

THE ETHICS OF “PARASITIC COMPUTING:”
FAIR USE OR ABUSE OF TCP/IP OVER THE INTERNET?

Robert N. Barger

Computer Applications Program

University of Notre Dame, Notre Dame, IN 46556

Phone: 574-289-8939

Fax: 574-289-2039

Email: rbarger@nd.edu

and

Charles R. Crowell

Department of Psychology and Computer Applications Program

University of Notre Dame, Notre Dame, IN 46556

Phone: 574-277-4774

Fax: 574-271-2058

Email: ccrowell@nd.edu

Running Head: THE ETHICS OF PARASITIC COMPUTING

THE ETHICS OF "PARASITIC COMPUTING:"
FAIR USE OR ABUSE OF TCP/IP OVER THE INTERNET?

Abstract

This chapter discusses the ethics of a proof-of-concept demonstration of "parasitic computing." A "parasite" computer attempts to solve a complex task by breaking it up into many small components and distributing the processing of these components to remote computers which perform this processing without the knowledge or consent of those owning the remote computing resources. This is achieved through the use of the TCP/IP Internet protocol and, in particular, the checksum function of this protocol. After a discussion of similar exploits, the ethical issues involved in this demonstration are analyzed. The authors argue that harm should be the standard for determining if parasitic computing is unethical. They conclude that a revised notion of the rights of ownership is needed when dealing with the shared nature of the Internet. Suggestions for future research are offered.

Keywords: checksum, covert channels, covert exploitation, denials of service, Idealism, Internet ethics, IP spoofing, malware, parasitic computing, Pragmatism, TCP/IP protocol

INTRODUCTION

This chapter will examine some of the issues raised by a proof-of-concept demonstration of "parasitic computing" reported in the journal *Nature* (Barabasi, Freeh, Jeong, & Brockman, 2001). In this type of computing, a "parasite" computer attempts to solve a complex task by breaking it up into many small components and distributing the processing related to those components over a number of separate remote computers. While the parasitic procedure represents a form of distributed computing, it differs importantly from other well-known examples such as the "Search for Extraterrestrial Intelligence" (SETI) Project (SETI@home, 2003). The distributed computing utilized in SETI involves volunteers from around the world who allow their local computers to be used for ongoing analysis of vast amounts of data obtained from a radio telescope constantly scanning the heavens. SETI allows anyone with a computer and Internet connection to download software that will read and analyze small portions of the accumulated data (SETI@home, 2003). In effect, SETI has created a super computer from millions of individual computers working in concert.

Like SETI, parasitic computing takes advantage of the power of distributed computing to solve complex problems, but the parasite computer induces "participating" computers, already connected to the Internet, to perform computations without the awareness or consent of their owners. By their own admission, Barabasi et al. (2001) were aware of the ethical issues involved in their demonstration of parasitic computing. On the project website they state: "Parasitic computing raises important questions about the ownership of the resources connected to the Internet and challenges current computing paradigms. The purpose of our work is to raise awareness of the existence of these issues, before they could be exploited" (Parasitic Computing, 2001). In this chapter, we will begin to explore these "important questions" by

focusing on the type of exploitation inherent in parasitic computing and by considering some of the ethical issues to which this new form of computing gives rise.

BACKGROUND

The proof-of-concept demonstration reported by Barabasi et al. (2001) involved a single “parasite” computer networked to multiple “host” web servers by means of the Internet. The underlying communication between the parasite and hosts followed the standard TCP/IP protocol. Within this context, the parasite exercised a form of *covert exploitation* of host computing resources, *covert* because it was accomplished without knowledge or consent of host owners, and *exploitation* because the targeted resources were used for purposes of interest to the parasite, not necessarily the host owners. Covert exploitation of networked computing resources is not a new phenomenon (Smith, 2000; Velasco, 2000). In this section, we will review a few common examples of covert exploitation including some that take advantage of known vulnerabilities in the Internet communication process.

Internet Communication Protocols

The Internet evolved as a way for many smaller networks to become interconnected to form a much larger network. To facilitate this interconnection, it was necessary to establish standards of communication to insure uniformity and consistency in the ways by which a computer attached to one part of the Internet could locate and exchange information with other computers located elsewhere. These standards, known as “protocols,” emerged through the influence of the Internet Society, the closest thing the Internet has to a governing authority. The de facto standard that has emerged for internet communication is a family of protocols known as the Transmission Control Protocol/Internet Protocol (TCP/IP) suite (Stevens, 1994).

This TCP/IP standard helps to insure certain levels of cooperation and trust between all parties employing the Internet.

