Vafa Dario Izadinia

Fingerprinting Encrypted Tunnel Endpoints

Submitted in partial fulfillment of the requirements for the degree Master of Science (Computer Science) in the Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, Pretoria, South Africa.

February 14, 2005.

Fingerprinting Encrypted Tunnel Endpoints

by
Vafa D. Izadinia

Abstract

Operating System fingerprinting is a reconnaissance method used by White-hats and Blackhats alike. Current techniques for fingerprinting do not take into account tunneling protocols, such as IPSec, SSL/TLS, and SSH, which effectively `wrap` network traffic in a ciphertext mantle, thus potentially rendering passive monitoring ineffectual. Whether encryption makes VPN tunnel endpoints immune to fingerprinting, or yields the encrypted contents of the VPN tunnel entirely indistinguishable, is a topic that has received modest coverage in academic literature. This study addresses these question by targeting two tunnelling protocols: IPSec and SSL/TLS. A new fingerprinting methodology is presented, several fingerprinting discriminants are identified, and test results are set forth, showing that endpoint identities can be uncovered, and that some of the contents of encrypted VPN tunnels can in fact be discerned.

Keywords: Fingerprinting, network forensics, protocol analysis, IPSec, IKE.

Degree: Magister Scientia
Supervisor: Prof. D. G. Kourie
Co-Supervisor: Prof. J.H.P. Eloff
Department of Computer Science

Acknowledgements

I would like to express my gratitude to the following people for their contribution to the success of this work.

- Prof. Derrick Kourie and Prof. Jan Eloff for their supervision.
- Marco Slaviero, for assistance with TeX and helpful insight and discussions.
- Jacques Malan, for initially helping develop the fingerprinting idea, and for subsequent helpful discussions of industry-related applications.
- Mioljub Jovanović and Sven Lankmans at the NMS TAC, and Philippe Thomsin for encouraging me to pursue this degree.

The greatest appreciation however, I would like to express to my parents and my sister, for their constant support, and for helping me maintain focus throughout.

Contents

List of Figures

List of Tables

Corroborating Material

A CD-ROM has been included at the end of this dissertation, containing supporting data. The contents of this CD-ROM are:

- This dissertation in electronic (PDF) format.
- The `lx-gw-1.tar` and `lx-gw-2.tar` files, containing all the *isakmpd* and *racoon* configuration files for the first and second IPSec gateway respectively, as discussed in Appendix A.
- The *isakmpd* and *racoon* network traces from the IPSec VPN tunnel setup, as discussed in Appendix B.
- The Ethereal protocol analyser `ethereal-setup-0.10.7.exe` Windows installer, with the required WinPcap `WinPcap_3_0.exe` installer, and the respective `ethereal-0.10.7.tar.gz` and `libpcap-0.8.3.tar.gz` source files for Unix/Linux systems.
- The `ipsec-tools-0.4.tar.gz` source file, containing the *racoon* user-space IPSec tools ported from the KAME project.
- The `isakmpd_20041012.orig.tar.gz` source file, containing the *isakmpd* daemon and supporting files, for use in Unix/Linux systems.
- The OpenSSL `openssl-0.9.7e.tar.gz` source file, needed for configuring a Certification Authority (CA) and generating x.509 certificates.

Chapter 1

Introduction

1.1 Motivation

An endless race, it seems, exists between those who want to protect information, and those who want to uncover it; between the proponents of privacy, and the proponents of transparency; between the minds behind systems designed to keep skilled intruders out, and those whose aim is to orchestrate a break, so simple, that it displays a fundamental flaw in the system´s design.

This is of particular importance to the digital world, where all information is essentially `available`, the challenge lying in how to obtain it. Faced with a predicament of this magnitude, the initial reaction is to throw cryptography at the problem. In late 2003, for example, commenting on his earlier book Applied Cryptography, Bruce Schneier referred to this by saying that some readers view cryptography as a magical `fairy dust` that can be sprinkled on any cryptographic problem, in order to solve it [111].

Oftentimes, the worlds greatest minds come together to develop protocols, and cryptographic algorithms, that are intended to increase the overall security of a system. More often than not, these protocols and algorithms are technically sound and astonishingly elegant in their simplicity [112, 110, 73]; more often than not, however, they are implemented in ways they were not

intended to be, or with minor `enhancements` that digress from the original design, which introduce a weakness in the system. These two: the implementation, and the relaxed adherence to the original standards, account for weaknesses in vendors´ products. It comes as no surprise then, when products boasting strong encryption, are bypassed, or broken-into.

Whether it be intruders attempting to break a system, designers trying to devise a way to better protect the information, or investigators putting together clues in the aftermath of a break-in; the more is known of the system as a whole, the more detailed the image that can be constructed. One method for acquiring this information is termed Operating System Fingerprinting, or OS Fingerprinting for short [127, 90, 123, 11, 77]. One form of OS Fingerprinting for example, probes the target system by sending well-known, specially crafted, or otherwise unexpected information to the target, in the attempt to elicit a response from it. These responses, in turn, differ from operating system to operating system (and even between different versions of the same OS); thus enabling the interested party to determine what OS the target system is running. Fingerprinting can be used by intruders to determine what avenue of attack to employ when breaking into a system; it can be used by investigators to piece together the network path along which an attack took place; and it can also be used by system designers and developers in an attempt to better disguise the identity of a system.

OS Fingerprinting can be used by legitimate network administrators, as well as by individuals of ill-intent, wishing to obtain information otherwise hidden behind a firewall. This is known as network mapping [13, 81] or topology discovery [63]. Once the fingerprinting process has produced a list of machines with matching OS details, an attacker may determine which machines are most vulnerable, and prepare to launch an attack, exploiting the machines weaknesses.

1.2 Encrypted Tunnel Endpoint Fingerprinting

As Virtual Private Network (VPN) [26] technology which allows home users to be connected to their corporate networks by means of encrypted tunnels, is becoming ubiquitous, it follows that new avenues of attack are being devised, and that endless race is entering a new lap.

Two of the most widely deployed VPNs, albeit for different applications, are IPSec and SSL VPNs. IPSec VPN tunnels are established at Layer 3 of the OSI stack [32, 3] whereas SSL VPNs are referred to as Application-Layer or Layer 7 VPNs [92, 119]. When deploying VPN tunnels, the endpoints of which are frequently placed behind firewalls, effectively bypassing them; being able to glean information about the devices with which the tunnel is being established (i.e., the tunnel endpoints) may enable an attacker to bypass the firewall altogether. Figure 1.1 shows a typical deployment of VPN gateways (terminators/concentrators) placed behind firewalls in two communicating corporate networks, resulting in the VPN tunnel being established with no knowledge of the firewalls, since the encapsulated IPSec traffic would appear as payload to regular IP traffic[1].

This dissertation is the product of a study of VPN implementations (both IPSec and SSL), available on the most widely used workstation and server OSs; these are Solaris 9, OpenBSD 3.5, Linux (2.6 kernel), Windows 2000, and Windows 2003. It explores the question of whether solid internetworking protocols and cryptographic algorithms that are relied on for end-to-end security, are weakened by variations in their commercial implementation.

A series of tests have been conducted, and a methodology developed by

[1]This statement assumes that by designing the IPSec link in this way, the enterprise is agreeing to allow the IPSec traffic through the firewall. It should also be noted for completeness, that IPSec traffic can be identified as such by the ESP or AH headers following the regular IP headers. IPSec traffic is therefore not *indistinguishable* from regular IP traffic, but harmless content in legitimate IPSec packets *is* indistinguishable from malicious content.

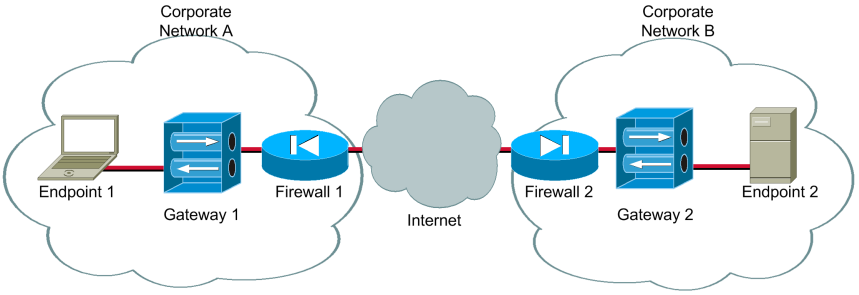


Figure 1.1: Depiction of a VPN tunnel established *through* firewalls, initiating and terminating on gateways behind the firewall in each corporate network

which OSs on either end of the VPN tunnels can be fingerprinted based on the peculiarities of their IKE/IPsec protocol implementation. This is a lengthy process that relies mainly on traffic and protocol analysis.

1.3 Related Work

Up until now, fingerprinting has been done by one of the following methods:

- Observing the behaviour of the TCP/IP stack in the target host [127]; that is, how the stack reacts to malformed IP datagrams, to IP datagrams with invalid lengths, to TCP packets with an incorrect combination of flags set, and other typical TCP/IP behaviour such as default TCP windows sizes, and initial sequence number (ISN) analysis.

- Banner-grabbing [45]; that is, making FTP, telnet, and HTTP connections to the target machine, and recording (or `grabbing`) the default banner displayed (such as *Welcome to Machine-Name, running RedHat Linux 8, Kernel 2.4.19*).

- Gleaning information by attacking specific services on the target machine, such as the SMTP service [12], and observing its response in the face of errors; such as sending a `MAIL FROM` without a `HELO`, using `MAIL`

`FROM <>` with an empty address, or using invalid source addresses, to name but a few.

- Observing the target machine´s ARP etiquette [24]; that is, noting the target machine´s Layer 2 reaction to spoofed frames (frames with incorrect source addresses), and the period and frequency of the subsequent ARP requests.

- And most recently, by observing ICMP responses from the target hosts [4, 5] which are, more often than not, a result of vendors non-adherence to RFCs.

More particular to fingerprinting IKE implementations is the work by Hills [62], which initiates a connection to an IKE daemon on the target machine, but does not complete the Phase 1 (Main Mode) handshake. Since not being able to complete the handshake can be legitimate behaviour of a client that is trying to connect on a particularly noisy connection, the server retransmits at preset intervals (known as the server´s `backoff strategy`. This type of fingerprinting, termed `NTA Backoff Pattern Fingerprinting` observes the particular manner in which a particular IKE daemon retransmits, and contrasts it to that of other IKE daemons, thus building an identifier or fingerprint.

1.4 Layout

Chapter 2 starts with a mention of Fingerprinting, its history and subsequent application to network traffic, and an overview of the two most effective and well-known tools for fingerprinting available today, provides a background to IPSec and its key-exchange protocols, and finishes off with an overview of SSL and TLS, the new player in the VPN arena. Chapter 3 presents the laboratory layout used for these tests, addresses the basics of packet capture and analysis, of path altering attacks, and explains the test methodology

developed and employed in this project. Chapter 4 presents the test results and findings; Chapter 5 depicts a typical attack scenario and presents ways in which to defeat fingerprinting. Finally Chapter 6 wraps things up with a summary and a glimpse into future work. Appendices A and B provide sample configurations on how to configure IPSec, and how to capture and analyse the resulting network traffic.

Chapter 2

Background

This chapter presents an overview of the core components upon which this study is based. The aim is to acquaint the reader with the concepts and terms that will be used later. In the first section, conventional definitions of forensics science are presented and contrasted with their more recent application to the digital world. The second section provides an introduction to VPN tunnels. Familiarity with the OSI protocol stack, as well as with Network Security concepts, is assumed on the part of the reader.

2.1 Forensic Investigation

With the concepts of ``Forensic Computing`` and ``Network Forensics`` finding more widespread use in Technical Reports [100, 68], Journal Papers [128, 27], magazine articles [70], academic papers [1], and books [107], it would serve us well to take a brief glance at the history of Forensic Science, in order to better understand why we place trust in it. The history of Forensic Science is filled with interesting anecdotes, some of them very recent. In a 2002 court case in the U.S. for example, federal district judge Louis Pollak surprisingly ruled that fingerprint evidence was not reliable enough to be admitted as evidence; and then later reversed his own ruling [39]. Is this a

recurring pattern in forensic science? The following brief introduction hopes to answer this question.

The birth of *forensics* came about in 44 BC when Antistius, a physician at Caesar´s court, upon examining the Emperor´s lifeless body declared that of all the stab wounds, the one that had fatally wounded Caesar was the one on his chest. The term *forensics* comes from the fact that Antistius made this announcement before the *forum*, a term used in ancient Rome to denote a public place ``forming the centre of judicial and public business`` [89]. Reports vary, but the generally accepted earliest known account of investigative forensic work, dates back to an account by Roman lawyer Quintilian who, in *The Major Declamations*, relates a case where a blind man was accused of using his sword to fatally wound his father. In spite of the sword being present in the wound at the time when the deceased was found, Quintilian (in 1000 AD) was able to prove, by means of the bloody handprints leading from the accused man´s bedroom to his father´s, that he had in fact been framed by his stepmother.

The earliest known written account on Forensic Medicine can be found in *The Washing Away of Wrongs* by Sung Tz´u (in 1247 AD), where he presents guidelines for determining the cause of death in cases where there is reasonable doubt. Up until this point, coroner work required no formal medical training. It would not be until 1926 that coroners in England would be required to have five years of experience in medical practice. During the same period, Northern Ireland required coroners to hold a law degree, in addition to a medical degree [64]. The existence of contours in human fingertips was discovered by the Italian physician Marcello Malpighi in 1686. This was further developed in 1823 by Jan Purkinje, of Czech origin, who identified nine distinct patterns, and created a classification system based on them. In 1880, Henry Faulds, a British physician working in Japan, published a paper where he put forward the idea of using fingerprints to link a suspect to a crime. This was the first recorded instance of the notion of forensic science

being used to solve crimes [29]. Hans Gross, made the first contribution to the field of forensics, from the Law establishment. Gross, a professor of Law at the University of Graz, published a paper in 1891, entitled *Criminal Investigation*, in which he explained the use of physical evidence in order to solve a crime [106]. In 1892 Sir Francis Galton published the book *Finger Prints* in which he contracted Jan Purkinje´s classification system from nine distinct patterns, to four. The years 1897, 1902, 1904 and 1946 all saw groundbreaking criminal convictions relying on matching fingerprints and other physical evidence as primary evidence [106]. Current fingerprint classification consists of comparing one set of fingerprints to another, according to a set of common characteristics, and then following this with an analysis of detailed *points* in the fingerprint. There is at present no agreement among experts as to the least number of matching *points* that conclusively identifies someone. In the Netherlands for example, the number of *points* that need to match is 12, whereas in South Africa, it is 7 [57].

More recently, 1984 saw Sir Alec Jeffreys develop the first DNA *fingerprinting* test, which was used in 1987, in separate cases in the U.S. and Britain respectively, to solve criminal cases. DNA fingerprinting, also referred to as DNA *typing*, is a method that ``allows for the unambiguous identification of the source of unknown DNA samples.`` This differs from previously used methods such as fingerprinting by means of blood type ``which can only exclude a subject`` in that DNA typing can ``provide positive identification with great accuracy`` [43]. The detail of what exactly constitutes the DNA *fingerprint* can be lost in a heap of acronyms and genetics jargon; however [43] summarises it as ``..the variations in DNA sequences between individuals as determined by differences in restriction enzyme cleavage patterns are known as restriction fragment length polymorphisms (RFLPs)[1]``. In a much-publicised case, DNA tests were conducted in 1994, on tissue samples of Franzisca Schankowska, a Polish lady wrongly believed to have been a

[1] A detailed description of the workings of DNA typing can be found in [84].

surviving member of the family of Tsar Nicholas II of Russia, Tsarina Anastasia, proving that she bore no relationship to the Tsar´s family. A recent *major milestone* in the field of Forensic Science has been the announcement in 1999 by the U.S. Federal Bureau of Investigations of their fully-functional, nation-wide database, the Integrated Automated Fingerprint Identification System (IAFIS). The IAFIS allows electronic submission and search of fingerprint information [106].

What is known today as *forensic science* is thus a product of over two thousand years of contributions from expert practitioners in the fields of Law, Law Enforcement, and Medicine. It can be argued that most of the advances have been made in the last 170-odd years. In fact, contributions to Forensic Science from the early 1800´s onwards, far outweigh those prior to that date. Taking this background into account, it can safely be concluded that our present-day reliance on the forensic process rests on a well-grounded process of scientific discovery.

2.1.1 Computer Forensics

With most things traditional finding expression in the digital world, it is not surprising that forensic science should also find application in the computer sciences. This happens as the natural progression of the scientific process of discovery. With institutions of learning acting as the guarantors of this process[2]; it thus carries the trust we placed in its more conventional form. The term ``Computer Forensics`` appears to have been coined in the late 1980s [128] or early 1990s [91], and it has been summarised by Keneally in [69] as referring to the ``tools and techniques [used] to recover, preserve, and examine data stored or transmitted in binary form.``

At first glance, it may appear that Computer Forensics is little more

[2]In addition to the large number of universities that have departments of *Forensic Science* or *Forensic and Investigative Sciences*, there is a growing trend for universities to present courses on *Digital Forensics* and *Computer Forensics.*

than the digital equivalent to its older sibling *Medical Forensics*. However these two branches of the same science bear not only similarities, but also differences. To begin with, they are both multidisciplinary in nature: Medical Forensics spans the fields of Medicine, Law, Criminology, and Law Enforcement whereas Computer Forensics covers the fields of Computing, Law, Criminology, Law Enforcement, Information Sciences, and Computer Engineering. The similarity here being that, as suggested by Yasinac et al, in [128], Computer Forensics covers technologically disparate fields[3]. In [69], Keneally also mentions two distinctions between these disciplines. Firstly, computer forensic tools and methods were not developed in controlled environments, with trial-and-error techniques, in the hope of inferring relevant facts, but rather to ``solve specific problems on known platforms within given parameters...``. In addition to this, [69] notes that computer forensic analysis has to deal with new challenges posed by the differences between physical and digital evidence: ``The mutable, fleeting, and intangible nature of digital evidence stands in contrast to the persistent physical features used in other disciplines.`` In fact, in [105] Rogers rather interestingly states that the ``eyewitness of today and tomorrow could be a computer generated `log file`.``

In practical terms, Computer Forensics deals with the reliable duplication of data at the crime scene, and its subsequent analysis. This can be as high-level as observing the file modification, access, and creation times (collectively known as MAC[4] times), or as detailed as combing through the hard disk in order to find data potentially hidden in unallocated or *slack* space.

[3]Computing and Law, for instance.

[4]Not to be confused with the Media Access Control protocol used by Ethernet, nor with Message Authentication Codes, where two parties sharing a key k can ensure the integrity of a message m by computing a `tag` g where the tag=g(k,m).

2.1.2 Network Forensics

In the BlackHat Briefings Asia 2000, one of the presentations was subtitled ``What We Already Know About Your Network.`` [9]. The presentation itself presented a model for describing Internet connected devices, in mock ASN.1 notation, it is however the title that is most intriguing: *What We Already Know About Your Network*. Could it be that information that is not intended to leave the corporate network, does in fact leak out? Are firewalls not meant to shield the internal network from outside attacks? Is information leaking *in spite* of all the expensive perimeter defence[5]? If so, what, by how much, and where exactly? More importantly: can it be stopped?

It is here that Network Forensics, and its collaborative associate, Operating System Fingerprinting become relevant. Being a relative newcomer to the `investigative computing` domain (web resources place the first use of the term, somewhere around the mid-to-late-90s), it is understandable that the relevant definitions are still developing. One source refers to it somewhat restrictively as *reconstructive traffic analysis* [27]. A more complete definition is found in [34]: ``Network forensics involves the analysis of real-time and post-hoc digital evidence of possible attacks or criminal activities perpetrated against, or executed using, a computer network.``

The work illustrated by the present author falls within the bounds of this definition. Although network forensic analysis is generally used in the context of `cleaning up after hacker attacks`, its scope is much wider. *Self-Defending Networks*, a concept introduced by Cisco Systems [21], the traffic processing and real-time vulnerability analyses employed by products like SecurVantage Studio by Securify [113] and the DeepSight Threat Management System by Symantec [33] all tie into *Network Forensics*. The coverage which this topic has received in academic literature attests to the large number of domains it spans.

[5]The term *perimeter defence* is used in network computing to denote defence measures that protect the entire outer edge of the network.

One active area of research for example is *IP Traceback*, which aims to devise ways of (accurately) tracking down the source of DoS and DDos attacks on the Internet. In [16], Burch and Cheswick propose a technique for doing just that. It assumes, however, that IP datagrams[6] travel along symmetric routes, an assumption that can for the most part, not be applied to the Internet. The reason for this is that the Internet has fault-tolerant and self-healing properties, such that if a router along a source-destination path goes down, there is rapid re-routing of datagrams to the destination. This behaviour provides for *asymmetric* routing paths. Several other papers for example, present new techniques to encode ``tracking`` information into the IP header [108, 117], and in special intermediate routers [116], fueled primarily by the upsurge in DoS activity. Another area where network forensic analysis is used is in studying the spread of Internet worms, many such studies are conducted by CAIDA´s *Network Telescope* [86]. Their analysis of the Code-Red worm [88] for example, shows that 359,000 network hosts were infected within 14 hours, and that at its peak, the infection rate was 2000 hosts per minute. The spread of the Sapphire worm (also known as the SQL Slammer worm) [87] was markedly worse: 100,000 hosts were infected in 10 minutes, and when it peaked (in 3 minutes), it was scanning 55 million IP addresses per second. Most recently when observing the Witty worm, CAIDA researchers found that 110 hosts were infected within 10 seconds, and 160 within 30 seconds. The chances of the worm spreading this fast are ``vanishingly small – worse than $10^{-607}$`` [115]; and therefore pointed to a number of `seeded` machines which had been previously compromised, and prepared to release the worm in tandem. These studies represent monitoring and analysis advances in the field of network forensics.

Up until 1998, determining what OS ran on any particular networked de-

[6]To employ the terminology used by Stevens in [120], ``The unit of data that TCP sends to IP is called a TCP *segment*. The unit of data that IP sends to the network interface is called an IP *datagram*. The stream of bits that flows across the Ethernet is called a *frame*.``

vice, be it a PC, a network-connected printer, or any device with a network connection, depended on a number of rudimentary approaches. For example, one might connect to the known (or guessed) service on the target machine, expecting to be greeted by a descriptive `banner` reading for example: ``RICOH Maintenance Shell. User Access Verification. Enter Password:``. This method known as *banner grabbing* trusts the information entered statically in the `/etc/banner` file (in UNIX variants) by the administrator, but since the contents of this file can be modified to display any message, this does not constitute a reliable method of fingerprinting. Another approach would be to issue particular commands once connected to the target machine: for example one could issue a `GET` request when connected to the HTTP port, or an FTP `SYST` command when logged-in as an anonymous user to an FTP server. Yet another approach is to use SNMP (by doing an *snmpwalk* on the target machine, for instance). Finally, one could simply check the target host or network´s Domain Naming System (DNS) `INFO` records. These methods, provided they worked at all, could provide a coarse level of granularity; that is, they could tell one that a machine is running `OpenBSD` and not `IRIX`, and perhaps a limited amount of version information held statically on files modifiable by the administrator (such as `/etc/issue` in UNIX variants), but they could not provide any further detail.

OS Fingerprinting was further refined in 1998, when Fyodor Yarochkin introduced *nmap*, and with it, made familiar the idea of determining a networked system´s OS with a reasonable level of accuracy. Yarochkin´s particular strategy for doing this is by ``querying its TCP/IP stack`` [127], a process referred to as *TCP/IP stack fingerprinting*. It should be noted that at a few other TCP/IP stack fingerprinting programs existed prior to *nmap*, such as `checkos`, `Su1d` and `Queso`; however *nmap* greatly expanded on their capabilities, and introduced the concept of a *fingerprint file*, where each OS´s TCP/IP peculiarities could be recorded (prior to this, they were hard-coded into the program). *Nmap* introduced a new range of tests

that enabled one to determine, with a greater level of accuracy than previously possible, what OS the target system was running. These new tests were mostly TCP packets with infrequently or incorrectly-used TCP flags set (i.e., TCP flags used in an incorrect combination), IP datagrams with a non-standard combination of flags, or even IP datagrams with legitimate flags set in an inconsistent manner: for example, some OSs set the DF (Don´t Fragment) bit, and others don´t; more interestingly, some OSs set the DF bit in particular cases, and don´t set it in other cases [127], and also some ICMP tests, such as ICMP Error Message Quenching, which differentiates between the rate different OSs´ send ICMP error messages. In general, *nmap* developed the idea of observing different OSs´ deviations from the RFC standards published by the IETF. And so, *OS Fingerprinting* entered the mainstream[7].

Within a short span of time, other fingerprinting programs and methodologies emerged, most of which worked by sending queries, and observing non-standard responses. In 1999 Krishnamurthy and Arlitt conducted a study to determine various Web-sites´ compliance to the HTTP/1.1 standard, and although this does not constitute *OS Fingerprinting*, it is one of the studies that paved the way for using RFC non-compliance in fingerprinting [75]. It is around this time that the concept of non-obtrusive or *passive* fingerprinting was born. The difference between *active* and *passive* fingerprinting is that in the former, probes are launched against the target system, and its responses observed, whereas with the latter, no probes are sent; the network is `tapped` by means of a tool like *tcpdump* or *snoop*, and information is inferred by simply observing the target host´s network etiquette. Nazario in [90] attributes its first use to a hacker who in mid-1999, and writing under the handle `Photon`, ``posted the nmap-hackers list with some ideas of passive operating system fingerprinting``. In quick succession, papers were written and tools appeared, which all revolved around this idea of passive

[7]At present, *nmap* can accurately identify over 500 operating systems and variations thereof.

OS fingerprinting. Notable amongst them are the papers by Nazario [90], Spitzner [118], and Zalewski´s paper explaining his passive fingerprinting tool *p0f* [130].

Active fingerprinting has its limitations however, as noted in [90]. For example, two problems with *active* fingerprinting methods are that firstly, the probing packets can be firewalled, limiting a probe´s effectiveness and secondly that the probing packets can be detected, potentially raising an alarm. *p0f* (now in its latest version *p0f v2*) sports an array of features, such as the ability to detect load-balancers, or the ability to detect machines hidden using NAT and IP masquerading (it does this cleverly, by determining if there are various OS fingerprints for the same IP address). *p0f* also attempts to determine the link type (i.e., Ethernet/modem, GPRS), deducing it from the link´s MTU. As *p0f* is not active, it does not send probes like *nmap*, but rather observes the behaviour of machines that connect to (or through) the host machine, machines that the host tries to connect to, but with which it cannot establish a connection, and machines communicating in proximity to the host machine[8].

In 2001, Arkin expanded on the concept of fingerprinting using ICMP request/responses [5]; a concept that had been initially addressed in [127]. Together with this article, Arkin presented the *Xprobe* tool (now in its latest version *Xprobe2*), which complements Yarochkin´s *nmap*. Initially Arkin discerned different OSs by observing a few of their digressions from the ICMP RFCs. For instance, RFC 1349 specifies that ``an ICMP reply message is sent with the same value in the IP Type of Service (TOS) field as was used in the corresponding ICMP request message.`` (The IP TOS is denoted by an 8-bit field in the IP header.) [2], yet some OSs reply to an ICMP query with a TOS value that does not correspond to what they received in the ICMP query. Another more comical observation in [5] was that Solaris engineers may well have mis-read RFC 792, which specifies that ``The internet header

[8] *p0f* can accurately fingerprint over 200 operating systems.

plus the first 64 bits of the original datagram´s data`` should be included an ICMP error message: Solaris 2.x however, includes the first 64 *bytes* (512 bits) of the original datagram´s data. *Xprobe2* has greatly expanded on the functionality and effectiveness of the original *Xprobe*; it is modular in design, and employs fuzzy logic to correlate the responses in order to determine what the probed OS is. *Xprobe2* also adds a TCP-based OS fingerprinting module, in order not to rely solely on ICMP exchanges, since ``Focusing on one niche only is not useful in the long run`` [6].

In 2002 Bordet published a paper on fingerprinting SMTP servers [12]. His method worked by sending valid and invalid commands to the target mail server(s) and observing to what extent their reactions digressed from what was specified in the relevant RFCs[9]. For example, connecting to a mail server, sending a valid `MAIL FROM` and not issuing a `HELO` first is allowed by some servers and disallowed by others. As another example, using a `MAIL FROM: <>` with an empty address field should be allowed (as per the RFCs), but some mail servers disallow it. Bordet´s work adds to the body of knowledge in the Network Forensics field, by presenting a method of fingerprinting based on a known service. In 2002 the *Intranode Research Team* published a paper (and accompanying proof-of-concept tool called RING) delineating a new method of OS fingerprinting. They called it fingerprinting based on *temporal analysis*. It introduces an interesting and previously unexplored idea, that of observing a target host´s behaviour in the face of TCP congestion [123]. RFC 793 suggests a behaviour that hosts must observe when faced with delays, for example, the TCP standard specifies that ``In the absence of knowledge about the sequence numbers used on a particular connection, the TCP specification recommends that the source delay for MSL seconds before emitting segments on the connection, to allow time for segments from the earlier connection incarnation to drain from the system`` [101]. As with most

[9]The RFCs involved in his study were: RFC 821 (*Simple Mail Transfer Protocol*), RFC 1425 (*SMTP Service Extension*), and RFC 1985 (*SMTP Service Extension for Remote Message Queue Starting*).

other approaches discussed above, observing an implementation´s digression from the RFC may reveal information about the target system.

A summary mention should be made of the latest newcomer to the OS fingerprinting foray: RPC Fingerprinting. First presented at Black Hat Windows Security Briefings in Seattle, it is an idea developed by Seki. Its novelty lies in that it uses RPC signatures for determining an OS´s identity. Seki´s method allows discerning between various flavours of the Windows operating system, and even within different service packs installed. The study is presented in greater detail in [114].

For the sake of completion, it should be said that taking into account the work done up till now in the OS fingerprinting arena, our use of *network forensic analysis* can then justifiably extend to incorporate OS Fingerprinting, and still stay well within the bounds of its definition as the ``analysis of ... digital evidence ... against ... or using, a computer network`` as put forward in [34].

