Packet Level Normalisation

This paper proposes that any Signature Based Passive Network Intrusion Detection (NID) deployment is incomplete without an 'In-line' 'Packet Level Normaliser' [1]. A number of published papers will be selectively reviewed, assessing their contribution to the development of this field. Focusing on the Network Layer, a 'walkthrough' of the IP protocol will be followed by a Lab where the Normaliser 'norm' [2] will be employed to illustrate core concepts. Packets will be manufactured using 'NetDuDe' [3] and 'Fragroute' [4]...

Part 1 Packet Level Normalisation

Abstract
This paper proposes that any Signature Based Passive Network Intrusion Detection (NID) deployment is incomplete without an 'In-line' 'Packet Level Normaliser' [1]. A number of published papers will be selectively reviewed, assessing their contribution to the development of this field. Focusing on the Network Layer, a 'walkthrough' of the IP protocol will be followed by a Lab where the Normaliser 'norm' [2] will be employed to illustrate core concepts. Packets will be manufactured using 'NetDuDe' [3] and 'Fragroute' [4]. The output will be in 'tcpdump' [5] format. The paper culminates with a brief review of current normaliser technology.

Required Knowledge
This paper assumes a working knowledge of the TCP/IP protocol. A sound understanding of network security issues and security product deployments with specific emphasis on Signature Based Passive NID technology. Familiarity with Linux and also tcpdump 'hex' output will be an advantage.

Terminology
The term 'Normalisation' first appears in 'Network Intrusion Detection: Evasion, Traffic Normalisation, and End-to-End Protocol Semantics' [1]. The term 'Scrubbing' appears in 'Transport and Application Protocol Scrubbing' [6]. These terms are interchangeable, this paper will use the term Normalisation throughout. The terms Insertion and Evasion first appear in 'Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection' [7]. The term 'Ambiguous', which Ptacek and Newsham define as "if an important conclusion about a packet cannot be made without a secondary source of information" [7].

What is NOT covered in this paper
To describe all of the TCP/IP ambiguities and exploit techniques available is unrealistic for a paper of this length, as is a comprehensive discussion of the white papers reviewed. Current Normalisation development is split into two distinct areas, Packet Level (Network and Transport Layer) and Application Level (Application Layer). The topic of this paper is Packet Level Normalisation. This refers to analysis of IP, TCP, UDP, and ICMP header information, Application Level normalisation examines Telnet, DNS, HTTP etc. header (and optionally payload) information. The reader wishing to know more about this topic is directed to a white paper on the subject by Benjamin Sapiro at the SANS Reading Room [8]. This paper will concentrate on IP exclusively.

Why Normalisation?
In a perfect world every process that wished to use the TCP/IP Protocol to communicate, would use the same TCP/IP implementation, which would naturally have been written under strict adherence to all relevant guidelines.

Consequently, the requirement for Packet Level Normalisation would vanish in a puff of logic. However, there is not one implementation, but many, (not one vendor but many) and each can behave uniquely under identical stimulation. Herein lies the ambiguity, necessitating the normalisation. From an economics perspective this is no more than the usual consequence of market forces. From a Security Professional's perspective this is an enormous problem as it has lead to the development of a multitude of 'cracker toolz' which gleefully exploit these behaviours.

Behaviours? What Behaviours?

At SIGCOMM 96 Vern Paxon published what was to be the first of three papers concerning the behavioural characteristics of the Internet at large. This was to be the first study of the behaviour of the TCP/IP protocols in the wild [9]. What shocked his audience was the existence of widespread variations in routing characteristics. The next paper published at SIGCOMM 97 [10] investigates these phenomena further, using the same data but now focusing on the packet dynamics. Again the conclusions were not comforting; he found a wide range of behaviour including regular violation of TCP protocol. The third paper, [11] delivered at the same conference, presents a discussion of the development and implementation of 'tcpanaly' (the analysis tool used for all three papers), in Paxon's words, "a tool for automatically analysing a TCP implementation's behaviour". Here, he turns his attention to analysis of specific vendor's implementations of TCP/IP. The analysis performed on initially 8 and ultimately 11 major TCP implementations (Table 1), displayed a strikingly large range of behaviours amongst them.

Implementation	**#Sender**	**#Receiver**	**Lineage**
BSDI 1.1, 2.0, 2.1a	3,394	3,650	Reno
DEC OSF/1 1.3a, 2.0, 3.0, 3.2	1,874	1,897	Reno
HP/UX 9.05, 10.00	1,456	1,553	Reno
IRIX 4.0, 5.1, 5.2, 5 .3, 6.2A	3,046	3,119	Reno
Linux 1.0	23	26	Indep.
NetBSD 1.0	122	121	Reno
Solaris 2.3, 2.4	5,705	5,535	Indep.
SunOS 4.1	4,353	4,156	Tahoe
Linux 2.0.30, 2.1.36	46	19	Indep.
Trumpet/Winsock 2.0b, 3.0c	7	6	Indep.
Windows 95, Windows NT	8	8	Indep.
Total	**20,034**	**20,043**	

Table 1: TCP implementations studied.
Source: Handley et al [1]

Paxon continues; "'tcpanaly' has coded within it knowledge of a large number of TCP implementations. Using this, it can detect whether a given trace appears consistent with a given implementation. Could this possibly be a description of TCP/IP Stack Fingerprinting developed by Paxon during the data collection period of 94/95? The Fyodor classic, "Remote OS detection via TCP/IP Stack Fingerprinting" article of Oct 98 [12] (apparently accepted these days as common knowledge), makes no mention of this research. Later, Paxon discusses the characteristics of the BSD TCP 'Tahoe' release in 88 and the later 'Reno' release of 90, [13] and surmises that while almost all

other implementations studied derive their behaviour from these, they each have their own particular textures. Helpfully, he goes on to outline each of the specific variations or "bugs" as he cheerfully describes them. To the perceptive reader this paper was to reveal a positive cornucopia of ambiguity in the TCP/IP protocols, ambiguity that was ripe for exploitation.

Enter two very perceptive blokes

Thomas Ptacek and Timothy Newsham published their paper "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection" [7], early in the following year, Jan 98. In Ptacek's own statement, this paper "broke every network IDS on the market" [14]. Essentially the paper consists of two premises. The first states there is not enough information on the wire for a NID to perceive what is happening at an end system. The second advises that as a NID 'fails open' it can be compromised with no affect to the rest of the network. We are swiftly presented with the three now famous exploit concepts, Insertion, Evasion, and Denial of Service, which in the hands of Ptacek and Newsham circumvented the then top four most popular NIDS on the market. Denial of Service, as they readily admit, is not their invention per se, but they do outline a number of attacks targeted specifically at NID functionality. This made them very popular with everyone, except the railroads and the banks.

Subsequent years saw many embarrassed NIDS Manufacturers using the Ptacek and Newsham paper as litmus in 'developing' their product range. There was also the inevitable explosion of what can only be euphemistically describe as "test code" being made available to the public at large. This code was entirely based on Ptacek and Newsham's published exploits. The most widely known would probably be 'fragrouter' [15] early 99, from Dug Song, maker of 'dSniff', and Horizon, with his Phrack Article Dec 1998, "Defeating Sniffers and Intrusion Detection Systems" [16]. Finally, proving that you can never have too much of a good thing, and, as the Application layer was feeling a bit left out of the fun, Rain Forrest Puppy publishes "A look at whisker's anti-IDS tactics" in mid 99 [17]. Expanding upon the well-known CGI exploit "phf" (used by Ptacek and Newsham as their test attack) with an armada of obfuscation techniques designed to bypass a NID, all targeted at http.

