Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

A Finite State Machine Model of TCP Connections in the Transport Layer

J. Treurniet and J.H. Lefebvre

Defence R&D Canada – Ottawa
TECHNICAL MEMORANDUM
DRDC Ottawa TM 2003-139
November 2003

Canada

A Finite State Machine Model of TCP Connections in the Transport Layer

J. Treurniet and J. H. Lefebvre

Defence R&D Canada – Ottawa

Technical Memorandum

DRDC Ottawa TM 2003-139

November 2003

Abstract

Finite state machines can be used to detect anomalous behaviour in TCP traffic by describing the progression of a connection through states as a result of events based on header flags. The method was applied to real traffic to understand its realistic use and it was found that for the time period analysed here, on the order of 37% of TCP connections do not follow the TCP protocol specifications. The majority of these are a result of malicious activity, and approximately 4% are due to benign anomalies such as unresponsive hosts and misconfigurations. The method may be applied as a network security measure, as a network management tool or as a research tool for the study of TCP behaviour on the Internet.

Résumé

Les modèles d'automate à états finis peuvent servir à déceler les comportements anormaux dans le trafic TCP en décrivant la progression d'une connexion dans ses différents états en fonction des événements basés sur les indicateurs des en-têtes. La méthode a été appliquée au trafic réel pour en comprendre l'utilisation réaliste et on a constaté que, pour la période de l'examen, environ 37% des connexions TCP ne respectaient pas les spécifications du protocole TCP. La majeure partie de ces connexions découlaient d'activités malicieuses, alors que 4% étaient causées par des anomalies bénignes, comme des hôtes qui ne répondaient plus ou des erreurs de configuration. Cette méthode peut être appliquée comme mesure de sécurité du réseau, comme outil de gestion du réseau ou comme outil de recherche pour l'étude du comportement du protocole TCP dans Internet.

This page intentionally left blank.

Executive summary

Background

The protocols employed on computer networks are intended to be well-defined via Request for Comments (RFC) documents [8]. In reality, there is enough freedom in the RFC specifications of the protocols that they can vary widely from one implementation to the next [2, 3, 4].

Here, we study the Transmission Control Protocol (TCP), which is the most prevalent protocol on the Internet today [1]. TCP is a reliable protocol, and its behaviour follows a pattern that is somewhat predictable: a three-way handshake is followed by data exchange, which is followed by a closing. The "flags" in the TCP header convey the commands that carry a connection through these stages. By defining the states of a Finite State Machine (FSM) [17] to reflect the stages of a connection and using the flags as the events that bring about transitions among the states, we model a TCP connection as an FSM.

To use the model for strict anomaly detection [10], a failure state is introduced to indicate the occurrence of a disallowed event or an attempted illegal transition. When the time-ordered flags of a TCP connection is input into the FSM, if the connection enters a failure state or otherwise does not complete, the connection is flagged as anomalous. The state that the connection was in and the event that led to the failure can be stored to give an indication of the reason that the failure occurred. The TCP FSM can be used to detect some network management issues and network security events, and it can also be used as a research tool to study the behaviour of TCP on the Internet.

Principal Results

Traffic for a 24 hour period in August 2000 was input into the TCP FSM to test the realistic use of the algorithm. Two large-scale scans were easily identified. Some slow scans were found by examining the periodicity of anomalies coming from the same source address. The FSM model's reporting method makes it obvious whether a response was made to a scan packet. Side-effects of filtering rules were identified.

Backscatter from scans and denial of service attacks on other networks appeared in our data, as did unresponsive hosts. The effect of network delays could also be seen, and was sometimes disruptive. Some connections were abandoned, verifying accounts from the Internet community that some web connections get stuck in a half-closed state [26]. Other connections that appeared to be abandoned were a side-effect of a possible NAT-related problem: IP addresses and/or ports were being changed during a connection. Other cases of a malfunctioning of a TCP implementation were observed.

Significance of Results and Future Work

The FSM representation of TCP connections provides a means of identifying packet flows on a network that do not conform to the expected behaviour of the protocol. The results

show that the concept is viable and encourages investigation of the algorithm as an strict anomaly detection Intrusion Detection System (IDS).

We would like to extend the model to include traffic generation, i.e. inclusion of anomalies in generated traffic data. The model may also provide a basis for a Markov-based statistical anomaly detection algorithm, where the probability of each connection's validity is calculated. Analysis of the evolution of TCP over time is another application of interest.

J. Treurniet and J. H. Lefebvre; 2003; A Finite State Machine Model of TCP Connections in the Transport Layer; DRDC Ottawa TM 2003-139; Defence R&D Canada – Ottawa.

Sommaire

Renseignements généraux

On veut que les protocoles utilisés dans des réseaux informatiques soient bien définis à l'aide d'appels de commentaires (RFC) [8]. En fait, il y a suffisamment de jeu dans les spécifications des protocoles dans les RFC qu'elles peuvent varier considérablement d'une implantation à une autre [2, 3, 4].

Ici, nous étudions le protocole TCP (Transmission Control Protocol), qui est actuellement le protocole le plus répandu dans Internet [1]. Le protocole TCP est fiable et son comportement est assez prévisible : une poignée de main à trois, suivie d'un échange de données, puis d'une fermeture. Les " indicateurs " de l'en-tête du protocole TCP contiennent les commandes qui sont utilisées au cours des trois étapes de la connexion. Nous avons pu modéliser une connexion TCP sous forme d'automate à états finis (FSM) en définissant les états d'un FSM [17] de sorte qu'ils correspondent aux étapes d'une connexion et en utilisant les indicateurs comme s'il s'agissait des événements qui entraînent les transitions entre les différents états.

Pour utiliser le modèle strictement comme détecteur d'anomalies [10], on introduit un état de défaillance pour indiquer l'occurrence d'un événement interdit ou une tentative de transition illégale. Lorsque les indicateurs d'une connexion TCP, triés par ordre chronologique, sont entrés dans le FSM, ce dernier indique que la connexion est anormale si une défaillance se produit ou si la connexion n'est pas complétée pour une raison ou une autre. L'état dans lequel se trouvait la connexion et l'événement qui a entraîné la défaillance peuvent être stockés pour donner une indication de la raison pour laquelle la défaillance s'est produite. Le FSM TCP peut être utilisé pour déceler des problèmes de gestion de réseau et des événements de sécurité du réseau. Il peut aussi être utilisé comme outil de recherche permettant d'étudier le comportement du protocole TCP dans Internet.

Résultats principaux

Le trafic généré pendant une période de 24 heures en août 2000 a été introduit dans le FSM TCP pour tester l'utilisation réelle de l'algorithme. Il a été facile d'identifier deux tentatives de scannage de ports à grande échelle. On a également découvert des tentatives de scannage lent en examinant la périodicité d'anomalies provenant de la même adresse source. Grâce à la méthode de rapport du modèle FSM, les réponses aux paquets de scannage sont bien évidentes. On a aussi identifié les effets secondaires des règles de filtrage.

Nos données permettaient de voir la rétrodiffusion du scannage de ports et d'attaques entraînant un refus de service dans d'autres réseaux, ainsi que des hôtes qui ne répondaient pas. De, plus, il était possible de voir les effets des délais dans le réseau, qui, parfois, causaient des perturbations. Certaines connexions ont été abandonnées, ce qui permettait de vérifier les rapports d'utilisateurs d'Internet selon lesquels il arrive que des connexions restent figées dans un état semi-fermé [26]. D'autres connexions qui ont semblé être abandonnées étaient un effet secondaire d'un problème possible de NAT : des adresses IP et/ou

des ports étaient changés au cours d'une connexion. On a aussi observé d'autres cas de mauvais fonctionnement d'une implantation TCP.

Importance des résultats et travail qui reste à faire

La représentation FSM des connexions TCP permet d'identifier les acheminements de paquets dans un réseau qui ne sont pas conformes au comportement prévu du protocole. Les résultats montrent que le concept est viable et il encourage l'examen de l'algorithme utilisé strictement comme logiciel de détection des intrusions (IDS).

