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

DEFENCE R&D DÉFENSE

A Finite State Machine Algorithm for Detecting TCP Anomalies

An Examination of the 1999 DARPA Intrusion Detection Evaluation Data Set

J. Treurniet

Defence R&D Canada – Ottawa
TECHNICAL MEMORANDUM
DRDC Ottawa TM 2005-168
November 2005

Canada

A Finite State Machine Algorithm for Detecting TCP Anomalies

An Examination of the 1999 DARPA Intrusion Detection Evaluation Data Set

J. Treurniet

Defence R&D Canada – Ottawa

DRDC Ottawa TM 2005-168

November 2005

Author

Original signed by J. Treurniet

J. Treurniet

Approved by

Original signed by J. Lefebvre

J. Lefebvre
Head/Network Information Operations Section

Approved for release by

Original signed by A. Ashley

A. Ashley
Head/Document Review Panel

Abstract

The Transmission Control Protocol (TCP) is a well-defined protocol, and as such, a finite state machine can be defined to reflect the progression of a TCP connection, including deviations from the protocol standard. The 1999 DARPA Intrusion Detection Evaluation (IDE) data set was used to test the model. It was found that the model effectively detects implementations of TCP that do not follow the protocol standard and network events such as loss of availability. The model was then used to analyse the TCP anomalies of an operational data set, in which several instances of scanning activity were also successfully detected. A comparison of the anomalies found in the Week 1 DARPA IDE data set with those found in the operational data showed that the behaviour of the simulated TCP traffic does not contain the variations found in an operational setting.

Résumé

TCP (Transmission Control Protocol) est un protocole bien défini et, par conséquent, on peut construire une machine à états finis pour représenter l'évolution d'une connexion TCP, y compris les écarts par rapport au protocole standard. L'ensemble de données IDE (Intrusion Detection Evaluation ou évaluation de la détection des intrusions) DARPA 1999 a été utilisé pour tester le modèle. On a constaté que ce dernier permettait effectivement de détecter les implémentations de TCP qui ne respectaient pas le protocole standard, ainsi que les événements réseau comme la perte de la disponibilité. On s'est ensuite servi du modèle pour analyser les anomalies, par rapport à TCP, d'un ensemble de données opérationnelles, dans lequel plusieurs cas d'activités de scannage ont également été détectés avec succès. Une comparaison des anomalies trouvées dans l'ensemble de données Week 1 DARPA IDE et de l'ensemble de données opérationnelles révèle que le comportement du trafic TCP simulé ne prend pas en compte les variations que l'on constate dans un contexte opérationnel.

This page intentionally left blank.

Executive summary

A Finite State Machine Algorithm for Detecting TCP Anomalies

J. Treurniet; DRDC Ottawa TM 2005-168; Defence R&D Canada – Ottawa; November 2005.

Implementations of the Transmission Control Protocol (TCP) contain a wide variety of inconsistencies. Research and analysis of Internet traffic has been performed in an attempt to understand what is really happening on the Internet, as opposed to what is expected to occur based on the TCP Request For Comments (RFC) specifications. The RFC defines a set of rules that guide the packet exchange within a connection. It follows that a TCP connection can be modelled using a finite state machine (FSM) by defining the set of states that are necessary to describe the progression of a connection and the set of allowed events that induce transitions between these states. Disallowed events and transitions can then be reported as anomalies.

The algorithm for this TCP FSM model was tested first with operational data. However, it was quickly realised that an undocumented data source was insufficient to test its application to the detection of network security events. In the field of network security, the documented data source that has become the standard is the 1999 DARPA Intrusion Detection Evaluation (IDE) data set. This simulated data was created with the intent to benchmark intrusion detection systems, but is also useful for the testing of new network security methodologies.

The TCP FSM model was tested using the "Week 1" portion of the 1999 DARPA IDE data set, which is known to be free of malicious activity. It was found that the model effectively detects implementations of TCP that do not follow the protocol standard, and network events such as loss of availability. Investigation of the anomalies showed that the DARPA IDE data contains an undocumented network event: the data for the morning of March 4, 1999 contains repeated rejected connection attempts to external web servers. Dropped packets were also confirmed to exist within the data.

The model was then used to locate the TCP anomalies in an operational data set. Investigation of the anomalies showed a number of TCP implementations that deviate from the protocol standard. Several instances of scanning activity were found, some of which were temporally slow and would not be detected by conventional methods.

The comparison of the anomalies found in the two data sets shows that the TCP behaviour in the DARPA IDE data does not completely reflect the real behaviour of TCP traffic on the Internet. The DARPA IDE data set is lacking in "Internet crud", which arises partially due to stray packets resulting from attacks on other networks,

the variations in TCP implementations, and to a small degree, corrupted packets.

The effort required to simulate "normal" background traffic would be substantial. The definition of "normal" itself is subjective, and may differ greatly from one day to the next. For our purposes, the DARPA IDE data sets are sufficient, as the data certainly contains a subset of actual Internet behaviour, and it has been concluded that the effort required to generate "normal" background traffic would outweigh the benefits to our research program.

In comparing the two data sets, the assumption was made that network scanning was the only malicious activity generating anomalies on the operational network. Scans were detected by grouping anomalies and manually investigating each group. In future, an automation of this process will be developed, emphasizing the detection of slow scans.

