Intrusion Detection Systems and IPv6*

Arrigo Triulzi

arrigo@northsea.sevenseas.org

Abstract

In this paper we will discuss the new challenges posed by the introduction of IPv6 to Intrusion Detection Systems. In particular we will discuss how the perceived benefits of IPv6 are going to create new challenges for the designers of Intrusion Detection Systems and how the current paradigm needs to be altered to confront these new threats. We propose that use of sophisticated dNIDS might reduce the impact of the deployment of IPv6 on the current network security model.

Keywords: Intrusion Detection Systems, IDS, NIDS, Firewalls, Security, IPv6.

1 Introduction

The field of Intrusion Detection is generally divided into two large categories: Host-based Intrusion Detection Systems (*HIDS*) and Network-based Intrusion Detection Systems (*NIDS*). The *HIDS* label is often used for tools as diverse as anti-virus programs, the venerable UNIX *syslog* and intrusion detection software such as *portsentry*. The *NIDS* label is similarly abused extending from firewalls to network intrusion detection software proper such as *Snort* [8]. A further specialisation is that of "distributed" NIDS, or *dNIDS*, which addresses large NIDS systems distributed over a wide network or designed following a distributed-computing paradigm. An excellent example of the latter category is Prelude IDS [10][1]. We shall be discussing NIDS and in particular the rising need for dNIDS in the context of IPv6.

Historically, long before the appearance of *Snort* and other tools of the trade, most of the world was using *tcpdump* [11] for intrusion detection on the network: a "down to the bare metal" packet dumping utility. The very first Network Intrusion Detection System, called *Network Security Monitor*, was written by Todd Heberlein and colleagues at UC Davis under contract with LLNL between 1988 and 1991 [2, 1]. It was then extended to create *NID* which found widespread use in the US military [5]. This was rapidly followed by *Shadow*, written by Stephen Northcutt and others during 1996 [3, 4, 6] again for use by the US military. From *Shadow* onwards there has been an explosion of free and commercial NIDS products which is beyond the scope of this brief introduction (the more historically minded reader will find more detailed information in [12]).

There has been little effort to expand the current set of NIDS to support the IPv6 protocol, mainly due to a lack of demand. Despite years of forecasts of "doom and gloom" when discussing the famous exhaustion of IPv4 addresses there has been little uptake of IPv6 with the exception of countries such as Japan which have been very active in promoting it [13, 14]. This has meant that very little non-research traffic has actually travelled over IPv6 and hence the impetus for new attacks making use of IPv6 features has been absent.

A trivial example is the author's personal mail server and IDS web site [15] which has been reachable on IPv6 since December 2001: there has been only a *single* SMTP connection over IPv6 in over a year. This is to a mail server with an average of a thousand connections per day.

At first glance this might indicate that there is little point in pursuing IDS under IPv6 but this should instead be thought of as an opportunity to be ready *before* the storm hits as opposed to catching up afterwards as in the IPv4 space.

2 IPv4 and IPv6

The discussions around IPv6 started a while back, despite what the current lack of acceptance might suggest, with the first request for white papers being issued in 1993 as RFC1550 [16] using the name

*or "Why giving a sentry an infinite block list is a bad idea".

[1] Although they do define themselves as a "hybrid" IDS as they combine some HIDS facilities within a dNIDS design.

IPng, for "IP New Generation". By the end of 1995 the first version of the protocol specification was published as IETF RFC1883 [17] which formed the basis for the current IPv6 deployment. For a detailed description of the header formats the reader is referred to [21].

The key differences between IPv6 and IPv4 can be summarised briefly as:

- Simplified header,
- Dramatically larger address space with 128-bit addresses,
- Built-in authentication *and* encryption packet-level support,
- Simplified routing from the beginning,
- No checksum in the header,
- No fragmentation information in the header.

We shall now take a critical look at the specific differences which are relevant to the IDS professional.

2.1 Simplified header

The comparison between an IPv4 header and an IPv6 header is striking: the IPv6 header is cleaner, with fewer fields and in particular everything is aligned to best suit the current processors. The rationale behind this change is simple: memory is now cheap, packing data only means harder decoding on processors which assume data is aligned on 4 or 8 byte boundaries.

