The Ethics of Coexistence: Can I Learn to Stop Worrying and Love the Logic Bomb?

John Aycock
Department of Computer Science
University of Calgary
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4
Email: aycock@ucalgary.ca

Anil Somayaji
School of Computer Science
Carleton University
1125 Colonel By Drive
Ottawa, Ontario, Canada K1S 5B6
Email: soma@scs.carleton.ca

John Sullins
Department of Philosophy
Sonoma State University
1801 East Cotati Ave.
Rohnert Park, CA 94928
Email: john.sullins@sonoma.edu

***Abstract*—Computer security attacks are frequent fodder for ethical analyses, but the ethics of computer security *defenses* are not often examined. We address this by considering a topical problem in computer security. In an age of so-called "advanced persistent threats" that lurk undetected on computer systems for long periods of time, it is increasingly unrealistic to expect a computer system to be permanently free of malicious software. Recognizing this, we posit the idea of a "cosecure system" – a cosecure system, by design, would allow legitimate software and malicious software to coexist safely on the same machine.**

We take an unusual tack to software design and use ethical concerns to guide the design of a cosecure system, rather than building a cosecure system and then performing an *ex post facto* ethical analysis.

The principal tenets of security that must be upheld are confidentiality, integrity, and availability, and any system purporting to be secure has an ethical duty to the system user to uphold these. This is the starting point for our design process, and we proceed to look at how a cosecure system may be implemented. What we arrive at by going through this ethics-based software design becomes a proof by contradiction: we are forced to conclude that it is not possible, in fact, for malicious and legitimate software to coexist; a cosecure system as we have described it cannot be built.

This allows us to see traditional computer security defenses in a new light. If we cannot uphold key security properties in the *best* case, where a system is expressly designed to allow coexistence of malicious and legitimate software, what does that imply about the defenses of the actual computer systems we use?

We propose that a community defense is an alternative that eludes previous ethical issues, as well as being defensible from an information ethics point of view.

I. Introduction

Ethical issues are much easier to examine and debate when there is obvious harm that may result from actions and consequences: euthanasia, slavery, weapons development. The same principle seems to hold in the ethics of computer security. Virus writing is potentially harmful, for example, and elicits ethical analyses – indeed, it appears impossible to write a text on computer ethics without a discussion about computer viruses [1]–[4]. Whether "good" viruses and worms can exist has also been examined from the viewpoint of ethics [5], [6], as has proactive research looking to identify new security threats [7], [8]. But there is no need to examine computer security defenses from an ethical point of view because they are always doing good. . . or are they?

As a simple example, a web site's passwords are too easy to guess, so an administrator concerned about brute-force and dictionary attacks installs software that requires users to make strong passwords. Users, in turn, respond by writing passwords down [9] and reusing passwords [10]; attackers switch to phishing and keylogging to steal passwords, attacks that can no longer be detected by the web site's server. This example involves choice, an action taken in the name of security, an action with consequences. As such, it may be subjected to ethical analysis, as can computer security defenses in general.

Most current defenses implicitly assume that the world is divided into two parts: the protected part and the unprotected part. Defense mechanisms attempt to keep the protected part from being compromised; at best, they have no effect on the unprotected part. Malware clearly coexists with legitimate software in current defense scenarios, but (ideally) in the unprotected part. In this paper we look at whether malware coexistence can be extended to apply within a *single* system and still retain security for the user, using a new type of system called a *cosecure system*. Cosecure systems pose a number of ethical challenges, which we use to give insight into traditional defenses and point towards alternative defense approaches.

The rest of this paper proceeds as follows. Section II introduces the concept of cosecure systems. We perform an ethical analysis of cosecure systems in Section III; Section IV looks at an alternative approach, and Section V concludes.

II. Cosecure Systems

Current approaches to defend computers against threats would make a great deal of sense to anyone from medieval times. Erect some walls (more walls are better) and place everything to be protected inside the walls. A breach of the walls is disaster.

Firewalls blocking inbound connections are a perfect embodiment of this metaphor, of course, as are Trojan horses being used to break through defenses. The other current defensive mainstay, anti-virus software, effectively builds walls too by scanning files, memory, and incoming packets to keep threats out. This idea of protection is reflected culturally, as Johnston observes [11, page 24]: 'The antivirus industry's grand narrative is that it is successfully producing antidotes that update computer software against global viral threats.'

