The Strange Tale of the

DENIAL OF SERVICE

Attacks Against GRC.COM

by Steve Gibson, Gibson Research Corporation

Page last modified: Jun 02, 2001 at 22:48

Nothing more than the whim of a 13-year old hacker is required to knock any user, site, or server right off the Internet.

I believe you will be as fascinated and concerned as I am by the findings of my post-attack forensic analysis, and the results of my subsequent infiltration into the networks and technologies being used by some of the Internet's most active hackers.

An Open Letter to the Internet's Hackers

What Happened?

On the evening of May 4th, 2001, GRC.COM suddenly dropped off the Internet:

GRC.COM Bandwidth Utilization

8:00 PM, Friday, May 4th
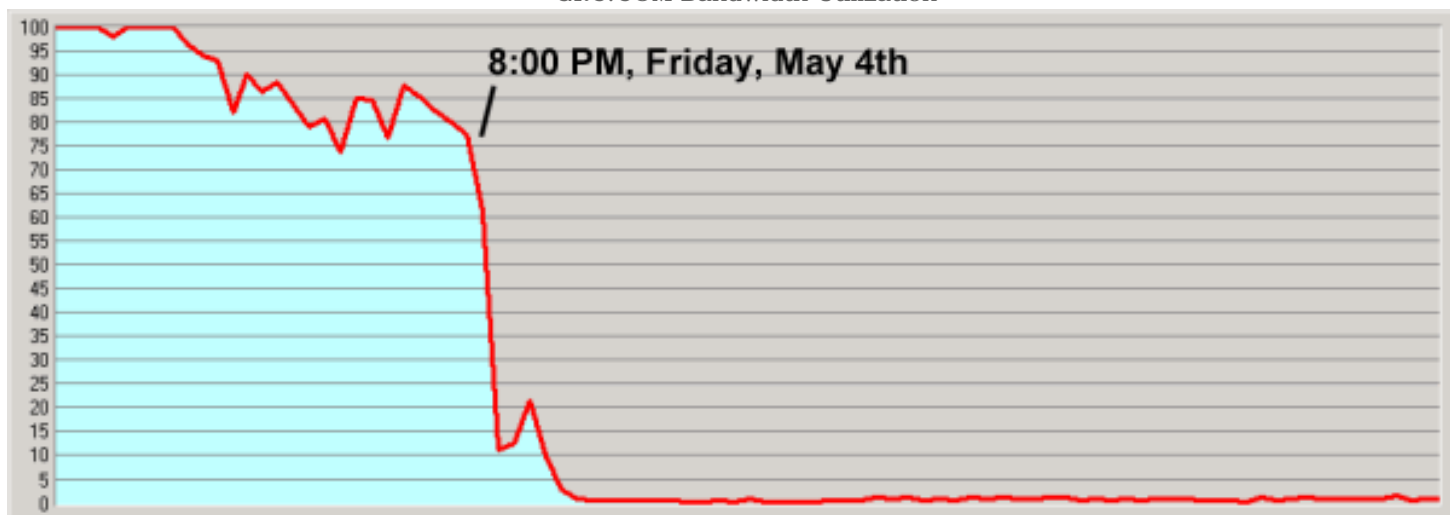
Stemming the Flood with Our ISP

Within a minute of the start of the first attack it was clear that we were experiencing a "packet flooding" attack of some sort. A quick query of our Cisco router showed that both of our two T1 trunk interfaces to the Internet were receiving some sort of traffic at their maximum 1.54 megabit rate, while our outbound traffic had fallen to nearly zero, presumably because valid inbound traffic was no longer able to reach our server. We found ourselves in the situation that coined the term: **Our site's users were being denied our services.**

I had two priorities: I wanted to learn about the attack and I wanted to get us back online.

I immediately reconfigured our network to capture the packet traffic in real time and began logging the attack. Dipping a thimble into the flood, I analyzed a tiny sample and saw that huge UDP packets — **aimed at the bogus port "666" of grc.com** — had been fragmented during their travel across the Internet, resulting in a blizzard of millions of 1500-byte IP packets. Mixed into this appeared to be ICMP debris from large-packet ping commands.

We were drowning in a flood of malicious traffic and valid traffic was unable to compete with the torrent.

At our end of our T1 trunks, our local router and firewall had no trouble analyzing and discarding the nonsense, so none of our machines were adversely affected. But it was clear that this attack was not attempting to upset our machines, **it was a simple brute force flood**, intended to consume all of the bandwidth of our connection to the Internet **. . .** and at that it was succeeding all too well.

Gibson Research Corporation is connected to the Internet by a pair of T1 trunks. They provide a total of 3.08 megabits of bandwidth in each direction (1.54 megabits each), which is ample for our daily needs.
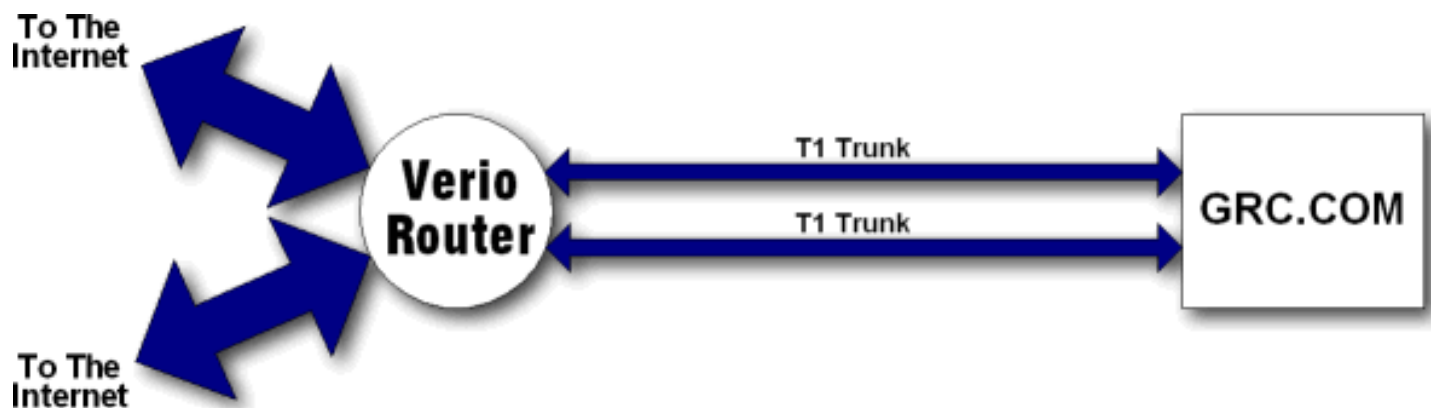
As you can see from the schematic diagram above, the Verio (our ISP) router that supplies our T1 trunks enjoys two massive 100 megabit connections to the Internet. But from there all of the traffic bound for us must be funnelled through our two T1 trunks. Therefore, in order for the congestion of our T1's to be relieved of the malicious traffic, the "bad packets" had to be filtered **before they left Verio's router**. In this way, the packet flood would be stopped at a high-bandwidth point — upstream of the T1 choke point — thus allowing the "good packets" to slip past the bad packets and cross our T1's in peace.

It took us a while . . .

Due to a comedy of changed telephone area-codes, mistyped eMail addresses, slept-through pager beeps, the attack's Friday night timing, and Verio continually insisting that I be processed through the regular channels of "the system" (which I kept explaining did not seem to be working), **seventeen hours passed before I was able to get a competent Verio engineer on the other end of the phone.** But once I had a competent engineer on the phone, and armed with my analysis of the attack pattern **. . .**

In two minutes we applied "brute force" filters to the Verio router, shutting down all UDP and ICMP traffic ... and GRC.COM instantly popped back onto the Internet.

That part was pretty cool. We were still very much under attack, but **because the attack was prone to filtering** (thank goodness) we were able to have Verio's router "weed out" the bad packets and return us to almost normal operation.

An Attack "Prone to Filtering" ?

Yes. Fortunately — as we'll see below — the attacking machines were all security-compromised Windows-based PC's. In a fluke of laziness (or good judgement?) that has saved the Internet from untold levels of disaster, Microsoft's engineers never fully implemented the complete "Unix Sockets" specification in any of the previous version of Windows. (Windows 2000 has it.) As a consequence, Windows machines (compared to Unix machines) are blessedly limited in their ability to generate deliberately invalid Internet packets.

It is impossible for an application running under any version of Windows 3.x/95/98/ME or NT to "spoof" its source IP or generate malicious TCP packets such as SYN or ACK floods.

As a result, Internet security experts know that non-spoofing Internet attacks are almost certainly being generated by Windows-based PC's. Forging the IP address of an attacking machine (spoofing) is such a trivial thing to do under any of the various UNIX-like operating systems, and it is so effective in hiding the attacking machines, that no hacker would pass up the opportunity if it were available.

It is incredibly fortuitous for the Internet that the massive population of Windows-based machines has never enjoyed this complete "Unix Sockets" support which is so prone to abuse. But the very bad news is **. . .**

This has horribly changed for the worse

with the release of Windows 2000 and the pending release of Windows XP.

For no good reason whatsoever, Microsoft has equipped Windows 2000 and XP with the ability FOR ANY APPLICATION to generate incredibly malicious Internet traffic, including spoofed source IP's and SYN-flooding full scale Denial of Service (DoS) attacks! (See my WinXP & DoS Page.)

While I was conducting research into the hacker world following these DoS attacks, I encountered evidence — in attack-tool source code — that malicious hackers are already fully aware of the massive malicious power of the new versions of Windows and are waiting impatiently for the "home version" of Windows XP to arrive in the homes of millions of less clueful end users.

When those insecure and maliciously potent Windows XP machines are mated to high-bandwidth Internet connections, we are going to experience an escalation of Internet terrorism the likes of which has never been seen before.

If I fail in my mission to convince Microsoft to remove this from Windows XP, the historical problems with Internet attacks promise to pale in comparison to what will begin happening as Windows XP is deployed next year.

Thanks to the fact that the fleet of attacking machines were Windows PC's, they were unable to send TCP SYN packets to our port 80 (which would have crippled us completely), and were only able to flood us with UDP and ICMP packets (which we could temporarily ignore).

Working with our ISP we were able to filter our receipt of the malicious packets before they were able to reach our T1 trunks. We were able to continue offering our TCP-based services (Web/FTP/News) even while under continuing attack.

The Attack Profile

We know **what** the malicious packets were, and we will soon see (below) exactly how they were generated. But we haven't yet seen where they all came from. During the seventeen hours of the first attack (we were subsequently subjected to several more attacks) we captured 16.1 gigabytes of packet log data. After selecting UDP packets aimed at port 666 **. . .**

I determined that we had been attacked by **474** Windows PC's.

This was a classic "Distributed" Denial of Service (DDoS) attack generated by the coordinated efforts of many hundreds of individual PC's.

- Where do these machines reside?
- Who owns them?
- Who are their ISP's?
- What sort of users are running Windows PC's infested with potent Internet attack Zombies?

A determination of the network domains hosting the attacking machines revealed the following, hardly surprising, cast of Internet end user service providers:

104	home.com	5	inreach.net	3	voyager.net
51	rr.com	5	telus.net	3	lvcm.com
20	aol.com	5	gtei.net	3	co.uk
20	mediaone.net	4	tpo.fi	2	cdsnet.net
17	uu.net	4	rcn.com	2	enter.net
14	btinternet.com	4	isoc.net	2	cgocable.net
14	shawcable.net	4	uswest.net	2	knology.net
14	optonline.net	3	dialsprint.net	2	com.au
14	ne.jp	3	a2000.nl	2	fuse.net
9	chello.nl	3	grics.net	2	lrun.com
9	ntl.com	3	linkline.com	2	dialin.net
8	videotron.ca	3	eticomm.net	2	bellsouth.net
7	ad.jp	3	prestige.net	2	psnw.com
7	psi.net	3	warwick.net	2	pacificnet.net
6	uk.com	3	supernet.com	2	tds.net

Domains hosting two or more security-compromised, attack Zombie, Windows PC's

(The balance of the 474 Windows PC's not represented above were scattered across a multitude of domains, one machine apiece.)

It probably comes as no surprise that the top two U.S. residential cable-modem Internet service providers — **@Home** and **Road Runner** — provide Internet connectivity to the host machines most often sought by malicious hackers for the installation of bandwidth flooding Zombie attack Trojans.

While I was monitoring several online hacker hangouts (with the aid of custom spy-bots I created for the purpose — more on that below), I often overheard hackers referring to various lists of "cable Bots" and saying things like "Heh, but how many of his Bots are cable?"

It is clear that the "cable Bot" — a remote control Zombie program installed on a high bandwidth, usually on, Windows machine — has become a highly sought-after resource among malicious "Zombie/Bot running" Internet hackers.

Five Additional Attacks

After the first May 4th attack, we were left alone for eight days. As far as I knew then, the attack had been a one time event like so many of the denial of service attacks you hear about. But then, shortly before midnight Saturday May 12th, we were hit with a repeat of the previous attack lasting eight hours into May 13th.

Here is a summary of the attacks and our actions:

- **May 4th — First attack**, 17 hours before we were able to filter it at our ISP. The attack continued unabated behind the ISP's filters.

- **May 13th — Second attack**, identical to the first and using the same attacking Windows machines. Our communication with Verio was still problematical (at best), so we were knocked off the Internet for eight hours until Verio re-established our previous filters. Unfortunately, I was beginning to learn that Verio's advertised and promised 24/7 service was really only 8/5.

- **May 14th — Attacks 3a and 3b**. Since Verio was still blocking malicious traffic to the grc.com server, this third attack was retargeted at the IP of our firewall. This was one machine closer to the Internet and still on our side of our T1 trunks. So the malicious traffic was again crossing our T1's and we were promptly knocked off the Internet.

  By this time I had the home phone numbers for key members of Verio's engineering staff (<grin>) so we quickly changed the router's filter to block the firewall's IP address and grc.com popped back onto the Internet.

  However, the hackers were apparently watching our success at blocking their attacks. So two hours later the attack resumed, this time aimed at one of the two T1 interfaces of our Cisco router. This again knocked us off the Internet because malicious traffic was again crossing our bandwidth-limited T1's to our local router. Back on the phone to Verio, we decided to completely shut down that T1. GRC.COM limped back onto the Internet running on a single T1.

- **May 15th — Attack 4**. This one started earlier in the evening (at 5:00 PM PST) and knocked us off the Internet for six and a half hours (until 11:30 PM PST) before fading of its own accord. Due to bugs in Cisco's v12.0 IOS routing software (which we fully characterized and worked around several days later) we were unable to comprehensively filter this attack. Rather than engaging in another night of cat & mouse "guess the IP" as we had the night before, I decided to remain off the Internet, collect attack logging data, and take the opportunity to defragment our server's hard drives while weathering the storm.

As you might imagine, these attacks had gone from initially being interesting — from an Internet Security standpoint — to becoming a significant distraction and annoyance. So, on Wednesday May 16th, I got my local (really good) Verio engineer on the phone and we went to work methodically designing and testing a comprehensive set of Cisco router filters so that Verio's router could protect the entire grc.com domain including its T1 interfaces.

By this time, I had assembled an exact profile of the malicious traffic being generated during these attacks. Thanks to the fact that they were sourced from security-compromised Windows machines, they were subject to filtering. During this process we discovered, confirmed, and worked around the several significant packet filtering bugs in Cisco's v12.0 IOS which had caused our previous headaches.

Finally, ducking down behind these new filters, we held our breath and headed into another evening of malicious Internet attacks **. . .**

● **May 16th — Attack 5**. Being as prepared as we now were paid off completely. The grc.com domain never felt the massive attack being waged on the other side of Verio's router. The next day I asked my local Verio engineer how many "UDP/666" malicious packets had been stopped by our filters:

<div align="center">

12,248,097

</div>

The attacking Windows machines generate maximum-size 64k byte UDP packets, but only the first 1500 byte "fragment" of each packet carries the packet's port "666" destination. Therefore, for every identified "666" packet blocked, approximately 43 additional maximum-size "packet fragments" were also blocked. We therefore estimate that our filters running in Verio's router blocked at least **538,916,268** malicious packets that night.

● **May 17/18/19/20th — Attack 6**. The exact dates and times are unknown because we were completely shielded by the configuration of Verio's router. But when we checked the router's "UDP/666" hit counter on the morning of Monday May 21st we found that the blocked "666" count had jumped from its previous value of 12,248,097 to a whopping 54,528,114 packets, leading us to conclude that the filters had weathered, by this time, at least :

<div align="center">

2,399,237,016 malicious packets.

</div>

— **nearly 2.4 BILLION malicious packets.**

<div align="center">

If the attacking machines had been running Windows 2000 or the home-targeted version of Windows XP, as they certainly will be next year, we would have been utterly defenseless and simply forced off the Internet.

This is what anyone on the Internet can soon expect.

(See my WinXP & DoS Page for more information about this.)

</div>

"Wicked" Speaks

On May 15th, after weathering our fourth DDoS attack, the following newsgroup posting, claiming responsibility and credit for the multiple attacks against GRC.COM, appeared on our news server:

> hi, its me, wicked, im the one nailing the server with udp and icmp packets, nice sisco router, btw im 13, its a new addition, nothin tracert cant handle, and ur on a t3.....so up ur connection foo, we will just keep comin at you, u cant stop us "script kiddies" because we are better than you, plain and simple.

Welcome to the brave new world of the 13 year-old Internet terrorist.

I immediately and politely replied to "Wicked" via the newsgroup. I invited him to write to me via eMail through any anonymous channel of his choosing. I wanted to understand what had precipitated these multiple attacks, and I wanted to see what I could do to get them to stop.

