infra* Section II.A.

[93] *See generally* MICHAEL SIKORSKI & ANDREW HONIG, PRACTICAL MALWARE ANALYSIS: THE HANDS-ON GUIDE TO DISSECTING MALICIOUS SOFTWARE (Alison Law et al. eds., 2012).

[94] This fragment is written in a language known as C. *See generally* BRIAN W. KERNIGHAN & DENNIS M. RITCHIE, THE C PROGRAMMING LANGUAGE (2d ed. 1978). C is the ancestor of many currently used programming languages. Neil DuPaul, *The History of Programming Languages Infographic*, VERACODE (Apr. 15, 2013), https://www.veracode.com/blog/2013/04/the-history-of-programming-languages-infographic [https://perma.cc/ZPR2-HDR5].

The first, of course, is that the code itself—the first portion of each line—is completely incomprehensible to anyone but a programmer. The second thing to note is that the second half of each line, the portion enclosed between /* and */, is readable. These are known as "comments" and are ignored by the computer. This particular example is artificial, in that not only are there more comments than is customary, they're redundant to the code and would be seen as bad practice. That is, a comment like "Print their sum" is rather useless, since to any programmer it is apparent that is what the line does.[95] A better comment is one that explains why the sum is being printed, or what the input values mean.[96] Nevertheless, even to experienced programmers comments can be easier to read than code.

There is another, more subtle point: to a first approximation, every line of code interacts with every other. For example, in this case the first line says the storage locations "a" and "b" can only hold integers. It is straightforward to make them hold numbers with fractional parts, e.g., 1.5, by writing:

```
float a, b;
```

instead, but that would require certain changes to the corresponding scanf line to let it read so-called "floating point" numbers. Furthermore, and less obviously, the printf statement would require certain changes as well.

---

[95] Even non-programmers can see the root word "print" and "a + b". These mean exactly what they appear to mean. H. James de St. Germain, *Commenting*, JIM'S CS TOPICS, https://www.cs.utah.edu/~germain/PPS/Topics/commenting.html#:~:text=Comments%20should%20be%20useful%20high,easy%20to%20read%20as%20English. [https://perma.cc/M5QR-JE6D].

[96] Proper commenting style is routinely taught in introductory programming classes. *See id.*

Computers derive much of their power from two features: the ability to choose among alternatives and the ability to repeat activities. Suppose we wanted to change the above program to subtract two numbers, but always to subtract the lesser from the greater:

```
int a, b;               /* Reserve two storage locations */

scanf("%d", &a);            /* Read the first value */
scanf("%d", &b);            /* Read the second */
if (a > b)
            printf("%d\n", a - b);
    else
            printf("%d\n", b - a);
```

We could also do this repeatedly, for many pairs of numbers:

```
int a, b;               /* Reserve two storage locations */

while (1) {
    scanf("%d", &a);        /* Read the first value */
    scanf("%d", &b);        /* Read the second */
    if (a > b)
            printf("%d\n", a - b);
    else
            printf("%d\n", b - a);
}
```

We will forbear explaining these program fragments in detail, save to note that the last example has a bug: it will run forever, even when there is no more input available.[97]

---

[97] *See generally* KERNIGHAN & RITCHIE, *supra* note 94. There is an old joke about why programmers starve to death in the shower: the instructions on shampoo bottles often say "Lather, rinse, repeat," with no instruction on when to stop. Computers, after all, take things literally; programmers, perforce, must learn to do the same. Victor Raskin, *Verbal Play in Computer Jokes*, BELB, http://www.belb.info/studenti/mincheva/verbal%20play.htm [https://perma.cc/662L-P4LB].

B. Bugginess

Consider the following minor variant of the first programming fragment shown above, but where the two storage locations are called "k" and "l" instead of "a" and "b":

```
int k, l;                   /* Reserve two storage locations */

scanf("%d", &k);            /* Read the first value */
scanf("%d", &l);            /* Read the second */
printf("%d\n", k + 1);      /* Print their sum */
```

In fact, the two are *not* identical; in the second fragment, the sum printed is not of the two values, but of the first value, "k", and the number one.[98] Depending on the font used, the difference between the digit one and the 12th letter of the alphabet can be extraordinarily hard to spot. A programmer whose eyes see the comment is likely to glance only fleetingly at the actual code, and thus not spot the error. Errors—and sabotage—from similarly confusing patterns are by no means unknown in the real world.[99]

Simple bugs can have drastic consequences. A missing hyphen contributed to the loss of Mariner 1, the first space probe launched toward Venus.[100] The first launch of the Ariane 5 rocket, after $7 billion and a decade of development, failed because of a simple software error.[101] The first launch of the Space Shuttle Columbia, before a worldwide live television audience, was aborted on the launchpad due

---

[98] One of the authors of this article copied and pasted the code fragments above into an actual program, to verify that the behavior is as stated.

[99] In what is likely the best-documented sabotage incident, Princeton computer science professor Ed Felten described what was an apparent attempt to plant a back door—a deliberate security hole known only to its authors—in Linux, relying on just such a trick. Ed Felten, *The Linux Backdoor Attempt of 2003*, FREEDOM TO TINKER (Oct. 9, 2013), https://freedom-to-tinker.com/2013/10/09/the-linux-backdoor-attempt-of-2003/ [https://perma.cc/5KLY-3J45].

[100] David R. Williams, *Mariner 1*, NASA SPACE SCI. DATA COORDINATED ARCHIVE, https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=MARIN1 [https://perma.cc/2T5J-PMX2].

[101] James Gleick, *Little Bug, Big Bang*, N.Y. TIMES MAG. (Dec. 1, 1996), https://www.nytimes.com/1996/12/01/magazine/little-bug-big-bang.html [https://perma.cc/4R2K-QGL3].

to a bug.[102] Another notable example was a misplaced "break" statement[103] that led to the failure of most of AT&T's long distance network.[104] These bugs were immediately visible. Some bugs, though, are silent; one must look (and look hard) for them.[105]

Bugs are an omnipresent hazard in software. All non-trivial software packages have bugs;[106] even large-scale efforts to prevent them have their limits. Three of us wrote:

> [I]t is important to know a fundamental tenet of software engineering: bugs happen. In his classic *The Mythical Man-Month*, Frederick Brooks explained why:
>
> > First, one must perform perfectly. The computer resembles the magic of legend in this respect, too. If one character, one pause, of the incantation is not strictly in proper form, the magic doesn't work. Human beings are not accustomed to

---

[102] John R. Garman, *The "BUG" Heard 'Round the World: Discussion of the Software Problem Which Delayed the First Shuttle Orbital Flight*, 6 ACM SIGSOFT SOFTWARE ENGINEERING NOTES, Oct. 1981, at 3.

[103] In computer programs, a fundamental construct is "looping": repeated execution of a segment of code, generally until some condition is met. A "break" statement is one way to terminate execution of a loop. *See generally* KERNIGHAN & RITCHIE, *supra* note 95.

[104] BRUCE STERLING, THE HACKER CRACKDOWN 35 (1992) ("The 'break' was *supposed* to 'break' the 'if clause.' Instead, the 'break' broke the 'switch' statement.").

[105] S*ee, e.g.*, the descriptions of the Juniper and telnet issues, *infra* Section II.C.

[106] The problem of buggy code has been known since the dawn of computing. STERLING, supra note 105 (internal citations omitted). *See also* Tim Menzies & Thomas Zimmermann, *Software Analytics: What's Next?*, IEEE SOFTWARE, Sept.-Oct. 2018, at 64, 64-65 ("As soon as people started programming, it became apparent that programming was an inherently buggy process. Maurice Wilkes, speaking of his programming experiences in the early 1950s, recalled the following: . . . the realization came over me with full force that a good part of the remainder of my life was going to be spent in finding errors in my own programs."). By the early 1970s, researchers had developed formulas to predict the number of bugs in a program, based on its size and complexity. *Id.* at 65; *see also* Ronald L Rivest, *On the Notion of "Software Independence" in Voting Systems*, 366 PHIL. TRANSACTIONS ROYAL SOC'Y A 3759, 3760 (2008) ("Finding all errors in a large system is generally held to be impossible in general or else highly demanding and extremely expensive. Our ability to develop complex software vastly exceeds our ability to prove its correctness or test it satisfactorily within reasonable fiscal constraints.").

> being perfect, and few areas of human activity demand it. Adjusting to the requirement for perfection is, I think, the most difficult part of learning to program.
>
> Because computers, of course, are dumb—they do exactly what they are told to do—programming has to be absolutely precise and correct. If a computer is told to do something stupid, it does it, while a human being would notice there is a problem. A person told to walk 50 meters then turn left would realize that there was an obstacle present, and prefer the path 52 meters down rather than walking into a tree trunk. A computer would not, unless it had been specifically programmed to check for an impediment in its path. If it has not been programmed that way—if there is virtually any imperfection in code—a bug will result. The circumstances which might cause that bug to become apparent may be rare, but it would nonetheless be a bug.[107]

Although it may be hard for non-programmers to count individual bugs or the result of efforts to prevent them, computer security vulnerabilities—which are mostly due to buggy code—are generally more visible.[108]

---

[107] Steven M. Bellovin et al., *Lawful Hacking: Using Existing Vulnerabilities for Wiretapping on the Internet*, 12 NW. J. TECH. & INTELL. PROP. 1, 27 (2014).

[108] All large software products have a "ticketing system" where bug reports and feature requests are noted. In principle, this could be used to assess the bug rate, and many companies do this internally. However, this information is often carefully guarded. Ticketing systems for open source software are often public; however, distilling the data down to useful information—eliminating the duplicates, figuring out which entries are user error and hence incorrect, even deciding what is an actual bug and what is simply code that doesn't work quite as desired, even if that is the intended behavior—requires a great deal of effort. Clint Fontanella, *What's a Ticketing System?*, HUBSPOT (July 14, 2020), https://blog.hubspot.com/service/ticketing-system [https://perma.cc/LK8Y-7MY4]. Furthermore, it is unclear if the behavior of open source developers is similar enough to that of commercial developers. *See*, *e.g.*, Steven M. Bellovin, *The Open Source Quality Challenge*, SMBLOG (Apr. 29, 2009), https://www.cs.columbia.edu/~smb/blog/2009-04/2009-04-29.html [https://perma.cc/7TN3-HU53]*; see also* Brooks, *supra* note 87.

Microsoft provides an excellent case study. By 2001, the company realized it was in the throes of a security crisis. "'No Microsoft executive could have any conversation with any enterprise customer about anything but Microsoft's bad security,' said Steve Lipner, who was the company's director of security assurance at the time."[109] The Gartner Group warned its clients against using IIS, Microsoft's web server.[110] In response, in 2003, the company initiated a massive effort to produce more secure software[111] and had all of its 8,500 Windows developers take security training.[112] Within months the effort showed notable successes. Steven Lipner, who was Microsoft's director of engineering strategy, told the *New York Times*, "Some of the tougher security standards . . . have shown measurable improvement in Windows Server 2003, which shipped earlier this year. The number of security vulnerabilities detected so far is half as many as at this stage after the release of Windows Server 2000."[113] Microsoft's security development process has vastly improved the security of the company's products (a measure of its effectiveness is that the process has been emulated by Cisco and Adobe).[114] Nonetheless, security flaws (again, due to bugs) still occur: in August 2018, "Microsoft pushed 17 updates to fix at least 60 vulnerabilities in Windows and other software, including two 'zero-day' flaws that attackers were already exploiting before Microsoft issued patches to fix them."[115] In other words,

---

[109] LANDAU, *supra* note 65, at 64.
[110] Jack Kapica, *Gartner Slams Microsoft IIS Server,* GLOBE & MAIL (Sept. 25, 2001), https://www.theglobeandmail.com/technology/gartner-slams-microsoft-iis-server/article22621594/ [https://perma.cc/8ZW2-EHZ6].
[111] Steve Lohr, *Fixing Flaws, Microsoft Invites Attack*, N.Y. TIMES (Sept. 29, 2003), https://www.nytimes.com/2003/09/29/business/fixing-flaws-microsoft-invites-attack.html [https://perma.cc/5HWN-4YEX] ("At Microsoft, much more time is now being set aside in the design cycle of products for security considerations, a mandate approved by senior management this spring. 'There is a shift from mainly an emphasis on working features to an emphasis on trustworthy and secure computing,' said Steven B. Lipner, director of security engineering strategy at Microsoft.").
[112] *See* LANDAU, *supra* note 65, at 64.
[113] Lohr, *supra* note 111.
[114] Tim Rains, *The Secret of the SDL*, MICROSOFT (July 2, 2014), https://cloudblogs.microsoft.com/microsoftsecure/2014/07/02/the-secret-of-the-sdl/ [https://perma.cc/9A4Z-W67H].
[115] Brian Krebs, *Patch Tuesday, August 2018 Edition*, KREBS ON SECURITY (Aug. 18, 2018), https://krebsonsecurity.com/2018/08/patch-tuesday-august-2018-edition/ [https://perma.cc/2K6H-5MER].

although good software development processes are a significant help, they are not a panacea; bugs (and security holes) have nevertheless persisted, more than fifteen years after Microsoft began its initiative.

C. Testing and Assurance

Naturally, software vendors are aware of this problem and try to find and eliminate bugs. The most common method is testing, which involves feeding assorted sample inputs to a program to see if the correct answer is produced. [116] This, however, has limitations. The most fundamental issue with testing was noted decades ago by Edsger Dijkstra, an esteemed computer scientist: "Program testing can be used to show the presence of bugs, but never to show their absence!"[117] That is, a test case might yield an incorrect result, thereby demonstrating a bug; however, it is impossible to tell if there is a bug that has not been triggered by a particular set of inputs: there are far too many possible situations to try them all. In fact, it is mathematically impossible to find all failures.[118]

Some problems fundamentally cannot be detected simply by testing. The most obvious example is encryption. By definition, the output of a good encryption algorithm is indistinguishable from a random string of bits.[119] Indeed, if an encryption algorithm's output can be distinguished

---

[116] *See generally* PAUL AMMANN & JEFF OFFUTT, INTRODUCTION TO SOFTWARE TESTING (Cambridge U. Press, 2d ed. 2017).