The header is simplified by removing all the fields which years of experience with IPv4 have shown to be of little or no use. The best way to visualise the cleaner and leaner IPv6 basic header is to look at some tcpdump output representing the same transaction (an ICMP Echo Request packet) between the same two hosts over IPv4 and IPv6.

```
14:39:29.071038 195.82.120.105 > 195.82.120.99: icmp: echo request (ttl 255, id 63432, len 84)
0x0000   4500 0054 f7c8 0000 ff01 4c6e c352 7869        E..T......Ln.Rxi
0x0010   c352 7863 0800 1c31 3678 0000 3e5f 6691        .Rxc...16x..>_f.
0x0020   0001 1562 0809 0a0b 0c0d 0e0f 1011 1213        ...b............
0x0030   1415 1617 1819 1a1b 1c1d 1e1f 2021 2223        ............!"#
0x0040   2425 2627 2829 2a2b 2c2d 2e2f 3031 3233        $%&'()*+,-./0123
0x0050   3435 3637                                     4567

14:40:04.096138 3ffe:8171:10:7::1 > 3ffe:8171:10:7::99: icmp6: echo request (len 16, hlim 64)
0x0000   6000 0000 0010 3a40 3ffe 8171 0010 0007        `.....:@?..q....
0x0010   0000 0000 0000 0001 3ffe 8171 0010 0007        ........?..q....
0x0020   0000 0000 0000 0099 8000 60fe 4efb 0000        ..........`.N...
0x0030   bc5e 5f3e 2f77 0100                            .^_>/w..
```

The first obvious difference is in the version nibble, a six instead of a four. We then notice how the IPv6 header appears to consist mainly of zeros. This is because the first eight bytes only contain essential data, everything else being relegated to so-called "extension headers" if need be. In this particular case the packet contains no extension headers, no flow label, simply the payload length, next header (indicating an ICMPv6 header), and hop limit (64, hex 40 in the packet dump). This is followed by the 128 bit addresses of source and destination comfortably aligned on 8-byte offsets making the header disassembly efficient even on 64-bit CPUs.

¿From an IDS perspective this is excellent because on modern CPUs taking apart the IPv4 header to detect subtle packet crafting is very inefficient due to the alignment of the data fields. With IPv6 the decomposition of the various fields of the header and extension headers can take place efficiently. This can only mean a gain in per-packet processing speed, an important measure when Gbit/s interfaces are brought into play.

Performance is further enhanced by the lack of fragmentation information in the header: this means that the basic header does not need to be decoded for fragmentation information. Indeed, fragmentation is now dealt with by extension headers. This does not necessarily make the fragmentation attacks developed for IPv4 stacks obsolete (see [23] for a discussion of these attacks) as incorrect datagram reassembly can still take place.

2.2 Larger address space

At first glance the fact that IPv6 offers 2^{128} addresses might seem a blessing after all the problems with address exhaustion in IPv4. The indiscriminate allocation of IPv4 addresses, which were initially thought to be plentiful, brought us protocol violations such as *Network Address Translation*, at times also referred to as "masquerading", and other "patches" to work around the problem.

If the computers currently connected to IPv4 were moved over to an IPv6 infrastructure then in effect all that would happen is a renumbering of hosts. A consequence of IPv6 is to be more than a substitute of IPv4 and bring us closer to the concept of "ubiquitous networking": the larger address space as an incitement to connecting *everything* to the Internet. A number of companies are already working on so-called "intelligent homes" and in particular Internet-connected home appliances. The deployment of IPv6 will make it possible for households to be assigned ample addresses for each of their appliances to be directly connected to the Internet and report on the state of the fridge, the washing machine and so on.