The Development of Normalization

The next two papers are the products of simultaneous research during the period of 1999-2000 and it is here that the concepts and constructions of normalizer technology can be seen to take place. As previously stated, they contain a great deal of content that cannot possibly be covered in a paper of this length, including the design goals and performance specifications of their tools. They also contain a wealth of Transport Layer information. For instance, the management of TCP state transition, including strategies for defeating stealth scanning using the IP ID field [18]. These papers are highly recommended as bedtime reading companions to the usual diet of RFC's.

'Transport and Application Protocol Scrubbing', March 2000 [6]

The first published response to Ptacek and Newsham and well worth the wait. As the title suggests there are two independent normalisation systems prescribed herein. The paper starts with a succinct description of a normaliser's design criteria, "to convert ambiguous network flows into well-behaved flows that are unequivocally interpreted by all downstream endpoints" [6]. Thereby, they say, eliminating 'Insertion and Evasion' attacks at a stroke. By definition, this locates the Normaliser at the protected network boundary (Figure 1), the proverbial 'bump on the wire' just like a Firewall (or Gateway IDS). All this neatly implemented in FreeBSD at kernel level, what more could one desire?

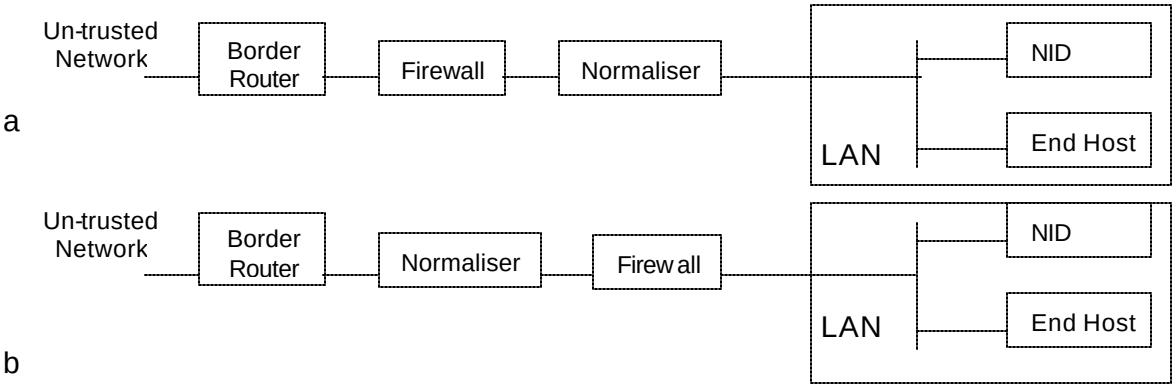
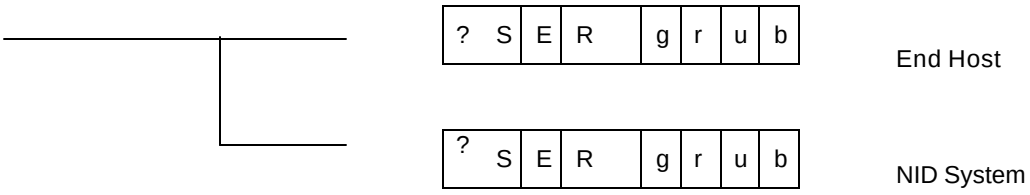


Figure 1. Illustrating possible configurations of a Normaliser

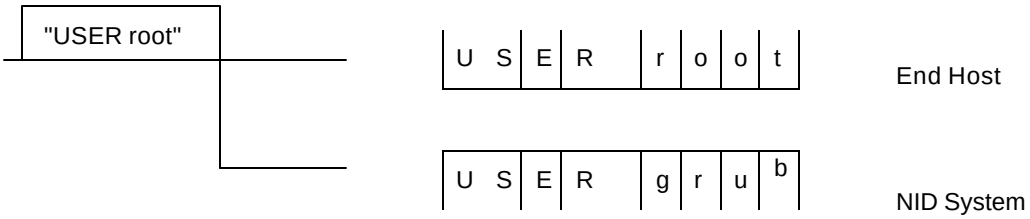
There is lively debate, as to which side of the Firewall the normaliser should be placed, 'assuming' that it is a separate host system (Fig 1a and 1b). Another possibility, suggested by Malan et al [6], is that it could be added internally to a Firewall System, and would function after the Firewall has given access to the data stream. This presumably would be once the packet had been checked against the filter rule set, indicating a possible performance advantage at this location. Since version 3.0 Dec 2001, OpenBSD's [19] statefull firewall, 'pf' (packet filter)[20], substantiates this theory by containing a rudimentary normaliser called 'scrub' [20]. The FAQ sample file shows 'pf' loading 'scrub' before NAT and its filter set, but after it defines its networks and accepted protocols. However, that may not be an indicator of its ultimate location along the data flow within the kernel. Ultimately, positioned 'in-line' with the dataflow enables the normaliser to have a 'fail-closed' facility.

Utilising as an example of transport layer ambiguity, Malan et al [6] then present the Ptacek and Newsham fragmentation overlap attack:

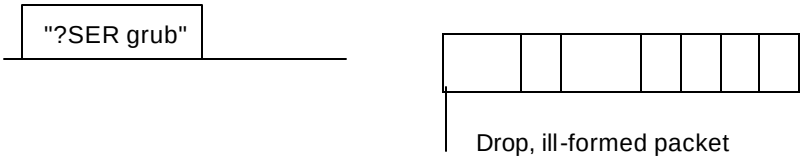
An attacker is trying to log on to an end system as root, whilst fooling the NID that it's connecting as a regular user. As the end system and the NID both reconstruct overlapping TCP sequences differently the following penetration is successful.



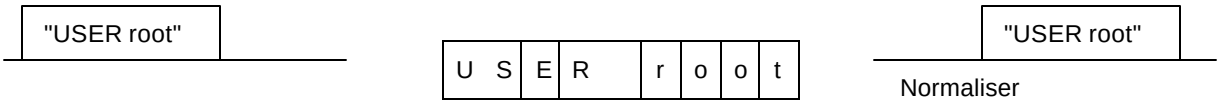
(a) Host and NID system after attacker sends a hole.



(b) Attacker filling in the whole and confusing the NID system.



(c) Normaliser enforces single interpretation.



(d) Attacker filling in the whole and sending new data.

Figure 2. Example of forward / backward fragmentation overlap showing the differences between a NID - End Host interpretation and a normalized data stream.
Source: Malan et al [6]

1a Firstly the attacker sends a data sequence with a gap at the start, TCP requires that this is filled before delivery to the application layer and so waits [21].

1b Next the Attacker fills the gap but also adds a different user name (note identical character length). Herein lies the ambiguity, the facilitator of a successful attack. According to page 29 RFC 791 "In the case that two or more fragments contain the same data either identically or through partial overlap, this procedure will use the most recently arrived copy in the data buffer and datagram delivered"[22]. However, some systems keep the old set (Table 2), and in this example, the end system uses 'Forward Overlapping' whilst the NID uses 'Backward Overlapping'.