Nous aimerions étendre le modèle de façon à inclure la génération du trafic, c.-à-d. l'inclusion des anomalies dans les données sur le trafic généré. Le modèle pourrait aussi servir de base à un algorithme de détection des anomalies statistiques fondé sur les principes de Markov, où la probabilité de la validité de chaque connexion est calculée. L'analyse de l'évolution à plus long terme du protocole TCP est une autre application qui pourrait être intéressante.

J. Treurniet and J. H. Lefebvre; 2003; A Finite State Machine Model of TCP Connections in the Transport Layer; DRDC Ottawa TM 2003-139; R & D pour la défense Canada – Ottawa.

Table of contents

List of figures

List of tables

1 Introduction

The Transmission Control Protocol (TCP), which is the dominant protocol in use on the Internet today [1], contains a wide variety of inconsistencies in its implementation [2, 3, 4]. Research and analysis of Internet traffic has been performed in an attempt to understand what is really happening on the Internet [3, 5, 6, 7], as opposed to what is expected to occur based on the TCP Request For Comments (RFC) specifications [8].

Since its inception, the Internet has become widely available and we have become dependent on computers and networks. Network scans are often the first steps in an attack, the detection of which is easy to achieve if they are temporally fast, but more challenging if they are temporally slow. Work is underway to detect these scans by detecting anomalous packets and using clustering techniques to correlate the events [9], but traditional anomaly detection methods are known to suffer from a high degree of false positives.

"Strict anomaly detection" defines a set of permitted events and detects activity which represents exceptions to those events [10]. This idea is also discussed as "protocol anomaly detection" [11, 12]. In intrusion detection systems, protocol anomaly detection has been implemented at the application layer (see e.g. [13, 14, 15]). In the present work, strict anomaly detection is performed on TCP at the transport layer. We model a TCP connection as a Finite State Machine by defining the set of states that are necessary to describe the progression of a connection and the set of allowed events that bring on transitions between these states. Disallowed events and transitions are reported as anomalous.

A snapshot of TCP connection anomalies can impart information important to network management, such as unresponsive hosts and behaviours that lead to resource consumption, and to network security, particularly in the area of intrusion detection for scanning activity. Researchers interested in the behaviour of TCP on the Internet (e.g. [16]) can see how often and in what sense the Internet experiences anomalies in TCP implementations. Those involved in TCP traffic simulation/generation may wish to include traffic with these types of anomalies if the desired result is to reflect the actual use of the TCP protocol. The appeal of the model is in its simplicity and its utility in a wide range of applications.

2 Theory

The exchange of TCP packets between two hosts on the same ports forms the basis of a TCP connection. Since the exchange of packets follows a pattern defined by a finite set of rules, it is described well by a Finite State Machine (FSM) model [17].

This section explains the relevant details of the TCP protocol, and describes how we represent a TCP connection as an FSM, using the TCP flags as the attribute of packets that bring about transitions from one state to the next.

2.1 Transmission Control Protocol

The Transmission Control Protocol (TCP), defined in RFC 793 [8], is a connection-oriented, reliable protocol. A TCP packet has a header which includes source and destination IPs and ports, sequence numbers, acknowledgement number and most important to this discussion, includes flags that carry information important to the progression of the connection [8, 18]. The flags are shown in Table 1[1]. Only certain flag combinations are valid.

Flag	Meaning
U	Urgent pointer valid (data contained in packet is urgent)
A	Acknowledgement number valid (acknowledge receipt)
P	Push data (flush the buffer)
R	Reset connection
S	Synchronize sequence numbers (initiate connection)
F	No more data, finish connection

Table 1: *The TCP flag bits and their meanings.*

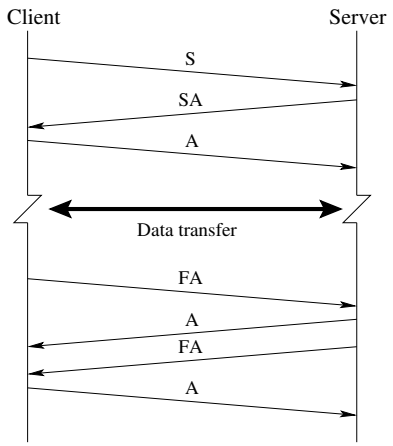

Figure 1: *Diagram of a "textbook" TCP connection.*

Figure 1 shows a typical TCP connection with graceful termination. The connection is "negotiated" by the two parties involved in what is known as the "TCP three-way handshake": the client requests a connection by sending a packet with a SYN flag (represented by the arrow labelled S); the server, if it chooses, agrees to open a connection with transmission of its own SYN packet (represented by the arrow labelled SA). Whenever a packet is sent, the receiver must acknowledge receipt with transmission of a packet with the ACK flag set so that the sender knows that the packet was received. These ACKs are often piggybacked on other packets to reduce network clutter, which is why the second SYN appears as a SYN-ACK. A packet acknowledging the second SYN completes the handshake and data transfer may begin (represented by the thick arrow as an exchange of packets in both directions).

[1] We do not include congestion control flags ECE and CWR as they do not affect the progression of the connection.

On closing, either the client or the server may request that the connection be ended with transmission of a FIN packet, and the other agrees to close the connection with transmission of its own FIN packet. In an ungraceful termination, the connection is "torn down" by transmission of a reset (R) from either the client or the server.

In practice, stages of a connection are only allowed to exist for a set amount of time, to free up resources that aren't being used. Table 2 shows the timeouts for a connection chosen for this model [18]. Most Berkeley-derived systems set a time limit of 75 seconds on the establishment of a new connection. The keepalive timer is a feature available in many TCP implementations. It is the amount of time that passes before a TCP will send a probe to an inactive connection, and is given in [18] as 2 hours. Many Berkeley-derived TCP implementations prevent an infinite wait for the arrival of the second FIN in a graceful termination by using a timeout of 10 minutes and 75 seconds. TIME_WAIT is defined as twice the maximum amount of time a packet may exist in the network before being discarded. This value, the only timeout defined as part of the protocol, is given in [8] as 4 minutes, but may be as low as 1 minute depending on the TCP implementation [18].

Timeout	Time
Establishment	75 seconds
Keepalive	2 hours
Waiting for second FIN	10 minutes 75 seconds
TIME_WAIT	4 minutes

Table 2: *Timeouts used to define the end of a connection.*

2.2 TCP Finite State Machine Model

In simple terms, an FSM progresses a system through a sequence of states via transitions from one state to the next. Transitions occur in response to events [17]. The traditional state transition diagram for TCP is a graphical representation of two separate but related finite state machines, one for the client and one for the server. We have simplified the traditional TCP state transition diagram to remove the differentiation of client and server.

The overall state of the connection is what is considered in this model. We define seven states for this model. The *Listen* state is the imaginary starting point for all connections. To establish the connection, the connection must complete the handshake by passing through two states, *Connection requested* and *Connection established*, at which point the connection may enter the *Data transfer* state. If the connection is terminated gracefully, it must negotiate the closing and pass through the two states *Closing* and *Closed*. If it is terminated abruptly, it may skip *Closing* and proceed directly to *Closed*. We introduce one further state, *Failure*, to represent that the connection has strayed from the protocol specification by attempting to access a state out of order or by introducing an illegal event. The states used in the TCP FSM model are summarised in Table 3.

A TCP connection progresses from state to state based on the information contained in the headers of the packets exchanged. An event in this FSM of a TCP connection is interpreted

State	Symbol	Description
Listen	L	All connections must start in a *Listen* state.
Connection requested	H_1	When the first SYN is sent, the connection is in the *Connection requested* state (Handshake, stage 1).
Connection established	H_2	When the SYN is acknowledged, the connection enters the *Connection established* state (Handshake, stage 2).
Data Transfer	X	When the handshake is complete and until the closing begins, the connection is in a *Data transfer* state.
Closing	C_1	If the connection is terminated gracefully, the *Closing* state is entered when one side has sent a FIN packet.
Closed	C_2	If the connection is terminated via a RST packet or the second FIN packet, the connection enters the *Closed* state and stays there.
Failure	$\oslash$	If an event or transition that is not allowed occurs, the connection enters the *Failure* state and stays there.

