Chapter 17

ACTIVE TRAFFIC CAPTURE FOR NETWORK FORENSICS

Marco Slaviero, Anna Granova and Martin Olivier

Abstract Network traffic capture is an integral part of network forensics, but current traffic capture techniques are typically passive in nature. Under heavy loads, it is possible for a sniffer to miss packets, which affects the quality of forensic evidence.

This paper explores means for active capture of network traffic. In particular, it examines how traffic capture can influence the stream under surveillance so that no data is lost. A tool that forces TCP retransmissions is presented. The paper also provides a legal analysis—based on United States and South African laws—which shows that few legal obstacles are faced by traffic capture techniques that force attackers to retransmit data.

Keywords: Network forensics, active traffic capture, TCP retransmission

1. Introduction

Traffic capture has long been the mainstay of network forensics, providing the raw data which is analysed and dissected to reveal important information. This capturing or "sniffing" is an inherently passive process. Typically, the capturer places a network interface in "promiscuous" mode, and records each frame as it arrives. Passive capture can, in some circumstances, be a hindrance. Consider a situation where the capturer has missed a frame due to some error in the network or at the capturer. This gap in the record weakens the confidence in the information gleaned.

This paper proposes a new paradigm termed active capture, in which the capturer influences the communication stream under examination (within the bounds of the applicable protocols) to provide as clear a picture as possible of the event at hand. The influence could occur

at almost any position in a network stack, and careful consideration is required to determine the optimal point for manipulating the traffic stream.

Active traffic capture is a controversial issue. Our purpose is to show that active capture is possible in a technical as well as legal sense. The technical and legal positions are discussed with regard to a hypothetical scenario. Alice, the IT security officer of Super Software Closed Corporation (SSCC), discovers that Eve, an aspiring cracker, is trying to penetrate the network entrusted to Alice. The picture becomes clearer and the evidence mounts as Alice passively monitors the network. At some point, the understanding of Eve's actions blurs because one of the packets involved in Eve's attack on the victim machine is lost. All Alice has to do to rectify the situation is to send a request for the lost packet to be resent. Since it is not unusual for such requests to come through, Eve would not be aware that she is being monitored.

Based on the hypothetical scenario, two questions arise: Is forcing Eve to resend lost packets technically feasible? Is it legal for Alice to engage in such behaviour and if so under what conditions?

This paper does not describe an active capture tool for network forensics. Rather, it discusses avenues for future research with the knowledge that active capture techniques may have legal standing. To this end, the technical discussion focuses on the placement of a capturer with respect to the traffic stream, a TCP implementation that forces data retransmission, and how traffic might be influenced during its capture.

The legal component discusses relevant United States (US) and South African (SA) legislation, including the US Electronic Communications Privacy (ECP) Act, the SA Constitution and related case law; the SA Electronic Communications and Transactions (ECT) Act; and the SA Interception Act.

The paper is structured as follows. The next section, Section 2, discusses technical issues related to active traffic capture using TCP retransmission. Section 3 examines the notion of "influence" and how it affects communications. Section 4 presents relevant legislation and case law. The final section, Section 5, presents the conclusions and avenues for future work.

2. Retransmission Technicalities

Packet-switched networks are relatively unreliable. Stone and Partridge [18] report Internet packet error rates as high as 1 in 1,100. Most data network implementations have many disparate modules; a small error or malfunction in just one module often creates error states in other

modules. Corruption is generally introduced at a physical level, in which signals on a medium are disrupted by faulty hardware or interference. Software errors are not unknown; the most common destination for erroneous packets is the bit-bucket, where the packets are simply discarded. In severely congested networks, data segments are lost when receivers (mostly routers) run out of buffer space to store incoming packets. Packets may also be lost due to unreachable routes or looping routes.

In the face of these obstacles, a retransmission strategy is used to ensure that data eventually gets to the correct receiver. Simply put, data is sent and resent until it is received.

Returning to the notion of disparate modules constituting a network, it is helpful to view them in a vertical fashion, with one module stacked on top of another. ISO's Open Systems Interconnect (OSI) model [8] is the most commonly used reference stack. Error states can occur at any of the seven layers in the OSI model.

Each layer in the OSI model can retransmit data if it detects errors; the function is not limited to certain modules. In the lower layers of the stack, collisions are common, and they are even expected in Ethernet networks. Collisions are quite normal because Ethernet is a multiple access medium: whenever a node wishes to send it does so. The OSI standard allows nodes to detect when multiple senders transmit at the same time. If collisions are detected, the senders halt and wait before retransmitting [1].