"Wicked" soon wrote to me through an eMail account at YAHOO.COM:

> yo, u might not thing of this as anyomous, but its not real info, its a stolen earthlink, so its good, now, to speak of the implemented attacks, yeah its me, and the reason me and my 2 other contributers do this is because in a previous post you call us "script kiddies", atleast so i was told, so, i teamed up with them and i knock the hell out of your cicso router, and....im building up more bots, no, not sub seven lame ass script trojans, i made my own, and it seems quite effective does it not? seems to me that ur backbone has trouble handling the crap sent at it, go ahead and drop icmp pings, u still need to say "NO" to them so it still takes bandwith, thats where tracert comes in, to find the t3 box ur on, nice, i see u stop it as-of today, :) good for you, now ill find ways around it and we can keep playing these games, i find it very fun, shout out to hellfirez and drgreen, and yeah the hellfirez from subseven, hes a friend and he isnt a script kiddie u stupid fuck...now, if u wish to talk to me in person, hows irc??? talk to WkD, the nick wicked was taken, good luck :)

My reply to this note from "Wicked" carefully explained that he really had me all wrong. I pointed to my "Acknowledgement of Debt to the World's Hackers" at the bottom of my "NanoProbe" page. I explained that while I did feel there was a distinction between an elite hacker and a script kiddie, I was someone who always took pains to be respectful of others' egos (when possible), and that I was unlikely — unless provoked — to casually refer to anyone using a derogatory term. I told him that while I was aware of a dispute that had erupted several weeks before in one of our newsgroups, and reportedly involved his friends "HeLLfiReZ" and "DrGreen", I had neither read nor participated in any of that conflict.

"Wicked" replied that perhaps he had "misjudged me", since he was only going by word of mouth. He volunteered to speak with his friends and call off the attacks. He promised that there would be no further attacks from then on **. . .** after which he attacked us on the evening of May 16th, saying afterwards:

```
is there another way i can reach you that is
secure, (i just ddosed you, i aint stupid, im
betting first chance ud tracert me and call fbi)
you seem like an interesting person to talk to
```

"i just ddosed you ..."   Indeed.

Fortunately, that was the first night of our new and (so far) impregnable router filters, so we felt nothing across our T1's while Verio's router counted and discarded nearly five hundred and thirty-nine million (538,916,268) malicious bandwidth-consuming attack packets.

From my dialog with "Wicked", I saw that these repeated attacks were "fun" for him. He was like a child pulling the legs off a spider to see what it would do, watching it flail and attempt to get away from its tormentor. And, as we have seen, he experiences absolutely no remorse and has no regard for any damage being done as a consequence. He believes that he can not and will not be caught. Hiding behind the anonymity created by the Internet's trusting technology, he exhibits no social conscience.

I hope it is becoming clear to everyone reading this, that we can not have a stable Internet economy while 13 year-old children are free to deny arbitrary Internet services with impunity.

I wanted these attacks to stop, but I was certainly in no position to make any "parental" demands of "Wicked". While we were essentially functional, hiding behind our router filters, we could not remain behind them forever. We were unable to send and receive "ping", "trace route" and UDP fragments — all crucial requirements for full Internet function. In the long term this would pose serious problems for the delivery of GRC's Internet security testing services.

I had to find a better solution.

Earthlink Turns a Blind Eye.

I wasn't yet sure what I wanted to do about this 13 year-old problem, but I felt that I ought to broaden my options. "Wicked's" several postings to our newsgroups, and his eMail to me through his yahoo.com account, were all originated from the same small IP address range corresponding to the small ISP Genuity, BBN Planet, in Kenosha, Wisconsin — an Earthlink reseller. This correlated perfectly with "Wicked's" claim to be using a "stolen earthlink".

Since I had collected the exact dates, times, and IPs for each of "Wicked's" several dial-up connections, I felt that perhaps Earthlink could preserve the access and phone records in case the FBI might need them later. If his home phone number could be determined, we could identify him. I knew that Earthlink would never reveal such information to me, but I just wanted them to preserve the evidence against the possibility of future need.

Two months before this, Earthlink's privacy officer, Les Seagraves, and I met and formed a good relationship during our quest to understand the peculiar Earthlink Browser Tag. Unfortunately, Les' voicemail explained that he would be out of town through the end of the month. So I got the name of Earthlink's director of corporate communications, Kurt Rahn, from a well-placed press contact of mine. Kurt was prompt with eMail, and he made lots of motivated-sounding noises, but nothing more ever happened. After waiting hopefully for several days, I finally spoke to Kurt on the phone and allowed myself to sound a bit perturbed. It had no discernible effect. His many promises to have Earthlink's security people get in touch with me never resulted in a single contact from anyone.

Hackers take note: Earthlink appears to be a safe haven for your operations. From everything I have seen, Earthlink couldn't care less WHAT you do, so long as someone pays the bill.

Further note: The day after I wrote this, Les Seagraves returned from his trip and immediately returned my original voicemail message. Les was sincerely apologetic and wonderful when I explained the situation. So I feel self-conscious over being as harsh about Earthlink's response as I have been here. But what I wrote is exactly what happened, and I don't know how else we will ever get ISP's to spend some money, and get involved in security issues, unless we begin holding them accountable for their inaction.

The lights may be on, but nobody's @Home

I have recorded the IPs and account numbers of more than 100 @home subscribers who have security-compromised Windows machines currently running active Trojan attack Zombies. As we will see below, each of those machines also receives a complimentary copy of the latest version (v2.21) of the incredibly invasive Sub7Server Trojan. This grants the hacker who is controlling the Zombie — the "Zombie-master" — absolute control over his victims' machines. Among the many invasions the Sub7Server Trojan enables is monitoring every keystroke for the

purpose of capturing online passwords, credit card numbers, eBanking passwords and you name it.

Now, you might think that this would be significant to @home's chief of security, Todd Welch, but it isn't. I tried to talk to him on the phone, leaving a detailed voicemail describing the situation, but I was shuffled off into the system and asked to eMail the IP's to "abuse@home.com". Refusing to have the machine IP's disappear and never to know what, if anything, had been done, I called back the next day and got Todd on the phone. I have no idea why, but he didn't sound at all happy to be talking with me. It was as if he wished this problem would just go away — or that at least, I would.

I explained that many of the compromised and Zombie-infected @home machines were showing a machine name of ***.sfba.home.com**, which I presumed, and he reluctantly confirmed, stood for "San Francisco Bay Area". Since @home is in Redwood City on the Bay Area Peninsula, I thought that perhaps I could fly up to their offices, then he and I could make a few house calls on some Bay Area Zombie-infected @home subscribers.

> **I was itching to get my hands on one of those nasty nightmares that had been plaguing us for the previous two weeks so that I could take it apart and figure out what made it tick.**

I told Todd that after I had dissected a Zombie, I might be able to come up with a way for @home to scan their network to find all of them. It turns out that **I have found a way**, but again, Todd and @home couldn't be bothered. He declined all cooperation of any sort, curtly adding that they work with the FBI, and no one else. As we will see next, this is a policy in dire need of change. Nice as it sounds on the surface, the realities of Federal government involvement mean that most of the time Todd and @home **. . .** do nothing.

Okay, time to call the Feds . . .

My Conversations with the FBI

In stark contrast to my frustrating conversations with commercial Internet service providers, **I had two terrific dialogs with the FBI's top cybercrime agents.** One of the agents, based in Hayward, California, oversees much of the Western region and is a frequent visitor to grc.com. (With tongue in cheek, he proudly informed me that his "Shields were UP!") The other agent, based right here in my home city, is one of those working on the case of the Russian hacker extortion ring, for which I had recently created the PatchWork utility. He said that he carries a copy of PatchWork around with him to quickly demonstrate how vulnerable Windows servers are due to missing security patches. All of this made our introductions much simpler and smoother.

Both FBI guys said similar things:

- They explained that until $5,000 of damage had been done, no crime had even been committed. That's the law. And due to the peculiar nature of GRC.COM's business model (such as it is :), these attacks were stirring up interest in my forthcoming research and it wasn't even clear that we were going to be economically damaged in any way.

- Secondly, they said that even if we did manage to meet the $5,000 minimum required for "Wicked's" activities to qualify as criminal, their staffs were overloaded and swamped with cases involving companies that had lost huge sums of money to Internet crime. Furthermore, since the cost of an FBI prosecution was in the neighborhood of $200,000, they needed to prioritize their cases based upon prosecuting criminals who were responsible for causing large dollar losses. "Wicked's" attacks, no matter how annoying, failed to qualify.

- And finally, they said that since "Wicked" was only 13 years old, nothing much would happen to him, even if the preponderance of evidence demonstrated that he was behind these attacks. They said that a couple of agents might go out to his home and have a talk with his parents, but in this country his youth was an impenetrable shield. This, of course, further discouraged the costs which would be incurred through any investigation.

Contrary to what you might presume, I did not regard any of this as particularly bad news. I felt that I should do what I could do in the legal arena, because I should. But I really didn't have any desire to be responsible for putting a 13 year-old behind bars. I have since told "Wicked" that if he doesn't wise up, in five years his "youthful offender shield" is going to dissolve and he could find himself in some serious trouble. He says that he was already in trouble with the FBI when he was eight — for hacking government servers. His computer was taken away until he was ten, then he was carefully monitored for another year until he was eleven. But now he's right back at it. <sigh>

The local FBI agent did ask for the IP's and domain names of the 474 machines that participated in the May 4th attack. I forwarded the information to him immediately, so now they have that stuff.

But I was still no closer to having any REAL answers . . .

Getting My Hands on a Zombie

An angry phone call from the head of Information Services at Texas A&M University gave me an idea. In some initial media coverage after our first attack, I had mentioned that a single machine at Texas A&M University (tamu.edu) had been responsible for generating about twenty times more malicious traffic than any other. The I.S. guy was pissed off that I hadn't contacted them immediately, and I suddenly felt like a total fool: I could have contacted him and we could have probably tracked down the infected machine. I immediately volunteered to fly to Texas and begged him not to destroy any evidence until I got there. But it turned out that the machine was within the student residential network (resnet.tamu.edu) — the student dormitories — and that since the May 4th attack, the University had closed down for late spring break. Drat!

But that gave me an idea: Although the big ISP's are apparently so big that they no longer need to care about their customers, the smaller users — like TAMU — might be much more receptive. So I sent out a mass of eMail to every smaller ISP and domain

administrator of the infected attacking machines. I explained the situation and asking for their help.

Someone discovered the Zombie and sent it to us!

I will never know whom to thank because they dropped the Zombie into our anonymous web-based Spyware drop-box. But it was all I needed to learn how these Zombie's operate and then infiltrate the Zombie High-Command . . .

The Anatomy of a Windows Attack Zombie

The Zombie program I received was named "**rundIl.exe**". (Note the capital "I" in the filename.) This struck me as significant since "rundll.exe" is a frequently used and often seen component of Windows systems. Changing the first lower-case 'l' (el) to an upper-case 'I' completely hides the difference under Windows 9x systems because the font used by the Windows registry renders those two characters as a featureless vertical bar. Anyone inspecting the Windows registry for suspicious files will see: "rundll.exe" and miss the fact that it's actually "rundIl.exe". Clever.

My inspection of the 15,904 byte Zombie program quickly revealed it to be an IRC (Internet Relay Chat) client. So I decided to sacrifice a PC to the Zombie by deliberately infecting it while keeping it under observation with a packet sniffer running on an adjacent machine. I freshly reformatted a laptop, installed a completely clean copy of Microsoft Windows, named the machine "Sitting Duck" . . . and turned it on.

The Zombie immediately connected with a remote, pre-programmed, IRC chat server. It then joined a secret and password key-protected channel on that server . . . and waited for instructions.

It didn't have long to wait.

I watched in fascination as many other Zombies — hundreds of others — arrived and departed the secret "Zombie meeting grounds" of the IRC server.

Somewhere, Windows users were innocently turning on their PC's. Lacking any effective personal firewall security (we will see later that BlackICE Defender provides no protection), the Zombies running secretly and silently inside those machines were connecting to this IRC server. They maintained persistent connections for the duration of that PC's access to the Internet. The Zombie and its master don't care whether the machine is cable-connected, DSL, or dial-up — though higher-speed connections are always preferred, as are machines that tend to be "on" most of the time. After all, you just never know when you're going to need to go attack someone.

While I was watching this sad drama, **suddenly and with no warning everything went crazy:** The packet sniffer's packet display became a blur as its scrollbar "thumb" rapidly shrunk to its minimum size. Thousands of packets were being logged per

second! Since I was nervous during this first incursion into hacker territory, my first thought was that I had somehow already been discovered, and my little "Sitting Duck" laptop was under attack.

But the cable-modem I was using to guarantee my anonymity revealed the truth: The RECEIVE light was dark, but the **TRANSMIT light was ON SOLID!**

I immediately shut down the Zombie-infected PC and scrolled the packet log back before the beginning of the attack. I found the command that the Zombie running in my laptop had received just before all hell broke loose **. . .**

My laptop had participated in an all-out Denial of Service attack against a machine in Finland!

Yikes! This was unacceptable. I wanted to keep active Zombies running here so that I could study their behavior, but I could not have them participating in Internet Denial of Service attacks. So I hacked the Zombie to kill its ability to send damaging packets.

From that point on I ran only "Attack-Neutered Mutant Zombies"

Later that night I received another surprise . . .

The Arrival of the Sub7Server Trojan

Not content to simply have a fleet of willing attack Zombies, "Zombie-masters" routinely download the latest version of the rapidly evolving Sub7Server Trojan into all of their "client" machines. Since Zombies are continually joining and leaving the central IRC server, Sub7 downloads are generally performed on the pool of currently connected machines several times per day. (After all, you certainly wouldn't want an old version of the Sub7 Trojan running in any of your Zombied machines.)

The Sub7Server Trojan is massively invasive. It has been designed to give its master virtually complete control over the compromised PC. This includes complete file system inventorying and file access, and real-time keyboard keystroke logging. Any user with Sub7 in their machine might as well have the hacker standing right next to them watching every move they make while using the computer.

Although it was not the focus of my research, examining the latest version of Sub7 was an education in itself. Sub7 is downloaded by the Zombie from a "free web pages" web server as a single file named "HardcoreS7.exe". When this executable file is run it breaks apart into two randomly-named files so that it is not possible to search by filename. The original "HardcoreS7.exe" file is then deleted. Sub7 insinuates itself into Windows in a few clever ways. It installs in the seldom used "run=" line of the deprecated WIN.INI file. It also installs under the "Run" key of the registry, and it inserts a much smaller 10k "runner" into the Windows Shell "command/open" key. All of this pretty much guarantees that Sub7 will keep running inside the system. It's difficult to shake it loose.

When Sub7 awakens inside a machine it begins listening on the random port it chose

during its installation. This means that security scanners can no longer scan for "the Sub7 port" since there isn't one. After setting itself up for business, Sub7 sends out notifications to hackers of its availability through two different channels: It joins a special Sub7 IRC chat server where it posts a notice listing all the details required for its connection and use:

Sub7Server v.BoNuS 2.1
installed on port: 27374,
ip: 293.492.59.748,
victim: Frog's Legs,
password: ribbit

At the same time, identical information is posted to a newsgroup server through a web server CGI script. Hackers no longer need to "scan" the Internet for vulnerable machines running Sub7 **. . .** each copy of Sub7 phones home providing complete connection details.

I Needed My Own Stealth Spy-Bots

Untangling and extracting the meaning from the packet capture dialog of the average attack-neutered mutant Zombie is never fun. Moreover, separating the good stuff from the noise just adds to the burden. So I soon realized that I needed to create my own "Zombie simulators" that would logon to IRC chat channels, hob-nob with the Zombie locals and their masters, while logging everything that transpired and automatically alerting me of anything important.

So I downloaded a copy of the Internet RFC 1459 for Internet Relay Chat (IRC) Protocol and figured out how IRC works.

By the time I was done, I had written a handful of application-specific tools for infiltrating and spying on hacker and Sub7 Trojan IRC channels. The tools chronicled dialogs, and captured references to other server, hacker and Zombie channels. They automatically spawned new instances to begin monitoring newly discovered servers. They snagged passing URLs and quickly downloaded anything that was referenced. I even got quite fancy and built a Markov-chain finite-state statistical dialog modeller. It monitored the flow of IRC channel nicknames and automated the process of determining who was talking to whom, and who were the "bosses" who commanded the most power and respect.

I learned an amazing amount about this bizarre world of zombie-running hackers. During the process I witnessed a truly disturbing number of nightly Internet attacks. At this point I had two goals:

I wanted to thoroughly understand this technology for its own sake. After all, I am fascinated by the issues surrounding Internet security and privacy. I also wanted to understand how everything worked so that I could, if possible, defend against any future similar attacks. (For example, by infiltrating, commandeering, and neutralizing any attacking force of Zombies.)

I also still needed to "get through to" Wicked somehow. I needed to convince him to

lay off his repeated attacks.

My IRC Chat with the ^b0ss^

I had learned a great deal about the Zombies, and I knew that "Wicked" had not created his own as he had claimed. By analyzing the binaries of all the various Zombies my spy-bots had collected, I could pretty much follow the evolutionary "lineage" of this strain of Zombie. I finally found the hacker ("^b0ss^") whose Zombies "Wicked" had "hex edited" in order to create those that had been attacking grc.com.

One afternoon, one of my spy-bots intercepted a conversation taking place between that hacker ("^b0ss^") and another nicknamed "lithium_". Their dialog revealed that "^b0ss^" was creating a new Zombie for "lithium_", editing it to report to a different secret IRC channel using a different password. Unaware that they were under surveillance, they spoke openly of their plans. I didn't discover that interchange until later that evening, but my URL interceptor and downloader had automatically snagged a copy of the new Zombie (this time named "win.exe") and had downloaded it into my Zombie-repository for safe keeping.

Peeking into this new Zombie's now-quite-familiar guts, I immediately noticed something odd: "^b0ss^" had apparently made a small mistake with his Zombie hex editing. He had separated the new strings for the channel and the password key with a period (.) rather than a null (0). This Zombie would not hunt.

I saw an opportunity to help.

Now, don't get me wrong here:
I'm no Zombie lover.

I have NO desire to aid and abet the hackers in their pursuit of unlawful Internet terrorism. "^b0ss^" and "lithium_" would have soon figured out that something was wrong with the new Zombie — and fixed their simple mistake. But I felt that my pointing it out, bringing an olive branch with me into "Zombie-land", would be the perfect means of establishing a constructive dialog with "^b0ss^ (who was likely to be at least somewhat freaked out when I just popped into his private inner-Zombie-sanctum.) So, I ask you to PLEASE KEEP IN MIND that I had a deliberate need and intent underlying the following dialog which took place with "^b0ss^".