2.2 VPNs and Tunnels

As this study focuses on the fingerprinting of ``encrypted tunnel endpoints``, it is important to note the difference between *encrypted tunnel* and *non-encrypted tunnel* endpoints. The term `tunneling` and the phrase `protocol encapsulation` both refer to wrapping one protocol data unit (or *datagram*) within another. Recalling that these network protocol data units consist of a *header* portion, containing steering and control information such as *source address*, *destination address*, and *checksum*, and a data or *payload* portion, we can introduce new terminology: let the datagram being wrapped be termed the encapsulat*ed* datagram; and let the datagram that will be transporting the encapsulated datagram be termed the encapsulat*ing* datagram. It can then be said that tunneling is the process whereby the entire encapsulated datagram becomes the payload of the encapsulating datagram, and is

transported through the network using the encapsulating datagram´s header information. This can be used to describe network tunnels in general, and encrypted tunnels as a subset thereof. Adapting slightly from the notation used by Aqun et al, in [3], this can be described as: [Y[tunnel header[X]]], where Y is the encapsulat*ing* protocol, and X is the encapsulat*ed* protocol. We can then say that an *encrypted tunnel* differs from a *non-encrypted tunnel* in that in the former, the encapsulated protocol is encrypted, whereas in the latter, it is not. This definition has been diagrammatically represented in Figure 2.1. Figure 2.2 shows a practical representation of this definition: IPSec operating in tunnel mode.

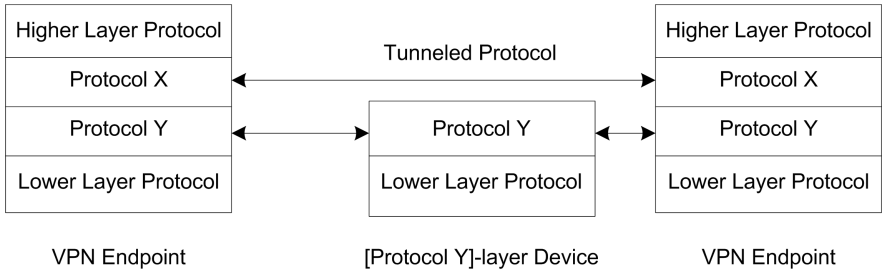


Figure 2.1: Definition of encapsulat*ed* and encapsulat*ing* protocols, adapted from [3].

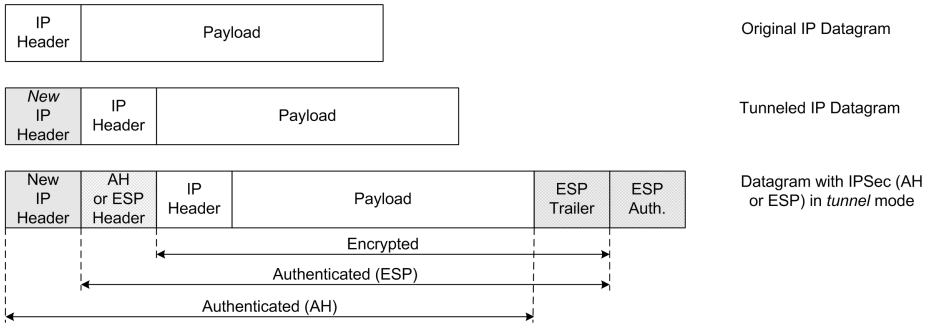


Figure 2.2: IPSec operating in tunnel mode, adapted from [32].

There are many different types of encapsulation.[10] Virtual Private Net-

[10] [60] lists no less than 258 RFCs that deal with encapsulation (although it must be

works (VPNs) include network protocol encapsulation at layers 4 [10, 122], 3 [56, 46, 94, 71] or 2 [55, 121] of the OSI stack. As explained in [8], a VPN incorporates *virtual networking* (organizing a geographically dispersed set of users into the same `virtual` group, allowing them access to the same set of network resources) and *private networking* (using cryptography to ensure integrity and confidentiality as data traverses untrusted networks). As this study applies to the forensic analysis of encrypted traffic from remote connectivity devices (VPN appliances, as well as VPN implementations in various OSs), its focus will be on IPSec and SSL VPNs — the two prevailing end-user VPN technologies. The other widely-employed type of VPN is the MPLS VPN, however this type of VPN is used for traffic management and not for providing user privacy and confidentiality. It thus falls out of the scope of this study.

2.2.1 IPSec VPNs

The IPSec standard consists of a suite of protocols defined in a number of RFCs (RFCs 2401-2412, 2764) and IETF Drafts[11]. It is a series of security enhancements which are present by default in the IPv6 standard. Due to IPv6´s slower-than-expected rollout however, an IETF Working Group was set up with the aim of developing security *extensions* that would work with IPv4 [8, 40]. A detailed, and often referenced study of IPSec is available in [47]. In their study, Ferguson and Schneier´s principal criticism of the standard was its complexity, noting (among other things) that ``A more complex system loses on all fronts. It contains more weaknesses to start with, it is much harder to analyze, and it is much harder to implement without introducing security-critical errors in the implementation`` and that ``any single weakness can destroy the security of the entire system.``

IPSec can operate in either `transport mode` or in `tunnel mode`, and

granted that there are a few ``April Fool´s`` RFCs in the list).

[11]The complete list of drafts is available on http://www.ietf.org/ids.by.wg/ipsec.html

with either of the two defined protocols, *Authentication Header* (AH, IP protocol 51) which provides authentication but no encryption, and *Encapsulation Security Payload* (ESP, IP protocol 50) which provides encryption, but makes authentication optional. From a cryptographic point of view, Schneier notes that ``in virtually all cases, encryption without authentication is not useful`` and recommends that ``ESP authentication always be used, and only encryption be made optional`` [47]. In transport mode the original IP payload is encrypted, and an IPSec header is placed immediately following the original IP header; in the case of transport mode with ESP, a trailer is also appended to the IP datagram. This means that the original IP header is not encrypted, and can thus be used in traffic analysis attacks. In tunnel mode, as shown in Figure 2.2 the original IP datagram *in its entirety* forms part of a new IP datagram, with either the ESP or AH transform applied. Transport mode is meant for communication between hosts, not traversing gateways[12], whereas tunnel mode is meant for use between gateways. As stated in [47], ``This creates a lot of extra complexity: two machines that wish to authenticate a packet can use a total of four different modes: transport/AH, tunnel/AH, transport/ESP ..., and tunnel/ESP ...``. Instead of evaluating IPSec, however (a task which has been done to an exhaustive level of detail in [47]), the present study focuses on the tunnel establishment and tear-down procedure, more precisely on IKE, the key-exchange protocol used in IPSec.

The history of IKE´s development is peppered with political squabbling, and even after its release as an RFC, it has been the subject of harsh criticism. It derives from several other *key management* protocols: the *Internet Security Association and Key Management Protocol* or ISAKMP, Oakley, Photuris, and SKEME. Often considered unsuitable to fulfill the task for which it was designed, the IETF convened a working group to `fix` the

[12]Gateways are actually not required to support transport mode, although many of them do [32].

shortcomings present in IKE; this was named the Son of IKE (SOI), and later IKEv2. The present overview of IKE has thus been divided into IKEv1 and IKEv2. Since this study will only cover working IPSec implementations (IKEv1 implementations, since IKEv2 has not been ratified as a standard as of yet, and thus should not have found its way into any consumer products.) the IKEv1 overview below will be more extensive than that of IKEv2.

IKEv1

IKE consists of two distinct phases of operation, named *IKE Phase 1* and *IKE Phase 2*. Phase 1 establishes an ISAKMP Security Association (SA). Phase 2 uses this SA to derive keying material, and set up IPSec SAs. An SA is ``the method by which traffic traveling between two end points will be protected`` [40]. There are two kinds of SAs in IPSec talk: ISAKMP SAs (sometimes referred to as IKE SAs), and IPSec SAs. IPSec SAs are simplex, or unidirectional (that is, two of them have to be set-up for any duplex connection), whereas ISAKMP SAs are bidirectional. In short, an IKE Phase 1 exchange establishes an encrypted (*secure*) channel which is then used in the negotiation of the IPSec SAs (in Phase 2). Since this study is not a cryptographic analysis of IKE, but rather a forensic analysis of the tunnel establishment and tear-down processes, the emphasis will be on the Phase 1 exchanges, as Phase 2 traffic is already encrypted and unrecognizable to the forensic examiner — it is, so to speak, too late in the game.

Phase 1 has two ``modes`` [58]: *aggressive* mode and *main* mode. Main mode consists of 6 messages: messages 1 and 2 negotiate the cryptographic protocols and parameters, messages 3 and 4 conduct the Diffie-Hellman (DH) exchange, and messages 5 and 6 provide Perfect Forward Secrecy, or PFS. Aggressive mode consists of 3 messages, since it does not offer PFS. This might appear counterintuitive, as it may seem logical that if PFS requires 2 messages, and aggressive mode does not provide PFS, then the aggressive mode exchange should be 4 messages (6 - 2 = 4). This is not the case,

however. In aggressive mode, the Diffie-Hellman exchange takes place in the first two messages, and the second and third messages serve for each side to prove that they know the DH value and their respective secret [98] (that is, the second message carries more information in aggressive mode than it does in main mode). Phase 2 is referred to as Quick Mode, and is an exchange consisting of only three messages, all of which are encrypted.

Key types have been specified in the RFC for each of these modes, these are: pre-shared secret keys, public key signatures, and two variants of public encryption (Public Key Encryption, and Revised Public Key Encryption), bringing to 8 the total number of variants of Phase 1 [98]. As Perlman [96] mentioned, one of IKE´s biggest problems is its ``terminal complexity``, another is the fact that it is ``very inefficient``.

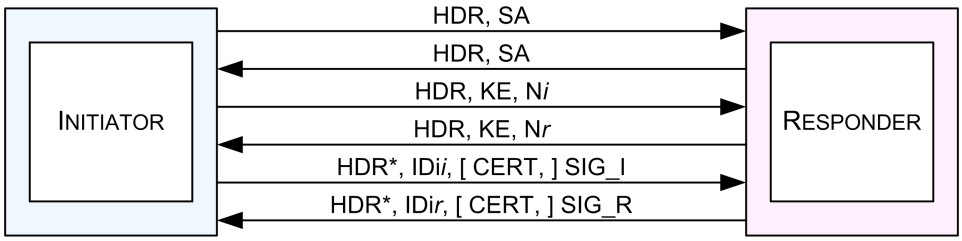


Figure 2.3: IKE Phase 1 Main Mode using Signatures, as per RFC 2409 [58].

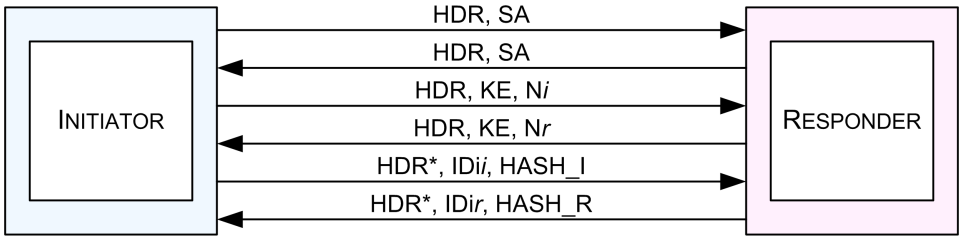


Figure 2.4: IKE Phase 1 Main Mode using Pre-Shared Keys (PSKs), as per RFC 2409 [58].

IKE Phase 1 Main Mode implemented using Signatures and Pre-Shared Keys is shown in Figures 2.3 and 2.4 respectively. The difference can be ob-

served in messages five and six of the exchange, where the signature payload (SIG) in the first figure is replaced by the hash payload (HASH) in the second.

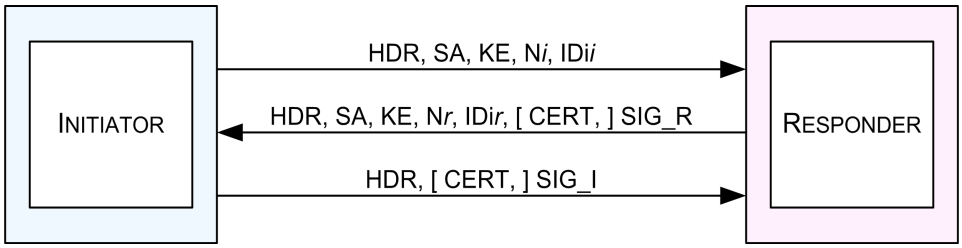


Figure 2.5: IKE Phase 1 Aggressive Mode using Signatures, as per RFC 2409 [58].

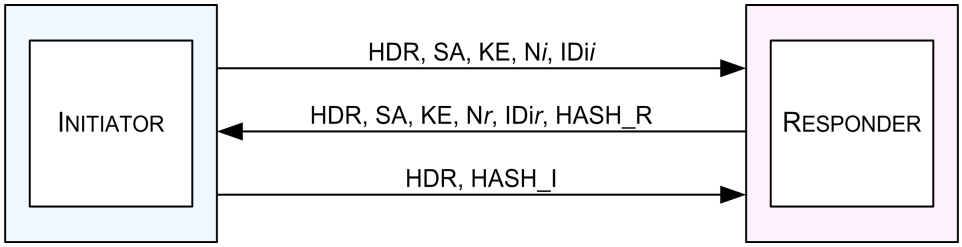


Figure 2.6: IKE Phase 1 Aggressive Mode using Pre-Shared Keys (PSKs), as per RFC 2409 [58].

IKE Phase 1 Aggressive Mode is shown in Figures 2.5 and 2.6 implemented using Signatures and Pre-Shared Keys respectively. RFC 2409 states that SIG_I or SIG_R (where I and R start for *Initiator* and *Responder*) is the result of the negotiated digital signature algorithm applied to HASH_I or HASH_R respectively [58]. The computation of both HASH_R and HASH_I relies on SKEYID, which is ``a string derived from secret material known only to the active players in the exchange``. The procedure to be followed in order to obtain the three variations of SKEYID (SKEYID_a, SKEYID_d, and SKEYID_e, for authentication, derivation, and encryption respectively), together with the procedure to obtain the HASH values for Signatures and Pre-Shared Keys, is shown in Table 2.1.

For signatures: `SKEYID = prf(Ni_b | Nr_b, g^xy)` For pre-shared keys: `SKEYID = prf(pre-shared-key, Ni_b | Nr_b)`

`SKEYID_d = prf(SKEYID, g^xy | CKY-I | CKY-R | 0)`
`SKEYID_a = prf(SKEYID, SKEYID_d | g^xy | CKY-I | CKY-R | 1)`
`SKEYID_e = prf(SKEYID, SKEYID_a | g^xy | CKY-I | CKY-R | 2)`

`HASH_I = prf(SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b | IDii_b )` `HASH_R = prf(SKEYID, g^xr | g^xi | CKY-R | CKY-I | SAi_b | IDir_b )`

Table 2.1: `SKEYID` and `HASH` generation for Signature and Pre-Shared Key modes in IKE, as per RFC 2409 [58].

IKEv2

The inherent problems in the IKEv1 design brought about the formation of the IKEv2 WG. Their document *Design Rationale for IKEv2* (which together with all other IKEv2 WG documents is currently in IETF draft form) explains the design choices on which the IKEv2 standard was based. Regarding backwards compatibility with IKEv1, IKEv2 ``does not interoperate with version 1, but it has enough of the header format in common that both versions can unambiguously run over the same UDP port`` [42]. IKEv2 furthermore ``preserves most of the features of the original IKE, including identity hiding, perfect forward secrecy, two phases, and cryptographic negotiation, while greatly redesigning the protocol for efficiency, security, robustness and flexibility`` however although it is a ``major redesign of IKEv1``, IKEv2 is not ``backwards compatible with IKEv1`` [59]. In contrast to IKEv1, which was able to (in theory) do identity hiding from active attackers, IKEv2 is able to do identity hiding ``of both parties, from passive attackers``. IKEv1´s public key encryption modes were seen to provide very similar support to the other two modes, so IKEv2 only supports public signature keys and preshared keys [97]. Another enhancement that IKEv2 has introduced is protection against the ``polling attack``, where the initiator/attacker opens a connection to the target IP address, simply to find out the endpoint´s identity [97]. IKEv2 drops the `Phase 1 SA` and `Phase 2

SA` wording to rather refer to the two SA-creation steps as the IKE-SA and the CHILD-SA exchanges. These exchanges are shown in Figures 2.7 and 2.8

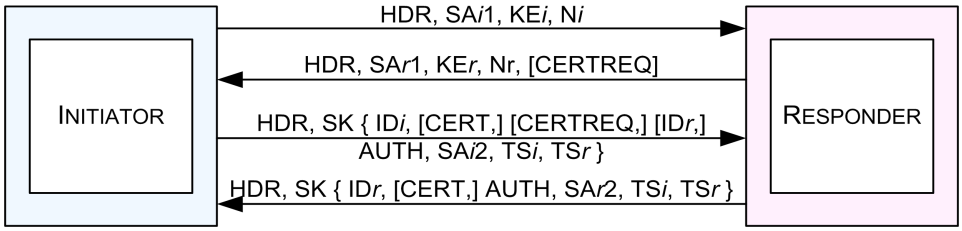


Figure 2.7: IKEv2 Initial Exchange, as per draft-ietf-ipsec-ikev2-17 [41].

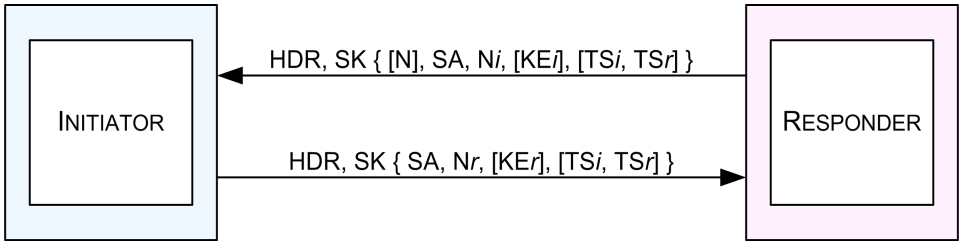


Figure 2.8: IKEv2 Child-SA Creation Exchange, as per draft-ietf-ipsec-ikev2-17 [41].

2.2.2 SSL/TLS VPNs

SSL/TLS VPN technology found widespread acceptance between 2001 and 2003 when a large number of network device vendors started offering SSL/TLS VPN-capable or dedicated devices[13], with some vendors having been offering SSL VPN solutions since 1997 [7]. Market research companies like Frost & Sullivan, Gartner, and Infonetics Research forecast that the worldwide SSL/TLS VPN sales are set to grow steadily reaching the $1 billion mark

[13] At present these are: Array Networks - Array SP, Aventail EX-1500 SSL VPN, Cisco VPN 3000 Series Concentrator, F5 Networks Firepass 1000 and Firepass 4000, Juniper NetScreen SA 5000 Series, NetScaler 9400, Netilla Security Platform (E-Class), Nokia SSL-VPN Secure Access System, Nortel Atleon SSL VPN, PortWise mVPN, Symantec Safeweb SSL VPN, amongst others.

by 2008 [19, 52]. SSL/TLS VPN technology offers an alternative to Small and Medium Enterprises (SMEs) that do not have need for the full site-connectivity service offered by IPSec VPNs. Before comparing IPSec and SSL VPNs however, it is important to note that they were developed to address different needs, and that they can be considered complementary technologies. As explained by Perlman and Kaufman in [98], ``The goal of SSL was to deploy something totally at the user level, without changing the operating systems, whereas the goal of IPSec was to deploy something within the OS and not require changes to the applications.``

In contrast to IPSec VPNs which operate at layer 3 of the OSI stack, and are thus able to provide security at the network protocol layer, SSL/TLS VPNs operate at the transport layer [14] and can provide granularity on an per-application level. Two of the most common criticisms leveled against IPSec VPNs (which SSL/TLS VPNs don´t suffer from) is the complexity involved in rolling-out and managing the VPN client applications, and each product´s foibles that make interoperability difficult. In fact, in a 2002 report, ICSA Labs which conducts VPN interoperability testing and ``certifies over 95 percent of the installed base of firewall and AV products in the marketplace`` [65] stated that ``the [IPSec] VPNs biggest black eye in the industry today is the lack of interoperability between products`` and further lightheartedly remarked that ``trying to integrate functionality between two disparate VPN gateways is like trying to nail Jell-O to a wall.`` SSL/TLS VPNs, on the other hand, provide a high degree of interoperability, since there is no *VPN client* to be installed on the client machine; all that is required is a web browser that supports SSL/TLS. The underlying protocols that make SSL/TLS VPNs possible are SSL 3.0 and TLS 1.0, but not all vendors support TLS as a protocol option, thus making their products *SSL VPNs* and not *SSL/TLS VPNs*. By mid-2004 there were only 6 SSL/TLS VPN products that had

[14] At least one vendor has chosen to refer its SSL VPN offering as being a *Layer 7 SSL VPN* [18]

been certified `interoperable` by ICSA Labs[15]

SSL

Secure Sockets Layer (SSL) is a patented protocol, owned by Netscape Communications Corporation. It started as SSL version 1.0 in 1993 (this version of the standard was never released), went through a revision in 1994 as SSL 2.0 and reached its final form in 1995 with the release of the SSL 3.0 standard. SSL 2.0 had a number of flaws, such as a weak 40-bit export (non U.S.) version, a weak MAC construction, and an unauthenticated field denoting the MAC padding length. It was also vulnerable to the ``cipher-suite rollback`` attack, which allowed an attacker to force weaker encryption to be used. [125]. The SSL protocol is layered: the *Record Protocol Layer* provides ``confidentiality, authenticity, and replay protection over a connection-oriented reliable transport protocol such as TCP`` [125]; the *Handshake Protocol* does SSL´s key exchange. Over the years, there have been numerous security advisories concerning SSL, mostly vendor-implementation related. The Computer Emergency Response Team Coordination Center (CERT/CC) database at Carnegie Mellon for instance, which houses one of the largest collections of security advisories and new vulnerabilities world-wide, listed 295 SSL-related entries.

Notable among these are the attacks by Klima, Pokorny, and Rosa [72], which exploits SSL/TLS alert messages sent as a result of incorrect plaintext lengths or incorrect SSL/TLS numbers in the plaintext, and remote timing attacks such as the one conducted Brumley and Boneh [14], where the authors ``were able to extract the SSL private key from common SSL applications such as a web server (Apache+mod_SSL) and a SSL-tunnel.`` Schneier, in a recent issue of his monthly *Cryptogram* presented his view on

[15]These are: Aventail´s EX-1500 SSL VPN Appliance, F5 Networks´ Firepass 1000 and Firepass 4000, Juniper´s NetScreen Secure Access SSL VPN Appliance, Netilla Networks´ Netilla Security Platform (E-Class), Netscaler´s NetScaler 9400, and PortWise AB´s PortWise mVPN [65].

recent SSL protocol weakness discoveries, saying that ``Even if SSL were irrevocably broken, it wouldn´t affect Internet security very much. There are two reasons. One, SSL is almost never used in a secure manner. And two, SSL doesn´t solve an important security problem.`` Schneier then elaborated saying that although SSL protects the *channel* between the client and the server, the problem often lies at the endpoints (for example key loggers on the client capture credit card information, or vulnerable servers allow credit card data to be stolen in bulk). He then concluded saying, ``Security is only as strong as the weakest link, and SSL is nowhere close to being the weakest link.`` Schneier and Wagner provide a thorough analysis of security concerns surrounding SSL 2.0 and SSL 3.0 in [125].

Figure 2.9: SSL Handshake for establishing a new session, as per draft-freier-ssl-version3-02 [50].

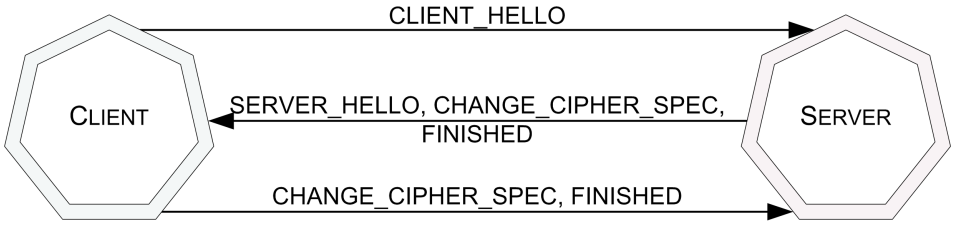


Figure 2.10: SSL Handshake for resuming an existing session, as per draft-freier-ssl-version3-02 [50].

The SSL handshake is described in Figures 2.9 and 2.10

TLS

SSL and TLS are not interoperable. This means that a server running SSL cannot establish an encrypted session with a client using TLS. Both must run either SSL or TLS. Figure 2.11 shows the TLS 1.0 handshake, where the differences between it and the SSL handshake can be clearly seen in messages 2 and 3. The reason why SSL and TLS are not interoperable is that SSL (2.0 and 3.0) use MD5 for generating keying material, whereas TLS uses a combination of SHA1 and MD5 for generating keying material. The different keying material generation processes are shown in Tables 2.2, 2.3, and 2.4. What adds emphasis to the differences between SSL 2.0, SSL 3.0, and TLS 1.0, is the problem concerning their FIPS-140-1 validation. A Federal Information Process Standard or *FIPS* specification is issued by the United States Government National Institute of Standards and Technology (NIST) for use by the US Government and related agencies. Protocols require FIPS compliance before being employed by the US Government and its related agencies[16]. SSL 2.0 and SSL 3.0 did not pass FIPS-140-1 compliance testing, whereas TLSv1 did. On a more detailed level, the reason was the different hashing algorithms used in keying material generation. MD5 is not a FIPS-approved standard, whereas SHA1 is. Table 2.2 shows how SSL 2.0 generates keying material by simply taking an MD5 hash of some data, Table 2.3 shows that SSL 3.0´s keying material generation process is more complex, and that even though a SHA1 hash is used, the keying material still relies on MD5. The TLS keying material generation is different: as can be seen in Table 2.4, ``TLS 1.0 uses a MD5 and SHA1 hash of some secret data XOR´ed together for key generation.``. SHA1 hashes are larger than MD5 hashes, therefore ``each bit of the MD-5 hash is XORed with a different bit in the SHA1

[16]It is important at this time to note that the reason why reference is being made to the US Government and its agencies, is purely historical: the Internet was originally developed as a project for the US Department of Defense, therefore many of the protocols and standards developed and used within the context of the Internet, can trace their lineage to the US Government.

hash.``. The result of this is that keying material generation relies on the SHA1 algorithm´s strength, and not on that of the MD5 algorithm. This allows SHA1 and therefore TLS 1.0, to pass FIPS-140-1 compliance testing.

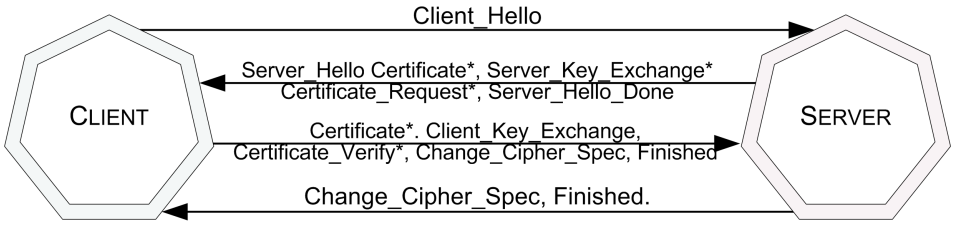


Figure 2.11: TLS Handshake as per RFC 2246 [37].

MASTER-KEY =	SECRET-KEY-DATA + CLEAR-KEY-DATA
EY-MATERIAL-0 =	MD5[MASTER-KEY, "0", CHALLENGE, CONNECTION-ID]
KEY-MATERIAL-1 =	MD5[MASTER-KEY, "1", CHALLENGE, CONNECTION-ID]
KEY-MATERIAL-2 =	MD5[MASTER-KEY, "2", CHALLENGE, CONNECTION-ID]

Table 2.2: Routine for generating keying material in SSL 2.0 [38].

PKI

No overview of SSL/TLS would be complete without mention of the Public Key Infrastructure (PKI), primarily because both SSL and TLS offer several modes for key exchange, and the use of digital certificates is only one of them (SSL 3.0 supports RSA, DiffieHellman, Fortezza, and digital certificates, whereas TLS supports RSA, DiffieHellman and digital certificates). However both SSL and TLS are predominantly used in a web environment and use of digital certificates for key exchange is the most convenient and thus the most prevalent. With public key cryptography finding widespread use in networking protocols (i.e., IPSec, SSL) and applications (i.e., PGP,

```
master_secret =
MD5(pre_master_secret + SHA('A' + pre_master_secret +
ClientHello.random + ServerHello.random)) +
MD5(pre_master_secret + SHA('BB' + pre_master_secret +
ClientHello.random + ServerHello.random)) +
MD5(pre_master_secret + SHA('CCC' + pre_master_secret +
ClientHello.random + ServerHello.random));
```

```
key_block =
MD5(master_secret + SHA(‘A’ + master_secret +
ServerHello.random +
ClientHello.random)) +
MD5(master_secret + SHA(‘BB’ + master_secret +
ServerHello.random +
ClientHello.random)) +
MD5(master_secret + SHA(‘CCC’ + master_secret +
ServerHello.random +
ClientHello.random)) + [...];
```

Table 2.3: Routine for generating keying material in SSL 3.0 [38].

```
PRF(secret, label, seed) = P_MD5(S1, label + seed) XOR
P_SHA-1(S2, label + seed);
```

```
master_secret = PRF(pre_master_secret, "master secret",
ClientHello.random + ServerHello.random)
```

```
key_block = PRF(SecurityParameters.master_secret, "key expansion",
SecurityParameters.server_random +
SecurityParameters.client_random);
```

Table 2.4: Routine for generating keying material in TLSv1 [38].