The Transmission Control Protocol (TCP) [11], which is higher in the stack, introduces reliability in IP [10] networks. (IP is the most prevalent packet-switched network protocol.) A receiving TCP tracks the bytes sent to it. If loss is detected, it informs the sending TCP, which then retransmits the lost sequence.

Finally, several higher-level protocols include facilities for retransmitting data. For example, the common email standard SMTP instructs mail servers to continually attempt to deliver mail until the destination acknowledges receipt. Thus, if a connection is reset while mail is being transferred, the originator will attempt to reconnect and resend the undelivered message [12].

All the retransmissions described so far are performed automatically. However, certain errors are not easily solvable by protocol specifications and these errors are usually reported to humans to investigate and solve, perhaps by manual retransmission. An example is the `504` error reported by web servers when a user attempts to visit a web page while the server is overloaded [4]. In this case, the retransmission occurs manually upon user input.

Returning to the attack scenario, assume that Alice wishes to track what Eve is doing for investigative or evidentiary purposes. Note that Alice is not a party to the communication, but she can record the communication. If a particular sequence of bytes in Eve's attack on a victim machine is lost to Alice, it might be possible to force the network stack on Eve's machine to resend the lost data. Retransmission can occur at various network layers, but the choice of layer at which to force retransmission is dependent on the type of attack.

If one of the lower stack layers is chosen, say the data-link layer, then a number of restrictions are placed on Alice. Foremost is overcoming the wide variety of link layer protocols. Under Ethernet, the goal of willingly forcing retransmission is extremely difficult, if not almost impossible. It would require the monitor to know that it has not read the packet while it is busy receiving the packet. This is because Ethernet interfaces only retransmit a frame if a collision occurred; there is no way to explicitly request retransmission and the monitor can only generate a collision. Also, this assumes the decision is made at the hardware level, while sniffers generally operate in software. In any event, such retransmission would require special hardware and is not explored any further in this paper. Consequently, higher layers of the stack must be examined.

Retransmission is very application specific at the highest layers of the network stack. If a monitor hopes to force retransmission, its software would have to process each higher layer protocol. This is a massive task given the vast array of protocols.

However, it is not unreasonable to conceive a situation where a monitor is interested in a specific protocol. A good example is a case where a blackmailer sends an email from an Internet cafe to a victim. Investigators suspect an individual, but may wish to gather more evidence, specifically time correlation data between when the email was sent and when the suspect was at the Internet cafe. Investigators could use a tool that fakes a bounced email to the suspect. The suspect, thinking that the original email was not received by the victim, then sends a second email. In such a situation, writing specialised software for a protocol might be worth the effort, but for general use, a compromise between the higher and lower stack layers would be necessary.

The techniques presented in the paper are applicable to reliable protocols, such as TCP, which guarantee that data arrives in the correct sequence. (UDP is a close cousin of TCP, but it does not provide reliability assurances.) However, TCP is agnostic towards the data it carries; the data is simply passed to the receiving application. TCP has a number of built-in facilities for retransmitting packets when losses are detected. Because TCP processing occurs in software on virtually all

TCP-capable nodes, it is possible to examine packets and determine if packets have been lost or read incorrectly *post factum*, and then request retransmission of the packets.

2.1 TCP Retransmission

This section discusses TCP retransmission. Background information about TCP is provided; readers are referred to [11] for additional details.

2.1.1 Sequence Numbers and Retransmission. Under TCP, the data being carried is viewed as a byte stream [11] by the sender $\mathcal{S}$, where each byte is numbered sequentially modulo 2^{32}. When a packet is sent, the number of the first data byte of the packet is carried along with the data. This is called the sequence number of the packet, which is denoted by *seq*.

A receiving TCP $\mathcal{R}$ sends periodic acknowledgements to $\mathcal{S}$ indicating θ, the point in the stream up to which all bytes have been received. Thus, if a packet is lost or corrupted, $\mathcal{R}$ simply keeps sending acknowledgements holding θ.

Vanilla TCP uses timeout queues to implement retransmission. When a packet is first sent, a copy is placed in a timeout queue. If an acknowledgement is received that covers the data in the packet on the timeout queue, it is deleted from the queue. Whenever a timeout occurs, the packet is resent, and it remains in the queue.