As shown in Figure 1, the TCP/IP protocol suite usually is represented as a layered stack where the different layers correspond to separate aspects of the network communication process (Stevens, 1994). The bottommost *link* layer in the stack corresponds to the physical hardware (i.e., cables, network cards, etc.) and low-level software (i.e., device drivers) necessary to maintain network connectivity. The middle two layers represent the *network* and *transport* layers, respectively. Roughly speaking, the network layer is responsible for making sure that the “packets” of information being sent over the network to a remote computer are being sent to the proper destination point. Several different forms of communication are employed by this layer, but IP is the main protocol used to support packet addressing. The transport layer, just above network, uses TCP as its main protocol to insure that packets do, in fact, get where they are supposed to go. In essence, at the sending end, the TCP layer creates and numbers packets, forwarding them to the IP layer, which figures out where they should be sent. At the receiving end, the TCP layer reassembles the packets received from the IP level in the correct order and checks to see that all have arrived. If any packets are missing or corrupt, TCP at the receiving end requests TCP at the sending end to re-transmit. The top layer of the stack contains *application* services users employ to initiate and manage the overall communication process, applications like file transfer (FTP), email (POP and SMTP), and web browsing (HTTP).

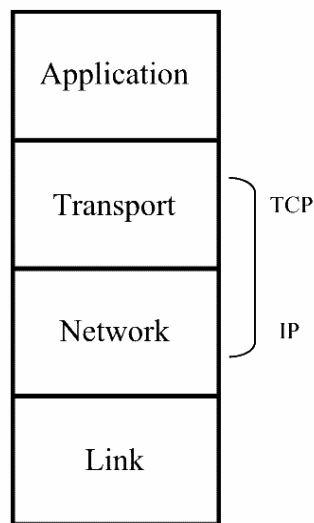


Figure 1. Layers of the TCP/IP protocol. Adapted from Stevens, W. R. (1994). *TCP/IP*

Illustrated, Volume 1. Reading, MA: Addison-Wesley, Figure 1.1.

Worms, Viruses, and Trojan Horses

Covert exploitation of computing resources has taken many forms over the years, some more malicious than others. Perhaps the most nefarious examples are those involving what is called “malware,” short for malicious software, designed to damage or disrupt a system (Wiggins, 2001). Malware often takes the form of worms, viruses or Trojan horses, problems that have become all too common in recent years and do not need to be explored further here. Suffice it to say that while there is some debate about the precise distinctions among these variants of malware (Cohen, 1992), it is clear that all operate covertly to insinuate themselves into computer systems for purposes of exploitation.

IP-related Vulnerabilities

Prior to the widespread use of networks and the Internet, the spread of malware among stand-alone computers was dependent upon transfer by means of removable media like floppy disks. With the advent of networking, and the attendant increase in email usage, many other methods became available for gaining unauthorized access to computing resources. While email still may be the most common method used to achieve the spread of malware (Wiggins, 2001), certain forms of covert exploitation associated with vulnerabilities in the TCP/IP protocol have been known for some time (Bellovin, 1989). Space limitations preclude a detailed treatment of this matter here, but three categories of vulnerability will be mentioned: IP spoofing, denials of service, and covert channels. Each represents exploitation of the “trust” relationships Barabasi et al. (2001) describe as being inherent in the TCP/IP protocol.

IP spoofing, as described by Velasco (2000), is a method whereby a prospective intruder impersonates a “trusted” member of a network by discovering its IP address and then constructing network packets that appear to have originated from this source. Other network computers may then accept those packets with little or no question because they seem legitimate and further authentication is not mandatory under the TCP/IP protocol (i.e., trust is assumed). While the technical details of this approach are rather intricate, involving both the impersonation process itself as well as a method for disabling TCP acknowledgements sent back to the system being impersonated, intruders have used this technique to establish communications with remote computers, thereby potentially “spoofing” them into further vulnerabilities and/or unauthorized access (Velasco, 2000).

Denials of service involve malicious attempts to degrade or disrupt the access of network members to a particular host by consuming the TCP/IP resources of the host or the bandwidth of the network itself (Savage, Wetherall, Karlin, & Anderson, 2000). Like IP spoofing, denials

of service usually exploit TCP/IP trust and also normally involve some effort to conceal the identity of the perpetrator. An important protocol vulnerability here is based on how the TCP layer responds to an incoming request from the network for communication. TCP trusts that such requests are legitimate and therefore, upon receipt, it automatically reserves some of its limited resources for the expected pending communication. By flooding a host with bogus requests and withholding follow up from the presumed senders, a malicious perpetrator literally can “choke” the host’s communication capacity by keeping its resources in the “pending” mode. Alternatively, by flooding the network with bogus traffic, a perpetrator can consume bandwidth effectively preventing legitimate traffic from reaching its intended destination.