GPG, S/MIME), all of which assume communication with another party, the spotlight fell on practical discovery and validation of public keys [95, 78]. Public key cryptography, however has not solved the problems of ``public-key acquisition, recognition, revocation, distribution, re-distribution, validation and, most importantly, key-binding to an identifier and/or key-attribution to a realworld entity [51].``

PKI, based on X.509/PKIX digital certificates and Certification Authorities (CAs) offers a scalable architecture for key distribution and validation

by means of certificates and certificate revocation lists (CRLs). PKI can be defined as ``the entire coherent collection of protocols, technologies, and written policies that define how an organization maintains, distributes, creates, and validates public keys and their associated identification information [20].`` Its success is due in part due to its ease of use, supporting the principle that ``Security needs to be convenient, or users will circumvent it [95].`` The PKI model proposes a method of implementing a chain-of-trust by making use of a trusted third party; this model is different from the referral or collaborative methods found in the PGP and SKIP models respectively [51]. The trusted body (Certification Authority) is at the top of the chain of trust (there can be more than one such ``CA root`` body along a certification path, one validating the other). Before issuing a certificate to an individual or organisation, it verifies that the information corresponding to the individual/organisation is correct, and then signs this verified information. This becomes a *digital certificate* that can be associated with a person or organisation. Upon receipt of a certificate, the receiver checks the validation path of the certificate, and accepts it if the certificate is still valid. There are a number of root CAs: EuropePKI, GlobalSign, SwissSign, Thawte, and VeriSign, to name but a few.

The PKI model has a number of shortcomings however, summarised in Ellison and Schneier´s article entitled ``Ten Risks of PKI: What You´re not Being Told about Public Key Infrastructure`` [44]. For the sake of brevity, only a few of the points are reproduced below. The entire list is available in [44]:

- *Who do we trust and for what?*: From whom does the CA derive its authority? Who made it trusted?
- *Who is using my key?*: What happens if a virus infects a computer and signs something without the owner´s knowledge?
- *Is the CA an authority?*: Who granted the authority to an SSL CA

to control that permission? Is the control of that permission even necessary? No harm is done if an uncertified server were allowed to use encryption.

- *How secure are the certificate practices?*: Certificates are not like some magic security elixir, where one can just add a drop to your system and it will become secure. Certificates must be used properly if one wants security.

One of Schneier´s often-quoted critiques of PKI revolves around the secure channel being established with an unknown entity: ``when you establish an SSL connection with your browser, there´s a visual indication that the SSL protocol worked and the link is encrypted. But who are you talking securely with? ..but if you don´t look, its much like going into a private room with the lights off: you might know that someone else is there and your conversation is private, but until you know who that other person is, you shouldn´t reveal any secret information`` [44, 111].

Another unrelated critique of PKI revolves around the amount of information leaked by certificates. Persiano and Visconti, in [99] present concerns regarding the amount of personal information that is leaked as a digital certificate traverses the network in clear form. Two examples they offer are firstly of an attacker who can, for example eavesdrop on the connections to a site and build a database of users who connect, and secondly of the manager of a public database who can discern what data has been accessed by each individual user; which could constitute a violation of privacy [99]. Both of these run counter to the notion of security and privacy.

This concludes our overview of the core technologies discussed in future chapters.

```
HDR is an ISAKMP header whose exchange type is the mode. When written as HDR* it indicates
payload encryption.

SA is an SA negotiation payload with one or more proposals. An initiator MAY provide multiple
proposals for negotiation; a responder MUST reply with only one.

<P>\_b indicates the body of payload <P> -- the ISAKMP generic payload is not included.

SAi\_b is the entire body of the SA payload (minus the ISAKMP generic header)-- i.e. the DOI,
 situation, all proposals and all transforms offered by the Initiator.

CKY-I and CKY-R are the Initiator's cookie and the Responder's cookie, respectively, from the
ISAKMP header.

g^xi and g^xr are the Diffie-Hellman ([DH]) public values of the initiator and responder
respectively.

g^xy is the Diffie-Hellman shared secret.

KE is the key exchange payload which contains the public information exchanged in a Diffie-Hellman
exchange. There is no particular encoding (e.g. a TLV) used for the data of a KE payload.

Nx is the nonce payload; x can be: i or r for the ISAKMP initiator and responder respectively.

IDx is the identification payload for "x".  x can be: "ii" or "ir" for the ISAKMP initiator and
responder respectively during phase one negotiation; or "ui" or "ur" for the user initiator and
responder respectively during phase two.

SIG is the signature payload. The data to sign is exchange-specific.

CERT is the certificate payload.

HASH (and any derivative such as HASH(2) or HASH_I) is the hash payload. The contents of the hash
are specific to the authentication method.

prf(key, msg) is the keyed pseudo-random function-- often a keyed hash function-- used to generate
a deterministic output that appears pseudo-random.  prf's are used both for key derivations and for
authentication (i.e. as a keyed MAC).

SKEYID is a string derived from secret material known only to the active players in the exchange.

SKEYID_e is the keying material used by the ISAKMP SA to protect the confidentiality of its messages.

SKEYID_a is the keying material used by the ISAKMP SA to authenticate its messages.

SKEYID_d is the keying material used to derive keys for non-ISAKMP security associations.

<x>y indicates that "x" is encrypted with the key "y".

--> signifies "initiator to responder" communication (requests).

<-- signifies "responder to initiator" communication (replies).

|  signifies concatenation of information-- e.g. X | Y is concatentation of X with Y.

[x] indicates that x is optional.
```

Table 2.5: Key to the names of variables used in the IKE exchanges, as per RFC 2409 [58].

Chapter 3

Fingerprinting: Test Methodology

In this chapter, the practical tests, together with their component elements, are described in detail. The practical differences between physical and virtual networks are discussed first, followed by the lab layout which was used to run the simulations. This is followed by a brief introduction to packet capture and protocol analysis. Following this introduction, the methodology employed is described, and the criteria for choosing fingerprint characteristics is discussed. The chapter closes with a brief summary of the key points presented.

3.1 Virtual vs. Physical Networks

Initially in this study, it was thought that it would be both cost-efficient and convenient to make use of VMware Workstation[1] instead of a physical network comprising of real machines. VMware Workstation is a program that allows `virtual` machines (referred to as *guest OSs*) to operate on top of, or within, physical machines (referred to as *host OSs*). VMware Workstation also allows these virtual machines to communicate with each other by means

[1] VMware Workstation 4.5 [124]

of a `virtual network`.

After a series of tests however, it became evident that although the VMware network approximated the behaviour of a physical network, there were some instances where the behaviour of the VMware network was dissimilar to that of a real (physical) network. For example, in the VMware network layer 2 broadcasts were forwarded across gateway machines by default, whereas in a physical network, this was either not possible (Windows 2000 Advanced Server) or had to be enabled manually (Linux, OpenBSD, Windows 2003 Enterprise Server). Since the purpose of this study was to observe real-world behaviour, it was decided that simulations be conducted in a physical network composed of real machines, and to only use VMware when the real network proved uncooperative.

It should be noted that this study focused on the IKE key exchange messages between two gateway machines (initiated on behalf of two endpoint machines, traversing n routers, where $n \geq 1$) and that the details of the underlying network are unimportant[2]. This led to an interesting finding, however: although the operating systems in question supported VPN tunnels in *gateway* mode[3] in some cases these OSs could not form part of a physical network topology that would accommodate VPN tunnels in gateway mode. Using the OSI 7-Layer networking model as a reference [120], this means some OSs could not behave as gateways (or *bridges*) at Layer 2, being unable to forward ARP requests on the same physical link that other OSs in the test had been able to. Since Layer 3 (IP) connectivity requires Layer 2 ARP resolution, lack of communication at Layer 2 leads to lack of communication at Layer 3. Gateway communication at Layers 2 and 3, in turn, is a requirement for cryptographic key exchange at Layers 5 and 4 (i.e.,

[2]Unimportant because so long as the key exchange succeeds between machines on the different networks, it can be assumed that the underlying Layer 3 connectivity must be in place.

[3]VPN tunnels established by gateways on behalf of endpoints, as opposed to tunnels established between endpoints.

IKE over UDP).

As a result of this, some simulations were conducted on a physical network, whereas others were conducted on a VMware network, where this problem was not evident. The machines that comprised the physical network were all Intel Pentium IIIs, running at 733MHz, with 128MB of RAM. The virtual machines in the VMWare network were Intel Pentium 4s, running at 1.8GHz, also with 128MB of RAM. It is important to note that the roles of `gateway`, `router`, and `endpoint` were all played by ordinary PCs (and in the case of VMware, *virtual* PCs).

3.2 Laboratory Layout

The aim of laboratory design was to represent the minimum topology that would approximate a typical branch office VPN scenario, where two disparate networks are connected by setting up a tunnel through the Internet. In a simple real-world case, this can be accomplished by having two endpoints in two separate LANs[4]. The Internet is depicted as a network cloud, which can contain n routers, where $n \geq 1$. The gateway is concerned with connecting each LAN to the Internet, and so it has one network interface card (NIC) on each network. When depicting this level of network detail, a NIC connecting to the internal network is referred to as an *internal* interface, and similarly a NIC connecting to an external network (such as the Internet) is referred to as an *external* interface. In Figure 3.1 for example, Gateway 1´s *internal* interface is in the 172.16.0.0/24 network and its *external* interface is in the 10.0.0.0/24 network.

As can be seen in Figure 3.1, four networks were used: the 172.16.0.0, 172.16.1.0, 10.0.0.0, and 10.0.1.0 networks. Although two of the networks were Class A networks (10.0.0.0 and 10.0.1.0), and two were Class B networks

[4]With one endpoint initiating the connection, and one responding to, or terminating the connection; in this study, they were located in the 172.16.0.0/24 and the 172.16.1.0/24 networks

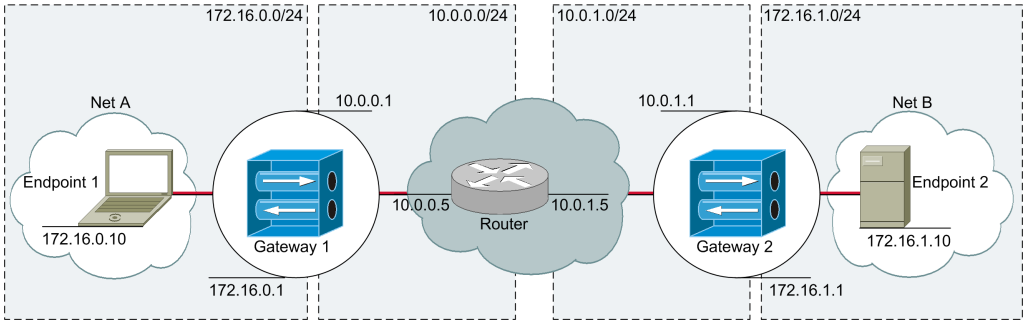


Figure 3.1: Laboratory Layout

(172.16.0.0 and 172.16.1.0), they were all employed as Class C networks, as can be seen by the `/24` prepended to the network number, denoting the Class C netmask (255.255.255.0)[5]. The symbols used to represent the gateways depict VPN-enabled devices, with two tunnels, one in each direction, corresponding to the unidirectional IPSec SAs discussed in Chapter 2. The router is shown within an IP cloud, since this is a model that holds in a real-world scenario; with the only change being the number of routers within the IP cloud (a factor that is not relevant to these tests). The internal networks (Net A and Net B) are depicted in lighter-coloured clouds, denoting multiple networked devices, from which only the two endpoints are relevant to these tests.

Given the small number of machines, it was decided to use static routes instead of running a routing protocol. In this setup, each of the endpoints had its adjacent gateway as the *default gateway*, the gateways had the router as their *default gateway*, and the router had static routes into every network. In this manner, the prerequisite of having full layer-3 connectivity was accomplished prior to any attempt to secure the communication. The reason for requiring unsecured Layer 3 connectivity before IPSec can be configured on a link, is that even though IPSec operates at Layer 3, the key exchange handled by IKE, which is fundamental to the tunnel establishment takes place above

[5]Detailed information on IP addressing and subnetting is available in [28]

UDP, that is, above Layer 4 –which in turn requires a functioning Layer 3 connection.

3.3 Packet Capture & Analysis

For the purpose of this study, the manner in which the key exchange could be recorded and analysed also had to approximate reality, in that it should depict a plausible scenario where a person or organisation of ill-intent would be able to eavesdrop on the tunnel setup and teardown. Ideally, in order to make this study most relevant, it should be possible to model a real-world scenario where the attacker has full access to the network traffic originating from Net A, and destined to Net B. There are two locations in Figure 3.1 where this could be possible. The first is between an endpoint and a gateway, which would assume that the attacker has penetrated either corporate network (Net A or Net B), and the second is between either gateway and the router, which would require a path altering attack (routing protocol poisoning attack). Although breaking into (and out of) a corporate network can be viewed as a trivial task by some, simulating it in a laboratory environment for the purpose of this research would constitute a digression from the original theme. For this reason, the second scenario was chosen, because as an attack, it does not require breaking into an corporate network, but rather simply tapping into the network and passively listening to the traffic. The following section describes how the scenario of scrutinizing traffic on the Internet by inserting a rogue router, and forcing traffic to be redirected through it, is a realistic one.

3.3.1 Altering Traffic Path

When an enterprise connects two geographically disparate networks by means of a tunnel, it cannot predict the end-to-end path that the tunneled traffic

will take[6], and as such, exposes itself to potential attackers on the public internet. Given the high volume of traffic that traverses routers on the public internet, it is unlikely that attackers would choose this avenue of attack. *Unlikely*, however, does not mean *impossible*. Routers on the public internet communicate by means of routing protocols[7], since the alternative, manually managing static routes to each destination on the Internet, is infeasible. It is these routing protocols that are vulnerable to deception by dedicated attackers.

Among routing protocols that are in widespread use, and susceptible to path alteration attacks are the Interior Gateway Routing Protocol (IGRP), the Enhanced Interior Gateway Routing Protocol (EIGRP), the Routing Internet Protocol version 1 (RIPv1) and version 2 (RIPv2), and the Open Shortest Path First protocol (OSPF). Protocols used in the same environments, not for routing, yet also susceptible to attack are the Cisco Discovery Protocol (CDP), the ICMP Router Discovery Protocol (IRDP), the Hot Standby Router Protocol (HSRP), the Border Gateway Protocol (BGP) and ICMP redirects.

What makes these attacks possible, and thus renders routers vulnerable to being hoaxed into rerouting traffic, is that routers by definition, rely on the information provided by their routing protocols and by the routing protocols running on their peer systems, to update their routing tables. Sometimes these routing protocols provide a means of authenticating routing updates[8]. However in practice these authentication mechanisms are often either not

[6]This is a property of dynamic routing paths, a feature of the Internet, and a product of ARPANET research.

[7]Routing protocols can be classified into *internal* routing protocols, which are typically employed within an organisation´s network, and *external* routing protocols, typically used within Service Provider clouds. The routing protocols discussed here are both internal and external.

[8]EIGRP and OSPF, for example, allow routing updates to be accompanied by an MD5 hash.

configured[9], misconfigured, or not used at all[10].

In essence, these attacks are possible because an attacker can insert a rogue router into the public internet, and `convince` its peers that it is the legitimate *next hop*, once this is accomplished, the attacker has access to all traffic traversing the rogue router. What makes this particularly valuable, other than being able to capture all encrypted traffic in the hope of brute-forcing the key and decrypting the contents at some future point in time, is that depending on the tunneling protocol, it may be possible to force an error in the connection, and thus cause the protocol to renegotiate the tunnel (not just rekey), and in this manner, to expose the key exchange to the attacker. As described by Linder in [80], this category of attacks can be mounted on Layer 2 and Layer 3. On Layer 2 it can be done by giving false link-layer address information to both parties (accomplished by ARP interception), and on Layer 3 by giving false *next hop* information to either one or both parties. In an IRDP attack, the attacker sends IRDP updates and at the same time makes the default gateway temporarily unavailable (using DoS attacks, or ARP interception, for example), and thus becomes the default router. In EIGRP, if authentication is not used, the attacker can spoof its source network, and join as an EIGRP `neighbour`; after this, it is able to inject new routes into the network. In BGP it is possible for the attacker to inject bad updates, make use of TCP sequence number attacks, and layer 2 man-in-the-middle attacks, see Figure 3.2. More detailed explanations on mounting these attacks are available on [80] and [79].

3.3.2 Capturing Traffic

The method whereby idle traffic[11] can be intercepted is referred to as *packet sniffing* [109]. On Ethernet networks, packet sniffing is possible because at

[9] by `not configured` is meant that the *default* configuration is left untouched.

[10] OSPF for example, authenticates all routing updates, however one legitimate method of authentication is ``none``.

[11] by `idle` is meant traffic not specifically destined to the listening host

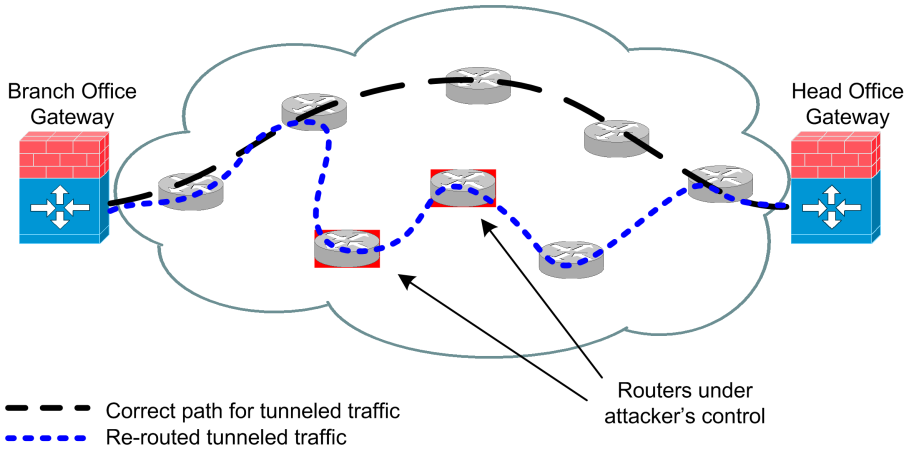


Figure 3.2: Re-routing traffic by means of path altering attacks. Adapted from [32]

Layer 1, the physical specification of Ethernet (as well that of as IEEE 802.3) transmits the electrical signal on the wire in what equates to a broadcast [109]. Hosts communicating on Ethernet networks are able to listen to all traffic traversing the link, but only look at traffic containing their Ethernet address. In order to intercept network traffic not destined for the monitoring station (`sniffing`), the host OS places the NIC corresponding to the network to be sniffed, in what is termed *promiscuous* mode, thus instructing the network card driver to pass all traffic to the sniffing application [109].

For the purpose of capturing the traffic containing the key exchange, the opensource tool tcpdump was used [67]. If run without specifying options, *tcpdump* listens on the first ethernet adapter, and displays only the header information of all sniffed traffic. In order to study a Layer 3 or Layer 4 protocol key exchange, it is advisable to capture the entire Ethernet frame. tcpdump can do this, when given the -vvv flag for extra verbosity, and the -s option to denote the snap length: in this case, the value of the snap length should be no less than the link MTU. Once the packet capture had been completed, the Ethereal protocol analyser [23] was used to provide a detailed

representation of the packet contents, and especially the ISAKMP payload data.

3.4 Test Procedure

In order to configure the IPSec implementations on each gateway machine, official documentation on the topic was obtained from the corresponding vendors (Microsoft, Solaris, OpenBSD), and from the official developers´ sites (ISAKMPd and Racoon), and the instructions were carefully followed. The ICSA Labs recommendations delineated in [76] were followed as cryptographic options to Phase 1 and Phase 2 were concerned. These recommendations are listed in Table 3.1.

Phase 1	Main Mode
Phase 2	Quick Mode
Device Authentication	Preshared Keys: (Phase 1 identities = IP Address) Digital Certificates: (Phase 2 identities = IP Address, FQDN, DN, or User FQDN)
Encryption	3DES (NULL = w/o confidentiality)
Authentication/Integrity	SHA-1
Diffie-Hellman Group	Group 2
ESP or AH	ESP
Mode	Tunnel
PFS On (DH Group 2)	PFS Off
Traffic Selectors	ANY
Phase 2 Identities	IP Address, Subnet, or Ranges
Lifetimes	Relatively short, based on time, not traffic volume

Table 3.1: ICSA Labs guidelines on IPSec VPN tunnel creation for testing purposes [76].

Two tests were conducted: The first consisted in sending an ICMP `ECHO REQUEST` (*ping*) to endpoint 2 from endpoint 1, causing gateway 1 to establish an encrypted tunnel through the router to gateway 2, and allow the `ECHO`

REQUESTs to traverse the tunnel to endpoint 2, and return the ECHO REPLYs from endpoint 2; as soon as 5 echo replies had been received, the *ping* would be stopped, and the tunnel torn down manually[12]. The second test consisted in using the *telnet* application to connect from endpoint 1, to the day-time port (port 13) on endpoint 2; this would cause the gateways to set up the tunnel, with endpoint 2 returning the current date and time, and ending the telnet connection, the tunnel would then be torn down on each gateway.

Searching for fingerprints was broken down into three steps:

- analysis of the connection summary,
- analysis of the IKE exchange, and
- analysis of the ESP-protected IP datagrams.

The connection summary provided an overview of the packet exchange between the two gateway machines (source and destination addresses, protocol type and protocol information). Analysis of the IKE exchange consisted in scrutinizing the ISAKMP payload of each IP datagram involved in the IKE Phase 1 and Phase 2 exchange, and looking for anomalies, differences between the various implementations, and contrasting this information to the IPSec DOI, ISAKMP, and IKE RFCs (2407, 2408, and 2409 respectively). The last step, analysing the encrypted (ESP-protected) IP datagrams examined the datagrams which contained the ICMP ECHO REQUEST and REPLY messages, and the TCP handshake and disconnect messages, in order to determine what could be inferred from their encrypted payload.

In the following chapter, the findings resulting from the fingerprinting tests will be presented and discussed.

[12] in UNIX and Linux machines this was done by sending the IKE daemon a TERM signal, and in Windows machines this was done by deactivating the IPSec policy.

Chapter 4

Fingerprinting: Test Results

This chapter details the findings of the fingerprinting and protocol analysis process. Each of the unique discriminants, which forms part of the fingerprint for each IKE/IPSec implementations is described first, including references in the RFCs comprising the IPSec standard. Each of these RFC-defined discriminants is then contrasted with its behaviour as exhibited by each implementation. The Appendices provided at the end of this document provide guidelines on how to reconstruct the experiments that produced these results. Appendix A delineates steps on how to configure an IPSec VPN tunnel on the Linux 2.6 kernel, first using *racoon* and then *isakmpd*. Appendix B explains how the network traces were analysed, using the two Linux platforms as an example, in order to identify two distinct behavioural patterns. The present chapter closes with a summary and a diagrammatic representation of the fingerprints.

4.1 IPSec Tunnels, IKE daemons

The process of fingerprinting an encrypted tunnel endpoint is a tricky one. Following the initial messages of the key exchange, all the communication is encrypted. It is for this reason that the process could be better defined

as fingerprinting the key exchange process that takes place between the two endpoints. This key exchange, ISAKMP/IKE in IPSec, is handled on the gateway system by a daemon process that listens for connections on a particular port (UDP 500 for IKE). In the case of each of the implementations tested, fingerprinting the IKE daemon was analogous to fingerprinting the IPsec implementation (or the OS, for that matter), since each OS has its own IKE daemon. Linux, is the notable exception. There are currently two IKE daemons available for the Linux 2.6 platform: *isakmpd*[1] and *racoon*. isakmpd is the OpenBSD ISAKMP/IKE daemon, which has been ported to Linux, and racoon is the ISAKMP/IKE daemon developed by the KAME project [102]. Both make use of the IPSec implementation on the 2.5.x and 2.6.x Linux kernels (that is, they will not work on 2.2.x and 2.4.x kernels). The Linux tests were conducted using the 2.6.8.1 kernel.

At the outset of the testing phase, it was decided that IKE implementations of five OSs would be observed, and that the IPSec tunnels would be configured to use pre-shared keys (PSKs) and x.509 certificates (certs) as per the ICSA Labs recommendations, bringing the total number of tests to twelve[2]. Tutorials on how to configure an IPSec tunnel recommend using PSKs initially, because it is the most straight-forward method. The use of PSKs however, is not recommended in large enterprise deployments, because manually distributing and managing statically generated keys is not a solution that scales well in a real-world deployment, and this is why x.509 certificates are most widely used. The tests were meant to reflect both of these scenarios. Results showed, however, that the ISAKMP/IKE exchanges differed little whether PSKs were used, or certs. The one clear difference was the *Authentication Method* type: it was `PSK(1)` when pre-shared keys

[1]Daemons in UNIX and UNIX-like OSs (analogous to *services* in Windows OSs) generally have a *d* appended to the name of the process they represent; for example isakmp*d* is the ISAKMP *daemon*.

[2]The total number of tests was projected to be twelve, not ten, because the Linux tests involved two different IKE implementations, therefore Linux counted as 4 (2PSK + 2cert).

were used, `RSA-SIG(3)` when certs were used in all OSs other than Solaris, and `RSA-ENC(4)` when certs were used in Solaris 9. Because the results of using PSKs were near identical to when certs were used, the findings discussed below are only those of ISAKMP/IKE tunnels configured using x.509 certificates. Following the test phase described in Chapter 3, twenty discriminants were identified, a combination of which can be used to single out each IPSec implementation. Of these twenty, four can be considered *potential* discriminants, so named because although their behaviour was similar among the platforms tested, it can be conjectured that they would differ, provided that the IKE/IPSec-platform sample size was bigger (i.e., to include VPN devices). The *potential* discriminants can be identified in Table 4.6 at the end this chapter, by the fact that they hold true under all platforms. All the discriminants are detailed below, classified into the three steps that were taken to find them, as mentioned at the close of Chapter 3:

- Discriminants resulting from analysing the connection summary,
- Discriminants resulting from analysis of the IKE exchange, and
- Discriminants resulting from analysis of the ESP-protected IP datagrams.

It should be noted that for brevity, the initials MM and QM have been used below and in Table 4.6 as per the ICSA Labs recommendations [76], to denote IKE Main Mode (Phase 1) and IKE Quick Mode (Phase 2) respectively. The message number in each exchange, where the discriminant was observed, is appended to the initials; that is, MM4 refers to the fourth message in the Main Mode exchange, QM2 refers to the second message in the Quick Mode exchange, and so on.

4.1.1 Discriminants from Connection Summary

In order to study the connection summary, the entire network trace[3] was loaded into the Ethereal traffic analysis tool, and exported using the `Packet summary line` option. The output is equivalent to a that of a tcpdump with the least verbosity, recording no detail about the connection other than time-stamps, source and destination addresses, and protocol type. Examples of the level of detail visible when working with packet summaries, can be seen in Tables 4.3, 4.4, and Figures B1 and B2 of Appendix B.

DF Bit Set in IP Header

As described by the OSI 7-layer model, before being transmitted from one host to another, a layer-n protocol must be encapsulated within a layer n-1 protocol. Depending on the Layer-2 protocol in use, there is a limit on the maximum allowable frame size. This property is known as the Maximum Transmission Unit (MTU). MTU sizes differ depending on the transmission media: Ethernet´s MTU is 1500 bytes, IEEE 802.3´s MTU is 1492 bytes, and X.25´s is 576 bytes, to name but a few. If the upper layer protocol (in this case, IP) needs to send datagrams larger than the link MTU, it splits the datagram into smaller pieces, each of which is smaller than the link MTU. The fragments are reassembled when they reach the destination, a process known as *fragmentation and reassembly* [120].

There are issues surrounding IP fragmentation, which make it undesirable or even unsuitable in some cases. Firewalls, for example, may have trouble allowing IP fragments which have arrived out of order, since the non-initial IP fragments may not match filter rules, and allowing non-initial fragments may open the door for fragmentation attacks. Other examples of the undesirable

[3]Throughout the text, the terms *network trace* and *packet capture* are used to denote the same thing, namely the result of running the `tcpdump` command, and observing Layer 2 (and up) communication between the two end hosts, and the gateways establishing the IPSec tunnels.

properties of IP fragmentation include the computing overhead required to fragment and reassemble an IP datagram, the memory required to buffer the fragments until they can be reassembled, and the fact that if any of the IP fragments is lost in transit, the entire IP datagram needs to be retransmitted [22].

Whenever IP fragmentation is not desired, the Don´t Fragment (DF) bit on the IP header can be set. When the DF bit is set, if the IP datagram wishes to traverse a link whose MTU is smaller than the datagram´s size, the intervening router will discard the datagram, and send an ICMP `fragmentation needed but don´t fragment bit set` error back to the originator [120].

ISAKMP Informational Sent Between Main Mode and Quick Mode

RFC 2408 defines the ISAKMP Informational exchange as ``a one-way transmittal of information that can be used for security association management`` [83]. RFC 2408 and RFC 2409 further state, that once the ISAKMP SA has been established (that is, once Phase 1, has completed), ``either party may initiate Quick Mode, Informational, and New Group Mode exchanges`` [83, 58].

With regards to ISAKMP Informational messages, RFC 2408 states ``All ISAKMP implementations MUST[4] implement the Informational Exchange and SHOULD implement the other four exchanges. However, this is dependent on the definition of the DOI and associated key exchange protocols.`` [83].

When interpreting RFC´s wording, with regards to recommendations, RFC 2119 should be consulted for the intended meanings. *MUST*, *REQUIRED*, or *SHALL* for example, ``mean that the definition is an absolute requirement of the specification.`` *SHOULD* or *RECOMMENDED* ``mean that there may exist valid reasons in particular circumstances to ignore a

[4]Here, as elsewhere in the text, MUST, MUST NOT, SHOULD, and SHOULD NOT have been quoted exactly how the appear in the RFCs, maintaining the capitalization.

particular item, but the full implications must be understood and carefully weighed before choosing a different course.`` By `MUST implement the Informational Exchange` it is therefore understood that generating and processing ISAKMP Informational messages has to be supported by all implementations.

Sending an ISAKMP Informational message after Main Mode and before Quick Mode is therefore allowed, and not in violation of the standard.