Operating System	Overlap Behaviour
Windows NT	Always Favours Old Data
4.4 BSD	Favours New Data For Forward Overlap
Linux	Favours New Data For Forward Overlap
Solaris 2.6	Always Favours Old Data
HP-UX 9.01	Favours New Data For Forward Overlap
IRIX 5.3	Favours New Data For Forward Overlap

Table 2 IP fragment overlap behaviour for various OS's,
Source: Ptacek Newsham [7]

The paper continues to outline a host of Transport Layer ambiguities including malformed headers, TTL attacks and TCP creation / 'tear-down' problems. Essentially reasserting that with such a large number of ambiguities inherent in the end systems the NID, no matter how powerful, will not be able to address all possible permutations. The Normaliser however, can deliver the uniformity that the NID needs for confident flow reconstruction and end-host prediction behaviour. Fig 1c and 1d show how an intervening Normaliser, by enforcing a single interpretation of the attackers data flow, eliminates the ambiguity for all downstream systems. Consequently neutralising the attack.

Three months later they publish another paper " Defeating TCP/IP Stack Fingerprinting" [23] which basically outlines the steps necessary to normalise egress traffic thereby preventing active or passive fingerprinting. All this research culminated in late 2000 with the Lighthouse Project [24]. A logical manifestation of their previous work, this project deployed a large-scale network monitoring system including scrubbing, anti-finger printing systems and the additional normalization of routing protocols like BGP and OSPF. The system also included an embryonic anti DoS system.

In 2001 Vern Paxon with Mark Handly and Christian Kreibich publish 'Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics.' [1]. Explaining the idea of normalization (protocol scrubbing) was "simultaneously" invented by themselves and 'Malan, et al [6] back in 1999/2000, they state:

> The key distinction between our work and TCP/IP Scrubbers is that we attempt to develop a systematic approach to all potential normalisations and we emphasize the implications of various normalisation with regard to maintaining or eroding the end-to-end transport semantics defined be the TCP/IP protocol suit. [1]

They also consider the issue of successful DoS attacks and subsequent recovery (cold-start) at the normaliser itself. The strength of this paper is its methodology, which sets out to identify and analyse all possible normalisations. In order to accomplish this feat, a journey through the entire protocol header is prescribed. At each field/header element, the possible range of values are contemplated, their semantics and what, if anything, an attacker could exploit. This list develops the basic functionality of the normaliser, for example to drop or modify a packet in order to nullify said

attack. Finally, this response is analysed with respect to any effect it may have had on end-to-end protocol semantics, the results are then re-submitted into the analysis, for example, removing the Do Not Fragment (DF) flag would break 'path MTU discovery' [25]. The paper uses IP v4 as an example and directs our attention to 'norm-TR-2001'[26] for additional data on ICMP TCP and UDP. Some of the conclusions appear extremely unlikely as plausible attack vectors, nevertheless, the philosophy of, 'normalise everything because you never know…', must make sound policy.

IP Header 'Walkthrough'

The IP protocol will now be examined in detail. This 'walkthrough', based on the Handley et al [1] paper, includes a number of additional exploit scenarios, not all of which are neutralised by 'norm', but should be addressed during the design of any production normaliser.

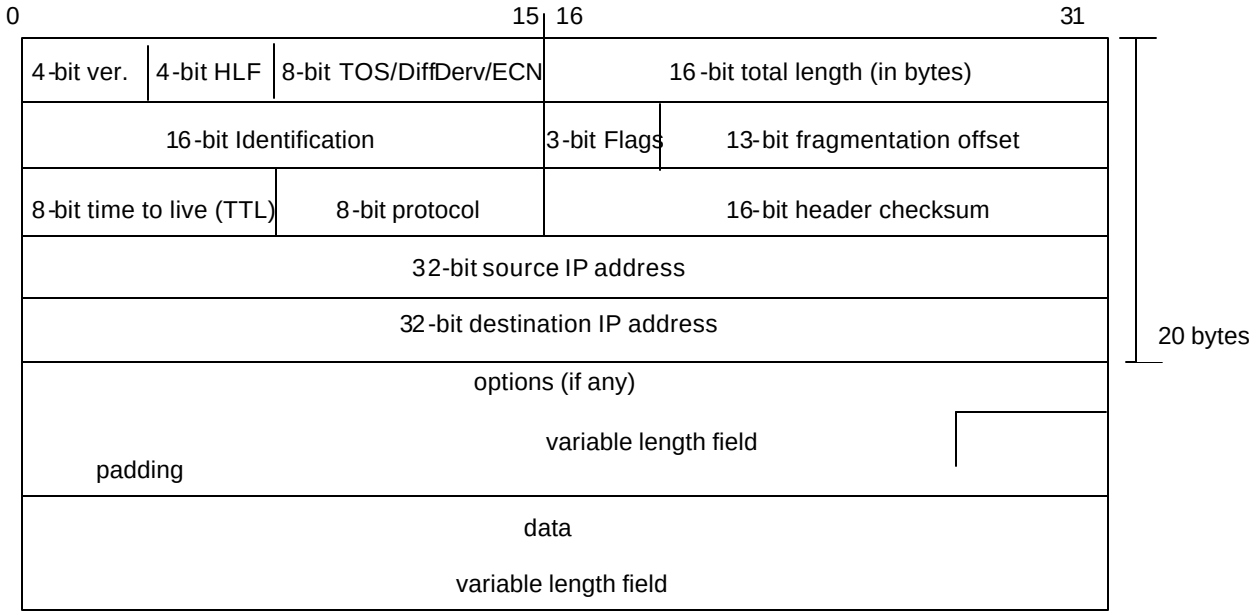


Figure 2, Fields in an IP header. Source: Stevens TCP/IP Volume 1 [21]

Field	Version
Discussion	Only pass packets with an IP version that the NIDS understands
Action Effect on Semantics	Drop all non IPv4 packets None, as the packet is already ill formed
Field	Header Length
Discussion	It is possible to send packets with incorrect header length that may be accepted by the end system, but could also be discarded by intermediate systems e.g. routers, leaving the NID unsure whether the packet was processed or not
Action	Action: Drop if header length shows less than 20 bytes (min IP with no options). Drop if it shows more than the packet length (also refer to total length field)
Semantics	None as the packet is already ill formed

