International Journal of Advance Research, IJOAR .org
Volume 1, Issue 11, November 2013, Online: ISSN 2320-9194

PARASITIC COMPUTING: PROBLEMS AND ETHICAL CONSIDERATION

Abstract

Parasitic computing is programming technique where a program in normal authorized interactions with another program manages to get the other program to perform computations of a complex nature. It is, in a sense, a security exploit in that the program implementing the parasitic computing has no authority to consume resources made available to the other program.The paper takes a look at the ethical issues of parasitic computing and suggest a look into the current operation of the internet TCP/IP.

Keyword:

parasitic computing, internet, TCP/IP

Introduction

Parasitic computing is programming technique where a program in normal authorized interactions with another program manages to get the other program to perform computations of a complex nature. It is, in a sense, a security exploit in that the program implementing the parasitic computing has no authority to consume resources made available to the other program.

In this model, which we call `parasitic computing'; one machine forces target computers to solve a piece of a complex computational problem merely by them in standard communication. Consequently, the target computers are unaware that they have performed computation for the benefit of a commanding node. As experimental evidence of the principle of parasitic computing, we harness the power of several web servers across the globe, which-unknown to them-work together to solve an NP complete problem. Unlike `cracking' (breaking into a computer) or computer viruses, however, parasitic computing does not compromise the security of the targeted servers, and accesses only those parts of the servers that have been made explicitly available for Internet communication.

Like the "Search for Extraterrestrial Intelligence" SETI@home project Philips(1999), parasitic computing decomposes a complex problem into computations that can be evaluated independently and solved by computers connected to the Internet; unlike the SETI project, however, i 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 . In effect, SETI has created a super computer from millions of individual computers working in concert.it does so without the knowledge of the participating servers.(Robert et al,2003)

This is a type of distributed computing technique known as parasitic computing invented by computer scientists of the University of Notre Dame and questions its practice ethically. In August 2001, four researchers at the University of Notre Dame – Albert-László Barabási, Vincent W. Freeh, Hawoong Jeong and Jay B. Brockman invented a very sophisticated computing technique known as parasitic computing based upon this behavior of TCP/IP (Barabási, Freeh, Jeong, & Brockman, 2001). A reliable communication over internet via TCP/IP is a complex process and requires a significant amount of computation to validate the integrity of the datagram being sent and received between two nodes. The integrity of a data segment is maintained by validating the result of certain operations on the bytes of 16-bit Checksum field in its TCP packet. Figure below displays a TCP pseudo-header with 16 bit checksum field starting at bit offset 224.

TCP checksum function

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 exactcomplement would be added together forming a unitary value (Barabasi, et al., 2001). 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. 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.

Bit offset	0–3	4–7	8–15	16–31
0	Source address			
32	Destination address			
64	Zeros		Protocol	TCP length
96	Source port			Destination port
128	Sequence number			
160	Acknowledgement number			
192	Data offset	Reserved	Flags	Window
224	Checksum			Urgent pointer
256	Options (optional)			
256/288+	Data			

Fig.1

Literature review

How communication over internet via TCP/IP works

Consider a scenario where a user is trying to visit a website. When user informs a browser the website URL (uniform resource locator), the browser opens a transmission control protocol (TCP) connection and connects to the web server.
After establishing this connection, browser issues a hyper-text transmission protocol (HTTP) request via already opened TCP connection. This TCP message is then carried to the destination (web server) via internet protocol (IP). In this process of transmitting message from source (user) to destination, IP might break entire message into several pieces commonly addressed as TCP packets. These packets are then transmitted to the destination IP address via different routes. Once the destination receives all packets, a response is returned to the source via the same TCP channel. The original message is then reassembled
via consecutive steps involving TCP and IP and is interpreted as HTTP request. After that, the
web server sends a response (webpage HTML) back to the user (CISCO). Thus, even such a simple communication over internet requires significant amount of computation at all network stages and only cooperation and trust between all involved parties can guarantee a successful communication over internet.

In parasitic computing, this trust based relationship of machines connected to the network is exploited to make other machines perform a certain mathematical operations on certain data without an authorization. Albert-László, Vincent, Hawoong and Jay used a parasitic computer to solve the well known NP-complete satisfiability problem, by engaging various web servers physically located in North America, Europe, and Asia, each of which unknowingly participated in the experiment Babarasi etal,2001.
Like SETI@home project, parasitic computing decomposes a problem into several small problems which are mutually exclusive and can be solved independently via machines connected to the network. Parasitic computing can be a very effective technique when it comes to solve NP Complete problems such as Circuit SAT, 3 SAT, etc.
These problems are currently considered as some of world's most complex and time consuming problems. These problems generally have a set of solutions which itself is a subset of a set of possible solutions.

This behaviour can be described as the following:
$S \subset \{s1,s2,s3...sn\}, n>0$
Although any possible solution to such problems can be verified quickly, there is no known efficient way to identify a solution in the first place. In fact, the most notable characteristic for such problem is that there is no fast solution. The time required to solve such problem is exponentially proportional to the size of the problem. So, as the size of the problem grows, the time required to find all solutions of the problem grows exponentially. In fact, time required to solve a moderately large NP-Complete problem can easily reach billions if nottrillions of years using any kind of modern computing technology we have available today. For this reason, even just determining whether there is a fast solution to such problems or not is one of the principal unsolved problems of computer science.