Let us now wear the paranoid IDS implementor's hat: to connect to the Internet each of these appliances must run an embedded operating system with a TCP/IP stack of some description. Furthermore it needs to have sufficient processing power to offer something like a simple web server for configuration and user interaction. Let us further imagine that there is a security flaw with this software which allows a remote user to take over control of the appliance and use it as a Distributed Denial of Service host (see [22] for an exhaustive discussion of DDoS). Assuming Internet-connected home appliances become widely used then we are effectively facing the possibility of an attack by an innumerable amount of simple systems which will not be trivial to patch[2].

Furthermore IPv6 has, by design, simplified the way a device can find its neigbours on a network segment. A pre-defined link-local "neighbour discovery" procedure for allocating an IPv6 address using ICMPv6 has been drawn up which uses the 48bit Ethernet address as a differentiator between hosts . A NIDS now loses what little chance it had of discovering devices having a dynamically provided address as the allocation is no longer "public", simply changing an Ethernet card can easily blind address-based rules[3].

¿From this we can draw the conclusion that a number of very useful features of IPv6 can seriously turn against the IDS community as it gains a foothold in the Internet and more devices, in particular embedded devices, are produced with IPv6 Internet connectivity.

2.3 Authentication and Encryption

Authentication and encryption are available in IPv4 through the use of IPsec [18, 19] which has not been widely deployed until recently. The main field of application has been that of VPN tunnels and this limited interest has meant that until most router vendors had implemented it there were few users. Furthermore, these few were mainly users of implementations running on open source operating systems. The main factors blocking the widespread use of IPsec have been the complicated setup and key-exchange protocol which, although necessary for proper security, did require more knowledge than the average systems manager possessed.

This has meant that the IDS community has been mainly concerned with channels protected by SSL or TLS (see [20] for a formal definition of TLS). For example an attack over HTTP which is visible to a NIDS installed between the source and the destination becomes invisible if transmitted over HTTPS as the standard "string in the payload" matching will fail. A number of solutions to the encrypted channel problem have been postulated: from session key sharing by the web server allowing "on-the-fly" decryption to server-side storage of keys for later off-line decryption of captured packets. One solution is to use *ssldump* [30] to decode SSL but of course you need control of both endpoints to obtain the session keys.

The deployment of IPv6 has the potential to worsen the "encrypted attack problem" quite dramatically as IPv6 has had authentication and encryption built-in since its inception[4]. One of the default extension

[2] Remote updating of software does not improve the situation much as various incidents with Microsoft's Windows Update facility have shown (see, amongst the many, [27]).

[3] Note that DHCP is being extended to IPv6 as DHCPv6 but the availability of "neighbour solicitation" as default provides a far greater challenge.

[4] One should note that a few features of IPv4 were rarely used as intended, for example Quality of Service and indeed, amongst the IP options, a "Security" option which is often referred to as "something used by the US DoD" and deals with classification. A quick look at a Unix system's include file `/usr/include/netinet/ip.h` is a recommended read.

headers for IPv6 is the "IPv6 Authentication Header" (AH) which is nothing other than the same AH mechanism as used in IPsec but for IPv6.

As IPv6 deployment increases it will be interesting to see if, unlike IPsec, the AH mechanism is more widely used and, in particular, if trust can be moved from the weak "secure web server" model[5] to the protocol layer for a large number of applications.

This is a double-edged sword for the IDS community. It is very tempting to think that widespread availability of encryption and authentication improves security but all this does is move the goal posts. There is *no guarantee* that the traffic being carried by the authenticated and encrypted link is legitimate and furthermore it will now be illegible as far as the NIDS is concerned. The solution of "sharing the keys" as in the web server scenario becomes the nightmare of sharing the keys of *every host* on an IPv6 network with the NIDS.

3 Intrusion Detection

Intrusion Detection is moving in two opposite directions: one is the "flattening" towards the low-end of the market with vendors attempting to sell "shrink-wrapped" IDS packages, the other is the "enterprise" attempting to pull together all security-related information for a large company.

Both suffer from the same problem: customers are used to viewing IDS as "something which looks like an Anti-Virus". This view is strengthened by the behaviour of most systems: you have rules, they trigger, you attempt to fix the problem and you upgrade or download the rules. The fundamental problem is not so much the perception of IDS as the perpetration of a dangerous methodology.