[117] Edsger W. Dijkstra, *Structured Programming*, *in* SOFTWARE ENGINEERING TECHS. 65, 66 (J. N. Buxton & B. Randell eds., 1970), http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1969.PDF.

[118] AMMANN & OFFUTT, *supra* note 116, at 20 ("the problem of finding all failures in a program is undecidable."). "Undecidable" is a technical term in computer science and mathematics; it means that no possible algorithm can always find the correct answer. *Undecidability*, U. OF ROCHESTER, https://www.cs.rochester.edu/u/nelson/courses/csc_173/computability/undecidable.html [https://perma.cc/RV9A-759Z].

[119] The notion that ciphertext should be random is due to C. E. Shannon's article *Communication Theory of Secrecy Systems*, but his phrasing is quite mathematical and does not quite match current usage. *See* C.E. Shannon, *Communication Theory of Secrecy Systems*, 28 BELL SYS. TECH. J. 656, 656-715 (1949). A more accessible reference simply assumes that randomness is a requirement for a good cipher. JUAN SOTO, JR., U.S. DEPT. COM., RANDOMNESS TESTING OF THE ADVANCED ENCRYPTION STANDARD CANDIDATE ALGORITHMS (1999) ("One of the criteria used to evaluate the Advanced Encryption Standard candidate

from a random string, that algorithm is considered seriously flawed.[120] However, the presence of randomness is not itself sufficient for correctness; the encryption keys must also be chosen properly.[121] That is a result of the principle established by Auguste Kerckhoffs in 1883: one cannot assume that the encryption algorithm is secret.[122] Therefore, the security of an encryption algorithm must lie in the secrecy of the key, and ensuring that secrecy is thus critical to an encryption's algorithm working appropriately.[123] Unfortunately, testing will not necessarily reveal that secrecy. Two examples better illustrate this point. The first example concerned an encrypted version of the Telnet protocol.[124] Due to a bug in the key generation mechanism, most of the time a constant key would be used.[125] Anyone who knew this key could

---

algorithms was their demonstrated suitability as random number generators. That is, the evaluation of their output utilizing statistical tests should not provide any means by which to computationally distinguish them from a truly random source."). The actual modern definition of encrypting a message, though quite mathematical, assumes that an ideal encryption algorithm generates a sequence of random bits. *See*, *e.g.*, Shafi Goldwasser & Silvio Micali, *Probabilistic Encryption*, 28 J. COMPUT. & SYS. SCI. 270, 272 (1983) ("More specifically, a binary message will be encrypted bit-by-bit as follows: a "0" is encoded by randomly selecting an x such that $B(x) = 0$ and a "1" is encoded by randomly selecting an x such that $B(x) = 1$. Consequently, there are many possible encodings for each message. However, messages are always uniquely decodable.").

[120] Goldwasser & Micali, *supra* note 119, at 271.

[121] Today, an encryption key is generally a very large number; depending on the particular encryption algorithm used, the keys can be tens or even hundreds of digits long. Rob Stubbs, *Classification of Cryptographic Keys*, CRYPTOMATHIC (Feb. 19, 2018), https://www.cryptomathic.com/news-events/blog/classification-of-cryptographic-keys-functions-and-properties [https://perma.cc/JE5Y-ZSCP].

[122] Auguste Kerckhoffs, *La Cryptographie Militaire*, 9 J. DES SCI. MILITAIRIES 5, 12 (1883) (Fr.) ("Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi." ["The system must not require secrecy and can be stolen by the enemy without causing trouble"]).

[123] *Id.*

[124] Telnet, a now-obsolescent protocol, was primarily used for command-line access to remote computers. Users thus had to enter a login name and password, hence the need for encryption. Tyson Supasatit, *Can You Encrypt Telnet?*, EXTRAHOP (Aug. 14, 2019), https://www.extrahop.com/company/blog/2019/telnet-security-how-to-encrypt-telnet-sessions/ [https://perma.cc/G4P5-2D2F].

[125] The fact of the problem was announced in CERT Advisory CA-1995-03, but no details were provided, though the true situation was well known in the security community. CERT DIVISION, CARNEGIE MELLON U., 1995 CERT ADVISORIES 20 (Carnegie Mellon U. Software Engineering I. ed., 2017). According to a comment on a blog, the actual cause was deliberately obfuscated. John Gilmore, Comment to *Vendors Are Bad for Security*, LINKS: BEN LAURIE BLATHERING (May 13, 2008, 2:09 PM), https://www.links.org/?p=327 [https://perma.cc/8NV7-WKSE].

trivially decrypt the traffic.[126] Simple testing would not have found the problem; the output would have looked random despite the constant key.[127] Only an examination of the code itself would have demonstrated the problem that the same encryption key was being used for all inputs.

The second incident was more sinister. The problems started with an algorithm for generating random numbers that was originally put forth by the National Security Agency (NSA). In NSA's words, the agency had "covertly influence[d] and/or overtly leverage[d] . . . commercial products designs . . . mak[ing] the systems in question exploitable."[128] The system in question was Dual EC_DRBG, a random number generator that provided bits for an encryption key.[129] Recall Kerckhoff's

---

[126] CERT DIVISION, *supra* note 125; Gilmore, *supra* note 125.

[127] Because of technical details of how the encryption actually worked, multiple encryptions of the same data would look different, despite the fact that the same key was used. *See generally*, Mihir Bellare & Phillip Rogaway, Introduction to Modern Cryptography (2005) (collection of class notes), available at https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf; JASON GARMAN, KERBEROS (2003); PETER LOSHIN, TCP/IP CLEARLY EXPLAINED 221-35 (4th ed. 2002).

[128] *Secret Documents Reveal N.S.A. Campaign Against Encryption,* N.Y. TIMES (Sept. 5, 2013), https://archive.nytimes.com/www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html [https://perma.cc/AYC3-FM5Z].

[129] Random numbers are used to generate encryption keys but finding genuinely random bits—that is to say, bits that are hard to predict—is a computationally difficult problem. Consequently, a common method instead is to start with some truly random bits, then use a mathematical function to expand these into a longer sequence of pseudo-random bits. The Elliptic Curve Digital Random Bit Generator (Dual EC-DRBG) is one such function; it uses elliptic curves to do so. The curve relies on two default parameters. The choice of these parameters is crucial, for anyone knowing the arithmetic relationship between the parameters would be able to predict the "random" numbers generated by the algorithm—making the random numbers highly nonrandom indeed. In particular, if encryption keys were based on these "random numbers," anyone knowing the relationship between the two parameters would have a significant head start in guessing the bits of the encryption key. Despite early concerns about a possible cryptographic backdoor—such a backdoor permits a much more efficient search for the key, making the encryption algorithm far less secure than it would otherwise be—in Dual EC_DRBG, the National Institute for Standards and Technology recommended this method for generating random numbers. *See* DAN SHUMOW & NIELS FERGUSON, ON THE POSSIBILITY OF A BACK DOOR IN THE NIST SP800-90 DUAL EC PRNG 2-7 (2007), http://rump2007.cr.yp.to/15-shumow.pdf; *see also* ELAINE BARKER & JOHN KELSEY, U.S. DEP'T. COM., NIST SP 800-90, RECOMMENDATION FOR RANDOM NUMBER GENERATION USING DETERMINISTIC RANDOM BIT GENERATORS 13-67 (2007) (withdrawn in Jan. 2012 and succeeded by SP 800-90A). This was after RSA Security LLC had made the algorithm the default random bit generator in its popular BSafe encryption toolkit. Joseph Menn, *Exclusive: Secret Contract Tied NSA and Security Industry Pioneer*, REUTERS (Dec. 20, 2013, 4:05 PM) https://www.reuters.com/article/us-usa-security-rsa/exclusive-secret-contract-tied-nsa-and-

Principle; the security of the system lies in the security of the key. But the NSA effort meant that the bits of the key were predictable and thus that any information encrypted through that system could be easily decrypted by the NSA.

In principle, the communication would, however, be secure against other interceptors, which did not know the relationship between a crucial two parameters—and thus could not easily guess the random bits and determine the encryption key. This situation did not last. A Juniper firewall used the DUAL_EC_DRBG random number generator. Apparently, someone switched the crucial parameters in the standard DUAL_EC_DRBG algorithm with other numbers; possibly, this allowed the party who did this to have its own back door.[130] But simple testing would not detect the problem. The output would, after all, appear to be properly encrypted.

But programmers had added a second back door to the Juniper firewall[131] that allowed anyone who knew a secret password to log into the firewall as a privileged user.[132] The password was "<<<

---

security-industry-pioneer-idUSBRE9BJ1C220131220. The algorithm was widely used, including in such applications as SSL/TLS, the protocol for securing web transmissions. Matthew Greene, *The Many Flaws of Dual EC-DRBG*, A FEW THOUGHTS ON CRYPTOGRAPHIC ENG'G (Sept. 18, 2013), https://blog.cryptographyengineering.com/2013/09/18/the-many-flaws-of-dualecdrbg/ [https://perma.cc/6YPJ-F4BA].

[130] Stephen Checkoway et al., *A Systematic Analysis of the Juniper Dual EC Incident*, *in* PROCEEDINGS OF THE 2016 ACM SIGSAC CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY 468. The full story is vastly more complicated and includes other back doors and bugs. "The Dual_EC generated initial output that was supposed to then be run through the ANSI generator. The output from the second random generator would theoretically cancel out any vulnerabilities that were inherent in the Dual_EC output. . . . Except Juniper's system contained a bug, according to Willem Pinckaers, an independent security researcher in the San Francisco area who examined the system with Weinmann. Instead of using the second generator, it ignored this one and used only the output from the bad Dual_EC generator." Kim Zetter, *Researchers Solve Juniper Backdoor Mystery; Signs Point to NSA*, WIRED (Dec. 22, 2015, 1:29 AM), https://www.wired.com/2015/12/researchers-solve-the-juniper-mystery-and-they-say-its-partially-the-nsas-fault/ [https://perma.cc/2MEU-5EYW].

[131] Sean Gallagher, *Researchers Confirm Backdoor Password in Juniper Firewall Code*, ARS TECHNICA (Dec. 21, 2015, 1:15 PM), https://arstechnica.com/information-technology/2015/12/researchers-confirm-backdoor-password-in-juniper-firewall-code/ [https://perma.cc/H7YQ-GNHT].

[132] Checkoway et al., *supra* note 131, at 468. It is not known if the two back doors were inserted by the same party or by different parties. Zetter, *supra* note 131.

%s(un='%s') = %u," a value that resembles a string that might be used for debugging.[133] There is no conceivable testing regimen that would have found this flaw.

There are still other ways that programming bugs may lay hidden. Sometimes, bugs can be very rare or highly unlikely because they are timing-dependent. Consider the space shuttle bug.[134] The details are complex; let it suffice to say that it would only occur one out of sixty-seven times the shuttle's computers were booted.[135] The AT&T network flaw would only be triggered when certain events occurred within 1/100 of a second of each other.[136] Testing would be very unlikely to uncover this issue.

There have been inexplicable problems in forensic software too. In one case in Pennsylvania, three different experts concluded there was a DNA match to the defendant, but their probability estimates—all derived from software—differed wildly.[137] It is impossible to understand such differences without access to the source code, nor, for that matter, to know if the problem is a different mathematical model or simply code that implements the model incorrectly.

The limitations of testing have, of course, led to the development of other methods for finding and eliminating software bugs. Those have helped but have not solved the problem. In a previous work, several of us summarized the issue this way:

> We will not recount the myriad techniques other than testing that have been tried in an effort to eliminate bugs;

---

[133] Gallagher, *supra* note 131*; see also*, *infra* Section II.D. (discussing programmer blindness).
[134] *See* Garman, *supra* note 102.
[135] *Id.* at 9.
[136] *See* STERLING, *supra* note 104, at 48.
[137] Commonwealth v. Foley, 38 A.3d 882, 887 (2012) ("The experts differed in their estimates of the probability that someone other than Foley would possess DNA matching the DNA found in the sample—Conway testified that the probability that another Caucasian could be the contributor was 1 in 13,000; Dr. Cotton testified that the probability was 1 in 23 million; and Dr. Perlin testified that it was 1 in 189 billion."); *see also* Stephanie J. Lacambra et al., *Opening the Black Box: Defendants' Rights to Confront Forensic Software*, CHAMPION, May 2018, at 28, https://www.eff.org/files/2018/07/30/champion_article_-_lacambra_forensic_software_may_2018_07102018.pdf.

> let it suffice to say there have been many. These include formal mathematical methods, better programming and debugging tools, different organizational and procedural schemes, improved programming languages, and more. Many of these ideas have helped, but none have proved a panacea. The ability to produce error-free code is the Holy Grail of systems development: heavily desired but unattainable.[138]

In summary, software as it exists today (and likely for the foreseeable future) will always have bugs. Consequently, criminal defendants need access to source code to safeguard their constitutional rights.

D. Adversarial Audits

In light of the axiomatic notion that software will always have bugs, the question then becomes how courts should address this problem. Our basic thesis is that adversarial audits—examination and testing of software by defendants—is necessary for a fair trial. Put another way, we assert that outside testing will find flaws that the vendor did not. Why should this be?

In one sense, this is a statement that does not require proof. The American legal system is fundamentally based on the premise that the adversarial system is the best way to reach the truth. Indeed, as legal scholar Ralph Grunewald has written, "[t]he Supreme Court shares the view that facts are best proven dialectically through a complex process of persuasion and holds that truth 'is best discovered by powerful statements on both sides of the question.'"[139] Furthermore, "[t]he very premise of our adversary system of criminal justice is that partisan advocacy on both sides of a case will best promote the ultimate objective that the guilty be convicted and the innocent go free."[140]

---

[138] Bellovin et al., *supra* note 107, at 28.
[139] Ralph Grunewald, *Comparing Injustices: Truth, Justice, and the System*, 77 ALB. L. REV. 1139, 1156 (2014) (quoting United States v. Cronic, 466 U.S. 648, 655 (1984)).
[140] Herring v. New York, 422 U.S. 853, 862 (1975).