Table 3: *The states of the TCP FSM.*

as the combination of TCP flags in an arriving packet. We group the flags into categories as shown in Table 4 such that the flag combinations in a group cause the same state transition. We introduce the event label "Others" to handle the potential presence of bad flag combinations, e.g. the SF combination requests both to begin and end a connection. Such packets are either "crafted" or appear as a result of corruption of the packet.

Event label	Flag set	Event description
S	{S}	Request to open connection.
SA	{SA}	Agree to open connection.
A^{PU}	{A, PA, AU, PAU}	Acknowledgement of receipt.
F^{APU}	{F, FA, FP, FU, FPA, FPU, FAU, FPAU}	Request to close connection.
R^{APU}	{R, RA, RP, RU, RPA, RPU, RAU, RPAU}	Tear down connection.
Others	The set of all remaining flag combinations	Bad flags.

Table 4: *The TCP flag combinations used to define events in the FSM.*

Graphically, the progression of a connection exhibiting normal, expected behaviour as it is commonly seen on the Internet is represented in Figure 2. All connections begin in the *Listen* state. With an S event, the connection enters the *Connection requested* state, and an

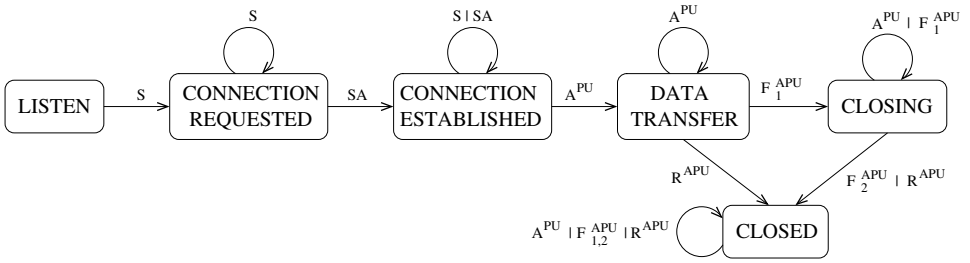

Figure 2: *Graphical representation of the TCP FSM. The events are the flag combinations of Table 4 and the states are as defined in Table 3. For clarity, the failure state is not shown.*

SA event brings the connection to the *Connection established* state. Since almost 10% of all SYN packets are retransmitted [2], we allow an S event to keep the connection in the *Connection requested* state, and likewise for S and SA events in the *Connection established* state. An A^{PU} event completes the handshake and takes the connection to the *Data transfer* state, and more A^{PU} events keep it in that state, while a F^{APU} or a R^{APU} event will initiate a closing sequence for the connection. A R^{APU} event takes the connection directly to the *Closed* state, while a F^{APU} event leads to *Closing* and then to *Closed* after a second F^{APU} event occurs. No event changes the state after it enters *Closed*, except "Others".

The flag combinations for F^{APU} are extended in this diagram with subscripts (1,2) to denote directionality. For a complete closing, one FIN must be sent by each party; the subscript 1 denotes a FIN received from one direction and the subscript 2 denotes a FIN received from the opposite direction.

Directional differentiation similar to the above is included for the SYN packets involved in the handshake. In a valid connection, the SYN must originate from the client and the SYN-ACK from the server. Since the symbols for these states are already different, we do not use subscripts.

In the analysis of a connection, some attempt must be made to differentiate between behaviour that is TCP and effects of the networks such as out-of-order packet delivery (due to variations in path) and packet replication [5]. To deal with these network effects, we allow an ACK packet to have no effect on the *Connection requested* state and a SYN-ACK packet to have no effect in the *Connection established* state.

Table 5 shows the transition table for the FSM. For example, if the connection was in the *Connection established* (H_2) state, and the next event was the arrival of a packet with PUSH and ACK (A^{PU}) flags, the subsequent state would be *Data transfer* (X). Note that the F_2^{APU} event cannot be applied until the F_1^{APU} event has occurred; this is denoted by the – symbol in the transition table. The *Failure* state is accessible from every state, and once it is reached, the system remains in that state (known as an absorbing state).

Allowed TCP behaviour that is indicative of improper activity is not considered to be valid in this model. While the sequences $\{S–R^{APU}\}$, $\{S–SA–R^{APU}\}$ and $\{S–SA–F^{APU}\}$ are tech-

State \ Event	S	SA	A^{PU}	F_1^{APU}	F_2^{APU}	R^{APU}	Other
L	H_1	$\oslash$	$\oslash$	$\oslash$	–	$\oslash$	$\oslash$
H_1	H_1	H_2	H_1	$\oslash$	–	$\oslash$	$\oslash$
H_2	H_2	H_2	X	$\oslash$	–	$\oslash$	$\oslash$
X	$\oslash$	$\oslash$	X	C_1	–	C_2	$\oslash$
C_1	$\oslash$	$\oslash$	C_1	C_1	C_2	C_2	$\oslash$
C_2	$\oslash$	$\oslash$	C_2	C_2	C_2	C_2	$\oslash$
$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$

Table 5: *The transition table for the TCP FSM. The states are as defined in Table 3 and the flag combinations are as defined in Table 4.*

nically allowed, they may indicate a scan. Similarly, timed-out connections could indicate a scan or an unresponsive host.

2.3 Detection of Anomalies

To detect anomalous behaviour using the TCP FSM, the sequence of events (flags) is taken as input for the transition table with initial state *Listen*. Transitions occur as a result of these events as per Table 5. If the final state of the sequence is not *Closed* (C_2), then the connection is flagged as anomalous. The notation we use here to describe an anomaly failure state is:

$$\oslash = \text{current state} \mid \text{event} \tag{1}$$

For streams that terminate in an allowed state other than *Closed*, we denote the event as "timeout" to signify that the stream has ended but the connection has not been closed.

As an example, Denial of Service (DoS) using a SYN flood will trigger many $H_2 \mid$ timeout events for one host. Naptha-type DoS [19], which exploit the Established and FIN_WAIT_1 states, would trigger multiple X $\mid$ timeout or $C_1 \mid$ timeout failures. A TCP half-open scan may appear as $H_1 \mid R^{APU}$, $H_1 \mid$ timeout, $H_2 \mid R^{APU}$ or $H_2 \mid$ timeout, depending on the network's configuration. Scans for OS fingerprinting such as those produced by QueSO [20] will also be detected as failures in the *Listen* and *Connection requested* states. The anomaly types associated with a QueSO scan are shown in Table 6.

3 Implementation of the Algorithm

To get an idea of the realistic application of the model, the anomaly detector was applied to unfiltered traffic collected at the external interface of 3 class B networks using tcpdump [21] via the SHADOW scripts [22]. Only the first 68 bytes of each packet were collected to obtain the full TCP header. At peak times, an hourly file contained on the order of 2 million packets.

Packet flags	Anomaly type
SYN	$H_1 \mid \{R^{APU}, \text{timeout}\}$
SYN+ACK	$L \mid SA$
FIN	$L \mid F^{APU}$
FIN+ACK	$L \mid F^{APU}$
SYN+FIN	bad flags
PSH	$L \mid A^{PU}$
SYN+1+2	$H_1 \mid \{R^{APU}, \text{timeout}\}$

Table 6: *The packets sent by the QueSO tool and the anomaly types that will detect each.*

The TCP FSM algorithm was written in MATLAB, using as libraries a suite of MATLAB functions developed at DRDC Ottawa for the analysis of network traffic data [23]. In practice, each packet was read sequentially from the raw traffic file. Each socket pair (port-IP pair) was assigned a stream holding the complete trace for that socket. When the connection was deemed complete by timeout (as per Table 2), the stream was removed. If the connection was not in the *Closed* state, it was output as an anomaly.