When he left one of the main Sub7Server areas and appeared over in the room where he tends his Zombies, I had all of the secret channel names and pass-keys at my disposal. So I just jumped into the room and said "heh". While I was using a Windows IRC chat client, one of my Spy-Bots was automatically recording and logging our dialog from that "long under surveillance" Zombie haven:

```
<Gibson> heh
<^b0ss^> who are you
<Gibson> Hi B0ss.  I'm steve gibson (grc.com) ... ShieldsUP,
<Gibson> OptOut, Leaktest ... and all that stuff.
<^b0ss^> how did you get in here?
<^b0ss^> your not a IRCop
<Gibson> As you might know, my site was attacked (but I don't
<Gibson> think by your bots) a few weeks ago.
<Gibson> Some guy, calling himself "Wicked"
<^b0ss^> my bots?
<^b0ss^> no no
<^b0ss^> I know wicked
<^b0ss^> it was not my bots I promise
<^b0ss^> Wicked has his own
<Gibson> Hey, it's okay
<^b0ss^> alot of bots
<^b0ss^> heh
<Gibson> I know.
<^b0ss^> yeah
<^b0ss^> I promise it wasn't mine
<Gibson> I wanted to let you know that the bot
<Gibson> you made earlier for Lithium would not work
<^b0ss^> what about the bot?
<^b0ss^> you know Lithium
<Gibson> since it has "periods" (2E) instead of NULLS (0)
<^b0ss^> ?
<Gibson> separating the "Channel" and "Key" strings
<^b0ss^> you his friend
<Gibson> no.
<Gibson> I wanted to learn about this stuff
<Gibson> since Wicked was attacking me
<^b0ss^> then how did you know about this place
<^b0ss^> you have your own server?
<Gibson> So I wrote some fake bots to monitor various Bot
<Gibson> networks so that I could learn.
<^b0ss^> damnit
<^b0ss^> so you been spying?
<^b0ss^> hehe
<Gibson> Yeah
<Gibson> But not to worry, I'm no narc.
<Gibson> I just want to be left alone.
<^b0ss^> but how did you get the Key
<^b0ss^> I don't even know you
<^b0ss^> I don't bother anyone with my bots
<Gibson> Check out GRC.COM.  That's me.
<^b0ss^> okay
<^b0ss^> you don't like wicked?
<Gibson> Well ...... I can't say that I know him,
```

```
<Gibson> but he spent a few weeks blasting my site
<^b0ss^> damn
<Gibson> since he thought (he sez that Hellfirez and DrGreen told
<Gibson> him) that I was referring to them as "script kiddies" ...
<^b0ss^> hehe, I got enough bots to blast away a site
<^b0ss^> but I don't use them for that
<^b0ss^> lol
<Gibson> (You have 241 Bots.)
<^b0ss^> thats not it
<^b0ss^> not just on this server
<^b0ss^> how in the hell do you know how many bots I have?
<^b0ss^> damn
<Gibson> I've tracked 241 coming and going over the past four days.
<^b0ss^> let me get some of your bots
<^b0ss^> lol
<^b0ss^> I can't believe this shit, what kind of bot you have
<Gibson> Do you know where Wicked got his?  He claims that he
<Gibson> wrote it, but it looks like a pure hex-edit to me.
<^b0ss^> oh no
<^b0ss^> lol
<^b0ss^> he didn't make them
<^b0ss^> he got his bot from these bots in this room
<Gibson> You really ought to check out my site.  grc.com
<^b0ss^> I am right now
<^b0ss^> ;)
<^b0ss^> nice page
<Gibson> Yeah, I believe that about Wicked.
<Gibson> His channel is #pines1 and PassKey is "penile"
<Gibson> (pines1 is "penis1" with the vowels swapped).
<^b0ss^> lol
<^b0ss^> damn
<^b0ss^> you are pretty good
<Gibson> Anyway, last week I learned IRC protocol and wrote a
<Gibson> bunch of infiltration bots in order to figure out where
<Gibson> all these attacks were coming from.
<^b0ss^> hmmm
<Gibson> It looks like he's lost his dynDNS
<^b0ss^> you know what serve he keeps them all on
<^b0ss^> yup
<Gibson> yeah, I have his server, but I think he's off the air
<Gibson> for now and won't be bothering me again any time soon.
<^b0ss^> we had alot of bots on ips.mine.nu
<^b0ss^> but they took it down
<^b0ss^> for illegal use
<Gibson> cool! I was hoping that might be it.
<^b0ss^> oh, I wouldn't say that
<^b0ss^> he is gettin army back
<^b0ss^> heh
<^b0ss^> I know he has more
```

```
<^b0ss^> somewhere
<Gibson> I don't care if he wants to blast IRC folks,
<Gibson> but I haven't done anything to bother him.
<Gibson> If he blasts me again I'll take them away.
<^b0ss^> lol, he is 13
<^b0ss^> did you know that
<Gibson> Yeah, he said, and he writes like he is.  But I didn't
<Gibson> think he could really write that Bot from scratch.
<^b0ss^> which bot you talkin about
<^b0ss^> do you know mimic?
<Gibson> You call yours "evilbot" (version 0.4c) ...
<Gibson> he renamed it "WkD Bot" (version 1.0)
<^b0ss^> yeah
<Gibson> I don't know anyone. YOU are the first person I've
<Gibson> ever talked to on IRC. Wicked and I have eMailed.
<^b0ss^> mimic has a hell of a bot
<^b0ss^> so, you set up a bot in this channel spying?
<Gibson> Yep .... about a week ago.  I have a list of all
<Gibson> the attacks you've made, etc. etc.
<^b0ss^> shit
<Gibson> The one on a machine within IBM freaked me out.
<^b0ss^> so how did you get the key to my channel to get the bot in
<^b0ss^> IRCop?
<Gibson> Like I said, I just needed to learn about this
<Gibson> stuff so that I could defend myself.
<^b0ss^> man, I wouldn't attack you I promise you that
<^b0ss^> I have no reason
<Gibson> I asked all of the ISP's of the people whose
<Gibson> machines were attacking me for a Bot.
<^b0ss^> oh
<Gibson> Someone sent me one ... and from there I knew what I needed.
<^b0ss^> hehe
<Gibson> Then I wrote a custom "spy bot" and started monitoring
<Gibson> more and more conversations, following leads, URL's, etc.
<^b0ss^> hmmm
<Gibson> that's how I know about you making the new custom bot
<Gibson> for lithium this afternoon.
<^b0ss^> damnit
<Gibson> but when I finally looked at it I saw that it wouldn't work,
<Gibson> so I figured I'd introduce myself and let you know.  :)
<^b0ss^> lol
<Gibson> And of course the Bot itself knows how to logon here! <g>
<^b0ss^> yeah
<^b0ss^> good job
<^b0ss^> I must say
<Gibson> Well, it was nice to meet you.
<^b0ss^> nice to meet you to
<^b0ss^> You are pretty good
<Gibson> And, again, that Bot you made for lith earlier won't work.
```

```
<Gibson> so make sure he doesn't deploy it until you fix it for him.
<^b0ss^> may I ask how old you are?
<Gibson> I'm 46. (Been hacking since I was 14.)
<^b0ss^> lol, alright, thanx
<^b0ss^> damn
<^b0ss^> you are good
<^b0ss^> you gonna leave your bot in here?
<Gibson> Nope.  It's done it's job.  I'm working on a new web page
<Gibson> to talk about the Wicked attacks, and to explain this whole
<Gibson> bizarre world.
<^b0ss^> alright thanx
<^b0ss^> hehe, yeah
<Gibson> Check back at grc.com in a few daze
<^b0ss^> okay
<^b0ss^> I will
<Gibson> If you see Wicked, tell him we had a nice chat
<Gibson> and ask him to lay off.  I don't want to upset him,
<^b0ss^> okay
<Gibson> but I need to, and will, defend my site.
<Gibson> Thanks!
<Gibson> .
<^b0ss^> hehe, okay
<^b0ss^> welcome
```

As we now know,
A 13 year-old hacker . . .

. . . (living in Kenosha, Wisconsin) who goes by the hacker handle "Wicked", was informed by some senior hackers — among them "HeLLfiReZ" a member of the notorious Sub7 crew — that I had referred to them in an online forum, using the derogatory term "script kiddies". I had not. But these senior hackers were upset over a dispute that had erupted in one of our Internet security newsgroups.

"Wicked's" response was to team up with two other hackers, all of whom tend and manage large fleets of "IRC Attack Bots". They launched a concerted and extended "packet attack" against grc.com. In the slang that I learned while monitoring their many conversations, they "packeted" us. They did this, not using any tool they had written, and not possessing the ability to create such a tool themselves, but using a powerful "IRC Bot" that had been passed around extensively. Neither Wicked nor his friends know who wrote it or even where it came from.

The "Wicked" Attacks

"Wicked" and his IRC Bots communicate by logging onto an IRC server located at the domain "wkdbots.***.**" (I have blanked the upper portion of the domain to allow me to provide all other details.) This domain name is hosted by a dynamic DNS service, allowing Wicked to change the location of the IRC server, as needed, by pointing the "wkdbots" domain at a different IP address. This highlights one of the

several weaknesses of the IRC Bots system: A single discovered Bot reveals the IRC meeting place of the entire Bot fleet. The subsequent loss of access to their shared domain cripples the Bot network by denying its access to its central communications hub.

After "Wicked's" Bots connect to the "wkdbots" server, they join and monitor the secret "#pines1" IRC channel using the password key "penile". (You will note that "pines" is the word "penis" with the vowels transposed.) The IRC Bot which Wicked hex-edited to create his own uses a more straightforward channel and password key.

On the evening of May 4th, 2001, after logging onto the wkdbots IRC server and joining his Bots on the #pines1/penile channel, "Wicked" typed the following two commands which were immediately relayed, courtesy of their shared IRC chat server, to his awaiting fleet of IRC attack Bots:

```
!p4 207.71.92.193
!udp 207.71.92.193 9999999 0
```

Within moments, 474 security-compromised Microsoft Windows PC's, containing remote control attack "Zombies", were attacking the grc.com server.

A small sampling of the "Bot's" command language . . .

● **The leading exclamation point "!"** prefixes all of the commands accepted by the IRC Bot.

● **The !p4 command,** followed by the target IP address (in this case the IP of the grc.com server) executes the following standard Windows "ping" command with the following arguments:

```
ping.exe 207.71.92.193 -l 65500 -n 10000
```

This causes the Windows machine to send ten thousand very large (64 kbyte) "ping" packets to the machine at the specified IP. A bit of math shows that this is 655 megabytes of data. This doesn't generate a high-speed stream because the "ping" command waits for a reply before trying again. But if many machines are all pinging at once, the result is cumulative and can be significant. Since the ping command is being executed by a separate (ping.exe) program, the Bot is then free to wreak more havoc . . .

● **The !udp command,** does far more damage. It specifies the target IP, the number of huge UDP packets to send, and the inter-packet delay (zero in this case). The receipt of the !udp command shown above causes each Bot to send 9,999,999 maximum size UDP packets to the grc.com server as fast as the host machine's outbound bandwidth will allow.

In response to receiving the !udp command, each of the hundreds of attacking

machines replied:

```
PRIVMSG #pines1 :BoMbInG: 207.71.92.193, PaCkEtS: 9999999, DeLaY: 0
```

`...` and then began "bombing" grc.com with a blizzard of fragmented UDP and ICMP packets, thus consuming our bandwidth several times over and denying our services to the Internet.

A modest number of cable-modem connected home PC's, pumping out maximum-size UDP packets at their maximum upstream bandwidth, were easily able to flood and completely consume grc.com's bandwidth.

This was no "finesse" attack. There was nothing clever about it. The 13 year-old perpetrator had not created the attack tool, and never could. He barely even knew how it operated. But like someone who is handed a loaded gun, even lacking any understanding of how that gun operates, "Wicked" was able to pull the trigger.

GRC.COM was knocked off the Internet for 17 hours by a classic Distributed Denial of Service (DDoS) attack.

Before I leave the topic of Zombie/Bot commands, I thought you might find the **"!r"** command interesting. **"R"** must be short for "Ready" or "Report" or "Rally" because it immediately causes all currently connected and listening Zombie's to "report in". Here is a snippet that one of my spy-bots picked up when "^b0ss^" gave the "!r" command to his troops:

```
<^b0ss^> !r
<xknb> evilbot 0.4c ready for attack...
<oep> evilbot 0.4c ready for attack...
<kvmadj> evilbot 0.4c ready for attack...
<yqvc> evilbot 0.4c ready for attack...
<pvnlz> evilbot 0.4c ready for attack...
<jizl> evilbot 0.4c ready for attack...
<umc> evilbot 0.4c ready for attack...
<wzdr> evilbot 0.4c ready for attack...
<vqfvmh> evilbot 0.4c ready for attack...
<ossqd> evilbot 0.4c ready for attack...
<gyc> evilbot 0.4c ready for attack...
<lvk> evilbot 0.4c ready for attack...
<myv> evilbot 0.4c ready for attack...
<rozjh> evilbot 0.4c ready for attack...
<usxaw> evilbot 0.4c ready for attack...
<vsma> evilbot 0.4c ready for attack...
<xheweq> evilbot 0.4c ready for attack...
<bgpc> evilbot 0.4c ready for attack...
<mntt> evilbot 0.4c ready for attack...
```

```
<nngp> evilbot 0.4c ready for attack...
<mhonm> evilbot 0.4c ready for attack...
<fgjc> evilbot 0.4c ready for attack...
<gyk> evilbot 0.4c ready for attack...
<mkenx> evilbot 0.4c ready for attack...
<pnyd> evilbot 0.4c ready for attack...
<btkh> evilbot 0.4c ready for attack...
<qwbbd> evilbot 0.4c ready for attack...
<vst> evilbot 0.4c ready for attack...
<griv> evilbot 0.4c ready for attack...
<frf> evilbot 0.4c ready for attack...
<fdhcmmk> evilbot 0.4c ready for attack...
<kdu> evilbot 0.4c ready for attack...
<jsea> evilbot 0.4c ready for attack...
<yxkoo> evilbot 0.4c ready for attack...
<xcchvl> evilbot 0.4c ready for attack...
<elzll> evilbot 0.4c ready for attack...
<imebw> evilbot 0.4c ready for attack...
<jddp> evilbot 0.4c ready for attack...
<bpfzvk> evilbot 0.4c ready for attack...
<egja> evilbot 0.4c ready for attack...
<nqab> evilbot 0.4c ready for attack...
<fzdnt> evilbot 0.4c ready for attack...
<eyddzxs> evilbot 0.4c ready for attack...
<dshraf> evilbot 0.4c ready for attack...
<xigvx> evilbot 0.4c ready for attack...
<iaamew> evilbot 0.4c ready for attack...
<joog> evilbot 0.4c ready for attack...
<fhtveo> evilbot 0.4c ready for attack...
<rfktnd> evilbot 0.4c ready for attack...
<ojjntjw> evilbot 0.4c ready for attack...
<esx> evilbot 0.4c ready for attack...
<gxbgi> evilbot 0.4c ready for attack...
<hhenh> evilbot 0.4c ready for attack...
<wknt> evilbot 0.4c ready for attack...
<hwn> evilbot 0.4c ready for attack...
<nnk> evilbot 0.4c ready for attack...
<brmbpb> evilbot 0.4c ready for attack...
```

Those random-character evilbot nicknames, appearing in angle brackets at the front of each line, are the random names generated by each Zombie when it logs on to the IRC server. In the unlikely event of a nickname collision the Bot simply generates another.

The Birth of the IRC Bot (Windows Attack Zombie)

The computer press calls them "Zombies", but they are known as "Bots" or "IRC Bots"

within the hacker community. The particular strain of IRC Bot that attacked us calls itself an "evilbot".

"IRC Bots" are among the newer breed of Distributed Denial of Service (DDoS) agents deployed by the Internet's most active hackers. Whenever an IRC Bot hosting Windows PC is started, the Bot waits for the system to finish booting, then connects to a previously designated IRC server. Using a private password key, it joins a secret IRC channel that is not visible to other users of the IRC server **. . .** and awaits commands.

The use of a central IRC server provides significant anonymity, deployment, and logistical benefits to the hackers, although as we have seen, it is not without liabilities. I was able to successfully penetrate the Bot's world without much trouble.

IRC servers are often configured to deliberately obscure the names and IP addresses of their clients, thus providing anonymity to all users. Since this anonymity can only be breached through physical access to the server, many Bot armies are "run" from servers located on foreign soil where access is impossible to obtain.

Since IRC Bots "phone home" to the central server, the hacker does not need to know which specific machines are hosting his Bots. This allows Bots to be deployed in the wild by a wide variety of means. Hackers create Bot-carrying eMail viruses (frequently enabled by Microsoft's virus-friendly Outlook Express), they create infected Internet "Trojan" downloads, place Bots in USENET newsgroups, and do anything they can to get their Bots into other people's computers.

IRC Bots never need to be "scanned for" since all active Bots contact their home base IRC server whenever they "awaken". The various IRC Bots I have acquired and examined are just 15,904 bytes in size, so they are easily hidden as trojans within other, typically huge, Windows programs.

A Quick & Easy Check for IRC Zombie/Bots

If you have managed to read all the way through this lengthy and detailed adventure, I am sure you will agree that you do NOT want any of these nasty Zombies or their relatives running around loose inside your PC. Fortunately, it's quite easy to verify that your system is not currently infected by one of these IRC Zombie/Bots.

All of the IRC Zombie/Bots open and maintain static connections to remote IRC chat servers whenever the host PC is connected to the Internet. Although it is possible for an IRC chat server to be configured to run on a port other than "6667", every instance I have seen has used the IRC default port of "6667".

Consequently, an active connection to an IRC server can be detected with the following command:

```
netstat -an | find ":6667"
```

Open an MS-DOS Prompt window and type the command line above, then press the "Enter" key. If a line resembling the one shown below is NOT displayed, your computer does not have an open connection to an IRC server running on the standard IRC port. If, however, you see something like this:

```
TCP    192.168.1.101:1026    70.13.215.89:6667   ESTABLISHED
```

... then the only question remaining is how quickly you can disconnect your PC from the Internet!

A second and equally useful test can also be performed. Since IRC servers generally require the presence of an "Ident" server on the client machine, IRC clients almost always include a local "Ident server" to keep the remote IRC server happy. **Every one of the Zombie/Bots I have examined does this.** Therefore, the detection of an Ident server running in your machine would be another good cause for alarm. To quickly check for an Ident server, type the following command at an MS-DOS Prompt:

netstat -an | find ":113 "

As before, a blank line indicates that there is no Ident server running on the default Ident port of "113". (Note the "space" after the 113 and before the closing double-quote.) If, however, you see something like this:

```
TCP       0.0.0.0:113       0.0.0.0:0       LISTENING
```

... then it's probably time to pull the plug on your cable-modem!