Let us consider an example from a real life situation which is closest to IDS: the concept of a sentry at a checkpoint. When instructing a sentry one defines certain criteria under which something (be it a person or a vehicle of some description) is to be allowed to pass the checkpoint. One does *not* attempt to define all unauthorised behaviour or vehicles as the list is fundamentally infinite. This works quite well and improvements to the system are given by making the "pass rules" more stringent and well-defined.

3.1 White-listing – Describing normality

So why do IDS systems (and Anti-Virus systems for that matter) attempt to define all that is bad?

The answer is not as simple as one would wish: it is a mixture of historical development and the lure of "attack analysis". Historically in computing Intrusion Detection has always been the alerting to something being amiss, for example "bad logins". This was a good example of "white-listing" or the alerting on anything which was not known. Unfortunately "white-listing" has the disadvantage of generating a lot of messages and people started ignoring the output of `syslog` under Unix. As the alerting moved onto the network the idea somehow changed into the equivalent of an anti-virus (which at the time was already a well-developed concept for MS-DOS systems) losing the "white-listing" concept on the way.

Let us now elaborate the second point: the analysis of a new attack or indeed the search for new attacks is fashionable, interesting and challenging. It is therefore a matter of pride to be the first to publish the rule which will catch a new attack. Given the choice a security analyst would much rather play with a new attack than spend his time laboriously analysing the network to write "white-listing" rules.

If we consider the ruleset released with Snort version 1.9.0 (see [9]) we find definitions for 2321 rules. All of these rules are "alert" rules, defining traffic to be blocked. Then towards the end of January 2003 the Internet was swamped by a new worm targeting Microsoft SQL server installations (see [7] for an in-depth analysis). Anyone running the base Snort ruleset would have never noticed but neither would someone who had just updated his rules for the simple reason that no rule had yet been written.

Furthermore there is no limit to this process: new attacks are published, new rules are written and added to the list. Hence there is *no upper bound* on the number of rules.

How many sites really required remote access to their Microsoft SQL Server? Possibly a handful. So why was the IDS not instructed to alert on any incoming (or indeed, outgoing) connection to the relevant port?

[5] The security of SSL certificates issued by so-called "trusted parties" has been sadly found lacking in the crucial step of verifying the identity of the person or company requesting a certificate. These failures in identity verification make it difficult to equate "trust" with SSL certificates despite the marketing efforts.

The key paradigm shift which is required is precisely that NIDS move from a "flag what is known to be bad" mechanism to "flag what is not explicitly allowed". This mimics closely what has happened in firewall design. Originally firewalls were setup to block specific ports, reflecting the academic nature of the Internet, they are now setup to block everything except what is deemed safe in most installations. The drawback of a white-listing setup is that the number of "false positives" will increase dramatically each time a network change takes place without informing the security team. This does have the beneficial side-effect of enforcing proper communication between the networking and security groups. There are also sites for which white-listing will be fundamentally impossible (such as an ISP's co-location centre) until much more sophisticated "auto-whitelisting" software becomes available; but any site with a security policy, a small number of well defined external access points and internal routing points should be able to define suitable white-lists.

Finally a further benefit: the number of rules required to define white-lists is limited and known. This allows a correct measurement of the performance of a NIDS under realistic conditions. As more NIDS are connected to high-speed networks the issue of packet processing speed becomes of paramount importance. The larger the set of rules which needs to be applied against every single packet the slower the NIDS will perform. If a NIDS is not dropping packets with 1000 rules it is not necessarily the case that it will continue doing so with double the number. In particular as more and more content-based rules are deployed (which require expensive string-matching) the performance can only decrease further. One solution is of course to throw hardware at the problem but that is only a palliative cure, a better solution is attack the problem at its root by deploying white-lists.

3.2 Ubiquity – Monitoring needs to be pervasive

Once the paradigm has been shifted it needs to be completed. This entails the understanding that an isolated NIDS is of little use, just like a lone sentry asked to patrol an immense perimeter.