Since the timeout values in Table 2 are not always consistent for all TCP implementations, a new connection could begin before a old connection was removed. To account for these possibilities, if a SYN packet arrived after the handshake had been completed and it was a replayed packet, it was ignored. If it was not a replay, the current connection state was output, the stream was cleared and a new stream was started for the new connection. Therefore failures associated with SYNs appearing in the *Data transfer*, *Closing* or *Closed* states do not appear in the results of the analysis.

Note that in the first two and last two hours of analysis there will be "false positives" in the sense that failures will occur due to the truncation of the traffic data, i.e. streams that start or end in the middle of a connection. Consequently, one must extend the time period under examination by 2 hours at the beginning and at the end.

The SHADOW IDS runs a script to stop logging at the end of an hour, then runs a script to start a new log. This results in 2–3 seconds of dropped packets at the beginning of each hour. To account for the effect of the dropped packets at the beginning of each hour, anomalies were not reported under the following conditions:

$\{H_1, H_2, X, C_1\} \mid$ timeout: If the last packet in the connection is within 75 seconds of the hour start.

$L \mid \{SA, A, F, R\}$: If the first packet in the connection is within 75 seconds of the hour start.

$\{H_1, H_2\} \mid \{F, R\}$: If the first and last packets encompass the hour start within 5 minutes.

On average, this removed approximately 7% of the anomalies, mainly affecting the proportion of $X \mid$ timeout, $L \mid A^{PU}$, and $H_1 \mid F^{APU}$. This may introduce some false negatives, but the effect is expected to be negligible.

4 Results and Discussion

The result of applying the TCP FSM to the traffic data is discussed in this section. The results are interpreted from the points of view of network management, network security and research.

Each failure found within the data was investigated to determine the most probable causes. An inexhaustive list of these possible causes are shown in Table 7. Note that as discussed in Section 3, failure types $X \mid \{S, SA\}$, $C_1 \mid \{S, SA\}$ and $C_2 \mid \{S, SA\}$ do not appear in the results of the analysis.

The number of anomalies and percentage of anomalies determined from applying the TCP FSM are shown in Table 8 for data collected on August 21, 2000. This data shows that approximately 37% of the Internet TCP traffic does not obey the protocol on that day. This estimate includes scans and DoS backscatter. We can identfy by sheer volume a large-scale SYN scan and a bad flag scan, since the source host is the same and the destinations cover large subnets. If we remove these, the percentage of anomalous sessions is reduced to approximately 4%.

In the following sections, we discuss how the results can be interpreted for the areas of network management, network security and Internet traffic research.

4.1 Network Management

The health of the network can be monitored using the TCP FSM through analysis of the failures. Through comparison with a list of internal Internet servers to IPs associated with failure states due to timeouts, a network manager can be informed of the status of the network servers.

4.1.1 Unresponsive and non-existant hosts

Unresponsive and non-existant hosts fall into the $H_1 \mid R^{APU}$, $H_1 \mid$ timeout, $H_2 \mid R^{APU}$ and $H_2 \mid$ timeout categories. For $H_1 \mid R^{APU}$, the client sends SYNs and then sends a reset after it receives no response. For $H_2 \mid R^{APU}$, the client sends a SYN, the server replies with a SYN-ACK, and then the server resets the connection after it receives no response. In all of these cases, the cause may be a firewall rule preventing any packets destined to a disallowed ephemeral port. We observed all of the four failure categories in our data.

4.1.2 Delays

Delays in the networks can be seen when packets do not arrive in an appropriate amount of time, generating the anomalies $L \mid \{SA, A^{PU}, F^{APU}, R^{APU}\}$, $H_2 \mid R^{APU}$ and $X \mid$ timeout. For example, Table 9 shows a trace where four SYN packets were sent, the connection was established, and three SYN-ACKs appeared later. This trace showed itself as an $L \mid SA$ anomaly. Delays in SYN packets can also cause similar problems, as shown in

Failure	Anomaly Description(s)
L | SA	SYN-ACK scan‡
	Backscatter from DoS or scan with spoofed source†
	Delayed SA
	Spurious change in IP and/or port
L | A^{PU}	ACK scan
	Spurious changes in IP and/or port
	Delayed ACK (timeout)
L | F^{APU}	FIN scan†
	Delayed FIN
	Spurious change in IP and/or port
L | R^{APU}	RST scan†
	Backscatter from DoS or scan with spoofed source
	Delayed RST†
	Spurious change in IP and/or port
H_1 | F^{APU}	Malfunctioning TCP or application
H_1 | R^{APU}	TCP connect or half-open scan, port closed
	Benign attempted connection to closed port
	DoS†
	Client gives up on unresponsive or nonexistant host
H_1 | timeout	TCP connect or half-open scan with no response
	DoS†
	Unresponsive or nonexistant host
H_2 | F^{APU}	TCP connect scan, port open
	Closing piggybacked on handshake
H_2 | R^{APU}	TCP half-open scan, port open
	Unresponsive host - server sends R after timeout
	DoS†
	Firewall blocks ephemeral ports, SA never received
	Delayed SYN
H_2 | timeout	Unresponsive host
	Firewall blocks ephemeral ports, SA never received
	DoS†
X | timeout	Abandoned connection
	Client does not receive SA (delays, lost packets)
C_1 | timeout	Abandoned connection
Bad Flags	Scan
	Redundant flags (FIN and RST)
	Packet corruption

Table 7: *Examples of the anomalies that could be associated with each failure type.*

† Not seen in real traffic data.
‡ Possibly seen in real traffic data.

Anomaly	# total	% total	#(total-scans)	%(total-scans)
L | SA	228	0.04	228	0.03
L | A^{PU}	442	0.07	442	0.11
L | F^{APU}	679	0.11	679	0.17
L | R^{APU}	7398	1.20	7398	1.82
H_1 | timeout	18007	2.92	1980	0.49
H_1 | F^{APU}	2	0.00	2	0.00
H_1 | R^{APU}	3116	0.50	3116	0.77
H_2 | timeout	12	0.00	12	0.00
H_2 | F^{APU}	22	0.00	22	0.01
H_2 | R^{APU}	459	0.07	459	0.11
X | timeout	297	0.05	297	0.07
C_1 | timeout	1101	0.18	1101	0.27
Bad flags	194606	31.52	6	0.00
Normal	391127	63.34	391127	96.16
Total	617496	100.00	406760	100.00

Table 8: *The anomalies found in the August 21, 2000 data for 24 hours starting at midnight. Columns 4 and 5 show the data with the obvious large-scale SYN and bad flag scans removed.*

```
13:03:04.896645 A.42715 > B.110:  S 1299530124:1299530124(0) win 8760
13:03:08.393828 A.42715 > B.110:  S 1299530124:1299530124(0) win 8760
13:03:14.393888 A.42715 > B.110:  S 1299530124:1299530124(0) win 8760
13:03:26.394414 A.42715 > B.110:  S 1299530124:1299530124(0) win 8760
13:03:26.469582 B.110 > A.42715:  S 2899280544:2899280544(0) ack 1299530125 win 8760
13:03:26.470510 A.42715 > B.110:  .  ack 2899280545 win 8760
                  ⋮           ⋮           ⋮
13:03:27.536308 B.110 > A.42715:  P 2899280639:2899280648(9) ack 1299530162 win 8760
13:03:27.583752 A.42715 > B.110:  .  ack 2899280648 win 8760
13:03:27.604115 B.110 > A.42715:  S 1379570392:1379570392(0) ack 1299530125 win 8760
13:03:27.606111 A.42715 > B.110:  .  ack 2899280648 win 8760
13:03:27.726541 A.42715 > B.110:  P 1299530162:1299530166(4) ack 2899280648 win 8760
13:03:27.850403 B.110 > A.42715:  .  ack 1299530166 win 8760
13:03:27.851943 A.42715 > B.110:  P 1299530166:1299530168(2) ack 2899280648 win 8760
13:03:27.872141 B.110 > A.42715:  P 2899280648:2899280697(49) ack 1299530168 win 8760
13:03:27.872601 B.110 > A.42715:  F 2899280697:2899280697(0) ack 1299530168 win 8760
13:03:27.876398 A.42715 > B.110:  .  ack 2899280698 win 8760
13:03:27.876507 A.42715 > B.110:  F 1299530168:1299530168(0) ack 2899280698 win 8760
13:03:27.897295 B.110 > A.42715:  .  ack 1299530169 win 8760
13:03:53.340833 B.110 > A.42715:  S 1379570392:1379570392(0) ack 1299530125 win 8760
13:03:53.342853 A.42715 > B.110:  R 1299530125:1299530125(0) win 0
13:04:44.645843 B.110 > A.42715:  S 1379570392:1379570392(0) ack 1299530125 win 8760
13:04:44.648836 A.42715 > B.110:  R 1299530125:1299530125(0) win 0
```

