FOILING AN ATTACK
DEFEATING IPSEC TUNNEL FINGERPRINTING

Vafa D. Izadinia, D. G. Kourie, J.H.P. Eloff

ICSA Research Group
Computer Science Department
University of Pretoria
{vafa,dkourie,eloff}@cs.up.ac.za

ABSTRACT
This paper addresses some of the discriminants that make IPSec tunnel fingerprinting possible. Fingerprinting of VPN-tunnel endpoints may be desirable for forensic purposes, but in the hands of individuals of ill-intent, it undermines an enterprise network's perimeter security. Three ways of preventing the ill-use of this type of fingerprinting are presented. The first two, apply to enterprises wishing to make their VPN tunnels immune to fingerprinting. The third delves deeper into the conceptual, and is directed at the standards definition process, as used by the Internet Engineering Task Force (IETF) and to authors of security-related RFCs in particular. It addresses aspects in the Internet Key Exchange version 1 (IKEv1) RFC that have led to misinterpretations on the part of IPSec implementers, and describes the use of a form of process algebra known as Communicating Sequential Processes (CSP) in defining security-related standards to overcome RFC-related ambiguities.

KEY WORDS
Fingerprinting, IPSec tunnel, network forensics, CSP, obfuscation, IKE, RFC.

INTRODUCTION

The IPSec standard, and the Internet Key Exchange version 1 (IKEv1) in particular, has been the subject of several analyses [1, 2, 3], and has endured a fair amount of disapproval due to its complexity. Recent research has further shown that the IKEv1 (hereafter 'IKE') Phase 1 and Phase 2 tunnel-setup leaks characteristic information which can be used by an attacker to identify, or compose a 'fingerprint', of the systems establishing the tunnel [4]. In addition to the information leaked during tunnel-establishment, [4] showed that studying the ESP-protected (encrypted) traffic also reveals ICMP ECHO REQUEST and ECHO REPLY messages, and—most interestingly—the messages involved in a TCP Open and a TCP Close. Section 1 provides further background on fingerprinting in general, and VPN-tunnel fingerprinting in particular.

These findings bring to the fore a series of security-related questions: What happens when technology thought to be secure is found to reveal enough information for an attacker to strike? Is the solution to various passive and active fingerprinting techniques the all too familiar approach of strengthening the perimeter defence? When can a step taken to improve the security of a network, be deemed provably adequate?

Of immediate importance is the task of strengthening the perimeter defence, in order to thwart an attack that would make use of VPN-tunnel fingerprinting: this will be dealt with in Section 2. Section 3 proposes a method of countering this type of fingerprinting that does not follow the common pattern of 'reactionary defence'. The method proposed in section 3 can also be extended to prevent other types of Operating System (OS) fingerprinting, such as those described in [5, 6, 7, 8, 9]. Section 4 closes with a digest of the ideas presented in this paper.

1 BACKGROUND

This section begins by explaining Operating System fingerprinting (hereafter 'OS fingerprinting'). It then introduces the recent addition to OS fingerprinting: encrypted tunnel endpoint fingerprinting, as per the findings presented in [4]. The section closes with a potential attack scenario, which describes how individuals of ill-intent would employ this knowledge of fingerprinting to mount an attack on a network.

1.1 OS Fingerprinting

In the arena of network forensics, fingerprinting refers to a method whereby any operating system can be identified based on its responses to targeted probes (*active* fingerprinting) or based on the observation of its network etiquette (*passive* fingerprinting). A machine's OS can be fingerprinted through a variety of means. Initially this was done in a rudimentary fashion, by 'banner-grabbing': that is, making an FTP, HTTP, or telnet connection to the target machine, and 'grabbing' the banner that was displayed (as defined in `/etc/banner` in UNIX variants) [10]. More advanced methods were then developed by Yarochkin [9] and Arkin [5]. *nmap*, a tool developed by Yarochkin runs tests against the remote system's TCP/IP implementation, and determines the OS based on the response to these tests. *Xprobe2*, a tool developed by Arkin, fingerprints by observing ICMP responses from target hosts, and contrasts them to the RFC-defined standard, the fingerprint being the results of the vendor's non-adherence to the RFC.