How 'successfully' these defenses work is open to debate. A firewall provides no protection from users running malicious code behind the firewall via drive-by downloads and/or social engineering. Anti-virus software relies heavily on a reactive model[1] that is increasingly lagging behind the onslaught of barbarian malware assailing the walls [14]–[16]. In any case, anti-virus software can never fully succeed because its task has long been known to be undecidable [17]. Even the notion of multiple walls – defense-in-depth – is being called into question [18].

These strategies share an all-or-nothing approach to defense, where the goal is to keep all computers malware-free. An alternative is to look at the larger population of computers, accept that some of those computers will be afflicted by malware, and work on containing the problem to those computers. Some work has been done on this (e.g., virus throttling [19]), but the approach is not widespread and it still conceptually treats a computer's status as binary: infected or not infected.[2]

We are proposing a different approach in this paper. We argue that it is increasingly unrealistic to assume that a computer can be kept free of malware, and we must build systems where legitimate software is able to *coexist* with malicious software.[3] The challenge is to construct a system that permits a user to use the computer and have their data and actions uncompromised, yet also permits malware to be present and functional. We call such a computer not secure, not insecure, but *cosecure*. By way of analogy, consider a city. A city is well known to have crime taking place within it, and dark alleys better left unexplored, yet remains functioning regardless.

While there may be great technical challenges in building a cosecure system, we assume for purposes of ethical analysis that we are able to build one. What are the ethical implications? Furthermore, the details of this system have been left purposely vague; can the cosecure system design follow from ethical constraints?[4]

III. Ethics and the Cosecure System

From a conceptual, information ethics standpoint we can argue for the use of a cosecure system. Information entities can all be said to have a certain amount of intrinsic worth. It is only on close analysis of the code and a complete explanation of the computing environment the code is functioning in that any sort of judgment of the value of the code can be reached. This task is difficult, if not impossible, so our system defaults to one that acknowledges that any code may have some worth. But we are also not claiming that every piece of code in a user's system is benign to that user. Some will be working against their wishes, some will be working for their interests, and some will have neutral value. Thus we can see that very complex mixtures of harm, help, and ambivalence can occur from the point of view of all the various applications in the system.

The idea of a cosecure system also shines a different light on security. Specifically, are security professionals (and their progeny, security software) ethically obligated to provide full defense at all times? Other professions are not, such as police or rescue personnel; they are obligated to do what they can but no one expects perfection. It seems that we place stronger demands on computer security, perhaps in part due to the differences between the physics of the real world (police can't be everywhere) versus the cyber world, but also because a single security breach may be potentially disastrous.

Certainly expectations on security software are high. Anti-virus software is used in computer forensics (e.g., [22]), for example, and as such its results need to be accurate enough to stand up in court. At the same time anti-virus and other security software are known to security professionals to yield an imperfect defense, as discussed in Section II. A cosecure system is more ethical than traditional systems in the sense that its very concept is honest and forthcoming in terms of the security it provides and does not provide, thus upholding a duty to be truthful on the part of security professionals.[5] While it can be argued that at certain times a half truth or "Platonic" lie might be justifiable, in this case there is no clear benefit to the user told the half truth that their system is fully protected by the purchase and installation of anti-virus software. There is no known placebo effect possible in this situation and in fact it works in the reverse. The user can operate their system with no idea about the presence of malware operating behind the scenes and beyond the defenses of anti-virus software. The only beneficiaries in these situations are the people selling the software and the people making the malware.

Another commonly-stated duty for computer professionals (e.g., [23], [24]), system administrators (e.g., [25]), and users alike (enshrined in the legal principle of duty of care) is avoiding harm to others and not endangering the public. We must then consider if a cosecure system would violate this duty. The answer derives from what the malware on a cosecure system is permitted to do: malware that attacks other computers by attempting remote exploits for worm spread, or joins a distributed denial-of-service attack clearly causes harm to others. Malware that does not overtly attack is a bit more subtle ethically, but – for example – malware that proxies connections to a "mother ship" for purposes of phishing infrastructure [26] is not ethically neutral in that it facilitates harm being done.