> **Note** that a Windows IRC client program running in the PC **will** generate false-positive reports since these are tests for IRC client programs. So be sure to completely exit from any known IRC client programs BEFORE performing the tests above.

Personal Firewalls and IRC Zombie/Bot Intrusions

ZoneAlarm v2.6 (Free) —

The last of my testing was to see whether the firewall I keep telling everyone to use: **ZoneAlarm** — either FREE or Pro — would be effective in stopping the IRC Zombie/Bot and the Sub7 Servers that had taken up residence in my poor "Sitting Duck" laptop.

I downloaded the current, completely free, version of ZoneAlarm 2.6 from the ZoneLabs web site and installed it on the "Sitting Duck" laptop. Upon restarting the machine I was gratified to receive immediate notification that the Zombie/Bot was attempting to make an outbound connection to its IRC chat server.

Meanwhile, the Sub7 Trojan was sitting quietly waiting for someone to connect to it. So I used another machine to "Telnet" to the port the Sub7Server Trojan was listening on. Up popped ZoneAlarm asking whether the nonsense-looking random character name the Sub7Server had chosen for itself should be allowed to accept a connection from the Internet.

Perfect performance from ZoneAlarm.

Then I had a thought: What would Network ICE's BlackICE Defender do under the same circumstances?

🌑 BlackICE Defender v2.5 ($39.95) —

I did not have a current copy of BlackICE Defender around, but I felt that this was an important test. So I laid out $39.95 through Network ICE's connection to the Digital River eCommerce retailer and purchased the latest version (v2.5) of BlackICE Defender hot off the Internet. I had already removed all traces of ZoneAlarm and restarted the machine, so I installed BlackICE Defender, let everything settle down, and restarted the machine with my packet sniffer running on an adjacent PC.

As far as I could tell, BlackICE Defender had **ABSOLUTELY NO EFFECT WHATSOEVER** on the dialogs being held by the Zombies and Trojans running inside the poor "Sitting Duck" laptop. I knew that BlackICE Defender was a lame personal firewall, but this even surprised me.

The Zombie/Bot happily connected without a hitch to its IRC chat server to await further instructions. The Sub7 Trojan sent off its eMail containing the machine's IP and the port where it was listening. Then it connected and logged itself into the Sub7 IRC server, repeating the disclosure of the machine's IP address and awaiting port number. No alerts were raised, nothing was flashing in the system tray. The Trojans were not hampered and I received no indication that anything wrong or dangerous was going on.

I took a lot of grief after my [LeakTest utility]() cut right through BlackICE Defender. Network ICE told everyone that LeakTest was "being allowed through" because it was a completely benign Trojan. I knew that was a load of bull (and they must have too), but it didn't really matter to me, and I had no affirmative means of proving otherwise.

Well . . . I have that now, and so do you.

I performed one final test: As I had with ZoneAlarm, I attempted to connect to the Sub7Server Trojan running inside the "Sitting Duck" machine on the IP and listening port number the Trojan was advertising all over the Internet **. . .** and it worked perfectly. I received Sub7's "PWD" prompt asking me to login.

Anyone want an "only used once"
copy of BlackICE Defender?

I certainly have no use for it.

> **To anyone who is still stubborn enough to insist that BlackICE Defender is actually good for something:** PLEASE do not write to me. I don't want to hear it. I'm a scientist who will not find your mystic beliefs to be compelling. I respect your right to your own opinions, no matter how blatantly they fly in the face of logic and reality. That is, after all, the nature of faith. Happy computing. I suggest prayer.

What's Next?

I have concluded this research into the motivations behind, and the technologies of, the multiple Distributed Denial of Service (DDoS) attacks that were launched against GRC.COM during the month of May, 2001.

● I believe that I have learned everything there is to learn from these IRC Zombie/Bot style attacks. I have found and conversed with the important players, and I have analyzed their tools, technologies, and networks. I could spend the rest of my life pursuing the specifics of all possible exploits and toolz, but that's not where I want to go.

● I have learned that our industry's leading consumer ISP's are worse than useless when asked for any form of help relating to Internet security or the welfare of the paying customers. For reasons unfathomable to me they choose to disavow responsibility for the conduct of their users, and equally refuse to offer any help for their customers' Trojan-infected machines.

● I learned some bitter truths and realities about the nature of Federal government involvement. There are just too many large problems for the smaller ones — which may be destined to grow larger — to receive help or attention. I sincerely hope that the 13 year-old hacker known as "Wicked" figures out that his youthful shield will dissolve in five years. I believe that in the future the United States government, and the world at large, is going to become increasingly intolerant of Internet hacking. The penalties for transgression will be onerous.

● And finally, this experience has reaffirmed my commitment to two separate but closely related needs:

The threat represented by Microsoft's forthcoming Windows XP operating system, with its confirmed ability to easily generate malicious Internet traffic — for NO good reason — can not be overstated. The proper executives within Microsoft MUST be reached with this message so that those plans can be reviewed in light of the potential for their system's massive abuse of the inherently trusting Internet.

> For more information about this serious threat to
> the Internet, please see my **WinXP & DoS page**.

The days of an Internet based upon mutual trust among interconnected networks has passed. The Internet's fundamental infrastructure MUST BE SECURED before the Net becomes further threatened by increasing levels of malicious attacks. Since this requires irresponsible ISP's — who repeatedly demonstrate that they could not care less — to assume the sorts of local responsibility that they have so far deliberately shunned **. . .**

> **We need a tool to hold ISP's accountable and publicly demonstrate individual ISP irresponsibility.**

Given the universal reluctance they have demonstrated so far, I believe that only active public scrutiny will bring about the changes required to insure a reliable and secure future for the Internet.

The development of that tool is my next project.

The name of that FREE tool will be:

Spoofarino ™

(If you would like to be notified when Spoofarino is
ready, you are invited to join our user-managed
eMail system. You can leave any time)

Purchasing Info   **GRC Mail System**   **To GRC's Home**   **Tech Support**   **Discussions**
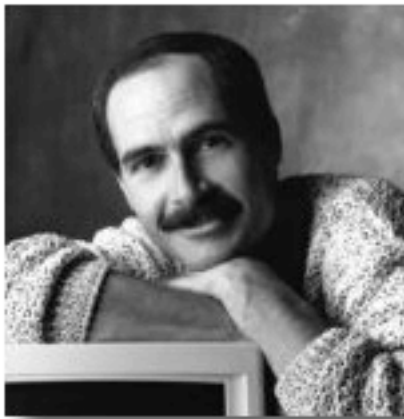
The contents of this page are Copyright (c) 2001 by Gibson Research Corporation.
SpinRite, ChromaZone, ShieldsUP, NanoProbe, the character 'Moe' (shown above),
and the slogan "It's MY Computer" are registered trademarks of Gibson Research
Corporation (GRC), Laguna Hills, CA, USA. GRC's web and customer **privacy policy**.

~ ~ ~

An Open Letter to the Internet's Hackers

Page last modified: Jun 01, 2001 at 12:47

I surrender.

I surrender right now, completely and unconditionally.

And I'm not kidding.

It is my intention to carefully and completely explain, to the entire world, **exactly why there is no defense** against the sorts of clever Internet attacks you guys can create.

I want to do that because the world still doesn't get it.

It occurred to me that you might think that I think I'm invulnerable after managing to block the IRC Zombie/Bot attacks, so I wanted to be SURE that you understood that I AM UNDER NO SUCH DELUSION.

I was talking to a reporter on the phone a few hours ago, during the first REAL, non-blockable attack we have ever experienced. And I calmly explained that we were under attack and off the Net. In a bit of a panic, he asked what I was going to do about it. So I told him that I was going to take a long walk on the beach — because you and I both know there's absolutely NOTHING I CAN DO to defend against a real, professional, Internet Denial of Service attack. So I might as well enjoy the day.

I have started working on a next set of pages to explain all this. The pages are not finished, but you can see what I have so far, and where I'm going with it: (Page 1) (Page 2)

So, I respectfully ask that you leave me alone and allow my site to stay on the Net. I know that you can easily knock me off. That's not even a question. But only if I'm here can I explain that to the rest of the planet.

Thank you for your consideration . . . and for your charity.

Steve

Purchasing Info   **GRC Mail System**   **To GRC's Home**   **Tech Support**   **Discussions**

The Escalating Threat of Internet

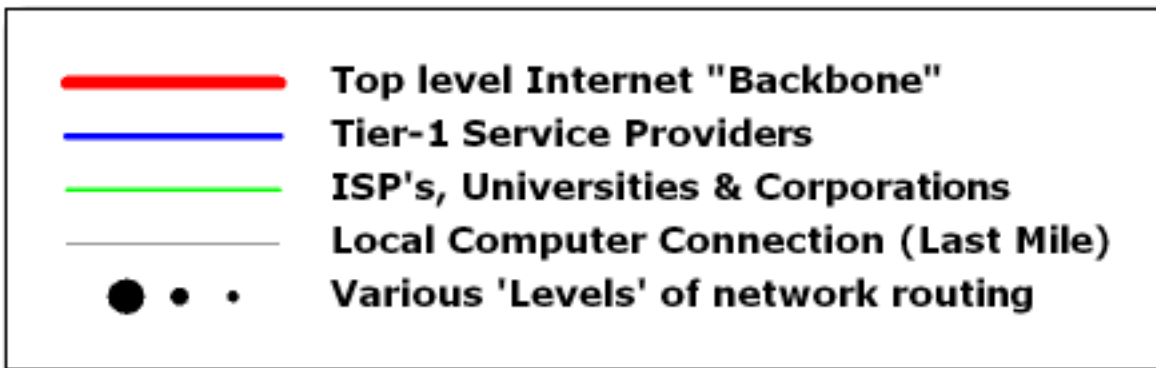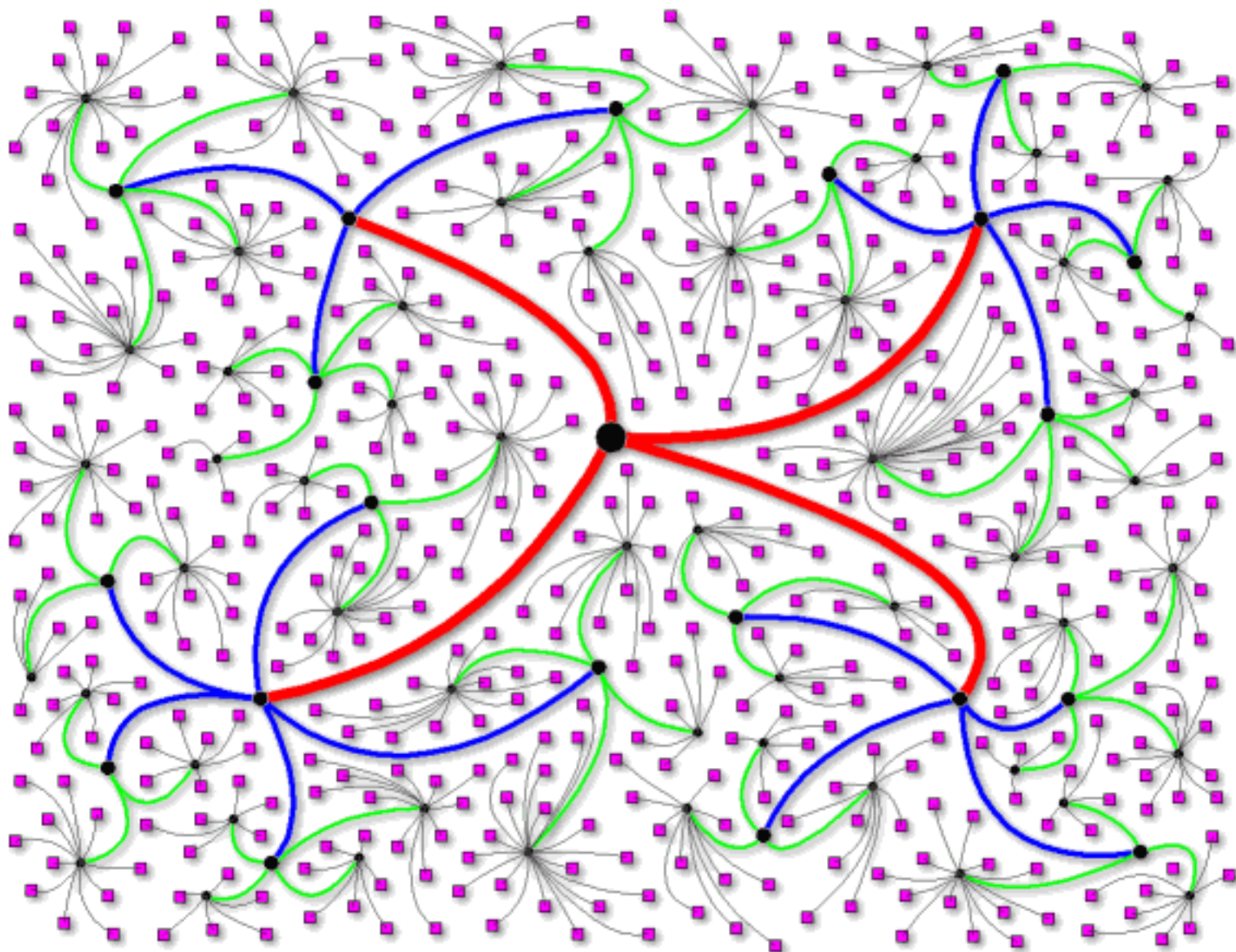# DENIAL OF SERVICE

by Steve Gibson, Gibson Research Corporation

Page last modified: Jun 13, 2001 at 10:58

To understand how Internet services can be denied, we must first understand how they are provided.

What IS the Internet?

As you've probably heard before, the **INTER-NET** is a network of **INTER**connected **NET**works:

▬▬▬	**Top level Internet "Backbone"**
▬▬▬	**Tier-1 Service Providers**
─────	**ISP's, Universities & Corporations**
─────	**Local Computer Connection (Last Mile)**
● • ·	**Various 'Levels' of network routing**

The diagram above shows a simplified, "tree-structured", non-redundant, multi-level network. In other words, a network of networks of networks of networks. As we'll see in a moment, the actual Internet is much more complex, but let's take it one step at a time.

The topmost **RED LEVEL** is commonly called the **Internet "Backbone"**. It is the main highest-traffic network that interconnects the **routers** of the second level.

Internet Routers — shown as round black "nodes" in this diagram — examine the destinations of individual Internet data packets to determine the best route to use in forwarding the data to its final destination.

The **BLUE LEVEL** is generally composed of large "Tier-1" service providers. They purchase "backbone" bandwidth from the major communications carriers and resell it to smaller service providers, universities, and corporations.

The **GREEN LEVEL** is generally composed of smaller (non-Tier-1) service providers, universities, and corporations. They typically purchase bandwidth from Tier-1 providers and interconnect the individual machines within their organizations into a "Local Area Network" (LAN).

The **BLACK LEVEL** is composed of the connections to individual machines. This is often called the "Last Mile" of the Internet and ranges from dial-up modems, to ISDN, to DSL, to Cable Modem, to local Ethernet segments.

The most significant aspect of this system is that through the multi-level networking of networks, every machine can potentially be connected to every other.

What's wrong with this simple system?

We all know from experience how incredibly reliable the Internet has proven to be. The original design, which was funded and roughly specified by DARPA — the U.S. Defense Advanced Research Projects Agency — specified that this "Internet" should be able to withstand and remain functional after an atomic bomb blast.

But returning to our first diagram above, you'll notice that simply "taking out" that main center node, joining the various red top "backbone segments", would instantly chop our "Internet" into four separate disconnected pieces. That's not good. And even absent an atomic bomb, technicians have been known to trip over cords or type in the wrong command, resulting in a loss of network connectivity. The network shown above could only work if every piece worked perfectly. In other words, this simplistic network design is extremely fragile.

It has another problem too:
This simple "tree-structured" architecture routes all traffic between top-level backbone segments through the center node. In other words, a large percentage of the TOTAL bandwidth of the Internet would be flowing through this node, and it's unlikely that any current technology on Earth could adequately handle that much traffic.

The solution is redundancy:

We are now going to mess up our elegant "conceptual" diagram by adding "peering links" between Internet routers at the same level. Though the resulting network is far more complex, its reliability skyrockets due to the redundancy of

traffic pathways:
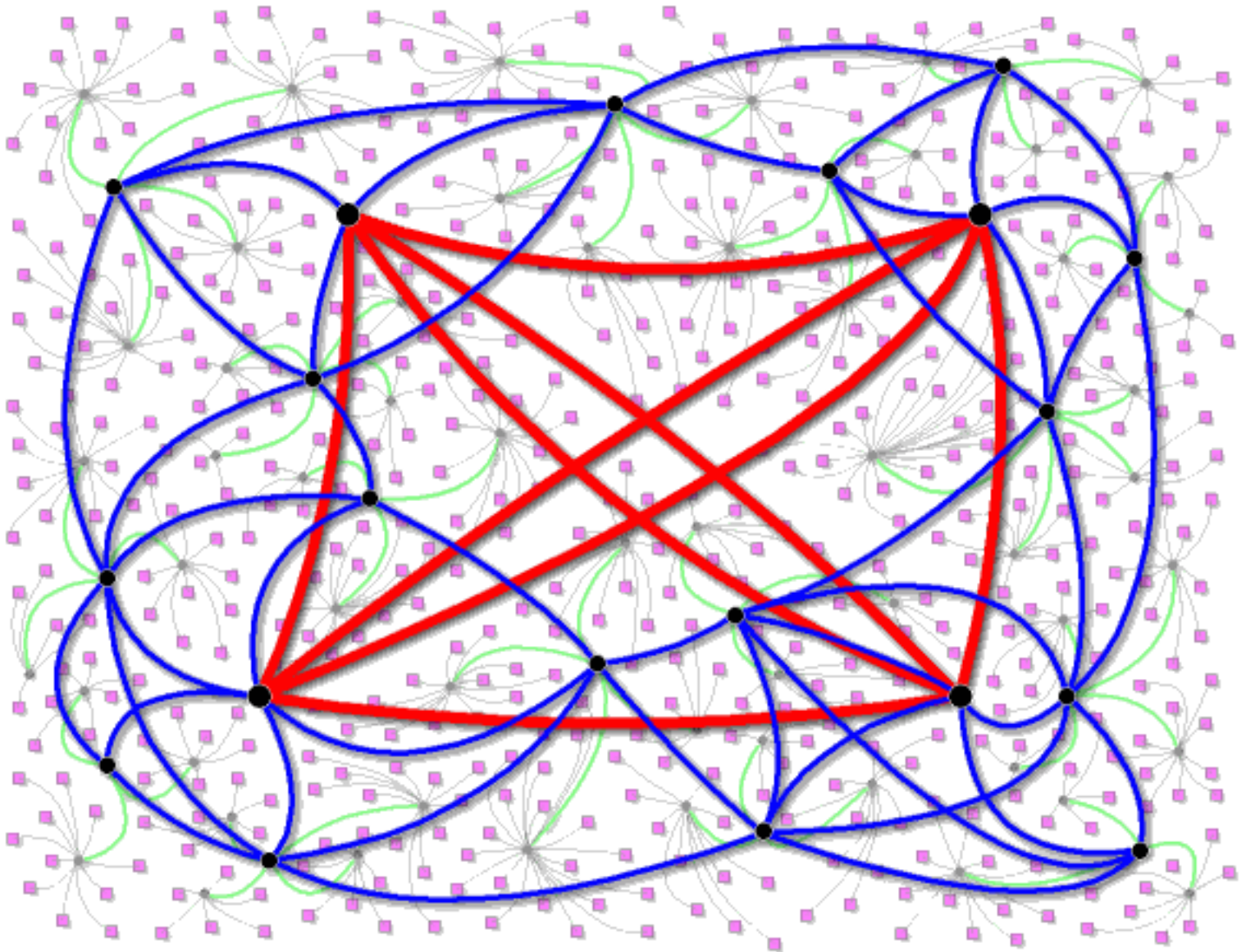
If you'll take a moment to examine this diagram you'll notice that the central router has been eliminated in favor of direct "peering" links (**RED**) among the top-level backbone routers. In cases where traffic demands it, duplicate links are established to deliver reliability and additional point-to-point bandwidth.