These forms of fingerprinting can determine, with varying degrees of precision, what operating system the target is running. This knowledge can enable an attacker to identify weaknesses on the target system, and by consulting a database of well-known vulnerabilities, make use of rogue code and gain privileged access to the system, and potentially also to the network.

The abovementioned fingerprinting methods and tools are examples of *active* fingerprinting. Since the method of fingerprinting depends on a series of tests or *probes* targeting the end system, it can be argued that measures can be taken to detect and counter these tests, thus preventing the fingerprinting from taking place. Zalewski's tool *p0f* performs *passive* fingerprinting, that is fingerprinting without generating any network traffic of its own. It determines what OS is running on the target system by observing its behaviour on the network [11].

1.2 Fingerprinting IPSec Tunnels

A growing awareness of the vulnerable nature of unencrypted or *cleartext* network traffic has brought in its wake a widespread use of network protocols that encrypt user data traversing the network[1], such as IPSec, SSL, and SSH. These protocols are able to *tunnel* network traffic: that is, encrypt one protocol (i.e., the encapsulat*ed* protocol) and transport it as payload on another (i.e., the encapsulat*ing* protocol) within the network. This makes the contents of the tunneled protocol unidentifiable to passive observers (sniffers) on the network. The potential problem lies in the fact that oftentimes virtual private network (VPN) tunnels are established between gateways which are placed behind firewalls, thus effectively bypassing a network's perimeter defence.

In earlier work [4], it was shown that although the IPSec protocol encrypts traffic, by observing the Phase 1 and Phase 2 exchanges of the Internet Key Exchange (IKE) protocol (the Key Exchange (KE) protocol used by IPSec), it is possible to identify what OS the two IPSec endpoints (gateways) are running. The study observed IKE exchanges in five OS implementations (namely Windows 2000, Windows 2003, Solaris 9 x86, *isakmpd* on Linux 2.6, and *racoon* on Linux 2.6), and resulted in unique fingerprints for each of these implementations. In addition to this, it was shown that the nature of the encrypted traffic (for example ICMP ECHO REQUEST, ICMP ECHO REPLY, TCP Open, and TCP Close) could be determined by passive observation. In short, this means that IPSec endpoints are also vulnerable to fingerprinting, and more importantly, IPSec gateways

[1]This may be observed by scanning through market research results for SSL and IPSec VPN devices for example [12].

located behind firewalls can become targets of attack, once their underlying OS is known. Two types of fingerprints for each OS were suggested: one based on the Transform-payload attribute value ordering in the ISAKMP Security Association payload of the first message of the Phase 1 Main Mode exchange in IKE, christened the *TAVO* fingerprint, and the other based on a set of discriminants which could be used as input to decision trees, in order to uniquely identify any of the OSs tested.

1.3 Mounting an Attack

It may be argued that in order to make use of this kind of fingerprinting, one must be able to observe the key exchange process, and even possibly be required to force a tunnel re-negotiation (in order to observe the Phase 1 and Phase 2 exchanges). Therefore, in order to add a dimension of practicality to the likelihood of an attack involving VPN-tunnel fingerprinting, route-altering attacks will briefly be covered.

Linder, in a 2001 presentation, detailed attacks against various routing protocols widely in use within Internet Service Provider (ISP) networks today [13]. These attacks rely on the dynamic nature of the Internet: that is, once Enterprise A makes a VPN connection to Enterprise B's network, neither enterprise can predict the path that the tunneled traffic will take to reach its destination. This traffic will be routed along *n* number of routers, and by means of path-altering attacks, a dedicated attacker can insert a router under his control into the service provider network, and convince the peer routers that it is a legitimate *next hop*, thus allowing the attacker to observe all traffic traversing the router.

With this knowledge at hand, consider the following plausible: Corporation A and Corporation B (hereafter 'A' and 'B' respectively) have exercised control over the telecommunications market for several years, in a kind of duopoly. 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 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 earlier, 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 isn't necessarily *secret*, the forensic examiner will likely trace X's steps, and provide this reconstructive evidence in court.