Methodology

Two computers communicating over the Internet, under disguise of a standard communications session. The first computer is attempting to solve a large and extremely difficult 3-SAT problem; it has decomposed the original 3-SAT problem in a considerable number of smaller problems. Each of these smaller problems is then encoded as a relation between a checksum and a packet such that whether the checksum is accurate or not is also the answer to that smaller problem. The packet/checksum is then sent to another computer. This computer will, as part of receiving the packet and deciding whether it is valid and well-formed, create a checksum of the packet and see whether it is identical to the provided checksum. If the checksum is invalid, it will then request a new packet from the original computer. The original computer now knows the answer to that smaller problem based on the second computer's response, and can transmit a fresh packet embodying a different sub-problem. Eventually, all the sub-problems will be answered and the final answer easily calculated.

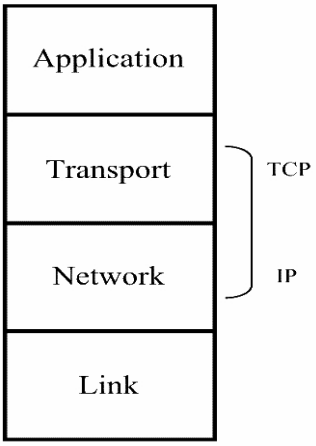

Figure 1. Layers of the TCP/IP

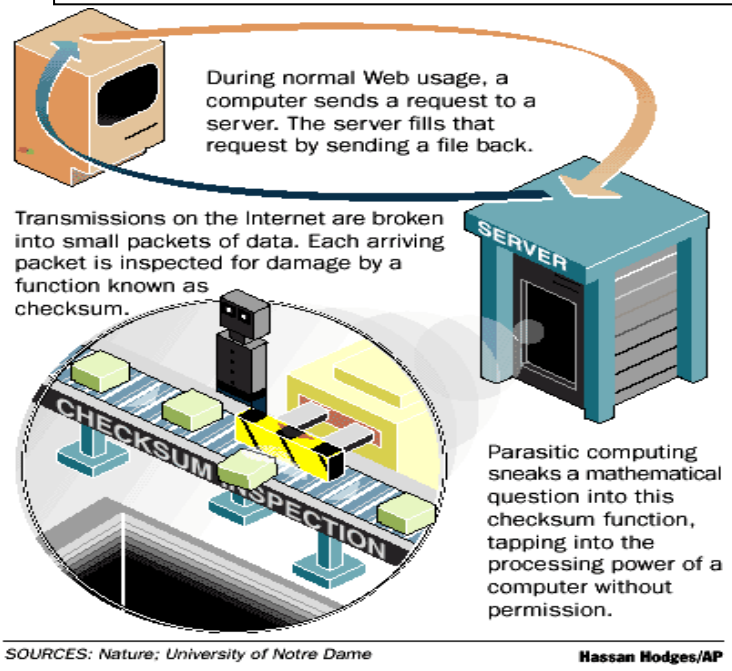

Fig 2.How parasitic computing works

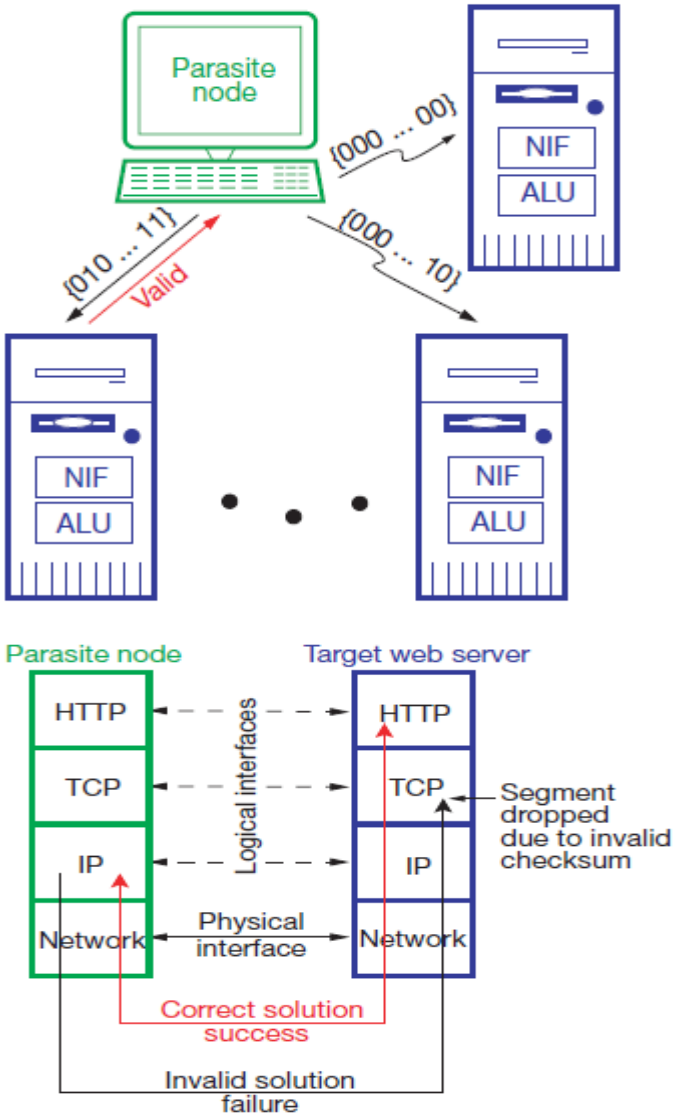

Figure 3.