For Intrusion Detection to be truly effective it is necessary to move from the sporadic installations designed to tick a box on the security policy to a proper monitoring system. There should be NIDS installations at every entry point into the network, be it external or indeed internal. There should be no situation in which there is a negative answer to the request "pull the data from that network segment".

Once a NIDS permeates the network it becomes possible to follow "alarm flows" and have more than the sporadic data points which a NIDS and maybe a few firewalls can offer. If ubiquitous monitoring is deployed with white-listing then it suddenly becomes possible to monitor abuse throughout the network and indeed "follow through" abuse in the network. It will no longer be possible for an internal issue to grow out of all proportions before it is noticed, often as it tried to "escape" via the firewalls.

3.3 Aggregating, correlating and reducing data

The larger the enterprise the heavier the requirements from an ubiquitous NIDS deployment, in particular in terms of staffing needs.

Once data is collected there is very little point in it being left on the collecting devices. The first requirement for an advanced NIDS deployment is to have a central "aggregation engine" which takes all the data from the sensors. This should probably be a database and most NIDS vendors these days offer this capability.

Having the data in a central location means that it is now available for correlation: a perimeter-wide scan for open ports should most definitely not be reported as a number of individual events but as a single instance of multiple events. The *instance* is a "port scan", the *multiple events* are the individual alerts issued by the sensors. This correlation can be made as sophisticated as required, for example correlating by originating subnet rather than single host or by correlating using routing "autonomous system" numbers.

Once the data is correlated it can be reduced: if the scan is to multiple locations but the mechanism is identical there is no need to store all the individual packets. It is sufficient to store a representative packet and then reference the collecting sensors to preserve the entity of the event.

Without these three crucial steps an advanced NIDS deployment can only fail under the weight of its own data and the expense in identifying and locating it across the whole network.

3.4 Network knowledge, advanced analysis and management

Transforming the collected data is still not enough. The bane of all NIDS installations is the incidence of false positives. Attempting to reduce false positives can often lead to the implementation of rulesets which are dangerously restricted in scope. As we discussed previously white-listing can *increase* the incidence of false positives unless appropriate communication between the networking and security teams is in place.

One solution to the false positive problem is to add knowledge of the network to the NIDS. A large number of products used by networking teams store information about the systems on the network from the system type to the operating system. This information should be fed into the NIDS to help it discriminate between attacks which should merely be logged as "informational" and those which instead require immediate action.

A trivial example is that of a web site running Apache. It is definitely of interest to catalogue attacks against Microsoft's IIS web server being used against this web site but clearly they do not pose much of a threat. Conversely an Apache attack should definitely command immediate attention. Even basic discrimination by "log level" can improve the quality of security analysis and response dramatically. It is much simpler to address a small number of problems which have been made immediately obvious by the high log-level than having to look for the same problems in the midst of thousands of irrelevant attacks.

Once such a sophisticated system is in place then much more advanced analysis is possible. One interesting option is that of "Differential Firewall Analysis" [24] where the correctness of firewall operation is verified by means of NIDS placed on both sides of the system under monitoring. The rationale behind such an analysis is that the worst case scenario for a firewall is that it is breached via a flaw in its software. There will be no record in the logs of this breach but the intruder will have penetrated the system (or indeed, a user on the inside might be connecting to the outside) and will remain totally undetected. Differential Firewall Analysis attempts to alert to such flaws by verifying and correlating traffic on both sides of the firewall.

Finally, how does one manage a system of this size? It is clear that there is no hope of managing rulesets individually: there needs to be a centralised rule repository from which rules for the sensors are "pushed" and activated in unison. This prevents those situations in which half of the sensors are running on the new ruleset and the other half on the older version rendering analysis impossible. Direct interaction with the individual sensors should be prevented with all management, from code updates to individually turning sensors on and off, controlled from a single centralised location where all state is kept. This is also a huge step towards 24x7x365 availability: if state is kept in a centralised location then it becomes much simpler to deploy replacement sensors or indeed fail-over to spares when failures occur.