Covert channels are forms of communication hidden or disguised within what appears to be legitimate network traffic (Smith, 2000). Using such channels, intruders may be able to gain unauthorized access to networked computing resources. As Smith (2000) indicates, certain Internet protocols, like TCP/IP, are susceptible to this potential problem. For example, information directed to or from the TCP layer is marked with a unique identification “header.” Generally, in the TCP/IP suite, each layer has its own distinct header. Certain spaces within these headers may be “reserved for future use” and therefore may not be checked or screened reliably. This space thus offers a vehicle by which a covert channel could be established. Data placed in this channel normally would not be subject to scrutiny within the TCP/IP protocol and therefore might be used for malicious purposes. Smith (2000) has reviewed several examples of this kind.

Other Covert Exploits

Bauer (2001) has identified several other forms of covert exploitation involving Internet protocol features. Unlike the circumstances described above, these exploits are not malicious

and appear to be largely harmless, yet they represent unauthorized uses of networked computing resources. Many of Bauer's examples apply to the uppermost layer of the TCP/IP protocol and therefore involve application services like email and HTTP rather than the inner workings of TCP/IP itself. For example, one way to exploit email systems as a means of temporary storage is by sending self-addressed mail through an open mail relay system and then disabling receipt until desired. For at least some period of time, the relay system thus will serve as a temporary storage unit. A similar exploit can be accomplished using a web server that is instructed to store information of interest to the server owner within cookies on the computers of those who browse to a particular webpage hosted on the server. Assuming they will eventually return to the site, the people with the cookies on their computers are unwittingly providing temporary storage to the server owner.

PARASITIC COMPUTING

As noted above, the proof-of-concept demonstration of parasitic computing reported by Barabasi et al. (2001) essentially was an experiment in distributed computing in which a complex problem was decomposed into computational elements that each had a binary, yes or no, outcome. The parasitic computer then "out-sourced" these elements to multiple web servers across the Internet. Each server receiving an element unwittingly performed its task and reported its binary outcome back to the parasite. The "participating" servers were induced to perform their tasks through another form of TCP/IP vulnerability. As Barabasi et al. (2001) note, their demonstration was predicated on the fact that "the trust-based relationships between machines connected on the Internet can be exploited to use the resources of multiple servers to solve a problem of interest without authorization" (p. 895). To understand how this was done, we need to look again at the TCP/IP protocol.

TCP Checksum Function

One feature of the TCP protocol that is very important to the Barabasi et al. (2001) implementation of parasitic computing is the checksum property. Checksum is that part of TCP layer operation that is responsible for insuring integrity of packet data being sent over the Internet. Before a packet is released to the IP layer (see Fig. 1) of the sending computer, TCP divides the packet information into a series of 16-bit words and then creates a one's complement binary sum of these words. The resulting so-called "checksum" value is a unique representation of the totality of information in that packet. The bit-wise binary complement of this checksum is then stored in the TCP header before the packet is sent. When the packet arrives at the receiving computer, the TCP layer there performs its own binary sum of all the information in the packet including the checksum complement. If the packet was received without corruption, the resultant sum should be a 16-bit value with all bits equal to 1 since the original checksum (i.e., the total arrived at by the sending computer) and its exact complement would be added together forming a unitary value (see Barabasi, et al., 2001, Fig. 2, for more details). If this occurs, the packet is retained as good and is passed to the application layer for action; if not, the packet is dropped and TCP waits for a pre-arranged retransmission of the packet by the sending computer.

As Freeh (2002) indicates, the TCP checksum function performed by the receiving computer is, in essence, a fundamental "add-and-compare" procedure, which forms the basis for any other Boolean or arithmetic operation. As a consequence, TCP can be exploited to perform computations without "invading" (i.e., hacking or cracking into) those systems induced to participate (Barabasi, et. al, 2001; Freeh, 2002). In this sense, then, parasitic computing is a "non-invasive" form of covert exploitation that does not penetrate beyond the TCP/IP layers of

the host. This differentiates parasitic computing from the other methods described above for capitalizing on IP-related vulnerabilities.