Figure 2 describes how parasitic computing works. As described in figure 3.1, the parasitic computer starts the process by transmitting specially generated messages to number of targeted web servers consisting of arithmetic and logic unit (ALU) and a network interface (NIF). The packet carrying one of possible solutions to the problem is inserted into the IP level bypassing the parasitic node's TCP. This can be seen in figure 3.2. The parasitic computer generates a message in such a way that if the solution is not valid, it will fail the TCP checksum on the destination machine and the packet will be dropped. But in the case when the solution is correct, it will be propagated to the HTTP layer via TCP. Since it is a behavior of a web server to respond to any requests coming to an HTTP layer regardless of whether it understands the request or not, the web server will send a response back to the parasitic computer that it has received an HTTP request Mujal(2010).
Thus the parasitic computer sends out a message for each possible solution as described in figure 3.1 with black arrow, it only receives responses back from the server when the possible solution is a one of the actual solutions of the problem. This is displayed with a red arrow in the figure 3.1

Ethical considerations

Worms, Viruses, and Trojan Horses

Exploitation of computing resources has taken many forms over the years, some more malicious than others. Perhaps the most notorious 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.

IP-related Vulnerabilities

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. 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 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. intruders have used this technique to establish communications with remote computers, thereby potentially "spoofing" them into further vulnerabilities and/or unauthorized access.

Denials of DoS) 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. Denial of service usually exploit TCP/IP trust and also normally involve some effort to conceal the identity of the perpetrator.

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 Robert(2003)

Since most of the computers connected to the network will be using TCP/IP, the resources available to the parasitic computer are virtually unlimited and almost all of the computer can be exploited. Furthermore, there is a very high possibility that servers can allocate their valuable CPU cycles to do the processing commanded by the parasitic node thus degrading overall performance of the applications running on the server and access efforts of the normal application user similar to that in the Denial of Service attack (DoS). Ganti & Xiao, 2008).
In order for this technique to be widely accepted, potential users need to answer some important ethical questions. the speed at which this technique is capable of solving NP-Complete problems is thrilling! But what about the possibility of a DoS (unintentional or intentional) attack as discussed above? Another ethical questions like what if terrorists gain their expertise on this technique? But the final question I would like to ask is:
just like we patch security holes in our applications, is this possibly a time to rethink a better and more secured protocol for communication over the internet? Shouldn't the security of underlying internet protocols used by billions of users worldwide have equal priority for its updates and patches if not higher than any of normal applications?

Under the rubric of Internet Ethics are basically variants of older ethical issues (e.g.,
1. Theft
2. Copyright infringement
3. Invasion of privacy) disguised in modern-day (i.e., electronic or digital) clothing(Robert,2003)

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.

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.

Conclusion

We can't deny patch security holes in our applications, is this possibly a time to rethink a better of transmitting information and more secured protocol for communication over the internet.

References

1.Parasitic Computing by Munjal Patel,January 30, 2010

2. Barabási, A.-L., Freeh, V. W., Jeong, H., & Brockman, J. B. (2001, August 30). Parasitic computing. *letters to nature* , *412*, pp. 894-897.

3. Phillips, D. T. (1999, May 23). *ET, phone SETI@home!* Retrieved January 20, 2010, from NASA: http://science.nasa.gov/newhome/headlines/ast23may99_1.htm

4. "Parasitic Computing"Seminar by:Kunal Goswami 05IT6006

5. Ganti, R. K., & Xiao, L. (2008). *Detection of Parasitic Computing.* Indiana: University of Notre Dame.
6. Robert N. Barger and Charles R. Crowell, 2003 THE ETHICS OF "PARASITIC COMPUTING:" FAIR USE OR ABUSE OF TCP/IP OVER THE INTERNET? Computer Applications Program University of Notre Dame, Notre Dame, IN 46556
7. SETI@home. (2012). Retrieved September 9, 2012, on the World Wide Web: http://setiathome.ssl.berkeley.edu/
8. Stevens, W. R. (1994). *TCP/IP Illustrated, Volume 1*. Reading, MA: Addison-Wesley. 9.Wiggins, G. (2001). Living with malware. Sans Institute.
10. Freeh, V.W. (2002). Anatomy of a Parasitic Computer. *Dr. Dobb's Journal,* January, 63-67.
11. Association for Computing Machinery. (1992). ACM Code of Ethics and Professional Conduct
12. Barquin, R. C. (1992). In pursuit of a 'ten commandments' for computer ethics. Computer Ethics Institute