Jacobson [6] noted that a lost packet can be inferred by assuming that when three acknowledgements with the same sequence number arrive, the packet starting with that sequence number has not arrived. The sender can then retransmit the supposed missing packet. This improvement is documented in [17], and is incorporated in virtually all modern TCP implementations. Other acknowledgement schemes exist for TCP [5, 7], but support for these schemes is not as widespread.

2.1.2 Forcing Retransmission. At this point the gist of how TCP provides a high degree of reliability to upper layers should be clear. The problem for a third party who is attempting to capture a particular stream is that if, due to local conditions or network congestion, a particular packet does not arrive at the capturing station, a portion of the evidence might be lost.

TCP uses two methods for resending packets: queue timeouts and duplicate acknowledgements. The latter can be forced on the sender, while the former cannot be directly effected by a remote host. Therefore, we can choose one of two positions for the capturing party: (i) on the path between the two end points, which we term an "interceptor," or

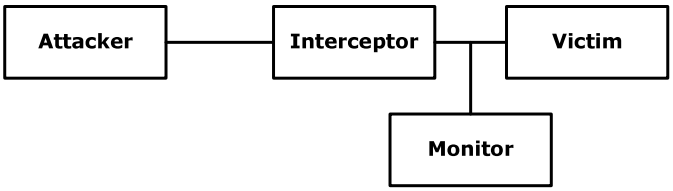

Figure 1. Placement of traffic capturing parties.

(ii) listening in on the communication at some point in the network, which we call a "monitor." Note that the monikers, interceptor and monitor, respectively, correspond to the concept of being a party to the communication, or not being party to the communication, in terms of the South African Interception Act [16]. Figure 1 shows the placement of capturing parties in an attack scenario. Note that only one position needs to be assumed by the capturer.

An interceptor has more influence on a communication stream than a monitor. This is because the stream passes through the interceptor, but only past the monitor. The interceptor can perform transformations on the stream, while the monitor can only react to packets as they go by. Also, an interceptor can force a retransmission by blocking the acknowledgement from the receiver or altering the sequence number. When the associated timeout on the sender expires, the packet is resent. A drawback of interception is the overhead and possible delays that are created because every packet must pass through the interceptor and be examined by it.

The situation for the monitor is more tenuous. Direct action must be taken to force a retransmission and such an action must be expeditious. The rationale is simple: the monitor has to send three duplicate acknowledgements to the sender before the receiver can send one legitimate acknowledgement. It may seem unreasonable to presume this to be possible; however, such a monitor is generally placed at the ingress points to a network. Note that equipping a monitor with more network resources than the receiver reduces the likelihood of the monitor having to force retransmissions since, if a packet is lost, it will probably be lost at the receiver, which will send legitimate duplicate acknowledgements.

The retransmission strategies framed thus far are based on the fact that the capturing party knows when a packet has gone missing. It is easy for the capturer to determine if packets have been reordered.

Specifically, the capturing party $\mathcal{C}$ stores the sequence number seq for each $(\mathcal{S}, \mathcal{R})$ pair. When a packet P arrives from $\mathcal{S}_i$ destined for $\mathcal{R}_k$, $\mathcal{C}$ checks if $(\mathcal{S}_i, \mathcal{R}_k).seq = P.seq$. If it fails, then the most $\mathcal{C}$ can deduce

```
1.  23:15:36.920548 IP eliot.1092 > fornax.ssh: P 1368050088:1368050089(1)
    ack 3457346665 win 64218
2.  23:15:36.920698 IP fornax.ssh > eliot.1092: . ack 0 win 64218
3.  23:15:36.920708 IP fornax.ssh > eliot.1092: . ack 0 win 64218
4.  23:15:36.920724 IP fornax.ssh > eliot.1092: . ack 0 win 64218
5.  23:15:36.921028 IP eliot.1092 > fornax.ssh: P 0:1(1) ack 1 win 64218
6.  23:15:37.052875 IP fornax.ssh > eliot.1092: . ack 1 win 5840
```

Figure 2. Packet trace.

is that a packet has arrived out of order. Checking for packet loss, however, requires an increase in storage space as the last few sequence numbers seen must be stored along with the times of arrival to determine if they have exceeded the Maximum Segment Lifetime. We ignore these complications and treat reordering as packet loss. This simplification is not unreasonable because it is not common to have many reordered packets in a single connection [3].