NP-complete Satisfiability Problem

To demonstrate how this exploitation of the TCP checksum function was possible, Barabasi et al. (2001) elected to solve an NP-complete satisfiability (SAT) problem via distributed computing. As described by these authors, the specific version of the problem was a 2-SAT variant involving a Boolean equation with 16 binary variables related by AND or XOR operators (see Barabasi, et al., 2001, Fig. 3, for more details). The method used to solve this problem involved parallel evaluations of each of the 2^{16} possible solutions. To accomplish these parallel evaluations, a TCP/IP Checksum Computer (TICC) was devised (see Freeh, 2002) that could construct messages containing candidate solutions to the problem, which were then sent, along with a template for determining the correct solution, over the Internet to a number of target web servers in North America, Europe, and Asia. Similar to the behavior of a biological “parasite,” the TICC acted to take advantage of the targeted “hosts” by inducing them to evaluate the candidate solutions they received against the correct solution template and return for each one a binary, “yes/no” decision to the TICC.

Inducement without security compromise (i.e., invasion) was achieved by exploiting the TCP checksum function on each targeted host. The parasitic TICC constructed special message packets for each host and injected them directly into the network at the IP layer. These messages contained one of the possible 2^{16} candidate solutions encoded as packet data in two sequential 16-bit words, along with a 16-bit version of the correct solution (in complemented form) substituted in place of the normal TCP checksum value. When the packet was received by the host computer, the two 16-bit words containing the candidate solution, which were

presumed by the host to be packet data, were added together with the complemented checksum (i.e., the correct solution) according to the usual operation of the TCP checksum function described above (see Barabasi, et al., 2001, Fig. 3, for more details). If the enclosed candidate solution was a correct one, then its one's complement binary sum would combine with the complemented correct solution, masquerading as the TCP checksum, to form a 16-bit unitary value, just as would occur if normal packet data had been transmitted without error. In response to a unitary sum, the host's TCP layer passed the packet up to the HTTP application layer, acting as if the packet were not "corrupted." However, because the packet's message was artificial and thus unintelligible to the host, it was prompted to send a response back to the parasitic TICC saying it did not understand the message. This returned response was an indication to the parasite that the candidate solution was, in fact, a correct one, a decision made automatically, but unwittingly, by the host in response to the "artificial" packet. Messages containing an incorrect solution failed the checksum test on the host and were presumed to be corrupt; therefore, no response was sent back to the parasite as per the standard behavior of TCP.

Barabasi et al. (2001) acknowledge the possibility that "false negatives" could have complicated their demonstration of parasitic computing. This complication arises from the fact that incorrect solutions were signified by no return response from the host. However, a lack of returned responses also could be caused by other means such as artificial packets that never made it to their intended hosts for evaluation, or by host responses that got lost on the way back to the parasite. So, this means that correct solutions could be erroneously categorized as incorrect if a false negative occurred. Reliability tests by the authors performed by repeatedly

sending correct solutions to hosts showed that false negatives occurred no more than 1 percent of the time and sometimes less than 1 in 17,000 cases (Barabasi et al., 2001, pp. 896-897).

ETHICAL ISSUES RAISED BY PARASITIC COMPUTING

As the first author of this chapter has noted elsewhere (Barger, 2001a), most ethical problems considered under the rubric of Internet Ethics are basically variants of older ethical issues (e.g., theft, copyright infringement, invasion of privacy) disguised in modern-day (i.e., electronic or digital) clothing. Parasitic computing may be unique, however, in that on the surface it seems to present a truly original conundrum: namely, whether or not it is ethical for a parasitic computer to generate Internet-based communications in a manner that induces remote “host” computers, freely attached to the Internet by their owners, to perform computations of interest to the parasite without the knowledge or permission of host owners. The ethical “gray area” here arises from the fact that the specific host resources targeted by the parasite already were part of the “public domain” by virtue of being attached to the Internet. Moreover, these resources were not instigated to do anything malicious or even out of the ordinary. However, the uses to which the host resources were put by the parasite clearly were not sanctioned in any explicit way by the host owners.

It is perhaps a truism to say that ethical behavior must be assessed against standards inherent within an accepted moral code. Is there an accepted “moral code” for computer ethics? In a white paper published by the Computer Ethics Institute, Barquin (1992) presented what he called the “Ten Commandments of Computer Ethics,” which amounts to a list of moral imperatives to guide ethical behavior related to the use of computing and information technology resources. These guidelines have become fairly well known and have been endorsed by other professional societies (e.g., Computer Professionals for Social

Responsibility, 2001). Barquin's "commandments" overlap with similar strictures contained in a statement published by the Association for Computing Machinery entitled the "ACM Code of Ethics and Professional Conduct" (Association for Computing Machinery, 1992). For purposes of the present discussion, certain of Barquin's "commandments" appear directly relevant to the ethics of parasitic computing.