At the major Tier-1 ISP (**BLUE**) level, the ISP's have created redundant links between many of their own routers, and have even established "peering relationships" with other ISP's in order to share bandwidth and increase their mutual reliability.

As you can see, with multiple redundant paths to router nodes at every level, a great many links and/or routers can be "down" or disconnected at any time without isolating any region of the entire network. While the smooth flow of data from affected points might suffer a longer than usual transit, especially if lower capacity links were overburdened by a rerouting of traffic, the system "fails soft", degrades gradually, and the data can find a way to its destination.

An actual Tier-1 ISP Network:

With this background, you have everything you need to make some sense of

Verio's national network infrastructure diagram:

As you can see, every "node" of Verio's network is serviced by at least two, and often many more, high bandwidth connections. Any link can go down without interrupting its supply of data.

This diagram is Verio's, so it does not show the main Internet backbone links, nor the many other ISP's whose coverage criss-crosses the United States and smaller regions to provide an incredible level of redundant Internetworking connectivity.

If this Internet interconnectivity interests you, I urge you to check out MANY of the pages on this excellent "Mapping Cyberspace" web site. You will be GLAD you did!

http://www.cybergeography.org/atlas/atlas.html

Armed with a strong understanding of the fundamental architecture of the Internet, let's proceed to examine how data is routed across this huge network . . .

To continue, please see: Internet Routing 101

Purchasing Info   **GRC Mail System**   **To GRC's Home**   **Tech Support**   **Discussions**

The Escalating Threat of Internet

DENIAL OF SERVICE

by Steve Gibson, Gibson Research Corporation

Page last modified: Jun 05, 2001 at 10:51

Armed with an understanding of the overall architecture of the Internet, we're ready to examine the way its data moves from place to place.

Internet Routing 101

The diagrams we saw on the previous page (The Internet: How it works) are so complex, and the specific details of how data packets travel from one machine to another are generally so unimportant, that the entire Internet is often regarded as a "network cloud" where data pumped in one side mysteriously emerges from the other:

But for our purposes, this simplification obscures the mechanisms used by malicious hackers to evade responsibility and detection, so we need to develop a diagram which is somewhat more precise:
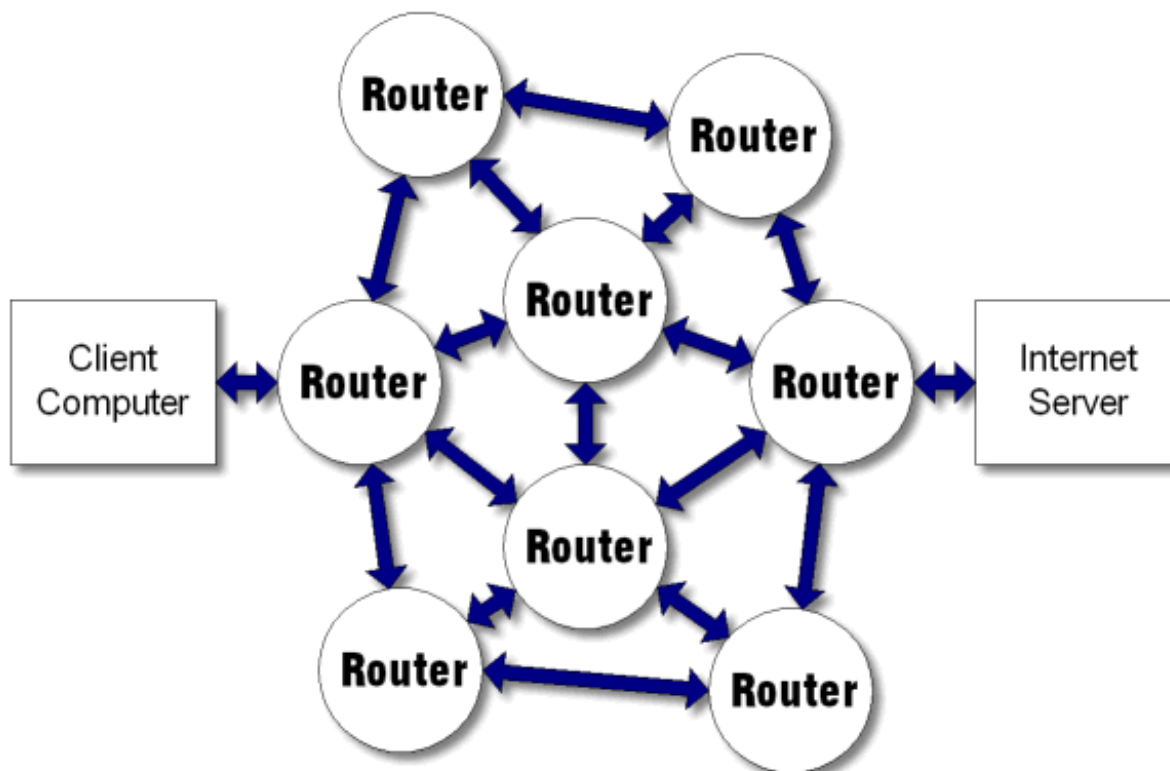
**This simple schematic diagram distills the
essence of what's important about the way
the internet handles and routes traffic.**

As you can see in the diagram above, data moves between the two endpoint
machines by "hopping" from one Internet router to the next and travelling across
the many connecting links in between. You can perform a simple experiment to
demonstrate the "router hopping" of your own computer's data packets for
yourself:

**Chart the "Router Hops" between
your computer and YAHOO.COM**

Internet capable machines traditionally include a UNIX-descended command known as "Trace Route" used to trace the
route taken by an Internet packet as it "hops" from router to router heading toward its final destination. Using this
command (right now) you can quickly chart the entire route between your machine and, for example, "yahoo.com":

The Trace Route command is shortened to just "tracert". So, open an MS-DOS Prompt window or whatever "command
window" your operating system uses, then enter this command and press Enter:

tracert yahoo.com

Something like this chart should be produced, showing each "hop" your data packet takes on its journey from your
computer to ours:

```
C:\>tracert yahoo.com

Tracing route to yahoo.com [216.115.108.245]
over a maximum of 30 hops:

  1    <10 ms    <10 ms    <10 ms   207.71.92.220
  2    <10 ms    <10 ms    <10 ms   gibson.customer.ni.net [207.71.92.240]
  3    <10 ms    <10 ms     15 ms   hs-0-0-328.a03.irvnca01.us.ra.verio.net
[207.71.76.241]
  4    <10 ms    <10 ms    <10 ms   ge-0-3-0.a05.irvnca01.us.ra.verio.net [129.250.46.37]
  5    <10 ms    <10 ms    <10 ms   p4-0-0-0.a05.lsanca01.us.ra.verio.net
[129.250.46.109]
  6    <10 ms    <10 ms    <10 ms   ge-6-2-0.r00.lsanca01.us.bb.verio.net
```

```
[129.250.29.126]
   7    <10 ms     16 ms     16 ms   p4-1-3-0.r01.snjsca03.us.bb.verio.net [129.250.2.113]
   8    <10 ms     31 ms     16 ms   p4-7-0-0.r06.plalca01.us.bb.verio.net [129.250.2.197]
   9    <10 ms     15 ms     16 ms   so0-0-0-622M.br2.PAO2.gblx.net [208.50.13.97]
  10    <10 ms    <10 ms     15 ms   so1-0-0-622M.cr1.pao2.gblx.net [208.50.169.141]
  11    <10 ms     16 ms     15 ms   pos7-0-2488M.cr2.SNV.gblx.net [208.50.169.86]
  12    <10 ms     15 ms     32 ms   206.132.254.41
  13    <10 ms     16 ms     16 ms   bas1r-ge3-0-hr8.snv.yahoo.com [208.178.103.62]
  14    <10 ms     16 ms     15 ms   img5.yahoo.com [216.115.108.245]


Trace complete.


C:\>
```

The first numbered line of the chart (1) will show the IP address of your network gateway (the first "router"), and the last numbered line will contain the IP address of one of yahoo.com's several servers which receive packets arriving at yahoo.com.

How is this chart created?
In the next section you will learn that NO TRACE of a packet's specific path is recorded or retained by the Internet. This means that there is NO WAY to know where an arriving packet came from if it deliberately forges it's origin. But if that is true, how can we generate a chart such as we just have? The answer to this question demonstrates the sublime cleverness of the Internet's original architects.

Looking back at our routing schematic diagram above, notice how the Internet's routing redundancy inherently creates the possibility of "routing loops". Every router is working to move each individual packet toward its individual destination. But if a router were to make a mistake, and accidentally kick a packet back to a router that had already forwarded that same packet to a different router, a "router loop" could form. In this case the packet might "circulate" among these confused routers forever!

Aware of this theoretical possibility (which actually happens frequently), the Internet's designers made a provision for a packet to "age" and finally expire on the Internet. Within a packet's body, a counter known as "time to live" (TTL) is initially set to some adequate count (typically 64) when the packet is first created by the originating machine. That counter is then decremented by one every time the packet arrives at any router. If the packet's TTL counter ever reaches zero, the router worries that this packet is having trouble reaching its destination. So, instead of forwarding the packet to its destination (which apparently isn't working very well) the packet is discarded. The router then creates a "supervisory packet" which is sent to the originating machine to tell it that the packet it originated had an "unreachable destination".

Returning to our "Trace Route" chart:
The clever trace-route program functions by deliberately creating "short-lived" packets that are designed to expire on their way to their destination machine (yahoo.com or wherever). For example, to determine the first "router" in the path, a packet would be generated with a time-to-live (TTL) set to **1**. The first router in the path would receive this packet, decrement the count to zero and return a "destination unreachable" packet to our originating machine. Then the TTL would be set to **2** in order to induce a response from the second router in the path **. . .** and so on.

Isn't that clever?

Okay.
You are probably getting a good general feeling for how this all works **. . .**

Individual blobs of data — which we're calling "packets" — each contain the source IP address of the machine that generated the packet and the destination IP address of the machine that will receive the packet. When any of these packets is dropped anywhere into the Internet, the Internet's routers will take over and bounce the packet to its destination. It's just incredibly cool.

This Internet system was developed by engineers who held it in the same sort of awe and reverence that I do. They were just so amazed that it worked at all, that the idea of some malicious hackers getting their jollies from the deliberate abuse of the system had to have been the farthest thing from their minds.

But we all know today that this is exactly what has happened. The following set of Internet packet routing facts serves to clearly outline the problem:

Internet packet routing facts:

● Each machine participating on the public Internet has at least one unique IP address. This is true for end-user client computers, servers of all sorts, and the intermediate "routers" and "gateways" that shuttle IP data packets toward their destination.

● Each IP packet travelling across the Internet carries the unique IP address of the machine which originated the packet — the packet's "Source IP" — and the unique IP address of the machine to which that packet is being routed — the packet's "Destination IP".

● Internet-connected local area networks (LANs) have a "gateway" router (located at the network's "gateway" IP address) which receives any traffic carrying a destination IP that lies outside of the local network. Machines within a LAN send locally-destined data packets directly to each other, but they send all data packets with non-local destinations to the network's gateway router for forwarding to machines located outside the local network.

● Since an Internet router's job is to forward individual packets onward toward their destinations, routers do not generally care where a packet came from. Consequently, routers typically pay attention only to the packet's destination IP address.

● The packet's "Source IP" is only used when it finally arrives at its destination. (Or if some routing trouble causes the router to send a notification back to the originating machine.) In any event, the Source IP provides the packet's "return address" by informing the receiving machine of the IP address of the machine that originated the packet and placed it onto the Internet.

● As you can see from the routing schematic diagram above, and from the diagrams on the previous page, Internet routers are usually "richly connected" to many other routers. So, when a packet arrives at a router, it uses "dynamic routing tables" to determine the best direction to forward the packet. And, significantly, when a packet is leaving a router, there is no practical means for determining from which previous router the packet had arrived.

● Today, it is generally true that an Internet data packet may be generated by any computer connected to the Internet, may enter the network at any location, and will be immediately routed to any destination machine specified. Only the Destination IP address is examined — no concern is given to the packet's Source IP address — and no record of the packet's route of travel across the Internet is retained. The packet's Source IP is the only statement of the packet's origin.

● If an Internet data packet were carrying an incorrect Source IP, no router or system on the Internet would know or care until the packet arrived at its destination.

● The Internet's massive data traffic, its dynamically changing multi-way interconnection complexity, the lack of individual packet routing history, and the TOTAL RELIANCE upon a packet's Source IP as the specification of the packet's origin, combine to provide nearly perfect anonymity for the originator of "Spoofed Source IP" packets.

The "Bad Guys" know all this, and they're laughing.

They know that the Internet protects them almost perfectly and that we are virtually at their mercy.

Let's proceed to examine the ways in which this system can be abused . . .

Purchasing Info   GRC Mail System   To GRC's Home   Tech Support   Discussions

Why Windows XP will be the

DENIAL OF SERVICE

Exploitation Tool of Choice
for Internet Hackers Everywhere

by Steve Gibson, Gibson Research Corporation

Page last modified: Jun 25, 2001 at 12:33

Page Updates & News:

- **06/15/2001 — Network Egress Filtering**
  I added a new section to examine and address the comments that "Network Egress Filtering" by ISP's is the "real solution" to the problem of Denial of Service attacks.

- **06/20/2001 — GRC.COM Attack Log**
  On Wednesday, June 20th, 2001, we were attacked by 195 Windows 2000 servers running insecure versions of Microsoft's IIS web server. IIS was the apparent point of hacker entry into the system. We describe the attack and list all attacking IP addresses and machine names (where available).

Another LONG page . . .

I know that this is another of my loooooong pages. I worry that it won't be nearly as fascinating as my account of Wicked and the DDoS attacks. However, this is a complex and important issue that can not be quickly summarized.

If you are someone who eats dessert first, I urge you to at least read the story of "'Junior' and his XP Gang". (click the link) then I hope you will come back up here and start from the beginning.

"Hacker" vs "Cracker"

When I want to describe the actions of a "malicious hacker" the term "cracker" falls far short of the mark for me. It just doesn't seem very malicious. So I have stayed with the term "malicious hacker" because it is precisely descriptive.

I know that the majority of people who proudly call themselves "hackers" honor the same ethical principles I do **. . .** so NO disrespect is EVER meant.

"Steve,

I've reviewed your note with the Windows network development architects, as well as our Corporate IT Security and Security Response groups. Your instincts are correct -- they thoroughly understood the nature of the issue . . ."

— A Microsoft Executive

With all due respect to Microsoft, I believe that either the right people within the organization are not yet fully aware of this issue, or that they have not really "thoroughly understood the nature of the issue."


Microsoft has a lot of really smart people — from Bill Gates and Steve Ballmer right on down the line. But they are human, and they sometimes make human mistakes. Sometimes it's worse than that, and as a company they're stubborn in the face of some really bad decisions. Like script-enabling their eMail clients so the virus du jour, like Melissa, can impersonate the user and happily eMail itself across the Internet to everyone in our address books.

What a DUMB thing.

My concern today is that we have another **SERIOUSLY DUMB IDEA** in the works from Microsoft in Windows XP.

I regret my silence when scripting was being added to eMail. It was the dumbest thing I had ever seen, but I didn't care since I use Eudora. So I didn't work to make the world take notice. Now eMail viruses are born daily to travel the Internet at light speed. And it could have — should have — been prevented.