ISAKMP Informational Sent at Tunnel Tear-Down

With regards to the transmission of ISAKMP informational messages, RFC 2409 states ``The base ISAKMP specification describes conditions in which one party of the protocol may inform the other party of some activity– either deletion of a security association or in response to some error in the protocol such as a signature verification failed or a payload failed to decrypt. It is strongly suggested that these Informational exchanges not be responded to under any circumstances. Such a condition may result in a ``notify war`` in which failure to understand a message results in a notify to the peer who cannot understand it and sends his own notify back which is also not understood.``

RFC 2408 refers to the sending ISAKMP Informational messages with a Delete Payload as a valid method of notifying the remote party that the sending party has deleted its SAs [83].

Sending an ISAKMP Informational message when the IPSec tunnel is being torn down, is therefore allowed, and not in violation of the standard. Sending one or more messages in response to the ISAKMP Informational messages is, however, discouraged by the standard.

ISAKMP Informational Sent from Initiator to Responder

In several places within RFCs 2407, 2408, and 2409 reference is made to the fact that ISAKMP Informational messages can be sent by either the Initiator

or the Responder. Two such references are ``once established, *either party* may initiate Quick Mode, Informational, and New Group Mode Exchanges`` [58] (Author´s emphasis), and ``When creating a Notification Payload, the transmitting entity *(initiator or responder)* MUST do the following ...`` [83] (Author´s emphasis).

ISAKMP Informational messages originating from either the Initiator or from the Responder are therefore allowed, and not in violation of the standard.

Six Messages in Main Mode as per RFC 2409

Regarding the number of messages in IKE exchanges, RFC 2409 unequivocally states ``Exchanges in IKE are not open ended and have *a fixed number* of messages. Receipt of a Certificate Request payload MUST NOT extend the number of messages transmitted or expected`` [58] (Author´s emphasis). All diagrams illustrating Phase 1 (Main Mode) in RFC 2409 depict it as consisting of six messages, in a request-response pattern, originating from the Initiator.

Therefore, an IKE/IPSec implementation employing anything other than six messages for IKE Phase 1 Main Mode, would be in violation of the standard.

Three Messages in Quick Mode as per RFC 2409

All diagrams describing Phase 2 (Quick Mode) in RFC 2409 depict it as consisting of three messages, in a request-response pattern, originating from the Initiator; and as per the above reference, IKE exchanges consist of a fixed number of messages.

Therefore, an IKE/IPSec implementation employing anything other than three messages for IKE Phase 2 Quick Mode, would be in violation of the standard.

Encrypted Transmission Prior to Quick Mode Completion

Referring to the use of the Commit Bit in either Main Mode or Quick Mode exchanges, RFC 2408 says that it ``is used to ensure that encrypted material is not received prior to completion of the SA establishment`` [83], suggesting that it is undesirable that encrypted communication should take place before the (successful) completion of the phase negotiation. The RFC´s stance is unclear however, as its use has not been explicitly forbidden, and no further reference to it was found in the RFCs.

It is therefore assumed that transmitting encrypted material during a phase negotiation, which is not implicitly part of the exchange, is not in violation of the standard, but could constitute bad practice.

MM5 and MM6 Fragmented

There was no reference to fragmentation in RFC 2407, 2408, or 2409, suggesting that the IPSec standard is mute on the subject. A Cisco Systems White Paper however, refers to fragmentation of IPSec traffic as undesirable: ``You really want to avoid fragmentation after encapsulation when you do hardware encryption with IPsec. Hardware encryption can give you throughput of about 50 Mbs depending on the hardware, but if the IPsec packet is fragmented you loose 50 to 90 percent of the throughput. This loss is because the fragmented IPsec packets are process-switched for reassembly and then handed to the Hardware encryption engine for decryption. This loss of throughput can bring hardware encryption throughput down to the performance level of software encryption (2-10 Mbs)`` [22].

As mentioned earlier, another undesirable trait of fragmentation is the need to retransmit the entire (fragmented) datagram, in the event that a single fragment is lost. Attempting IPSec communication (allowing fragmentation) over a noisy link could cause the communication to become impractically slow due to retransmission.

Since the RFCs make no mention of fragmentation, it can be assumed that

an implementation allowing fragmentation is not in violation of the standard, but in the view of secondary literature it can be seen as undesirable, and could be considered bad practice.

4.1.2 Discriminants from IKE Exchange

The task of analysing the network traces with the aim of studying the IKE exchange required access to all the fields of the IP payload. For this study, the network trace was loaded into Ethereal, and exported using the `Packet Details:All Expanded` option. This displays all data expanded and decoded, starting from the Layer 2 frame header.

Proposal Payload SPI Size = 0

With regards to the Security Parameter Index (SPI) Size value, RFC 2408 states ``the SPI Size is irrelevant and MAY be from zero (0) to sixteen (16). If the SPI Size is non-zero, the content of the SPI field MUST be ignored. If the SPI Size is not a multiple of 4 octets it will have some impact on the SPI field and the alignment of all payloads in the message. The Domain of Interpretation (DOI) will dictate the SPI Size for other protocols``[83]. The DOI document referred to is RFC 2407, which contains two references to SPI Size, both for IPSec Notify Message types, but none for the Proposal Payload.

It can thus be assumed that any value for SPI Size is valid, but given the explanation of MUST in RFC 2119 (that MUST denotes an absolute requirement of the specification), an implementation would be in violation of the standard if it does not ignore the contents of the SPI field in the event of the SPI Size value being anything other than zero.

Commit Bit Set in Quick Mode

RFC 2408 provides a detailed explanation of the Commit Bit, its purpose and use (condensed here for brevity): `` This bit is used to signal key exchange synchronization. It is used to ensure that encrypted material is not received prior to completion of the SA establishment. The Commit Bit can be set (at anytime) by either party participating in the SA establishment, and can be used during both phases of an ISAKMP SA establishment. ... If set(1), the entity which did not set the Commit Bit MUST wait for an Informational Exchange containing a Notify payload (with the `CONNECTED` Notify Message) from the entity which set the Commit Bit ... The receipt and processing of the Informational Exchange indicates that the SA establishment was successful and either entity can now proceed with encrypted traffic communication`` [83].

Use of the Commit Bit is therefore optional. However in the event that it be implemented, it is an absolute requirement that the party which did not set the Commit Bit has to wait for an ISAKMP Informational message containing the `CONNECTED` Notify Message, and can therefore be considered in violation of the standard if it does not do so.

Use of Vendor-ID Payload

The Vendor-ID payload has been defined in RFC 2408 as containing ``a vendor defined constant. The constant is used by vendors to identify and recognize remote instances of their implementations. This mechanism allows a vendor to experiment with new features while maintaining backwards compatibility. This is not a general extension facility of ISAKMP ... Multiple Vendor ID payloads MAY be sent. An implementation is NOT REQUIRED to understand any Vendor ID payloads. An implementation is NOT REQUIRED to send any Vendor ID payload at all. ... Reception of a familiar Vendor ID payload in the Phase 1 negotiation allows an implementation to make use of Private USE payload numbers (128-255)`` [83]. It is important

to note that if all IKE/IPSec implementations made use of the Vendor-ID payload, this could by itself, constitute a fingerprint, because a string that uniquely identifies the vendor would be used. As will be shown later, only two of the implementations tested, made use of the Vendor-ID string.

It is therefore clear from this definition that the use of Vendor-ID payloads is optional, and that receipt of familiar payloads entitles the implementation to make use of `Private USE` payloads.

Transform Payload Numbering Begins at 1

As per RFC 2408, The Transform payload ``contains information used during Security Association negotiation. The Transform payload consists of a specific security mechanism, or transforms, to be used to secure the communications channel. The Transform payload also contains the security association attributes associated with the specific transform.`` and further ``Transform # (1 octet) - Identifies the Transform number for the current payload. If there is more than one transform proposed for a specific protocol within the Proposal payload, then each Transform payload has a unique Transform number.`` Regarding the numbering of Transform payloads, RFC 2408 reads `` There may be several transforms associated with a specific Proposal payload each identified in a separate Transform payload. The multiple transforms MUST be presented with monotonically increasing numbers in the initiator´s preference order.``

An implementation can thus be considered to be in compliance with the standard, if the Transform payload numbers increase monotonically, however, since no mention is made of what the initial number should be, it can be assumed that *any* starting number is valid, be it 0, 1, or anything else.

Unique Order of SA Attribute Type Values in MM1 Transform Payload

Attribute Values are classified as either Basic (B) or Variable Length (V), and the only restriction imposed by RFC 2407 is that ``Attributes described as basic MUST NOT be encoded as variable. Variable length attributes MAY be encoded as basic attributes if their value can fit into two octets.`` Furthermore, in RFC 2408 the SA Attributes field is depicted as variable in length, suggesting that any number of attributes can be included. RFC 2407 adds the only limitation, stating that ``An SA Life Duration attribute MUST always follow an SA Life Type which describes the units of duration.`` Sample Attribute Classes, Values and Types can be seen in Table 4.1.

Class	Value	Type
Encryption Algorithm	1	B
Hash Algorithm	2	B
Authentication Method	3	B
Group Description	4	B
Group Type	5	B
Group Prime/Irreducible Polynomial	6	V
Group Generator One	7	V
Group Generator Two	8	V
Group Curve A	9	V
Group Curve B	10	V
Life Type	11	B
Life Duration	12	V
PRF	13	B
Key Length	14	B
Field Size	15	B
Group Order	16	V

Table 4.1: SA Attribute Classes, Values and Types, as per RFC 2409 [58].

The task of ordering the SA Attribute Type values can then be understood to have been left to the implementers, the only absolute condition being that

the Life Type attribute must always be followed by a Life Duration attribute.

Unique Order of ISAKMP Payloads in MM4

RFC 2408 states ``An ISAKMP message has a fixed header format . . . followed by a variable number of payloads.`` RFC 2409 further states ``The SA payload MUST precede all other payloads in a phase 1 exchange. Except where otherwise noted, there are no requirements for ISAKMP payloads in any message to be in any particular order.`` RFC 2408 later adds ``While the ordering of payloads within messages is not mandated, for processing efficiency it is RECOMMENDED that the Security Association payload be the first payload within an exchange.``

It can thus be assumed that the ordering of ISAKMP Payloads is left up to the vendor, but that there ``may exist valid reasons in particular circumstances`` to position the SA payload as the first one, and as RFC 2119 states, ``the full implications must be understood and carefully weighed before choosing a different course``. Table 4.2 lists the payload types found in the ISAKMP header, together with their associated values, as defined in RFC 2408.

First Payload Type in MM3 and MM4 is Key Exchange(4)

The diagrammatic description of IKE Phase 1 authenticated using signatures (x.509 certificates) in RFC 2409 shows that in the second message from the Initiator, the second item is the Key Exchange, suggesting that this should be the first payload in MM3. (Indeed, this is how most vendors interpreted it.) There is an interesting observation however: RFC 2409 states that when Phase 1 authenticated with a revised mode of public key encryption is used, ``If the HASH payload is sent it MUST be the first payload of the second message exchange and MUST be followed by the encrypted nonce. If the HASH payload is not sent, the first payload of the second message exchange MUST be the encrypted nonce. In addition, the initiator may optionally send

Next Payload Type	Value
NONE	0
Security Association (SA)	1
Proposal (P)	2
Transform (T)	3
Key Exchange (KE)	4
Identification (ID)	5
Certificate (CERT)	6
Certificate Request (CR)	7
Hash (HASH)	8
Signature (SIG)	9
Nonce (NONCE)	10
Notification (N)	11
Delete (D)	12
Vendor ID (VID)	13
RESERVED	14 - 127
Private USE	128 - 255

Table 4.2: ISAKMP Header Payload Values and Types, as per RFC 2408 [83].

a certificate payload to provide the responder with a public key with which to respond.`` A *nonce* in key-management protocols, it should be noted, refers to a pseudo-random number ``issued in an authentication protocol to ensure that old communications cannot be reused in `repeat attacks` [126].``

Since the practical tests discussed in this chapter were conducted using x.509 certificates, this definition would be relevant if it appeared under the ``IKE Phase 1 Authenticated With Signatures`` of RFC 2409. However it appears under the description of an authentication method which allows *optional* x.509 certificates. Thus as in several other instances, the RFC creates an ambiguity. On the one hand it may be argued by a vendor that RFC compliance depends on `Key Exchange` being the first payload in MM3; on the other hand, a vendor can also argue that a certificate is an optional case in Phase 1 as a revised mode of public key encryption. As will be shown

later, all but one vendor interpreted the first meaning to be correct.

This reasoning applies to `Key Exchange` as the first payload of the second message from the Responder (MM4) also. If Section 5.1 of RFC 2409 was followed, then `Key Exchange` would be the first payload, otherwise `Nonce` would be the first payload. Indeed, both these possibilities could be argued to be RFC-compliant.

First Payload Type in MM5 and MM6 is Identification(5)

The argument presented above also applies to `Identification` as the first payload of the third message from the Initiator as well as that of the Responder (MM5 and MM6). If Section 5.1 of RFC 2409 was followed, then `Identification` would be the first payload, but if Section 5.3 was followed, `Hash` would be the first payload.

Certificate Request Payload Type (7) Visible in MM4

The same is true for the `Certificate Request` payload. If the authentication is viewed as consisting of x.509 certificates, then a `Certificate Request` should be visible in MM4, but if the authentication is viewed as a special case of a revised mode of public key encryption, then MM4 would contain <Nr_b>PubKey_i, <KE_b>Ke_r, <IDir_b>Ke_r, where Nr is the Responder´s nonce, PubKey_i is the Initiator´s public key, KE is the Key Exchange payload, Ke_r is the key to the symmetric encryption algorithm negotiated earlier, IDir is the Initiator and the Responder´s identification, and the <x>y notation means x is encrypted using y.

4.1.3 Discriminants from ESP Traffic

ESP-protected traffic is encrypted, and it would seem logical that its perusal should reveal nothing. This proved not to be the case, however. Analysing the ESP-protected traffic resulted in successfully identifying patterns within

the encrypted stream, which corresponded to the plain text traffic. This could constitute, to a small measure, defeating the purpose of encrypting the traffic in the first place. In order to be able to discern anything from the encrypted packet exchanges, all the information available was viewed and analysed for recurring patterns. These results contributed to fingerprinting in that they partially laid bare the behaviour of each implementation`s TCP/IP stack, irrespective of encryption. These are partial discriminants however. Observing the behaviour of other IKE/IPSec implementations on other platforms will determine whether these *partial* discriminants can be promoted to *complete* discriminants. These tests nevertheless showed that wrapping the contents of an IP datagram in cryptography does not necessarily hide its contents.

Encrypted Ping (ICMP 8:0, 0:0) Distinguishable

As mentioned in section 3.4, a *ping* is the name given to the process (or program) that sends an ICMP `ECHO REQUEST` (type 8, code 0) to a destination address, and receives an ICMP `ECHO REPLY` (type 0, code 0) in return. A characteristic of a *ping* is that provided an `ECHO REPLY` is received, and provided that no attempt is made to mask the communication, the datagram size is fixed[5]. Another characteristic is that if the *ping* command is issued without additional options, it will, depending on the Operating System, send ICMP packets at a certain frequency. These two characteristics are also evident when observing encrypted traffic. The tests conducted showed that ESP-enabled (encrypted) ICMP communication appeared at regular intervals on the network, and that its datagram size was consistently unchanged (108 bytes). Tables 4.3 and 4.4 show the summary of unencrypted and encrypted ICMP traffic respectively. These results raise two important points. Firstly identifying encrypted traffic as being an ICMP `ECHO REQUEST` or `ECHO REPLY` opens the door to the possibility that the contents of other types of encrypted traffic may also be identified. Secondly, when working through large network

[5]However, ICMP packets of arbitrary size can be created manually with little effort.

traces looking for anomalous traffic patterns, the ability to isolate any single portion of the traffic, aids the investigator by narrowing down the search space.

Time	Source	Destination	Protocol Type	Protocol Info.
5.358251	172.16.0.10	172.16.1.10	ICMP	Echo (ping) request
5.371944	172.16.1.10	172.16.0.10	ICMP	Echo (ping) reply
6.975349	172.16.0.10	172.16.1.10	ICMP	Echo (ping) request
6.977195	172.16.1.10	172.16.0.10	ICMP	Echo (ping) reply
8.606745	172.16.0.10	172.16.1.10	ICMP	Echo (ping) request
8.606753	172.16.1.10	172.16.0.10	ICMP	Echo (ping) reply

Table 4.3: Summary view of unencrypted *ping*.

Time	Source	Destination	Protocol Type	Protocol Info.
35.772926	10.0.0.1	10.0.1.1	ESP	ESP (SPI=0x5709f15c)
35.782612	10.0.1.1	10.0.0.1	ESP	ESP (SPI=0x09c50632)
38.572154	10.0.0.1	10.0.1.1	ESP	ESP (SPI=0x5709f15c)
38.601115	10.0.1.1	10.0.0.1	ESP	ESP (SPI=0x09c50632)
41.082521	10.0.0.1	10.0.1.1	ESP	ESP (SPI=0x5709f15c)
41.102320	10.0.1.1	10.0.0.1	ESP	ESP (SPI=0x09c50632)

Table 4.4: Summary view of encrypted *ping*.

Encrypted TCP Handshake Discernible

As mentioned previously, the two connectivity tests conducted for the IPSec tunnel to be established (or *while it was established*, as in the case of Solaris 9 x86, or isakmpd on Linux, which would establish a tunnel regardless of traffic on the link) were a *ping* and a *telnet* from EndPoint 1 to TCP port 13 (the daytime server) on EndPoint 2 (referring to Figure 3.1). When connecting to the daytime server, the default behaviour is for the server to respond by sending the current date and time, and immediately close the connection. Since the connection was to *TCP* port 13, the TCP Handshake and TCP

Close sequences were present in the encrypted traffic. RFC 793 (Transmission Control Protocol) is the official TCP specification, and it includes a state diagram, adapted in Figure 4.1, which clearly explains the handshake and close. For a typical TCP connection, the sequence can be described as follows: the server starts off in the `CLOSED` state (the state at which it is not listening on the desired TCP port). Once the server enters the `LISTEN` state (awaiting connections from clients), it can receive a `SYN` from a client wishing to connect to it. Once this happens, it sends a `SYN+ACK` in response, and moves into the `SYN_RCVD` state, if it then receives an `ACK` from the client, it moves into the `ESTABLISHED` state, it is in this state that data transfer takes place (the transfer of the Time and Date in our example).

Either by observing whether the DF bit is set or not, or by hypothesizing that these messages are small enough not to need fragmentation, it can be inferred that an encrypted TCP handshake would consist of three distinct packets, of near-close or identical size. It can also be concluded that, assuming the remote host is listening on the specified TCP port, these three messages will take place in close succession. Lastly, because source and destination IP addresses are visible, the TCP handshake can easily be spotted in the network trace. The tests conducted showed that the TCP handshake was easily identifiable among the encrypted traffic, and that the size of the three messages (84 bytes for the first two messages, and 76 bytes for the third) were consistent across all platforms tested.

Encrypted `TCP Close` Discernible

As in the case of the TCP Handshake, the TCP Close is also defined in RFC 793, and the associated state diagram can be seen in Figure 4.1. From the `ESTABLISHED` state, the server can either wait for, or issue, a `FIN`. However in our case, the daytime server automatically disconnects after sending the Time and Date to the client, so in this example the server issues a `FIN` to the client, causing the server to move into the `FIN_WAIT_1` state. From this state,

the server, upon receipt of an ACK from the client, sends nothing in response, and enters the FIN_WAIT_2 state. At this point, *it* has closed its end of the TCP connection, meaning that no further data will be sent to the client. During this time, the client has entered the CLOSE_WAIT state, and by issuing the server with a FIN+ACK, it enters the LAST_ACK state. Upon receipt of the client´s FIN+ACK, the server sends the last ACK, and enters the TIME_WAIT state, where it waits for the 2MSL timer to expire. The rationale behind waiting for twice the MSL (*2MSL*), is that in the event that the server´s last ACK has been lost (and that the client did not receive it), enough time is provided for the client to time-out and resend its last FIN, prompting the server to resend the last ACK, and eventually return the TCP to its beginning state.

Analysis of the encrypted traffic showed the 4 messages comprising the TCP Close (FIN+ACK, ACK, FIN+ACK, ACK) in all implementations tested. The size of these messages was also consistent throughout all implementations: 76 bytes each. Although the tests conducted did not take into account the possibility of inferring TCP Close by observing the 2MSL timeout, it is likely that in real-world TCP traffic, the 2MSL timeout will be clearly visible amidst the encrypted traffic, and that it will help in pinpointing the location of the TCP Close.

4.2 Fingerprint of IKE/IPSec Implementations

What follows, is the collection of discriminants which form the fingerprint for each IKE/IPSec implementation. The fingerprint is first discussed and then represented in two-dimensional space by what can be termed a *circular fingerprint*. This circular fingerprint displays triangles in two concentric circles, each pair of triangles (1 in the outer circle, and 1 in the inner) denotes one fingerprint discriminant. If the two triangles form a diamond shape, this means that the discriminant was present (equivalent to a `Yes` in Ta-

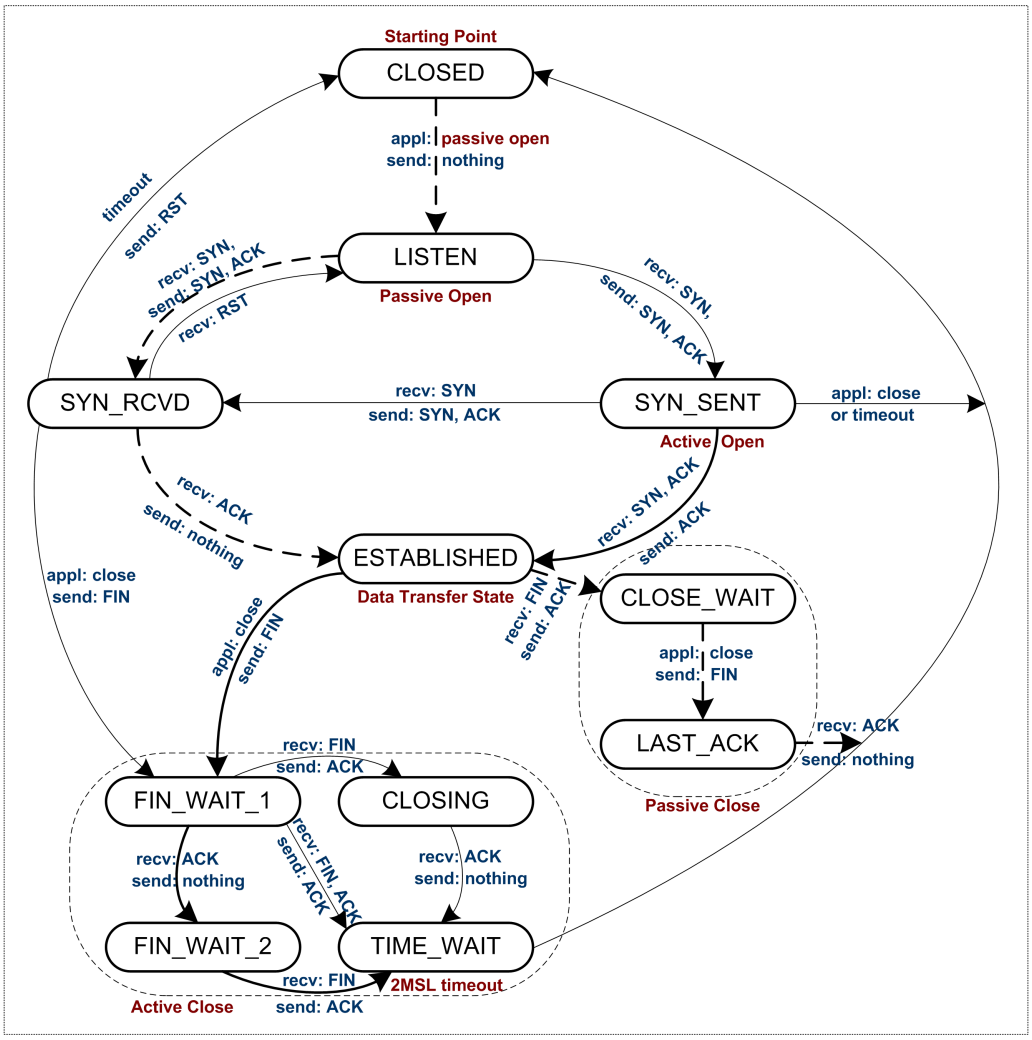


Figure 4.1: The TCP State Diagram, adapted from [101] and [120].

ble 4.6), and if the triangles form a pseudo-hourglass (which is actually the inverse space of an X), it means that the discriminant was not present (equivalent to a `No` in Table 4.6). Displaying each platform´s discriminants in a `circular` fashion, was the result of the author´s study into novel non-tabular means of data representation. Tests also showed that the order of SA Attribute Type values in the Transform Payload of MM1 can alone serve as absolute fingerprints, in all but two cases. This discriminant, christened the *TAVO fingerprint* (for Transform payload Attribute Value Order), is also presented for each of the platforms tested and the respective attributes listed in Table 4.1. The section closes with a matrix representing all fingerprints, followed by three decision trees employing the fingerprints.

4.2.1 Microsoft Windows 2003 Enterprise Server

The most significant finding resulting from the analysis of the Microsoft Windows 2000 and 2003 IKE/IPSec implementation was that their Phase 2 (Quick Mode) consisted of four messages instead of the RFC-defined 3. This is a major deviation from the standard, and likely to cause interoperability problems with other IPSec implementations. The contents and purpose of this fourth Quick Mode message (QM4) were explained in *The Cable Guy* [36], a Microsoft TechNet publication: QM4 contains a `NOTIFICATION` from the responder, the payload of which is a `CONNECTED` notification message, ``this message, which is used by IPSec peers running Windows XP or Windows 2000, is not required by the IKE standard. It is used to prevent the initiator from sending IPSec-related packets to the responder before the responder is ready to receive them.`` As was covered in the previous section, RFC 2408 makes mention of the problem of either party sending encrypted communication prior to the completion of Phase 2 (Quick Mode). The standard offers a solution to this problem, however: use of the Commit Bit, which RFC 2408 states is used ``to ensure that encrypted material is not received prior to completion of the SA establishment``. The standard further states

that in the event that the Commit Bit is set, ``the entity which did not set the Commit Bit MUST wait for an Informational Exchange containing a Notify payload (with the `CONNECTED` Notify Message) from the entity which set the Commit Bit``. The Microsoft Windows 2003 and 2000 implementations set the Commit Bit, but do not send the ISAKMP Informational message containing the `CONNECTED` notify message, rather choosing to communicate the `CONNECTED` message as the payload of a fourth message to Quick Mode. These two choices, namely the use of the Commit Bit without the accompanying ISAKMP Informational, and the addition of a fourth message to Quick Mode, place this implementation in violation of the official IETF standard on two accounts. Although untested, it is likely that these two digressions from the standard may cause interoperability problems when trying to establish an IPSec VPN tunnel between a Microsoft gateway and another vendor´s gateway. That said, it should also be noted that the net result of Microsoft´s IKE/IPSec implementation achieves its purpose by means of this alteration to the standard and works well in practice; meaning that when setting up an IPSec tunnel between two peer Windows (2000 or 2003) machines, no IPSec encrypted traffic is exchanged before IKE Phase 2 Quick Mode is completed. Given that our tests showed two other implementations which did not achieve this goal, it is commendable that Microsoft did, albeit by digressing from the standard.

For the purpose of fingerprinting, it can be noted that the two Microsoft implementations tested were the only ones which made use of the Commit Bit; they were also the only two which made use of the Vendor-ID payload. The Windows 2003 implementation contained four Vendor-ID payloads in MM1, namely: `MS NT5 ISAKMPOAKLEY`, `Microsoft L2TP/IPSec VPN Client`, `draft-ietf-nat-t-ike-02`, and `0x26244d38eddb61b3172a36e3d0cfb819` (which could not be decoded by Ethereal). Both Windows implementations tested were also the only two out of the group of five to not have the DF bit set in their IPSec communication. As a result of this, the

last two messages in Phase 1 Main Mode (MM5 and MM6) were fragmented. As has been referenced earlier, [22] explains why this is not good practice. Table 4.5 shows the Main Mode and Quick Mode exchanges. There are two important points to note from this table: firstly, the fifth and sixth messages of the Main Mode exchange (MM5 and MM6) can be seen to be fragmented, and secondly, the IP fragments have been labeled as IP and not ISAKMP by Ethereal. The reason for this being that the IP fragment contains no data in its header to denote it as part of an IPSec connection — it can only be differentiated by its encrypted payload.

Source	Destination	Protocol Type	Protocol Info.
10.0.0.1	10.0.1.1	ISAKMP	Identity Protection (Main Mode)
10.0.1.1	10.0.0.1	ISAKMP	Identity Protection (Main Mode)
10.0.0.1	10.0.1.1	ISAKMP	Identity Protection (Main Mode)
10.0.1.1	10.0.0.1	ISAKMP	Identity Protection (Main Mode)
10.0.0.1	10.0.1.1	ISAKMP	Identity Protection (Main Mode)
10.0.0.1	10.0.1.1	IP	Fragmented IP protocol (proto=UDP 0x11, off=1480)
10.0.1.1	10.0.0.1	ISAKMP	Identity Protection (Main Mode)
10.0.1.1	10.0.0.1	IP	Fragmented IP protocol (proto=UDP 0x11, off=1480)
10.0.0.1	10.0.1.1	ISAKMP	Quick Mode
10.0.1.1	10.0.0.1	ISAKMP	Quick Mode
10.0.0.1	10.0.1.1	ISAKMP	Quick Mode
10.0.1.1	10.0.0.1	ISAKMP	Quick Mode
10.0.0.1	10.0.1.1	ESP	ESP (SPI=0xe1463796)

Table 4.5: Microsoft Windows 2003 IPSec VPN tunnel set-up, showing IKE Phase 1 and Phase 2 exchanges.

The TAVO fingerprints for Windows 2003 and Windows 2000 are identical: $[1, 2, 4, 3, 11, 12]$.