Thou shalt not use a computer to harm others or interfere with their computer work

These imperatives, abstracted from Commandments 1 and 2, clearly position as unethical any form of "malware" or other type of covert exploitation of computer resources with harmful purpose or consequences. Benign forms of exploitation without mal-intent, like the Barabasi et al. (2001) demonstration of parasitic computing, would seem under this mandate to be an instance of "no harm, no foul." One difficulty here, however, lies with the assessment of harm. Directly harmful effects to a user as a result of someone else's covert exploitation are one thing, but indirect consequences may be quite another. How does one know for sure what might have happened had the covert exploitation not been ongoing, or what might have been precluded by its presence? In an interview with one of the parasitic project authors, reported in the Sept. 6, 2001, issue of *Security Wire Digest*, Vincent Freeh said this about that: "It's sort of like meeting a friend and leaving one car in a shopping center parking lot. You've used the facilities in an unintended way that doesn't benefit the provider of the parking lot--most people wouldn't consider that unethical. But if you bring in a fleet of cars, impacting people who come to do business at the store, I think that's unethical" (McAlearney, 2001).

The second difficulty, alluded to in the above comment by Freeh, arises from the potential for future harm given an escalation of the exploitation. As the authors of an on-line dictionary, The Word Spy, put it in comments under their entry for 'parasitic computing:' "The bad news

is that, although messing with a few checksums won't cause a perceptible drop in the performance of the server, hijacking millions or billions of checksum calculations would bring the machine to its digital knees. It's just one more thing to keep Web site administrators chewing their fingernails" (The Word Spy, 2001). The authors themselves acknowledge this possibility in stating that (a presumably escalated form of) parasitic computing "could delay the services the target computer normally performs, which would be similar to a denial-of-service attack, disrupting Internet service" (Barabasi et al., 2001, p. 897). But, as Freeh (2002) points out, the TICC implementation employed by these authors is not likely an effective platform for the launch of denial-of-service attacks.

In the final analysis, then, we are not convinced that the demonstration of parasitic computing reported by Barabasi et al. (2001) was harmful in any sense. However, the potential may well exist for harmful effects stemming from an elaborated or escalated form of parasitic computing.

Thou shalt not use others' computing resources without authorization or snoop in their files

These imperatives, adapted from Barquin's Commandments 2 and 7, potentially pose the most serious ethical challenges for parasitic computing and also highlight the ethical "gray area" associated with this matter. Parasitic computing does not appear to us in any way to be a form of invasive "hacking." We thus agree with the authors' contention that "unlike 'cracking' (breaking into a computer) or computer viruses, however, parasitic computing does not compromise the security of the targeted servers..." (Barabasi et al., 2001, p. 895). Clearly, there was no "snooping," data corruption, or even residue (back doors, etc.) as a result of their implementation of parasitic computing. Moreover, it is important to emphasize that lack of "awareness" is not necessarily the same thing as lack of authorization. As the authors note,

their version of parasitic computing operated “without the knowledge of the participating servers” (Barabasi et al., 2001, p. 895), but does this mean it was unauthorized?

To answer this question we must pose two related queries. One is: To what extent is this TICC version of parasitic computing just a form of normal Internet communication? Barabasi et al. (2001) contended that “parasitic computing moves computation onto what is logically the communication infrastructure of the Internet, blurring the distinction between computing and communication” (p. 897). If this is true, then parasitic computing as implemented by these authors involves nothing more than mere Internet traffic. From the viewpoint of the receiving “host” computers, this certainly was true. As noted above, the specially prepared packets sent by the parasite were reacted to in exactly the same way as all other Internet traffic and were indistinguishable from “normal” packets in terms of their consequences for the host, at least at the level of TCP. That the special packets sent by the parasite were not “normal” does not automatically imply that they exercised a form of unauthorized access.

Therefore, a second important question is: To what extent does the owner of a web server consent implicitly to the exercise of TCP/IP functions on that machine, which happen automatically because of network traffic anyway, by virtue of the fact that it was connected to the Internet in the first place? There are different viewpoints here. On the parasitic computing website at Notre Dame, under a FAQ section, the authors both ask and answer the following question: "How do I reliably stop parasitic computing from occurring on my web server? [Answer] Unplug it from the net" (Parasitic Computing, 2001). This answer certainly suggests that any computer connected to the Internet has made an implicit acknowledgement that TCP/IP functions may be exercised on that machine. Barabasi et al. (2001) state their position

on this matter even more clearly by saying that parasitic computing “accesses only those parts of the servers that have been made explicitly available for Internet communication” (p. 895).

But, is connecting a computer to the Internet the same thing as giving permission for any conceivable use of TCP/IP? Server owners could argue that, even if one grants that those parts of a computer connected to the Internet are in the public domain, there is still a reasonable expectation that Internet traffic will be constituted only in conventional ways. When it is not so constituted, this could be construed as a form of protocol abuse.