Even in technical matters, one side's expert witness testimony is rebutted by cross-examination and the other side's experts, rather than having the judge appoint and question a neutral witness. There is no *a priori* reason why software issues should be handled differently. There are also several technical reasons to treat software witnesses in the same manner as human expert witnesses. These include the inability of programmers to see their own errors, the fact that the program specifications[141] themselves may be in error (and thus although the program matches the specifications, the program does not do what it should be doing), inadequacy of testing data, and, finally, the proof history: too many programs viewed as correct have been shown to function incorrectly when subjected to adversarial audits that have uncovered serious errors. We discuss each of these phenomena in turn.

The first is a phenomenon we will call "programmer blindness." Just as writers are often bad at proofreading their own text, programmers are bad at reading their own code. Their eyes will skip over errors. This deficiency is often countered by peer review when someone else reads over the code before it is put into the production system,[142] but even that is not a fail-safe method. It is often the case that peers are not truly independent reviewers because programmers often have similar training—and thus tend to make the same mistakes.[143]

Just as there are excellent editors and proofreaders, there are also people who are skilled at avoiding conventional assumptions about software. This ability is at the heart of computer security, whether one is trying to protect a system or break into it.[144] Few companies have such people on their development teams, but a diligent defense attorney could and should seek one out. Indeed, as we increasingly rely on software for providing evidence and making decisions, such a person should be part of any criminal defense team.

---

[141] *See* discussion of erroneous specifications *infra* Section II.D.
[142] *See, e.g.*, Peter C. Rigby & Christian Bird, *Convergent Contemporary Software Peer Review Practices*, *in* PROCEEDINGS OF THE 2013 9TH JOINT MEETING ON FOUNDATIONS OF SOFTWARE ENGINEERING 202, 203 (2013).
[143] *See generally* John C. Knight & Nancy G. Leveson, *An Experimental Evaluation of the Assumption of Independence in Multiversion Programming*, 12 IEEE TRANSACTIONS ON SOFTWARE ENG'G 96, 97 (1986).
[144] *See generally* STEVEN M. BELLOVIN, THINKING SECURITY (2016).

Second, software testing is done according to the program's requirements and specifications.[145] In other words, a party external to the programming team defines the objective: what the inputs and outputs should look like, how errors should be handled, how fast the program must run, etc. The tester's job is to take each requirement and verify that it is met. Often, though, the specifications themselves are faulty. In one study of fifty actual security errors, researchers found that twenty two of them were due to errors in the requirements or specifications.[146] Ordinary testing will not uncover such problems because the program matches the specifications; the problem is that the specifications themselves are incorrect.[147]

The third reason is the nature of testing in the software development process. Software testing is not random; rather, the tester carefully crafts inputs. This testing is done in two ways. The older type is requirements-

---

[145] This is well understood by practitioners, though not always discussed in the academic literature. *But see* Gilles Bernot et al., *Software Testing Based on Formal Specifications: A Theory and a Tool*, 6 SOFTWARE ENG'G J. 387, 387 (1991) ("With the emergence of formal specification languages, it becomes possible to also start from the specification to define some testing strategies in a rigorous and formal framework.").

[146] *See* Carl E. Landwehr et al., *A Taxonomy of Computer Program Security Flaws*, 26 ACM COMPUTING SURV. 211, 216 (1994).

[147] Because the specifications for commercial programs are rarely published, and because these specifications are often extremely technical, we generally learn of such errors only when a security hole or serious bug is found that can be traced back by outsiders to a specification. A recently discovered flaw in Microsoft Windows—a flaw that is more than 20 years old—provides just such an example. In Windows, a program running in one window can send a message to a program running in another window. These messages can be things like "resize yourself," "terminate,", "open this URL," "here is some input," and more. Since some programs have more privileges than others—software installers need high privileges to change the system, while browsers (which are vulnerable to nasty web pages) have fewer permissions than normal—the ability to send such messages is normally restricted. However, Microsoft implemented the "Text Services Framework" to handle such functionality as Chinese language input. Obviously, a multilingual input program must be able to talk to all other running programs, so there were no restrictions for any such messages. This in turn allowed for improper messages to be sent by an attack program that pretended to be part of this framework, which in turn led to security problems. *See* Jim Salter, *A Look at the Windows 10 Exploit Google Zero Disclosed this Week*, ARS TECHNICA (Aug. 15 2019, 6:45 AM), https://arstechnica.com/information-technology/2019/08/a-look-at-the-windows-10-exploit-google-zero-disclosed-this-week/ [https://perma.cc/9ZXF-XBPB].

based: try inputs that stress or exceed requirements.[148] This approach tries to find what are known as "boundary conditions," which are places where the behavior of the program is likely to change.[149] For example, if a certain input field is allowed to be 200 characters long, a tester might try 199, 200, 201, and 2000 character inputs. Similarly, because 2,147,483,647 is the largest number some programs can handle, 2,147,483,648 is a useful test.[150]

Another testing strategy is code-based. There are certain technical aspects of the code that suggest certain inputs should be tried. For example, it is desirable to exercise as much of the program as possible. A mechanism known as a "code coverage tool"[151] tells the tester which lines of the software have or have not been exercised. Referring back to our subtraction example,[152] such a tool could verify that both branches of the "if" statement were tested. However, modern software is so complex that complete coverage during testing is not possible because there are just too many possibilities. In a testing trial, though, certain input values might be fixed to allow testers to vary other inputs in a way that would not have been feasible during the development process.

The Draeger Alcotest 9510 breathalyzer provides an example of how a code-based testing strategy affects criminal defendants. According to

---

[148] *See* AMMANN & OFFUTT, *supra* note 116, at 150 ("The input *domain* is defined in terms of the possible values that the input parameters can have. . . . The tester does not need to understand the implementation; everything is based on a description of the inputs.").

[149] The optimal strategy for generating good test data is in fact an active research area. *See, e.g.*, Mats Grindal et al., *Combination Testing Strategies: A Survey*, 15 SOFTWARE TESTING, VERIFICATION, AND RELIABILITY 167, 167 (2005) ("A literature search has revealed 16 different combination strategies described through more than 40 papers published between 1985 and 2004.").

[150] Generally, modern computers use 32-bit integers. One bit is used to indicate positive or negative; the range of integer values therefore ranges from -2,147,483,648 to 2,147,438,647. *See* Nickolas T. Lanza, *How Do You Get the Maximum and Minimum Values for Integer Data Types Based on the Operating System?*, MEDIUM (Jan. 29, 2018), https://medium.com/@nickolasteixeira/how-to-explain-to-my-wife-what-i-do-how-do-you-get-the-maximum-and-minimum-values-for-integer-befdc263a3a2 [perma.cc/9EJ2-26F7].

[151] *See, e.g.,* Mustafa M. Tikir & Jeffrey K. Hollingsworth, *Efficient Instrumentation for Code Coverage Testing*, 27 SIGSOFT SOFTWARE ENG'G NOTES 86, 86 (2002). The purpose of testing is, of course, to ensure that the code is correct. If test cases do not exercise certain lines of code, then the test cases cannot have verified whether those lines of code are correct.

[152] *See supra* Section II.A.

press reports, the unit is sensitive to variations in the outside temperature.[153] An adversarial examination of such a device could test it at the observed temperature during the police stop, rather than what was available in the vendor's test lab.

The fourth reason why software issues should be treated differently is purely empirical. In other scenarios, adversarial testing has found flaws in software that had been certified by outside parties. The best-documented example is voting machines, where outside auditors have *always* found flaws.

One of the first published independent audits of election systems was of leaked code to a Diebold electronic voting system.[154] The authors' conclusion, after analysis, was blunt: "Our analysis shows that this voting system is far below even the most minimal security standards applicable in other contexts."[155] They also noted problematic aspects that could only be learned by looking at the complete source code.[156] For example, they found no indications that any formal requirements documents or ticket tracker were used even though both of these are, as noted, part of most structured development processes.[157]

Another well-known example is California's "top-to-bottom review" of the electronic voting machines that were to be used in its 2008 elections. Independent teams were contracted to examine the system source code for vulnerabilities that might not have been discovered in the certification process.[158] The results cast significant doubt on the

---

[153] *See* Zack Whittaker, *Researchers Say a Breathalyzer Has Flaws, Casting Doubt on Countless Convictions*, ZDNET (May 10, 2018), https://www.zdnet.com/article/draeger-breathalyzer-breath-test-convictions/ [https://perma.cc/6QER-ZKJN].
[154] T. Kohno et al., *Analysis of an Electronic Voting System*, *in* PROCEEDINGS OF THE IEEE SYMPOSIUM ON SECURITY AND PRIVACY 27, 28 (2004), available at https://ieeexplore.ieee.org/document/1301313.
[155] *Id.* at 27.
[156] *Id.* at 37.
[157] *Id.* at 34.
[158] Cal. Secretary of State, Top-to-Bottom Review of Electronic Voting Systems Certified for Use in California Elections, (draft for public comment Mar. 22, 2007), https://www.sos.ca.gov/elections/ovsta/frequently-requested-information/top-bottom-review [https://perma.cc/2DM3-F2GJ] (archiving documents related to the review). The review was initiated by then-Secretary of State Debra Bowen, pursuant to her responsibilities

integrity of every one of the systems. A review of one evaluated system explained that "[v]irtually every important software security mechanism is vulnerable to circumvention."[159] Moreover, the previous certification processes[160] failed to catch bugs that one would expect to have been caught: "There is evidence in the documentation of vendor-initiated source code reviews as part of the independent testing process, however the broken cryptography architecture indicates that more audits by qualified security experts are required."[161]

Subsequent adversarial examinations of voting systems have shown similar results, as summarized by author Matt Blaze in his 2017 Congressional testimony, which reported that every current voting system examined over the course of a weekend at the DEFCON hacking conference was found to suffer from exploitable vulnerabilities due to unpatched or undetected software defects.[162]

Computerized voting systems provide interesting parallels to the software systems used to collect and process evidence in criminal cases. There is broad consensus among elections experts that modern software

---

in accordance with California law. *Id.*

[159] MATT BLAZE ET AL., SOURCE CODE REVIEW OF THE SEQUOIA VOTING SYSTEM 82 (2007), https://votingsystems.cdn.sos.ca.gov/oversight/ttbr/sequoia-source-public-jul26.pdf.

[160] There are no national standards for the certification of voting systems. Each state jurisdiction sets its own requirements, which are often administered on an ad hoc basis. California requires certification by the Secretary of State, Cal. Elec. Code § 19006(a) (West 2014) ("All voting systems be certified or conditionally approved by the Secretary of State, independent of voluntary federal qualification or certification, before they are used in future elections to ensure that the voting systems have the ability to meet accuracy, accessibility, and security standards."). The Federal standards are voluntary. *See* Help America Vote Act of 2002, 52 U.S.C. § 20961(b)(1) (2002) ("The Development Committee shall assist the Executive Director of the Commission in the development of the voluntary voting system guidelines"); *id.* § 21001(d) ("Nothing in this subpart may be construed to require a State to implement any of the voluntary voting system guidelines or any of the voluntary guidance adopted by the Commission with respect to any matter as a condition for receiving a requirements payment.").

[161] BLAZE ET AL., *supra* note 159, at 42.

[162] *Cybersecurity of Voting Machines: Hearing Before the H. Comm. on Oversight and Gov't Reform, Subcomm. on Info. Tech., & Subcomm. on Intergovernmental Affs.,* 115 Cong. 11 (2017) (statement of Matt Blaze, Assoc. Professor, U. Pa.).

systems are, by virtue of their design, too complex and unreliable to be relied upon for determining the outcomes of civil elections.[163]

The consensus that software is inherently unreliable and suspect has led elections experts to advocate for election system designs that do not depend on software for the correctness of election outcomes. A widely recognized statement of this criteria was proposed by computer scientist Ronald Rivest as Software Independence.[164] In particular:

> A voting system is strongly software-independent if an undetected change or error in its software cannot cause an undetectable change or error in an election outcome, and moreover, a detected change or error in an election outcome (due to change or error in the software) can be corrected without re-running the election.[165]

At first blush, a requirement for software independence might appear to preclude the use of any software (or computers) in elections altogether. But that is not necessarily the case. Software independence simply requires that any software-based system be designed in a way that allows for recovery of correct results even if the software had failed in some way. For example, a voting system that employs paper ballots (marked by a voter) might perform an initial tally by computerized scanners, but still retain the ability to use the original paper ballots (which reflect the true intent of the voters) for recounts and audits, which can be conducted by hand and without dependence on the software.[166] Such a system uses software, yet is still software independent under this definition.