3.5 Deploying Intrusion Detection suitable for IPv6

The introduction of IPv6 into an environment with a sophisticated NIDS deployment as the one we have been describing represents less of a worry. The most important hurdle is perhaps that of authentication and encryption: a sophisticated NIDS would want to at least verify the validity of the AH in each packet if not check the contents of the ESP. This is perhaps the least tractable of problems: despite the presence of hardware cryptographic acceleration cards and support for them in a number of open source operating systems (in particular OpenBSD, see [25, 26]) there is a noticeable difference between offering fast crypto support for SSL and SSH key generation and decrypting packets on the fly at megabit/s rates.

4 Conclusions

Besides integrated cryptography there is little else to differentiate IPv6 from IPv4 technology from an NIDS point of view with the exception of the larger address set. It is the *uses* of IPv6 technology which present the greatest challenges as they might finally achive what used to be the golden grail a few years ago of "everything on the Internet" which IPv4 did not fulfil. This will make "white listing" of paramount importance as the definition of thousands of blocking rules based on source addresses will simply no longer be possible. Furthermore the proliferation of connected devices will make accurate, pervasive and timely monitoring a core necessity for any enterprise or large network.

The single largest contribution to network security in a large environment is education. There is no substitute for generating awareness of dangers such as the blind opening of attachments or the installation

of unauthorised software. It has also been the security industry's greatest failure that, as more and more sophisticated tools became available, the issue of education has remained.

The deployment of IPv6 will place increased pressure on the requirement for a paradigm change from the current localised solutions to a much more distributed system and in particular from the "anti-virus lookalike" to a system finally resembling a proper sentry.

An NIDS should only ever be deployed as part of an information security policy that it needs to monitor, just like a sentry is part of a physical security policy. Similarly, as we do not hand a sentry a list of every person banned from crossing a checkpoint, we should not attempt to define rules for every possible type of "bad traffic". We should instead concentrate on working out what traffic is allowed on a network and define everything else as bad. With IPv6 the size of the address range (2^{128} possible addresses) and the much more dynamic nature of IPv6 addressing would make blacklisting in firewalls an almost impossible exercise.

It would be commendable if the current IPv6 test back-bone, 6Bone, started deploying dNIDS to see what challenges await us before widespread deployment of the new protocol. If we consider the trend towards large distributed computing (European Data Grid, Asian Data Grid and other similar projects, see [28, 29]) which will require more and more address space and network communication, then dNIDS will have to become the security monitoring solution. The amount of processing power and network bandwidth make these grids a formidable opponent should they fall into the wrong hands (and DDoS is precisely about creating "attack" grids). This threat means that in a distributed environment it is pointless to address monitoring at a few, disconnected, points on the grid. It has to be pervasive and ubiquitous, reporting centrally to the CERT responsible for that particular grid which needs to be able to take prompt and informed action before the problem spreads.

It is perhaps surprising that NIDS design has come a full circle. Careful reading of the early papers on *NID* and *Shadow* describe nothing other than dNIDS systems with the exception of a centralised alert database and enterprise-class management facilities. It is the author's belief that the reason for this is that, finally, the understanding that a NIDS is *not* anti-virus software by a different name is starting to diffuse in the industry.

5 Acknowledgements

The author would like to thank the Programme Committee for the kind invitation to SPI2003 and Dr. Diana Bosio and Dr. Chris Pinnock for valuable comments to the text.

References

[1] Todd Heberlein. *"NSM, NID and the origins of Network Intrusion Detection"*. Private Communication, July 2002.

[2] Todd Heberlein et al. *"A Network Security Monitor"*. Proceedings of the IEEE Computer Society Symposium, *Research in Security and Privacy*, pages 293-303, May 1990.

[3] Stephen Northcutt. *"The History of Shadow"*. Private Communication, July 2002.

[4] Stephen Northcutt and Judy Novak. *Network Intrusion Detection*. 2nd Edition, Chapter 11, pages 198-199 and 220-221. New Riders, 2001.

[5] Stephen Northcutt and Judy Novak. *Network Intrusion Detection*. 2nd Edition, Chapter 11, page 198. New Riders, 2001.