2.2 Proof-of-Concept Tool

A proof-of-concept tool that forces TCP retransmissions was developed. The tool uses the `libpcap` library [19] for packet capture and raw sockets to transmit forged packets. It can force retransmissions of packets that match certain criteria (using `tcpdump` expressions), enabling specific connections can be targeted.

The test network comprised three machines organised into two networks. The machines `eliot` and `narthex` constituted one network, and `fornax` the other. The machine `eliot` had retransmissions forced on it, `narthex` was the monitor, and `fornax` was the client. The two networks were linked by a slow line to exaggerate the time intervals between forged packets and legitimate packets.

Figure 2 displays a packet trace recorded on `narthex`. The machine `eliot` sends one byte of data to `fornax` in the first packet. The monitor, `narthex`, forges three packets from `fornax` to `eliot` (Steps 2–4). The acknowledgement number in the forged packets is unchanged (shown as `0`), so `eliot` retransmits the packet in Step 5. Finally, the legitimate packet arrives from `fornax`, acknowledging the first packet.

Of particular interest in the packet trace are the time intervals between events. The creation and transmission of forged packets occurred in 176μs, and the forced response arrived 304μs later. As noted previously, a slow link was purposely used between `fornax` and `eliot`; thus, the legitimate packet arrived 132ms after the initial data was sent. This simple test shows that a relatively unsophisticated tool needs a few hundred microseconds to enact the retransmissions.

3. Influence on Communications

A key question regarding traffic capture is whether the communication is unduly compromised by the capturing process. If it turns out that, when monitoring or intercepting a stream, data instead of only control packets were injected into the traffic, then the evidentiary value would be significantly reduced.

It can be argued that by merely plugging into the spanning port of a switch, additional load is created, which could affect the timing of a monitored conversation. The methods proposed in the previous section assume a much more proactive approach to capturing traffic, which certainly is a form of influence. The burden on the monitor is then to show how the influence, whatever it was, did not create, alter or delete data. Since the TCP retransmission technique uses control packets, which are part of the protocol, and data being transferred is not altered in any way, such influence is legitimate.

To verify that no undue influence has taken place, packet logs of the entire communication should be recorded. The logs would show where packets were inserted into the stream and what the effects were.

Of course, the capturer should follow standard investigative policies and practices that are reasonable and demonstrably reliable. The discussion of this issue is outside the scope of this paper.

Although our forced retransmission technique is limited to specific protocols, it is a first step towards active monitoring, which creates numerous opportunities for exploration. Such exploration must be carried out very carefully, otherwise the evidence collected would be inadmissible in court. The next section, which examines two legal systems, shows that merely using active techniques is not grounds for discounting evidence.

4. Legal Framework for Computer Interception

The legality of forcing network retransmissions depends on the laws of the country where retransmissions are attempted. This section focuses on the United States and South African legal regimes.

4.1 United States Framework

Two main acts constitute the U.S. legal framework that governs the interception of and access to information: (i) the Electronic Communications Privacy (ECP) Act (18 U.S.C. § 2510-2521), and (ii) the Stored Communications Act (18 U.S.C. § 2701-2707). For the purposes of this paper, however, only the ECP Act is relevant.

The ECP Act applies to all electronic communications, including the use of e-mail, cell phones, satellite communications and computer-to-computer communications [20]. As a general rule, any interception or disclosure of electronic communication as well as the use or procurement of equipment to intercept such electronic communication is prohibited (18 U.S.C. § 2511(1)) and any violation is levied a fine of $500 or higher. The few exceptions to the rule are: consent to interception, interception in the course of performing duties at the workplace, and court orders authorising such conduct (18 U.S.C. § 2511(2)).

Returning to the Super Software Closed Corporation (SSCC) scenario, it is certain that Eve would not consent to Alice intercepting the attack on SSCC's network. Neither would she consent to a retransmission request. Furthermore, it may be difficult for Alice to obtain a court order that authorises these activities.

There should, however, be no doubt that if Alice resorts to forcing network retransmissions to compile a complete forensic report on Eve's unlawful network penetration, she would definitely be acting within the normal course of her duties as SSCC's IT security officer. As such, Alice's conduct would be legal and Eve would have no recourse against either Alice or SSCC (18 U.S.C. § 2511(2)(i); also see *Quigley v Rosenthal* 327 F.3d 1044 (10^{th} Cir. 2003)).