---

[163] *See generally* NAT'L ACADS. OF SCIS., ENG'G, AND MED., SECURING THE VOTE: PROTECTING AMERICAN DEMOCRACY (2018), https://www.nap.edu/catalog/25120/securing-the-vote-protecting-american-democracy [https://perma.cc/B9PU-FDEQ].
[164] *See generally* Ronald L. Rivest & John P. Wack, *On the Notion of 'Software Independence' in Voting Systems*, 366 PHIL. TRANSACTIONS SERIES A, MATHEMATICAL, PHYSICAL, AND ENG'G SCIS. 3759 (2008), https://people.csail.mit.edu/rivest/RivestWack-OnTheNotionOfSoftwareIndependenceInVotingSystems.pdf.
[165] *Id*. at 3763.
[166] *See id.*

Flaws in voting systems threaten the integrity of an election; flaws in DNA analysis software can put the wrong person in jail. In one widely publicized incident, defense attorneys in New York City were granted access to the source code to the Forensic Statistical Tool ("FST") developed by the office of the chief medical examiner.[167] On a motion from ProPublica, the software was made publicly available and affidavits from a defense expert witness were unsealed.[168] He explained that by examining the source code he was able to learn of behavior in the match calculation that was never documented anywhere:

> An instance of undocumented behavior of the FST program is noted in the calculation of likelihood ratios performed in "Comparison.cs". A routine included in the source code file ("Comparison" class), named "CheckFrequencyForRemoval" appears to perform the following behavior:
> 1. Check all replicate (evidentiary) genotypes for any locus that contains alleles whose frequency sums to ≥ 0.97 in any of the four subpopulations ("Asian," "Black," "Caucasian," and "Hispanic).
> 2. Remove these loci from the likelihood ratio calculations.
> During this review, I encountered no notice, either intended or actual, provided to the user of FST that any loci were removed from the likelihood ratio calculation. I found no indication that this behavior is intended during my examination of FST-related publications and the FST Validation materials.[169]

---

[167] Lauren Kirchner, *Federal Judge Unseals New York Crime Lab's Software for Analyzing DNA Evidence,* PROPUBLICA (Oct. 20, 2017, 8:00 AM), https://www.propublica.org/article/federal-judge-unseals-new-york-crime-labs-software-for-analyzing-dna-evidence [https://perma.cc/4BPK-RUJD].

[168] Lauren Kirchner, *Traces of Crime: How New York's DNA Techniques Became Tainted,* N.Y. TIMES (Sept. 4, 2017), https://www.nytimes.com/2017/09/04/nyregion/dna-analysis-evidence-new-york-disputed-techniques.html [https://perma.cc/8A7T-MPFU].

[169] Memorandum in Support of Defendant's Motion in Limine, Exhibit C at 20, United States v. Johnson, S.D.N.Y 2016 (1:15-cr-00565-VEC).

Furthermore, the behavioral change in the match calculation was done after the validation study of the FST package: "There is at least one indication of a deviation in behavior between the version of FST used during its validation study and the version of FST provided to me." [170]

Beyond that example, there may be other lurking problems. Deviation from good coding practice is a red flag. Systems engineer Nathaniel Adams wrote:

> Martin Fowler describes code smells as, "A code smell is a surface indication that usually corresponds to a deeper problem in the system." In this sense, a smell is not a defect in itself but is a deviation from good coding practices, which can indicate underlying software defects. Coding conventions and style guides purposely prevent code smells. Coding practices such as writing long routines or complex classes can obfuscate underlying issues due to the difficulty of comprehending large segments of code.[171]

He goes on to note that "[t]he FST source code presents a number of basic smells such as routine length and complex classes."[172] In other words, the style of coding suggests that the program is below normal professional standards and may have other, not yet detected problems. It is extremely difficult to detect such "smells" without access to source code.

We thus see that, as an empirical matter, adversarial audits can uncover relevant issues in software relied upon in prosecutions. We are not aware of any studies of why this should be—why independently certified software should prove so vulnerable when subjected to an adversarial audit. We speculate that, in the absence of such audits, there is little basis for comparison other than price, and hence little opportunity for market forces to produce improvement. This is similar

---

[170] *Id.* at Exhibit A, at 9.
[171] *Id.* at Exhibit C, at 13 (citing Martin Fowler, *Code Smell,* MARTINFOWLER.COM (Feb 9, 2006), https://martinfowler.com/bliki/CodeSmell.html [https://perma.cc/9ZRP-AHG5]).
[172] *Id.*

to code developed in a competitive market, where base functionality is all-important and the winner is often the first player to the market: security often takes a back seat.[173] If price is in fact the primary differentiator, vendors' primary incentives are to reduce their costs, e.g., by reducing the cost of the entire software development process.

We do not mean to suggest that adversarial audits are a panacea. Nevertheless, experience shows that they do help.

E. Machine Learning

Machine learning algorithms raise a different set of issues.[174] Machine learning algorithms, popularly known as "artificial intelligence" or "AI," are at the heart of many of today's technologies, including all major search engines.[175] However, these algorithms have to be "trained," i.e., fed known initial data.[176] If the training data is deficient, the output of the algorithm will be incorrect.

This problem has arisen a number of times. Professor Ignacio Cofone wrote:

> An illustrative employment example is a machine learning system that Amazon recently developed to rank job candidates. The system displayed a significant bias against female candidates, justifiably triggering public outcry. Because the algorithm was trained using Amazon's existing hiring data under the idea that Amazon's current employee choices are a good proxy for

---

[173] NAT'L RSCH. COUNCIL ET AL., TRUST IN CYBERSPACE 67 (Fred B. Schneider ed., 1998) ("For software-intensive products, early arrival in the marketplace is often critical to success in that marketplace. This means that software development practice becomes distorted to maximize functionality and minimize development time, with little attention paid to other qualities. Thus, functionality takes precedence over trustworthiness.").

[174] See generally Steven M. Bellovin et al., *Privacy and Synthetic Datasets,* 22 STAN. TECH. L. REV. 1 (2019), for a description of machine learning.

[175] Gabriel Jiménez, *Artificial Intelligence Applications in Search Engines,* MEDIUM (Dec. 18, 2018), https://medium.com/aimarketingassociation/artificial-intelligence-applications-in-search-engines-437c57f8b265 [https://perma.cc/H77S-A55W].

[176] Bellovin et al., *supra* note 174, at 4.

> Amazon's desired employee choices, the algorithm reflected existing hiring practices. These practices, however, to the surprise of the algorithm's programmers, ended up being sexist.[177]

Microsoft had similar problems with its Tay "chatbot." A chatbot is designed to respond to users who send it messages, learning as it goes. However, Tay learned from the worst of the Internet, going "full Nazi" in less than 24 hours.[178] Microsoft realized that the problem was training data, saying "[i]n an emailed statement given later to Business Insider . . . : 'The AI chatbot Tay is a machine learning project, designed for human engagement. As it learns, some of its responses are inappropriate and indicative of the types of interactions some people are having with it.'"[179]

In other words, for software based on machine learning, defendants should have access not only to the source code but also to the training data. Without that data, it is impossible to understand what output a program might produce. In explaining problems with automated analysis of computer programs, Professor Kroll et al. wrote:

> Perhaps the most obvious approach is to disclose a system's source code, but this is at best a partial solution to the problem of accountability for automated decisions. The source code of computer systems is illegible to nonexperts. In fact, even experts often struggle to understand what software code will do: inspecting source code is a very limited way of predicting how a computer program will behave. Machine learning, one increasingly popular approach to automated decision making, is particularly ill-suited to source code analysis because it

---

[177] Ignacio Cofone, *Algorithmic Discrimination Is an Information Problem,* 70 HASTINGS L.J. 1389, 1397-98 (2019).

[178] James Vincent, *Twitter Taught Microsoft's AI Chatbot to Be a Racist Asshole in Less Than a Day*, VERGE (Mar. 24, 2016, 6:43 AM), https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist [https://web.archive.org/web/20190312132945/https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist].

[179] *Id.*

> involves situations where the decisional rule itself emerges automatically from the specific data under analysis, sometimes in ways that no human can explain.[180]

The same, of course, is true for evidentiary software.

III. Source Code and Constitutional Issues

Software and algorithms now play a critical role in many criminal prosecutions. Depriving such defendants access to the underlying code risks depriving defendants of their rights. For example, there are many ways for criminal defendants to challenge breathalyzer results, including attacks of the device's margin of error, its calibration, and the breath sample itself.[181] However, there are not many examples in which courts analyzed the use of algorithms and software, let alone whether any errors in specific program should be the basis for a challenge by criminal defendants. Indeed, courts have often been challenged by new technological developments.[182]

Some state courts have received challenges to convictions using various types of breathalyzers.[183] The defendants typically argued for the ability to examine the source code for these devices. They generally make two types of arguments. First, they argue that the source code must be provided pursuant to *Brady v. Maryland*, in which the Court held that prosecutors must provide defendants with all exculpatory evidence.[184] Second, defendants assert the failure to provide the source code violates the Confrontation Clause of the Sixth Amendment.[185] Third, they

---

[180] *See* Joshua A. Kroll et al., *Accountable Algorithms*, 165 U. PA. L. REV. 633, 638 (2017).
[181] Paul A. Clark, *The Right to Challenge the Accuracy of Breath Test Results Under Alaska Law*, 30 ALASKA L. REV. 1, 6-8 (2013).
[182] *See generally* Riley v. California, 573 U.S. 373 (2014); Kyllo v. United States, 533 U.S. 27 (2001).
[183] Cowley & Silver-Greenberg, *supra* note 39.
[184] Brady v. Maryland, 373 U.S. 83, 87 (1963) (holding that the prosecution must provide exculpatory evidence to a defendant). The case involved a murder committed by one of two men being tried in separate trials; the prosecution had a confession from the murderer but failed to share this evidence with the other.
[185] U.S. CONST. amend. VI; *see also* People v. Umpierre, 951 N.Y.S.2d 382, 383 (Sup. Ct. 2012); State v. Lindner, 252 P.3d 1033, 1036 (Ariz. Ct. App. 2010) (finding against the

maintain that denial of access to source code constitutes a due process violation.[186] We discuss each of these approaches in turn.

A. *Brady* Violations

The Supreme Court in *Brady* was very clear: the prosecution must provide exculpatory evidence to a defendant. In *Brady*, two people, the petitioner and another person, were charged with first degree murder, but tried separately.[187] At trial, Brady conceded his guilt in the murder, but argued that he did not kill the victim.[188] In preparation for trial, his defense attorney sought the extrajudicial statements by the other person.[189] The prosecution withheld one in which he admitted to committing the killing.[190] The Supreme Court held "that the suppression by the prosecution of evidence favorable to an accused upon request violates due process where the evidence is material either to guilt or to punishment, irrespective of the good faith or bad faith of the prosecution."[191]

In cases where software provides evidence, courts have had varying interpretations of *Brady*. In some sense, the issue is what constitutes "suppression"—for if the prosecution does not have access to the source code behind evidence in the case, the prosecution cannot be considered to be suppressing such evidence.

The 2008 decision *City of Fargo v. Levine* before the North Dakota Supreme Court involved Glenn Levine, whom the police had arrested for driving under the influence based in part on the results of an Intoxilyzer.[192] Levine retained Dr. Robert Howard as an expert because he was "chief operating officer of Medscan Laboratory and Advanced

---

defendant's assertion that failure to provide breathalyzer source code violates the Confrontation Clause).

[186] *See Lindner*, 252 P.3d at 1034, n.1 (stating that defendant also raised due process right to discover source code); *Brady*, 373 U.S. at 87.

[187] *Brady*, 373 U.S. at 84.

[188] *Id.*

[189] *Id.*

[190] *Id.*

[191] *Id.* at 87.

[192] 747 N.W.2d 130, 131-32 (N.D. 2008).

Drug Testing, a company providing third-party drug and alcohol testing for employers."[193] In his work, Howard was familiar with the Intoxilyzer as well as having professional experience writing source code.[194] Howard explained that "the source code used in Intoxilyzer machines varies among jurisdictions and, when purchasing a machine, the buyer specifies certain types of programming."[195] He testified that without the specific Intoxilyzer's source code, he could not properly analyze Levine's results, but the trial court nonetheless denied the motion to compel production of the source code.[196]

The North Dakota Supreme Court explained that the prosecutor neither possessed nor controlled the Intoxilyzer source code.[197] Levine had furthermore failed to provide any evidence that that city possessed or controlled the source code in its Intoxilyzer.[198] The state supreme court concluded that the prosecutor did not commit a *Brady* violation by failing to produce source code that was not in the city's possession.[199]

In 2008, the Supreme Court of Montana also heard a challenge to a denial of a request for the source code for the Intoxilyzer 8000.[200] In *Peters*, CMI, the manufacturer of the Intoxilyzer 8000, raised concerns about trade secrets in Kentucky state court.[201] However, CMI agreed to provide the defense expert access to the source code in its Kentucky corporate headquarters subject to a protective order that required the defense expert to destroy his computer hard drive when he completed the review and left CMI headquarters.[202] Although the defense expert did not object to signing a protective order, he considered the CMI

---

[193] *Id.* at 132.
[194] *Id.*
[195] *Id.*
[196] *Id.*
[197] *Id.* at 133.
[198] *Id.*
[199] *Id.*
[200] State v. Peters, 264 P.3d 1124, 1127 (Mont. 2011).
[201] *Id.* at 1127; *see also* Wexler, *supra* note 11, at 1419 (citing Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 490 (1974)) ("The U.S. Supreme Court has even recognized that the 'substantial risk' of undetectable theft of trade secrets may be socially useful in that it encourages innovators to seek patents, which require disclosure.").
[202] *Peters*, 264 P.3d at 1129.

agreements to be too restrictive and declined to sign them.[203] Specifically, the defense expert explained that he feared signing the CMI agreements because the company "'threaten[ed] my license to practice law if I violated'" the agreements.[204] He further explained that "'it's not that I won't sign it. *I don't believe I can sign it—without subjecting myself to sanctions*.'"[205] In other words, the criminal defendants were not able to present any defense to the charge against him because CMI essentially cowed the expert witness by threatening his livelihood and liberty.

In *Peters*, the Supreme Court of Montana explained that the prosecution did not have the Intoxilyzer 8000 source code.[206] It determined that because CMI did provide the defense expert reasonable access to the source code the district court did not abuse its discretion in limiting the information that the prosecution was required to provide.[207] Consequently, the Supreme Court of Montana declined to address claims based on either a *Brady* violation or the Confrontation Clause.[208] However, this approach ignored the legitimate concerns raised by the defense expert.