From a recent SANS Security article (See the Security Editor's Note)
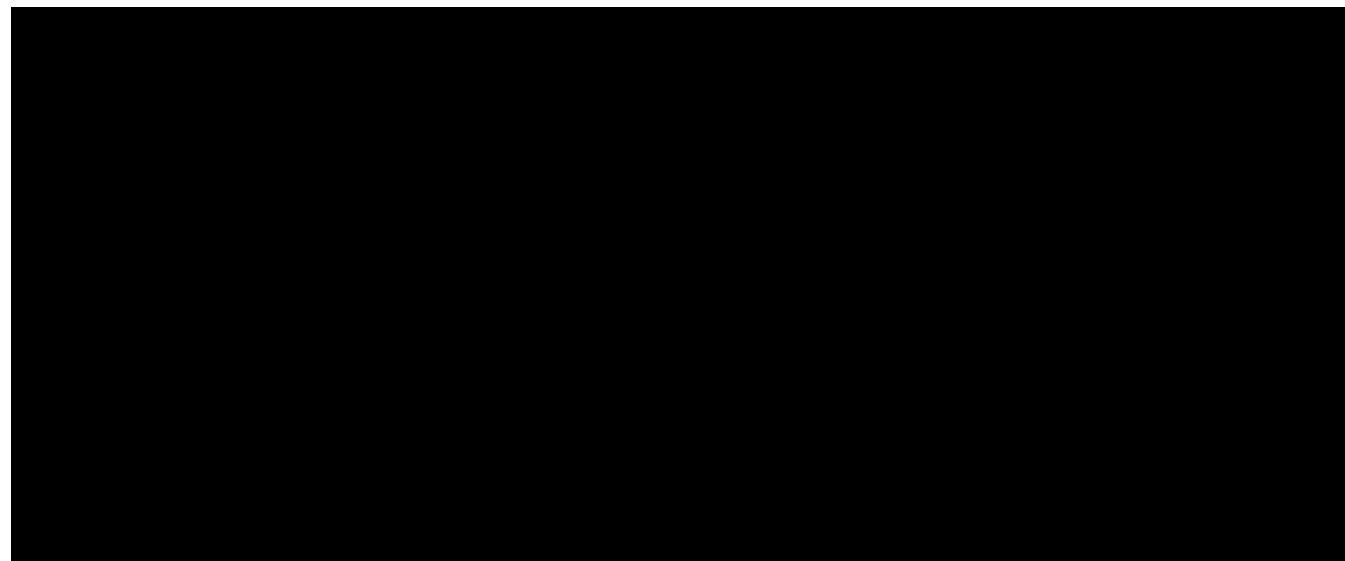
13 & 14 June 2001— Malicious E-Mail Could Cause Problems for Japanese Wireless Internet Customers

A Japanese wireless phone carrier has warned subscribers of its I-Mode wireless Internet service that malicious e-mail messages could cause their phones to dial an emergency number, make lots of calls, or freeze the phone screen. The company advises its customers not to open e-mail from unknown sources and offers suggestions for thwarting the potential problems.

The ComputerWorld Story   The CNET Story

[Editor's Note: Script-enabled mail and web clients are a disaster, and apparently G3 cell phone manufacturers have fallen into the same trap as the designers of MS Outlook and the people who invented Javascript for web browsers. At least with Netscape the user can disable Java and Javascript for web and mail. One suspects that G3 (third generation) cell phone users will not be so lucky.]

This time, with the disaster of Windows XP support for "RAW SOCKETS" looming, **there is still time to get Microsoft to yank it out.** But as the correspondence below demonstrates, I have not yet managed to reach the right people or convince them that they must.

What are "Sockets"?
And why are some of them "Raw"?
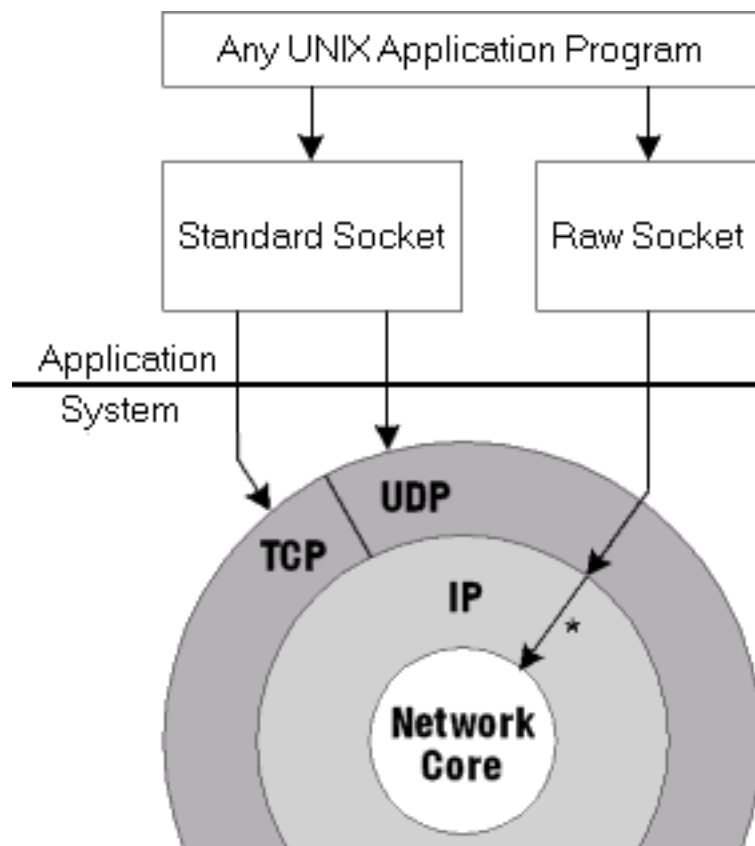
Circa 1981 — The Computer Systems Research Group (CSRG), at the University of California at Berkeley, first mated the Unix operating system to the Internet. This was done by implementing the Internet protocols and creating a so-called "TCP/IP Stack" for Unix. This is shown as concentric regions in the diagram to the left.

To simplify the task of creating Internet-communicating applications, the CSRG designed a simplified abstraction of the complex underlying protocols. They dubbed this abstraction "Sockets" or "Berkeley Sockets". Under this system, application programs request easily-programmed Internet "sockets" and are insulated

from the details of the underlying network protocols.

Data is exchanged across the Internet by either establishing a bi-directional "TCP Connection" between two machines, or by sending a uni-directional "UDP Datagram" message from one machine to another. Both of these data transferring operations employ **standard sockets.**

Smooth and orderly traffic flow across the Internet requires machines to inform each other of various non-data events such as closed ports, network congestion, unreachable IP addresses, etc. The ICMP (Internet Control Message Protocol) was created to fill this need.

The operating system's built-in TCP/IP stack automatically and transparently generates and receives most of these "Internet plumbing" ICMP messages on behalf of the machine. To facilitate the creation of Internet plumbing applications, such as "ping" and "traceroute", which also employ ICMP messages, the Berkeley designers allowed programmers to manually generate and receive their own ICMP, and other, message traffic. As shown in the diagram above, the Berkeley Sockets system provides this power through the use of a so-called "Raw Socket". A Raw Socket short-circuits the TCP/IP stack to open a "backdoor" directly into the underlying network data transport.

This provides full and direct "packet level" Internet access to any Unix sockets programmer.

Beyond their use for supporting simple "ping" and "traceroute" commands, the original Berkeley designers intended **Raw Sockets to be used for Internet protocol research purposes only.** Because they fully appreciated the inherent danger of abuse of Raw Sockets, they deliberately denied Raw Socket access to any applications not running with maximum Unix "root" privileges. User-level applications were thus prevented from accessing and potentially abusing the Raw Sockets capability. (See asterisk '*' in diagram above.)

Full Raw Sockets were created as a potent research tool. They were NEVER INTENDED to be shipped in a mass-market consumer operating system.

The Traditional (safe) Microsoft Stack

Any Windows Application Program

Standard Socket   Raw Socket

Application
System

UDP

TCP

IP

Network
Core
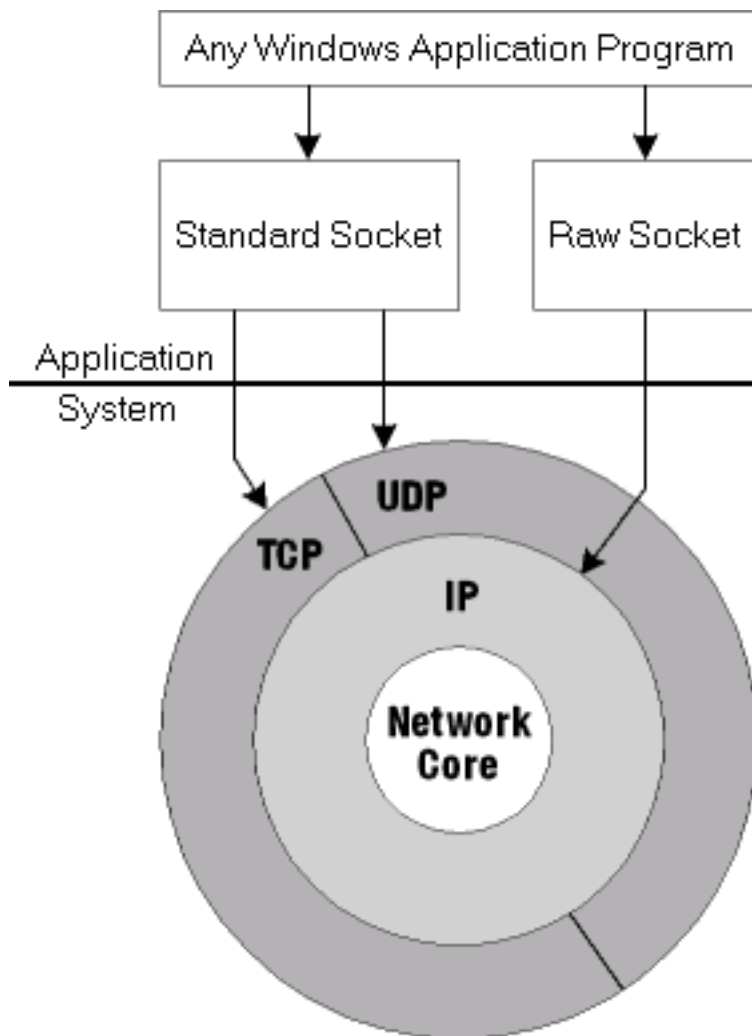
Compare the schematic diagram we've been looking at above, to Microsoft's traditional Windows Sockets (WinSock) implementation to the left.

You will notice that the Windows' Raw Socket's connection **does not "penetrate"** the encompassing IP wrapper layer.

This means that while Windows' Raw Sockets can be readily used for their intended and safe purpose of generating valid ICMP ping and traceroute packets, application programs are effectively cut off from direct "lower-level" access to the underlying physical Internet.

Note: I am FULLY aware that full raw socket-style access can be created by modifying any standard Windows operating systems through the addition of third-party device drivers. I have been a user of such tools for years. However, as I demonstrate below, aftermarket operating system modifications have proven to be irrelevant to the purposes of malicious hackers.

Therefore, as I stated in my DDoS Strange Tale Report, and as I will demonstrate and prove conclusively below . . .

Windows' traditional lack of full Berkeley Unix Raw Socket support has been a silent blessing that has undoubtedly contributed hugely to the stability of the global Internet of the past.

It is the Internet's future that concerns me greatly . . .

What IS the threat from Full Raw Sockets?

I constructed the diagrams above in the form of insulating layers surrounding the system's network core to help demonstrate that the operating system's IP and TCP/UDP protocol layers serve to protect the Internet from direct access by malicious application software running inside the system.

Any system whose fundamental architecture prevents

applications from gaining "Raw" access to the Internet will be MUCH harder to exploit.

As I will show with concrete examples below, these layers of "Internet insulation", which were traditionally provided by Microsoft's "half-baked" Raw Socket implementation, HAVE BEEN A VERY GOOD THING for the Internet so far. But now Microsoft has fatally pierced this insulation by providing full Unix-style Raw Sockets in a high-volume, impossible to secure, consumer operating system.

As proven by the success of the Windows NT server platform (lacking full Raw Sockets support) and the successes of the Internet-connected Windows 95/98/ME platforms (also lacking full Raw Sockets), **full Raw Socket support is absolutely unnecessary for the use of ANY benign Internet applications.** Extensions to the Internet protocols, which represent a valid use for Raw Sockets, would be performed within the operating system's network core. "Sockets" are an application-level interface, not a system level resource, and applications have no valid need for full Raw Sockets. **None.**

In other words, what Microsoft has done with Windows 2000 and Windows XP, is to add a number of powerful and completely unnecessary networking features because, they say, "some people complained about Windows lack of full Raw Socket support". However, it will ONLY be Internet-hostile malicious code that will need to use the advanced "direct access" provided by Windows' new full Raw Socket support.

With Microsoft's traditionally-limited Raw Sockets support, Windows applications were UNABLE to "forge" or "spoof" the machine's actual IP address to hide the source of any malicious traffic they might generate. This source address "spoofing" prevents effective backtracking through the Internet. Windows applications were also unable to generate deliberately malicious "SYN flooding" style attacks, which are essentially unfilterable, and are used to effectively attack any sort of Internet TCP-connection server. (Note that even if the Internet is someday able to block spoofed source IP's, the creation of fraudulent TCP connections, which is not possible using "standard sockets" but is trivial with full Raw Sockets, will remain a problem.)

Until the advent of Windows 2000 & XP, the most common and familiar, complex, potent, and untraceable Denial of Service and Distributed Denial of Service attacks **have only been generated** from Unix-family operating systems. Due to the sheer volume of Windows XP machines soon to be loose in the world, Unix systems will quickly be supplanted as the premiere launching pad for new torrents of Denial of Service floods. This will have an unfortunate corollary effect for XP users:

> **The huge number of Windows XP machines will motivate hackers to find new ways into those machines — AND THEY WILL.   Then users of Windows XP machines will become the most sought-after target for penetration.**

In other words, the use of the high-power, mass-market and unsecurable Windows XP operating system, promises to paint a big target on every user of that system.

In the hands of a clueful hacker, fully-supported Raw Sockets is the enabling factor for the creation of a series of "Ultimate Weapons" against which the fundamentally trusting architecture of the global Internet currently has no

effective defense.

Windows XP is the malicious hacker's dream come true.

My Initial Dialog with Microsoft . . .

A look at my initial attempts to prevent this **. . .**

Hi Greg,

It's been a while since we've talked.

I'm writing to you first for some navigational direction. Windows 2000 and the forthcoming new MS platforms offer something never before seen in any Microsoft platform: A complete implementation of the Windows sockets RAW SOCKETS specification.

While, as a networking developer, I *love* the idea of having this much power, there is a serious DARK SIDE to this which troubles me greatly: For the first time ever, software running on Windows platforms -- including, presumably, the Home-Targeted Windows XP -- will be able to trivially generate IP packets carrying spoofed Source IP addresses.

Before now, the many tens of thousands of Trojans and Zombies being installed into insecure Windows boxes across the Internet on high-bandwidth connections have been COMPLETELY UNABLE to spoof their source IP's. This has been a blessing, since, until now, only UNIX derived boxes have had complete RAW_SOCK support.

But with Windows 2000, and WinXP, etc. … Windows applications will be able to forge their "return address" -- which spells catastrophe for the integrity of the Internet.

I would appreciate having you forward this note to whomever should receive it. I need to understand Microsoft's formal position on this before I go off and make a big bunch of noise and draw the world's attention to this impending threat to the operation and security of the global Internet.

Thanks for your time and attention to this matter.

Oh! … and while I've been intending to mention this danger for some time, I stumbled upon a chunk of hacker source code a few days ago that frightened me a lot:

(Taken directly from hacker source code)

```
>-------------------------------------------------------------
>
> 6. Some words about DDoS from Windows OS.
>    The new feature IP_HDRINCL that comes with win2k can make
>    windows to a powerful DDoS server because it enables IP-
>    spoofing!
>
>    THE IP_HDRINCL
>    setsockopt(ssock, IPPROTO_IP, IP_HDRINCL, (char *)&bOpt,
>    sizeof(bOpt));
>
>    That means win2k-servers can become a base for DDoS that
>    is equal to Unix servers.
>
>-------------------------------------------------------------
```

I sincerely hope that you and Microsoft will sufficiently appreciate the significance of this problem, and the danger it represents.

Please keep me in the cc-chain and let me know what, if anything, transpires.

All the best!

_____

Steve.

Greg was terrific about following up. I received a few progress reports as my note went through "channels". Then a couple of days later I received the following complete reply:

Steve,

I've reviewed your note with the Windows network development architects, as well as our Corporate IT Security and Security Response groups. Your instincts are correct -- they thoroughly understood the nature of the issue … I guess the response is sort of "good news, bad news" story, depending on your point-of-view. In a nutshell, it will be very much harder to get hostile code running on a Windows XP system than on a Windows 9x, or even a Windows 2000 system … but a determined hacker still can … If you'd like to speak to someone about the issue in more detail, I'd be happy to arrange.

Here's the summary:

● Your concern is the ease of spoofing the IP address under which a Windows 2000 or Windows XP system operates on the Internet. You believe that our providing a raw sockets option will make it much easier for malicious parties to develop zombie code that is capable of operating with a spoofed source address.

● You're particularly concerned with Windows XP which, as a consumer

system, will be very widely represented on the Internet and operated by naive home users who won't have the time or expertise to take active steps to configure their systems securely.

* Windows 9x and NT systems have for some time offered the capability to send raw packets from the NDIS layer. This capability is just as exploitable for IP spoofing as the IP_HDRINCL API is.

* The real issue here is more the ability to get control of the zombie system than the ease of writing the zombie code that exploits it. If I can get code running on a system, it's pretty much guaranteed that I can write code to exploit it. The issue is only one of how much code I have to load onto the zombie and how hard it is to get it right (neither a trivial issue, but neither a showstopper for the hacker).

* It is much harder to get hostile code running on a properly configured Windows 2000 system than on an Windows 9x system. While proper security configuration takes some work and expertise, we do provide tools and guidelines to help with this task. Because Windows 2000 primarily appeals to technical home users, we believe this is a reasonable balance.

* It will be very much harder to get hostile code running on a Windows XP system than on a Windows 9x, or even a Windows 2000 system. The integrated Internet Connection Firewall will be enabled by default for users who go through the wizard that connects an XP system to the Internet. A number of system default settings have been tightened to make it harder to get code that makes it through the firewall to run (for example, Outlook XP, Outlook 2000 SR2, and Outlook Express V6 all process HTML e-mail messages in the IE Restricted Sites zone).

In summary, our security folks believe that it will be significantly harder to get an army of zombies running on XP systems than has been the case today with Windows 9x. Unfortunately, even if we prohibited sock_raw, determined hackers can go around that restriction …

Finally, as to the actual "why are providing a raw sockets option" ? , I'm told it's less about "need", and more a response to customer demand for Winsock standard compliance.

As I said, please let me know if you'd like to speak with someone about this. And if you think we are absolutely nuts, then I really would like to know about that, too. I can only assume that you will expose your point of view to your readership on your web site, and if you think we are being insanely irresponsible, then I'd rather hear it from you first, than read it on your site…

Thanks, and have a great weekend!