However, what exactly is abusive about the exercise of TCP/IP functions on a web server by non-standard Internet traffic? An analogous question could be posed about “non-standard” uses of any other physical sensors (e.g., motion sensors, electric eyes, etc.) located in the public domain? The most obvious answer here would focus on categorizing any exercise of public domain resources for purposes other than they were intended as being forms of abuse. By this view, electing to sleep on a picnic table in a public park or picnic on a children’s merry-go-round would constitute abusive behavior, just as would asking a remote computer to help solve a problem that is of no interest to its owner. Again, however, there is not unanimous accord on this matter. Two of the parasitic computing researchers expressed a different view, as quoted in a Nature Science Update about parasitic computing by Whitfield (2001). In that article, Jay Brockman was quoted as saying: “The web is a source of information. If someone can contrive a question so as to get an answer other than the one intended, I view that as clever, rather than subversive.” Along the same lines, Freeh reportedly indicated, “If you have a public service, I have the ability to use it for a purpose that you didn't intend” (Whitfield, 2001).

Yet, surely, there must be some limits on unintended uses of public or private resources, some line of demarcation between acceptable and abusive. Of course, this is the very problem

with “gray areas.” In them, it is often very difficult to discern sought-after boundary lines. As a result, some like Mark Rash, an attorney who used to prosecute hackers, will say that parasitic computing probably qualifies as trespassing, although it was done without malice (National Public Radio, 2001). Others, like Freeh, will see it as being more like parking one’s car in a shopping center’s lot without going in to shop: no harm, no foul. In the end, with respect to the use of public resources, we are persuaded that harm should be the arbiter of abuse.

The principal reason for our “harm being the arbiter of abuse” position in relation to parasitic computing lies in the shared nature of the Internet as a public resource. Traditional views of ownership rights involving the idea that an owner has sole control over use and profit from personal property (Calabresi & Melamed, 1972) do not apply strictly, in our opinion, when personal and public property are juxtaposed in a dependency relationship as they are in the context of computers being connected to the Internet. Because the act of connecting to the Internet in order to receive communication services is optional and elective, such action necessarily involves an implicit consent to a form of property sharing that trumps sole propriety. This act is analogous to placing a mailbox on the public easement in front of one’s house in order to receive postal service. In these instances, personal property (mailboxes or TCP/IP layers) are made explicitly available to the public so that valued communication services thereby can be obtained. The dependency of the private upon the public arises here because one’s personal communications in these instances must pass through a public handling system. Of necessity, this means that one’s mailbox or TCP/IP protocol is subject to public interrogation, which amounts to a kind of shared relationship between owner and public.

Of this shared relationship, at least two things can be said. First, as Barabasi et al. (2001) have noted, such actions are undertaken by an owner with a sense of trust that no harm will befall that which is made publicly accessible. In the case of a mailbox this means a trust that it will not be smashed, painted shut, locked, or stuffed so full of irrelevant material that intended mail will no longer fit. Any such circumstances would constitute a violation of this trust and therefore would harm the owner. Similarly, in making an Internet connection, a computer owner trusts that no one will attempt to invade or gain access to the computer, affect its inner workings, or attempt to deny or circumvent its access to the public communications system by means of the shared TCP/IP layer made available to the public.

Second, the nature of the implicit consent involved is limited. In the case of the mailbox, the limitation restricts public access only to the mail receptacle and does not extend it to the yard or the house. For the computer owner, consent to public access is limited to the TCP/IP layer and is not extended generally to the RAM, processor, hard drive, or other peripheral resources connected to the computer.

PHILOSOPHIC PERSPECTIVES

We now briefly examine the two basic opposite worldviews, Idealism and Pragmatism, along with the ethical viewpoints to which these positions give rise as they relate to parasitic computing. The Idealist derives greater meaning from ideas than things. Ideas do not change, so reality is basically static and absolute. The Idealist philosopher Immanuel Kant used what he called the “Categorical Imperative” to assess the ethics of any action. The first form of his Categorical Imperative states: "Act only on that maxim by which you can at the same time will that it should become a universal law" (The Internet Encyclopedia of Philosophy, 2001). In other words, if you wish to establish (or adhere to) a particular moral or ethical standard, you

must be willing to agree that it would also be right for anyone else to follow that standard. It seems, then, from an Idealist perspective that the originally intended purposes of things (i.e., the ideas upon which they were based) would weigh heavily in ethical judgments regarding their use or abuse. Thus, using anything, even an Internet protocol, for any purpose other than its intended one would be unethical on this view.