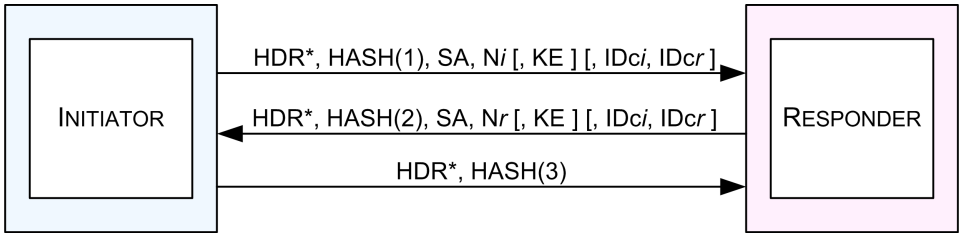


Figure 4.2: IKE Phase 2 Quick Mode, as defined in RFC 2409 [58].

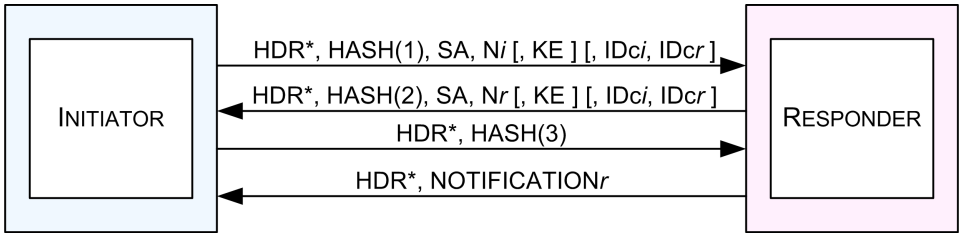


Figure 4.3: IKE Phase 2 Quick Mode, as implemented in Microsoft Windows 2000 and 2003. Adapted from [36].

4.2.2 Microsoft Windows 2000 Advanced Server

The IKE/IPSec implementation of Windows 2003 and Windows 2000 displayed many of the same characteristics, but there were also a few differences. In contrast to Windows 2003´s 4 Vendor-ID payloads, Windows 2000 MM1 carries one Vendor-ID payload: `Microsoft Win2k/WinXP`. Both implementations send ISAKMP Informational messages at tunnel tear-down, but Windows 2000 sends three such messages, whereas Windows 2003 sends four. Windows 2003 also carries five payloads in MM4, namely 4, 10, 7, 130, 130 (as per Table 4.2). The Windows 2000 MM4 carries the first three payloads (4, 10, and 7), but excludes the Private Use payloads (130, 130); the payloads appear in the same order as in Windows 2003 MM4.

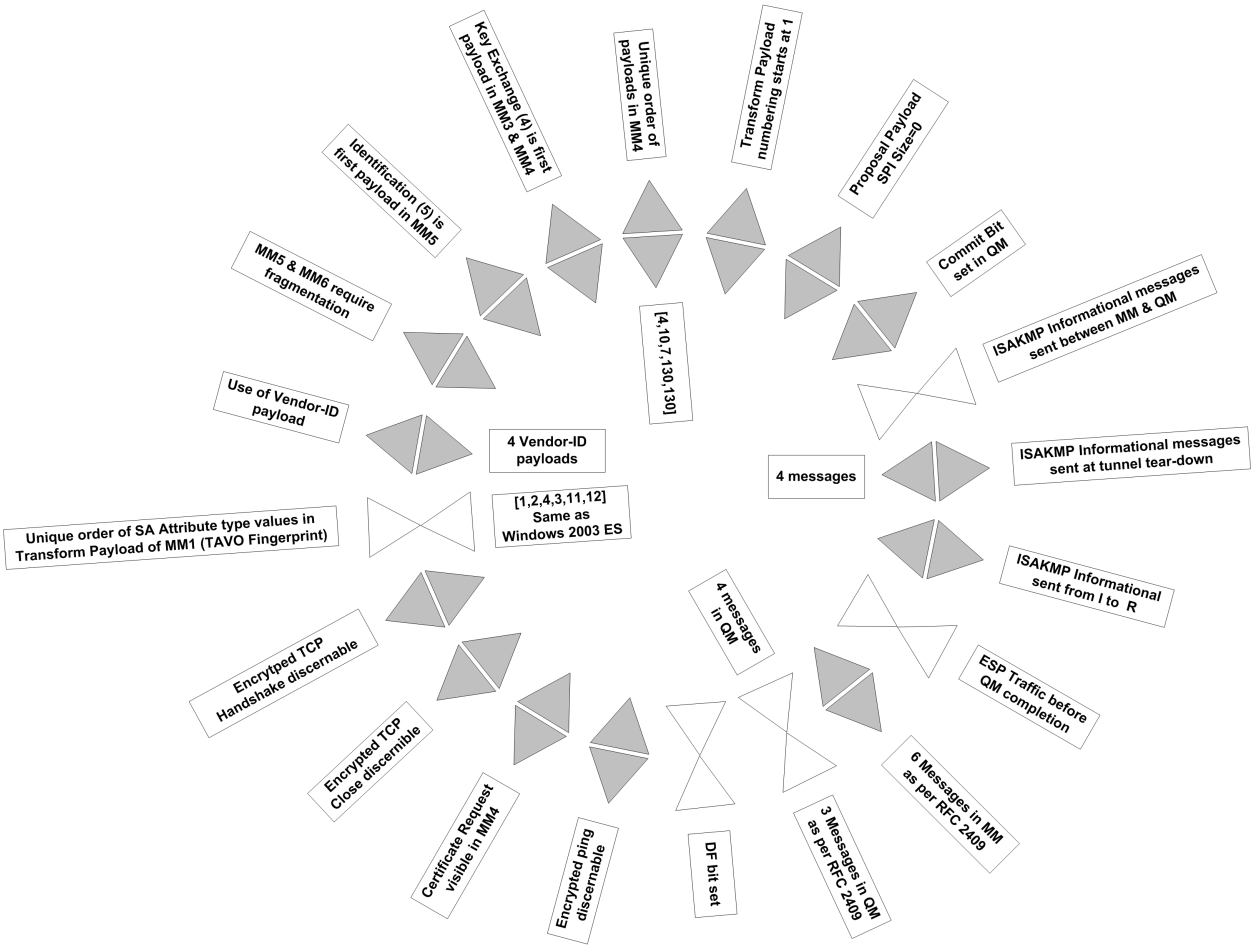


Figure 4.4: Circular Fingerprint of IKE/IPSec Windows 2003 Enterprise Server.

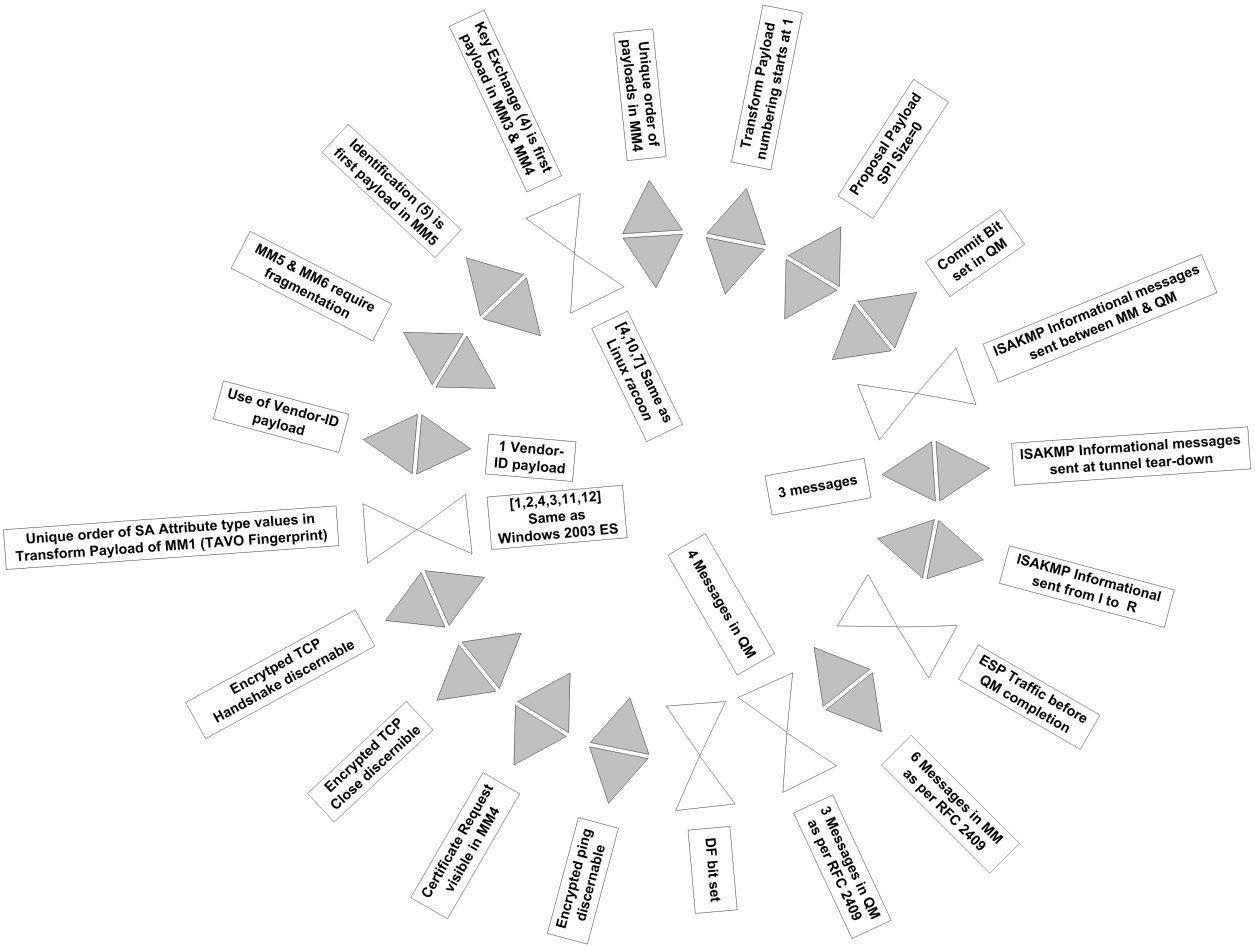


Figure 4.5: Circular Fingerprint of IKE/IPSec Windows 2000 Advanced Server.

4.2.3 Sun Microsystems Solaris 9 x86

Sun Microsystems Solaris 9 for x86 presented a range of variations that set it apart from the other implementations, and was the only implementation holding different values to the rest in the case of five discriminants.

To start with, the Solaris 9 for the x86 IKE/IPSec implementation was the only one holding a Proposal Payload SPI Size not equal to 0 (its SPI value size was 8). It was also the only one not sending ISAKMP Informational messages at tunnel tear-down. This however, may be attributed to the design of IPSec on Solaris 9, which assumes the commencement of IPSec-encrypted traffic to be the machine bootstrap process. In contrast, other implementations allow the IKE daemon (Linux) or IPSec service (Windows) to be terminated or disabled — a procedure which in turn prompts the issuance of ISAKMP Informational messages. Solaris 9 was also the only implementation where ISAKMP Informational messages were sent from Responder to Initiator, the inverse of all the others. As discussed earlier, it would appear that the Solaris 9 IKE/IPSec implementers considered x.509 certificates to be a special case of the revised mode of public key encryption, which caused the first payload in MM3, MM4, MM5, and MM6 to be `Hash (8)`, `Nonce (10)`, `Hash (8)`, `Hash (8)`, respectively — in contrast to all other implementations which treated x.509 as the public signature case, and whose MM3-MM4 and MM5-MM6 first payloads were `Key Exchange (4)` and `Identification (5)` respectively. This is also believed to be the reason for MM4 to have `UNKNOWN-ID-TYPE: 155`, `151`, and `55`[6], where the other implementations (with the exception of *isakmpd* on Linux) carry a Certificate Request payload.

To their credit, Solaris 9 implementers may well have been following RFC recommendations when choosing to interpret x.509 certificates as a special case of the revised mode of public key encryption. Referring to the advantages

[6]These three values represent payload values over three runs; that is, the payload had a different value at each run.

of authentication using the revised mode of public key encryption over that of authentication with signatures, RFC 2409 states ``This authentication mode retains the advantages of authentication using public key encryption but does so with half the public key operations`` and further ``This solution adds minimal complexity and state yet saves two costly public key operations on each side. In addition, the Key Exchange payload is also encrypted using the same derived key. This provides additional protection against cryptanalysis of the Diffie-Hellman exchange`` [58] suggesting that it may be a better option.

Two other peculiarities displayed by the Solaris 9 implementation were the fact that the tunnel setup showed IPSec-encrypted (ESP) communication being sent from Initiator to Responder prior to the completion of Phase 2 (Quick Mode). There was also one ISAKMP Informational message sent between the completion of Main Mode and prior to the start of Quick Mode. Apart from the TAVO and circular fingerprints, Solaris 9 can be uniquely identified by the order of the ISAKMP payloads in MM4, these are 10, 4, and 5, as per Table 4.2.

The TAVO fingerprint for Solaris 9 x86 is: $[3, 2, 1, 4, 11, 12]$.

4.2.4 *isakmpd* on Linux kernel 2.6

As mentioned in the previous chapter, the testing procedure included obtaining documentation from the developers´ websites, on how to configure IPSec on each of the different platforms. If the behaviour of the resulting IPSec VPN tunnels was suspicious, its configuration was re-checked, and re-deployed. If the peculiar behaviour persisted (and was consistent over several runs), it was attributed more to the implementation, than to the tester´s error; although, of course, human error on the part of the tester cannot be ruled-out. One particularly interesting behavioural pattern observed was that of *isakmpd* on Linux, and is discussed below. *isakmpd* was originally written for the OpenBSD operating system, by Hallqvist and Provos [53],

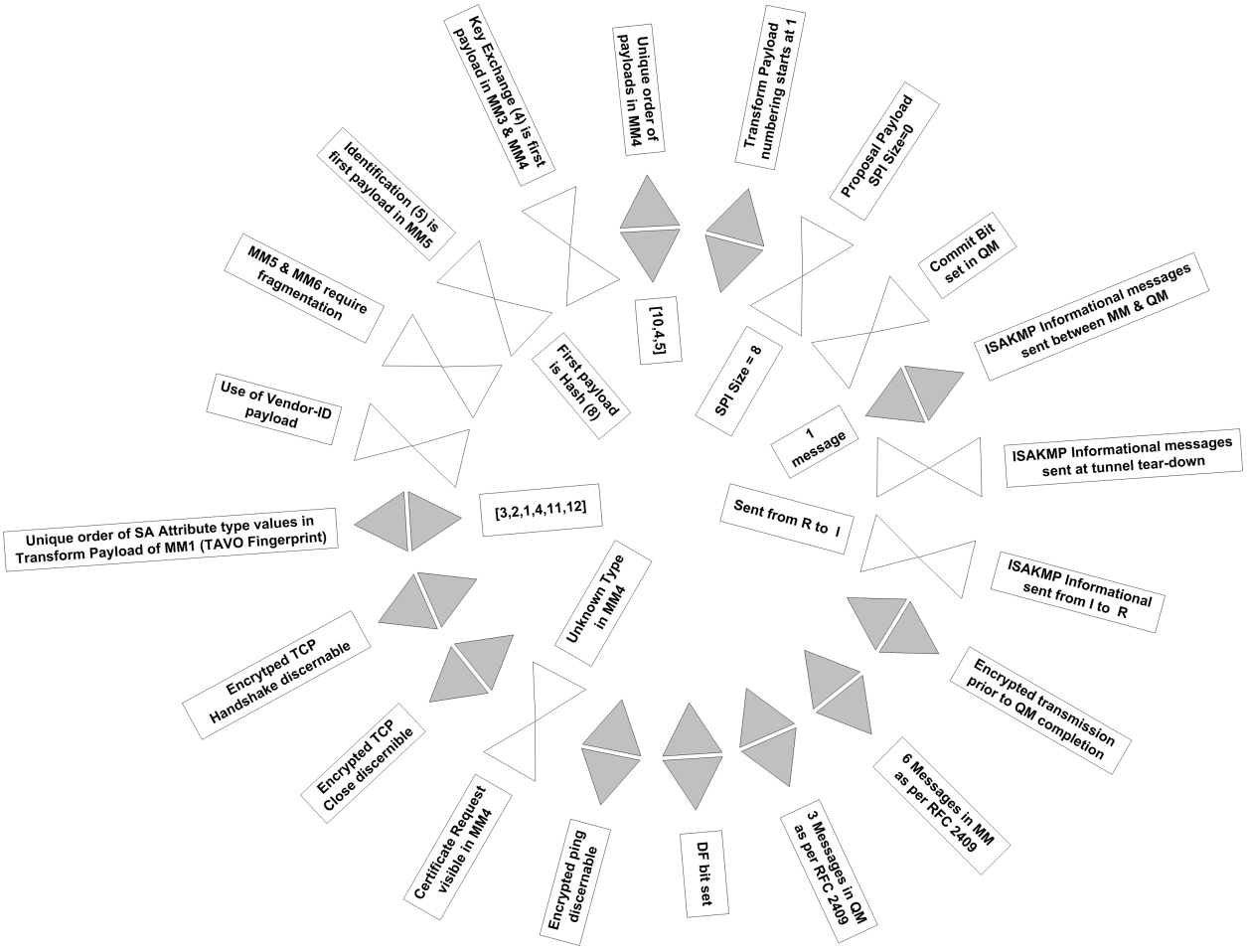


Figure 4.6: Circular Fingerprint of IKE/IPSec Solaris 9 for x86.

then ported to Linux by Walpuski [54].

In contrast to the other IPSec implementations (with the notable exception of Solaris 9) which initiate IPSec negotiation only when one side attempts to communicate to the other (through *ping* or *telnet* for example), in the tests conducted for this study, *isakmpd* would attempt to establish an IPSec tunnel with its peer, at the instant that the daemon was run. This meant that there would be a period of time during which one gateway would attempt to establish a tunnel with an unreachable peer. This behaviour is not particularly undesirable. However what made it more unusual (and perhaps undesirable by reason of its imprudent use of bandwidth), was that the test data consistently showed that both peers would establish an IPSec tunnel (that is, successfully complete IKE Phase 1 *and* Phase 2), send two or three IPSec-encrypted (ESP) packets (which by their payload size appeared to be the ICMP `ECHO REQUEST/REPLY`s that formed part of the scheduled tests), and then proceed to re-negotiate the IPSec tunnel. The packet captures therefore show two complete sets of Phase 1 (Main Mode) and Phase 2 (Quick Mode) negotiations in very close succession, often interleaved with ESP traffic. After the second IPSec tunnel had been established, communication would proceed uninterrupted. The tunnel re-establishment was therefore not consistent (the IPSec tunnel would be re-established only once). This ruled out the possibility that a timer value (such as *Life-Duration*) was expiring and causing either IKE Phase to rekey[7].

Another peculiarity present in the *isakmpd* IKE/IPSec implementation was that the Certificate Request payload was not visible in MM4. However MM3 and MM4 both had Key Exchange (4) as the first payload, and the first payload of MM5 and MM6 was Identification (5), like on all the other platforms (with the notable exception of Solaris 9). The absence of the Certificate Request (7) payload in MM4 meant that the order of payloads

[7]In IKE, *re-keying* is a process whereby new keying material is generated (to be used as session keys, for example).

was 4, 10, as opposed to 4, 10, 7 (as observed in Windows 2000), making it a unique identifier. Lastly, the *isakmpd* implementation was the only one among all those tested in which the Transform Payload numbering scheme began at 0; all others began at 1. The TAVO fingerprint for *isakmpd* on Linux 2.4.6.8.1 is $[1, 2, 3, 4, 11, 12]$.

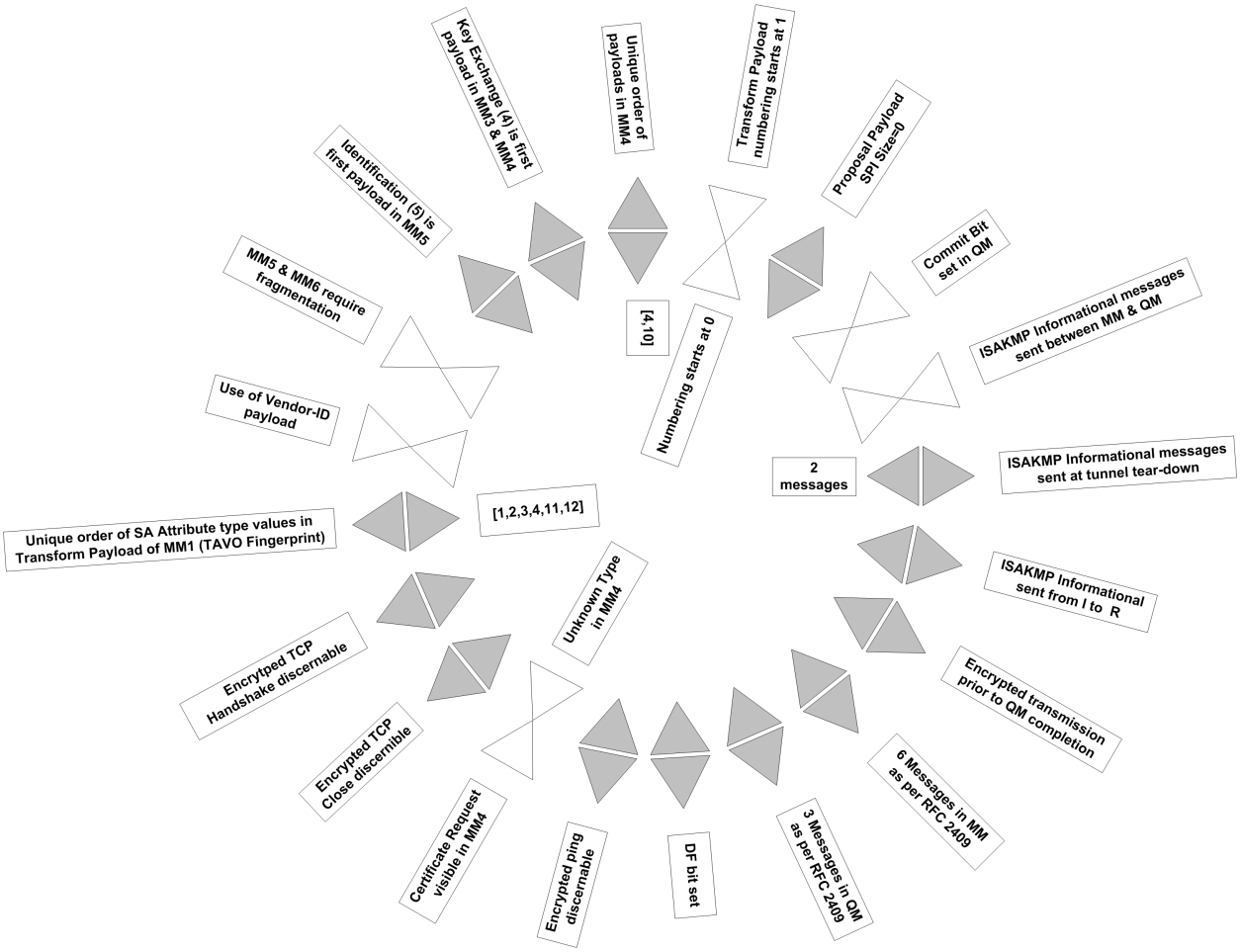


Figure 4.7: Circular Fingerprint of IKE/IPSec *isakmpd* on Linux 2.4.6.8.1 kernel.

4.2.5 *racoon* on Linux kernel 2.6

Racoon is part of the KAME project [102] which ``is a joint effort of six companies in Japan to provide a free IPv6 and IPsec (for both IPv4 and IPv6) stack for BSD variants to the world.`` Racoon was initially developed for the FreeBSD/NetBSD/OpenBSD platforms, but has also been ported to Linux. It should be noted that among the three IKE/IPSec implementations for Unix and Linux that were tested (Solaris, Linux *isakmpd*, and Linux *racoon*), *racoon* was the most straight-forward and easy to configure. It was also the one with the least questionable behaviour. For example, it was the only non-Windows implementation *not* to allow encrypted communication prior to QM completion, the ordering of payloads in MM4 was 4, 10, 7, identical to that of Windows 2000, similar to *isakmpd´*s 4, 10, and also similar to Windows 2003´s 4, 10, 7, 130, 130. Racoon was the implementation which had the most attributes in common with those of other platforms (suggesting willing adherence to the standard).

The TAVO fingerprint for *racoon* on Linux 2.4.6.8.1 is $[11, 12, 1, 3, 2, 4]$.

4.2.6 OpenBSD 3.5 i386

The reader will note that there is no `OpenBSD` column in Table 4.6, and that is because while configuring the OpenBSD IKE/IPSec implementation (*isakmpd*), a bug which caused the machine to crash due to a kernel panic, was discovered by the author of this work. OpenBSD is a robust operating system, it carries the slogan `*Secure by Default*`, which means that the default installation has been hardened to weed-out the most common security vulnerabilities, and can be considered secure out of the box (and certainly, more secure than other operating systems). The scope of the bug was limited, since it required three commands to be run as *root* (administrator) before rendering the machine remotely vulnerable; so it was not a critical situation which would immediately affect thousands of OpenBSD in-

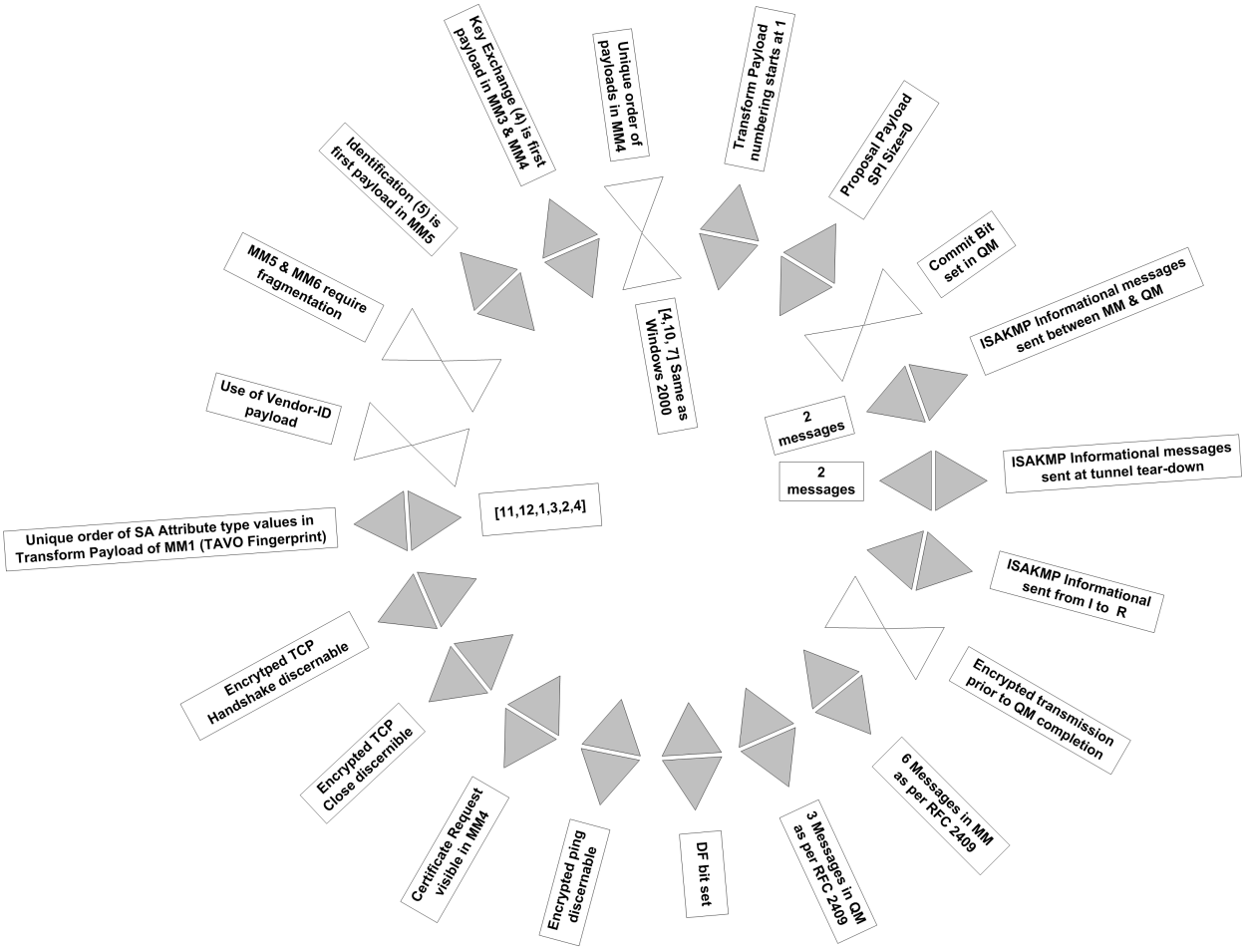


Figure 4.8: Circular Fingerprint of IKE/IPSec *racoon* on Linux 2.4.6.8.1 kernel.

stallations the world over (although until the patch is applied, they remain vulnerable). It was therefore decided by the author to disclose the bug in a manner that would ensure that the least inconvenience would be incurred by its revelation. A three-tiered approach was chosen: the Computer Emergency Response Team Coordination Centre (CERT-CC) would be contacted first, followed by the OpenBSD team, then after a week, pending the release of a patch to fix the bug, it would be publicly disclosed.

As planned, CERT-CC was contacted first, but no communication was received in reply. The OpenBSD team was contacted next, and within 12 hours a patch was issued that stopped the machine from crashing (due to its non-critical nature, it was categorised as a `Reliability Fix`). The public disclosure was then made to the SecurityFocus *Bugtraq* mailing list, and has since appeared mirrored in other sites: Checksum.org, Secunia.com (advisory SA12394), Securiteam.org, and Zone-h.org, among others.

The bug rested in the bridging code of OpenBSD 3.5, and its handling of IPSec traffic. The steps taken were:

- `# ifconfig bridge0 create` to create the bridging interface.
- `# brconfig bridge0 add fxp0 add fxp1 up` to add two network interfaces (fxp0 and fxp1) to the bridge interface, and to enable it.
- `# brconfig bridge0 link2` to enable IPSec processing on the bridge.

