

Re: Parasitic Computing

Source: <http://www.derkeiler.com/Newsgroups/comp.os.linux.security/2002-01/0024.html>

From: Michael Erskine (osiris@deltaville.net)

Date: 01/01/02

From: osiris@deltaville.net (Michael Erskine)

Date: 1 Jan 2002 08:25:38 -0800

My earlier post was made to get people to look into the concept, read what they found and form an opinion.

This is exactly what I was referring to and I believe it was in either Dr. Dobb's or LJ that I first read of it... a month or so ago...

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

>

> Exactly, that has got to be a joke. A quote from the page:

>

> Our implementation of parasitic computing is

> not efficient. If it is made efficient, it

> could offer unlimited computational power.

Actually, my comments weren't a joke. Although at this point I don't view the technology as much of a threat, more of a toy. Even so, one can easily make the argument that <fill in the blank> was a toy early on.

>

> At least they know it is currently not usable for

> distributed computations.

They know that it is not usable at this time for significant distributed computations.

> But anybody with knowledge of

> networking would say that it is never going to be usable.

Anyone with knowledge of networking would say that *anything* is possible in a complex system.

> Checksums are designed to be fast to computer, so that

> any computer is able to produce outputpackets with

> correct checksum at the rate it can send them onto the

> network.

Which rate has essentially doubled every year for the past thirty

years.

- > *Computing the checksums locally is always going*
- > *to be easier, faster, and more reliable than sending*
- > *them to another computer and looking for a reply.*

Agreed. That is not the point. The point is that these people are able to *steal* computational time from a remote host, without cracking it. The computational time is a freebie result of the way IP works.

- >
- > *What remains is only that people can flood you with TCP*
- > *packets if they want to.*

What remains is for someone to find a better, faster, easier method of exploiting this concept.

Remember the prime directive of development?

"First make it work. Then make it work better, faster, smaller." It now works. It didn't in the year 2000. What will it be in the year 2020?

Here is my thinking on this:

Network bandwidth is (as you know) like a river flowing into the ocean. If you go up stream you find less and less bandwidth per node. Since this problem is essentially a bandwidth limitation problem the limits of the computational resource are well distributed thru the contributing nodes as a natural consequence of the evolution of the network itself. Upstream nodes, end user systems, have small bandwidth but large wasted computational power. Downstream nodes, server systems, have high bandwidth and less wasted computational power. Therefore stealing computational cycles (if it can be done efficiently) becomes lucrative to the BIG GUYS, not to the LITTLE GUYS.

If the parasite node is located on a small pipe in some upstream tributary, he will likely never be able to steal significant resources. If OTHO the parasite is downstream where the bandwidth flows "like a river" he can easily distribute a significant problem across thousands of much slower upstream nodes (victims).

The difference between this technology and the SeTi project is that SeTi is borrowing compute time from willing contributors. This technology can steal compute time from unknowing victims. Never mind that it is currently infeasible. This method of *mining* computational resources from the network itself is proof of a concept. At this time it is only an academically interesting concept.

Imagine a system based upon the concept. A parasite system, designed specifically to distribute a problem across a hundred million computers. In each processing cycle it forwards a hundred million bits (packets) to a hundred million computers. Maybe a processing cycle takes a day with current technology. How long will it take in five years, ten, fifty? Suppose that someone decides to develop a tiny little engine of some sort and stick it in every freakin' MicroSuX release forever. This little puppy does NOTHING but take in a UDP packet, do an add, and send it back. Sure we are talking about a completely different technology ... or are we? Maybe such a thing already exists? Maybe there is some way that one can get more back from the victim than a simple error message, maybe there is a way to get back 8, 16, maybe even 32 bits of compute response from the victim *ALREADY*.

At some point this technology will become a reality and no joke. Essentially we are looking at method of attacking massively parallel problems upon a vast scale, what remains to be done is make it practical.

That is the point.

Laterz

-m-

-
- **Next message:** [Ian Jones: "Re: \[suse\]\[apache\]\[ssl\] https://localhost => error messages"](#)
 - **Previous message:** [acva: "Re: I want to make a Linux IPSEC VPN"](#)
 - **In reply to:** [Kasper Dupont: "Re: Parasitic Computing"](#)
 - **Next in thread:** [Ian Jones: "Re: Parasitic Computing"](#)
 - **Next in thread:** [Tsu Dho Nimh: "Re: Parasitic Computing"](#)
 - **Reply:** [Ian Jones: "Re: Parasitic Computing"](#)
 - **Messages sorted by:** [\[date \] \[thread \] \[subject \] \[author \] \[attachment \]](#)