The Pragmatist finds meaning neither in ideas nor things. Rather, it is found in experience or change. Pragmatism might therefore be regarded as a more expedient approach than Idealism. Since this philosophical view holds that everything can change, there is no permanent essence or identity. In terms of ethical behavior, this outlook implies that all moral values must be tested and proven in practice since nothing is intrinsically good or bad. If certain actions work to achieve a socially desirable end, then they are ethical and good (Barger, 2001b).

In contrast to a Kantian Idealist perspective, Barabasi et al. (2001) seem to take, if anything, a Pragmatist approach to the ethics of parasitic computing. The Pragmatist prefers to look at intent or consequences to see how it could affect the morality of an action, whereas the Idealist would concentrate on the intrinsic character of the act itself, something the Idealist believes is unaffected by intent or consequences. If a complex problem can be solved more effectively by distributing the computational load among many remote computers that are induced to perform calculations unwittingly, then such actions are acceptable, provided they represent some form of "greater good."

We must acknowledge, however, that these authors depart from full-blown Pragmatism when they note that their demonstration of parasitic computing currently is a very inefficient way to implement distributed computing. To this effect they state: "To make the model viable, the computation-to-communication ratio must increase until the computation exported by the

parasitic node is larger than the amount of cycles required by the [host] node to solve the problem itself instead of sending it to the target. However, we emphasize that these are drawbacks of the presented implementation and do not represent fundamental obstacles for parasitic computing. It remains to be seen, however, whether a high-level implementation of a parasitic computer, perhaps exploiting HTTP or encryption/decryption could execute in an efficient manner" (Barabasi et al., 2001, p. 897). Obviously, this statement openly questions the effectiveness (i.e., the "greater good") of their method. It also illuminates the need for further research on this topic. Also, these authors are less than fully pragmatic when they express concern about the potential harm that could be caused by parasitic computing were it to be conducted with malicious intent. As we have mentioned above, Barabasi et al. (2001) explicitly acknowledge the possibility of denial-of-service-like attacks using their general approach, however inefficient such attacks might be. Such openly stated concerns by these authors once again challenge the possibility that parasitic computing may conform to the pragmatist's emphasis upon the "greater good." Regardless of one's position on the ethics of parasitic computing, the motivation behind this line of research is laudable. As stated on the parasitic computing web page, these researchers say: "By publishing our work we wish to bring the Internet's various existing vulnerabilities to the attention of both the scientific community and the society at large, so that the ethical, legal and scientific ramifications raised by it can be resolved" (Parasitic Computing, 2001).

CONCLUSIONS AND FUTURE TRENDS

The demonstration of parasitic computing by Barabasi et al. (2001) has provided us with an interesting and provocative example of how many computers can be recruited to work together in pursuit of solutions to complex problems even when the respective participants are unaware

of their involvement. How is one to evaluate the ethics of parasitic computing? Even the researchers involved seem to differ among themselves on this question. Judging by their published remarks, Vincent Freeh and Jay Brockman do not find ethical problems with the covert exploitation involved in their demonstration of parasitic computing. In contrast, when asked the question "Was it ethical?" in a National Public Radio interview, Albert-Laszlo Barabasi replied: "That's a very good question. My thinking [is] that it is not. And that was one of the reasons actually why we haven't really pushed much further" (National Public Radio, 2001). Such disagreement among principals is vivid testimony to the above-noted ethical "gray area" associated with parasitic computing, an area which we think calls for ongoing examination and debate.

The question of where research on parasitic computing is headed is an interesting one from several perspectives. From an efficiency standpoint, as noted above, some attention could be focused on how the overall paradigm can be improved along the lines noted by Barabasi et al. (2001): "To make the model viable, the computation-to-communication ratio must increase until the computation exported by the parasitic node is larger than the amount of cycles required by the node to solve the problem itself instead of sending it to the target" (p. 897). One way to do this, as noted by Freeh (2002), is to increase the number of packets sent by a parasite to a single host. Such an approach also would increase the importance of dealing with the reliability problems associated with parasitic computing (i.e., false negatives and false positives). In terms of the TCP/IP vulnerability exploited by parasitic computing, Freeh (2002) has noted that more work is needed to better define and understand the viability and level of threat associated with various forms of exploitation inherent in Internet protocols. Finally, with respect to a justification for parasitic computing, Vinton Cerf, the co-inventor of the TCP/IP

protocol suite, has noted that "one should also consider that compute cycles are highly perishable-if you don't use them, they evaporate. So, some justification [for parasitic computing] might be found if the arriving packets had lowest possible priority for use of the computing cycles." He would not, however, condone the use of parasitic computing on machines whose owners had not authorized such use (personal communications, September 12 & 21, 2003). This observation suggests that a form of parasitic computing that took advantage of "cycle priority" and/or a "cycle donor pool" might be a less ethically challenging alternative.