Meanwhile in 2012, in *State v. West*, the Court of Appeals of Oregon considered Donald West's appeal of his drunk driving conviction.[209] Defense counsel challenged the results of an Intoxilyzer 8000.[210] Oregon law establishes a presumption of admissibility of the breathalyzer results provided that they comply with methods approved by the Oregon Department of State Police.[211] Rules issued by Oregon's State Police explicitly authorize the use of the Intoxilyzer 8000.[212]

---

[203] *Id.*
[204] *Id.*
[205] *Id.* (emphasis added).
[206] *Id.* at 1131.
[207] *Id.* at 1132.
[208] *Id.*
[209] 279 P.3d 354, 356 (Or. Ct. App. 2012).
[210] *Id.*
[211] OR. REV. STAT. § 813.160(1) (2011); *see also West*, 279 P.3d at 356.
[212] OR. ADMIN. R. 257-030-0120 (2006); *see also West*, 279 P.3d at 357.

In *West*, the appellate court explained that the prosecution had met its *Brady* obligations because it provided all information that it had regarding the Intoxilyzer 8000.[213] Defense counsel had furthermore failed to demonstrate that the source code was material, but instead simply posited that it "'could be used to show the manufacturer's bias toward producing evidence that will lead to convictions.'"[214] Consequently, the appellate court determined that "*Brady* is not authority for a defendant obtaining evidence of unknown import to test whether it helps or hurts his case."[215]

In 2013, in *State v. Marino*, the Court of Appeals of North Carolina heard an appeal challenging a jury verdict finding Jory Marino guilty of driving while intoxicated.[216] Defense counsel filed a motion seeking *Brady* material, which the trial judge essentially granted ordering that the prosecution provide defense counsel "with 'all downloaded and non-downloaded data in its possession that was generated from [the] Intoximeter [used to analyze defendant's breath.]'"[217] But the trial judge did not order the production of the Intoximeter's source code.[218] This meant that the defense was not in a position to determine if the Intoximeter breath analysis software worked correctly.[219]

Pursuant to *Brady*, a "prosecutor is not required to deliver his entire file to defense counsel, but only to disclose evidence favorable to the accused that, if suppressed, would deprive the defendant of a fair trial."[220] Missing from the decision is any clarification of who decides what suppressed information might deprive the defendant of a fair trial. How do they make that determination?

In *Marino*, the appellate court determined that defense counsel had not established that the Intoximeter source code would be favorable or material to his defense regarding either the guilt phase or the punishment

---

[213] *West*, 279 P.3d at 359.
[214] *Id.*
[215] *Id.*
[216] 747 S.E.2d 633, 634 (N.C. Ct. App. 2013).
[217] *Id.* at 635.
[218] *Id.*
[219] *See id.*
[220] United States v. Bagley, 473 U.S. 667, 675 (1985).

phase of his prosecution.[221] The court moreover characterized defense counsel's interest in this source code and the possibility that it could result in exculpatory evidence to be speculative at best.[222] The court did not acknowledge—and perhaps did not know—that software often lies, not out of programmers' malfeasance, but as a result of design or coding errors. In *Marino*, the appellate court noted that *Brady* evidence must be favorable or material to guilt or punishment.[223] However, it ignored the significant factor that the prosecution must actually possess evidence in order to be able to determine whether the evidence is favorable.[224] Without the code to examine, the defense is not a position to make this determination either. The court is effectively setting up a Catch-22 proposition.

The fact that prosecutors often do not have any copies of the breathalyzer source code is a significant issue regarding *Brady* concerns. In Minnesota, a trial judge ordered the prosecution to produce to the defense attorney "the complete computer source code for the operation of the Minnesota model of the Intoxilyzer 5000 currently in use."[225] Specifically, the trial judge determined that Minnesota "owned the source code for the Intoxilyzer 5000EN model created exclusively for the state."[226]

In response to this order, the Commissioner filed a writ of prohibition with the appellate court seeking to bar the trial judge from enforcing this order.[227] The fact that Minnesota owned part of the source code pursuant to its contract with CMI was significant for the Minnesota Supreme Court's decision to uphold the appellate court.[228] Specifically, the state supreme court rejected the Commissioner's argument that he did not possess or control the source code based on this contractual language.[229] This decision did not concern *Brady* materials because the court

---

[221] *Marino*, 747 S.E.2d at 638.
[222] *Id.*
[223] *Id.*
[224] *See Brady*, 373 U.S. at 87.
[225] *In re* Comm'r of Pub. Safety, 735 N.W.2d 706, 709 (Minn. 2007).
[226] *Id.*
[227] *Id.*
[228] *Id.* at 712.
[229] *Id.*

determined that Minnesota had the source code and should produce it consistent with the trial judge's order. It stands as a contrast to the situations where prosecutors do not have access to the source code.

Defendants have generally failed in obtaining the source code for breathalyzers based on arguments pursuant to *Brady*. For example, they often cannot demonstrate the need for the source code. It is, of course, extremely difficult to prove that the source code has errors if the code is not available for examination. The other wrinkle is that often the prosecution can establish that it does not have the source code either. Consequently, a court cannot very well compel the prosecution to produce something that it does not possess.[230]

B. Confrontation Clause

In addition to *Brady* challenges, criminal defendants may argue that the failure to provide the source code for breathalyzers also violates the Confrontation Clause. Specifically, the Sixth Amendment of the Constitution provides that "[i]n all criminal prosecutions, the accused shall enjoy the right to . . . to be confronted with the witnesses against him."[231]

One might think that this is a straightforward rule, but in recent years it has become anything but. The confusing situation had its genesis in a relatively uncomplicated murder case, *Crawford v. Washington* that, due to issues of admissibility of evidence from the defendant's wife, made its way to the Supreme Court.[232] The evidence in that decision concerned testimony from a living person, but the implications of

---

[230] One problem is that CMI may not have the source code for a given Intoxilyzer. *See* Affidavit of Brian Faulkner at ¶ 5, State v. Bonakoske (Fla. Charlotte County Ct. June 10, 2011) (No. 08-1726-T) (In response to requests for source code, CMI's Manager of Engineering swore that "CMI is unable to produce, with any degree of certainty, each and every section of source code from software revisions that were not put into service by the customer. Consequently, CMI cannot reliably reconstruct version 8100.10 which directly corresponds to software version 8100.10 included in the instrument initially tested, but not utilized, by the Florida Department of Law Enforcement."). Of course, if the source code is non-existent for CMI as well, it begs the question of how it can reliability be assessed at all.

[231] U.S. CONST. amend. VI.

[232] 541 U.S. 36, 36 (2004).

Crawford stretch all the way to software. We discuss the Confrontation Clauses in two parts: (i) how the courts have ruled about expert witness testimony post *Crawford* and (ii) the implications of *Crawford* on evidentiary software.

1. Expert Witness Testimony post-*Crawford*

Interpreting the Confrontation Clause, the United States Supreme Court determined that the clause afforded criminal defendants the right to cross-examine any witnesses who "bear testimony" against the defendant.[233] In *Crawford*, the defendant, Michael Crawford, confronted a man he believed had attempted to rape his wife, Sylvia Crawford.[234] Believing the alleged rapist was reaching for a weapon, Michael Crawford stabbed the man.[235] Sylvia Crawford's statement to the police did not corroborate her husband's claim of self-defense, and Michael Crawford was charged with attempted murder.[236]

The prosecution sought to have Ms. Crawford testify, but Washington's spousal privilege law prevented her from testifying against her husband without his consent.[237] The prosecution was able to introduce the wife's statement because it was admissible pursuant to a hearsay exception.[238] The defendant argued that the statement's admission violated his Confrontation Clause rights.[239] The trial court admitted the statement, finding that it had "adequate 'indicia of reliability.'"[240] After the jury found Crawford guilty, the state appellate court reversed finding that the statement was inadmissible before the Washington Supreme Court reinstated his conviction.[241]

---

[233] *Id.* at 51.
[234] *Id.* at 38.
[235] *Id.* at 38-39.
[236] *Id.* at 39-40.
[237] *Id.* (discussing WASH. REV. CODE § 5.60.060(1) (1994)).
[238] *Id.* (citing State v. Burden, 841 P.2d 758, 761 (Wash. 1992)).
[239] *Id.*
[240] *Id.*
[241] *Id.* at 41.

The United States Supreme Court grappled with whether the admission of Ms. Crawford's out-of-court statement violated the Confrontation Clause because the defendant could not cross-examine the statement.[242] In a unanimous decision, the Court determined that its admission violated Crawford's Sixth Amendment rights.[243] Justice Scalia wrote that the Framers of the Constitution intended the Confrontation Clause to bar these types of out-of-court statements.[244] In finding Ms. Crawford's statement inadmissible, the Court overruled *Ohio v. Roberts*,[245] and determined that analysis of the statement's reliability was inadequate in preserving the Sixth Amendment rights and that courts must address the evidence's testimonial nature.[246]

Evidence deemed to be testimonial may not be introduced against a criminal defendant unless the witness is unavailable *and* the defendant had a previous opportunity to cross-examine the witness regarding the substance of the proposed evidentiary statement.[247] Legal scholar Jennifer Mnookin explained that in *Crawford* a critical issue concerned *whether* "the statement [was] made in circumstances that suggest[s] its likely future relevance as testimony in a criminal [trial]?"[248] Thus, business records, Call Detail Records, web searches, and the like would not be covered by *Crawford* decision's analytical reasoning[249]—but breathalyzer tests and DNA tests are.

In *Crawford*, the Court dealt with testimonial evidence in the case of a human witness whose testimony the prosecution introduced during the trial, but who was not available for cross examination by the defendant during trial.[250] Our interest is in (particular types of) expert-witness testimony, and so we turn to a series of post-*Crawford* Supreme Court

---

[242] *Id.* at 38.
[243] *Id*. at 68-69.
[244] *Id.* at 43-50.
[245] 448 U.S. 56 (1980).
[246] *Id.* at 67-68.
[247] *Id.* at 53-54.
[248] Jennifer Mnookin, *Expert Evidence and the Confrontation Clause After Crawford v. Washington*, 15 J.L. & POL'Y 794 (2007).
[249] *Id.*
[250] *Crawford*, 541 U.S. at 38.

rulings on the admissibility of testimony—or not—by expert-witnesses in which the Court subsequently ruled.

In *Melendez-Diaz v. Massachusetts*, the Supreme Court heard arguments about a prosecution involving an alleged drug dealer.[251] Specifically, the state presented certificates from an in-state laboratory that tested whether the numerous plastic bags containing a white powder the defendant allegedly secreted within a police car as he was being transported to the station were cocaine.[252]

The state laboratory tested all the bags from the scene as well as those located in the police cruiser.[253] At trial, the prosecution introduced the bags of cocaine into evidence along with three certificates of analysis establishing that the bags tested positive for cocaine.[254] Specifically, "[t]he certificates reported the weight of the seized bags and stated that the bags '[h]a[ve] been examined with the following results: The substance was found to contain: Cocaine.'"[255] Pursuant to Massachusetts law, the lab analysts swore before notaries public to the findings in the certificates.[256]

Instead of the laboratory technicians who analyzed whether the packages contained cocaine and the amount of narcotics, the prosecution simply presented the certificates.[257] These certificates were, as required by state law, prepared by the State Laboratory of the Massachusetts Department of Health.[258] The lab technician who prepared the certificates did not appear in court, but the certificates were sworn to by analysts from the lab in front of a notary public.[259] Relying on the requirements put forth in *Crawford,* the defendant, Luiz

---

[251] 557 U.S. 305, 307-08 (2009).
[252] *Id.* at 308. The amount of cocaine was significant, as that would determine the charge. *Id.*
[253] *Id.*
[254] *Id.*
[255] *Id.*
[256] *Id.* (citing MASS. GEN. LAWS ANN. ch. 111, § 12 (West 2006)).
[257] *Id.* at 308-09.
[258] *Id.* at 308.
[259] *Id.*

Melendez-Diaz, objected, claiming that the documents were testimonial in nature.[260] The Court concurred.[261]

Justice Scalia, writing for a majority that included Justices Stevens, Souter, Thomas, and Ginsburg, concluded that Melendez-Diaz's Confrontation Clause right was violated when he was barred from confronting these technicians.[262] Addressing the certificates of analysis, the Court explained that as they were affidavits, "[t]here is little doubt that the documents at issue in this case fall within the 'core class of testimonial statements.'"[263] To the state's argument that the analysts are not subject to confrontation, the Court responded that "[t]he Sixth Amendment guarantees a defendant 'the right to be confronted with the witnesses *against him*.'"[264] Specifically, the Court explained that "Confrontation is one means of assuring accurate forensic analysis . . . designed to weed out not only the fraudulent analysis, but the incompetent as well."[265] The Court cautioned that "[s]erious deficiencies have been found in the forensic evidence used in criminal trials."[266]

One such situation in Massachusetts became public a few years later. A chemist employed in a Massachusetts state drug testing lab, Annie Dookhan, had been falsifying drug reports in an attempt to build an impressive number of tests she was analyzing.[267] In many cases, she failed to run analyses that she reported to the courts.[268] Dookhan pleaded guilty to "27 counts of misleading investigators, tampering with

---

[260] *Id.* at 309.
[261] *Id.* at 329.
[262] *Id.*
[263] *Id.* at 310 (Thomas, J., concurring in part and concurring in the judgment) ("[T]he Confrontation Clause is implicated by extrajudicial statements only insofar as they are contained in formalized testimonial materials, such as affidavits, depositions, prior testimony, or confessions" (quoting *White v. Illinois,* 502 U.S. 346, 365 (1992))).
[264] *Id.* at 313.
[265] *Id.* at 318-19.
[266] *Id.* at 319.
[267] Katie Mettler, *How a Lab Chemist Went from "Superwoman" to Disgraced Saboteur of More Than 20,000 Cases,* WASH. POST (Apr. 21, 2017), https://www.washingtonpost.com/news/morning-mix/wp/2017/04/21/how-a-lab-chemist-went-from-superwoman-to-disgraced-saboteur-of-more-than-20000-drug-cases/?noredirect=on [https://perma.cc/4YV4-TV6Z].
[268] *Id.*

evidence, and filing false reports." [269] Ultimately, prosecutors in eight Massachusetts counties dismissed charges in over 21,000 drug cases as a result.[270]