Table 9: *A trace showing delayed handshake responses in a connection, which does not affect the progression of the connection.*

Table 10. In this case, four identical SYN packets are sent, one is answered with a SYN-ACK to establish a connection, but another is answered with a reset. Since the sequence numbers of the SYN packets were identical, the connection was torn down. This trace appeared as an $H_2 \mid R^{APU}$ anomaly.

```
02:36:07.560182 A.1511 > B.25:  S 416064000:416064000(0) win 32768
02:36:11.434188 A.1511 > B.25:  S 416064000:416064000(0) win 32768
02:36:21.474896 A.1511 > B.25:  S 416064000:416064000(0) win 32768
02:36:45.576086 A.1511 > B.25:  S 416064000:416064000(0) win 32768
02:36:45.673741 B.25 > A.1511:  S 3033524924:3033524924(0) ack 416064001 win 17520
02:36:45.675492 B.25 > A.1511:  R 3033524925:3033524925(0) ack 416064001 win 17520
02:36:45.682636 A.1511 > B.25:  .  ack 3033524925 win 32768
02:36:45.780013 B.25 > A.1511:  R 3033524925:3033524925(0) win 0
```

Table 10: *A trace showing a detrimental effect of network delays.*

4.1.3 Abandoned connections

Abandoned connections fall into the $X \mid \text{timeout}$ and $C_1 \mid \text{timeout}$ categories. Some connections are legitimately abandoned, however other connections that appear to have been abandoned are a result of misbehaviours, which can be seen to be related to the other anomalies. For example, the trace shown in Table 11 triggers both a $C_1 \mid \text{timeout}$ and an $L \mid A^{PU}$ failure. This trace was caused by a spurious change in the port and IP of a connection. The sequence numbers lead to the conclusion that this is in fact the same connection. A possible cause is a malfunctioning NAT device.

```
09:01:38.350627 A.38005 > x.y.z.23.80:  S 2773978047:2773978047(0) win 8760
09:01:41.842543 A.38005 > x.y.z.23.80:  S 2773978047:2773978047(0) win 8760
09:01:42.068782 x.y.z.23.80 > A.38005:  S 675782083:675782083(0) ack 2773978048 win 8760
09:01:42.071801 A.38005 > x.y.z.23.80:  .  ack 675782084 win 8760
09:01:42.076422 A.38005 > x.y.z.23.80:  P 2773978048:2773978303(255) ack 675782084 win 8760
09:01:42.077198 A.38005 > x.y.z.23.80:  P 2773978303:2773978463(160) ack 675782084 win 8760
09:01:42.334801 x.y.z.23.80 > A.38005:  .  ack 2773978463 win 8345
09:01:44.722807 A.38005 > x.y.z.23.80:  F 2773978463:2773978463(0) ack 675782084 win 8760
09:01:44.840595 x.y.z.23.80 > A.38005:  .  ack 2773978464 win 8345
09:01:47.346971 x.y.z.13.80 > A.4363:  P 675782084:675782369(285) ack 2773978464 win 8760
09:01:47.347015 x.y.z.13.80 > A.4363:  F 675782369:675782369(0) ack 2773978464 win 8760
09:01:47.348160 A.4363 > x.y.z.13.80:  R 2773978464:2773978464(0) win 0
09:01:47.348227 A.4363 > x.y.z.13.80:  R 2773978464:2773978464(0) win 0
```

Table 11: *A trace showing a sudden change in one of the ports and one of the IPs in a connection. This trace resulted in an $L \mid A^{PU}$ anomaly and a $C_1 \mid \text{timeout}$ anomaly.*

4.1.4 Malfunctioning TCP

An example of a malfunctioning TCP or application (anomaly $H_1 \mid F^{APU}$) is shown in Table 12. Here, host A sends a connection request to host B, and host B acknowledges receipt

of the packet but does not send its own SYN packet to complete the connection establishment. Host B then attempts to gracefully close a connection that has not been established with the FIN-ACK packets. The cause of the behaviour is unclear.

```
12:31:20.965783 A.37569 > B.80:  S 4117960417:4117960417(0) win 8760
12:31:24.513149 A.37569 > B.80:  S 4117960417:4117960417(0) win 8760
12:31:24.807301 B.80 > A.37569:  .  ack 4117960418 win 16384
12:31:30.513687 A.37569 > B.80:  S 4117960417:4117960417(0) win 8760
12:31:30.878542 B.80 > A.37569:  .  ack 4117960418 win 16384
12:31:42.613337 A.37569 > B.80:  S 4117960417:4117960417(0) win 8760
12:31:43.008129 B.80 > A.37569:  .  ack 4117960418 win 16384
12:31:51.290266 B.80 > A.37569:  F 3854453158:3854453158(0) ack 4117960418 win 16384
12:31:51.290268 B.80 > A.37569:  F 3854453158:3854453158(0) ack 4117960418 win 16384
12:31:52.417395 B.80 > A.37569:  F 3854453158:3854453158(0) ack 4117960418 win 16384
12:31:53.983493 B.80 > A.37569:  F 3854453158:3854453158(0) ack 4117960418 win 16384
12:31:57.045194 B.80 > A.37569:  F 3854453158:3854453158(0) ack 4117960418 win 16384
12:32:03.033479 B.80 > A.37569:  F 3854453158:3854453158(0) ack 4117960418 win 16384
12:32:06.628788 A.37569 > B.80:  S 4117960417:4117960417(0) win 8760
12:32:06.907044 B.80 > A.37569:  F 3854453158:3854453158(0) ack 4117960418 win 16384
12:32:18.585312 B.80 > A.37569:  F 3854453158:3854453158(0) ack 4117960418 win 16384
```

Table 12: *A trace showing an example detected as anomaly type* $H_1 \mid F^{APU}$.

4.1.5 Backscatter effects

When performing a DoS, the attacker will always spoof the source IP of the packets sent to addresses other than his own to redirect the responses. When performing a network scan, an attacker may attempt to hide his identity among a large number of other addresses by repeating the scan with spoofed source addresses.

Such methods will appear as anomalies $L \mid SA$ or $L \mid R^{APU}$ [24], and have no effect other than bandwidth consumption. Our results show a relatively high number of anomalies of this type, implying that the target IP range is often used as a spoofed source in DoS or scans. Backscatter was seen in the data as $L \mid R^{APU}$ but not as $L \mid SA$.

4.2 Security

Anomalies in TCP flag sequences can be indicators of malicious activity. In our data set, scans were the only threats identified by the FSM that targeted the experimental network. The types of scans identifed with the FSM were TCP connect scans, half-open scans, and stealth scans.

Scanning techniques are used in the reconnaissance stage of a network attack to gather information about the network and eventually exploit vulnerable systems. In particular, the attacker is looking for opened ports and wants to identify operating systems and applications. The most popular scanning techniques involve active probing, whereby an attacker sends a crafted packet and depending on the response, obtains a better understanding of the

potential weaknesses in the targeted system. Scans can also allow an attacker to determine the presence of filtering on the network [25].