Field	Type of Service (TOS)/Differentiated Services(Diff Serv)/Explicit Congestion Notification (ECN)
Discussion	A complicated field, indicating different information to different devices: 1 It is assumed that TOS is now redundant 2 Manipulating Diffserv bits could cause them to be dropped at an internal Diffserv enabled router 3 ECN is possibly too new for certain devices and so their processing of this field is unpredictable 4 Fingerprinting techniques exist to manipulate the bits in this field to measure the end host response [12]
Action	1. None 2. If the entire site does not use Diff serve then zero the bits. 3. Clear all bits unless the connection previously negotiated use of ECN, this is specific to a cold start of the normaliser. Optionally remove attempts to negotiate ECN 4. Clear all bits
Semantics	2 None 3 If Diffserv is not being used then the bits should be zero anyway, if it does then Diffserv is broken 4 None, for the systems that have not negotiated ECN. Removing will circumvent any ECN benefit 5 None
Field	Total Length.
Discussion	When the total length here does not match the total length indicated by the link layer, then we cannot predict what length some systems will choose, or if they decide to discard the packet. Also if the total length field is smaller than the header length field a similar problem can arise
Action	Drop packet, if total length is greater than Link layer length. Trim if total length is less than the link layer length
Semantics	None, only ill-formed packets are dropped
Field	IP Identifier
Discussion	To maintain datagram integrity upon fragmentation, each fragment uses the same ID. Many systems advance this number in a predictable manor. Giving rise to stealth scanning [20]
Action	Handley et al [1] suggests encryption of the ID's. On the other hand, to fool the scanner, 'ACK' keep-alive packets could be used, this would however use extra bandwidth (neither of which are implemented in 'norm'). An alternative solution could be the use of random generated numbers, aka. OpenBSD's ID sequence numbers
Semantics	None
Field	Must be Zero
Discussion	If the bit is set then it is uncertain how an end or intermediate systems will react
Action	Set bit to zero
Semantics	None, bit is required to be zero
Field	Do Not Fragment. (DF)
Discussion	1. Can be used to selectively drop packets at MTU boundary therefore preventing NID from knowing the state of the end system. (the NID could very well be on a different MTU segment) 2. Packets arriving with the DF flag set and a non-zero fragmentation offset are illegal, but again, how an end system handles this is an unknown

Action	1. Clear DF on incoming packets 2. Discard the packet
Semantics	1. Breaks "MTU path discovery" (ICMP type 3 code 4) [25] and creates additional processing of DF enabled packets for fragmentation and re-assembly 2. None, ill-formed packet
Note	Path MTU discovery, proposed in 1990, was a mechanism for optimising bandwidth. The original IP specifications required that for any destination not on the same subnet, the datagram size should be the lesser of the first hop MTU (Local Network) or 576. Realistically, the Internet now runs on Ethernet (MTU 1500) so the original design goals for path discovery (maximising bandwidth) are somewhat mute.
Field	More Fragments (MF) Fragment Offset. Both fields are interpreted together and so are treated as such here
Discussion	This is one that Ptacek and Newsham used to break all those NIDS. The machines they tested even failed to reassemble 'in-order' fragments, never mind duplicates or overlaps! (Fig 2.) 1. Ambiguity can arise when the NID sees two overlapping fragments with differing data, and is subsequently unsure how the end system will reassemble them. Or the end system may 'timeout' partially received fragments differently to the NID. (ICMP Type 11 Code1) [27] 2. Ping of death fragments, a fragment offset greater than 65535 are illegal and can crash some old systems
Action	2. Reassemble all incoming fragments in the normaliser, rather than forwarding them, then if the MTU requires, re-fragment for transit to the end system 3. Drop Packet
Semantics	1. No effect as reassembly is an acceptable operation of routers 2. None, packet is ill formed
Note	Fragmentation reassembly is a standard feature for a normaliser as all upper layer protocols must be delivered to their respective processing modules (ICMP TCP UDP) before retransmittal to the end host. As this is a state based operation there must be a facility within the normaliser which can counter 'stateholding' based attacks, (Denial of Service). For example Jolt2 [28], sends ICMP echo requests as fragments with the same fragment ID but with duplicated non 0 fragment offsets to fill the data buffer. One potential solution, may be the employment of a user configurable reassembly timer. The original recommendation for a reassembly timer is 15 seconds increasing to the maximum value 4.25 minutes [22]. This value can increases if the TTL of an arriving fragment is greater than the current value. (ICMP Type 11 Code1) [27]
Field	Time To Live TTL
Discussion	An attacker can use the TTL field to deceive the NID into mistaking the state of the end system (in a similar manor to manipulating the DF flag used with path MTU discovery). This is classed as an insertion attack. Using very low TTL's certain packets expire after the NID but before reaching the end system. Handley et al [1] give three solutions to this issue, two of which require normaliser to rely on other systems which, under the circumstances, is an unacceptable risk
Action	Set the normaliser to upgrade any incoming low TTL to a site-specific minimum, for example, the maximum number of hops across the site
Semantics	This is probably the most contentious normalisation (possibly why Handley et al propose the aforementioned risky solutions) as it breaks a number of protocols and programs: 1. It could easily generate routing loops as it is essentially undermining the original concept of the TTL field.

	2. It will break the 'traceroute' program, which utilises low TTL's 3. Degrade performance on Multicasting as all systems inside the normaliser would appear to be at exactly the same distance (hop count)
Field	Protocol
Discussion	Indicating which upper layer protocol is encapsulated. It's generally a firewall's duty to block specific protocols, so this should be out of the scope of the normaliser, however, it could be configured to block any protocols unknown to the NID. This would also block scans using obscure protocols to map hosts by obtaining ICMP type3 code 2 replies [12]
Action	Enforce specific user defined protocols.
Semantics	None
Note	The contents of this field direct the normaliser's IP module to deliver the packet to the specific upper layer module, for example TCP or UDP
Field	IP Header Checksum
Discussion	An 'older' end system may receive and accept packets with incorrect checksums [7]. This is highly unlikely, considering the checksum is recomputed and verified at each point the header is processed. For example, a router will recalculate the checksum when either decreasing the TTL field or fragmenting a packet at a MTU boundary or changing the IP Options (timestamp etc) before forwarding the packet
Action	Drop packet
Semantics	None, packet is ill formed
Field	Source Address
Discussion	If invalid like a 127.0.0.1 or a 0.0.0.0 or a class D or E, the NID may not know what the end host is likely to do with it. Potentially the normaliser could also drop internal addresses, which shouldn't be coming into the site (source routing perhaps), one would however wish to see a these logged for additional investigation
Action	Drop the packet
Semantics	None packet is ill formed
Field	Destination Address
Discussion	May also cause unexpected behaviour at end system if invalid, like local broadcast addresses (Smurf attacks)[29] or Class E addresses
Action	Drop the packet
Semantics	None, destination is illegal
Field	IP Options
Discussion	When an IP packet is fragmented, certain options should be carried on all fragments and others on just the first. For example 'Basic' and 'Extended Security' are to be carried on all fragments but 'Record Route' on the first fragment only. Unless the NID knows how each end system will react to incorrectly presented options there is ambiguity. Strict or Loose Source Routing can also be used to circumvent the NID segment, and using man-in-the-middle attacks can spoof a trust relationship by re-routing traffic back to it instead of the trusted host. The use of these options is actively discouraged and some routers now have a facility to disable them. The end system could also be independently configured to drop Source routed packets creating further ambiguity. Finally, the NID would also have to know how each end system rejected malformed IP Option packets and perform similar routines
Action	Remove IP Options from incoming packets

Semantics	At the end-to-end level there should be none, as IP Options should not effect the upper layers. It will break certain archaic diagnostic facilities like record route and timestamp option, and the now historic U.S. DoD. Security Options
Field	Padding
Discussion	Required to be ignored by the end system, a possible covert channel facility much like the famous Loki [30] in ICMP
Action	Zero out all the padding bits
Semantics	None, the field is explicitly ignored

The Lab

Unfortunately, not all the tools used herein are available for every platform, NetDuDe and 'norm' are currently Linux only, and while tcpdump and fragroute do have Windows counterparts this Lab will be exclusively Linux.