After *Melendez-Diaz*, the Supreme Court ruled on another case in which lab certificates were introduced, but the technician who prepared it did not appear in court: *Bullcoming v. New Mexico*.[271] The police arrested Donald Bullcoming for DWI after he caused an accident.[272] After he refused to take a breathalyzer, the police obtained a search warrant to obtain a blood sample at a local hospital.[273] However, the blood was analyzed at a state laboratory.[274] At Bullcoming's trial, the prosecution introduced the blood alcohol test results, but the technician who conducted the test was unavailable to testify.[275] Instead, another lab technician who was familiar with the test equipment and procedures testified.[276] The trial court determined that although the analysis of blood-alcohol levels was testimonial, the certifying analyst did not need to appear in court; in-court testimony from another analyst from the same lab had sufficed to meet the requirements of the Confrontation Clause.[277]

Relying on *Melendez-Diaz*, the Supreme Court of New Mexico had already ruled that the lab analysis was testimonial. Despite that determination, the court concluded that the lab report was nonetheless "functionally identical to live, in-court testimony, doing precisely what a witness does on cross-examination."[278] Making clear that testimonial is testimonial, the United States Supreme Court was having none of this argument: "if an out-of-court statement is testimonial, it may not be introduced against the accused at trial unless the witness who made the

---

[269] *Id.*
[270] *Id.*
[271] 564 U.S. 647, 651 (2011).
[272] *Id.* at 652.
[273] *Id.*
[274] *Id.* at 652-53.
[275] *Id.* at 655.
[276] *Id.*
[277] *Id.* at 655-56.
[278] State v. Bullcoming, 226 P.3d 1, 8 (N.M. 2008) (quoting Melendez-Diaz v. Massachusetts, 557 U.S. 305, 308 (2009)).

statement is unavailable and the accused has had a prior opportunity to confront the witness."[279] In an observation that is important in considering evidentiary software, the majority opinion noted that, "[i]n any event, the comparative reliability of an analyst's testimonial report does not overcome the Sixth Amendment bar. This Court settled in Crawford that the 'obviou[s] reliab[ility]' of a testimonial statement does not dispense with the Confrontation Clause."[280]

At its core, the Confrontation Clause safeguards defendants against testimonial hearsay statements.[281] The Supreme Court has explained that "[t]estimonial statements of witnesses absent from trial have been admitted only where the declarant is unavailable, and only where the defendant has had a prior opportunity to cross-examine."[282] Nonetheless, the Court has hesitated in defining all types of testimonial statements.[283] Indeed, in assessing whether a statement is testimonial, the Supreme Court has explained that "courts should look to all of the relevant circumstances."[284] Moreover, in analyzing *Crawford*, the Court noted that the decision "did not offer an exhaustive definition of 'testimonial' statements. Instead, *Crawford* stated that the label 'applied at a minimum to prior testimony at a preliminary hearing, before a grand jury, or at a former trial; and to police interrogations.'"[285] Such opaque standards hardly give courts much guidance.

Legal scholars Jennifer Mnookin and David Kaye have noted that:

> [f]or the most part, [lower courts] were reluctant to read *Crawford* as requiring any significant changes to the preexisting methods for introducing forensic science testimony. In the first years following the *Crawford* decision, most (though certainly not all) trial courts instead endeavored to shoehorn the traditional methods

---

[279] *Bullcoming*, 564 U.S. at 657.
[280] *Id.* at 661 (quoting Crawford v. Washington, 541 U.S. 36, 61 (2004)).
[281] *See* U.S. CONST. amend. VI.
[282] *Crawford*, 541 U.S. at 59.
[283] *See* Davis v. Washington, 547 U.S. 813, 822 (2006).
[284] Michigan v. Bryant, 562 U.S. 344, 369 (2011).
[285] *Crawford*, 541 U.S. at 68.

> for presenting forensic science testimony into this new framework without requiring modifications.[286]

For example, in the *Levine* case, the defendant retained a highly qualified expert who sought the source code, but the court had concluded that, as the government did not have the source code, the prosecutor had not committed a *Brady* violation by failing to provide the code to the defendant.[287] This reasoning was faulty. At issue was not a Brady violation—providing potentially exculpatory evidence in possession of the government to the defendant—but rather a failure to satisfy the requirements of the Confrontation Clause.

The North Dakota Supreme Court determined that its state law allows criminal defendants to cross-examine either the director of the state crime laboratory or one of its employees.[288] However, this decision preceded *Melendez-Diaz* by about a year, and the state supreme court rejected the Confrontation Clause argument because the defendant chose not to call any such witness, but instead pled guilty.[289] In other words, the defendant waived his claim to a Sixth Amendment violation.

In Arizona, state law prevents criminal defendants from challenging the admissibility of breathalyzer results. Specifically, the legislature mandated that "[t]he inability of any person to obtain manufacturer's schematics and software for a quantitative breath testing device that is approved . . . shall not affect the admissibility of the results of a breath test."[290] Thus, when Michael Lindner asserted that the failure to require the prosecution to provide the source code and a witness to cross-examine violated the Confrontation Clause, the Arizona Court of Appeals rejected Lindner's arguments, distinguishing *Melendez-Diaz* as

---

[286] Jennifer Mnookin & David Kaye, *Confronting Science: Expert Evidence and the Confrontation Clause*, 2012 SUP. CT. REV. 99, 105.
[287] City of Fargo v. Levine, 747 N.W.2d 130, 132-33 (N.D. 2008).
[288] *Id.* at 135 (first discussing N.D. CENT. CODE § 39-20-07(5) (2019); and then discussing Berger v. State Highway Comm'r, 394 N.W.2d 678, 686 (N.D. 1986)).
[289] *Id.*
[290] ARIZ. REV. STAT. ANN. § 28-1323(A)(5) (2006).

only requiring the technician who performed the blood tests as opposed to the designer of the equipment used by that technician.[291]

In *Lindner*, the appellate court determined that because the officer who conducted the breath tests appeared at trial it was not necessary to require the breathalyzer's designer to appear also in order to discuss the source code.[292] The court concluded that a constitutional violation did not occur.[293]

In *West*, yet another DWI prosecution, the defendant sought the source code for the Intoxilyzer 8000. The 2012 decision by the Oregon Court of Appeals rejected this argument, finding that there was "an opportunity to cross-examine witnesses and to impeach the state's evidence, including the Intoxilyzer test results." [294] The appellate court reasoned that the admission of the Intoxilyzer's certificates of accuracy prepared by technicians, but without their testimony, did not violate the Confrontation Clause because these certificates were not testimonial in nature.[295]

In *Marino,* a 2012 North Carolina case, the defendant asserted he needed the source code in order to cross-examine "his primary accuser, the Intoximeter."[296] Looking to *Melendez-Diaz*, the North Carolina Court of Appeals explained that the Supreme Court's decision provided defendants with the right to "cross-examine those individuals involved in the production of testimonial documents to be introduced at trial, *such as the technician operating the Intoximeter in the present case*," but held that *Melendez-Diaz* did not provide a basis for the defendants to examine the breathalyzer source code.[297] This reasoning in *West* flies in the face of the *Melendez-Diaz*. This is especially so in light of the fact

---

[291] State v. Lindner, 252 P.3d 1033, 1035-36 (Ariz. Ct. App. 2010) (citing Melendez-Diaz v. Massachusetts, 557 U.S. 305, 310-11 (2009)).
[292] *Id*. at 1036.
[293] *Id.*
[294] State v. West, 279 P.3d 354, 361 (Or. Ct. App. 2012).
[295] *Id.*
[296] State v. Marino, 747 S.E.2d 633, 638 (N.C. Ct. App. 2013).
[297] *Id.* at 639 (emphasis added).

that not all state technicians can be trusted—as the conviction of Annie Dookhan in Massachusetts clearly establishes.[298]

The situation was different in Minnesota; there, a state court determined that source code is essential to safeguarding a defendant's rights. In *State v. Underdahl*, the Minnesota Supreme Court determined that defendants who failed to provide any information or documentary evidence that the breathalyzer source code was relevant to their criminal defense would be denied such source code.[299] However, the Minnesota Supreme Court further concluded that a defendant who provided evidence that "an analysis of the source code may reveal deficiencies that could challenge the reliability of the Intoxilyzer" should be entitled to engage in discovery regarding that source code.[300]

Interpreting *Underdahl*, the Court of Appeals of Minnesota addressed a defendant who challenged the revocation of his driver's license based on an implied consent statute and an alcohol concentration in excess of the legal limit.[301] James Lund submitted an affidavit from a forensic scientist who explained the significance of access to the source code for the defendant.[302] Moreover, the forensic scientist opined that an analysis of such code is necessary to determine whether they functioned

---

[298] The Dookhan scandal is not the only one to rock the criminal justice system leading to the overturning of numerous convictions. *See generally* Keith L. Alexander, *Prosecutors Criticize D.C. Crime Lab's Handling of Some DNA Evidence*, WASH. POST (Mar. 5, 2015), https://www.washingtonpost.com/local/crime/dc-prosecutors-criticize-city-crime-labs-handling-of-some-dna-cases/2015/03/05/b5244f88-bea4-11e4-b274-e5209a3bc9a9_story.html [https://perma.cc/77Q5-9AGW]; Kyle Swenson, *Broward Crime Lab Scandal Could Taint Many Cases*, MIA. NEW TIMES (Aug. 7, 2014), https://www.miaminewtimes.com/news/broward-crime-lab-scandal-could-taint-many-cases-6396401 [https://perma.cc/CS7U-BJA4]; Mark Hansen*, Crime Labs Under the Microscope After a String of Shoddy, Suspect and Fraudulent Results*, ABA JOURNAL (Sept. 1, 2013), http://www.abajournal.com/magazine/article/crime_labs_under_the_microscope_after_a_string_of_shoddy_suspect_and_fraudu [https://perma.cc/RG6M-36LX]; Maurice Chammah, *After Drug Lab Scandal, Court Continues to Reverse Conviction*, TEX. TRIBUNE (Mar. 27, 2013), https://www.texastribune.org/2013/03/27/after-drug-lab-scandal-court-reverses-convictions/ [https://perma.cc/R27P-3NHN] (reporting that "a former employee with a DPS crime laboratory in Houston may have fabricated the results of thousands of drug tests").

[299] 767 N.W.2d 677, 685 (Minn. 2009).

[300] *Id.* at 685-86.

[301] Lund v. Comm'r of Pub. Safety, No. A08-1408, 2009 WL 1587135, at *1 (Minn. Ct. App. June 9, 2009).

[302] *Id.*

accurately.[303] Notwithstanding this expert affidavit, the trial court denied the request for discovery of the source code.[304] The *Lund* appellate court reversed and remanded, finding that discovery was appropriate based on *Underdahl* because "a showing that discovery of the source code is relevant to the accuracy of a petitioner's test results is sufficient to support a motion for discovery of the source code."[305]

The Court of Appeals of Minnesota heard similar challenges seeking the breathalyzer source code in criminal prosecutions for driving while intoxicated. For example, the appellate court reversed and remanded the trial court's decision to deny discovery of the source code when the defendant submitted a declaration from a computer forensics expert[306] as well as a complaint filed by Minnesota against the manufacturer of the Intoxilyzer.[307] Similarly, in *State v. Veldhuizen*, the Court of Appeals of Minnesota concluded that the trial court abused its discretion in denying a request for discovery of the source code where defendant submitted an expert affidavit describing the source code and its effect of test results as well as information regarding "two inconsistent data reports produced from the same underlying intoxilyzer test."[308]

Yet the Court of Appeals of Minnesota affirmed criminal convictions for driving while intoxicated when the defendants failed to demonstrate before the trial court how the source code would assist them in challenging the charge against them: "Cornish did not present a proffer or analysis in support of either his pretrial motion to compel or his timely motion for a new trial that is comparable to the submission found sufficient in *Underdahl II*."[309] The Cornish court determined that his

---

[303] *Id.*
[304] *Id.*
[305] *Id.* at *3; *see also* Duncan v. Comm'r of Pub. Safety, No. A08-2237, 2009 WL 2366280, at *4 (Minn. Ct. App. Aug. 4, 2009) (holding that the trial court abused its discretion in denying discovery of the source code).
[306] State v. Granse, No. A09-2192, 2010 WL 4451243, at *2 (Minn. Ct. App. Nov. 9, 2010).
[307] Minnesota *ex rel.* Campion v. CMI of Kentucky, Inc., No. 08-603, 2009 WL 2170134, at *4 (D. Minn. July 16, 2009).
[308] Nos. A08-2110, A08-2112 & A08-2113, 2009 WL 1684494, at *2 (Minn. Ct. App. Nov. 9, 2010).
[309] State v. Cornish, No. A08-1228, 2010 WL 2035610, at *4 (Minn. Ct. App. May 25, 2010).

proffers were insufficient to establish that the source code would be useful in challenging the breathalyzer evidence.[310]

Similarly, in *State v. Garberg*, the Court of Appeals of Minnesota heard another appeal challenging the denial of access to a breathalyzer source code.[311] The court determined that the defendant "failed to make a threshold evidentiary showing that the source code information may relate to his guilt or innocence, negate his guilt, or reduce his culpability" before the trial court as mandated in *Underdahl*.[312] This inconsistency is troubling. As we have shown, the problem of buggy (and hence unreliable) code is endemic in computer software; one cannot assume, absent substantial evidence, that some particular piece of software is correct.