[1]We recognize that anti-virus heuristics, for example, may detect unknown malware proactively, but it is still fair to say that anti-virus is largely reactive. See [12], [13] for more on anti-virus software.

[2]Or, in keeping with the virus throttling work, misbehaving or behaving.

[3]Kursawe and Katzenbeisser [20] also considered coexistence, but did not examine ethical issues at all.

[4]Note we are not simply claiming that a computer system will have 'invisible' ethical values built in as a side effect of its construction [21]. We are instead exploring how to incorporate ethics into the software design process so that a system that upholds ethical values results.

[5]While not binding on security professionals, we note that this is a general moral imperative in the ACM Code of Ethics ('Be honest and trustworthy' [23]), and is also reflected in the IEEE Code of Ethics ('. . . be honest and realistic in stating claims . . . based on available data' [24]).

It follows that there must be constraints on the malware in a cosecure system in order for the system to operate in an ethical fashion. This substantially informs a cosecure system's design. A cosecure system must be able to enforce constraints and must therefore have some form of unimpeachable executive provided through a secure operating system kernel or virtualization. This may further extend to having a trusted, secure boot process.

For the remainder of the cosecure system design, we turn to the basic tenets of security: confidentiality, integrity, and availability [27], [28]. As these underpin our notion of security, an ethical case can be made that a cosecure system has a duty to provide these to a legitimate user.

Availability of a cosecure system addresses whether or not a user has the appropriate resources (e.g., CPU time, memory, bandwidth) for their legitimate computer usage. Resource scheduling is well-studied in operating systems; looking at CPU time, for instance, a cosecure system could employ lottery scheduling [29] with the user possessing the majority of "tickets," which can then be passed as necessary to other processes acting on behalf of the user.

Preserving integrity in a cosecure system implies that a user's programs and data cannot be modified by the malware sharing the system.[6] Access control plays an important role here – changes are permitted only on the behest of the legitimate user – and a secure software update process can be used to prevent malicious updates being introduced [30].

Access control also figures prominently in confidentiality in a cosecure system. A legitimate user's data cannot be allowed to be exfiltrated by malware in any way. This obviously includes local data, but also data dynamically generated: webcam images, microphone data, keystrokes, mouse clicks, screen shots, network traffic. In essence, only the user must be allowed access to cosecure system I/O. Even supposedly isolated malware processes cannot access user data, because of the risk of leaking it via a covert channel [31].

What does this leave? Any process not initiated by a legitimate user cannot access data on a system, except for its own, and has no I/O access apart from perhaps a limited amount of network bandwidth (limited to deter DDoS attacks and other nefarious uses). In essence, we are left with only a (distributed) computing platform for malware, eerily reminiscent of Shoch and Hupp's "worms" at Xerox PARC [32].

We have also ignored an important point, namely how malware gets onto a cosecure system to begin with. If malware is going to be on a system, it must get there in one of three ways: be installed on the system initially, be installed by user action, or by subverting an existing legitimate application. If the malware is present on the system initially, then it is insecure, not cosecure, at the outset. If a user explicitly installs the malware, perhaps through social engineering, then the malware is acting on behalf of the user and is legitimate, in the cosecure system sense. If the malware is injected into a legitimate user application, then again the malware will appear to be acting on behalf of a legitimate user. We therefore assert that, strictly following ethical design constraints, a cosecure system cannot be built.

A cosecure system is still useful, however, because it allows us to view traditional systems anew. There is no meaningful way for malware and legitimate users to coexist on the same machine *even if the system is expressly designed to allow it*, and have ethical duties upheld: duties to not harm others, and duties to the user to provide confidentiality, integrity, and availability. The cosecure system essentially acts as a proof by contradiction. In practice, of course, the situation is even worse – traditional systems are not as restricted as our mythical cosecure system, and systems with "secure" booting, "secure" kernels, and "secure" virtualization are invariably found to be subvertible (e.g., [33], [34]).

Herein lies the dilemma. On one hand, we cannot observe ethical duties if malware is allowed on a user's computer; on the other hand, we know that current defenses are flawed. Nor are we absolved from this dilemma by defense in depth. Piling defenses – flawed or specialized to a single type of attack, or both – haphazardly together and trusting to blind faith that invaders will be kept at bay is not an ethically laudable position either.