The TCP FSM can also be used to verify policy implementations (i.e. filtering rules) and observe their impact on network behaviour.

4.2.1 TCP connect and half-open scans

The TCP connect scanning technique involves attempts to establish a full connection using the three-way TCP handshake. If the handshake is completed, the attacker then knows that the port on the targeted host is open. The attacker then closes or tears down the connection. When a three-way handshake cannot be completed, the attacker knows that the port is closed or that the targeted host is not accessible.

The typical flag sequences observed when the targeted port is opened are $\{\text{S–SA–A}^{\text{PU}}\text{–R}^{\text{APU}}\}$ or $\{\text{S–SA–A}^{\text{PU}}\text{–F}^{\text{APU}}\text{–A}^{\text{PU}}\text{–F}^{\text{APU}}\text{–A}^{\text{PU}}\}$. They resemble a typical TCP connection except for the fact that no data is exchanged and that they can often be correlated with other similar connection attempts. Since our TCP FSM is based solely on flag sequences, there is no distinction between TCP connect scans involving open ports and legitimate TCP connections. Hence, we did not expect this TCP FSM to catch TCP connect scans when the port of the targeted host is open. For the most part, this is what we observed. However, we discovered that the $\text{H}_2 \mid \text{F}^{\text{APU}}$ failure will catch a special case of the TCP connect scans involving open ports.

The $\text{H}_2 \mid \text{F}^{\text{APU}}$ failure triggered on 22 TCP connections (Table 8) made up of flag sequences similar to $\{\text{S–SA–F}^{\text{APU}}\text{–A}^{\text{PU}}\text{–F}^{\text{APU}}\text{–A}^{\text{PU}}\}$. In this case, the TCP three-way handshake was completed with a F^{APU} packet as opposed to an A^{PU} packet. This is perhaps a more efficient way of terminating the connection immediately after it has been established, through piggy-backing the closing on the handshake, and as mentioned in Section 2.2 is technically allowed. The presence of payload is a factor that may determine intent; 12 of our anomalies carried payload and 10 had none. For the former, the data exchanged appeared to be legitimate. For the latter, it is more likely that these were connect scans where the targeted port was open.

In the TCP half-open scanning technique, a crafted SYN packet is sent. If the targeted port is open, a SYN-ACK is received, and the handshake is not completed. A RST is sent to tear down the connection, triggering an $\text{H}_2 \mid \text{R}^{\text{APU}}$ failure. In contrast to the network management case, the client tears down the connection. We saw evidence of this type of traffic in our data.

When the targeted port is closed, a RST is the response from the targeted host, which will trigger an $\text{H}_1 \mid \text{R}^{\text{APU}}$ failure. If the host does not exist or the response is blocked (e.g. by a firewall), an $\text{H}_1 \mid \text{timeout}$ failure will result. In the traffic we saw a large-scale SYN scan that was blocked by our firewall which was responsible for 89% of the $\text{H}_1 \mid \text{timeout}$ events.

4.2.2 Stealth scans and inverse mapping

Stealth scans are scans that gather information about targeted sites while attempting to evade detection by firewalls and IDS. For example, some firewalls will be configured to block incoming packets with *only* the SYN flag set, hence a SYN-FIN packet will pass through undetected. In the TCP FSM, a SYN-FIN packet will trigger a bad flag failure. In fact, a large-scale SYN-FIN scan was detected in our traffic data and was responsible for 194600 bad flag failures.

We also detected a slow ACK scan, which is usually used to map out firewall rules. In particular, it can help determine whether a firewall is stateful or just a simple packet filter that blocks incoming SYN packets. In this scan, an ACK packet is sent (triggering an $\mathrm{L} \mid \mathrm{A}^{\mathrm{PU}}$ failure). If a RST comes back, the port is classified as "unfiltered". If nothing comes back (or if an ICMP destination unreachable is returned), the port is classified as "filtered".

Inverse mapping is a stealth scanning technique that makes use of crafted packets, including SYN-ACK, FIN, XMAS (FIN, PUSH and URG flags) and NULL (no flags set) flag combinations. RFC 793 [8] defines the required behaviour of TCP in response to these types of packets: if the targeted port is closed, any of the above packets will produce a RST; if the targeted port is open, there will be no response by the targeted host. If the targeted host does not exist or is unreachable, there will also be no TCP response. The above scans will trigger $\mathrm{L} \mid \mathrm{SA}$, $\mathrm{L} \mid \mathrm{F}^{\mathrm{APU}}$ and bad flag failures. RST scans are other inverse mapping techniques. The RST scan sends RST packets to a broad range of IPs, triggering $\mathrm{L} \mid \mathrm{R}^{\mathrm{APU}}$ failures. Routers will report any non-existant IPs through ICMP messages and there will be no response from existing hosts. We did not detect this type of activity in our traffic data.

4.2.3 Aggregating scanning events

Table 8 shows large-scale SYN and bad flag scans as $\mathrm{H}_1 \mid$ timeout and bad flag failures. These scans were performed very quickly and were readily identifiable, however scanning can be done more subtly by extending them over a longer period of time. Many IDS rely on the frequency of packet arrival to determine the presence of a scan (e.g. Snort [13] detects a scan if x packets arrive in y seconds), which overlooks slow (quiet) scans. Some automated scanning tools allow the user to specify how fast or slow the scan should occur, and send packets at periodic intervals [25]. It is also easy to write a simple scanner that randomizes time intervals between probes.

To mine the data for either fast or slow scans, one week of traffic data was analysed. We focussed on the anomaly types that may indicate scanning activity: $\mathrm{L} \mid \{\mathrm{SA}, \mathrm{A}^{\mathrm{PU}}, \mathrm{F}^{\mathrm{APU}}, \mathrm{R}^{\mathrm{APU}}\}$, $\mathrm{H}_1 \mid \{\mathrm{R}^{\mathrm{APU}}, \text{timeout}\}$, $\mathrm{H}_2 \mid \{\mathrm{F}^{\mathrm{APU}}, \mathrm{R}^{\mathrm{APU}}\}$ or bad flags. To find slow scans, the elapsed time between successive anomalies for each source IP in the anomaly set was calculated, and a histogram of the time differences was created to identify periodicity in the scanning attempts. The bin sizes were chosen to catch both short- and longer-term scans, with intervals defined by the set $\{0$, 1 min, 10 min, 1 hr, 5 hr, 12 hr, 1 day, 2 days, 5 days$\}$. For example, Table 13 shows the partial output of a program written to mine for

scans (`mine.m`). This particular result is for a slow SYN scan. The last line of the output is the counted bins, with edges defined as above. These bins are represented graphically in Figure 3. The fourth bin count tells us that 7 failures originated from the same IP with a time interval of between 1 and 5 hours. The results of Table 13 correspond to the traffic shown in Table 14.

```
Source IP: scanner
       Anomaly types:  2
Number of anomalies:  16
      Number of IPs:  15
     Number of ports:  1
Periodicity for anomaly type 2 HOST SCAN:
0    3    1    7    2    1    1    0    0
```

Table 13: *Partial output of* `mine.m` *for a slow SYN scan. Anomaly type 2 is* $H_1 \mid$ timeout. *See text for bin set (time between anomalies).*

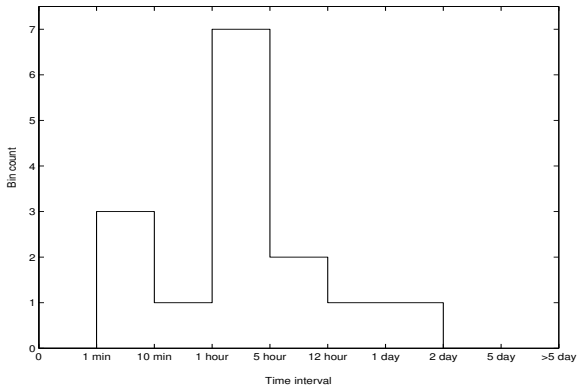

Figure 3: *Graphical representation of the program output in Table 13.*

4.2.4 Role of the Firewall

Monitoring complete connections, both incoming and outgoing, allows us to verify the role of the firewall and its configuration. Our traffic data showed evidence that the firewall was blocking specific ephemeral ports. The internal client sends a SYN, the external server sends a SYN-ACK directed to the blocked ephemeral port and does not complete the handshake. These failures appeared as $H_2 \mid R^{APU}$ and $H_2 \mid$ timeout. This has an impact on network performance, and is an indication that the firewall rules could be improved.