Greg

I will respond, in detail, to these points Microsoft has raised. But first I would like to raise and respond to some of the other questions raised early in this controversy **. . .**

A Standard by any other Name

Windows 2000 (NT5) was the first Microsoft Windows platform to bring the full "Berkeley Sockets" specification — including Raw Sockets — to WinSock. Until this time, there had been essentially NO APPARENT CHANGE in Microsoft's TCP/IP stack.

In fact, while discussing the OS Fingerprinting capabilities of his well-known "nmap" Internet scanner, nmap's author Fyodor, has this to say about nmap's detection of Windows versions:

> […] Even with all the tests above, nmap is unable to distinguish between the TCP stacks of Win95, WinNT, or Win98. This is rather surprising, especially since Win98 came out about 4 years after Win95. You would think they would have bothered to improve the stack in some way (like supporting more TCP options) and so we would be able to detect the change and distinguish the operating systems. Unfortunately, this is not the case. The NT stack is apparently the same crappy stack they put into '95. And they didn't bother to upgrade it for '98.
>
> But do not give up hope, for there is a solution. You can simply start with early Windows DOS attacks (Ping of Death, Winnuke, etc) and move up a little further to attacks such as Teardrop and Land. After each attack, ping them to see whether they have crashed. When you finally crash them, you will likely have narrowed what they are running down to one service pack or hotfix.
>
> I have not added this functionality to nmap, although I must admit it is very tempting :).

If you read through Fyodor's fingerprinting description, you will see that nmap is a superlatively sensitive detection tool that can generally sense even the tiniest changes in the implementation of a system's TCP/IP stack. (Fyodor is certainly a "hacker" in the truest and most positive sense of the term.) Yet Microsoft's stack apparently never changed **. . .** until it suddenly changed completely.

Several people have been quoted in the press defending Microsoft's stance by simply stating that "following standards is a good thing". In this context it is important to recognize that "Unix Sockets" is simply a "specification"; it has never been ANY sort of recognized "standard". For this reason, the proper way to view the past situation, is that the traditional Windows Sockets system implemented only as much of the Raw Sockets portion of the full Berkeley specification as was useful and required for Windows application software.

In other words, Microsoft's original "WinSock" was exactly right for a consumer operating system.

I agree that following good and safe SPECIFICATIONS can be a good thing. But it seems to me that blindly following ANY recipe, whether it's a specification or a standard, and lacking an understanding and independent evaluation of its role in the intended application, is tantamount to replacing your own judgement with someone else's. For a well-informed person, abdicating responsibility is not always a good

thing.

Perhaps these people have never actually programmed the Windows Sockets system (as I have extensively). There is very little about Microsoft's Sockets — with their wild extensions (which I love by the way) — that follows the Berkeley specification. So the truth is, that either with or without full Raw Socket support, Windows Sockets never has been, and never can be "standard".

Windows 2000 -vs- Windows XP

Other people have noted that Windows 2000 is already out in the world with full Raw Socket support. They seem to believe that my lobbying so firmly against the subsequent release of WinXP — with its similar Raw Sockets — is the equivalent of closing the barn doors after all of the horses have escaped.

These people miss the essential aspect of "scale". If Microsoft were going to sell only a few thousand copies of Windows XP, I would not be wasting either your time or mine with this entire issue. But whereas Windows 98 and Windows ME have been largely uninteresting upgrades, Microsoft has loaded so many new goodies into WinXP that it will make for a compelling Christmas season.

Microsoft has executed this perfectly: In the near future, Windows XP is going to become THE generic consumer personal computer operating system.

Therefore, my concern is with the **DEFAULT** feature-set of the system and with the probable size of that feature-set's installed base. Sure, I wish that Windows 2000 were **also "Raw Sockets Neutered"** so that malicious hackers could not assume that all Windows 2000 machines were exploitable.

I have no problem with the idea of an after-market add-on download pack from Microsoft, or of making the "deluxe stack" available in a Windows XP resource kit or MSDN subscription.

It is the idea that EVERY CONSUMER MACHINE will have such dangerous capabilities that are NOT NEEDED AT ALL for Internet connectivity, that strikes me as being SO unnecessarily dangerous and . . . ultimately . . . dumb dumb dumb!

Network "Egress" Filtering & ISP Responsibility

Many thoughtful and well-informed people have suggested that the **real responsibility** for stopping these attacks lies not with the behavior of the user and/or their Internet-connected machine (e.g. Windows XP), but with the Internet's ISP's. These people point to well-known and long-established RFC's (Internet standards documents) (RFC 2267) (RFC 2827) and other Internet "Best Practice" documents which recommend that packets carrying spoofed source IP addresses should not be allowed to "egress" (leave) the ISP's locally-controlled network. Such clearly invalid packets should simply be discarded as they attempt to "escape" out onto the global Internet.

The beauty of "network egress filtering" is that each ISP becomes responsible for curtailing the IP spoofing of their own users. As I explain on my (still unfinished) DoS pages, once a forged packet "gets loose" from the ISP, and out onto the Internet, the task of tracking it back to its source is essentially impossible. The only opportunity to "block and drop" a spoofed packet is while it's still within the ISP's local network where it is EASILY identifiable as invalid and forged. Once that packet "egresses" onto the main Internet backbone, it's too late.

Adding Egress Filtering

For many ISP's, implementing egress filtering is as simple as adding a SINGLE LINE to the configurations of their various routers. For example, Cisco routers have included this option for years, merely requiring the addition of this single line:

```
ip verify unicast reverse-path
```

In most cases, that's all there is to it. However, despite the fact that this has been known and discussed for more than three years (the issue date on RFC2267) the vast majority of ISP's have simply not bothered with this simple security measure.

I believe that proponents of ISP network egress filtering are COMPLETELY correct. I have stated this at the conclusion of my previous page describing the Wicked DDoS Attacks. My announced plans for "Spoofarino", a free, user-oriented utility for encouraging ISP accountability for the lack of egress filtering, has already been discussed by the computer press. Today, the practice of **network egress filtering** is more the exception than the rule, but we can hope that it will be widely adopted as these issues attain increasing visibility in the future.

However, this potential for an improvement in the Internet's infrastructure notwithstanding, it is important to recognize that **. . .**

Egress filtering does NOT solve the whole problem.

While egress filtering will be a good thing once it exists, it fails to solve the problems of Denial of Service attacks in two ways:

- **Local Domain Spoofing:**
  Egress filtering operates by verifying that a packet's "return address" lies within the local network domain. Egress filtering DOES NOT, and can not, verify that the packet ACTUALLY CAME FROM the designated machine within that domain. Since the local network domain being managed by a router often includes thousands of valid IP addresses, any machine may still generate forged packets which appear to be sourced from any other addresses within its immediate neighborhood.

  Therefore, the site under attack will still have difficulty filtering the attack and/or identifying the true attacking machine(s). Rather than identifying and perhaps blocking the individual IP addresses of malicious machines, the inbound routers of a site under attack would need to temporarily ban entire "malicious networks" from access. The effect of this would be that sites under attack would "go dead" to whole regions of the Internet which contain "locally spoofing machines". This is hardly an optimum solution, and even so, it requires a degree of router-blocking sophistication which is uncommon.

- **Non-Spoofing Attacks:**
  The widespread availability of trivial source IP address spoofing is **only one** of the problems created by Windows XP's support for full Raw Sockets. Unlike any previous, unmodified, consumer Windows operating system, Windows XP supports the generation of **SYN-packet floods**.

  The Windows-hosted distributed attacks against grc.com employed 474 machines flooding our Internet connection with ICMP and UDP traffic. We were fortunately able to filter those attacks, and remain on the Web, **only because** those attack-hosting Windows machines were unable to generate SYN-floods.

  Attacks hosted on the future Windows XP consumer operating system will have no such limitations. Non-spoofed attacks, which will never be blockable by egress filtering, will be far more damaging when hosted by Windows XP than previous consumer versions of Windows.

As this analysis demonstrates, network egress filtering is undoubtedly a good thing for the long term future of the Internet. But it does not, and can not, provide a cure-all solution to the problem of the Internet protocol abuse promoted by the existence of Windows XP's full Raw Socket support.

Microsoft Reacts

After the May 31st release of my widely read [DDoS attack report](), in which I was strongly critical of Microsoft's publicly stated and confirmed intentions to equip the consumer-targeted Windows XP with full Raw Sockets capabilities, Microsoft produced and publicized a formal response on their TechNet web site. You should take a look at that so that you have an updated sense for Microsoft's position:

[http://www.microsoft.com/technet/security/raw_sockets.asp]()

I will summarize what I read as Microsoft's stated position, point-by-point, and reply to each in turn:

Microsoft's Position:

> ## This is not really anything new, since previous versions of Windows had support for Raw Sockets.

HUH?!!

This is a very disappointing position for Microsoft to be taking. Within the present context, they MUST KNOW that this is simply not the truth. The entire debate centers upon the distinction between partial and full Raw Socket support. So I can only presume that Microsoft is hoping to achieve some quick public relations damage-control, even at the **ultimately extreme cost** of sacrificing the truth.

On Friday, June 8th, the TechNet page referenced above states:

> " **. . .** as raw sockets implementations are already present in Linux, VMS, Unix, Mac OS X, and even in previous versions of Windows."

And in their first reply to me, shown above:

> "Windows 9x and NT systems have for some time offered the capability to send raw packets from the NDIS layer. This capability is just as exploitable for IP spoofing as the IP_HDRINCL API is.

The following three examples provide concrete evidence of Microsoft's apparent confusion over the issue of full Raw Socket support in previous versions of Windows:

- **Proof #1:**
  My original note to Microsoft quoted from the "readme" file of a [Windows DDoS attack tool named Skydance v3.03](http://www.megasecurity.org/trojans/skydance/Skydance3.03.html):

  "The new feature IP_HDRINCL that comes with win2k can make windows into a powerful DDoS server because it enables IP-spoofing!" …and…

  "That means win2k-servers can become a base for DDoS that is equal to Unix servers."

  Also appearing on that readme page under the section "Client Usage", is the following:

  "The Client will try to use a spoofed source address. You should test your spoofing-ability first to ensure that you can not be revealed. The test will fail on WinNT and Win9x/Me systems. It should not fail under Win2000."

  How much more plain can that be? Windows 9x/ME and WinNT DO NOT HAVE the Raw Socket capability to spoof the machine's actual IP address. This was only added into Windows 2000 and is now being carried down into the consumer market by Windows XP.

  Here, located on the "megasecurity" hacker site (provided with their knowledge and permission), is the entire "readme" page for this typical Windows DDoS attack tool. Note that it is not currently useable on any consumer-grade Windows systems **. . .** but Microsoft's XP will soon be changing that, to the delight of malicious hackers everywhere:

    http://www.megasecurity.org/trojans/skydance/Skydance3.03.html

I got a personal chuckle out of the last paragraph on that page. It talks about the trouble with the new personal firewalls and suggests that by renaming the DDoS Trojan to the name of a common Internet application (like "ping.exe") — which presumably has firewall permissions — you may be able to get past outbound blocking firewalls by "impersonating" a permitted application.

Those of you who have been following my work at grc.com will note that this was **exactly** the personal security problem I anticipated when I created and promoted my free [LeakTest utility](). Its goal was to bring market pressures to bear, thereby inducing firewall vendors to prevent this trivial exploit. At the time of LeakTest's release, all but one firewall was vulnerable to this. But today, every reputable firewall has been updated as a direct consequence of LeakTest's influence.

As you might imagine, the personal firewall vendors were as furious with me then, as Microsoft appears to be now. But those vendors are happy today, and their users are much safer.

(So as not to confuse people, I should mention that BlackICE Defender still fails the LeakTest, and would therefore presumably allow this Trojan to operate. However, Network ICE has stated that, despite the declarations on their web site, BlackICE is not a firewall. So it is exempt from the class of products I refer to as "reputable firewalls".)

While you're at the megasecurity site, take just a moment to browse through their catalog of the Trojans which will soon be competing for space on Windows XP hard drives:

http://www.megasecurity.org/trojans/

Impressive.

Proof #2:
Located on Internet.com's "CodeGuru" site is another of the many typical articles and code samples floating around the Internet which highlights the unique power of Windows 2000 and — soon — Windows XP:

http://www.codeguru.com/network/tcpip_lib31.html

The ZIP file, which may be freely downloaded from that page, contains complete source code for this set of "C++" library functions to leverage the "WinSock" system for the purpose of experimenting with the Internet.

Of particular interest is the included "Attack" program which, according to [that program's readme.txt file](), currently only runs under "w2k". Why? Because Windows 2000 is the only Windows operating system that currently supports full Raw Sockets. But as we all certainly know by now, this will soon change.

Under the "How does it work?" section, the author explains:

"When I send only the SYN, and spoof the address of a non working address

(no host over there to reset the connection), the remote system will never get the SYN+ACK response and it will wait until that connection will time out (around 20 seconds), assume I'll send 60,000 of this sockets, the amount of resource I'll tie up will do some damage."

Under the "Requirements" section, the author explains:

"Currently working only under w2k."

Right.

This author is describing a classic SYN flooding attack using a spoofed Source IP. NO PRIOR VERSION OF WINDOWS allows its applications to arbitrarily generate Internet packets. As this example demonstrates, deliberately invalid — and malicious — SYN packets can NOT be generated unless the application is running on Windows 2000 **. . .** or, soon, Windows XP.

This sample highlights another interesting aspect of Microsoft's poor judgement in this matter:

The threat of attacks is NOT ONLY from surreptitiously installed remote-control Zombie/Bot Trojans, but also from PC hobbyists who will soon be able to gleefully launch untraceable spoofed IP SYN-flooding attacks from the comfort of their own bedrooms. Presumably after finishing their homework.

The typical teenage hacker has not had access to Windows 2000. He or she has been limited to playing video games on Windows 95/98/ME. But this Christmas will change all that: When "Junior" asks Mom and Dad if he can get an upgrade to the new really cool Microsoft Windows XP for Christmas, Mom and Dad will smile and nod. "What a GREAT idea!" they think to themselves.

Yeah. Great.

- **Proof #3:**
  One of the most well known and sophisticated remote-control attack Zombies is named: **Trinoo**. This remote-control Bot was originally written to run on compromised Unix- and Linux-style platforms which, as we all know by now, have traditionally been unique in having the ability to generate spoofed source IP flooding attacks.

  Trinoo was such a successful attack tool over on the *NIX platforms, that it was "ported" to the Windows environment under the name: **WinTrinoo**.

  But, of course, WinTrinoo's malicious capabilities are somewhat limited under Windows. Unlike its Unix cousin, **WinTrinoo can neither spoof source IPs, nor generate SYN flooding attacks.** You KNOW that WinTrinoo's authors know how to spoof source IPs and generate SYN floods. They did it for Trinoo. So why doesn't WinTrinoo have the same power under Windows? You know why: Because Windows has traditionally lacked support for the full Raw Sockets specification.

  Next year, after **"The XP Christmas of Death"** has passed, tens of millions of home PC's will be happily running Windows XP. How many minutes do you

think it will take for **"WinTrinoo2"** to arrive on the scene and for it to take full advantage of XP's Unix-style full Raw Socket support?

> **A note to the Internet's Hackers:** It would be TERRIFIC if you were to name your second-generation WinTrinoo version: **WinTrinoo-XP !!**

So now, in light of what you've just seen, reconsider the intent behind Microsoft's summarized position, as documented above:

"This is not really anything new, since previous versions of Windows had support for Raw Sockets."

What are they thinking up there in Redmond?

I hope it is clear to you, in light of this little bit of evidence (there's an endless amount more), that the currently planned release of Windows XP into the consumer market, represents a crucial mistake. And given that Microsoft is fully aware of this, a shocking example of corporate hubris.

NEXT

This quote is taken directly from Microsoft's TechNet page referenced above:

From the TechNet Page:

> **"The presence of operating system-level functions to manipulate data packets is not a critical factor in the number of DDOS attacks. If it were, the explosion in DDOS attacks should have already occurred, as raw sockets implementations are already present in Linux, VMS, Unix, Mac OS X, and even in previous versions of Windows."**

We have just examined the obfuscation that was apparently intended at the end of that quote. (Regarding the applicability of full Raw Sockets to previous versions of Windows.) I hope you're no longer fooled by that. Let's look at the rest of it.

"If it were, the explosion of DDOS attacks should have already occurred..."

Perhaps Microsoft hasn't been reading about the rapid rise (explosion) in the number of DDoS attacks which is already occurring. One must wonder how they could be unaware of this since they have, themselves, been frequent targets of those attacks. Furthermore, they must know, as I demonstrated above, that the widespread availability of Linux and Unix, with their "system-level functions to manipulate data packets" are **clearly responsible** and are a "critical factor" in the number of DDoS attacks.

It is precisely because of the rapid growth in the number of hobbyist-owned Unix and Linux boxes — often configured insecurely then compromised with Trojans — that we are now seeing a rapid growth in the number of DDoS attacks.

Microsoft is about to massively escalate this problem!

Although it is not completely clear what message Microsoft intended to convey with that quote, what they APPEAR to be saying here is something along the lines of:

"Everyone else has full Raw Sockets, . . . so why shouldn't we?"

Assuming that this is what that quote was trying to say, it raises a good question which is worth exploration:

As we have seen, it is indeed unfortunate that "everyone else" has full Raw Socket support. The Internet has already been suffering the consequences. That problem is certainly going to grow with time and needs to be dealt with as well.

The fact that Microsoft was not the first to make this crucial mistake on the Internet in no way reduces their now-fully-informed responsibility to prevent their negligent compounding of the problem.

The installed-base of consumer Windows operating systems dwarfs that of all other platforms combined. In shipping their Windows XP system, squarely targeted at the home and small office user, tens of millions of existing Windows platforms which have never had full Raw Socket support, will be upgraded overnight into powerful Internet attack platforms. And all new computers sold after Windows XP's release will have that built-in.

If you think the Internet is in trouble now, just wait.

NEXT

From the TechNet Page:

> **"Nor is the absence of such functions a significant impediment to such attacks. Most modern operating systems allow new functions — including networking functions — to be added via installable drivers. An attacker who had the ability to install zombie software on another user's machine could just as easily install a network driver to provide any functions it needed, including functions to disguise the source address of the attack."**

That is absolutely true . . . and absolutely irrelevant.

Every one of the three concrete examples we looked at first demonstrated that — in actuality — the lack of the DEFAULT AVAILABILITY of full Raw Socket support in traditional versions of Windows, **completely prevented** that malicious tool from gaining access to IP spoofing and TCP flooding capability. We KNOW that they all

wanted it. Trinoo has it under Unix and Linux, but WinTrinoo doesn't under Windows. The other program examples apologized that they were only useable under Windows 2000 because of W2K's support for full Raw Sockets.