References

- Association for Computing Machinery. (1992). ACM Code of Ethics and Professional Conduct. Retrieved September 9, 2003, on the World Wide Web:
<http://www.acm.org/constitution/code.html>
- Barabasi, A.-L., Freeh, V. W., Jeong, H., & Brockman, J. B. (2001). Parasitic computing. *Nature*, 412, 894 – 897.
- Barger, R. N. (2001a). Is computer ethics unique in relation to other fields of ethics? Retrieved September 9, 2003, on the World Wide Web: <http://www.nd.edu/~rbarger/ce-unique.html>
- Barger, R. N. (2001b). Philosophical Belief Systems. Retrieved September 9, 2003, on the World Wide Web: <http://www.nd.edu/~rbarger/philblfs.html>
- Barquin, R. C. (1992). In pursuit of a ‘ten commandments’ for computer ethics. Computer Ethics Institute. Retrieved September 9, 2003, on the World Wide Web:
http://www.brook.edu/its/cei/papers/Barquin_Pursuit_1992.htm
- Bauer, M. (2001). Villian-to-victim computing and applications: Abuse of protocol features. Retrieved September 9, 2003, on the World Wide Web:
<http://www1.informatik.uni-erlangen.de/~bauer/new/v2v.html>
- Bellovin, S. M. (1989). Security problems in the TCP/IP protocol suite. *ACM Computer Communications Review*, 19(2), 32-48.
- Calabresi, G., & Melamed, D. A. (1972). Property rules, liability rules and inalienability: One view of the cathedral. *Harvard Business Review*, 85, 1089-1128.
- Cohen, F. (1992). A formal definition of computer worms and some related results. *IFIP-TC11 Computers and Security*, 11(7), 641-652.
- Computer Professionals for Social Responsibility. (2001). The ten commandments of computer

ethics. Retrieved September 9, 2003, on the World Wide Web:

<http://www.cpsr.org/program/ethics/cei.html>

Freeh, V.W. (2002). Anatomy of a Parasitic Computer. *Dr. Dobb's Journal*, January, 63-67.

McAlearney, S. (2001). Parasitic computing relatively benign. *Security Wire Digest*, 3(68).

Retrieved September 9, 2003, on the World Wide Web:

<http://infosecuritymag.techtarget.com/2001/sep/digest06.shtml>

National Public Radio. (2001). *All things considered*, August 29. Retrieved September 9, 2003,

on the World Wide Web: <http://www.npr.org/ramfiles/atc/20010829.atc.14.ram>

Parasitic Computing. (2001). Retrieved September 9, 2003, on the World Wide Web:

<http://www.nd.edu/~parasite/>

Savage, S. Wetherall, D., Karlin, A., & Anderson, T. (2000). Practical network support for IP

traceback. *Proceedings of the 2000 ACM SIGCOMM Conference*, August, 295-306.

SETI@home. (2003). Retrieved September 9, 2003, on the World Wide Web:

<http://setiathome.ssl.berkeley.edu/>

Smith, J. C. (2000). Covert shells. Retrieved September 9, 2003, on the World Wide Web:

<http://gray-world.net/papers/covertshells.txt>

Stevens, W. R. (1994). *TCP/IP Illustrated, Volume 1*. Reading, MA: Addison-Wesley.

The Internet Encyclopedia of Philosophy. (2001). Immanuel Kant (1724-1804): Metaphysics.

Retrieved September 9, 2003, on the World Wide Web:

<http://www.utm.edu/research/iep/k/kantmeta.htm>

The Word Spy. (2001). Entry for "parasitic computing" (posted Dec. 6, 2001). Retrieved

September 9, 2003, on the World Wide Web:

<http://www.wordspy.com/words/parasiticcomputing.asp>

Velasco, V. (2000). Introduction to IP Spoofing. Retrieved September 9, 2003, on the World

Wide Web: http://www.giac.org/practical/gsec/Victor_Velasco_GSEC.pdf

Whitfield, J. (2001). Parasite corrals computer power. *Nature*, August 30. Retrieved September

9, 2003, on the World Wide Web: <http://www.nature.com/nsu/010830/010830-8.html>

Wiggins, G. (2001). Living with malware. Sans Institute. Retrieved September 9, 2003, on

the World Wide Web: <http://www.sans.org/rr/paper.php?id=48>

Figure Caption

Figure 1. Layers of the TCP/IP protocol. Adapted from Stevens, W. R. (1994). *TCP/IP*

Illustrated, Volume 1. Reading, MA: Addison-Wesley, Figure 1.1.