The behaviour of the firewall can also be used to verify that the site policy is being correctly enforced. Rejected connection attempts should correspond to the filtering rules. For example, we saw external clients attempting to connect to services that are not permitted by the policy. Such attempts triggered $H_1 \mid R^{APU}$ and $H_1 \mid$ timeout failures, as in the case of the large-scale SYN scan discussed in Section 4.2.1. The response of internal hosts to connection attempts is also important in assessing policy enforcement, e.g. for an $H_1 \mid R^{APU}$ or

```
08/18/2000 10:30:00.729317 scanner.4907 > my.net.78.26.80
08/18/2000 13:16:38.240864 scanner.3238 > my.net.98.56.80
08/20/2000 08:24:52.872996 scanner.3307 > my.net.22.186.80
08/20/2000 08:30:42.234438 scanner.3766 > my.net.15.112.80
08/20/2000 09:45:44.229834 scanner.2047 > my.net.7.188.80
08/20/2000 09:48:14.606899 scanner.2265 > my.net.130.171.80
08/20/2000 12:48:43.979432 scanner.1232 > my.net.216.199.80
08/21/2000 09:02:34.802720 scanner.2356 > my.net.23.92.80
08/21/2000 15:02:37.838110 scanner.4264 > my.net.9.59.80
08/21/2000 22:00:21.167523 scanner.2860 > my.net.22.194.80
08/21/2000 22:01:44.118879 scanner.2970 > my.net.113.177.80
08/22/2000 01:44:50.597606 scanner.1600 > my.net.113.177.80
08/22/2000 06:01:03.933332 scanner.2974 > my.net.244.159.80
08/22/2000 07:19:32.872888 scanner.1444 > my.net.233.237.80
08/22/2000 07:51:12.299813 scanner.4062 > my.net.227.250.80
08/22/2000 12:07:32.988795 scanner.1318 > my.net.214.68.80
```

Table 14: *The tcpdump output of the traffic data comprising the slow SYN scan (sanitized). The time between successive anomalies generates the bin values in Table 13.*

$H_2 \mid \text{timeout}$ shows that a response was made whereas $H_1 \mid \text{timeout}$ shows that no response was made.

4.2.5 Other threats

While the only detected threats directed towards our network were scans, we anticipate that other threats will be triggered by the TCP FSM model. For example, in a SYN-flood DoS attack, a client floods the targeted host with SYN packets on an open port at a rate which quickly consumes all of the victim's resources. The victim responds with SYN-ACK and waits for the ACK to complete the three-way TCP handshake. At some point, there are too many connections waiting to be established and the victim can no longer accept connection attempts, and if possible, simply send RSTs. Intuitively, this type of traffic should trigger $H_1 \mid R^{APU}$, $H_1 \mid \text{timeout}$, $H_2 \mid R^{APU}$ and $H_2 \mid \text{timeout}$ failures. While it was not our objective to see how well this FSM would work as an anomaly-based IDS, it would be interesting to do a complete test with a labelled dataset of exploits.

4.3 Research

The evolution of Internet use is an interesting trend to capture. Were we to repeat our calculations for current Internet usage, we could compare characteristics such as the number of connections that time out in the FIN_WAIT_2 (C_1) state ($C_1 \mid \text{timeout}$). These anomalies may occur as a result of poorly implemented TCP or in web client applications [26], or they may be caused by a sudden change in port or IP, as discussed in Section 4.1.3. Our results from August 21, 2000 show that although the majority of these anomalies occur on HTTP ports, ftp, pop and imap, among others, are also present. Further investigation may tell us whether the problem lies with the OS implementation of TCP, or with the applications using it.

4.4 False Positives

By definition, all anomalies found using strict anomaly detection are anomalous and therefore there is no false positive rate to consider [10]. However, if we interpret false positives as more of a level of interest, we can discuss how, depending on the point of view of the user, the failure events of interest differ. For example, an $H_1 \mid \text{timeout}$ anomaly that is part of a scan is of great concern to one involved in network security, but is of less importance to a network manager. If the anomaly stemmed from a link outage, it becomes of greater concern to the network manager. For researchers interested in determining whether the TCP protocol is strictly being followed, all events are of interest. Minor modifications to the model can yield different results; for example the frequency of reset terminated connections is important to some [27, 28], and could be obtained by interpreting a reset packet as a failure and counting the failures of that type.

It is difficult to determine the intent of the user who sends a SYN and receives a RST in response (failure $H_1 \mid R^{APU}$). An attacker would use this information with the intent of finding and exploiting a vulnerable system. In the benign case, some error is made and the client discovers that the targeted port is not listening. For the $H_2 \mid F^{APU}$ failure, the intent may be a TCP connect scan, a legitimate packet with data piggybacked on the FIN packet or an application that just had nothing to say.

Very long sessions appear as anomalies in our traffic analysis, observed as $L \mid A^{PU}$ and $X \mid \text{timeout}$ failures. Such sessions have a greater chance of having the handshake occur before listening started, or the closing occur after listening stopped. The period of time where the SHADOW scripts stop and start are also an experimental complication and a potential source of false positives (Section 3).

In the process of analyzing the anomalous TCP connections, we found evidence of packets being dropped by the sniffer. In these cases, the traces showed TCP connections that were unimpaired by the lack of critical events (flag transitions). For example, in one case, a TCP connection was established without the exchange of a SYN-ACK packet, triggering a $H_1 \mid F^{APU}$ failure, while data was exchanged and the connection closed normally. This implies that the SYN-ACK packet was missed by the sniffer. Hence some of the anomalous TCP connections were caused by this artefact in the dataset. They were not reported in Tables 7 or 8 since they were not related to a network management or security issue.

Other potential experimental complications include additions, resequencing (in which the packet sniffer alters the ordering of the packets, possibly due to 2 streams in the sniffer, one outgoing, one incoming) and timing [2]. There is also a vantage point complication which may arise from the location of the packet sniffer on the network. We collected our data at a point outside of the firewall, and as a result for failure $H_1 \mid \text{timeout}$, we cannot be sure whether the response to the SYN was blocked or the SYN itself was blocked.

5 Conclusions and Future Work

The finite state machine representation of TCP connections provides a means of examining packet flows on a network that do not conform to the protocol. The model, based entirely on TCP flag transitions, has been shown to successfully detect TCP-based scanning activity, delays, outages, misconfigurations and other unexpected TCP behaviour. It also allows us to examine TCP's real behaviour on the Internet.

We have shown that slow scans can be identified through mining of the data. Existing correlation engines such as Spice [9] could be used to correlate the events found through application of this model. An important advantage to detecting scans with the TCP FSM method is that the failure type identifies whether a response was made.

A successful full connect scan cannot be detected with this method because the flag pattern obeys the protocol specification. There are properties of the full connect scan that can be used in this case; since data is not sent by the client in this scan, and the session remains in the *Data transfer* state for only a short time, program modification to track these properties should address this problem. Alternatively, the model could be modified to allow a transition from the *Connection established* state to the *Closing* state via a FIN-PSH-ACK packet, since the PSH bit implies that data is being carried.

It is important that future empirical results using the TCP FSM do not suffer from dropped packets. False positives that occur as a result of dropped packets from hourly data sets can be rectified by real-time detection.

For trend analysis, the data must be averaged over a longer timescale or over a wide variety of networks, as capturing one day on one network can lead to biased results due to scanning activity or repeated connection attempts to one host.