The issue of correct software occurs in a myriad of different situations, including in situations of far greater import than an individual DWI case. For example, recently, the federal government engaged in a nationwide prosecution of individuals who distributed child pornography images amongst themselves on a secured website known as Playpen.[313] Criminal defendants around the country had varying degrees of success in challenging the evidence used against them.[314]

In 2016, in *United States v. Michaud*, the federal government indicted Jay Michaud for offenses related to child pornography.[315] Defense counsel moved for the prosecution to produce the source code for the network investigative technique ("NIT") used by the FBI during its investigation.[316] The technique was designed to activate a "computer—wherever located—to send to a computer controlled by or known to the government, network level messages containing information that may assist in identifying the computer, its location, [and] other

---

[310] *Id.*
[311] No. A09-914, 2010 WL 772622, at *3 (Minn. Ct. App. Mar. 9, 2010).
[312] *Id.*
[313] *See generally* Brian L. Owsley, *Network Investigative Source Code and Due Process*, 14 DIGIT. EVID. & ELEC. SURVEILLANCE L. REV. 39 (2017).
[314] *See id.*
[315] No. 3:15-cr-05351, 2016 WL 337263, at *1-2 (W.D. Wash. Jan. 28, 2016).
[316] *Id.*; Order on Procedural History and Case Status in Advance of May 25, 2016 Hearing at 1, *Michaud*, 2016 WL 337263 (No. 3:15-cr-05351).

information."[317] The district court concluded that it would be prejudicial to the defendant if the government were allowed to enter evidence to which the defendant could not meaningfully respond because of lack of access to the NIT source code.[318] The court further explained given that "the discovery withheld implicates the defendant's constitutional rights,"[319] any "evidence of the NIT and the search warrant issued on the basis of the NIT should be suppressed, and the fruits of that search must also be suppressed."[320]

The prosecution then moved to dismiss the charges against Michaud without prejudice,[321] resulting in the dismissal of the charges against him.[322] However, Michaud was an exception; most defendants caught through the Playpen investigation failed to gain access to the source code despite filing motions to do so.[323]

Most recently, in a prosecution reliant upon DNA software for identification, the New York Supreme Court Appellate Division considered whether the Confrontation Clause required the prosecution to provide the defendant with source code.[324] Law enforcement had located DNA on the cord used to strangle the decedent.[325] After the defendant's arrest, the police took a buccal swab to gather his DNA for testing using a software program known as TrueAllele Casework System.[326] The defendant asserted that the state violated his

---

[317] *Michaud*, 2016 WL 337263, at *2.
[318] Owsley, *supra* note 313, at 45; *see also* Order Denying Dismissal and Excluding Evidence at 1, *Michaud*, 2016 WL 337263 (No. 3:15-cr-05351).
[319] *Id.*
[320] *Id.*
[321] Government's Unopposed Motion to Dismiss Indictment Without Prejudice at 3, *Michaud*, 2016 WL 337263 (No. 3:15-cr-05351).
[322] Order Dismissing the Indictment Without Prejudice at 1, *Michaud*, 2016 WL 337263 (No. 3:15-cr-05351).
[323] *See, e.g.*, United States v. Harney, No. 16-38, 2018 WL 1145957, at *12 (E.D. Ky. Mar. 1, 2018); United States v. Stepus, No. 3:15-cr-30028, 2018 WL 1257804, at *4 (D. Mass. Mar. 12, 2018); United States v. Zak, No. 16-CR-65, 2017 WL 4358140, at *4 (W.D.N.Y. Oct. 2, 2017); United States v. Spicer, No. 1:15-cr-73, 2018 WL 635889, at *5 (S.D. Ohio Jan. 31, 2018).
[324] *See* People v. Wakefield, 107 N.Y.S.3d 487, 495 (App. Div. 2019).
[325] *Id.* at 491.
[326] *Id.*

Confrontation Clause right by denying him access to TrueAllele's source code.[327]

The appellate court characterized the argument by the defendant, Wakefield, as novel because it was based on the source code being the out-of-court declarant.[328] Moreover, it determined that "the TrueAllele report [was] testimonial in nature."[329] Nonetheless, the court rejected the argument that the source code should be available to the defendant because the creator of TrueAllele who wrote its source code had testified about the pertinent algorithm.[330] In the court's opinion, there was no violation of Wakefield's Confrontation Clause right.[331] Having the algorithm's designer testify about the code did not answer, however, whether the software performed correctly; only an examination of the code could do that.

The decision in *Wakefield*, along with *Michaud*, *Underdahl*, and its progeny, demonstrates that on a case-by-case basis, some courts are starting to acknowledge that source code may be essential for defendants to exercise their Confrontation Clause rights. These cases, however, have largely not addressed the issue of correctness of evidentiary software.

2. Implications for Evidentiary Software

The limited Confrontation Clause jurisprudence in this area has developed based on the determination that criminal defendants do not really need access to the source code—and thus their constitutional rights are not violated when they lack this ability. As Jennifer Mnookin explained:

> [A] great many lower court opinions have wrestled with the potential Confrontation Clause implications of expert evidence that includes statements that might be classified

---

[327] *Id.* at 494.
[328] *Id.* at 495.
[329] *Id.* at 497.
[330] *Id.*
[331] *Id.*

> as testimonial. Most of these courts have endeavored to find ways around *Crawford's* dictates; unfortunately, most of the arguments proffered by these courts are deeply intellectually unsatisfying.[332]

The previously discussed Intoxilyzer decisions further demonstrate that point.[333]

Since *Melendez-Diaz* and *Bullcoming*, courts have wrestled with appropriately handling the Confrontation Clause when the expert witness is not a person but rather a result determined by computer software. As Professor Mnookin indicated, the key issue in *Crawford* is, "[w]as the statement made in circumstances that suggest its likely future relevance as testimony in a criminal prosecution?"[334] For evidentiary software, the answer is an unqualified yes. Its purpose is exactly to produce testimony to be relied upon in a criminal trial. The expert is not the policeman proffering the results of a breathalyzer test or a radar gun; the expert is the reasoning exemplified in the software that processes the data—and the implementation of that reasoning. In this regard, *Wakefield* reached exactly wrong conclusion.

To get it right, we need to go back to *Daubert*, which lays out very clearly how to determine the admissibility of expert witness testimony:

> The Rules—especially Rule 702—place appropriate limits on the admissibility of purportedly scientific evidence by assigning to the trial judge the task of ensuring that an expert's testimony both rests on a reliable foundation and is relevant to the task at hand . . .

---

[332] Mnookin, *supra* note 248, at 796.
[333] *See* City of Fargo v. Levine, 747 N.W.2d 130 (N.D. 2008); State v. Peters, 264 P.3d 1124 (Mont. 2011); Wexler, *supra* note 11; State v. West, 279 P.3d 354 (Or. Ct. App. 2012); *In re* Comm'r of Pub. Safety, 735 N.W.2d 706 (Minn. 2007); Affadivit of Brian Faulkner, State v. Bonakoske (Fla. Charlotte County Ct. June 10, 2011) (No. 08-1726-T); State v. Underdahl, 767 N.W.2d 677 (Minn. 2009); State v. Granse, No. A09-2192, 2010 WL 4451243 (Minn. Ct. App. Nov. 9, 2010); Minnesota *ex rel.* Campion v. CMI of Kentucky, Inc., No. 08-603, 2009 WL 2170134 (D. Minn. July 16, 2009); State v. Veldhuizen, Nos. A08-2110, A08-2112 & A08-2113, 2009 WL 1684494 (Minn. Ct. App. Nov. 9, 2010).
[334] Mnookin, *supra* note 248, at 794.

> [T]he trial judge, pursuant to Rule 104(a), must make a preliminary assessment of whether the testimony's underlying reasoning or methodology is scientifically valid . . . .[335]

The Supreme Court's analysis in *Daubert* warrants repeating: the trial judge is assigned the task of ensuring that an *expert's testimony rests on a reliable foundation* and must make a preliminary assessment of *whether the testimony's underlying reasoning is scientifically valid*.[336] As we already have documented, despite the best intentions of programmers, source code often has errors. Yet there is no way for a defendant to demonstrate the existence of any errors without having the code examined. Therefore, any approach that eschews access to the code for such an examination presents a serious gap in the protection of defendants' rights.

A trial judge is not in a position to determine whether the underlying reasoning provided by the program is valid or whether, if the underlying reasoning is correct, it has been properly implemented. Indeed, when it comes to technological developments that impact a criminal defendant's constitutional rights, many judges are ill-equipped to analyze the nuances related to such technology.[337] This realization begs the question of how best to address this issue and safeguard constitutional rights.

We do not live in a perfect world where there are no criminals, where the criminals that exist conduct only minor criminal activities that cause no serious, long-term harm, where all computer code works perfectly the first time, or where software errors are easy to find. Consequently, we do not have a solution that can completely solve the problem of

---

[335] Daubert v. Merrell Dow Pharms., Inc., 509 U.S. 579, 579-80 (1993).
[336] *Id.*
[337] *See, e.g.*, Brian L. Owsley, *Triggerfish, Stingrays, and Fourth Amendment Fishing Expeditions*, 66 HASTINGS L.J. 183, 210 (2014) ("[O]ther magistrate judges may have received applications using the pen register application and not realized that they were authorizing or denying use of a cell site simulator."); Brian L. Owsley, *The Fourth Amendment Implications of the Government's Use of Cell Tower Dumps*, 16 U. PA. J. CONST. L. 1, 18 (2013) ("[I]n my own informal survey of magistrate judges nationwide, many have informed me that they were unfamiliar with cell tower dumps.").

convictions based on faulty software. However, we do have a solution posed by evidentiary software that can level the playing field and enable defendants "to confront the witnesses against [them]."[338] It is a solution that lies already in the words of *Melendez-Diaz*, where the Court enunciated that "[c]onfrontation is designed to weed out not only the fraudulent analyst, but the incompetent one as well."[339]

In one regard, *Crawford's* purpose was rooting out hearsay ("even if the Sixth Amendment is not solely concerned with testimonial hearsay, that is its primary object, and interrogations by law enforcement officers fall squarely within that class"[340]) but in essence, the courts relying on source code that the defendant cannot cross examine is analogous to the courts relying on hearsay. In analyzing hearsay, Mnookin wrote that, post-*Crawford*, it is not acceptable to simply cross examine an expert witness if she herself has relied on hearsay; the defendant must be also be afforded the opportunity to cross examine those upon whom the expert relied in preparing her testimony.[341]

One might imagine an objection to this two-prong approach posited by Mnookin on the grounds that this process can lead to an almost infinite regression of required witnesses, each one suggesting that they have learned some small fact relevant to the case from the next, who must then be called to testify.[342] Mnookin and Kaye call into question that only the expert needs to be available and questioned as a witness by defense counsel:

> Take the testimony about the chemical makeup of a drug in *Melendez-Diaz*. Justice Scalia implies that the person who establishes the accuracy of the testing device is not required. But why not? Is that not a prerequisite to being able to be confident about the accuracy of the result? Arguably the person who interpreted the result is making the most central bottom-line judgment relevant to the

---

[338] *Levine*, 747 N.W.2d at 134.
[339] Melendez-Diaz v. Massachusetts, 557 U.S. 305, 319 (2009).
[340] Crawford v. Washington, 541 U.S. 36, 53 (2004).
[341] Mnookin, *supra* note 248, at 832-33.
[342] *Id.* at 833.

> defendant. Is this individual therefore the only one who needs to testify? What if the technician who prepared the sample somehow contaminated it? Should that technician be required to testify as well? None of these technicians is merely engaged in documenting a chain of custody. They are all taking steps that contribute to the final conclusion about the chemical composition of the substance at issue.[343]

However, the issue they raise (which they also answer) is not of concern here: the "witness" is only one deep. It is the software upon which the expert witness relied in order to provide her testimony.

The issue of hearsay is exactly the point here. The expert witness testifying about the results of a breathalyzer or DNA test is not testifying as to what the software does. Instead, that expert is testifying as to what the program designer and coder intended the software to do. Nor, for the same reason, is it adequate for the program designer or coder to testify—for they know only what they intended the software to do, not that the program does exactly that, no more, no less, in all circumstances. The harm that *Crawford* seeks to prevent—the inability of the defendant to confront a witness testifying against him—is denied when he, or his delegated authorities, cannot examine the code. Only by viewing the code can the defendant have the rights afforded to him by the Confrontation Clause. Given the prevalence of errors in software, defendants must have that right.

We propose a simple test (analogous by voting system software independence) for determining whether software-based systems that produce criminal evidence should be subject to adversarial analysis. If the relevant inputs to the system are retained, and all other information and algorithms required to reproduce the software-based evidence are available to opposing parties, then there is likely no compelling benefit to adversarial analysis of the software. In such cases, the correctness of the software's behavior can be confirmed or refuted simply by independently recalculating what the software was supposed to do and

---

[343] Mnookin & Kaye, *supra* note 286, at 152-53.

comparing the result with the evidence in question. If, on the other hand, such an independent recalculation cannot be done (either because some relevant inputs are not available or because the software relies on proprietary algorithms or other behavior), there is no way for an opposing party to determine whether the evidence is correct without expert adversarial analysis of the software. Such a test has the benefit of being relatively straightforward to apply by courts.

We must admit that our proposed solution is not quite as simple as those words might make it appear. Evidentiary code is typically developed by the private sector with the intent of profit. In other words, making the underlying code public is implausible. However, there are many ways for the software to be examined by the defense that do not require the code to accessible to the general public. The most obvious answer is a protective order, similar to those used in patent cases, be used in regard to source code. This approach was considered in *Peters*, but unfortunately for the criminal defendant the expert refused to sign the proposed protective order, resulting in an adverse decision for the criminal defendant.[344] Provisions of such orders commonly include requirements that analysis be done on computers not connected to the Internet, and that the analysts not have any means, *e.g.*, external disks, with which they can make copies of the software at issue.[345] Other mechanisms, including limited courtroom closures, are discussed by Wexler.[346]

C. Due Process Clause

The Fifth Amendment mandates that "[n]o person shall . . . be deprived of life, liberty, or property, without due process of law."[347] Similarly, the Fourteenth Amendment serves as a constraint on action by states: "nor shall any State deprive any person of life, liberty, or property,

---

[344] State v. Peters, 264 P.3d 1124, 1129, 1131 (Mont. 2011).
[345] Lydia Pallas Loren & Andy Johnson-Laird, *Computer Software-Related Litigation: Discovery and the Overly-Protective Order*, 6 FED. CTS. L. REV. 75, 99-100 (2012).
[346] Wexler, *supra* note 11, at 1409-10.
[347] U.S. CONST. amend. V.

without due process of law."[348] Due process is not easily defined, but at its core, the right of due process is about fundamental fairness.[349]

Courts have rejected criminal defendants' due process claims for source code when the government does not have the source code.[350] "Due process does not require the police to seek and find exculpatory evidence."[351] The increased move by society to outsourcing services might well lead to a situation in which law enforcement increasingly does not actually have the software that was used in the conviction.[352] Thus, any requirement that a defendant has the right to examine software might need to be addressed not only judicially, but also legislatively.