4.2 South African Constitution

The 1996 Constitution of the Republic of South Africa [14] radically changed the notion of privacy that existed in non-democratic South Africa. Privacy, as a right, was entrenched by Section 14 of the Constitution, and it includes the right to privacy in communications [2].

But this right—like others in the Bill of Rights [14]—is not absolute. There are always competing rights that need to be weighed against each other. In our scenario, it would be, firstly, Eve's right to privacy as opposed to the right to privacy of Super Software Closed Corporation (SSCC). Secondly, SSCC's right to privacy would have to be balanced with Eve's right to a fair trial as envisaged in Section 35(5) of the Constitution. Section 35(5) provides that: "Evidence obtained in a manner that violates any right in the Bill of Rights must be excluded if the admission of that evidence would render the trial unfair or otherwise be detrimental to the administration of justice." The right to fair trial is important here because Alice would in almost all instances request that packets be retransmitted to obtain more evidence for possible legal action against Eve.

Finally, the right to privacy is, in any event, subject to the general limitation clause contained in Section 36 of the Constitution. To this end, the ECT Act and the Interception Act (discussed below) would have to be interpreted to comply with the said requirements. The two acts are considered after the discussion on case law related to the right to privacy, which follows.

4.3 Case Law

During the pre-1996 Constitutional era, two important cases were decided about the right to privacy. First, in *S v I and Another* (1976 (1) SA 781 (R, A.D.)), the court held that the invasion of privacy, which was reasonably necessary, solely with a *bona fide* motive to obtain evidence of adultery was legal. In our scenario, the limitation of Eve's right to privacy would definitely tip the scale in favour of the entitlement to protect the privacy of SSCC. This would make evidence obtained through forced retransmissions legal and admissible in criminal proceedings against Eve.

A leading civil case, decided by the Appellate Division of the Supreme Court, the highest court in South Africa at the time, was *Financial Mail (Pty) Ltd. and Others v Sage Holdings Ltd. and Another* (1993 (2) SA 451 (A)). In this case, the court held that the right to privacy applied to natural persons as well as to organisations. The court decided that the facts justified a conclusion that, in light of contemporary *boni mores* (good morals) and the general sense of justice of the community, the company's right to trade and carry on its business without wrongful interference from others deserved proper protection under the law within the ambit of the right to privacy.

In 2004, the Cape High Court in *Huey Extreme Club v McDonald t/a Sport Helicopters* (2005 (1) SA 485 (C)) held, with reference to Section 14 of the Constitution, that an invasion of privacy would be justified only where the *boni mores* of the community dictated that the limitation is reasonable. This is in light of the interests of an organisation that were sought to be protected. Clearly, IT security for a contemporary and successful business is as important as the interest of public safety for Mr. McDonald in the Cape High Court case. By analogy, this case serves as authority as to why Eve's privacy should be limited in favour of SSCC's interests.

In the light of the case law, it is evident that constitutional protection as enforced by the court today is present if and when required. More clarity, however, may be obtained by considering legislation that applies specifically to the hypothetical scenario.

4.4 ECT Act

The Electronic Communications and Transactions Act 25 of 2002 [15] covers the interception of communications, including computer-related communications, in a very general way. Section 86 of the ECT Act prohibits any interception of data, whether passive or active.

Before we discuss the Interception Act itself, one more reference needs to be made. Sections 86(3) and (4) of the ECT Act make it an offence, *inter alia*, to create and/or buy any software programs that help overcome security measures that are in place for protecting data.

4.5 Interception Act

The Regulation of Interception of Communications and Provision of Communication-Related Information Act 70 of 2002 [16] addresses direct and indirect communication. Transfer of data is classified as "indirect communication" in the Preamble of the Interception Act, since it takes place over a "telecommunication system" [13]. Interception, pursuant to this act, encompasses any acquisition and including the redirection of the flow of any communication through any means. Such wide scope of application is bound to cover the actions of Alice when she manually requests a packet to be retransmitted by Eve's computer.

Two situations are covered by the Interception Act: (i) where Alice is a party to the communication in question, and (ii) where she is not. With respect to Figure 1, Alice is either an interceptor or monitor.