2 TRAFFIC OBFUSCATION

One manner in which to address the potential vulnerability that (this type of) fingerprinting poses, is by obfuscating the network traffic, so that even if an attacker were to fully observe the key exchange between IPSec peers, no uniquely identifiable trait could be discerned. Two types of traffic obfuscation are presented below: the first, traffic *normalisation* obfuscates traffic by bringing its characteristics in line with that of a pre-determined mean (such as an RFC). The second, traffic *dissimulation* obfuscates traffic by randomly changing its characteristics to make it appear to be something that it is not (a different OS, for example). Both these methods assume an extra processing step; be it an extra machine (the logical next-hop), or a process within the sending machine itself, which performs the obfuscation. In Linux, this can be done with the Netfilter (iptables) userspace packet queuing library *libipq*, which allows packets to be passed to userspace for manipulation before being passed back to the kernel.

2.1 Traffic Normalisation

As described earlier, when fingerprinting IPSec tunnel endpoints, the Transform-payload attribute value ordering (TAVO) in the first message of IKE Phase 1 (Main Mode) can serve as a fingerprint, as tests showed that it differed in the IKE/IPSec implementation of each operating system[2]. The TAVO fingerprints for Windows (2003 and 2000), Solaris 9 x86, Linux 2.6 with *isakmpd*, and Linux 2.6 with *racoon*, were `[1, 2, 4, 3, 11, 12]`, `[3, 2, 1, 4, 11, 12]`, `[1, 2, 3, 4, 11, 12]`, and `[11, 12, 1, 3, 2, 4]` respectively.

This information could be used re-construct the contents of the first message of the exchange (and similarly the second message of the exchange, as it is the first message of the responding party) for all platforms, to make them all follow an arbitrary order; numerically descending, for example: `[12, 11, 4, 3, 2, 1]`. This would not cause the message to fail an authentication check since this information is not encrpyted or authenticated yet, and its contents are passed in the clear. In fact, the Transform payload forms part of the Proposal payload, which is exchanged prior to any agreement as to the cryptographic algorithms to be used (for either authentication or encryption). This manipulation would also not cause the IP datagram's checksum to fail, since the checksum is computed over the IP header, and not its payload.

Another candidate for normalisation could be the Vendor-ID payload. The Vendor-ID payload is a "a vendor defined constant. The constant is used by vendors to identify and recognize remote instances of their implementations . . . An implementation is NOT REQUIRED to send any Vendor ID payload at all" [14]. As was observed in the tests conducted among five operating systems, only Windows 2003 and Windows 2000 made use of the Vendor-ID payload[3]. Assuming that the use of the Vendor-ID payload is not desired (because it uniquely identifies the operating system), it could be normalised in several ways. One way is by removing it altogether, and changing the 'Next Payload' value in the Secu-

[2]It should be noted that the TAVO fingerprint for Windows 2003 and Windows 2000 was the same, it could however be argued that it uniquely identify the 'Windows' IKE/IPSec implementation.

[3]Windows 2003 in fact makes use of 4 distinct Vendor-ID payloads, whereas Windows 2000 makes use of 1

rity Association (SA) payload header to ‘None’. It could also be normalised by reducing the number of Vendor-ID payloads to a predetermined number (2, for example), and replacing the original Vendor-ID either with a random or with a pre-determined string. This would be done across all platforms; and in the case that no Vendor-ID payload is included by default, two such payloads would be inserted in the first message of IKE Phase 1 (Main Mode).