Although the patch provided by OpenBSD stopped the machine from crashing, it did also prevent the bridge from working, so the scheduled lab tests for this platform were abandoned[8]. There is reason to believe, however, that the circular fingerprint diagram for OpenBSD would resemble that of *isakmpd* on Linux 2.6.8.1, since OpenBSD uses same IKE daemon. The TAVO fingerprint, however, is likely to be different.

[8]It is expected that this problem will have been resolved in the next version of OpenBSD.

4.3 SSL/TLS Tunnels

SSL/TLS is the newcomer to the VPN arena, and due to attractive pricing and easy installation, smaller organisations are choosing SSL/TLS VPNs over the previously pseudo-defacto VPN standard: IPSec. With the aim of applying the fingerprinting methodology from Chapter 3 to SSL/TLS tunnels, and after several manufacturers of VPN devices communicated the impossibility of their laboratory equipment being used by the author for the purpose of this research, an equivalent open-source suitable product was sought out. OpenVPN was the only open-source product which offered SSL/TLS-protected tunneling, and therefore the choice was made to use it. The testing procedure was short-lived, however.

Initial tests results showed that OpenVPN connections do not pass key exchange messages in clear text. Further tests showed that neither the handshake nor the rekeying information is discernible from an OpenVPN connection. Yonan, the author of OpenVPN, was contacted, and he confirmed that this is the correct behaviour, and that ``this is probably happening because OpenVPN encapsulates the raw SSL/TLS exchange inside of another frame which is used for things like adding a reliability layer on top of UDP and implementing –tls-auth`` [129]. In order to be able to view the SSL/TLS handshake, it would be necessary to make modifications to the OpenVPN source code. The scope of this research was to fingerprint current VPN implementations, and attempt to discern information from otherwise encrypted traffic. Since to do this for OpenVPN would require that the standard implementation be modified, it was decided that this should form part of future research.

4.4 Decision Trees

In order to present a practical application of the discriminants, this section applies the fingerprints in a decision tree format, to uniquely identify each IKE/IPSec implementation. Buntine refers to the common inference task performed by decision trees as that of ``making a discrete prediction about some object given other details about the object`` [15]. Decision trees are so named because ``decisions about class membership are represented at the leaf nodes. ... An example is classified using this tree by checking the current test and then falling down the appropriate branch, until a leaf is reached`` [15]. The decision trees employed in this section are not *true* decision trees in the sense that they are not initially grown to completion, and later pruned, as described by Buntine. These decision trees, shown in Figures 4.9, 4.10, and 4.11, follow the pattern of what Moore terms `binary categorical splits` [85], where each split is either of the form *Attribute equals value* or *Attribute doesn´t equal value*. Three examples are shown, two of which show how to isolate every IKE/IPSec implementation in four steps, and one which shows how to do this in three steps. Every intermediary node in the tree holds a discriminant value which answers to a binary (yes/no) question. This binary categorization isolates two implementations, or a set thereof. The penultimate intermediate node (that is, the node immediately above any successfully classified leaf nodes) can either ask a binary question, or discriminate between two non-binary values (such as the number of ISAKMP Informational messages, for example). The three examples shown cover ten discriminants and re-use one at a penultimate node, to demonstrate how different combinations of discriminants can be used by these decision trees, to successfully isolate each IKE/IPSec implementation.

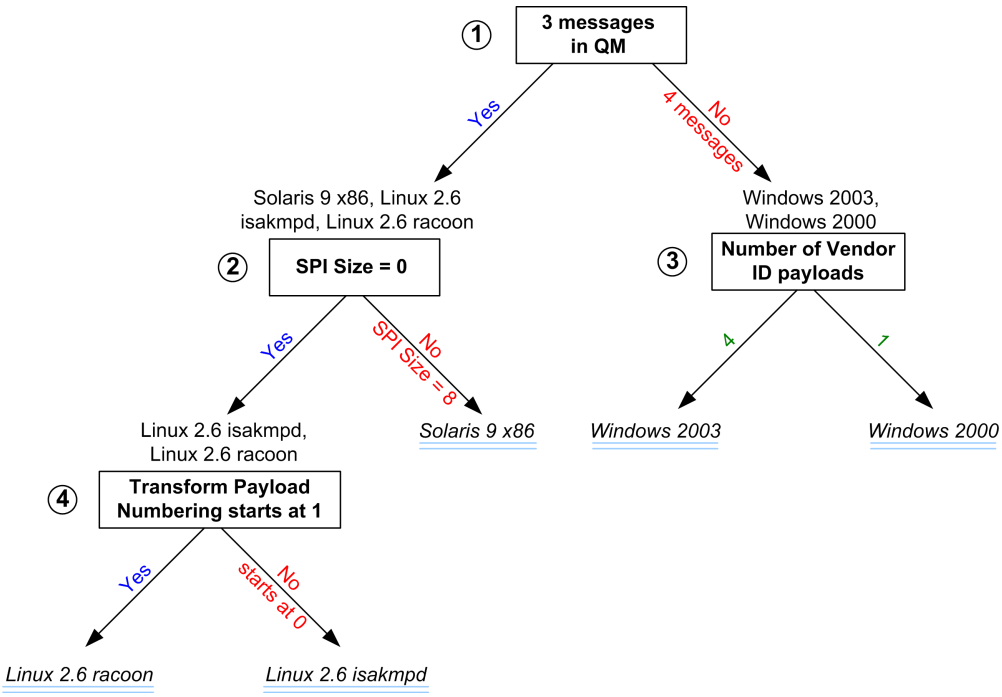


Figure 4.9: Decision tree using the discriminants to uniquely identify all platforms in *4* steps: Example 1.

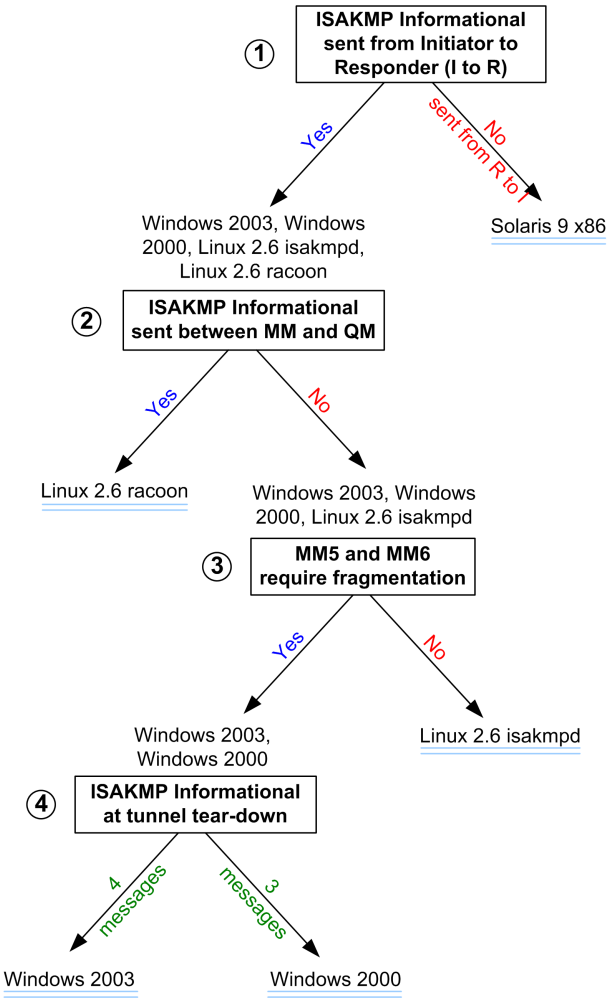


Figure 4.10: Decision tree using the discriminants to uniquely identify all platforms in *4* steps: Example 2.

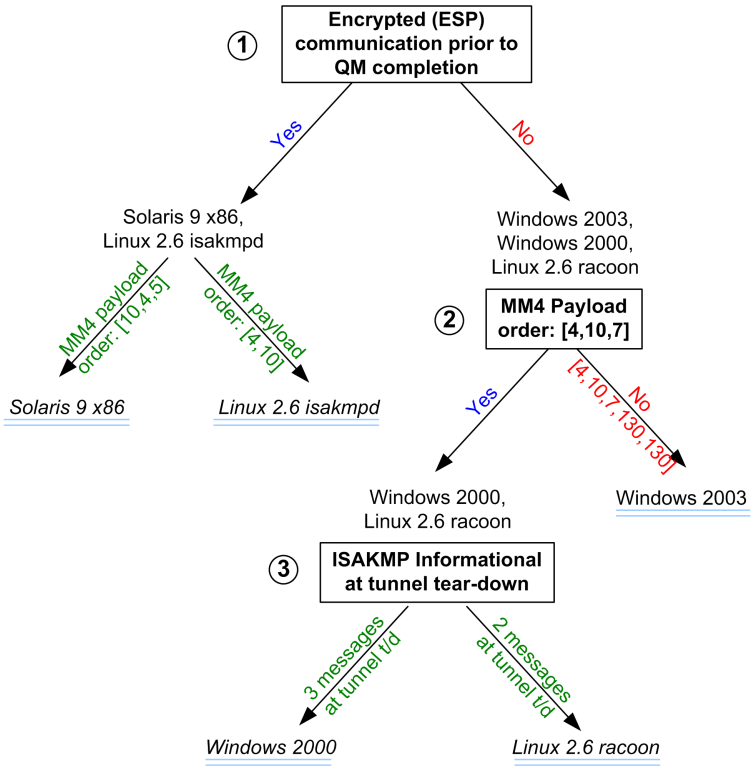


Figure 4.11: Decision tree using the discriminants to uniquely identify all platforms in *3* steps.

4.5 Summary

Our test findings led to the identification of two distinct fingerprints for each platform, the TAVO fingerprint, and the circular fingerprint (so named due to a new visualisation which was also the product of this research)[9]. Three examples of applying the circular fingerprint discriminants to a decision tree, in order to distinctly identify each platform, were also presented. These tests revealed that in addition to information that can be gleaned from the clear-text messages in the IKE exchange, there is also a selective amount of information that can be detected from the encrypted traffic, namely which packets in the encrypted stream are ICMP `ECHO REQUEST`s or `REPLY`s, when a TCP handshake is taking place, and when a TCP Close is taking place. This also led to the hypothesis that the `2MSL` timeout could provide information to confirm the presence of a TCP Close (and by inference, the presence of the initial handshake).

Not all tests were positive, however. The OpenBSD tests did not complete, due to a bug being discovered by the author in OpenBSD 3.5. The SSL/TLS tests were also short-lived, as by the OpenVPN design, protocol handshake information does not pass over the network in clear text. Future research can focus on `weakening` the standard OpenVPN package, by modifying the source code to make the SSL/TLS protocol handshake and control information visible to a passive attacker.

In addition to concluding remarks, the following chapter describes a tentative attack scenario. It shows how an attacker can make use of the findings presented in this chapter, together with concepts presented in previous chapters, to mount an attack. This scenario focuses on the practical application of these findings.

[9]The TAVO fingerprints for Windows 2003 and Windows 2000 is the same, however by using any unique criterion from the circular fingerprint, these two platforms can be differentiated.

	Windows 2003	Windows 2000	Solaris 9 x86	Linux 2.6 *isakmpd*	Linux 2.6 *racoon*
DF Bit set (in IP Header)	No	No	Yes	Yes	Yes
Any Message in Key Exchange is fragmented	Yes MM5 & MM6	Yes MM5 & MM6	No	No	No
Proposal Payload SPI Size = 0	Yes	Yes	No (8)	Yes	Yes
Commit Bit set in QM	Yes (QM2)	Yes (QM2)	No	No	No
Use of Vendor-ID Payload	Yes (4)	Yes (1)	No	No	No
ISAKMP Info. between MM & QM	No	No	Yes (1)	No	Yes (2)
ISAKMP Info. at tunnel tear-down	Yes (4)	Yes (3)	No	Yes (2)	Yes (2)
ISAKMP Info. sent from Initiator to Responder (I to R)	Yes	Yes	No (R to I)	Yes	Yes
6 Messages in MM as per RFC 2409	Yes	Yes	Yes	Yes	Yes
3 Messages in QM as per RFC 2409	No (4)	No (4)	Yes	Yes	Yes
ESP traffic before QM completion	No	No	Yes	Yes	No
Transform Payload starts at 1	Yes	Yes	Yes	No (Starts at 0)	Yes
Unique order SA Attribute Type values in Transform Payload of MM1 (TAVO Fingerprint)	No $[1,2,4,3,11,12]$ Same as Windows 2000	No $[1,2,4,3,11,12]$ Same as Windows 2003	Yes $[3,2,1,4,11,12]$	Yes $[1,2,3,4,11,12]$	Yes $[11,12,1,3,2,4]$
Unique number of payloads in MM4	Yes $[4,10,7,130,130]$	No $[4,10,7]$	Yes $[10,4,5]$	Yes $[4,10]$	No $[4,10,7]$
Key Exchange (4) is first payload type in MM3 & MM4	Yes	Yes	No MM3: Hash (8) MM4: Nonce (10)	Yes	Yes
Identification (5) is first payload type in MM5 & MM6	Yes	Yes	No Hash (8)	Yes	Yes
Cert. Request (7) payload type is visible in MM4	Yes	Yes	No UNKNOWN-ID-TYPE:155,151,55	No	Yes
Encrypted *ping* discernible	Yes (108 bytes)	Yes (108 bytes)	Yes (108 bytes)	Yes (108 bytes)	Yes (108 bytes)
Encrypted TCP Handshake discernible	Yes (84B, 84B, 76B)	Yes (84B, 84B, 76B)	Yes (84B, 84B, 76B)	Yes (84B, 84B, 76B)	Yes (84B, 84B, 76B)
Encrytped TCP Close discernible	Yes (76B each)	Yes (76B each)	Yes (76B each)	Yes (76B each)	Yes (76B each)

Table 4.6: IKE/IPSec Fingerprint Matrix.

Chapter 5

Conclusion

Fingerprinting can be thought of as a special case of traffic analysis. For an attacker, this could be a means by which to narrow down the search space for exploitable weaknesses on the target system. For forensic examiners, it could provide circumstantial evidence that would help determine *how* an attack was carried out, and how the vulnerable systems were identified, and could help to reconstruct a sequence of events that resulted in the cyber crime under investigation. Studying the research findings of studies in fingerprinting, could aid a software developer interested in adhering as closely as possible to the protocol standards, and designing a secure system that leaks as little unnecessary information as possible: to coin the term, a *reticent communicator*.

The results obtained as a product of this present research represent an attempt to contribute to the field of information gathering, specifically to the subject of traffic analysis, by exposing a new avenue of information gathering, namely that of fingerprinting the endpoints of a VPN tunnel. This study extracted discriminants from the IPSec VPN connection summary, from the IKE Phase 1 and Phase 2 exchange, and pointed to three instances of encrypted traffic leaking traffic information. Tests were conducted on four operating systems, using five IKE/IPSec implementations to provide empirical

data, and the positive outcome of these test suggests that further research is likely to produce finer-grained fingerprints, which will be addressed in greater detail in Section 5.2. In addition to the discriminants extracted, which can be used in a decision tree in order to isolate any of the five implementations, a singular fingerprint was also presented, namely the Transform-payload Attribute Value Order (or TAVO) fingerprint, which can be uniquely[1] used to identify an IKE/IPSec implementation.

Before detailing areas of future research, a potential attack scenario will be presented, in order to suggest a practical application for the VPN-tunnel fingerprints. The attack scenario can be modeled on the network in Figure 3.2, replacing `Branch Office` and `Head Office` for `Corporation A` and `Corporation B` respectively.

5.1 Potential Attack Scenario

Corporation A and Corporation B (hereafter `A` and `B` respectively) have exercised control over the telecommunications market for well over ten years, in a kind of duopoly. Even industry analysts suggest the possibility that internal agreements between the two corporations may be the reason for this strangle-hold. A new corporation, Corporation C, is determined to break into the market, but finds its efforts rendered ineffective by what could amount to an ambush by A and B. Corporation C (hereafter `C`) lodges a complaint with the regulatory body, and this story gets covered by the news media. Individual X (hereafter `X`), not affiliated to either party, senses a business opportunity, and decides to dedicate some of his spare time to monitor A and B´s networks, in the hope of obtaining incriminating information which

[1]The TAVO fingerprint can uniquely identify any of the five implementations tested with varying levels of detail: In the case of Linux, both *isakmpd* and *racoon* can be uniquely identified; in the case of Windows, both Windows 2000 and Windows 2003 implementations can be identified as `Windows` implementations. The TAVO fingerprint however, is unable to discern between them.

he can then sell to C.

X is a dedicated attacker. He starts by noting IP address ranges for both corporations, available from DNS records. He notes the route taken for requests to both A and B´s websites over a period of two weeks, and confident that he knows each corporation´s Internet Service Provider (ISP), he inserts a router close to A´s network, and another one close to B´s (near the customer edge). Using one of the path altering attacks described in Chapter 3, he manages to re-route both corporations´ network traffic through his routers. X discovers encrypted communication between both corporations, and he knows it is a good lead to follow. By monitoring the link between A and B, and using filters to isolate the encrypted traffic, he manages to capture tunnel setup traffic, and concludes that the encrypted traffic is the result of an IPSec tunnel. X studies the encrypted traffic in order to determine a connection profile. He knows that traffic from an HTTP server, for example, appears different than traffic from an SMTP server; he also knows that email sent via an encrypted tunnel is likely to be sensitive in nature. By reading security news, X knows that there is a likelihood that either A or B (maybe even both) have placed their VPN gateways behind the corporate firewall, meaning that the encrypted tunnel *passes through* the firewall. X knows that if he can gain control of either VPN endpoint, he will be one step closer to obtaining the corporate emails he seeks. He simulates a small network at home, configures an SMTP server, studies SMTP traffic patterns over an un-encrypted link, and decides to apply his findings to the encrypted link he is monitoring. Within a short period of time, X is able to trace TCP connections over the encrypted link and discerns SMTP traffic, he now needs control of either tunnel endpoint. He observes the next tunnel-setup, and by studying the IKE exchanges, he determines what operating system A and B´s VPN gateways are running. All X has to do now, is check through security archives for known exploits against the target system, or simply wait until the next exploit is discovered, and run it before the network administrators

have a chance to apply the fix.

What happens next depends on many factors: What operating system is it? Has there been a recent vulnerability targeting it? Have either A or B´s gateways been patched lately? How skilled is X? and most importantly, how seriously do A and B take security? If X successfully mounts the attack, his steps can be traced by the forensic examiners. ISP logs will show that an illegitimate *next hop* appeared for a period of time, and then disappeared; around the time of the suspected intrusion. By observing that traffic, and knowing that encrypted communication is not necessarily *secret*, the forensic examiner will likely trace X´s steps, and provide this reconstructive evidence in court.

5.2 Future Work

The purpose of the abovementioned scenario is not to trivialise an attack, but rather to demonstrate its plausibility. As has been shown, *fingerprinting* can play a role, however minor, in mounting an attack. It is hoped that this example highlights the potential and need for further research on this topic. Future work on the topic of fingerprinting IKE/IPSec implementations can be done on various fronts. One way to look for more discriminants, is to peruse the RFCs which comprise the IPSec standard. RFC 2409 for example, lists 48 instances of MUST requirements, and 12 instances of MUST NOT requirements. It is optimistic to think that all vendors have adhered to all of these, and even assuming that this is the case, there is a likelihood that the RFCs have been followed in slightly different ways. Table 4.6 serves as evidence to this assertion. Similarly, a study of which IKE/IPsec implementations support proprietary cryptographic algorithms, could yield fingerprint discriminants. This could be done by `querying` the remote implementation with a—to use terminology introduced in Chapter 4—MM1 message specifying multiple Transform Payloads (supporting different algorithms), and

observing the response.

Another potential avenue for research is the study of IKE/IPSec implementations in VPN devices. The results of this may be of interest, since vendors are integrating IPSec and Firewall capability into one device, meaning that successfully fingerprinting the VPN concentrator is the same as fingerprinting the firewall. Lastly, one method of fingerprinting, a variant of which has been tried by Hills in [62], is that of fingerprinting through service disruption — that is, how do different IKE/IPSec implementations react in the face of errors? Force errors on a link for time t, and watch the protocol recover.

A related area for future research is that of ways to counter fingerprinting: How can the discriminants detailed in Chapter 4 be rendered ineffectual? In order to mask the TAVO fingerprint, for example, the attribute value order in the Transform Payload could be *normalised*; that is, since the message in which it is carried, is sent unencrypted, an extra *hop* could be inserted within the organisation, which would either randomly rearrange the attribute values, or rewrite them to make them appear to be the TAVO fingerprint of another operating system. ICMP packets, as another example, can contain payloads of arbitrary length. In fact, several *ping* implementations provide an option to set the amount of payload the ICMP packets should contain. Masking ICMP and TCP traffic so they are not discernible within the encrypted stream does not fall directly under countering *fingerprinting*, but it is a related area of research. On a different note, a way to counter IKE fingerprinting is to make use of *manual keying*, which replaces IKE by a static file to provide the information that IKE uses to negotiate a connection.

To conclude, a word on finer-grained fingerprints. It was stated earlier that further research could produce finer-grained fingerprints of IKE/IPSec implementations. In short, if the IKE Phase 1 and Phase 2 exchanges were to be viewed as forming part of a state machine, such as the ISAKMP protocol state machine as defined in RFC 2408 [83], then entering any unknown state

could potentially lead to a fingerprint —not to mention a security vulnerability. For example, RFC 2408 states that Notify error-type messages, like `INVALID-PAYLOAD-TYPE`, `PAYLOAD-MALFORMED`, `UNSUPPORTED-EXCHANGE-TYPE`, and others, ``can be error messages specifying why an SA could not be established``, but does not make clear whether these messages can (or MUST or SHOULD) be sent at any later point in the exchange. Since the third and fourth messages of Main Mode (MM3 and MM4) are still in clear-text, it is not implausible that their corresponding payloads could be tampered with. The payloads in MM3 and MM4 are the Key Exchange and the Nonce payloads (in the case of authentication using signatures). The state the the IKE protocol machine would enter if either one of these payloads were null, or in some way mangled, is not explicitly defined[2]. This scenario could be simulated either by replaying a recorded connection and altering the payload contents, or by altering the message contents of a new connection using for example, the `libipq` library which is part of the Netfilter project [103].

[2]These cases are at least not *clearly* defined, as the author did not find reference to them in the RFCs.

Glossary

AH

Authentication Header (AH) is the part of IPSec that ensures that data packets all come from the same source and have not been tampered with [82].

Border Gateway Protocol (BGP)

The Border Gateway Protocol (BGP) is an interdomain routing protocol designed for the global Internet. Exterior border gateway protocols (EBGPs) communicate among different autonomous systems. Interior border gateway protocols (IBGPs) communicate among routers within a single autonomous system. It is defined in RFC 1163 [20].

CA

A Certification Authority (CA) is a third party that verifies user identity through a series of requirements, resulting in the issuance of a digital certificate. CAs have degrees of classes of assurance they offer, based on the due diligence performed to verify an individuals identity [31].

CAIDA

The Cooperative Association for Internet Data Analysis (CAIDA), based at the University of California´s San Diego Supercomputer Centre. It´s a collaborative undertaking among organizations with a strong

interest in keeping primary Internet capacity and usage efficiency in line with ever-increasing demand. CAIDA provides the world with a neutral framework to support cooperative technical endeavors that have the potential to be critical in meeting the demands of an exponentially growing system of networks [17].

Cisco Discovery Protocol (CDP)

The Cisco Discovery Protocol (CDP) is a Media- and protocol-independent device-discovery protocol that runs on all Cisco-manufactured equipment including routers, access servers, bridges, and switches[20].

DDoS Attack

Distributed Denial of Service Attack. A DoS attack the origin of which is spread among a large number of networked hosts on the Internet, thus launching the DoS attack in a *distributed* manner.

DH key agreement

The Diffie-Hellman (DH) key agreement is a cryptographic technique which allows two parties to exchange random numbers, perform a calculation and exchange the results to produce a new, apparently random number which can be used as a key or Shared Secret. Even an eavesdropper with total knowledge of all the exchanges involved cannot compute the same secret number [82].

DNS

The Domain Naming System (DNS) is a distributed database which provides the mapping/translation between the domain name and the individual IP address allocated to that host [66].

DoS Attack

Denial of Service Attack. An attack in which a server is targeted, and its resources exhausted by a large number of spurious requests, thus denying service to legitimate users.

EIGRP

Enhanced Interior Gateway Routing Protocol (EIGRP), an advanced version of IGRP, which ``provides superior convergence properties and operating efficiency, and combines the advantages of link state protocols with those of distance vector protocols.`` [20].

ESP

Encapsulation Security Payload (ESP) is the part of IPsec that ensures that all data packets are encrypted to prevent eavesdropping [82].

FIPS

Federal Information Process Standard (FIPS) is a set of standards that describe the handling and processing of information within governmental agencies [35].

HSRP

The Hot Standby Router Protocol (HSRP) ``enables two or more devices to work together in a group, sharing a single IP address, the virtual IP address. The virtual IP address is configured in each end user´s workstation as a default gateway address and is cached in the host´s Address Resolution Protocol (ARP) cache. In an HSRP group, one router is elected to handle all requests sent to the virtual IP address. An HSRP group has one active router, at least one standby router, and perhaps many listening routers`` [20].

ICMP

The Internet Control Message Protocol (ICMP), defined in RFC 792, operates at the same layer as IP, and forms an integral part of IP connectivity. ICMP messages are prepended with an IP header, and as such the can be said to traverse the network as the `payload` of an IP packet. ICMP messages are divided into two varieties: *query* and *error* messages. The nature of the ICMP message is determined by the numeric value of the *type* field (bits 0-7) and the *code* field (bits 8-15). For example, the widely used *ping* program makes use of Type 8, Code 0 (*Echo Request*) and Type 0, Code 0 (*Echo Reply*) [120].

ICMP Redirects

ICMP redirects are a special case of ICMP messages (type 5, code 0-3) that tell networked hosts that there is a better route to the destination than the current path, and that they should update their routing tables accordingly. ICMP redirects are defined in RFC 792.

IETF

The Internet Engineering Task Force (IETF) is an international non-profit, non-partisan standards body founded in 1986, composed of researchers, vendors, and individuals ``concerned with the evolution of the Internet architecture and the smooth operation of the Internet`` [25]. The IETF releases *Request for Comments* (RFCs) which are a product of a lengthy period of deliberation within a particularly assigned *Working Group* (WG), and are considered to be the ``Internet Standard`` within the subject matter they cover. Vendors and individuals are encouraged to make their products *RFC-compliant* and to be in line with the official standard, since the Internet is a distributed network of dissimilar systems, and the overall aim is to interoperate.

IGRP

Interior Gateway Routing Protocol (IGRP), a protocol developed by

Cisco Systems ``to address the issues associated with routing in large, heterogeneous networks`` [20].

IKE

The Internet Key Exchange (IKE) protocol used to exchange symmetric keys for performing IPsec [26]. IKE is defined in RFC 2409.

IKEv2

The Internet Key Exchange version 2 (IKEv2) is a re-design of the original IKE (IKEv1), to improve protocol efficiency, security, robustness and flexibility [26, 59]. IKEv2 is currently in IETF draft mode (last call), and should become an IETF proposed standard (RFC) shortly.

Internet Draft

A document that is offered for review to the IETF [26]. Upon approval this document may become an RFC.

IP

Internet Protocol (IP) is the network layer protocol for the Internet Protocol suite. IP addresses are assigned in blocks of numbers to user organizations accessing the Internet. In IPv4, each address is a 32-bit address in the form of x.x.x.x where x is an eight-bit number from 0 to 255, IPv6 addresses are 128bits in size [20].

IP Datagram

The IP Datagram is the fundamental unit of information passed across the Internet. It contains source and destination addresses along with payload data and a number of fields that define such things as the length of the datagram, the header checksum, and flags to say whether the datagram can be or has been fragmented [20].

IPSec

IP Security (IPSec) is the protocol used to give authentication and/or encryption to IP packets [26]. IPsec is defined in numerous RFCs and IETF drafts.

IRDP

The ICMP Router Discovery Protocol (IRDP) is an IETF standard, defined in RFC 1256.

ISAKMP

The Internet Security Association and Key Management Protocol is one of the component protocols of IKE, considered the basis for IKE [26]. ISAKMP is defined in RFC 2408.

ITU-T

The International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) is the telecommunication standardization sector of ITU and is responsible for making technical recommendations about telephone and data (including fax) communications systems for service providers and suppliers [20].

MAC Address

A NIC´s Ethernet or *Media Access Control* (MAC) address is a globally unique 48-bit hexadecimal string stored in read-only-memory on the NIC. The first 24-bits of the MAC address are the *Organizationally Unique Identifier* (OUI) which uniquely identifies the NIC´s manufacturer.

MAC Layer

The Media Access Control layer is a sub-layer of the Data Link Layer (Layer Two) of the ISO OSI Model responsible for media control [20].

MAC Times

Modification Access and Creation times (referred to as MAC Times) are filesystem timestamp values, denoting the time of file modification, access, or creation.

MD5

The Message Digest #5 (MD5) is a mathematical algorithm producing a short secure hash or message digest. Change any character of the message and the digest will be radically different. This forms a vital stage in producing a Digital Signature [82]. MD5 is defined in RFC 1321.

MPLS

Multi-Protocol Label Switching (MPLS) is a protocol whereby MPLS-enabled routers to make routing decisions based on labels instead of examining each IP header detail [66, 104]. MPLS VPNs are employed in service provider core networks, to allow faster routing. MPLS is defined in RFC 3031.

MSL

Maximum Segment Lifetime (MSL) is a value chosen by the TCP/IP implementation to denote the `maximum amount of time any segment can exist in the network before being discarded` [120].

MTU

Maximum Transmission Unit (MTU) is the maximum size of the layer 2 frames supported on a link. If the IP stack on a sending machine needs to send a datagram larger than the link´s MTU, then the datagram is *fragmented* [120].