Netdude [3], 'norm' [2] and 'fragroute' [4] are employed in a Lab environment to create examples of packet normalization. The Lab will have 'norm' read an input_file (un-trusted external network) and write an output_file (trusted/protected network). This data was created using two Mandrake9.0 distro. Linux boxes with the following installed prerequisites, gtk+-1.2.10 [32], glib-1.2.10 [33], tcpdump-3.7.2 [5], libpcap-0.7.2. [34]. Connected via a non-switched Ethernet hub.

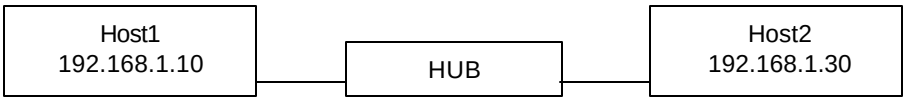


Figure3. LAB Network Diagram

Using `'host1'` to create an ICMP echo request with an IP timestamp option

```
$ ping -T tsandaddr 192.168.1.30 -c 1
```

Using `'host2'` tcpdump captured the complete packet with the following command

```
$tcpdump -i eth0 -w icmp -s 134
```

the original datagram

```
23:52:50.297651 192.168.1.10 > 192.168.1.30: icmp: echo request (DF)
                         4e00 0078 0000 4000 4001 471f c0a8 010a
                         c0a8 011e 4424 0d01 c0a8 010a 0104 5311
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0800 bbbe 0c07 0100
                         a46c c43e d28b 0900 0809 0a0b 0c0d 0e0f
                         1011 1213 1415
```

The following procedure will employ NetDuDe to manipulate the datagram then pass it through norm and compare the results.

As it was important to manufacture a datagram with IP Options present (in order to display norm's management of such) it was therefore necessary to increase the snaplength to capture it in its entirety. The default tcpdump snap

length is 68 bytes, including the 14 bytes of Ethernet header, leaving 54 bytes for the IP header and payload (encapsulated ICMP header and data). The packet above is 120 bytes of IP, (Total Length field in Bold), adding the 14 byte Ethernet header results in a snap length of 134.

Installation of norm and NetDuDe:

installation of netdude-0.3.3 is the usual :- `'./configure', 'make', 'su', 'make install'`

installation of norm-0.2.0 is the usual :- `'./configure', 'make', 'su', 'make install'`

Then edit the configure template file found in `'/usr/local/share/norm/configfile'` and modify the following:

`'debuglevel 5'`	Information log level, giving additional norm messages.
`'trusted 192.168.1.0/24'`	The 'trusted' and 'Interface' settings are specific to this lab.
`'interface eth0'`	Any packets coming from the trusted address space are not processed.

Norms first run copies the `'configfile'` to user's home directory `'~/.norm'`. A review of the file reveals the normalization switches, some of which have configurable variables like the TTL field.

Procedure: norm and NetDuDe

The following examples have been chosen to provide a broad illustration of a normaliser's potential. This process follows the End-to-End [1] paper's test procedure.

The trace was run through 'norm' without any modifications to baseline the equipment.

```
$ norm -ti input_file -to output_file
```

This resulted in no normalizations and no output file, norm has recognized the sending host is on a "trusted" network. Therefore, in order to present the trace as potentially hostile, the source IP address is changed using NetDuDe, the following command launches the GUI.

```
$ netdude input_file
```

The trace is first highlighted, then IP is selected from the tab, the source IP field is clicked allowing it to be changed to 194.168.1.10. The checksum field changes to red showing it's now incorrect, using the plug-in "checksum fix" from the 'pull-down' menu corrects this. The packet is then saved and presented to norm again. A positive result is the outcome. The normalised trace is copied to output_file with the following comments to standard output.

```
Did 2 norms, 2 IP, 0 TCP, 0 UDP, 0 ICMP.
```

Studying the output file using tcpdump reveals the normalisations;

```
$tcpdump -t output_file -nxx

04:16:33.707753 194.168.1.10 > 192.168.1.30: icmp: echo request
                        4e00 0078 0000 0000 ff01 597e c2a8 010a
                        c0a8 011e 4424 0d01 c0a8 010a 01f5 bec0
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0800 cf6c e206 0100
                        72aa c43e 1ea0 0500 0809 0a0b 0c0d 0e0f
                        1011 1213 1415 1617 1819 1a1b 1c1d 1e1f
                        2021 2223 2425 2627 2829 2a2b 2c2d 2e2f
                        3031 3233 3435 3637
```

The Do Not Fragment (DF) and the Time To Live (TTL) fields (respectively in bold) have been normalized. On comparison with the first trace, we find the DF bit is reset to zero and the ttl up to 255, as per the config file settings. The IP option field should have been removed but is still there. A review of norm's config file shows that the IP Options normalisation switch in the correct position, however, additional investigation reveals that the code to perform the normalisation was " …planned, but either not yet implemented or not yet tested at the time of writing" [1]. An object lesson to read the small print!

Nevertheless, running the file through norm again, it completes without any further comment proving that a second pass does not modify the trace further.

Norm has just performed the following, received a pseudo incoming packet, corrected it as per the config file settings and wrote it to output. The next test will require norm to recognize an ill-formed packet and drop it. This will be achieved by setting the Fragmentation Offset field to a value greater than 0 while the DF flag is also set.

```
$ netdude input_file
```

The trace is again highlighted, IP tab selected and the DF field chosen (underlined), the Fragment offset field is then selected (bold) and a value of 0x05 is inserted. The checksum is corrected and the packet is saved. Note: multiplying the offset field by the requisite 8 bytes gives an offset of 40 bytes (bold underlined).

```
04:16:33.707753 194.168.1.10 > 192.168.1.30: (frag 0:64@40)
                        4e00 0078 0000 4005 ff01 1979 c2a8 010a
                        c0a8 011e 4424 0d01 c0a8 010a 01f5 bec0
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0800 cf6c e206 0100
                        72aa c43e 1ea0 0500 0809 0a0b 0c0d 0e0f
                        1011 1213 1415 1617 1819 1a1b 1c1d 1e1f
                        2021 2223 2425 2627 2829 2a2b 2c2d 2e2f
                        3031 3233 3435 3637

$ norm -ti input_file -to output_file
```

The norm presents the following helpful output, it 'drops' the packet and the output file is empty.

```
IP:    fragment --  offset(40)  DF(*)  MF( )
IP:    DF set on fragment -- dropping.
Did 1 norms, 1 IP, 0 TCP, 0 UDP, 0 ICMP.
```

Finally, fragment reassembly, the trick that "broke" all those IDS's back in 98. The current version of 'NetDuDe' only splits the packet into two fragments which is hardly challenging, so for this last Lab Dug Songs 'Fragroute1.2' [4] will be used. This program is often confused with the similarly named "fragrouter" released by the same author in 1999. The following files must be installed prior to Fragroute: libevent, [35] and libdnet, [36] and lastly libpcap [34], (which for the purpose of this Lab has been previously installed).

Installation of Fragroute:

The usual procedure; `./configure`, `make`, `su`, `make install`.