Traffic generators that simulate TCP [29, 30] are mainly based on the RFC specifications. Those who generate traffic may wish to include an option for TCP traffic that is not officially allowed but is nevertheless present on the Internet. Using the proportionalities of anomalies found by implementing the TCP FSM, one can introduce failures using a Markov chain [31] approach. As well as a mechanism for traffic generation, Markov models may provide the capability of assigning a probability that a connection is valid. This may be the basis of a statistical anomaly detection engine.

The TCP FSM cannot be used alone as an intrusion detection system since it does not consider other factors such as content, however in conjunction with misuse detectors it may provide a more complete picture of what is happening on the network.

While it was not our objective to see how well this FSM would work as an anomaly-based IDS, it would be interesting to do a complete test with a labelled dataset of exploits. Our next step is to test the method more extensively using a set of attack traffic data, work that is currently in progress at DRDC.

References

1. Williamson, Carey (2001). Internet Traffic Measurement. *IEEE Internet Computing*, **5**(6), 70–74.

2. Paxson, V. (1997). Automated Packet Trace Analysis of TCP Implementations. In Steenstrup, M., (Ed.), *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 167–179. Cannes, France: ACM Press.

3. Padhye, J. and Floyd, S. (2001). On Inferring TCP Behavior. In *Proceedings of the ACM SIGCOMM '01 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 287–298. San Diego, California, United States: ACM Press.

4. Guha, B. and Mukherjee, B. (1997). Network security via reverse engineering of TCP code: Vulnerability analysis and proposed solutions. *IEEE Network*, **11**(4), 40–49.

5. Paxson, V. (1997). End-to-End Internet Packet Dynamics. In Steenstrup, M., (Ed.), *Proceedings of the ACM SIGCOMM '97 conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 139–154. Cannes, France: ACM Press.

6. Paxson, V. and Floyd, S. (1997). Why We Don't Know How to Simulate the Internet. In *Proceedings of the 29th Conference on Winter Simulation*, pp. 1037–1044. Atlanta, Georgia, United States: ACM Press.

7. Smith, F. D., Hernández-Campos, F., Jeffay, K., and Ott, D. (2001). What TCP/IP Protocol Headers Can Tell Us About the Web. In Vernon, M., (Ed.), *Proceedings of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pp. 245–256. ACM Press.

8. Postel, J. (1981). RFC 793: Transmission Control Protocol. (Online). http://www.rfc-editor.org/rfc/rfc793.txt.

9. Staniford, S., Hoagland, J., and McAlerney, J. (2002). Practical Automated Detection of Stealthy Portscans. *Journal of Computer Security*, **10**(1/2), 105–136.

10. Sasha/Beetle (2000). A Strict Anomoly Detection Model for IDS. *Phrack*, **10**(56).

11. Das, K. (2002). Protocol Anomaly Detection for Network-based Intrusion Detection (Online). SANS Institute. `http://www.sans.org/rr/intrusion/anomaly.php` (Jan. 2003).

12. Lemonnier, E. (2001). Protocol Anomaly Detection in Network-based IDSs (Online). E. Lemonnier. `http://erwan.lemonnier.free.fr/exjobb/report/protocol_anomaly_detection.pdf` (Jan. 2003).

13. Roesch, Martin (2003). Snort (Online). Martin Roesch. http://www.snort.org (Sept. 2003).

14. Enterprise Protection (Online). Internet Security Systems Inc.. http://www.iss.net/products_services/enterprise_protection/rsnetwork/guard.php (Apr. 2003).

15. Hernacki, B. (2003). Symantec Intrusion Detection Systems: Defining Protocol Anomaly Detection (Online). Symantec Corp.. http://www.symantec.com (Apr. 2003).

16. End-to-end Interest Maillist (Online). Postel Center for Experimental Networking. http://www.postel.org/end-to-end/index.html (Mar. 2003).

17. Hennie, F. (1968). Finite-State Models for Logical Machines, John Wiley & Sons.

18. Stevens, W. Richard (1994). TCP/IP Illustrated, Volume 1: The Protocols, Indianapolis, IN: Addison Wesley.

19. Smith, N. J. (Feb. 2001). Choking on Naptha: TCP/IP Network Denial of Service Vulnerabilities (Online). SANS Institute. http://www.giac.org/practical/Nicholas_J_Smith_GCIH.html (Jan. 2003).

20. Jordi Murgo (savage@apolstols.org). Index of /pub/tools/unix/scanners/queso (Online). CERIAS Security Archive. http://ftp.cerias.purdue.edu/pub/tools/unix/scanners/queso (Sept. 2003).

21. Jacobson, V., Leres, C., and McCanne, S. (Dec. 2002). TCPdump (Online). Lawrence Berkeley National Laboratory. http://www.tcpdump.org (Jan. 2003).

22. Naval Surface Warfare Center – Dahlgren Lab (2001). NSWC SHADOW Index (Online). U.S. Navy. http://www.nswc.navy.mil/ISSEC/CID (Jan. 2003).

23. Gregoire, M. and Lefebvre, J. H. (in preparation). The Network Traffic Analysis Toolbox. Technical Report. DRDC Ottawa.

24. Moore, D., Voelker, G. M., and Savage, S. (2001). Inferring Internet Denial-of-Service Activity. In *Proceedings of the 10th USENIX Security Symposium*, pp. 9–22. Washington, DC, United States: USENIX Association.

25. Fyodor (Feb. 2003). NMAP (Online). Insecure.org. http://www.insecure.org/nmap (Apr. 2003).

26. Apache HTTP Server Documentation Project. Connections in FIN_WAIT_2 and Apache (Online). Apache HTTP Server Version 2.0. http://httpd.apache.org/docs-2.0/misc/fin_wait_2.html (Mar. 2003).

27. End-to-end Interest Maillist. Frequency of RST terminated connections (Online). USC Information Sciences Institute. ftp://ftp.isi.edu/end2end/end2end-interest-1997.mail (Mar. 2003).

28. Floyd, S. (2002). Inappropriate TCP Resets Considered Harmful. (work in progress). draft-floyd-tcp-reset-03.txt.

29. McKenney, P.E., Lee, D.Y., and Denny, B.A. (Jan. 2002). TG Traffic Generator (Online). SRI International and USC/ISI PCEN. `http://www.postel.org/tg/tg2002.pdf` (Sept. 2003).

30. Lui, H. (Apr. 2000). SSFNET TCP Simulation Analysis by tcpanaly (Online). SSF Research Network. `http://www.ssfnet.org/Papers/tcpanaly.pdf` (Mar. 2003).

31. Ross, S. (1988). A First Course in Probability, New York, NY: Macmillan Publishing Company.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence R&D Canada - Ottawa
Department of National Defence
Ottawa, ON Canada K1A 0Z4

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

A Finite State Machine Model of TCP Connections in the Transport Layer (U)

4. AUTHORS (Last name, first name, middle initial)

Treurniet, Joanne R. and Lefebvre, Julie H.

5. DATE OF PUBLICATION (month and year of publication of document)	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (total cited in document)
October 2003	36	31

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

DRDC Ottawa NIO section

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)
15bf26	

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)
DRDC Ottawa TM 2003-139	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(x) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Full unlimited announcement

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

Finite state machines can be used to detect anomalous behaviour in TCP traffic by describing the progression of a connection through states as a result of events based on header flags. The method was applied to real traffic to understand its realistic use and it was found that for the time period analysed here, on the order of 37% of TCP connections do not follow the TCP protocol specifications. The majority of these are a result of malicious activity, and approximately 4% are due to benign anomalies such as unresponsive hosts and misconfigurations. The method may be applied as a network security measure, as a network management tool or as a research tool for the study of TCP behaviour on the Internet.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Finite State Machine, FSM, anomaly detection, intrusion detection, TCP, network security, network management

Defence R&D Canada

Canada's leader in defence
and national security R&D

R & D pour la défense Canada

Chef de file au Canada en R & D
pour la défense et la sécurité nationale

www.drdc-rddc.gc.ca