2.2 Traffic Dissimulation

It may be the case, that the attacker is particularly dedicated, and when faced with normalised traffic, is willing to exhaust the OS search space by guessing the OS of the IPSec endpoint. This paper introduces the term ‘traffic *dissimulation*’ to mean the ‘camouflaging’ or ‘masking’ of network traffic; a means whereby network traffic is made to appear to be something it is not. Although not recommended as a security measure on its own, traffic *dissimulation* can purchase valuable time, and even result in the attacker’s frustration. By *dissimulation* is meant the masquerading of traffic to make it appear to have a different identity. When studying the IPSec-protected (encrypted) traffic for example, ICMP ECHO REQUESTs and ECHO REPLYs (used by the `ping` command) were observed to consist of 108 bytes across all platforms, and to follow each other in close succession. RFC 792 defines a variable-length ‘Data’ field in the ECHO REQUEST or ECHO REPLY messages [15], and so a method of dissimulating this traffic could be to add random data into the ‘Data’ field of any ECHO REQUEST or ECHO REPLY message (sent by the endpoint in question). Another method of dissimulation could be that of inserting another vendor’s Vendor-ID payload, in order to ‘throw-off’ a potential attacker.

In general, an IPSec endpoint’s identity could be forged and portrayed to be that of another, by applying any IKE/IPSec implementation’s fingerprint discriminants on another. The work conducted in [4] showed for example that out of the five implementations tested, Windows 2003 and Windows 2000 were the only ones whose Phase 2 (Quick Mode) consists of 4 messages, instead of the RFC-defined 3, so an extra Quick Mode message could be added to any other vendor’s implementation, in order to confuse an attacker (and since Quick Mode exchanges are encrypted, the third message could be copied, and resent as the fourth). The same can hold for ISAKMP Informational messages, where some OS’s IKE/IPSec implementations send them between Main Mode and Quick Mode, and others do not, but rather send them at tunnel tear-down. Any attempt to make one OS’s IKE/IPSec implementation behave like another’s would constitute traffic dissimulation.

These methods can be classified as ‘patching and fixing’, that is, repairing a condition caused the misinterpretation of semantic language in the RFC standard.

The following section provides a suggestion on how to prevent the misinterpretation of the standard in the first place.

3 UNAMBIGUOUS PROTOCOL SPECIFICATION

Although the abovementioned discriminants which make the VPN-tunnel fingerprinting possible are not due to RFC ambiguities, there are several instances in which different IKE/IPSec implementers have interpreted the RFCs differently due to unclear wording. This can be attributed to defining a standard in natural language, as opposed to some formal language. In order to limit the many meanings that can be attributed to auxiliary verbs and other key words in RFCs, the Internet Engineering Task Force (IETF) ratified RFC 2119, entitled “Key words for use in RFCs to Indicate Requirement Levels”. RFC 2119 defines ‘MUST’, ‘REQUIRED’ or ‘SHALL’ for example, as meaning that “the definition is an absolute requirement of the specification” [16]. RFC 2409 however lists 49 instances of ‘MUST’ (capitalised) and 15 uses of ‘must’ (not capitalised) [17], while RFC 2119 states that “these words are often capitalized” [16]. As in several other cases, this may create an ambiguity for the implementer.

3.1 Ambiguities in RFC 2409

The IPSec standard is comprised of a number of RFCs, one of which is the IKE RFC (RFC 2409). RFC 2409 has come under fire for its unclear language, and although some of its ambiguities may be attributed to the politics of protocol specification (a topic discussed in [3]), some could be avoided by using a formal language, which has unambiguous semantics.

One of the fingerprint discriminants used in VPN-tunnel fingerprinting, is the Transform Payload Numbering: some implementations start counting at 0 and some at 1. In a case of allowing implementer discretion, but which may lead to uncertainty RFC 2409 states that if there is more than one transform, then “the multiple transforms MUST be presented with monotonically increasing numbers in the initiator’s preference order,” but leaves the choice of initial number to the implementer. Another place where RFC 2409 is particularly ambiguous is with respect to the treatment of x.509 certificates. They can be treated as either the *signature* case, or as a *revised mode of public key encryption*. 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 the third message of Main Mode. (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 a certificate payload to provide the responder with a public key with which to respond." However this definition appears under the description of an authentication method which allows optional x.509 certificates. Thus 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 the third message of Main Mode; 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. Tests conducted on five IKE/IPSec implementations showed that all but one vendor interpreted the first meaning to be correct.