The default settings will be used, and glancing at the config file, `/usr/local/etc/fragroute.conf`, shows fragroute's default setup will take the outgoing data stream (in this instance a single packet) and perform the following operations:

1. Segment the packet into 24 byte fragments
2. Randomise the delivery order
3. Add duplicate fragments
4. Insert anomalous packet/s

Procedure: Fragroute

As fragroute is designed to operate within a 'live' data stream it will not read tcpdump files, so a new packet will be created. First, fragroute is launched with the desired IP address as its target filter.

```
$ fragroute 192.168.1.30
```

The packet is created using the original command.

```
$ ping -T tsandaddr 192.168.1.30 -c 1

14:52:23.326715 194.168.1.10 > 192.168.1.30: icmp: echo request (DF)
                         4e00 0078 0000 4000 4001 8f83 c2a8 010a
                         c0a8 011e 4424 0d01 c0a8 011e 0100 089d
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0800 733e 0b25 0100
                         0bec d23e a96e 0600 0809 0a0b 0c0d 0e0f
                         1011 1213 1415 1617 1819 1a1b 1c1d 1e1f
                         2021 2223 2425 2627 2829 2a2b 2c2d 2e2f
                         3031 3233 3435 3637
```

Fragroute intercepts it and, after consultation with it's configuration file, creates the following packets. (fragroute message to standard output outlined below).

```
194.168.1.10 > 192.168.1.30: (frag 0:24@24+) [delay 0.001 ms]
194.168.1.10 > 192.168.1.30: icmp: type 76 code 103 (frag 0:24@0+) [delay
0.001 ms]
```

```
194.168.1.10 > 192.168.1.30: (frag 0:24@24+)
194.168.1.10 > 192.168.1.30: icmp: type 8 code 0 (frag 0:24@0+)
194.168.1.10 > 192.168.1.30: (frag 0:16@48)
194.168.1.10 > 192.168.1.30: (frag 0:16@48) [delay 0.001 ms]
```

The datagram is fragmented into 3 packets. The first, line 4, includes the upper layer protolol, the second fragment, line 3, with the third and last fragment at line 5. These packets are copied and randomised as per the configuration instructions and an anomalous packet injected into the data stream, line 2. Notice the delay configured on packets 2 and 6.

After collecting with tcpdump, (extending the snaplen to 138). The files have been received by the end system in the following order.

```
$ tcpdump -r input_file -n

14:38:55.399185 192.168.1.10 > 192.168.1.30: (frag 0:24@24+)
14:38:55.399272 192.168.1.10 > 192.168.1.30: icmp: echo request (frag
0:24@0+)
14:38:55.399344 192.168.1.10 > 192.168.1.30: (frag 0:16@48)
14:38:55.399446 192.168.1.10 > 192.168.1.30: (frag 0:24@24+)
14:38:55.399552 192.168.1.10 > 192.168.1.30: icmp: type-#76 (frag 0:24@0+)
14:38:55.399623 192.168.1.10 > 192.168.1.30: (frag 0:16@48)
```

The packets are then run through norm and the following information is included in the screen output.

```
[1]----------------------------------------------------------------
ETH:   received on if (null) (Ethernet) 98 bytes
ETH:   Handling Ethernet-II frame
IP:    fragment --  offset(24)  DF( )  MF(*)
IP:    mf:1 offset: 24 len 24
Larger hash size: 1 items.
[2]----------------------------------------------------------------
ETH:   received on if (null) (Ethernet) 98 bytes
ETH:   Handling Ethernet-II frame
IP:    fragment --  offset(0)  DF( )  MF(*)
IP:    mf:1 offset: 0 len 24
IP:    frag 0 off: 0 len:24 tlen:24
IP:    frag 1 off: 24 len:24 tlen:48
IP:    last frag has MF set
[3]----------------------------------------------------------------
ETH:   received on if (null) (Ethernet) 90 bytes
ETH:   Handling Ethernet-II frame
IP:    fragment --  offset(48)  DF( )  MF( )
IP:    mf:0 offset: 48 len 16
IP:    frag 0 off: 0 len:24 tlen:24
IP:    frag 1 off: 24 len:24 tlen:48
IP:    frag 2 off: 48 len:16 tlen:64
IP:    got all fragments, len=64
[NORM]: Injecting reassembled packet.
ICMP:  Echo request.
[4]----------------------------------------------------------------
ETH:   received on if (null) (Ethernet) 98 bytes
ETH:   Handling Ethernet-II frame
IP:    fragment --  offset(24)  DF( )  MF(*)
IP:    mf:1 offset: 24 len 24
[5]----------------------------------------------------------------
ETH:   received on if (null) (Ethernet) 98 bytes
ETH:   Handling Ethernet-II frame
IP:    fragment --  offset(0)  DF( )  MF(*)
IP:    mf:1 offset: 0 len 24
IP:    frag 0 off: 0 len:24 tlen:24
```

```
IP:    frag 1 off: 24 len:24 tlen:48
IP:    last frag has MF set
[6]-------------------------------------------------------------
ETH:   received on if (null) (Ethernet) 90 bytes
ETH:   Handling Ethernet-II frame
IP:    fragment --  offset(48)  DF( )  MF( )
IP:    mf:0 offset: 48 len 16
IP:    frag 0 off: 0 len:24 tlen:24
IP:    frag 1 off: 24 len:24 tlen:48
IP:    frag 2 off: 48 len:16 tlen:64
IP:    got all fragments, len=64
[NORM]: Injecting reassembled packet.
ICMP:  Packet has invalid checksum -- dropping.
Did 1 norms, 0 IP, 0 TCP, 0 UDP, 1 ICMP.
```

The IP normalisation was the ttl field, the rest of the packets once reassembled are discarded, norm presents the final assembled packet to the output_file.

```
14:52:23.326715 194.168.1.10 > 192.168.1.30: icmp: echo request
                         4e00 0078 0000 0000 ff01 1083 c2a8 010a
                         c0a8 011e 4424 0d01 c0a8 011e 0100 089d
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0800 733e 0b25 0100
                         0bec d23e a96e 0600 0809 0a0b 0c0d 0e0f
                         1011 1213 1415 1617 1819 1a1b 1c1d 1e1f
                         2021 2223 2425 2627 2829 2a2b 2c2d 2e2f
                         3031 3233 3435 3637
```

Current Implementations

This is a review only, and in no way comprises an exhaustive list, or does it attempt to provide any product comparison.

It is appropriate to start with the current status of 'norm', which has not been developed since its original inception, this can be explained by the development of 'scrub', the normaliser module for 'pf', OpenBSD's firewall [20]. Scrub uses the Handley et al paper [1] as its template. Currently, 'pf' (inc. scrub) is being ported to FreeBSD, NetBSD and Linux so the Open Source community (Apple Macintosh OS X users included) will shortly enjoy the performance of a highly regarded firewall and normalisation system for free. In the real world an Application Level Normaliser would be required to complete the system, thereby covering the Network, Transport and Application Layers, the Open Source package 'Hogwash' [37] would make a suitable companion. In principle, (not withstanding the additional processing load) there should be no reason why one should not be able to run hogwash on the same box as 'pf'. There is also a plugin for Snort [38], the very popular Open Source NID called 'inline' [39] this turns Snort into a Gateway IDS which could also 'sit' on the box.