One could arrive at the conclusion that the only solution is to do nothing defensively, but given the potential for direct harm to the user and indirect harm to others, this is not ethically permissible. (This may seem like a "straw man" argument, but it is a realistic scenario, when the number of Macs and mobile devices not running security software are considered.) We argue that there is an ethical obligation to apply current defenses even though, being imperfect, they cannot completely fulfill duties to the user. The obligation arises only because applying current defenses is *more* ethical than doing nothing. This is not unlike voting against bad candidates in an election rather than voting for a good candidate. It is a distasteful situation to be in, as a security professional, and we are forced to ask if there are alternatives.

IV. An Alternative Approach

To begin with, building walls and creating a fortress of a single machine (traditional defenses) or within a single machine (cosecure systems) is problematic from an information ethics point of view. Consider a physical building in a large city. Every building, even a private home, has some property that others can use, such as sidewalks and other areas where use is welcome, e.g., walking up to the door when the mail is delivered or if a friend visits, but is suspicious when strangers approach. But it is impossible, and even undesirable for a private home to attempt to become a fortress. It is antagonistic to the free flow of concord and commerce in the public sector. In the world of code, a fortress mentality works to thwart the benefits of free and open computing and this limits the growth of valuable information entities.

If the single computer as fortress is undesirable and unachievable, then we must look more broadly to the single computer as an entity in a community which permits the free flow of information within it. We already organize our computers into communities by networking them, and we should be organizing our defenses to reflect this too. The

[6]Data integrity and confidentiality could be addressed in part by assuming that a user's data is stored somewhere in the cloud, but then attacks involving the malware getting access to or modifying the cloud data would have to be prevented.

key missing concept in security terms is that of the group: collective responsibility. Individual computers in a network should not just be responsible for their own security; they must also help keep other machines secure. Similarly, system administrators have a responsibility to not just keep their systems safe but to keep other members of the community – the Internet – safe when they can [25]. Current mechanisms, however, are more based on a "rugged individualism" view of the world where everybody takes care of their own but nobody has a responsibility to anybody else.

A full discussion and analysis of this alternative approach would occupy a complete paper unto itself, but we briefly note that in practice, principles for community defense include the following:

- Information about attempted or successful security breaches must be disseminated widely in a timely fashion.
- Attacked systems must strive to prevent the attacker from targeting others.
- Systems should leave a forensic trail so others can learn from their demise.
- Systems must monitor their neighbors.
- Systems may disable themselves altruistically if doing so helps preserve the community.

Note that we are *not* assuming that all members of the community will be malware-free. This releases us from ethical problems related to imperfect defenses for any given user, as our duty is now to the larger community, where we have no choice but to coexist with malware. The altered security perspective also allows us to draw on ethical arguments developed for actions and consequences within human communities. To wit, the community of computers mirrors human-to-human relationships, and we conjecture that the ethical principles for humans may be analogously applied to networked computers in this way.

V. Conclusion

We cannot stop worrying about malware, and coexistence with malware on a single machine is not ethically tenable. Looking to coexistence within a larger community is one solution, but tolerating and embracing the inevitability of malware is a strange love indeed.

Acknowledgment

The first two authors' research was supported in part by the Natural Sciences and Engineering Council of Canada via ISSNet, the Internetworked Systems Security Network.

References

[1] S. Baase, *A Gift of Fire*, 2nd ed. Prentice Hall, 2003.

[2] T. Forester and P. Morrison, *Computer Ethics*, 2nd ed. MIT Press, 1994.

[3] D. G. Johnson, *Computer Ethics*, 3rd ed. Prentice Hall, 2001.

[4] H. T. Tavani, *Ethics & Technology: Ethical Issues in an Age of Information and Communication Technology*. Wiley, 2004.

[5] J. Aycock and A. Maurushat, ""Good" worms and human rights," *ACM SIGCAS Computers and Society*, vol. 38, no. 1, pp. 28–39, 2008.

[6] V. Bontchev, "Are "good" computer viruses still a bad idea?" in *Proceedings of the EICAR '94 Conference*, 1994, pp. 25–47.

[7] J. Aycock and A. Maurushat, "Future threats," in *17th Virus Bulletin International Conference*, 2007, pp. 275–281.