There are other examples of convoluted wording in RFC 2409, for example:

- "While Oakley defines "modes", ISAKMP defines "phases". The relationship between the two is very straightforward and IKE presents different exchanges as modes which operate in one of two phases."

- "Perfect Forward Secrecy (PFS) of both keying material and identities is possible with this protocol. By specifying a Diffie-Hellman group, and passing public values in KE payloads, ISAKMP peers can establish PFS of keys– the identities would be protected by SKEYID_e from the ISAKMP SA and would therefore not be protected by PFS. If PFS of both keying material and identities is desired, an ISAKMP peer MUST establish only one non-ISAKMP security association (e.g. IPsec Security Association) per ISAKMP SA. PFS for keys and identities is accomplished by deleting the ISAKMP SA (and optionally issuing a DELETE message) upon establishment of the single non-ISAKMP SA. In this way a phase one negotiation is uniquely tied to a single phase two negotiation, and the ISAKMP SA established during phase one negotiation is never used again."

3.2 Communicating Sequential Processes

CSP is one example of a class of formal specification languages, generally known as process algebras. It allows for the unambiguous specification of communicating processes such as protocols. This application of CSP has been addressed in [18] and taken a step further in [19] where it has been used for protocol analysis. IKE has also been formally analysed in [20] using a formal methods tool called NRL, which noted "Thus our formal analysis of IKE was mainly useful in establishing that the conflicting requirements were handled correctly, and in pointing out ambiguities in the specification that could lead to the requirements being violated" and further "Problems involving conflicting requirements have often been a fruitful area of application for formal methods; we should not be surprised that the same turned out to be the case here.”

In order to showcase the benefit of defining a standard using CSP, IKE Phase 1 Main Mode will be illustrated below, in CSP notation. It should be noted that this is an illustration, and not a definitive specification of IKE Phase 1 Main Mode. The reason for this, as explained further in the text, is that due to the RFCs being mute on some points, it would appear that it is not possible to derive a definitive specification from them.

The CSP notation employed below, makes use of processes, channels, and events. Line 1, for example defines IKE to be composed of processes Init and Resp (for Initiator and Responder). These processes are described as being parallel processes, because although in practice they depend on each other’s input to change state, they synchronise on events, such as ‘p(hdr1(main), sa(pp_i))’ sent in the output channel ‘ike_i’. Communication is modeled as the simplex channels ‘ike_i’ and ‘ike_r’ that carry atomic events, such as ‘(HDR_i, KE_i, N_i)’. Process Init is composed of sub-processes MainMode and AggrMode. The state that is entered after each of the six IKE Phase 1 Main Mode messages are sent, have been denoted as states Msg1Sent through Msg6Sent. The ‘!’ and ‘?’ operators denote output and input on a channel respectively. The ‘$\Box$’ operator denotes a choice: such as in Line 2, Phase 1 can either enter Main Mode or Aggressive Mode. The ‘$\rightarrow$’ operator denotes process flow: ‘a $\rightarrow$ b’ means ‘a then b’. The contents carried by each event, except those on lines 3-5 are defined in RFC 2409. In lines 3-5, ‘p(hdr1(main), sa(pp_i))’ denotes a packet with the ISAKMP Main Mode header, and the SA proposal payload as its content. Any event or content thereof, subscripted by ‘i’, means it pertains to the Initiator; those denoted by ‘r’ pertain to the Responder.

Lines 9-12, for example would be interpreted as: In the Msg3Sent state, the Init process waits for input from the Responder on the ike_r channel. If this input is of the form ‘(HDR_r, KE_r, N_r)’ then ‘(HDR*, IDii, [Cert], SIG_i)’ is sent as an atomic event by the Initiator through channel ike_i, leading to Init entering the Msg5Sent state. Otherwise, a Notify message is sent and the sate returned to Init. Alternatively, if no response is received from the Responder, a timeout is reached, a reject message issued on the ike_i channel, and state returned to Init (the Initiator’s commencing state).

Algorithm 1 IKE Phase 1 Main Mode modeled in CSP.

```
[1] IKE = Init || Resp.
[2]       Init = Init(MainMode) □ Init(AggrMode).
[3]       Init(MainMode) ike_i!p(hdr1(main), sa(pp_i)) → Init(Msg1Sent).
[4]       Init(Msg1Sent) = ike_r?x → if (x = notify) then Init
[5]           else if (x = sa(pp_r))
[6]           then ike_i!(HDR_i, KE_i, N_i) → Init(Msg3Sent)
[7]           else Init
[8]           | timeout → ike_i!reject → Init.
[9]       Init(Msg3Sent) = ike_r?x → if (x = (HDR, KE_r, N_r))
[10]          then → ike_i!(HDR*, IDii, [Cert], SIG_i) → Init(Msg5Sent)
[11]          else notify → Init
[12]          | timeout → ike_i!reject → Init.
[13]      Init(Msg5Sent) = ike_r?x → if (x = (HDR*, IDir, [Cert], SIG_r))
[14]          then → Init(Phase1Complete)
[15]          else notify → Init
[16]          | timeout → ike_i!reject → Init.
[17]      Resp = ike_i?p(x,y) → if (x = hdr(main)) then Resp(MainMode,y)
[18]          else if (x = hdr(aggr)) then Resp(AggrMode)
[19]          else ike_r!reject → Resp
[20]          | timeout → ike_r!reject → Resp.
[21]      Resp(MainMode,y) = if (y = sa(pp_i))
[22]          then (ike_r!notify → Resp | ike_r!sa(pp_r) → Resp(Msg2Sent))
[23]          else ike_r!reject → Resp
[24]          | timeout → ike_r!reject → Resp.
[25]      Resp(Msg2Sent) = ike_i?x → if (x = (HDR, KE_i, N_i))
[26]          then ike_r!(HDR, KE_r, N_r) → Resp(Msg4Sent)
[27]          else notify → Resp
[28]          | timeout → ike_r!reject → Resp.
[29]      Resp(Msg4Sent) = ike_i?x → if (x = (HDR*, IDii, [Cert], SIG_i))
[30]          then ike_r!(HDR*, IDir, [Cert], SIG_r) → Resp(Phase1Complete)
[31]          else notify → Resp
[32]          | timeout → ike_r!reject → Resp.
```

It is interesting to note that the exercise of modeling IKE Phase 1 Main Mode in CSP, prompted questions not directly answered in RFC 2409. The behaviour denoted in lines 8, 12, 16, 20, 24, 28, and 32, namely that an implementation should timeout and return to its initial state was conjectured for example. In the real world, this would be a condition faced by an implementation operating in a high packet-loss environment. Another issue raised by the CSP model of IKE Phase 1 Main Mode, and not directly addressed by RFC 2409, is that of the atomicity of IKE messages. It is tacitly assumed that all the message components will be correct—and indeed this is how it was modeled from line 6 onwards. However, the type of Notification message that could be sent in this case, such as INVALID-PAYLOAD-TYPE(1) or PAYLOAD-MALFORMED(16) are only defined in RFC2408 as being used for this purpose during SA establishment, not later in the exchange.

By making use of a process algebra such as CSP in RFCs, the IETF could reduce the number of RFC-related ambiguities to an acceptable minimum. This would in turn make it easier for the implementers not to misinterpret the RFC wording, and in turn, create implementations that do not digress wildly from the RFC, thus defeating fingerprinting at an elementary level. This same approach could be used to defeat active fingerprinting as performed by *nmap* and *Xprobe2*, as well as passive fingerprinting as done by *p0f*.

Although getting used to the CSP syntax and semantics may present a slight learning curve, it is arguably simple. Finite State Processes (FSP) for example, a CSP variant, is presented at an undergraduate level by Magee and Kramer [21]. It may be argued that it is something of a challenge to express the requisite sequence (or trace) of events in terms of CSP-defined processes, but it is precisely by being forced to answer the questions that naturally arise when trying to produce such a CSP spec, that ambiguities are uncovered—i.e. the very process of attempting to produce a spec is fruitful.

4 CONCLUSION

As has been shown, in the hands of individuals or corporations of ill-intent, remote-host fingerprinting can be a valuable reconnaissance tool. Being able to defend against active and passive fingerprinting can be advantageous to potential targets. With encrypted communication finding more widespread use, it is only natural that attackers will attempt to penetrate what are believed to be ‘cryptographically fortified’ systems and communications. The research in [4] showed that out of five IKE/IPSec implementations involved in the study, the operating systems at either end of an IPSec tunnel could be identified, based on their IKE Phase 1 and Phase 2 exchanges, and that the nature of some of the encrypted network traffic could be identified, simply by its behaviour.

This paper presented three methods of defeating this type fingerprinting: fingerprint obfuscation by traffic normalisation, by traffic dissimulation, but most importantly, defining protocols and RFCs in a process algebra such as CSP, resulting in non-ambiguous specifications, and therefore less fingerprintable implementations. The rationale employed can also be applied to defeating other fingerprinting methods.

CSP was chosen because of its clean structure. Future research could contrast the adequacy of CSP as opposed to other forms of process algebra. In order to further stress the need for defining standards using formal specification languages, an in-depth study of various standards-defined protocols is necessary, to identify those which have most suffered from security vulnerabilities. The Point to Point Tunneling Protocol (PPTP) could be one; the Layer 2 Tunneling Protocol (L2TP) another.

References

[1] R. Perlman and C. Kaufman. Key Exchange in IPSec: Analysis of IKE. *IEEE Internet Computing*, 4(6), 2000.

[2] R. Perlman. How to Build an Insecure System out of Perfectly Good Cryptography, 2002.

[3] N. Ferguson and B. Schneier. A Cryptographic Evaluation of IPSec. Technical report, Counterpane Internet Security, Inc., 1999.

[4] Vafa D. Izadinia. Fingerprinting Encrypted Tunnel Endpoints. MSc. Dissertation, Computer Science Department, University of Pretoria, South Africa, 2004.

[5] O. Arkin. ICMP Usage in Scanning: The Complete Know-How, version 3. 2001.

[6] O. Arkin and F. Yarochkin. ICMP Based Remote OS TCP/IP Stack Fingerprinting Techniques. *Phrack Magazine*, 11(Issue 57, File 7 of 12), 2001.

[7] J. Bordet. Remote SMTP Server Detection. 2002.

[8] R. Hills. NTA Monitor UDP Backoff Pattern Fingerprinting White Paper. Technical report, NTA, 2003.

[9] F. Yarochkin. Remote OS Detection via TCP/IP Stack Fingerprinting. *[Online]. Available: http://insecure.org*, 1999.

[10] f0bic. Examining Remote OS Detection using LPD Querying. 2001.

[11] M. Zalewski. Dr. Jekyll had Something to Hyde. 2000.

[12] J. Girard. Magic Quadrant for SSL VPNs, 1H04. Technical report, Gartner Research, 2004.

[13] F. Linder. Routing & Tunneling Protocol Attacks, 2001. Presented at BlackHat briefings, November 2001, Amsterdam.

[14] D. Maughan, M. Schertler, M. Schneider, and J. Turner. RFC 2408 - Internet Security Association and Key Management Protocol (ISAKMP). Technical report, IETF, 1998.

[15] J. Postel. RFC 792 - Internet Control Message Protocol. Technical report, IETF, 1981.

[16] S. Bradner. RFC 2119 - Key Words for use in RFCs to Indicate Requirement Levels. Technical report, IETF, 1997.

[17] D. Harkins and D. Carrel. RFC 2409 - The Internet Key Exchange (IKE). Technical report, IETF, 1998.

[18] S. Schneider. Security properties and csp. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, page 174. IEEE Computer Society, 1996.

[19] S. Schneider. Using CSP for protocol analysis: the Needham-Schroeder Public-Key Protocol. Technical report, Royal Holloway, University of London, 1996.

[20] C. Meadows. Analysis of the internet key exchange protocol using the nrl protocol analyzer. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 216–231. IEEE Computer Society, 1999.

[21] Jeff Magee and Jeff Kramer. *Concurrency: State Models & Java Programs*. John Wiley & Sons, 1999.