[6] Stephen Northcutt. *Intrusion Detection Shadow Style*. SANS Institute, 1999.

[7] Marc Maiffret. *"SQL Sapphire Worm Analysis"*. eEye Digital Security, January 2003, `http://www.eeye.com/html/Research/Flash/AL20030125.html`.

[8] Marty Roesch et al. *Snort – The Open Source NIDS*. `http://www.snort.org/`.

[9] Marty Roesch et al. *Snort – The Open Source NIDS*. Release 1.9.0, October 2002, `http://www.snort.org/dl/snort-1.9.0.tar.gz`.

[10] Yoann Vandoorselaere, Pablo Belin, Krzysztof Zaraska, Sylvain Gil, Laurent Oudot, Vincent Glaume and Philippe Biondi. *Prelude IDS*. `http://www.prelude-ids.org/`.

[11] *tcpdump*. http://www.tcpdump.org/.

[12] Paul Innella. *"The Evolution of Intrusion Detection Systems"*. SecurityFocus, November 2001, http://www.securityfocus.com/infocus/1514.

[13] *"Overview of IPv6 Projects around the World"*. IPv6 Forum, http://www.ipv6forum.org/navbar/links/v6projects.htm.

[14] *"UK IPv6 Resource Centre – Whois service"*. Lancaster University and IPv6 Forum, http://www.cs-ipv6.lancs.ac.uk/ipv6/6Bone/Whois

[15] Arrigo Triulzi. *"IDS Europe"*. https://ids-europe.alchemistowl.org/.

[16] S. Bradner and A Mankin. *"IP: Next Generation (INng) White Paper Solicitation"*. IETF, December 1993, http://www.faqs.org/rfcs/rfc1550.html.

[17] S. Deering and R Hinden. *"Internet Protocol, Version 6 (IPv6) Specification"*. IETF, December 1995, http://www.faqs.org/rfcs/rfc1883.html.

[18] S. Kent and R. Atkinson. *"IP Authentication Header"*. IETF, November 1998, http://www.faqs.org/rfcs/rfc2402.html.

[19] S. Kent and R. Atkinson. *"IP Encapsulating Security Payload (ESP)"*. IETF, November 1998, http://www.faqs.org/rfcs/rfc2406.html.

[20] T. Dierks and C. Allen. *"The TLS Protocol Version 1.0"*. IETF, January 1999, http://www.faqs.org/rfcs/rfc2246.html.

[21] Adolfo Rodriguez, John Gatrell, John Karas and Roland Peschke. *TCP/IP Tutorial and Technical Overview*. Chapter 17. IBM & Prentice Hall, October 2001, http://www.redbooks.ibm.com.

[22] Dave Dittrich. *"Distributed Denial of Service (DDoS) Attacks/tools"*. University of Washington. http://staff.washington.edu/dittrich/misc/ddos/.

[23] Jason Anderson. *"An Analysis of Fragmentation Attacks"*. SANS Reading Room, March 2001, http://www.sans.org/rr/threats/frag_attacks.php.

[24] Arrigo Triulzi, *"Differential Firewall Analysis"*. In preparation, February 2003. http://www.alchemistowl.org/arrigo/Papers/differential-firewall-analysis.pdf.

[25] *"Cryptography in OpenBSD"*. The OpenBSD project. http://www.openbsd.org/crypto.html.

[26] Theo de Raadt, Niklas Hallqvist, Artur Grabowski, Angelos D. Keromytis and Niels Provos. *"Cryptography in OpenBSD: An Overview"* in *Proceedings of Usenix*, 1999. http://www.openbsd.org/papers/crypt-paper.ps.

[27] John Leyden. *"Code Red bug hits Microsoft security update site"*. The Register, July 2001. http://www.theregister.co.uk/content/56/20545.html.

[28] *"The European Data Grid Project"*. CERN. http://eu-datagrid.web.cern.ch/eu-datagrid/.

[29] *"The Asian-Pacific Grid Project"*. http://www.apgrid.org/.

[30] Eric Rescorla. *ssldump*. http://www.rtfm.com/ssldump/