[8] J. Aycock and J. Sullins, "Ethical proactive threat research," in *Workshop on Ethics in Computer Security Research*, 2010, pp. 231–239.

[9] A. Adams and M. A. Sasse, "Users are not the enemy," *Communications of the ACM*, vol. 42, no. 12, pp. 40–46, 1999.

[10] B. Ives, K. R. Walsh, and H. Schneider, "The domino effect of password reuse," *Communications of the ACM*, vol. 47, no. 4, pp. 75–78, 2004.

[11] J. Johnston, *Technological Turf Wars: A Case Study of the Computer Antivirus Industry*. Temple University Press, 2009.

[12] J. Aycock, *Computer Viruses and Malware*. Springer, 2006.

[13] P. Szor, *The Art of Computer Virus Research and Defense*. Addison-Wesley, 2005.

[14] K. J. Higgins, "Antivirus rarely catches Zbot Zeus Trojan," Dark Reading, 16 September 2009.

[15] ——, "Study: Antivirus software catches about half of malware, misses 15 percent altogether," Dark Reading, 2 March 2009.

[16] J. Oberheide, E. Cooke, and F. Jahanian, "CloudAV: N-version antivirus in the network cloud," in *17th USENIX Security Symposium*, 2008, pp. 91–106.

[17] F. Cohen, "Computer viruses: Theory and experiments," *Computers & Security*, vol. 6, no. 1, pp. 22–35, 1987.

[18] M. E. Locasto, S. Bratus, and B. Schulte, "Bickering in-depth: Rethinking the composition of competing security systems," *IEEE Security & Privacy*, pp. 77–81, November/December 2009.

[19] J. Twycross and M. M. Williamson, "Implementing and testing a virus throttle," in *Proceedings of the 12th USENIX Security Symposium*, 2003, pp. 285–294.

[20] K. Kursawe and S. Katzenbeisser, "Computing under occupation," in *New Security Paradigms Workshop*, 2007, pp. 81–88.

[21] J. H. Moor, "What is computer ethics?" *Metaphilosophy*, vol. 16, no. 4, pp. 266–275, 1985.

[22] United States District Court, D. Puerto Rico, "N. Rivera-Cruz v. Latimer, Biaggi, Rachid & Godreau, LLP, et al." Civil No. 04-2377 (ADC), 16 June 2008.

[23] Association for Computing Machinery, "ACM code of ethics and professional conduct," http://www.acm.org/about/code-of-ethics, 1992.

[24] Institute of Electrical and Electronics Engineers, "IEEE code of ethics," http://www.ieee.org/about/corporate/governance/p7-8.html, last accessed 17 January 2014.

[25] USENIX Special Interest Group for Sysadmins, "System administrators' code of ethics," https://www.usenix.org/lisa/system-administrators-code-ethics, 2003.

[26] H. Project, "Know your enemy: Fast-flux service networks," 2007.

[27] M. Bishop, *Computer Security: Art and Science*. Addison Wesley, 2003.

[28] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*, 3rd ed. Prentice Hall, 2003.

[29] C. A. Waldspurger and W. E. Weihl, "Lottery scheduling: Flexible proportional-share resource management," in *1st Symposium on Operating Systems Design and Implementation*, 1994.

[30] G. Wurster and P. C. van Oorschot, "Self-signed executables: Restricting replacement of program binaries by malware," in *2nd USENIX Workshop on Hot Topics in Security*, 2007.

[31] B. W. Lampson, "A note on the confinement problem," *Communications of the ACM*, vol. 16, no. 10, pp. 613–615, 1973.

[32] J. F. Shoch and J. A. Hupp, "The "worm" programs – early experience with a distributed computation," *Communications of the ACM*, vol. 25, no. 3, pp. 172–180, 1982.

[33] S. T. King, P. M. Chen, Y.-M. Wang, C. Verbowski, H. J. Wang, and J. R. Lorch, "SubVirt: Implementing malware with virtual machines," in *2006 IEEE Symposium on Security and Privacy*, 2006, pp. 314–327.

[34] M. Steil, "17 mistakes Microsoft made in the Xbox security system," in *22nd Chaos Communication Congress*, 2005.