NAT

Network Address Translation (NAT) and IP Masquerading serve a private network (as per RFC 1918) behind a publicly-routable address.

NIC

Network Interface Card (NIC), the physical interface that allows a computer to be connected to the network.

NIST

National Institute of Standards and Technology (NIST), formerly known as the National Bureau of Standards. A unit of the US Commerce Department which promotes open standards and interoperability in computer-based industries [35].

Nonce

In cryptography, a time-variant parameter, such as a counter or a time stamp, that is used in key management protocols to prevent message replay and other types of attacks [48].

Oakley

A key determination protocol, defined in RFC 2412 [49].

OS

An Operating System (OS) is a program that provides a user interface and an application interface (which makes it possible for application programs to run) and manages computer system resources [93].

OSPF

Open Shortest Path First (OSPF) is superior to RIP in that it offers features such as least-cost routing, multipath routing, and load balancing. OSPF is proposed as a replacement to RIP [20].

Photuris

A key negotiation and session-key management protocol, an alternative to IKE. Photuris is defined in RFCs 2522 and 2523.

PKI

Public Key Infrastructure (PKI) denotes the mechanisms used both to allow a recipient of a signed message to trust the signature and to allow a sender to find the encryption key for a recipient [26].

PSK

Pre-Shared Keys (PSKs) are a method of manual key distribution in IPsec, whereby statically-generated keys of arbitrary size are shared among the communicating parties, for the purpose of securely agreeing on a symmetric key for encrypted communication.

RFC

Request for Comments (RFC) is the primary mechanism used by the IETF to publish documents, including standards. The document series, begun in 1969, describes the Internet suite of protocols and related experiments. Not all RFCs describe Internet standards, but all Internet standards are written up as RFCs. [26, 20].

RFLPs

Restriction Fragment Length Polymorphisms (RFLPs), are a manifestation of the unique molecular genetic profile, or `fingerprint` of an individual´s DNA.

RIPv1

Routing Internet Protocol version 1 (RIPv1) is the original interior gateway protocol that shipped with UNIX systems. RIPv1 uses hops as its routing metric. [20].

RIPv2

Routing Internet Protocol version 2 (RIPv2) is an enhancement on RIPv1. It includes the subnet masks of routes in the routing updates, and it updates in multicast instead of broadcast.

Route

The path that network traffic takes from its source to its destination. The route a datagram may follow can include many gateways and many physical networks. In the Internet, each datagram is routed separately [20].

Router

A system responsible for making decisions about which of several paths network (or Internet) traffic will follow. To do this, it uses a routing protocol to gain information about the network and algorithms to choose the best route based on several criteria known as routing metrics. [20]

Routing Table

Information stored within a router that contains network path and status information. It is used to select the most appropriate route to forward information along [20].

RPC

Remote Procedure Call (RPC) is a layer 5 protocol (it can operate on top of layer 4 protocols like UDP or TCP) that allows inter-process communication between networked peers; that is, it allows an RPC-enabled program to seamlessly execute instructions on a remote system.

RPM

The RPM Package Manager (RPM) is a powerful command line driven package management system capable of installing, uninstalling, veri-

fying, querying, and updating computer software packages. Each software package consists of an archive of files along with information about the package like its version, a description, and the like [61].

SAD

The Security Association Database (SAD) contains entries defining parameters associated with each SA, such as SA lifetime, SA lifetype, ESP encryption algorithm, AH authentication algorithm, etc.

SHA1

The Secure Hash Algorithm #1 (SHA1) computes a condensed representation of a message or a data file. When a message of any length is input, the SHA-1 produces a 160-bit output called a message digest. The message digest can then be input to the Digital Signature Algorithm (DSA), which generates or verifies the signature for the message. The creator of the digital signature and the verifier of the digital signature must use the same hash algorithm [20]. SHA1 is defined in RFC 3174.

SKEME

A secure and versatile key exchange protocol for key management over (the) Internet [74].

slack space

Wasted space that results, in some file systems, if a file is smaller than or just bigger than the cluster size. A cluster is the minimum amount of space allocated to a file, and since it is a fixed quantity, if the file is smaller than this fixed size, there will be space left over; similarly, in the case that the file `spills` over the cluster boundary by one byte, then an additional cluster is assigned, which in this case, would be almost entirely empty [30].

SMTP

The Simple Mail Transfer Protocol (SMTP), defined in RFC 821, is the current standard for sending, and relaying mail on the Internet.

SPD

The Security Policy Database (SPD) contains policy entries (discard, apply IPSec, or bypass IPSec) and is consulted for all inbound and outbound traffic.

SPI

Security Parameter Index (SPI), an index used within IPSEC to keep connections distinct. A Security Association (SA) is defined by destination, protocol and SPI. Without the SPI, two connections to the same gateway using the same protocol could not be distinguished [49].

SSL

Secure Sockets Layer (SSL) is a protocol for encryption and authentication of Internet connections [26]. SSL is a proprietary standard developed and owned by Netscape Communications Corporation.

TCP

The Transmission Control Protocol (TCP) is the major transport protocol in the Internet suite of protocols providing reliable, connection-oriented full-duplex streams [20]. TCP is defined in RFC 793.

Telnet

The virtual terminal protocol in the Internet suite of protocols. Allows users of one host to log into a remote host and act as normal terminal users of that host [20].

TLS

Transport Layer Security (TLS) is the standardized version of SSL [26]. TLS 1.0 is an IETF standard, and can be considered to be SSL 3.1. It is defined in RFC 2246.

UDP

The User Datagram Protocol (UDP) is a connectionless transport protocol that runs on top of IP. UDP, like TCP, uses IP for delivery; however, unlike TCP, UDP provides for exchange of datagrams without acknowledgments or guaranteed delivery [20].

VPN

A Virtual Private Network (VPN) is a private data network that makes use of the public telecommunication infrastructure, maintaining privacy through the use of a tunneling protocol and security procedures [26].

X.509

ITU-T Specification of the format of digital certificates [26].

Appendix A

Configuring IPSec: Working Examples

In order to develop a more practical understanding of the test results presented in Chapter Four, the reader is invited to configure IPSec on two sample platforms: *isakmpd* on Linux 2.6, and *racoon* on Linux 2.6. The configuration will assume two gateways, both having standard installations of each respective platform, and proceed to explain the steps required to set up an IPSec tunnel between them. For the purpose of this exercise, it is not necessary to assume the presence of two endpoint machines, since negotiations for the VPN tunnel will commence regardless of whether traffic originates at, or passes through the gateway. In the following chapter, the reader will be presented with the steps to analyse traffic resulting from the VPN tunnel establishment. The steps listed below for the *isakmpd* implementation were obtained from the *isakmpd* manual pages (`isakmpd(8), isakmpd.conf(5), isakmpd.policy(5)` [54]), the steps listed for the *racoon* implementation were obtained from the *racoon* manual pages (`racoon(8), racoon.conf(5), setkey(8)` [102]). These sources can be consulted for a more detailed explanation.

A.1 Prerequisites

These examples will illustrate the configuration of IPSec tunnels using X.509 certificates, however before starting with the IPSec configuration, it is necessary to make sure that Layer 2 and Layer 3 connectivity is present. This can be ascertained by issuing the `ping` command, and attempting to communicate from one gateway machine, to the other. If the `ping` fails (assuming that the physical Layer 1 connectivity is in place), it is possible that a machine in the path is either not forwarding IP datagrams, Layer 2 broadcasts, or that the routing is misconfigured. On Linux systems acting as routers (or gateways), `ip_forwarding` should be enabled, and depending on the physical network layout, it may also be necessary to enable `proxy_arp`. These can be enabled by issuing the following commands as root used respectively (assuming use of IPv4):

- `echo 1 > /proc/sys/net/ipv4/ip_forward`
- `echo 1 > /proc/sys/net/ipv4/conf/if/proxy_arp` where `if` is the interface on which `proxy_arp` should be enabled.

The `netstat -nr` and `route` commands will display the routing table, and show if the host has a route to the destination. If there is still no connectivity, `tcpdump` can be used to further diagnose the problem.

X.509 certificates for use in IPSec processing, can either be self-signed or issued by a Certification Authority (CA). Since a goal of this research has been to approximate a real-world scenario, it was decided at the outset that CA-issued certificates would be used. This required the relatively trivial task of configuring a machine as a CA.

The steps involved in issuing a CA-signed certificate are:

- Install OpenSSL on the CA machine. This can be done either by installing OpenSSL from RPMs, or by compiling OpenSSL from the

source distribution. Then as root user, issue the commands listed below. In places where the command-line is longer than one line, the backslash (\) denotes a line-break. When typing the command, the backslash can be omitted, and everything can be typed in one line; however if the backslash is typed, the shell will interpret it as a line-break (and not as a Carriage Return).

- `openssl genrsa -out /etc/ssl/private/ca.key 1024`.
  This will generate a 1024-bit private key with which the CA will sign certificates.

- `openssl req -new -key /etc/ssl/private/ca.key \`
  `-out /etc/ssl/private/ca.csr`
  This will generate a *certificate request*, and use the CA´s private key to sign it.

- `openssl x509 -req -days 365 -in /etc/ssl/private/ca.csr \`
  `-signkey /etc/ssl/private/ca.key \`
  `-extfile /etc/ssl/x509v3.cnf -extensions x509v3_CA \`
  `-out /etc/ssl/ca.crt`
  This generates a X.509 certificate in response to the previously-generated *certificate request*.

A.2 *isakmpd* on Linux 2.6

The CA machine is now configured to sign certificate requests from each of the peer systems wishing to take place in the IKE/IPSec exchange. X.509 certificates need to be issued for each of the IPSec gateways wishing to communicate. This is done in a similar way to that explained above: private keys are generated on each system, these are used to generate certificate requests, and finally the certificate requests are sent to the CA to sign. On the IKE/IPSec gateways the following commands should be run as root user:

- `openssl genrsa -out /etc/isakmpd/private/local.key 1024`
- `openssl req -new -key /etc/isakmpd/private/local.key \`
  `-out /etc/isakmpd/private/10.0.0.1.csr`
  where `10.0.0.1` corresponds to the IP address of the interface that will be one end of the VPN tunnel.

isakmpd requires additional identity information to be added to the X.509 certificate. This is added into the `subjectAltName` field in the certificate using the `certpatch(8)` utility. The following commands should be executed on the CA system (with `10.0.0.1` corresponding to the IP address of the tunnel endpoint: this will serve as the certificate identity.):

- `openssl x509 -req -days 365 -in 10.0.0.1.csr \`
  `-CA /etc/ssl/ca.crt -CAkey /etc/ssl/private/ca.key \`
  `-CAcreateserial -out 10.0.0.1.crt`

- `certpatch -i 10.0.0.1 -k \`
  `/etc/ssl/private/ca.key 10.0.0.1.crt 10.0.0.1.crt`

Once these steps have completed, the IPSec gateway´s certificate should be placed under `/etc/isakmpd/certs/` and the CA´s certificate should be placed under `/etc/isakmpd/ca/`.

Two other files should be configured before the IPSec tunnel can successfully start: these are the `isakmpd.conf` and the `isakmpd.policy` files. The `isakmpd.policy` file is identical on both IPSec peer gateways; the isakmpd.conf file however, requires minor changes, such as the `Listen-on`, `Local-address=`, `Address`, `ID` and `ISAKMP-peer` variables. In short, any variable or reference to the local and peer machines, should be changed when copying the `isakmpd.conf` file from one machine to the other. The

IP addresses used in the configuration files below correspond to those of the Laboratory Layout illustrated in Figure 3.1.

The `isakmpd.policy` file used on Gateway 1 is listed in Table A.1, followed by the `isakmpd.conf` file in Table A.2.

```
KeyNote-Version: 2
Authorizer: "POLICY"
Licensees: "DN:/C=ZA/ST=Gauteng/L=Pretoria/O=UP/OU=CA/CN=
   bsd-ca-1.msc.int/emailAddress=icsa@cs.up.ac.za"
Conditions: app_domain == "IPsec policy" &&
 esp_present == "yes" &&
 esp_enc_alg == "3des" &&
 esp_auth_alg == "hmac-sha" -> "true"
```

Table A.1: The *isakmpd.policy* file as it should appear on both IPSec Gateways [54].

With the `isakmpd.conf` and `isakmpd.policy` files in `/etc/isakmpd/`, the IPSec tunnels are ready to be activated. As discussed in Chapter 4, *isakmpd* attempts to establish the IPSec tunnel as soon as the `isakmpd` daemon is run, so by running `tcpdump` on another terminal (or window) and then invoking `isakmpd`, the key negotiation traffic can be observed. Checking the process table (by issuing `ps -def | grep isakmpd` for example) will show whether `isakmpd` is running. If the daemon fails to start, `/var/log/messages` should be checked for messages issued by the `isakmpd` daemon, in order to narrow the problem domain.

A.3 *racoon* on Linux 2.6

Configuring *racoon* is very straight-forward process, and by far the most simple of all the IKE implementations tested. Only one configuration file is required: `racoon.conf`. The same certificates generated for *isakmpd* can be used with *racoon*, but they need to be placed under `/etc/certs` on each machine. The `racoon.conf` used on Gateway 1 is listed in Table A.3. The

IP addresses used in the configuration files below correspond to those of the Laboratory Layout illustrated in Figure 3.1.

As opposed to *isakmpd*, *racoon* does not start IKE Phase 1 negotiation as soon as the `racoon` command is executed, but rather waits until there is traffic that requires IPSec processing. This can be observed by running a `tcpdump` in a new terminal (or window). In the event that the *racoon* configuration contains errors, and the `racoon` daemon fails to start (for example if an encryption or authentication algorithm is specified in the configuration file, but is not compiled in the kernel), an error is sent to the standard output, as well as an entry recorded in `/var/log/messages`. These two avenues should be explored in the event that debugging be necessary.

```
[General]
Listen-on= 10.0.0.1

[Phase 1]
10.0.1.1= lx-gw-2

[Phase 2]
Connections= gw1-gw2

[lx-gw-2]
Phase= 1
Local-address= 10.0.0.1
Address= 10.0.1.1
ID= gw1

[gw1]
ID-type= IPV4_ADDR
Address= 10.0.0.1

[gw1-gw2]
Phase= 2
ISAKMP-peer= lx-gw-2
Configuration= Default-quick-mode
Local-ID= NetA
Remote-ID= NetB

[NetB]
ID-type= IPV4_ADDR_SUBNET
Network= 172.16.1.0
Netmask= 255.255.255.0

[NetA]
ID-type= IPV4_ADDR_SUBNET
Network= 172.16.0.0
Netmask= 255.255.255.0

[Default-quick-mode]
DOI= IPSEC
EXCHANGE_TYPE= QUICK_MODE
Suites= QM-ESP-3DES-SHA-SUITE

[X509-certificates]
CA-directory= /etc/isakmpd/ca/
Cert-directory= /etc/isakmpd/certs/
Private-key= /etc/isakmpd/private/local.key
```

Table A.2: The *isakmpd.conf* file, as it should appear on Gateway 1 [54].

```
path certificate "/etc/certs";

remote 10.0.1.1 {
 exchange_mode main;
 certificate_type x509 "10.0.0.1.crt" "local.key";
 verify_cert off;
 my_identifier asn1dn;
 peers_identifier asn1dn;
 proposal {
 encryption_algorithm 3des;
 hash_algorithm sha1;
 authentication_method rsasig;
 dh_group modp1024;
 }
}

sainfo address 172.16.0.0/24 any address 172.16.1.0/24 any {
 encryption_algorithm 3des;
 authentication_algorithm hmac_sha1;
 compression_algorithm deflate;
}
```

Table A.3: The *racoon.conf* file, as it should appear on Gateway 1 [102].

Appendix B

Analysing Network Traces

Once the IPSec tunnels have been configured, and data successfully sent across them (a successful `ping` or `telnet` session for example), the next step is to capture the network traces of the tunnel set-up and tear-down, and study them for any determinants that might set them apart.In order to capture the traffic, the `tcpdump` command should be run on the an intermediate machine, with the `-vvv` option for most verbosity, the `-s1514` option to denote a snap size of 1514 bytes as per the ICSA Lab recommendations (in order to capture the entire Ethernet frame), and written to a file using the `-w` option. During testing, the command used was `tcpdump -vvv -s1514 -w /tmp/trace.cap`. The explanations below assume that there are two trace files, `isakmpd.cap` and `racoon.cap` corresponding to the traces of *isakmpd* and *racoon* respectively. This is a simplified version of the steps taken during the test phase of this research. The search for four discriminants will be described below, corresponding to the first two protocol analysis steps detailed in Chapters 3 and 4. This Appendix begins with a brief introduction to using the Ethereal Network Protocol Analyser.

B.1 Using the Ethereal Protocol Analyser

Ethereal is an open-source Network Protocol Analyser, currently in version 0.10.7. It is able to decode numerous protocols[1], and contains features that rival those of high-end commercial products, such as statistical representations, graphical traffic breakdown, connection tracking, among others. Ethereal runs on several UNIX variants, as well as on Windows systems. The main Ethereal window is divided into three panes: the top one offers a connection summary, the middle shows a detailed view of the contents of the captured datagrams, and the bottom is the hexadecimal representation of the captured traffic. Navigating through the different headers and fields within each datagram in the middle pane will highlight the corresponding hexadecimal values in the lower pane. Although being able to analyse traffic by looking at the hexadecimal representation is a valuable asset, tools like Ethereal decode the captured traffic in human-readable format, leaving the hexadecimal representation merely as a reference.

B.2 Analysing the Connection Summary

By opening the `isakmpd.cap` and `racoon.cap` two separate Ethereal windows, and studying the difference between the two top panes, the first discriminant is immediately visible: the presence of `ISAKMP Informational` messages between Main Mode and Quick Mode. The trace from *racoon´s* IKE handshake shows that two `ISAKMP Informational` messages are issued after Main Mode, and before Quick Mode, whereas these messages are absent in *isakmpd´s* IKE handshake. Another discriminant also immediately visible, is that both traces show `ISAKMPD Informational` messages at tunnel tear-down. These two discriminants are shown in Figures B.1, B.2, B.3, and B.4.

[1]The complete list of protocols that Ethereal can decode is available on www.ethereal.com/docs/dfref/.

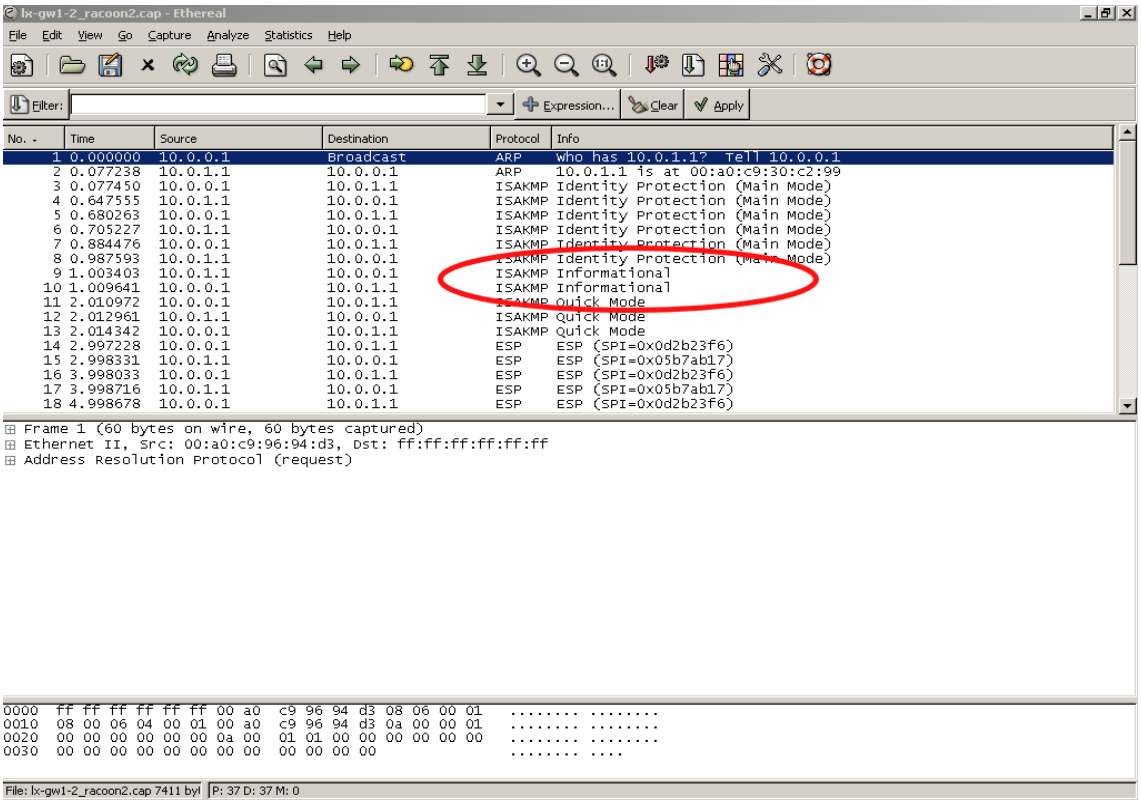


Figure B.1: Screen-capture of *racoon* IKE/IPSec trace showing ISAKMP Informational messages between Main Mode and Quick Mode.

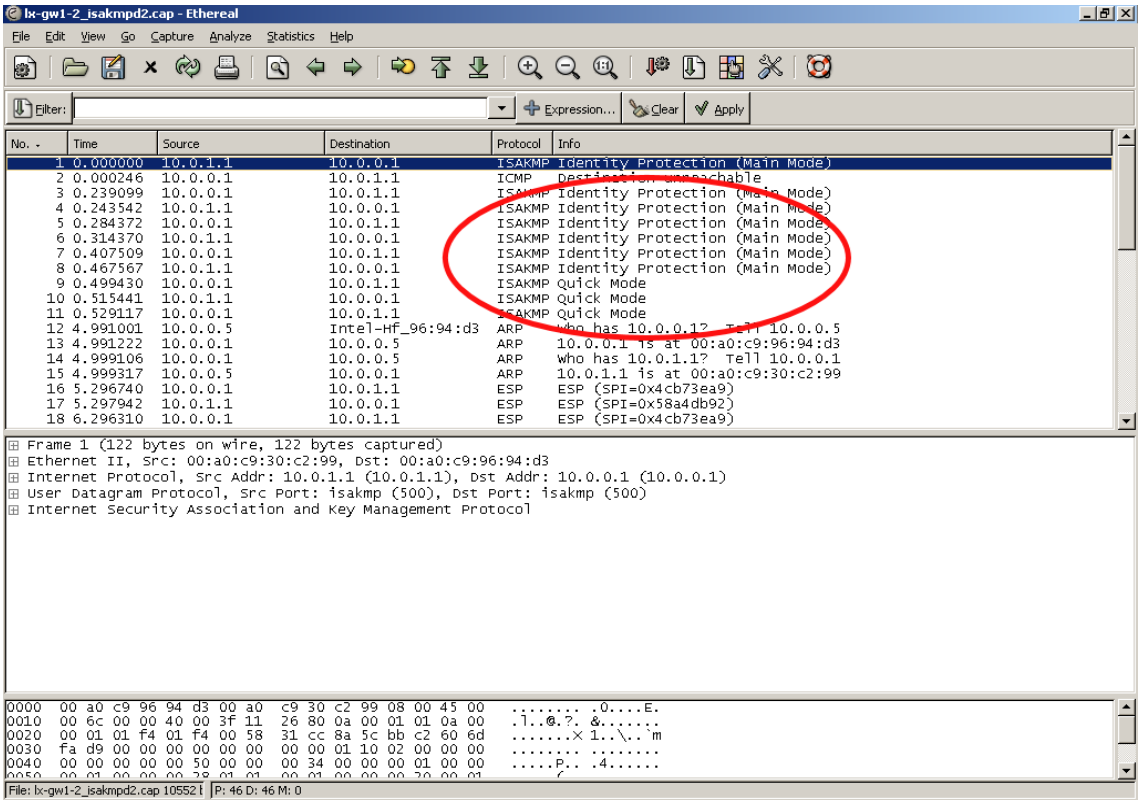


Figure B.2: Screen-capture of *isakmpd* IKE/IPSec trace showing no ISAKMP Informational messages between Main Mode and Quick Mode.

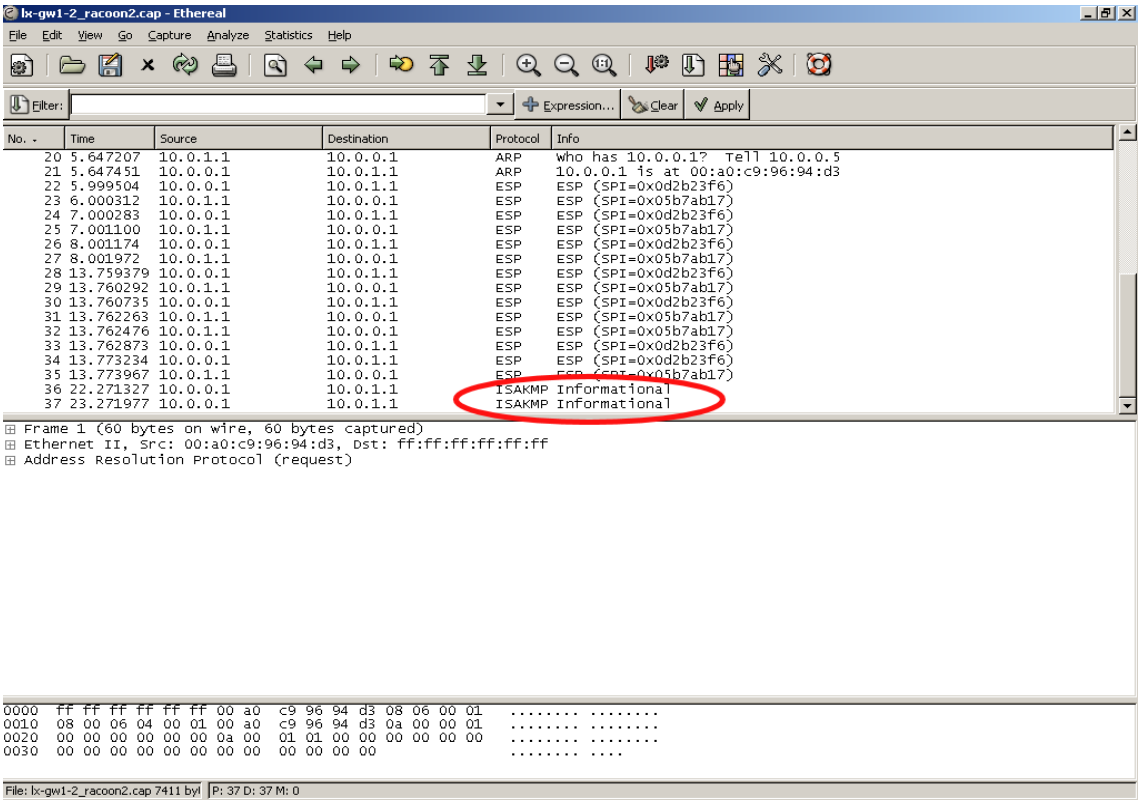


Figure B.3: Screen-capture of *racoon* IKE/IPSec trace showing ISAKMP Informational messages at tunnel tear-down.

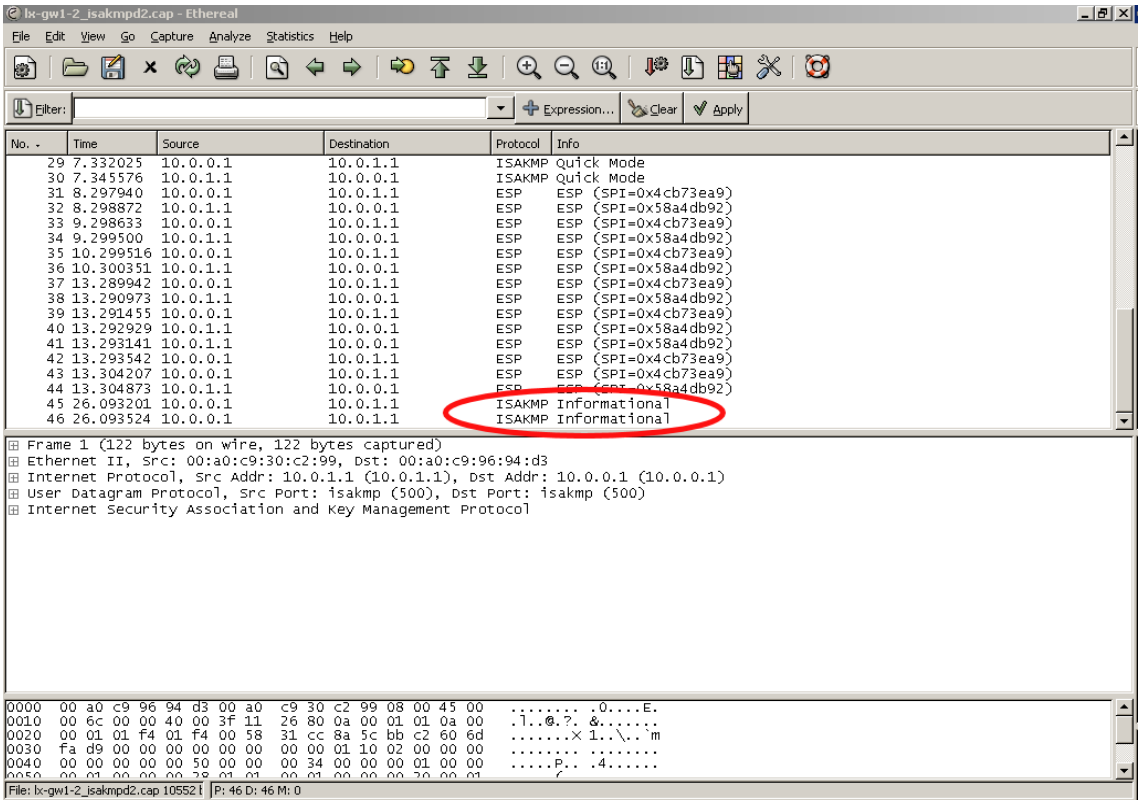


Figure B.4: Screen-capture of *isakmpd* IKE/IPSec trace showing ISAKMP Informational messages at tunnel tear-down.