While courts might be not be willing to compel discovery of the source code in civil proceedings, such as driver's license revocation hearings, discovery might be warranted in criminal proceedings.[353] In *Crandall*, the court indicated that due process was not an issue because it was a summary civil proceeding, but "[d]iscovery of the source code might be permitted for [a] criminal trial."[354] In the civil context, legislative solutions may be the only way for individuals to obtain this necessary information. Nonetheless, courts have a role in determining whether due process was violated. In the end, there is a fundamental unfairness for the government to rely on a technology and prevent a defendant from challenging its reliability when addressing the adverse consequences. Ultimately, courts have a role in determining whether a due process violation has occurred.

---

[348] U.S. CONST. amend. XIV, § 1.
[349] *See, e.g.*, Lassiter v. Dep't of Soc. Servs., 452 U.S. 18, 24-25 (1981); *see also* State v. Davis, 898 N.E.2d 281, 287 (Ind. 2008).
[350] *See* Mahan v. Bunting, No. 1:13-CV-00165, 2014 WL 1154054, at *3 (N.D. Ohio Mar. 20, 2014).
[351] *Id.* (quoting Stadler v. Curtin, 682 F. Supp. 2d 807, 818 (E.D. Mich. 2010)).
[352] *See, e.g.*, City of Fargo v. Levine, 747 N.W.2d 130, 133-34 (N.D. 2008).
[353] *See In re* Crandall, No. 2008–951, 2008 WL 4615633, at *3 (N.Y. Sup. Ct. Aug. 25, 2008), *aff'd sub nom.* Crandall v. Brovetto, 891 N.Y.S.2d 761 (2010).
[354] *Id.*

Machines based on software are susceptible to potential errors that lead to testimony that can wrongly convict individuals.[355] Without allowing defendants to have access to the source code related to their cases, they will be unable to have experts who can analyze whether the source code operates as it is intended to do or whether there are flaws that might lead to the conviction of an innocent defendant. Indeed, in 2008, in *State v. Kummer*, the Court of Appeals of Minnesota explained that failing to provide a defendant the source code for the Intoxilyzer 5000EN might violate her due process rights.[356]

Notwithstanding the *Kummer* court's decision, courts generally do not address the merits of whether the denial of access to source code for criminal defendants violates due process. Some reasons make sense. For example, in 2010, another Minnesota appellate court declined to address a due process violation based on the use of a breathalyzer when the defendant was convicted for driving under the influence based on a urine sample as opposed to the use of a breathalyzer.[357] Indeed, appellate courts regularly decline to address the issue because defendants raised it for the first time on appeal.[358]

In *Peters*, criminal defendants also asserted a claim based on a due process violation regarding convictions for driving under the influence.[359] In 2011, the Supreme Court of Montana declined to address the due process claim.[360] The court determined that CMI had offered to make the source code available to the defendants, but the defense expert refused the offer because he did not agree with CMI's

---

[355] Andrea Roth, *Machine Testimony*, 126 YALE L.J. 1972, 1989-90 (2017).
[356] No. A08-0533, 2008 WL 4472610, at *1 (Minn. Ct. App. Oct. 7, 2008).
[357] State v. Sterling, 782 N.W.2d 579, 581 n.2 (Minn. Ct. App. 2010).
[358] *See* Christian v. Comm'r of Pub. Safety, No. A13-1921, 2014 WL 1758374, at *2 (Minn. Ct. App. May 5, 2014); State v. Garberg, No. A09-914, 2010 WL 772622, at *3 (Minn. Ct. App. Mar. 9, 2010); Lund v. Comm'r of Pub. Safety, No. A08-1408, 2009 WL 1587135, at *3 (Minn. Ct. App. June 9, 2009); *see also* State v. Lindner, No. 1 CA-CR 09-0583, 2010 WL 2103532 at *1 (Ariz Ct. App. May 25, 2010) (Lindner "waived these claims . . . by failing to argue them" in his appellate brief) (citations omitted), *superseded by* 252 P.3d 1033 (Ariz Ct. App. 2010).
[359] State v. Peters, 264 P.3d 1124, 1129, 1132 (Mont. 2011).
[360] *Id.* at 1132 ("Courts should avoid constitutional questions whenever possible." (citing Kulstad v. Maniaci, 244 P.3d 722 (Mont. 2010))).

terms.[361] Specifically, he expressed concerns that he could be sanctioned and lose his law license based on the overly strict approach by CMI in its take-it-or-leave-it agreement.[362] Moreover, CMI expected the expert to work exclusively at its headquarters and to destroy his computer hard drive when the review was completed.[363] These demands by CMI also served to create a much more burdensome and costly review process.

In that context, the *Peters* court found that it was not necessary to review the due process claims.[364] However, this determination ignored the fact that it left the criminal defendants without any means to present a defense to the charges against them.

In *West*, the criminal defendant was indigent and sought state funds so that he could retain an expert to analyze the breathalyzer's source code.[365] After the trial court denied his request for such funds, West argued that this denial violated his due process rights.[366] Specifically, he asserted that the prosecution failed to demonstrate that the breathalyzer was reliable and that an expert to analyze the source code would assist him in obtaining exculpatory evidence.[367] In 2012, the Court of Appeals of Oregon rejected the due process argument, concluding that it constituted a fishing expedition.[368] It noted that West "relies instead on the assertion that, *if* various governmental agencies produced everything defendant requested . . . *then* an expert *might* be able to demonstrate that the documents reveal something useful."[369] This approach did not establish a basis for finding a due process violation.[370]

---

[361] *Id.*
[362] *Id.* at 1129.
[363] *Id.*
[364] *Id.* at 1132.
[365] State v. West, 279 P.3d 354, 360 (Or. Ct. App. 2012).
[366] *Id.*
[367] *Id.*
[368] *Id.*
[369] *Id.*
[370] *Id.* at 361.

In *Collins v. State*, the defendant was charged with DUI per se and DUI less safe, but the trial court only sentenced her based on a conviction for DUI less safe.[371] On appeal, she argued that her due process rights were violated because she was not able to access the source code for the Intoxilyzer 5000 for trial.[372] In 2014, the Court of Appeals of Georgia concluded that Collins' due process claim was moot because she was only convicted of DUI less safe, which was unrelated to the source code.[373]

In *Commonwealth v. Camblin*, the defendant challenged the denial pursuant to state and federal due process rights.[374] In 2015, the Massachusetts Supreme Judicial Court, which is the state's highest court, addressed the trial court's decision denying a motion in limine seeking to bar the breath test machine because it was scientifically unreliable.[375] Due process mandates that evidence presented by the prosecution must be reliable.[376] The court reversed, finding that the defendant was addressing a new technology and thus should have be able to challenge its reliability before the trial court.[377]

---

[371] 760 S.E.2d 606, 608 (Ga. Ct. App. 2014); *see also* GA. CODE ANN. § 40-6-391(a)(5) (2014) (establishing six ways in which a driver may be driving while impaired in some manner.) The statute defines DUI per se as "driv[ing] or be[ing] in actual physical control of any moving vehicle while . . . [t]he person's alcohol concentration is 0.08 grams or more at any time within three hours after such driving or being in actual physical control from alcohol consumed before such driving or being in actual physical control ended." *Id*. In other words, this definition fits the standard blood alcohol concentration. The statute also defines DUI less safe as "driv[ing] or be[ing] in actual physical control of any moving vehicle while . . . [u]nder the influence of alcohol to the extent that it is less safe for the person to drive." *Id.* § 40-6-391(a)(1). In order to meet this standard, it is not necessary to provide evidence of blood alcohol concentration. Instead, the prosecution simply needs to establish an impairment due to alcohol based on circumstantial evidence. *See, e.g.*, Davis v. State, 687 S.E.2d 854, 858 (Ga. Ct. App. 2009).

[372] *Collins*, 760 S.E.2d at 608.

[373] *Id.* at 608-09.

[374] 31 N.E.3d 1102, 1108 (Mass. 2015).

[375] *Id.* at 1104-05.

[376] *Id.* at 1111 (discussing Commonwealth v. Given, 808 N.E.2d 788 (Mass. 2004)).

[377] *Id.* at 1112.

IV. Conclusion

As the above legal discussion reflects, there are not many cases addressing a defendant's right to access of the relevant source code consistent with a constitutional right. In most of the breathalyzer cases, the defendants and their attorneys failed to properly present the various constitutional challenges such that there were very few discussions on the merits. Even in the few instances of decisions on the merits, the courts fail to understand the implications of the requests. For example, in *Camblin*, the court granted the request because it was a new technology; that, though, is not the fundamental issue.

Overall, the problem is that the courts have applied the wrong standard. This is exemplified by the Minnesota ruling in *Garberg*, where the court wrote that the defendant "failed to make a threshold evidentiary showing that the source code information may relate to his guilt or innocence, negate his guilt, or reduce his culpability."[378] In other words, the court felt that software should be presumed reliable, and that without specific indications to the contrary Garberg could not have access to the source code. This is in stark contrast to how human witnesses are treated, where defendants have a constitutional right to try to impeach their credibility. Software does not deserve elevated status. Defendants must have access to evidentiary source code.

Implementing this policy requires several modifications to current practice. First, of course, is a rule on when defendants should have such access. We suggest adapting the definition of "software independence" proposed by Professor Rivest for elections: "A voting system is *software-independent* if an undetected change or error in its software cannot cause an undetectable change or error in an election outcome."[379] Note carefully that this does not rule out the use of software; systems where software proposes an answer but a manual process verifies it can still be software-independent.[380] Here, we propose that if evidence is not

---

[378] State v. Garberg, No. A09-914, 2010 WL 772622, at *3 (Minn. Ct. App. Mar. 9, 2010).
[379] Rivest & Wack, *supra* note 164, at 3761.
[380] In election systems, this can be accomplished by risk-limiting audits, a statistical process that involves manual counting of a small subset of the ballots cast, even if they were tallied by

analogously "software-independent", the defense should be entitled to examine the software that produced it.

A second crucial point is that the source code must be available for production by the prosecution. Procurement contracts must specify delivery of source code for each version of the code or device. This has to be part of acceptance of the product by the government, with no payment tendered or use of the product until the source code has been delivered. Such a policy would avoid situations as in *In re Commissioner of Public Safety,* where the Commissioner had to be ordered to obtain the code,[381] or *Bonakoske*, where the vendor asserted that it no longer possessed the code.[382]

Finally, defendants must have reasonable access. Auditing a program is not done simply by staring at printouts of the code; rather, a variety of specialized tools are used, both to examine the code[383] and to do test executions.[384] Audits may require teams; furthermore, it must often be contracted out to specialist companies. Reasonable protective orders may be required,[385] but they should not be overly onerous, as in *Peters*.[386] Even these may not be enough. As we noted, training data must be provided if machine learning algorithms are involved.[387] Professor Bratus et al. suggest the need for information on external inputs, e.g., the time of day, the configuration of the operating system and supporting software, and the specific version of the evidentiary software in use.[388]

---

a computer. *See, e.g.*, Mark Lindeman & Philip B. Stark*, A Gentle Introduction to Risk-limiting Audits*, 10 IEEE SEC. & PRIV. 42, 42 (2012).

[381] 735 N.W.2d 706, 709 (Minn. 2007) (The Commissioner had to be ordered to "*obtain* and provide to [Underdahl's] counsel the complete computer source code.") (emphasis added).

[382] Affidavit of Brian Faulkner at ¶ 5, State v. Bonakoske (Fla. Charlotte County Ct. June 10, 2011) (No. 08-1726-T).

[383] A simple example of such a tool is one that finds each location where each variable is used.

[384] One well-known technique for finding bugs at runtime is known as "fuzzing." *See, e.g.*, Andy Greenberg, *Hacker Lexicon: What is Fuzzing?*, WIRED (June 2, 2016), https://www.wired.com/2016/06/hacker-lexicon-fuzzing/ [https://perma.cc/4L2N-D786].

[385] *But see* Wexler, *supra* note 11.

[386] State v. Peters, 264 P.3d. 1124 (Mont. 2011).

[387] *See* Vincent, *supra* note 178.

[388] Sergey Bratus et al., *Software on the Witness Stand: What Should It Take for Us to Trust It?*, *in* TRUST AND TRUSTWORTHY COMPUTING 396, 406 (Alessandro Acquisti et al. eds., 2010).

Arguably, providing access to source code will impose a burden on the prosecution. Many defendants will be economically unable to take advantage of such opportunities. But neither consideration is unique.

From a technical perspective, it is a virtual certainty that any significant software system contains bugs. As Christian Chessman wrote, "[i]t is difficult to overstate the fallibility of computer programs."[389]

As shown in *Chun* and *Foley*, breathalyzers and DNA-matching devices are not immune.[390] These bugs may or may not affect the results in any given case, but in our judicial system, the Constitution and fundamental considerations of due process demand that defendants should have the opportunity to find out—and that is only feasible with access to source code.

---

[389] Christian Chessman, *A "Source" of Error: Computer Code, Criminal Defendants, and the Constitution*, 105 CALIF. L. REV. 179, 228 (2017).
[390] State v. Chun, 943 A.2d 114, 126-31 (N.J. 2008); Commonwealth v. Foley, 38 A.3d 882, 887 (Pa. Super. Ct. 2012).