The commercial vendors have been slower to implement this technology, and in the fine tradition of sales jargon, designed to confuse the humble end user: Normalisation has been reborn or subsumed into 'In-Line IDS' or GatewayIDS' or even 'NIPS' (Network Intrusion Prevention Systems). In the words of Intruvent Networks Whitepaper, "Intrusion Prevention: Myths, Challenges, and Requirements" of April 2003, "could be thought of as something of a hybrid system, combining features of a standard IDS and a firewall" [40].

Nevertheless, there are systems out there calling themselves 'In-Line NIPS's' which do not appear to do any normalisation except packet rejection and offer the obvious ability to 'fail-closed' (a 'de rigeur' fix for the Ptacek Newsham 'fail-open', DoS attack). NIPS facilities, on the other hand, have been feature of certain NID's since way before the Ptacek Newsham paper (1998). In fact they utilise this 'feature' (say modification of a firewall rule set or two way TCP reset/'tear-down') as an example of how, using spoofed IP, the NID can be made to DoS it's own network.

Intruvent Networks, provide a hardware NID or "NIPS" which can amongst other configurations be set up as an 'In-Line' NIPS. Their Whitepaper states that in this mode, "all traffic passing through the device is protocol-scrubbed, ensuring that it complies with the relevant RFC's and acceptable practices and that no strange evasion or obfuscation techniques are being used." [40]. Network Associates acquired Intruvent Networks on the 14 May 2003.

If anyone still needed convincing about the necessity of normalisation, then just take a look at Arbor Networks 'PeakFlow' product [41]. Identified as one of the "10 IT startup companies to watch in 2001" by RedHerring [42] and with CISCO Systems as one of its major investors, Arbor has in the space of two years, obtained contracts with the US DoD and a host of Internet Service Providers around the globe. But it is Arbor's staff list that is most astonishing. With the exception of Handley et al, these individuals constitute a 'who's who' of normalisation and IDS research in the last 5 years. For instance:

Dr. Farnam Jahanian, founder and chairman of the board Dr. G. Robert Malan, founder, chief technology officer and vice president, product management	The logical progression from the 'Lighthouse Project' [24] is none other than PeakFlow!
Ted Julian, founder and chief strategist Dr. Craig Labovitz, director of network architecture	Previously, founder of security company '@stake' and Tim Newsham's ex boss while he was there Also ex Michigan University, published a paper on 'Internet Routing Instability' [43] with Malan and Jahanian in Oct 1997
Thomas H Ptacek, Product Manager, (not mentioned on their web site, but it is on his! [14])	What more is there to say, apart from what's Tim Newsham up to (and why is he so partial to fluffy bunnies? [44]
Dug Song, a member of the research and development team.	That he created Fragroute as late as April 2002, must prove, all be it circumstantially, that not all NID's are as effective as their vendors would have us believe

What a Normaliser can not do

Good security policy is based on layered defence. This paper is not proposing that Normalisation will solve all an organisations security needs, the two prior-discussed organisations view normalisation as one 'tool in the toolbox' in their fight against the Attacker. Normalisation does have its weakness, for example, an end system that runs out of memory will discard incoming packets that the NID (inc GID) may have already read and assumes the end system has processed it. CPU exhaustion and network saturation can also

cause this situation, therefore it is recommended that for 'high-profile targets' in an organisation, like the DNS and Web servers, Host IDS (HIDS) based monitoring should be employed.

An Alternative solution

It should be stated for the record that there is an alternative to packet level normalisation. It has been in existence (in theory) for a number of years and was obliquely alluded to in the Handley et al paper [1] as an alternative method to increasing the TTL field. In its most recent incarnation it has become known as "Active Mapping" [45]. In essence a modified NID calls the each end system and builds a profile database of its TCP/IP implementation. This file is then utilised during traffic monitoring, the theory being that the NID will then know whether the individual end system will accept or reject a specific packet. Another consequence of this 'mapping' is the NID knows how many hops away the system is therefore whether a packet with a low TTL will arrive. One problem about this method is that with such a large amount of 'stateholding' the NID becomes increasingly vulnerable to DoS attacks and subsequent 'fail-open'.

Conclusion

This paper has illustrated that certain vendors have developed products for the Internet that are not compliant with the RFC's. This ambivalence enabled the development of attack tools that not only provide reconnaissance by way of fingerprinting but can also deliver exploit payload to an end system whilst circumventing a variety of network security products, specifically, NIDS. A review of relevant Whitepapers published during the period 1996 to 2001, and the inescapable fact that products (e.g. Fragroute) are still being written exemplifies these issues are as current as they were at the time of the Ptacek and Newsham paper (1998). This paper continued with a 'walk-through' of the IP protocol identifying the relevant areas of concern and finally a Lab where packets were crafted and subsequently normalised. Concluding with a brief survey of 'real-world' implementations of normalisation as a component within a suite of network security products, this paper has demonstrated the necessity for all networks to employ normalisation or suffer the consequences.

References

1 Handley, Mark. Paxson,Vern. Kreibich, Christian.
"Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics"
Mid 2001. URL: (30 May 2003)

2 Kreibich, Christian. 'norm' can be found at
URL: http://sourceforge.net/project/showfiles.php?group_id=22071&release_id=69870 (30 May 03)

3 Kreibich, Christian. 'NetDuDe' can be found at
URL: http://sourceforge.net/project/showfiles.php?group_id=22071&release_id=112170 (30 May 03)

4 Song, Dug. 'Fragroute' can be found at
URL: http://naughty.monkey.org/~dugsong/fragroute/fragroute-1.2.tar.gz (30 May)

5 'Tcpdump' can be found at URL: http://www.tcpdump.org/release/tcpdump-3.7.2.tar.gz (30May 03)

6 Malan, G. Robert. Watson, David. Jahanian, Farnam. Howell, Paul.
"Transport and Application Protocol Scrubbing" March 2000.
URL: http://www.eecs.umich.edu/~rmalan/publications/mwjhInfocomm2000.pdf (30 May 2003)