In the first situation, the law is clear: monitoring by Alice is allowed according to Section 4(1) [16]. This section provides that: "Any person, other than a law enforcement officer, may intercept any communication if he or she is a party to the communication, unless such communication is intercepted by such person for purposes of committing an offence." As for the second set of facts, in the case of a monitor, the burden of proof is on Super Software Closed Corporation (SSCC) to demonstrate that the interception of indirect communication did, in fact, comply with the requirements of Section 6 of the Interception Act [16].

The first requirement is that interception must take place in the course of conducting normal business activities. Considering the facts here, it would entail providing proof that the interception falls within the powers of Alice who is charged with protecting SSCC's network from intrusions. Secondly, SSCC would have to show that there was consent of the system controller, whether express or implied, to the interception. Thirdly, the interception would have to be done for a legitimate purpose as defined in Section 6 of the act. In particular, the active gathering of information would have been done to: (i) establish the existing facts (e.g., of the very

intrusion), or (ii) investigate or detect unauthorised use of the network (i.e., to gather evidence for possible legal action against Eve).

5. Conclusions

This work should be of interest to the technical and legal communities. Forced retransmission is potentially a useful strategy for IT security officers and digital forensic investigators. The TCP retransmission technique based on duplicate acknowledgements enables data resending to be forced by a third party. Clearly, this technique can be very beneficial for investigative and evidentiary purposes.

It appears that manual requests for network retransmissions would be considered legal by United States and South African courts, and evidence retrieved by such means would be admissible in civil and criminal trials. IT security officers could, therefore, become more proactive in assisting law enforcement agencies and preventing illegal activities in United States and South African networks.

This approach to active traffic capture is limited to forcing the retransmission of individual TCP frames. Our current and future work is aimed at developing active capture techniques for other layers of the network stack.

Acknowledgement

The authors would like to thank the members of the ICSA Research Group for their comments.

References

[1] ANSI, Information Processing Systems: Local Area Networks – Part 3, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, American National Standards Institute, 1992.

[2] P. Bekker, T. Geldenhuys, J. Joubert, J. Swanepoel, S. Terblanche and S. van der Merwe, *Criminal Procedure Handbook (Sixth Edition)*, Juta and Company, Lansdowne, South Africa, 2003.

[3] J. Bellardo and S. Savage, Measuring packet reordering, *Proceedings of the Second ACM SIGCOMM Workshop on Internet Measurement*, pp. 97-105, 2002.

[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, Hypertext transfer protocol – HTTP/1.1, *RFC 2616*, Internet Engineering Task Force, June 1999.

[5] S. Floyd, J. Mahdavi, M. Mathis and M. Podolsky, An extension to the selective acknowledgement (SACK) option for TCP, *RFC 2883*, Internet Engineering Task Force, July 2000.

[6] V. Jacobson, Congestion avoidance and control, *Proceedings of the ACM SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 314-329, 1988.

[7] M. Mathis, J. Madhavi, S. Floyd and A. Romanow, TCP selective acknowledgement options, *RFC 2018*, Internet Engineering Task Force, October 1996.

[8] ISO, Information Processing Systems – OSI Reference Model – The Basic Model (ISO 7498-1:1994), International Organization for Standardization, 1994.

[9] V. Paxson, End-to-end Internet packet dynamics, *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures and Protocols for Computer Communication*, pp. 139-152, 1997.

[10] J. Postel, Internet protocol, *RFC 791*, Internet Engineering Task Force, September 1981.

[11] J. Postel, Transmission control protocol, *RFC 793*, Internet Engineering Task Force, September 1981.

[12] J. Postel, Simple mail transfer protocol, *RFC 821*, Internet Engineering Task Force, August 1982.

[13] Republic of South Africa, Telecommunications Act (Act 103), 1996.

[14] Republic of South Africa, Constitution of South Africa (Act 108), 1996.

[15] Republic of South Africa, Electronic Communications and Transactions Act (Act 25), 2002.

[16] Republic of South Africa, Regulation of Interception of Communications and Provision of Communication-Related Information Act (Act 70), 2002.

[17] W. Stevens, TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms, *RFC 2001*, Internet Engineering Task Force, January 1997.

[18] J. Stone and C. Partridge, When the CRC and TCP checksum disagree, *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 309-319, 2000.

[19] TCPDUMP (www.tcpdump.org).

[20] J. Winn and B. Wright, *The Law of Electronic Commerce (Fourth Edition)*, Aspen Publishers, New York, 2005.