B.3 Analysing the IKE Exchange Messages

By digging deeper into the IKE exchanges, and analysing the payload of the first messages, for example, we discover two other discriminants. The first is the Transform Attribute Value Order (TAVO) fingerprint referred to in Chapter 4, which is unique for both *racoon* as well as *isakmpd*, and the second is the fact that Transform Payload numbering starts at `1` for *racoon* but at `0` for *isakmpd*. These discriminants are shown in Figures B.5 and B.6 for *racoon* and *isakmpd* respectively.

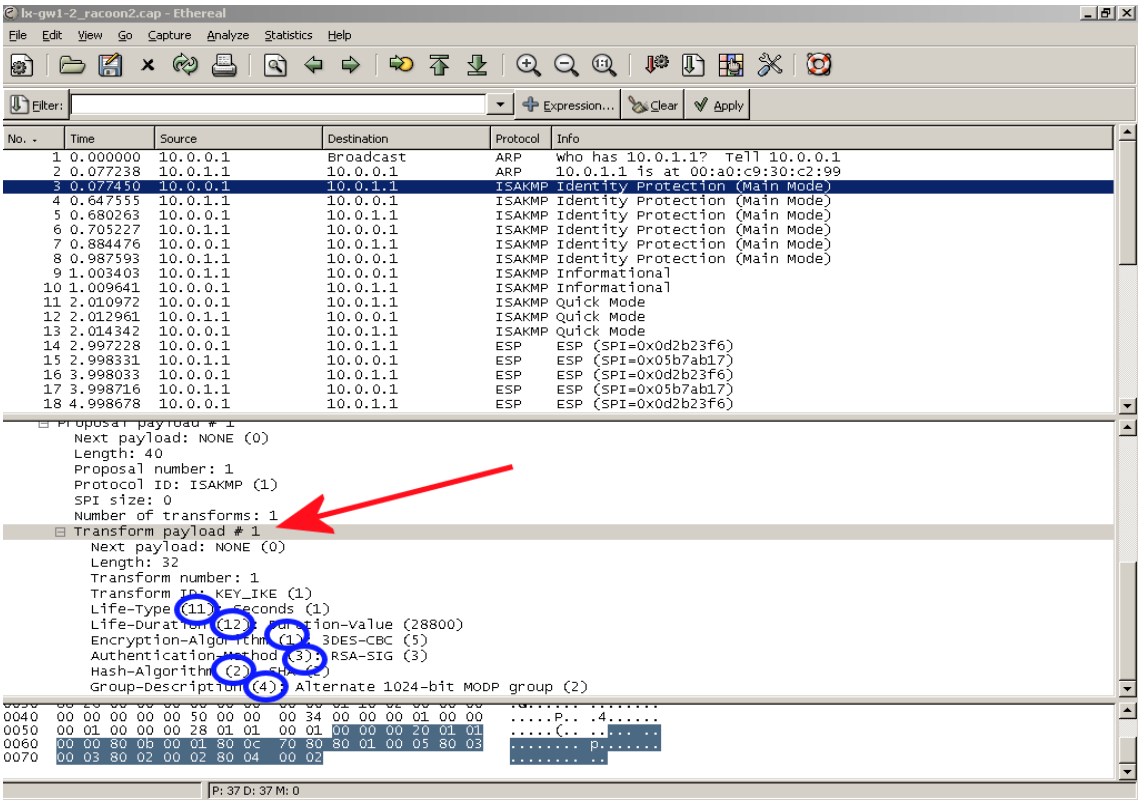


Figure B.5: Screen-capture showing Transform Payload numbering and TAVO fingerprint for *racoon* [11,12,1,3,2,4].

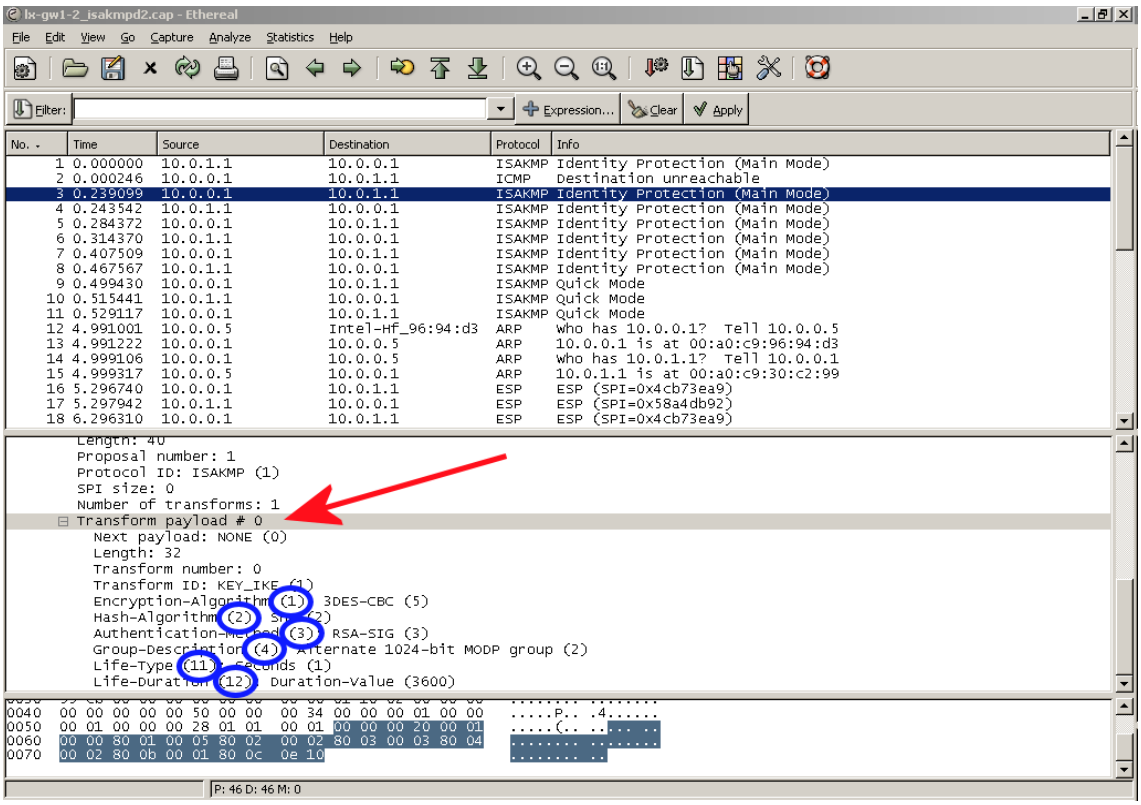


Figure B.6: Screen-capture showing Transform Payload numbering and TAVO fingerprint for *isakmpd* [1,2,3,4,11,12].

Bibliography

[1] T. Abraham and O. de Vel. Investigative profiling with computer forensic log data and association rules. In *Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, December 09-12, 2002*, pages 11–18, Maebashi City, Japan, 2002. 17

[2] P. Almquist. RFC 1349 - Type of service in the Internet protocol suite. Technical report, IETF, 1992. 26

[3] Z. Aqun, Y. Yuan, J. Yi, and G. Guanqun. Research on tunneling techniques in virtual private networks. *Correspondence Journal*, 21(6), 2000. Note: Journal entry may not be accurate (Original in Chinese, translated by babelfish.altavista.com). 6, 13, 29

[4] O. Arkin. ICMP usage in scanning: the complete know-how, version 3. [Online]. Available: http://sys-security.com, 2001. 15

[5] O. Arkin and F. Yarochkin. ICMP based remote OS TCP/IP stack fingerprinting techniques. *Phrack Magazine*, 11(Issue 57, File 7 of 12), 2001. 15, 26

[6] O. Arkin, F. Yarochkin, and M. Kydyraliev. The present and future of Xprobe2 - the next generation of active operating system fingerprinting. Technical report, Sys-Security Group, 2003. 27

[7] Aventail. Top 10 myths about SSL VPNs. Technical report, Aventail, 2002. 36

[8] F.M. Avolio. Security review: SSL VPNs. Technical report, Avolio Consulting, 2003. 30

[9] Batz. Casing the joint: what we already know about your network. [Online]. Available: http://www.blackhat.com, 2000. 22

[10] A. Bentolila. RSTunnel - reliable SSH tunnel. [Online]. Available: http://sourceforge.net/projects/rstunnel, viewed on 24 June 2004. 30

[11] R. Beverly. A robust classifier for passive TCP/IP fingerprinting. In *Passive and Active Network Measurement, 5th International Workshop, PAM 2004, Antibes Juan-les-Pins, France, April 19-20, 2004, Proceedings*, volume 3015 of *Lecture Notes in Computer Science*. Springer, 2004. 12

[12] J. Bordet. Remote SMTP server detection. [Online]. Available: http://greyhats.org, 2002. 14, 27

[13] A. Broido and K.C. Claffy. Internet topology: connectivity of IP graphs. In *Proceedings of SPIE International symposium on Convergence of IT and Communication, 2001*, 2001. 12

[14] D. Brumley and D. Boneh. Remote timing attacks are practical. In *12th USENIX Security Symposium*, pages 1–14. USENIX, 2003. 38

[15] W. Buntine. Learning classification trees. In D. J. Hand, editor, *Artificial Intelligence frontiers in statistics*, pages 182–201. Chapman & Hall, London, 1993. 91

[16] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *Proceedings of the 14th Systems Administration Conference, LISA 2000, New Orleans, December 3-8, 2000*, pages 319–327, New Orleans, U.S., 2000. 23

[17] CAIDA. CAIDA frequently asked questions. [Online]. Available: http://www.caida.org/home/about/faq.xml. 104

[18] P. Calvin. All SSL VPNs are not created equal. Technical report, Motivus Software, 2003. 37

[19] SSL VPN Central. Market research highlights. [Online]. Available: http://www.sslvpn.breakawaymg.com, 2004. 37

[20] Cisco Systems Inc. Cisco VPN solution centre glossary. [Online]. Available: http://www.cisco.com. 43, 103, 104, 105, 107, 108, 110, 111, 112, 113, 114, 115

[21] Cisco Systems Inc. Self-defending networks. [Online]. Available: http://www.cisco.com/go/security, viewed on 15 June, 2004. 22

[22] Cisco Systems Inc. IP fragmentation and PMTUD, document ID: 25885. Technical report, Cisco Systems, 2004. 60, 63, 78

[23] G. Combs. Ethereal: a network protocol analyzer. [Online]. Available: http://www.ethereal.com. 53

[24] Concept <concept@ihug.com.au>. Detection with ARP. [Online]. Available: http://ouah.kernsh.org/arposconc.txt, viewed 15 June 2004, 2000. 15

[25] IETF Consortium. Overview of the IETF, 1986. 106

[26] VPN Consortium. VPN technologies: definitions and requirements. Technical report, VPN Consortium, 2003. 13, 107, 108, 111, 114, 115

[27] V. Corey, C. Peterman, S. Sherarin, M.S. Greenberg, and J. van Bokkelen. Network forensics analysis. *IEEE Internet Computing*, 6(6), 2002. 17, 22

[28] 3Com Corporation. Understanding IP addressing: everything you ever wanted to know. Technical report, 3Com Corporation, 1996. 49

[29] British Broadcasting Corporation. Historic figures Henry Faulds (1843 1930). [Online]. Available: http://www.bbc.co.uk/history, viewed on 6 June 2004. 19

[30] DEW Associates Corporation. Cutting the slack (or maybe the fat!). [Online]. Available: http://www.dewassoc.com/kbase, viewed on 10 June, 2004. 113

[31] HID Corporation. HID - smart card glossary. [Online]. Available: http://www.hidcorp.com. 103

[32] IBM Corporation. *A comprehensive guide to virtual private networks*, volume I, II, III. IBM Redbooks, 1999, 2000. 6, 7, 13, 29, 31, 53

[33] Symantec Corporation. Symantec deepSight(TM) threat management system. [Online]. Available: http://enterprisesecurity.symantec.com, viewed on June 15 2004. 22

[34] J.P. Craiger, A. Nicoll, and B. Burham. An applied course in network forensics. In *Workshop for Dependable and Secure Systems, University of Idaho, September 23-25 2002*, 2002. 22, 28

[35] Cryptomathic A/S. Crptomathic e-security dictionary. [Online]. Available: http://www.cryptomathic.com/labs/techdict.html. 105, 110

[36] J. Davies. IKE negotiation for IPSec security associations. *The Cable Guy, June 2002 (a Microsoft TechNet publication)*, 2002. 7, 76, 79

[37] T. Dierks and C. Allen. The TLS protocol version 1.0. Technical report, IETF, 1999. 6, 41

[38] N.A. Donofrio. The issue of SSL v2 & v3, TLS v1 and FIPS 140-1. Technical report, Corsec Security Inc., 2001. 9, 41, 42

[39] M.C. Dorf. Admitting error and admitting fingerprints: is there a sound factual basis for the law? *FindLaw.com (March 2002)*, 2002. viewed June 4. 17

[40] N. Dunbar. IPSec networking standards - an overview. Technical report, Hewlett-Packard Company, Infrastructure Strategic Engineering Division, 2001. 30, 32

[41] C. Kaufman Editor. Internet key exchange (IKEv2) protocol. Technical report, IETF, 2004. Note: This is an IETF Draft and not yet an RFC. The latest version of this draft is draft-ietf-ipsec-ikev2-17.txt. 6, 36

[42] C. Kaufman Editor. Internet key exchange (IKEv2) protocol. Technical report, IETF, 2004. 35

[43] EDVOTEK. DNA fingerprinting I - identification of DNA by restriction fragmentation patterns. Technical report, EDVOTEK - The Biotechnology Education Company, 2000. 19

[44] C. Ellis and B. Schneier. Ten risks of PKI: what you're not being told about public key infrastructure. *Computer Security Journal*, 16(1):1–7, 2000. 43, 44

[45] f0bic. Examining remote OS detection using LPD querying. [Online]. Available: http://www.low-level.net, 2001. 14

[46] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. RFC 2784 - generic routing encapsulation (GRE). Technical report, IETF, 2000. 30

[47] N. Ferguson and B. Schneier. A cryptographic evaluation of IPSec. Technical report, Counterpane Internet Security, Inc., 1999. 30, 31

[48] ITS (U.S. National Institute for Telecommunication Sciences). Telecom Glossary 2000 - Definition: nonce. [Online]. Available: http://www.its.bldrdoc.gov. 110

[49] FreeS/WAN Project. FreeS/WAN glossary. [Online]. Available: http://www.freeswan.org, viewed 3 July 2004. 110, 114

[50] A.O. Freier, P. Karlton, and P.C. Kocher. The SSL protocol version 3.0. Technical report, IETF, 1996. Note: This is not an RFC but an IETF Draft. The latest version of the SSL 3.0 standard exists as draft-freier-ssl-version3-02.txt. 6, 39

[51] E. Gerck. Overview of certification systems: X.509, PKIX, CA, PGP & SKIP. Technical report, MCG, THE BELL, 2000. 42, 43

[52] J. Girard. Magic quadrant for SSL VPNs, 1H04. Technical report, Gartner Research, 2004. 37

[53] N. Hallqvist and A.D. Keromytis. Implementing Internet key exchange (IKE). In *Proceedings of the Freenix 2000 USENIX Annual Technical Conference, June 18-23, 2000, San Diego, California, USA*. USENIX, 2000. 83

[54] N. Hallqvist and N. Provos. ISAKMPd manual pages - isakmpd(8), isakmpd.conf(5), isakmpd.policy(5). 9, 85, 116, 120, 122

[55] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. RFC 2637 - point-to-point tunneling protocol PPTP. Technical report, IETF, 1999. 30

[56] S. Hanks, T. Li, D. Farinacci, and P. Traina. RFC 1701 - generic routing encapsulation (GRE). Technical report, IETF, 1994. 30

[57] M. Hanson. Fingerprint-based forensics identify Argentina's desaparecidos. *IEEE Computer Graphics and Applications*, 20(5), 2000. 19

[58] D. Harkins and D. Carrel. RFC 2409 - The Internet key exchange (IKE). Technical report, IETF, 1998. 6, 7, 9, 32, 33, 34, 35, 45, 60, 62, 67, 79, 83

[59] D. Harkins, C. Kaufman, and R. Perlman. Overview of IKEv2. Technical report, IETF, 2001. 35, 107

[60] P. Hennequin. Index RFC: encapsulation. [Online]. Available: http://www-lor.int-evry.fr/ pascal/RFC, viewed on 23 June 2004. 29

[61] R.P. Herrold. www.rpm.org homepage. [Online]. Available: http://www.rpm.org. 113

[62] R. Hills. NTA monitor UDP backoff pattern fingerprinting white paper. Technical report, NTA, 2003. 15, 101

[63] B. Huffaker, D. Plummer, D. Moore, and K. Klaffy. Topology discovery by active probing. Technical report, Cooperative Association for Internet Data Analisys, 2001. 12

[64] B. Huston. History of forensic medicine. [Online] Available: http://www.autopsy-md.com/History.html, viewed on 4 June 2004. 18

[65] ICSA labs, a division of TruSecure corporation. ICSA Labs Home. [Online]. Available: http://www.icsalabs.com, viewed on 3 July 2004, 2004. 37, 38

[66] InteRoute. Interoute - glossary. [Online]. Available: http://www.interoute.com/glossary. 104, 109

[67] V. Jacobson, C. Leres, and S. McCanne. tcpdump/libpcap public repository. [Online]. Available: http://www.tcpdump.org. 53

[68] S. Janes. The role of technology in computer forensic investigations. *Information Security Technical Report*, 5, 2000. 17

[69] E. Keneally. The law: computer forensics. *;login: The Magazine of USENIX and SAGE*, 27(4), 2002. 20, 21

[70] E. Kenneally. Computer forensics. *;login: The Magazine of Usenix & Sage*, 27, 2002. 17

[71] S. Kent and R. Atkinson. RFC 2401 - security architecture for the Internet protocol. Technical report, IETF, 1998. 30

[72] V. Klima, O. Pokorny, and T. Rosa. Attacking RSA-based sessions in SSL/TLS. Cryptology ePrint Archive, Report 2003/052, 2003. 38

[73] N. Koblitz. *A course in number theory and cryptography, graduate texts in mathematics, No 114, second edition*. Springer Verlag, 1994. 11

[74] H. Krawczyk. SKEME: a versatile secure key exchange mechanism for Internet. In *Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, page 114. IEEE Computer Society, 1996. 113

[75] B. Krishnamurthy and M. Arlitt. PRO-COW: protocol compliance on the web. Technical Report 990803-05-TM, AT&T Labs, 1999. 25

[76] ICSA Labs. IPSec technical product configuration guidelines. Technical report, ICSA Labs - a division of TruSecure Corporation, 2003. 9, 54, 58

[77] D. Lee, J. Rowe, C. Ko, and K. Levitt. Detecting and defending against web-server fingerprinting. In *Proceedings of the 18th Annual Computer Security Applications Conference, 2002*. IEEE Computer Society, 2002. 12

[78] J. Lewis. Public key infrastructure architecture, v1, 09 Jul 1997. Technical report, The Burton Group, 1997. 42

[79] F. Linder. Phenoelit - land of packets. [Online]. Available: http://www.phenoelit.de. 52

[80] F. Linder. Routing & tunneling protocol attacks. Presented at Black-Hat briefings, November 2001, Amsterdam. 52

[81] B. Lowekamp. Combining active and passive network measurements to build scalable monitoring systems on the grid. *ACM SIGMETRICS Performance Evaluation Review*, 30, 2003. 12

[82] Nortel Networks Ltd. Nortel Networks - glossary of security jargon. [Online]. Available: http://www.nortelnetworks.com. 103, 104, 105, 109

[83] D. Maughan, M. Schertler, M. Schneider, and J. Turner. RFC 2408 - Internet security association and key management protocol (ISAKMP). Technical report, IETF, 1998. 9, 60, 61, 62, 63, 64, 65, 69, 101

[84] A. Meeker-O'Connell. How DNA evidence works. [Online]. Available: http://science.howstuffworks.com/dna-evidence.htm, viewed on 8 June, 2004. 19

[85] A.W. Moore. Lecture on decision trees. [Online]. Available: http://www.cs.cmu.edu/ awm, 2001. 91

[86] D. Moore. Network telescopes overview: what is a network telescope? [Online]. Available: http://www.caida.org/outreach/presentations/2003, 2003. 23

[87] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The spread of the Sapphire/Slammer worm. *IEEE Security & Privacy*, 2003. 23

[88] D. Moore, C. Shannon, and J. Brown. Code-Red: a case study on the spread and victims of an Internet worm. In *Proceedings of the*

2nd Internet Measurement Workshop (IMW), November 2002, pages 273–284. ACM Press, 2002. 23

[89] J. Morse and F.C. Mish. Merrian-Webster online dictionary. [Online]. Available: http://www.m-w.com. 18

[90] J. Nazario. Passive system fingerprinting using network client applications. [Online]. Available: http://crimelabs.com, 2000. 12, 25, 26

[91] New Technologies Inc. Computer forensics defined. [Online]. Available: http://www.forensics-intl.com/def4.html, viewed on 10 June, 2004. 20

[92] OpenReach Inc. IPSec vs. SSL: why choose? [Online]. Available: http://openreach.com, 2002. 13

[93] O'Reilly Media Inc. Learning Debian GNU/Linux - glossary. [Online]. Available: http://www.oreilly.com/catalog/debian/chapter/book. 110

[94] C. Perkins. RFC 2003 - IP encapsulation within IP. Technical report, IETF, 1996. 30

[95] R. Perlman. An overview of PKI trust models. *IEEE Network*, November/December, 1999. 42, 43

[96] R. Perlman. How to build an insecure system out of perfectly good cryptography, 2002. 33

[97] R. Perlman. Understanding IKEv2: tutorial, and rationale for decisions. page 13. Technical report, IETF, 2003. 35

[98] R. Perlman and C. Kaufman. Key exchange in IPSec: analysis of IKE. *IEEE Internet Computing*, 4(6), 2000. 33, 37

[99] P. Persiano and I. Visconti. User privacy issues regarding certificates and the TLS protocol. In *Proceedings of the 7th ACM conference on*

Computer and communications security, Athens, Greece, pages 53–62. ACM Press, 2000. 44

[100] M. Pierce. Detailed forensic procedure for laptop computers. Technical report, SANS Institute, 2003. 17

[101] J. Postel. RFC 793 - transmission control protocol. Technical report, IETF, 1981. 7, 27, 75

[102] KAME Project. Racoon manual pages - racoon(8), racoon.conf(5), setkey(8). 9, 57, 87, 116, 123

[103] Netfilter Project. Netfilter/IPtables homepage. [Online]. Available: http://www.netfilter.org. 102

[104] Infonetics Research. Integrated IPSec/MPLS services and SSL-based VPNs fuel solid growth in VPN. Technical report, Infonetics Research, 2003. 109

[105] M. Rogers. The role of criminal profiling in the computer forensics process. *Computers and Security*, 22(4), 2003. 21

[106] N. Rudin and K. Inman. Forensic science timeline. [Online]. Available: http://www.hbo.com/autopsy/swf/timeline, viewed on 8 June 2004. 19, 20

[107] T. Sammes, B. Jenkinson, and A.J. Sammes. *Forensic computing: a practitioner's guide, 1st edition*. Springer Verlag, 2000. 17

[108] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Network support for IP traceback. *IEEE/ACM Transactions on Networking*, 9(3), 2001. 23

[109] M. Schiffman. *Building open source network security tools - components and techniques*. Wiley Publishing, Inc., 2003. 52, 53

[110] B. Schneier. *Applied cryptography, second edition*. John Wiley & Sons, 1996. 11

[111] B. Schneier. *Secrets and lies: digital security in a networked world*. John Wiley & Sons, 2004. 11, 44

[112] B. Schneier and N. Ferguson. *Practical cryptography, 1st edition*. John Wiley & Sons, 2003. 11

[113] Securify Inc. SecurVantage studio. [Online]. Available: http://www.securify.com/products, viewed on 15 June, 2004. 22

[114] H. Seki. Fingerprinting through RPC. BlackHat Windows Security Briefings, Seattle. [Online]. Available: http://www.blackhat.com, 2004. 28

[115] C. Shannon and D. Moore. The spread of the Witty worm. [Online]. Available: http://www.iit.upco.es/palacios/seguridad, viewed on 16 June, 2004, 2004. 23

[116] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, S.T. Kent, and W.T. Strayer. Hash-based IP traceback. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–14, San Diego, California, U.S., 2001. 23

[117] D.X. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proceedings of IEEE INFOCOM'2001 (20th)*, volume 2, pages 878–886, 2001. 23

[118] L. Spitzner. Passive fingerprinting. [Online]. Available: http://www.securityfocus.com/infocus/1224, viewed on 16 June, 2004, 2000. 26

[119] J. Steinberg. SSL VPN security: secure remote access from any web browser. Technical report, Whale Communications, 2003. 13

[120] W.R. Stevens. *TCP/IP Illustrated, volume 1 - the protocols*. Addison-Wesley Professional; 1st edition, 1994. 7, 23, 47, 59, 60, 75, 106, 109

[121] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. RFC 2661 - layer two tunneling protocol L2TP. Technical report, IETF, 1999. 30

[122] M. Trojnara. Stunnel - universal SSL wrapper. [Online]. Available: http://www.stunnel.org, 24 June 2004. 30

[123] F. Veysset, O. Courtay, and O. Heen. New tool for remote operating system fingerprinting. Technical report, Intranode Software Technologies, 2002. 12, 27

[124] VMware Inc - an EMC Company. VMware is virtual infrastructure. [Online]. Available: http://www.vmware.com. 46

[125] D. Wagner and B. Schneier. Analysis of the SSL 3.0 protocol. In *The Second USENIX Workshop on Electronic Commerce Proceedings, USENIX Press, November 1996*, pages 29–40. USENIX, 1996. 38, 39

[126] Webster's online dictionary. Definition of Nonce by Webster's Online Dictionary. [Online]. Available: http://www.webster-dictionary.org. 69

[127] F. Yarochkin. Remote OS detection via TCP/IP stack fingerprinting. [Online]. Available http://insecure.org, 1999. 12, 14, 24, 25, 26

[128] A. Yasinac, R.F. Erbacher, D.G. Marks, M.M. Pollitt, and P.M. Sommer. Computer forensics education. *IEEE Security & Privacy*, July/August, 2003. 17, 20, 21

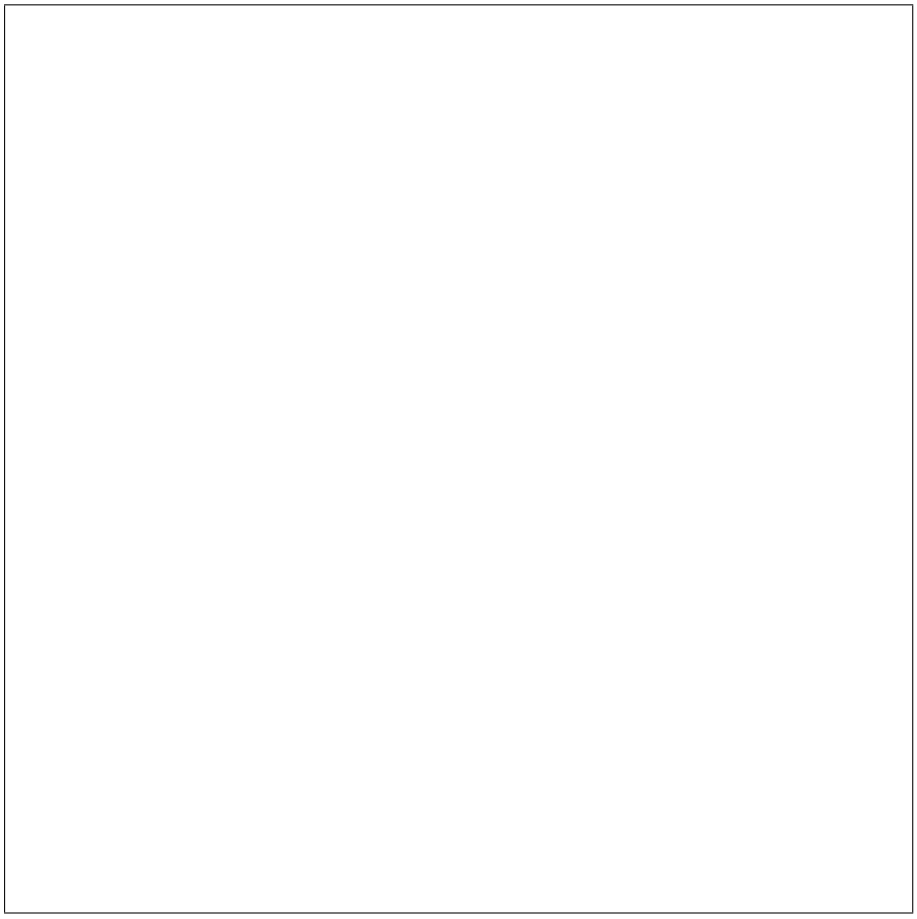
[129] J. Yonan. Private email correspondence dated September 20, 2004. 90

[130] M. Zalewski. Dr. Jekyll had something to Hyde. [Online]. Avaliable: http://lcamtuf.coredump.cx/p0f, viewed on 17 June 2004, 2000. 26

Corroborating Material CD-ROM.

Contents:

- `Vafa_Izadinia_MSc.pdf`
- `lx-gw1_isakmpd1.cap, lx-gw1_isakmpd2.cap, lx-gw1_isakmpd3.cap`
- `lx-gw1_racoon1.cap, lx-gw1_racoon2.cap, lx-gw1_racoon3.cap`
- `ethereal-0.10.7.tar.gz, libpcap-0.8.3.tar.gz`
- `ethereal-setup-0.10.7.exe, winpcap_3_0.exe`
- `isakmpd_20041012.orig.tar.gz`
- `lx-gw-1.tar, lx-gw-2.tar`
- `ipsec-tools-0.4.tar.gz`
- `openssl-0.9.7e.tar.gz`



Attached CD.