7 Ptacek, Thomas H. Newsham, Timothy N. "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection" Jan 1998

URL: http://www.insecure.org/stf/secnet_ids/secnet_ids.html (30 May 2003)
8 Sapiro, Benjamin. "Application Level Content Scrubbers" August 22, 2001. URL: http://www.sans.org/rr/paper.php?id=800 (30 May 2003)
9 Paxson, Vern. "End-to-End Routing Behavior in the Internet" Presented at SIGCOMM96. Feb 1996 URL: http://www.acm.org/sigcomm/sigcomm96/papers/paxson.pdf (30 May 2003)
10 Paxon, Vern. "End-to-End Internet Packet Dynamics" Presented at SIGCOMM97. Sept 1997 URL: http://www.acm.org/sigcomm/sigcomm97/papers/p086.pdf (30 May 2003)
11 Paxon, Vern. "Automated Packet Trace Analysis of TCP Implementations" Presented at SIGCOMM97. Sept 1997 URL: http://www.acm.org/sigcomm/sigcomm97/papers/p054.pdf (30 May 2003)
12 Fyodor. "Remote OS detection via TCP/IP Stack Fingerprinting" October 1998 URL: http://www.insecure.org/nmap/nmap-fingerprinting-article.html (30 May 2003)
13 Fall, Kevin. Floyd, Sally. "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP" July 1996. URL: http://www.icir.org/floyd/papers/sacks.pdf (30 May 2003)
14 Ptacek, Thomas H. comments from his personal Web site. URL: http://www.sockpuppet.org/tqbf/ (30 May 2003)
15 Song, Dug. Fragrouter 1.6 A network intrusion detection evasion toolkit" Feb 1999. URL: http://www.securityfocus.com/data/tools/fragrouter-1.6.tar.gz (30 May 2003)
16 Horizon. "Defeating Sniffers and Intrusion Detection Systems" appeared in Phrack Magazine Volume 8, Issue 54 Dec 25th, 1998 URL: http://www.phrack.org/show.php?p=54&a=10 (30 May 2003)
17 Puppy, Rain Forrest. "A look at whisker's anti-IDS tactics" Mid 1999 URL: http://www.wiretrip.net/rfp/txt/whiskerids.html (30 May 2003)
18 Sanfilippo, Salvatore. " A new tcp port scan method" December 1998 URL: http://lists.insecure.org/lists/bugtraq/1998/Dec/0082.html (30 May 2003)
19 OpenBSD's home page URL: http://www.openbsd.org/ (30 May 2003)
20 "PF: The OpenBSD Packet Filter" taken from the OpenBSD FAQ page URL: http://www.openbsd.org/faq/pf/ (30 May 2003)
21 P Stevens, W. Richard. TCP/IP Illustrated, Volume 1: The Protocols. Addison Wesley Longman, Inc, 1994
22 Postel, Jon. "Internet Protocol, Protocol Specification" September 1981 URL: ftp://ftp.rfc-editor.org/in-notes/rfc791.txt (30 May 2003)
23 Smart, G. Matthew. Malan, Robert. and Jahanian, Farnam. "Defeating TCP/IP Stack Fingerprinting" August 2000 URL: http://www.usenix.org/events/sec2000/smart.html (30May 2003)
24 The Lighthouse Project web site URL: http://www.eecs.umich.edu/lighthouse/ (30 May 2003)
25 Mogul, J. Deering, S." Path MTU Discovery" Nov 1990 URL: ftp://ftp.rfc-editor.org/in-notes/rfc1191.txt (30 May 2003)
26 Handley, Mark. Paxson, Vern. Kreibich, Christian. Cited as REF [4] in "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics" This file has not appeared at the location indicated.
27 Postel, Jon. " Internet Control Message Protocol" September 1981 URL: ftp://ftp.rfc-editor.org/in-notes/rfc792.txt (30 May 2003)
28 Castro, Michael. " Jolt2: DoS that is not just for Windows Anymore" SANS Reading Room. August 11, 2000 URL: http://www.giac.org/practical/Michael_Castro_GSEC.doc (30 May 2003)
29 SANS Information "Help Defeat Denial of Service Attacks: Step-by-Step" March 2000. URL: http://www.sans.org/dosstep/ (30 May 2003)
30 daemon9 AKA route. "Project Loki" August 1996 URL: http://www.phrack.org/phrack/49/P49-06 (30 May 203)
31 Song, Dug. 'Fragroute' can be found at URL: http://naughty.monkey.org/~dugsong/fragroute/fragroute-1.2.tar.gz (30 May 2003)
32 Gtk+ 1.2.10 can be found at URL: ftp://ftp.gtk.org/pub/gtk/v1.2/gtk+-1.2.10.tar.gz (30 May 03)
33 Glib 1.2.10 can be found at URL: ftp://ftp.gtk.org/pub/gtk/v1.2/glib-1.2.10.tar.gz (30 May 03)
34 'Libpcap' can be found at URL: http://www.tcpdump.org/release/libpcap-0.7.2.tar.gz (30 May 03)
35 'Libevent' can be found at URL: http://naughty.monkey.org/~provos/libevent-0.7a.tar.gz (30 May 03)
36 'Libdnet' can be found at URL: http://prdownloads.sourceforge.net/libdnet/libdnet-1.7.tar.gz (30 May 03)
37 Anonpoet. Hogwash Information Site: URL: http://hogwash.sourceforge.net/ (30 May 2003)
38 Snort binaries can be found at URL: http://www.snort.org/dl/ (30 May 2003)

39 'In-Line' a module to enable Snort in Gaterway IDS Mode
URL: http://www.snort.org/dl/contrib/patches/inline/(30 May 2003)
40 Intruvent Networks, "Intrusion Prevention: Myths, Challenges, and Requirements" April 2003 (a form is required before access to this paper)
URL: http://www.intruvert.com/technology/white_papers.htm (30 may 2003)
41 Arbour Networks Home page
URL: http://arbornetworks.com/ (30 May 2003)
42 Shafer, Scott Tyler. "Ten to watch for the Red Herring 100" May 7, 2001
URL: http://www.redherring.com/mag/issue97/1780019178.html (30 May 2003)
43 Labovitz, Craig. Malan, G. Robert. Jahanian, Farnam. " Internet Routing Instability" Oct 1997
URL:http://www.eecs.umich.edu/~rmalan/publications/lmjInfocomm99.pdf (30 May 2003)
44 Tim Newsham's personal web page URL: http://www.lava.net/~newsham/ (30 May 2003)
45 Authors unknown! "Active Mapping: Resisting NIDS Evasion Without Altering Traffic" Nov 6 2002.
URL:http://www.cs.berkeley.edu/~ushankar/research/active/activemap.pdf (30 May 2003)

Last Updated: December 17th, 2017

Upcoming SANS Training

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

SANS Security East 2018	New Orleans, LAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Amsterdam January 2018	Amsterdam, NL	Jan 15, 2018 - Jan 20, 2018	Live Event
Northern VA Winter - Reston 2018	Reston, VAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SEC599: Defeat Advanced Adversaries	San Francisco, CAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SANS Dubai 2018	Dubai, AE	Jan 27, 2018 - Feb 01, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NVUS	Jan 28, 2018 - Feb 02, 2018	Live Event
Cyber Threat Intelligence Summit & Training 2018	Bethesda, MDUS	Jan 29, 2018 - Feb 05, 2018	Live Event
SANS Miami 2018	Miami, FLUS	Jan 29, 2018 - Feb 03, 2018	Live Event
SANS Scottsdale 2018	Scottsdale, AZUS	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS London February 2018	London, GB	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Southern California- Anaheim 2018	Anaheim, CAUS	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS Secure India 2018	Bangalore, IN	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS Dallas 2018	Dallas, TXUS	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Brussels February 2018	Brussels, BE	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Secure Japan 2018	Tokyo, JP	Feb 19, 2018 - Mar 03, 2018	Live Event
Cloud Security Summit & Training 2018	San Diego, CAUS	Feb 19, 2018 - Feb 26, 2018	Live Event
SANS New York City Winter 2018	New York, NYUS	Feb 26, 2018 - Mar 03, 2018	Live Event
CyberThreat Summit 2018	London, GB	Feb 27, 2018 - Feb 28, 2018	Live Event
SANS London March 2018	London, GB	Mar 05, 2018 - Mar 10, 2018	Live Event
SANS San Francisco Spring 2018	San Francisco, CAUS	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS Secure Osaka 2018	Osaka, JP	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS Secure Singapore 2018	Singapore, SG	Mar 12, 2018 - Mar 24, 2018	Live Event
SANS Paris March 2018	Paris, FR	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS SEC460: Enterprise Threat Beta	OnlineCAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced