Experimenting with an Intrusion Detection System for Encrypted Networks

Vik Tor Goh*

Faculty of Science and Technology,
Information Security Institute,
Queensland University of Technology,
GPO Box 2434, Brisbane QLD 4001, Australia
E-mail: v.goh@qut.edu.au
*Corresponding author

Jacob Zimmermann

Information Security Institute,
Queensland University of Technology,
MS-714, Level 7, 126 Margaret St,
Brisbane QLD 4001, Australia
E-mail: j.zimmerm@isi.qut.edu.au

Mark Looi

Faculty of Science and Technology,
Information Security Institute,
Queensland University of Technology,
GPO Box 2434, Brisbane QLD 4001, Australia
E-mail: m.looi@qut.edu.au

Abstract: Network-based Intrusion Detection Systems (NIDSs) analyse network traffic to detect instances of malicious activity. Typically, this is only possible when the network traffic is accessible for analysis. With the growing use of Virtual Private Networks (VPNs) that encrypt network traffic, the NIDS can no longer access this crucial audit data. In this paper, we present an implementation and evaluation of our approach proposed in Goh et al. (2009). It is based on Shamir's secret-sharing scheme and allows a NIDS to function normally in a VPN without any modifications and without compromising the confidentiality afforded by the VPN.

Keywords: NIDS; network-based intrusion detection system; encrypted networks; VPN; virtual private network; IPsec; IP Security; Shamir's secret-sharing; SNORT.

Reference to this paper should be made as follows: Goh, V.T., Zimmermann, J. and Looi, M. (2010) 'Experimenting with an Intrusion Detection System for Encrypted Networks', *Int. J. Business Intelligence and Data Mining*, Vol. 5, No. 2, pp.172–191.

Biographical notes: Vik Tor Goh is a PhD Candidate attached to the Information Security Institute, Queensland University of Technology in Australia. He received his BEng (Hons.) Electronics (2002) and MEngSc (2006) Degrees from Multimedia University, Malaysia. His research interests include intrusion detection, digital watermarking and coding theory. He is also certified as a CISSP and CCNA-CCAI.

Jacob Zimmermann has been a Full-Time Researcher at the Queensland University of Technology in the areas of intrusion detection, vulnerability analysis and network traffic analysis since 2005. He was awarded a PhD in Computer Science from the University of Rennes 1, France in 2003, for his research on information flow-based intrusion detection. His current research interests include analysing vulnerabilities in web services-based systems and intrusion detection in encrypted networks.

Mark Looi is a Professor and the Head of the School of Information Technology at the Queensland University of Technology. He graduated with an Electronics Engineering degree and a Computer Science degree in 1990, and received his PhD in 1994. Over the years, he has worked variously on areas involving network and information security, smart cards, electronic transaction security, mobile networks, wireless networks and both indoor and outdoor positioning technology. He has successfully supervised over 20 PhD students to date.

1 Introduction

Intrusion Detection Systems (IDSs) have traditionally been used to detect security policy violations or other attacks, either at a host or network level. Accordingly, a *host-based* IDS (HIDS) operates within a host, monitoring access logs and system calls for signs of potential abuse. On the other hand, a *network-based* IDS (NIDS) typically operates at a higher level of granularity, analysing network traffic for instances that are considered or assumed malicious.

If the audit data cannot be accessed due to corruption or encryption, the IDS will not function properly. Our work focuses on the effects of end-to-end encrypted networks on NIDS operations. We are motivated by the growing use of encrypted networks that obfuscate all network traffic between any two hosts in the network. Commonly used network encryption protocols like Secure Sockets Layer (SSL) and various other Virtual Private Network (VPN) protocols can mask malicious traffic and evade detection by perimeter security such as firewalls and NIDS. In these conditions, a NIDS as such cannot function because network traffic is encrypted and therefore cannot be analysed.

In this paper, we present the results of our work towards enabling a NIDS to function normally and detect attacks propagated through encrypted networks. To secure the detection loophole introduced by encrypted networks, network administrators usually use a specific network setup. In this setup, the encrypted network from an external host can be terminated at a network gateway that has a NIDS co-located on it. Once decrypted by the gateway, the traffic is analysed

by the NIDS before being sent as cleartext to the internal host. Wagner et al. (2006) proposed such an approach for VPNs. However, this design is not suitable when the encrypted network is established end-to-end, i.e., directly from an external host to the internal host. Furthermore, this design cannot be used when internal hosts establish VPN connections between themselves. Although not prevalent now, we forsee the growth of such peer-to-peer encrypted networks in the future especially with the adoption of IPv6 that has built-in encryption capabilities.

Our work thus focuses on developing a detection framework that integrates a NIDS into an end-to-end encrypted network. Using our framework, a NIDS can function within the encrypted infrastructure to detect intrusions. It does this without compromising the network's confidentiality or integrity. For most NIDSs, such integration has never been directly addressed. There has always been an implicit assumption that unencrypted network traffic is available and it has been up to the network administrator to ensure this.

The paper is organised as follows. Section 2 presents relevant previous works while in Section 3, we give an overview of the solution outlined in Goh et al. (2009). We also discuss some of the implementation details in this section. Section 4 presents some performance evaluation of the current software prototype and the paper concludes in Section 5.

2 Related works

Most NIDSs passively sniff the network for malicious traffic, but this is not possible with encrypted networks. Although modifying the network architecture to accommodate the NIDS does seem to address the problem (Wagner et al., 2006), this solution is still limited. A NIDS co-located on a gateway is vulnerable and susceptible to attacks if the gateway in which it resides is itself compromised.

Alternatively, active sniffing techniques can be used. A man-in-the-middle approach is a technique where the sniffer makes independent connections with two communicating peers and relays network packets between them, with each peer believing that it is communicating directly with its counterpart. Yamada et al. (2007) noted that with the NIDS-in-the-middle, all encrypted network traffic can be decrypted with the private keys of both peers. This design involves the use of a complicated Public Key Infrastructure (PKI) which implies heavy key management overhead. Furthermore, the NIDS-in-the-middle is unusable with some key exchange protocols such as Diffie-Hellman (Diffie and Hellman, 1976).

To the best of our knowledge, three other approaches that attempt to address the problem of detecting intrusions or attacks over encrypted networks have been proposed.

The first approach detects misuse of network encryption protocols such as SSL, IP Security (IPsec), or Secure Shell (SSH). A misuse in this context is defined as a case where the encryption protocols do not transition from one state to another in an expected and predictable manner. Whenever such transition anomaly is detected, the detection system assumes that the encryption protocol is being misused, i.e., an attacker is attempting to exploit it.

To detect attack against such protocols, the detection systems by Md. Fadlullah et al. (2007), Yasinsac and Childs (2001), and Joglekar and Tate (2004) require

an accurate specification of a properly working network encryption protocol. Any deviations from this specification are considered as attacks. For instance, Joglekar and Tate (2004) defined specifications for the SSL protocol. When there is a large number of failed SSL session negotiation attempts, the IDS raises an alert. However, the task of defining the specification is not trivial. Every possible and legitimate state must be modelled. Moreover, application-level attacks (SQL Injection, Buffer Overflow, Cross-Site Request Forgery) function by delivering malicious payload. As such, they are not detected by this technique, because the payload is not decrypted and therefore not analysed.

The second technique uses statistical traffic analysis to infer information from frequently observed patterns in the communication process. For example, patterns derived from observation of frequency and size of network packets can reveal certain trends. Yamada et al. (2007), Piccitto et al. (2007) and Foroushani et al. (2008) used this technique to monitor patterns such as packet size and inter-arrival time of SSL and VPN traffic to identify malicious traffic. Statistical analysis is nevertheless limited in scope due to the few traffic patterns that can practically be deduced purely from observing the network. Furthermore, a large volume of network traffic has to be analysed before any obvious trends begin to emerge.

The two different approaches described are thus suitable for use as intrusion detection systems in encrypted network because they do not directly analyse the payload contained within the encrypted packet. Instead, these approaches extract useful information by monitoring outwardly observable patterns and behaviours. Nonetheless, application-level attacks are not detected because both these detection methods do not analyse the payload of encrypted packets.

Although a HIDS could have been used to detect attacks over encrypted networks, it has the added overhead of being intrinsically complex. A HIDS has to monitor many distinct aspects of the host such as system calls and file statuses. Besides, a traditional HIDS only has a local view of potentially malicious activity and does not correlate events between different hosts that may be indicative of the severity of the attack as a whole at the network level.

To address this problem Abimbola et al. (2006) developed a framework that retrieves decrypted network traffic for deep packet inspection. They suggested that IDS sensors be installed on every network hosts where decrypted network traffic can be fed directly into a NIDS. Although their framework can retrieve decrypted network traffic for analysis, it does not address the fact that sensors residing in the targeted hosts can be defeated if the host is compromised.

Due to the limited scope of the traffic analysis approach and complexities of specification-based IDS, we thus propose a detection framework based on deep packet inspection techniques. The framework directly integrates a NIDS into an encrypted network, allowing it to function as a passive sniffer for deep packet inspection. We achieve this without the need for complicated key management systems.

A key feature of the proposed framework is that it does not compromise on confidentiality. Private network traffic will remain confidential and protected from unauthorised network sniffers by the underlying VPN encryption protocol. Besides the sender and receiver, the only other device with access to the network traffic is the NIDS, hosted on a trusted device. In a high security environment, if an organisation's security policy dictates that all traffic be analysed for potential

threats, the use of our framework to analyse encrypted network traffic by a NIDS is ultimately not different from using HIDS or IDS sensors residing within hosts themselves.

In this paper, we present the design of our framework, the implementation of a prototype and its performance evaluation. As far as we are aware, this work is the first research that attempts to integrate a NIDS together with an encrypted network in such a way that deep packet inspection can be carried out without compromising traffic confidentiality.

3 Overview of detection framework

3.1 General approach

At the core of the detection framework is a protocol that replicates network traffic from a sender and makes that replicated traffic available to the NIDS as well as to the receiver. This protocol uses the underlying VPN encrypted network to guarantee that network traffic's confidentiality remains protected by encryption against third-party packet sniffer. It does this by adding another layer of protection above the VPN protocols in the network layer of the OSI network model. Also, the VPN network do not need to be modified in any way to accommodate our framework.

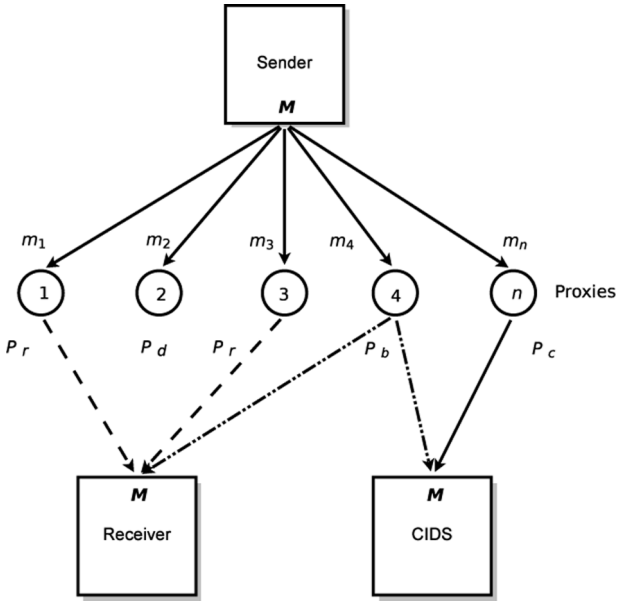
In our work we propose that a Central IDS (CIDS) be used together with forwarding proxies as in Figure 1. As shown, the CIDS operates from a separate and dedicated host in the network. If a sender wishes to send network packets to a receiver, it will do so via the forwarding proxies. This packet routing is achieved using a custom network interface driver installed in the sender. The forwarding proxies in turn ensure that the network packets are forwarded to the receiver and CIDS. Our approach can thus be summarised by the following principle:

All traffic sent to a receiver by a sender must be replicated and forwarded also to the CIDS, without the possibility of the sender withholding traffic from the CIDS or forging fake traffic, and while maintaining the confidentiality and integrity of the encrypted network.

Since the proposed approach works on top of the VPN encrypted network, all outgoing network traffic will be encrypted. As a result, every connection between network devices is encrypted, thus guaranteeing traffic confidentiality with respect to unauthorised third-parties or network sniffers is ensured. For instance, in Figure 1, the sender-to-proxies, proxies-to-receiver and proxies-to-CIDS connections are all encrypted by a VPN protocol. This peer-to-peer encryption can be achieved using Opportunistic Encryption in IPsec (Richardson and Redelmeier, 2005).

Although network layer encryption prevents unauthorised network sniffing, proxies are still able to access network packets relayed through them. This would expose the network packets to unwanted scrutiny and possible tampering while transiting through a proxy. To ensure confidentiality with respect to the proxies, we use secret-sharing. Each original packet is split into shares according to Shamir's scheme (Shamir, 1979), and it is the shares that are relayed through the proxies.

Figure 1 Proposed transmission scheme ($k = 2$)



This design thus ensures the packets' secrecy with respect to the proxies while avoiding the need for a PKI. The underlying encryption mechanism afforded by the VPN infrastructure ensures the packets' secrecy with respect to a third-party packet sniffer.

3.2 Protocol description

A (k, n) secret-sharing scheme splits a secret S into n shares, where and $k < n$ and $S = \{s_1, s_i, \ldots, s_n\}$. Knowledge of any k or more s_i recovers S while knowledge of any $k - 1$ or less reveals nothing. If some of the shares are incorrect, the original secret cannot be recovered; that is $\{s'_1, s'_2, s_i, \ldots, s_n\} \neq S$. Confidentiality is thus an inherent feature of a secret-sharing scheme since each proxy receives no more than one share of any given network packet, the relayed network traffic is essentially masked from the proxies.

These different components ensure that the sender, receiver and CIDS are the only parties in the network who can access a readable copy of the network traffic. The general algorithm implementing our protocol is shown in Table 1.

The replicated packets recovered at the CIDS and receiver are identical to the packet that was originally sent by the sender. This assured of because of the underlying VPN network. VPN protocols such as IPsec provides data integrity verification and host authentication (Kent and Atkinson, 1998a, 1998b). Both these features ensure that no other parties except the authorised ones are allowed to receive and modify data. Shares that have been tampered with either intentionally or unintentionally while en route to the receiver can be detected by IPsec and subsequently dropped.

The algorithm works at the network layer. As with most network layer protocols, it only provides best-effort packet delivery. If reliable packet delivery is required,

a suitable transport layer protocol such as TCP can be used. Besides the nominal packet loss rates, our algorithm actually induces additional packet loss due to the delivery uncertainties associated with the probabilities P_r, P_c, P_b and P_d. Despite that, because of the inherent redundancy of the protocol (an additional $n - k$ shares are available), the induced packet loss rates can still be kept to a minimum level. We discuss this further in Section 3.3.2.

Table 1 Algorithm for sending and receiving network packets using a secret-sharing scheme

Algorithm
1. Sender splits packet M into corresponding shares of $\{m_1, m_i, \ldots, m_n\}$ using a (k, n) secret-sharing scheme;
2. Sender selects n proxies from a pool of N proxies where $n < N$;
3. Each m_i is sent to one of the n forwarding proxies, p_i;
4. Proxy p_i does one of the four predefined actions;
• Forward to receiver only with probability P_r;
• Forward to CIDS only with probability P_c;
• Forward to both CIDS and receiver with probability P_b; or
• Drop the share with probability P_d.
5. If the receiver receives k or more m_i's, it recovers the packet M; and
6. If the CIDS receives k or more m_i's, it recovers packet M for analysis.

An attacker could evade detection if a malicious packet is sent to the receiver while another forged but harmless packet is sent to the CIDS, or if the CIDS receives less than k shares of a malicious packet. However, if the proxies comply with the algorithm and their actions are a-priori unpredictable, the attacker will not know beforehand which of the n proxies will forward to whom (CIDS, receiver or both). It is therefore difficult for the attacker to reliably determine which proxies should receive shares of the malicious packet and which should be omitted or receive shares of a forged packet. It is more probable that the shares will arrive mixed up, resulting in a corrupted packet. This is true if we assume that all the proxies are uncompromised and not collaborating with the attacker.

Consequently, we envision two types of attacks that can be detected by our proposed approach. They are as follows:

- *Application-level attack*. Our approach is essentially a tunnelling protocol. Network packets, malicious or otherwise, that are received will always match the original content. Therefore, attacks sent through the tunnel can be detected by a standard NIDS like SNORT (Roesch, 1999) or Bro. In any case, these attacks pose a traditional intrusion detection problem.
- *Evasion attack*. These attacks attempt to evade CIDS detection in our context, by trying to forge false packets or withhold traffic from the CIDS. This can be detected if a packet is corrupted upon recovery.

Although the probability of detecting these attacks is high as we prove later in Section 3.3.1, an attacker could possibly bypass the proxies and send directly to the receiver. If we assume that the network is initially 'clean', it is reasonable for a receiver to expect for packets from a sender via the proxies. Consequently, if the receiver receives data directly from the sender, it will ignore the non-compliant traffic. Source address spoofing is also not possible because of the peer authentication mechanism of the underlying VPN. A malicious sender will always be motivated to comply with the protocol if it wants its packets to be properly delivered.

From the algorithm, we know that n forwarding proxies are used whenever a packet is sent. These n proxies are randomly chosen by the sender. Doing so prevents malicious proxies from disrupting the communication process between honest senders and receivers. If a corrupted and misbehaving proxy is selected to forward a packet, it may not be selected again by an honest sender to forward another packet, provided that a sufficiently large N/n ratio. As a result, it is difficult for that malicious proxy to continuously disrupt the communicating peers.

3.3 Analysis

The mathematical model and results that were presented in Goh et al. (2009) did not fully account for the effects of the practical implementations. Our initial calculations assumed that the CIDS and receiver will wait for all n shares to arrive before packet recovery is carried out. In our implementation, packet recovery is immediately executed when the first k shares arrive. This minimises processing delay and reduces packet latency.

Our analysis considers the presence of a sophisticated attacker capable of creating and sending two types of packets to evade detection. They are the malicious packet meant only for the receiver and the forged but harmless packet meant only for the CIDS.

For the sake of clarity, we let M be the malicious packet while M' be the forged but harmless packet. Each of them is represented as $M = \{m_1, m_i, \ldots, m_\alpha\}$ and $M' = \{m'_1, m'_j, \ldots, m'_\beta\}$ where $\alpha + \beta \leq n$ and $\alpha, \beta \geq k$. For a (k, n) secret-sharing scheme used in the protocol, n is the number of forwarding proxies involved in relaying the shares of a network packet and k is the minimum number of shares that must be received for successful packet recovery.

According to the algorithm presented in Section 3.2, a proxy will randomly choose where to forward the share that it receives. Specifically, it does one of the following actions with fixed probabilities:

- Forward to receiver only with probability P_r.
- Forward to CIDS only with probability P_c.
- Forward to both CIDS and receiver with probability P_b.
- Drop the share with probability P_d.

These probabilities can be represented more concisely as marginal probabilities. The marginal probability of a proxy forwarding anything at all to the receiver is P_Γ.

We also let the marginal probability of forwarding anything at all to the CIDS be P_Σ. They are shown as equations (1) and (2) respectively.

$$P_\Gamma = P_r + P_b \tag{1}$$

$$P_\Sigma = P_c + P_b. \tag{2}$$

3.3.1 Probability of evasion attacks

We can now determine the probability of an evasion attack (or false negative) by first analysing the outcomes occurring at the receiver. The probability of the receiver receiving a out of α malicious shares, where $0 \leq a \leq \alpha$ is,

$$P_{\text{recv}}(a) = {}^{\alpha}C_a \, (P_\Gamma)^a \, (P_\Gamma')^{\alpha - a} \tag{3}$$

where ${P_\Gamma}' = 1 - P_\Gamma$. The notation ${}^{\sigma}C_{\varphi}$ is the binomial coefficient $\binom{\sigma}{\varphi}$. Likewise, the probability of the receiver receiving b out of β harmless but forged shares, where $0 \leq b \leq \beta$ is,

$$P_{\text{recv}}(b) = {}^{\beta}C_b \, (P_\Gamma)^b \, (P_\Gamma')^{\beta - b}. \tag{4}$$

The receiver can therefore receive any combination of a and b shares, each arriving in different order. However, for a successful attack, the receiver must have had received x shares of the a malicious shares first. This is due to the packet recovery process that is immediately carried out on the first x shares that arrives. The probability of this happening is given as,

$$P(x|a) = \begin{cases} \displaystyle\sum_{b=0}^{\beta} \frac{\frac{a!}{(a-x)!} \cdot (a+b-x)!}{(a+b)!} \cdot P_{\text{recv}}(a) \cdot P_{\text{recv}}(b) & x \leq a \\ 0 & x > a \end{cases}. \tag{5}$$

Similarly, equation (6) follows from equation (5) for the first y shares of the b harmless shares that are received by the CIDS.

$$P(y|b) = \begin{cases} \displaystyle\sum_{a=0}^{\alpha} \frac{\frac{b!}{(b-y)!} \cdot (a+b-y)!}{(a+b)!} \cdot P_{\text{cids}}(a) \cdot P_{\text{cids}}(b) & y \leq b \\ 0 & y > b \end{cases}. \tag{6}$$

where

$$P_{\text{cids}}(a) = {}^{\alpha}C_a \, (P_\Sigma)^a \, (P_\Sigma')^{\alpha - a} \tag{7}$$

$$P_{\text{cids}}(b) = {}^{\beta}C_b \, (P_\Sigma)^b \, (P_\Sigma')^{\beta - b}. \tag{8}$$

An evasion attack is successful if the receiver receives at least k malicious shares first and the CIDS receives at least k forged but harmless shares first in equations (5) and (6), respectively. Subsequently, the probability of false negative is given as,

$$P(FN) = \sum_{a=k}^{\alpha} P(x = k|a) \cdot \sum_{b=k}^{\beta} P(y = k|b). \tag{9}$$

We have the conditions $\alpha + \beta \leq n$ and $\alpha, \beta \geq k$. If the attacker uses the minimum number of shares for both α and β which is k, the total number of shares is $2k$. Evasion attacks thus becomes strictly impossible if $2k \geq n$.

3.3.2 Minimising packet loss

Our protocol is lossy by design because the proxies randomly drop shares with the probability P_d. If shares are dropped too often, the receiver or CIDS may not receive at least k shares to recover the network packet and network performance deteriorates as a result.

Therefore, to minimise frequent packet loss and maintain the reliability of the network, the parameters P_Γ, P_Σ as well as n and k have to be carefully selected and tuned. We begin by first determining the probability of a receiver receiving exactly x shares from any of the n proxies. This is expressed as follows:

$$P(x) = {}^{n}C_x \, (P_\Gamma)^x \, (P_\Gamma{}')^{n-x} \tag{10}$$

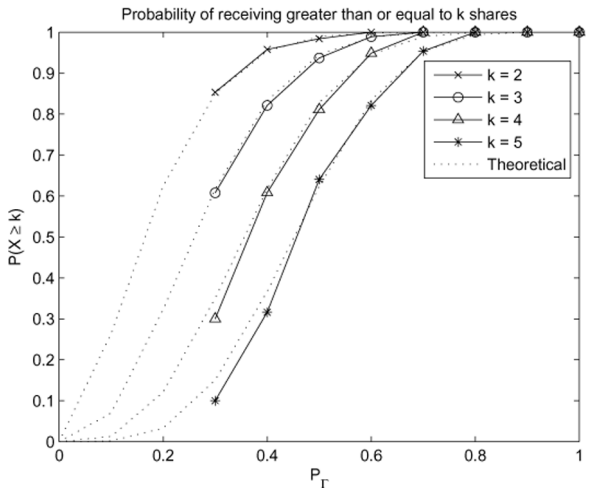
where $P_\Gamma{}' = 1 - P_\Gamma$. To properly recover a network packet, the receiver must receive at least k shares. The probability of this is:

$$P(X \geq k) = \sum_{x=k}^{n} P(x). \tag{11}$$

In order to minimise packet loss, we require that the receiver receives at least k shares from the proxies with very high probability. Ideally, this should be $P(X \geq k) \approx 1$. The parameters P_Γ, n and k have to be adjusted accordingly to achieve this.

To better visualise the relationship between these parameters and simplify the selection process, we plot equation (11) against P_Γ for various values of n and k. An example of this can be seen in Figure 2 for $n = 10$. In Figure 2, if we had chosen $n = 10$ and $k = 3$, any value of $P_\Gamma \geq 0.7$ is suitable because $P(X \geq k) \approx 1$.

Figure 2 The probability of the receiver receiving at least k shares for proper packet recovery when $n = 10$



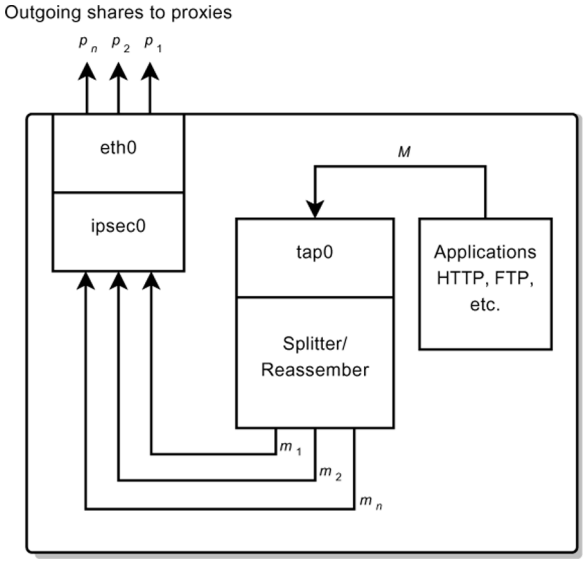
We can verify the results in Figure 2 by measuring the packet acceptance rate. This is suitable because every successfully recovered packet implies that at least k shares have been sent by the proxies to the receiver. We use the publicly available tool *mtr* to measure the packet acceptance rates of our implementation. The experimental results are shown as solid lines in the figure. As it can be seen, the experimental results confirm equation (11).

The same line of reasoning follows for $P(y)$ and $P(Y \geq k)$ from equations (10) and (11) but with respect to the CIDS instead.

3.4 Summary of implementation

We have implemented a Linux-based prototype of the proposed approach. It has been developed using the Python programming language, together with the Python/C API and the Twisted (2009) networking engine. This language and tools allowed for shorter development time while maintaining application reliability. The overall design of the prototype can be seen in Figure 3.

Figure 3 Design of prototype with tun/tap network interface



The implementation exposes a *tun/tap* virtual network interface to applications requiring network access. This ensures that application layer protocols such as HTTP, FTP or SMTP continue to function normally without requiring any patches or modifications.

From Figure 3, we see that outgoing network traffic is handed off to our userland application for processing (packet splitting and encapsulation) via the tun/tap network interface. The packet's corresponding shares are then sent to the receiver through the sender's VPN network interface. The process is reversed at the receiver's side for incoming traffic. Whenever k or more shares have been received by the CIDS or receiver, the packet is immediately recovered and verified. The cryptographic hash

function MD5 is used to quickly and accurately determine whether the packet is corrupted.

As mentioned earlier, our approach is a tunnelling protocol that routes network packets between two hosts in a network. Like many other tunnelling protocols, our prototype has been designed to be connectionless. This means that message delivery and packet ordering are not guaranteed. All network traffic are tunnelled using UDP over IP. A connectionless tunnel is easier to set up and manage because it does not need to manage flow control, acknowledgements or timers that may conflict with the inner tunnelled protocol. Since the exposed tun/tap interface provides applications simply with a virtual IP interface, TCP can evidently be used over our protocol if a connection-oriented link is required.

Forwarding proxies are not separate or standalone devices. Rather, all hosts in the network can assume the role of a proxy as well as being normal communicating peers. Each host has the forwarding capability integrated into the installed tun/tap network interface. The number of available proxies (N) therefore grows together with increasing number of hosts in the network.

4 Evaluation

A VPN with 12 hosts and one CIDS running a standard SNORT installation is used as the experimental network. The hosts consists of Virtual Machines running Linux Ubuntu 8.04. The network uses a 10/100 Mbps ethernet switch to connect all the hosts together. The VPN uses IPsec where Security Associations (SA) are established between all connected hosts.

We begin our evaluation by ensuring that our proposed approach can accurately detect application-level attacks and does not introduce any additional false positives. We apply a dataset of synthetic traffic against SNORT in a network with and without our implementation. The dataset consists of both malicious and non-malicious traffic. Throughout the experiment, we observe no differences in the number of SNORT alerts when our implementation is in use, compared to a network without it.

We do not evaluate the detection accuracy of SNORT by itself as this measurement is better carried out by its developers. Instead, we are only interested in determining the effects and responses that a new network environment has on the NIDS. As expected, the number of SNORT alerts remained consistent for both network setups. Our implementation does indeed function as a tunnelling protocol as far as legitimate traffic is concerned.

The following sections focuses on measuring the performance of our approach and its implementation on detecting attacks in encrypted networks.

4.1 Overhead and complexity

One concern in using our proposed approach is the implied network overhead. If a sender sends one packet of data to a receiver with our proposed approach, the data will be splitted into n shares. If each share is sent as one network packet, approximately n packets are needed. In addition, for the same packet of data, at most another n packets are sent by the proxies to the CIDS, as required by

our approach. Generally, for every t packets sent, the total network overhead is $2tn$ packets.

The $2tn$ increase in network overhead may not be considered efficient when large volumes of data are sent. For this reason, we believe that the approach should be used selectively. For instance, some traffic intensive applications like Voice over Internet Protocol (VoIP) and media streaming can be deemed safe and do not require our approach. On the other hand, applications such as e-mail, authentication or even web browsing can use our approach because of the bursty nature of such traffic.

The computational complexity involved in splitting and recovering the shares can be estimated. In a (k, n) secret-sharing scheme, a polynomial of degree $k-1$ is evaluated at n points. Readers are referred to Shamir (1979) for further details. Using the Horner algorithm, the polynomial can be expressed as multiple linear operations. This reduces the splitting process to a complexity of $O(n)$. Conversely, the packet recovery process uses a Lagrange interpolation polynomial. This operation is known to be quadratic, that is, a complexity of $O(k^2)$.

4.2 Experiment results

In the following experiment, we assume the presence of a sophisticated attacker, capable of launching evasion attacks. An evasion attack requires the attacker forge harmless network traffic in addition to the intended malicious traffic.

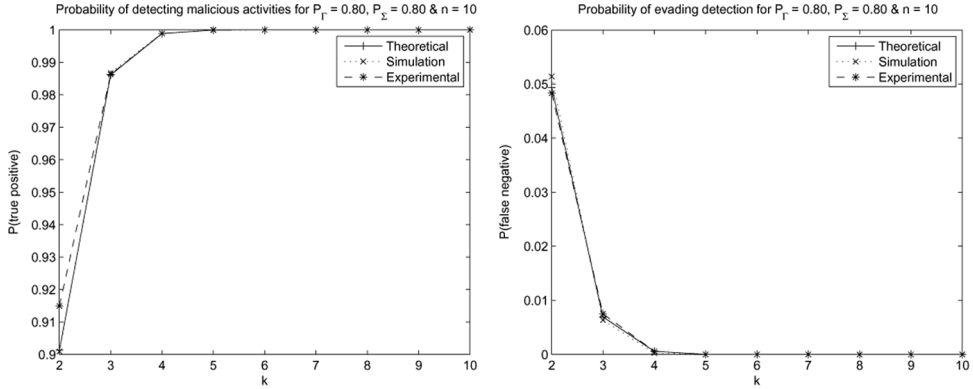
The harmless network traffic could be a trivial "`GET /index.html HTTP/1.1`" while the malicious traffic is "`GET /x90x90x90x90...`". A single `x90` represents the *No Operation* (NOOP) instruction and is usually harmless individually but when repeated as a sequence (NOOP sledge), it is often symptomatic of a buffer overflow-based malicious shellcode injection attack. Consequently, most misuse-based IDS such as SNORT use `x90` as a detection signature.

The goal of an attacker is to send the harmless traffic to the CIDS and the malicious traffic to the intended target host. According to the protocol, both the harmless and malicious traffic are routed through the forwarding proxies before arriving at the CIDS and target host. However, due to the random nature of the proxies, the two different types of traffic usually do not arrive at their intended destination.

In most cases, a mixture of shares will occur, resulting in corrupted network packets. These corrupted packets are easily detected and their presence indicate that there has been some malicious activities. We consider this as a *true positive* detection of a malicious sender. In our experiments, we attempt to evade detection by sending both the harmless and malicious network traffic. We repeat this 6000 times and measure the frequency of successfully detecting evasion attacks.

The probability of detecting an attack when $P_\Sigma = 0.8$ and $P_\Gamma = 0.8$ can be seen in Figure 4(a) while the probability of an attacker successfully evading detection and causing a *false negative* detection outcome is shown in Figure 4(b). Each figure shows plots for results obtained from the theoretical model that is equation (9), MATLAB simulations and our experiments. These plots match up very closely, indicating that the implementation behaved as expected. If the attack consists of t multiple successive packets, the probability of evading detection drops further at an exponential rate of $P(\text{false negative})^t$.

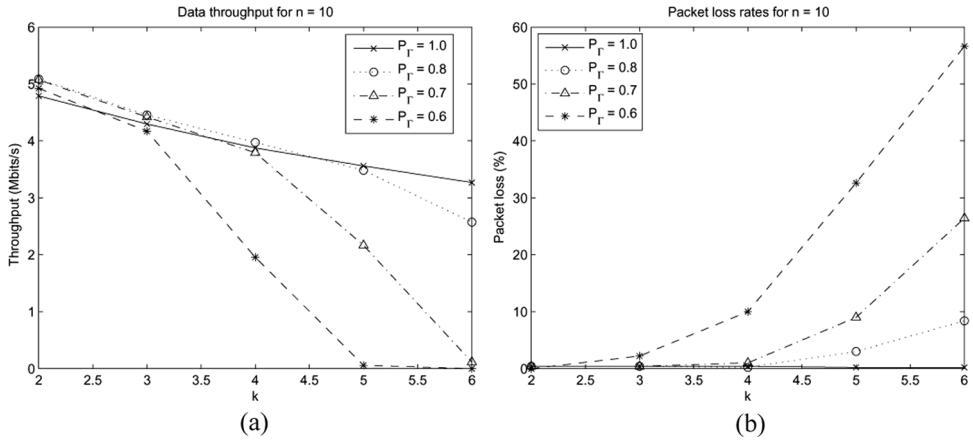
Figure 4 Probabilities of detector's outcome when $P_\Sigma = 0.8$ and $P_\Gamma = 0.8$ and $n = 10$: (a) probability of detection and (b) probability of evading detection



4.2.1 Single point-to-point link

We establish a single point-to-point connection between a pair of hosts and measure its achievable data rates. Figure 5(a) shows the data throughput (Mbits/s) of our prototype in our experimental network. We measure the results on the receiver using *iperf*. The different plots in Figure 5(b) are obtained when the parameter P_Γ is varied, thus changing the probability of the proxies forwarding a particular share to the receiver.

Figure 5 Data throughput and packet loss rates for different P_Γ: (a) data throughput and (b) packet loss



There is a general decline in throughput as k increases. Network latency depends on the parameter k because the receiver must have received at least k shares before the original packet can be recovered. Due to the unpredictable nature of the proxies, there exist situations when less than k shares are actually forwarded to the receiver. This is more prevalent with larger values of k. Since all proxies operate

independently and do not coordinate their actions among themselves, there is no guarantee that at least k shares will be delivered. As such, packet loss will occur and the overall latency increases. This is evident in Figure 5(b) which shows the corresponding packet loss rates.

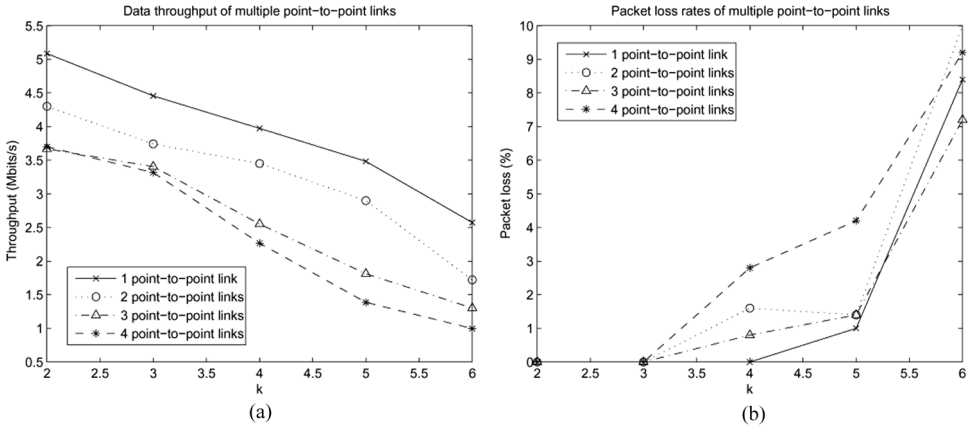
Additionally, with larger values of k, the receiver will have to wait longer for all k shares to arrive before the packet can be recovered. The increased latency results in lower throughput as can be seen in Figure 5(a). If we observe the plot for $P_\Gamma = 1.0$ in Figure 5(a), the throughput drops by about 31% for $k = 6$ compared to $k = 2$ even though the corresponding packet loss rate is 0% in Figure 5(b). This means that the increased latency is caused exclusively by the larger value of k and not because of packet loss.

4.2.2 Multiple point-to-point links

Following that, we measure both the throughput and data loss rates of the receiver under heavy network load conditions. To simulate a busy network, we carry out simultaneous point-to-point connections between different pairs of hosts in the network while measuring the data rates on a single host. All the point-to-point connections use our protocol.

For this round of experiments, we set $P_\Gamma = 0.8$ and vary the number of simultaneous point-to-point connections. Figure 6(a) shows the different data throughput plots while Figure 6(b) show the corresponding packet loss rates. Consistent with our earlier results, the data throughput generally decreases with increasing values of k because of increasing packet loss.

Figure 6 Data throughput and packet loss rates when protocol is used under heavy network load conditions while $P_\Gamma = 0.8$: (a) data throughput and (b) packet loss



Unlike Figure 5(a), the different plots in Figure 6(a) do not tend to overlap but instead have lower average throughputs with increasing network load. This could be attributed to our prototype that was implemented as a Python userland application. The practice of processing incoming network packets through a userland application before sending it as outgoing network packets is usually inefficient. Doing so requires that network packets be constantly passed between the

kernel and userspace, taking up valuable resources and computing time. It would have been more efficient if the prototype had been implemented as a kernel module instead.

As a compromise, we refactored our codes and also applied (Psyco, 2009) which is a Just-In-Time (JIT) compiler on our prototype to improve its performance. By doing so, the prototype's performance improved by a factor of about 200 times. This is significant improvement compared to the results obtained in Goh et al. (2009). The figures in this paper reflect the latest results.

4.3 Discussions

Higher throughput can be obtained by using smaller values of k, as indicated by Figure 5(a). Although this increases network performance, it also increases an attacker's chance of evading detection as shown in Figure 4(b). Therefore, it is important to select a reasonable value of k such that throughput can be maximised while minimising the probability of a successful attack.

A suitable value of k can be identified using Figure 2. For any given value of P_Γ (or P_Σ on an equivalent plot), we choose the greatest value of k that results in $P(X \geq k) \approx 1$. This gives us a minimum packet loss rate which in turn ensures high throughput. Besides that, the greatest value of k implies lower probability of a successful attack.

For instance, if we let $P_\Gamma = 0.7$, we can actually use $k = 2, 3$ or 4 because each of them results in $P(X \geq k) \approx 1$ as shown in Figure 2. However, of these three values, we use only $k = 4$ because this gives the lowest probability of a successful attack as shown in Figure 4(b). According to Figure 5(a), this still produces a relatively high throughput and is only a 14% reduction in throughput compared to the throughput when $k = 3$.

From Figure 6(a), we see that network throughput tends to decrease with increasing traffic in the network. This seems to indicate the high overhead of the protocol. We believe that these results are due in part to the bottleneck at the proxies. In the experiment, the forwarding proxies are not only forwarding shares between a sender and receiver, but they are themselves sending and receiving traffic. This strains both the Python userland application as well as the limited computing resources.

There are two approaches in addressing this bottleneck. Firstly, the implementation could be further improved upon. A kernel module may work reasonably well towards this end. Secondly, if the total number of hosts is increased, we can expect better network performance. As the number of hosts grows, so does the availability of proxies. This is because the forwarding capabilities are simply built-in features of the application running on each host. With a larger network, the load is spread more evenly across all proxies and does not overload any particular proxy.

Our approach thus far does not consider cases of multi-node conspiracy. A conspiracy in our context is a scenario where an attacker collaborates with already compromised proxies to further propagate its malicious activities without being detected by the CIDS. Rather than being unpredictable to the attacker, a conspiring proxy can actually dictate which share (malicious or harmless) be forwarded to whom (CIDS or receiver).

An important pre-condition to the multi-node conspiracy problem is the presence of already compromised hosts in the network. These hosts can be added one at a time into the network until the minimum threshold of k hosts is reached before any consipracy attempts are made. The addition of already compromised hosts occurs if mobile devices such as notebooks are permitted to join the network without any prior authorisation. With sufficient conspirators, attacks can be carried out without alerting the CIDS.

On the contrary, if the network is initially 'clean' and no compromised hosts are added, we believe that the conspiracy problem cannot happen unless k or more conspiring hosts are introduced simultaneously to the network. Assuming that there is a "single attacker in a clean network" only, the attacker will first have to 'recruit' sufficient conspirators. While doing so, our proposed approach should be able to detect the 'recruitment' activity. We intend to study this conspiracy problem in greater detail in our subsequent work.

4.4 Comparison with previous works

Md. Fadlullah et al. (2007), Joglekar and Tate (2004) use specification-based detection systems to detect attacks that specifically target encryption protocols. For example, many failed SSL session negotiations may imply repeated brute-force password-guessing attempts. This detection method is suitable for use in encrypted networks because it does not require the payload contained within the encrypted packet for analysis. Similarly, Yamada et al. (2007) and Piccitto et al. (2007) only monitor outwardly observable features of encrypted packets for indications of potential attacks.

These methods can only detect a subset of a broad range of possible attacks because they do not analyse the payload. In such case, application-level attacks such as buffer overflow or SQL injection attacks can escape undetected. On the other hand, our protocol ensures that the NIDS will always have application layer payload to analyse, increasing the likelihood of detecting application-level attacks. This is achieved without the need of a NIDS-in-the-middle network topology where all network traffic are routed through a single NIDS for analysis. Such a topology may inadvertently increase the risk of network downtime should the NIDS fail.

In Abimbola et al. (2006), IDS sensors that are capable of accessing already decrypted payload are installed on network hosts. These sensors channel the payload into a local NIDS for analysis. As with a HIDS, maintaining a local IDS in each host is difficult to deploy and maintain. Furthermore, the local IDS is susceptible to attacks if the host itself is compromised. Our protocol consists of only a custom network interface driver and a centrally managed CIDS. The driver only needs to be installed once and does not need to be further maintained. Any necessary updates such as installing the latest attack signatures need only be carried out on the CIDS, thus simplifying update procedures. In addition to being easier to manage, a CIDS can be better secured against attacks instead of redistributing resources to secure every host in the network that has a local IDS installed.

An important feature of our proposed protocol is the use of secret-sharing to send network packets via the proxies. Secret-sharing ensures that the proxies do not gain any insight about the content of the packet. This guarantees that the protocol does not compromise on confidentiality while trying to maintain network security.

Most importantly, by using secret-sharing instead of classical encryption algorithms such as AES and RSA, we avoid the need for a complicated PKI infrastructure to manage the keys.

The development of our approach has been motivated by the need to be able to monitor encrypted networks for malicious activities. Despite that, we believe that it can be adapted for applications where nodes of a multi-node network cannot be trusted.

Such is the case with *mobile ad-hoc networks* (MANETs). A MANET is formed when a group of mobile nodes cooperatively communicate with each other without a pre-established infrastructure. According to Tseng et al. (2006), a MANET is inherently trust-all-peers by design and therein lies its problem. A malicious node can corrupt other trusting nodes by forging incorrect data packets to evade detection. This problem bears similarities with our work, specifically the fact that not all nodes can be fully trusted to be truthful or follow the protocol. We will further examine this in the later phases of our work.

5 Conclusion

In this paper, we present a protocol that addresses the problem of analysing network traffic in encrypted networks. To the best of our knowledge, our protocol is the first work at directly integrating a NIDS into an encrypted network in such a way that deep packet inspection can be carried out. The protocol achieves this by ensuring that all network traffic bound for the receiver is also sent to a NIDS for inspection, while keeping data confidential.

Although the idea of performing deep packet inspection into encrypted networks seems to be counterproductive because it violates the basic premise that encrypted tunnels are completely confidential, it is still a logical defence mechanism. This is motivated by the notion that confidentiality can be misused by attackers as a covert channel to bypass perimeter security and evade detection.

Despite the growing need to secure this security gap, there has still been little research into this problem. As it is, most network administrators assume that this problem can be reasonably dealt with using host-based defences such as anti-virus or HIDS. These solutions do not scale very well and are oftentimes resource intensive. Statistical traffic analysis methods also do not provide sufficient scope for an accurate identification of malicious traffic.

The main contribution of this protocol is its ability in maintaining the integrity and privacy of the network traffic while making it difficult for an attacker to subvert detection. We do this by using a combination of encrypted network protocols such as IPsec together with the forwarding proxies to provide traffic confidentiality. Furthermore, our protocol avoids the need for a PKI entirely.

Our evaluations show that the protocol is able to detect both application-level and evasion attacks. The protocol does not cause additional false positives on top of the ones already generated by the NIDS. The results that we obtained through our evaluations also indicate that the protocol performs reasonably well.

In spite of that, our protocol does introduce some overhead and implementation challenges. Firstly, there is the communication overhead in the form of multiple shares of the same network packet. Additionally, the splitting and recovery of

shares using a (k, n) secret-sharing scheme is computationally expensive. As such, we believe it is best suited for applications such as e-mail and authentication, which is not traffic intensive but more bursty in nature.

One of the implementation challenges that had to be overcame is maintaining low packet loss rates even though our protocol is lossy by design. We have shown that this is possible by the proper selection of the parameters k, P_Γ, and P_Σ. Our recent research focuses on addressing the multi-node conspiracy problem. We aim to use the methods developed there to further enhance our ability to quickly identify and isolate malicious hosts.

Acknowledgements

This work is supported in part by funds from International Information Systems Security Certification Consortium, Inc., $(ISC)^2$.

References

Abimbola, A., Munoz, J.M. and Buchanan, W.J. (2006) 'Nethost-sensor: investigating the capture of end-to-end encrypted intrusive data', *Computers and Security*, Vol. 25, No. 6, pp.445–451.

Diffie, W. and Hellman, M.E. (1976) 'New directions in cryptography', *IEEE Transactions on Information Theory*, Vol. 22, No. 6, November, pp.644–654.

Foroushani, V.A., Adibnia, F. and Hojati, E. (2008) 'Intrusion detection in encrypted accesses with SSH protocol to network public servers', *International Conference on Computer and Communication Engineering (ICCCE'08)*, Kuala Lumpur, Malaysia, May, pp.314–318.

Goh, V.T., Zimmermann, J. and Looi, M. (2009) 'Towards intrusion detection for encrypted networks', *4th International Conference on Availability, Reliability and Security (ARES'09)*, March, IEEE Computer Society, Fukuoka, Japan, pp.540–545.

Joglekar, S.P. and Tate, S.R. (2004) 'Protomon: embedded monitors for cryptographic protocol intrusion detection and prevention', *International Conference on Information Technology: Coding and Computing (ITCC'04)*, April, IEEE Computer Society, Las Vegas, Nevada, USA, pp.81–88.

Kent, S. and Atkinson, R. (1998a) 'RFC 2406: IP encapsulating security payload (ESP)', November.

Kent, S. and Atkinson, R. (1998b) 'RFC 2402: IP authentication header (AH)', November.

Md. Fadlullah, Z., Taleb, T., Ansari, N., Hashimoto, K., Miyake, Y., Nemotoi, Y. and Kato, N. (2007) 'Combating against attacks on encrypted protocols', *IEEE International Conference on Communications (ICC'07)*, Glasgow, Scotland, June, pp.1211–1216.

Piccitto, D., Burschka, S. and Urvoy-Keller, G. (2007) *Traffic Mining in IP Tunnels*, Master's Thesis, Eurecom Institute, Sophia-Antipolis, France, September.

Psyco (2009) http://psyco.sourceforge.net/. This is an electronic document, Date retrieved: January 29, 2008.

Richardson, M.C. and Redelmeier, D.H. (2005) 'RFC 4322: opportunistic encryption using the internet key exchange (IKE)', December.

Roesch, M. (1999) 'Snort – lightweight intrusion detection for networks', *13th Large Installation System Administration Conference (LISA'99)*, November, Seattle, Washington, USA, pp.229–238.

Shamir, A. (1979) 'How to share a secret', *Communications of the ACM*, Vol. 22, No. 11, November, pp.612–613.

Tseng, C.H., Wang, S-H., Ko, C. and Levitt, K. (2006) 'DEMEM: distributed evidence-driven message exchange intrusion detection model for MANET', in Zamboni, D. and Krügel, C. (Eds.): *9th International Symposium on Recent Advances in Intrusion Detection (RAID'06)*, Volume 4219 of *Lecture Notes in Computer Science*, Springer-Verlag, Hamburg, Germany, September, pp.249–271.

Twisted (2009) http://twistedmatrix.com. This is an electronic document, Date retrieved: January 29, 2008.

Wagner, A., Dübendorfer, T., Hiestand, R., Göldi, C. and Plattner, B. (2006) 'A fast worm scan detection tool for VPN congestion avoidance', in Büschkes, R. and Laskov, P. (Eds.): *3rd International Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA'06)*, Volume 4064 of *Lecture Notes in Computer Science*, July, Springer-Verlag, Berlin, Germany, pp.181–194.

Yamada, A., Miyake, Y., Takemori, K., Studer, A. and Perrig, A. (2007) 'Intrusion detection for encrypted web accesses', *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, May, Niagara Falls, Canada, pp.569–576.

Yasinsac, A. and Childs, J. (2001) 'Analyzing internet security protocols', *6th IEEE International Symposium on High Assurance Systems Engineering (HASE'01)*, IEEE Computer Society, Boca Raton, Florida, USA, October, pp.149–159.