There is a huge PRACTICAL gulf between what COULD be accomplished in theory, and what IS ACTUALLY ACCOMPLISHED in practice.

Operating system kernel-level "packet drivers" are freely available on the Internet. Microsoft even provides a (buggy) sample of such a driver in their own "Platform SDK" (A sample kit for Windows developers.) But these have existed for years and have never been employed by any of the popular malicious tools. The reason for this is that it has never been nearly as simple as Microsoft makes it sound. Those solutions tend to be operating system version dependent and difficult to reliably install — especially remotely. As a result, and despite what might be possible, all of the evidence demonstrates that malicious tools exploit the interfaces provided by the native operating system.

The addition of full Raw Socket support to the DEFAULT Windows XP consumer product guarantees that the next-generation Windows-hosted tools of mass malicious exploitation will be far more powerful than any previously designed for today's Windows operating system.

It's just that simple.   **How could Microsoft NOT see that?**

NEXT

From the TechNet Page:

> **"The real issue is whether the attacker could run hostile code on another user's computer. Like viruses, Trojan horses and other hostile code, a zombie program can only run if an attacker can install it and run it."**

Wrong. The **REAL ISSUE** is that Windows XP puts an operating system in the hands of the consumer which allows ANY SIMPLE APPLICATION PROGRAM — whether installed by a malicious hacker or used by the system's owner — to trivially generate sophisticated source IP spoofing Denial of Service (DoS) TCP SYN floods. None of Microsoft's previous, unmodified, consumer-targeted Windows operating systems allowed this. This is a huge change for the worse.

Remember "Junior" whom we met awhile back when he asked his parents for a Windows XP upgrade for Christmas? Let's take a look at what "Junior" is up to after he returns to school from Christmas vacation **. . .**

The Story of "Junior" and his XP Gang

"Junior" is basically a good student and a good kid who likes computers and trades mp3 music files and Windows programs with his friends at school.

But not long after Christmas, one of his friends finds a cool new program written for Windows XP. If you try to run this program (which is called "NukeEmNow.exe") on the old versions of Windows, which they all had before Christmas, it just says: "You need WinXP to use this safely." and it won't do anything else.

This little Windows program allows anyone to launch a completely untraceable "personal" Denial of Service attack. It launches a protracted SYN flood with spoofed source IPs against any web site the user wishes. When you run it under Windows XP, a window pops up asking its user to enter the URL of the website to be driven from the Net.

A couple of the school's more knowledgeable young computer geeks explain that Windows XP is really cool because, unlike the earlier versions of Windows, XP lets you "Spoof" your computer's IP address to make you completely anonymous and invisible when attacking others on the Internet. The geeks caution that the program should only be run from a diskette, never stored on the hard drive, so that no evidence of its use is ever left behind. Since Windows XP has all the fancy full Raw Socket support built right in, the "NukeEmNow.exe" program, which was written in Visual Basic, fits easily on any spare diskette.

Of course, this program didn't exist and wasn't written before XP, because it wasn't possible under any traditional versions of Windows. But now many such programs are popping up all over the Internet. All it took was the new release of Windows.

Now, after school every day, "Junior" and his close friends — all with fresh Windows XP Christmas upgrades and cable modems — meet behind the Gym to decide whom they want to punish with their own little neighborhood-wide Distributed Denial of Service attack.

Feeling a bit like super-cool spies, glancing around stealthily, they synchronize their watches and each head to their respective homes, ready to click the "Launch Attack" button at the appointed time.

It is a virtual certainty that applications such as the hypothetical "NukeEmNow.exe" will be written for Windows XP, and that those applications will be used by malicious individuals of ALL ages. (I didn't mean to single-out teenagers.)

It is worth noting that since TCP "SYN" packets are extremely small (60 bytes) compared with data-carrying packets (1500 bytes), many more SYN packets can be sent per second than data packets. This gives a SYN flooding machine more

"packet potency" than one which is attempting to transfer valid data. The consequence of this, is that a **single SYN-flooding machine** can completely knock out any other machine connected at the same or lesser speed. Coupled with Windows XP, and a breed of cyber-war toys like the still-hypothetical "NukeEmNow.exe" described above, we can expect "one-on-one" cyber-battles between individuals. If someone doesn't like what someone else says or does, they are too easily blown off the Internet.

Let me say it again: This is all COMPLETELY UNNECESSARY since no Windows applications have ANY need for full Raw Socket support. No VALID use exists outside of an Internet research setting. Raw Sockets were only included by the original Berkeley designers for Internet protocol research. In a consumer computer system, they will only be exploited for malicious purposes.

So, as we see, the "real issue" (to quote Microsoft) is NOT whether the attacker can run hostile code on another user's computer. I submit that the "real issue" is whether a personal computer can be **much too easily programmed** to generate untraceable and maliciously damaging Internet traffic. Until now, for Windows, that answer was no, and as a direct consequence it was never done.

Windows XP flaunts its ability to trivially generate malicious traffic. You already know what will happen.

"Microsoft Security"

Here is a simple fact:

It is absolutely impossible to create a secure, consumer, personal computer.

Security is black and white. Either you are secure and protected, or you're not. Strange as it might seem at first, I don't blame Microsoft for their demonstrated inability to build a perfectly secure personal computer. After all, it's not possible. But I do hold Microsoft responsible for continually marketing and selling something they can never produce. And they MUST be held responsible for the consequences of believing their own marketing and press.

Ask any real security expert, like Bruce Schneier of Counterpane Internet Security. They will tell you flat out that it's an impossible task to secure a personal, consumer, computer. Why does Microsoft continually insist otherwise? Because it is what people desperately want to hear, and desperately want to believe.
Well . . . it's not possible.

Microsoft's software has NEVER been secure, and NEVER will be. With each generation of feature-rama upgrade, it becomes more and more complex, and less and less understandable. There can not be anyone left at Microsoft whose mind can still grasp the technical details of the entire system. They had to give that up with MS-DOS. Microsoft's lack of security foresight is single-handedly responsible for creating the eMail virus. Their consumer operating systems — as well as their high-end server platforms — are notoriously insecure.

Just two weeks ago (May 23rd, 2001) . . .

Windows MediaPlayer Security Fixes

Microsoft has released security fixes for its audio and video player. **The problem could allow hackers to run any code they want on your machine.** The security advisory has details for the technically inclined. Media Player 6.4 users should install a patch which has been posted on its security website. Users of Media Player 7 should install the latest Windows Media Player 7.1, which is available at Microsoft's media website.

7.1: http://www.microsoft.com/windows/windowsmedia/en/default.asp
Bulletin:
http://www.microsoft.com/technet/security/bulletin/MS01-029.asp

Whoops.

So the question is **. . .** Have **YOU** installed the appropriate patch described above? Oh, you didn't know about it? Gee. Microsoft now says that any hacker could run any code they want on your computer. This sort of thing is a DAILY OCCURRENCE with Microsoft and Windows. Sure, it's a daily occurrence because Microsoft and Windows are the biggest targets, but this also means that they have the biggest responsibility.

But, patching Windows doesn't always work either **. . .**

From the SANS Institute NewsBites service

13 & 14 June 2001— Exchange 2000 Patch Woes

The first patch Microsoft issued for an Exchange 2000 security flaw contained an error that caused servers to hang. The second, which contained outdated files, did the same thing. The company released a third version of the patch last week. One security consultant described the patch's effect as essentially launching a denial-of- service attack on one's own server.

The ZDNet Story   The ComputerWorld Story

And then there's what the hackers know **. . .**

From the SANS Institute NewsBites service

13 June 2001— Cracker Group Defaces More Sites Because They Can

A cracker group notorious for its defacing scads of Chinese web sites earlier this year has recently defaced a dozen sites worldwide; all the sites have in common is the word "security" in their domain names. In an email to CNET, the group claims that they target Windows NT and 2000 servers because they are so easy to infiltrate.

How, then, can anyone accept Microsoft's defense for adding incredibly exploitable and utterly unnecessary Internet technology into their base consumer-level system as: "Don't worry, THIS one will REALLY be secure." ??

You MUST KNOW that not long after its release, the world will begin finding huge security holes in Windows XP. Oh sure, Microsoft will issue patches. Then the users will be blamed for not installing them in a timely fashion. What's WRONG with those damn users anyway?

And even if, against all logic and our wealth of experience, you're on the fence with this question . . . WHAT IF MICROSOFT IS WRONG? There is just too much riding on the issue of the security of this completely unproven new system. We lose NOTHING if I am wrong about this and if Windows XP has its full support for Raw Sockets removed. But . . .

What if they are wrong?

Speaking of which, this just in . . .
June 11, 2001

Office XP Cracked Already

I was just sent the following email. Name withheld on request:

" I find it amazing that MS put so much effort into forcing registration and locking down Office XP to prevent piracy and yet just yesterday I was given a copy of Office XP pro with Frontpage and it was "Cracked". Now while I don't use pirated software nor do I have the need to do so. I did have to install it to see if what I was told about the CD was true. And I'll be darned if it wasn't. I put the CD in and the MS Installer kicked in asked me for a long CD Key and BOOM it went about its business. I filled in my installation choices I chose upgrade 10 minutes later I was running all of the applications. In the MS Office tools menu I saw the option to activate my software I clicked it and I got a Message state my products were already activated. I tried this online and offline, win 98 and win2k everything went flawlessly. Now I have removed the pirated software from my machine . . . "

As I said, it is impossible to create a secure consumer personal computer.

The Third-Party Joke

And EVEN IF Windows XP shocked the world by turning out to be secure, that will last ONLY until the myriad of Windows applications start loading themselves into the system.

How many people complain that the annoying "Comet Cursor" keeps getting installed into their computer whenever they visit certain web sites? That's Comet

Cursor's CODE being downloaded and run without the user's knowledge or permission. Microsoft's default web browser settings — which the typical user uses without a second thought — allows all manner of similar remote web-based exploits. What happens when a Windows XP user innocently surfs to a site that was set up to take over those machines?

When I reverse-engineered the Aureate/Radiate advertising Spyware, to create the OptOut spyware detector and removal tool, I mentioned that it would be trivial for any malicious hacker to commandeer the Aureate Trojan and cause it to do their bidding. One line added to an innocuous and unprotected file, will cause any of the more than 30 million Aureate Trojans to "phone home" to a different server. From there it's trivial to have the Trojan accept a file to download and then run it. And if all that's not worrisome enough, the Aureate Trojan undetectably runs within the Internet Explorer browser process; this lets it slip past the system's firewall by trading on the browser's Internet access permissions. So much for Windows XP security.

As any of you who run a personal firewall with "noisy logging" know, routine scans for the PC Anywhere remote control utility are STILL occurring on the Internet. Why? Because people installed PC Anywhere in their machines to give them remote access across the Internet. The only problem was, a great many of these people never bothered with a password. Thus, anyone could scan the Internet to find a machine running the "PC Anywhere" Trojan and "own it". So much for the "security" of that machine.

As you can see from these examples, the goal of an "absolutely" secure personal computer for the masses is impossible to achieve. It's true that exercising extreme care and caution can result in "a more secure PC". But that can only be achieved in degree, never absolutely. For example, even Windows 95 could be quite secure if the user were careful about its configuration and diligent about its use. But the average consumer can not be expected to appreciate the subtle and complex nuances of Internet security — especially when being stalked, tricked, and seduced by malicious hackers. Typical consumer computer users will tend to do insecure things. There's no practical means to prevent that, since that's what they want to do. More than anything else, personal computer users want freedom.

Windows NT and 2000 are supposedly "secure" operating systems, yet malicious Russian hackers have been breaking into those machines left and right, then stealing consumer credit card data. Has anyone ever supported the contention that WinNT and Win2000 are immune to viral infection? Has anyone ever contended that? I've never heard any such thing because we all know it's ridiculous.

We all know that no Windows system is inherently safe or secure. The truth is, that can NEVER change due to the customer-base these systems have been built to satisfy.

Yes, I'm Finally Finished

It is apparent that, for unfathomable and never articulated reasons of their own, Microsoft is DETERMINED to ship an inherently unsecurable, consumer-targeted operating system, containing the openly accessible Internet research interfaces known as Full Raw Sockets.

I think that's <u>really</u> dumb.

Steve

Purchasing Info

GRC Mail System

To GRC's Home

Tech Support

Discussions

The contents of this page are Copyright (c) 2001 by Gibson Research Corporation. SpinRite, ChromaZone, ShieldsUP, NanoProbe, the character 'Moe' (shown above), and the slogan "It's MY Computer" are registered trademarks of Gibson Research Corporation (GRC), Laguna Hills, CA, USA. GRC's web and customer **privacy policy**.

~ ~ ~

PatchWork

Internet Anti-Intrusion Patch Verification and Intrusion Evidence Scanner — for Microsoft Windows NT

(Click image to jump to PatchWork Information and Download Page)

PatchWork?

On Thursday, March 8th, 2001, the United States Federal Bureau of Investigation (FBI) disclosed details of an ongoing investigation into the organized intrusion, by Eastern European hackers, of more than forty, commercial, domestic, web sites.

See the Full FBI Announcement Here

These attacks were particularly disturbing because, in every case, the Russian hackers were simply exploiting well known and readily preventable vulnerabilities of non-updated versions of Microsoft's Windows NT web serving operating system.

Prior to the FBI's public announcement, I was contacted and asked to quickly create a tool to perform two specific functions for any Internet-connected Windows NT/2000 system:

- Rapidly scan the system's mass storage for evidence of files known to be used by hackers for system intrusion and also files implicated in the specific intrusions researched by the FBI.

- Analyze the Windows server for the presence of the specific vulnerabilities known to have been exploited by the Russian hackers.

Created in two days, and occupying just 30k bytes after being digitally signed with my secure digital signature, this new and COMPLETELY FREE utility, **PatchWork**, is ready for your use.

Pursuant to my agreement, PatchWork is being distributed by the Center for Internet Security and may be immediately and easily downloaded directly from their web site:

The Center for Internet Security

http://www.cisecurity.org

You can learn about The Center from their web site. You will find a link to their "PatchWork Page" at the top of their site's home page.

What Does PatchWork NOT do?

It is important for you to understand why PatchWork was created so that you will be able to apply it with maximum effectiveness.

PatchWork was designed from information provided by the FBI. This information was derived from their investigation into a series of directly related and coordinated intrusions into United States eCommerce and eBanking sites. Through this investigation, the FBI determined the "Attack Vectors" — the specific exploits — that were used by remote intruders to gain entry into Windows NT systems.

PATCHWORK ONLY CHECKS FOR AND ADVISES ABOUT THE PRESENCE OF THESE SPECIFIC VULNERABILITIES.

You must NOT confuse PatchWork with a general-purpose, comprehensive, patch-verification tool — it is not that. Such capability is beyond the scope of the present utility. We believe that if PatchWork gives your computer the "all clear", then that system will be hardened against the specific Internet eCommerce attacks that have been occurring with disturbing frequency. However, by no means should this PatchWork utility be used as a substitute for continuing comprehensive and proactive security measures.

It is our sincere hope that your use of this first simple "PatchWork" tool will help to highlight the need for taking your Internet security seriously. If we are able to help you solve a few security problems today — and surprise you a bit about the very real need for continuing vigilance — this tool will have been a success in our eyes.

Version History

v1.00 — Initial Release

v1.01 — Minor Cosmetic Tweak

* A paragraph was appended to the end of the file system scan reminding the user that we were only performing a simple file name match and that, consequently, any "hits" should be further researched before any conclusions are reached.

v1.10 — Function Enhancements

* We were able to obtain the file sizes of all but two known "bad" files. Therefore PatchWork was enhanced to check the suspect file size and to report both a "name match" and a file size match. This will essentially eliminate false positive reports.

* We learned that Microsoft's patching tools do not correctly verify the installed version of IIS prior to overwriting. Older versions were therefore being incorrectly "upgraded." PatchWork v1.10 now takes proactive responsibility to make sure all application patches will be safe and correct.

* PatchWork's initial MDAC recommendation was warning about unsafe registry keys even if the server's "/msdac" virtual root had been removed or renamed (which prevents the vulnerability). PatchWork now checks this before raising an alert.

* PatchWork was creating "option setting" registry entries for itself even when it was run under Windows 9x, for which the program is not intended. That behavior is now suppressed so that there are no registry side effects from running under Win9x.

PatchWork's Digital Signature

Since PatchWork is being downloaded from a server other than mine, and since copies will probably be passed around the Internet quite a bit, users need to have some way to verify that the original program has not been tampered with in any way. For that reason I have "digitally signed" the original PatchWork program with my personal, non-spoofable, cryptographic signature.

For instruction on checking the validity of the file's signature, click the link below:

How to Verify a Digital Signature

PatchWork Meets the Press

* **InternetNews — Firm to Air Online Security Tool for FBI**
  March 8th, 2001, by Carol King

* **ComputerWorld — FBI investigating widespread Web site break-ins by crime groups**
  March 8th, 2001, by Dan Verton

* **PC World.com — Hacker Wave Combines Break-Ins With Extortion**
  March 8th, 2001, by Jennifer O'Neill, Medill News Service

- **InfoWorld — [FBI warns e-commerce sites](#)**
  March 9th, 2001, by Margret Johnston and Joris Evers

- **USA Today — [FBI warns of organized hacker attacks](#)**
  March 9th, 2001, Washington / Associated Press

- **Washington Post — [Hackers Feast On Complacency](#)**
  March 9th, 2001, by Ariana Eunjung Cha

- **SF Gate News — [European Hackers Plunder U.S. Firms; FBI says 40 victims in 20 states were hit](#)**
  March 9th, 2001, by Bill Wallace

- **InfoWorld — [Security center issues anti-hacker tool](#)**
  March 12th, 2001, by Margret Johnston

- **Internet Week — [Failure To Keep Security Updated Can Cost More Than Customers](#)**
  March 15th, 2001, editorial by Wayne Rash

I sincerely hope you find **PatchWork** to be a useful and effective utility to aid in taking the first steps toward establishing proactive and ongoing management of your enterprise's Internet-connected server security.

Wishing you safe and secure use of the Internet!

Steve

[Purchasing Info](#)   [GRC Mail System](#)   [To GRC's Home](#)   [Tech Support](#)   [Discussions](#)