Sommaire

A Finite State Machine Algorithm for Detecting TCP Anomalies

J. Treurniet; DRDC Ottawa TM 2005-168; R & D pour la défense Canada – Ottawa; novembre 2005.

Les implémentations de TCP (Transmission Control Protocol) comportent une grande variété d'incohérences. On a effectué des recherches à ce sujet et analysé le trafic Internet dans le but de tenter de comprendre ce qui se passe réellement dans le réseau, par rapport à ce qui devrait se passer selon la RFC (Request For Comments) TCP. Cette RFC définit un ensemble de règles qui régissent les échanges de paquets à l'intérieur d'une connexion. Il s'ensuit qu'on peut modéliser une connexion TCP au moyen d'une machine à états finis, en définissant l'ensemble des états qui sont nécessaires pour décrire l'évolution de la connexion, et l'ensemble des événements autorisés qui induisent des transitions entre ces états. Les transitions et les événements interdits peuvent alors être signalés à titre d'anomalies.

L'algorithme correspondant à cette machine à états finis modélisant TCP a été testé tout d'abord avec des données opérationnelles. Cependant, on s'est rapidement rendu compte qu'une source de données non documentée était insuffisante pour tester l'application de cet algorithme à la détection des événements intéressant la sécurité du réseau.

Dans le domaine de la sécurité des réseaux, l'ensemble de données IDE (Intrusion Detection Evaluation) DARPA 1999 est devenu la source de données documentée standard. Ces données de simulation ont été créées dans le but d'évaluer des systèmes de détection d'intrusion, mais elles se révèlent également utiles pour le test des nouvelles méthodes de sécurisation des réseaux.

La machine à états finis qui modélise TCP a été testée au moyen de la partie "Week 1" de l'ensemble de données IDE DARPA 1999, dont on sait qu'elle ne comporte pas de codes malveillants. On a constaté que ce modèle permettait effectivement de détecter les implémentations de TCP qui ne respectent pas le protocole standard, ainsi que les événements réseau comme les pertes de disponibilité. L'étude des anomalies a révélé que l'ensemble DARPA IDE comportait un événement réseau non documenté : les données correspondant à la matinée du 4 mars 1999 incluaient des tentatives répétées de connexion à des serveurs Web externes, rejetées. On a également confirmé qu'il existait dans cet ensemble de données des paquets abandonnés.

On s'est ensuite servi de ce modèle pour repérer les anomalies par rapport à TCP dans un ensemble de données opérationnelles. L'étude de ces anomalies a révélé qu'un

certain nombre d'implémentations de TCP s'écartaient du protocole standard. On a constaté plusieurs cas de scannage, certains balayages temporairement lents ne pouvant pas être détectés par les méthodes classiques.

La comparaison des anomalies découvertes dans les deux ensembles de données a révélé que le comportement de TCP dans l'ensemble de données DARPA IDE ne reflétait pas fidèlement le comportement véritable du trafic TCP sur l'Internet. L'ensemble de données DARPA IDE ne comportait pas les "anomalies ou irrégularités" que l'on retrouve sur l'Internet, et qui résultent en partie de paquets isolés associés à des attaques sur d'autres réseaux, à des variantes dans les implémentations de TCP et, à un degré moindre, à des paquets corrompus.

Le travail requis pour simuler le trafic de fond "normal" serait considérable. Le mot "normal" lui même admet une définition subjective, qui peut varier considérablement d'un jour à l'autre. Pour nos fins, les ensembles de données DARPA IDE sont suffisants, puisqu'ils intègrent un sous ensemble permettant de simuler le comportement réel de l'Internet, et on a conclu que le travail nécessaire pour produire un trafic de fond "normal" ne serait pas compensé par les avantages, dans notre programme de recherche.

Dans la comparaison des deux ensembles de données, on a fait l'hypothèse que le scannage du réseau constituait la seule activité malveillante produisant des anomalies dans un réseau opérationnel. Ces scannages ont été détectés en groupant ensemble les anomalies et en examinant manuellement chaque groupe. Ce processus sera ultérieurement automatisé afin de permettre de mieux détecter les balayages lents.

Table of contents

1 Introduction

It has been shown that the Transmission Control Protocol (TCP) contains a wide variety of inconsistencies in its implementation [1–3]. Research and analysis of Internet traffic has been performed in an attempt to understand what is really happening on the Internet [2, 4–6], as opposed to what is expected to occur based on the TCP Request For Comments (RFC) specifications [7]. The RFC defines a set of rules that guide the packet exchange within a connection. It follows that a TCP connection can be modelled using a finite state machine (FSM) [8] by defining the set of states that are necessary to describe the progression of a connection and the set of allowed events that induce transitions between these states. Disallowed events and transitions can then be reported as anomalies.

The purpose of this document is to report on the effectiveness of the TCP FSM in the detection of TCP anomalies. The algorithm for this TCP FSM model was tested first with operational data [9]. However, it was quickly realised that an undocumented data source was insufficient to test the model's application to the detection of network security events. In the field of network security, the documented data source that has become the standard is the 1999 DARPA Intrusion Detection Evaluation (IDE) [10–12] data set. This simulated data was created with the intent to benchmark intrusion detection systems, but is also useful for the testing of new network security methodologies.

In the present work, the TCP FSM model is tested using the "Week 1" portion of the 1999 DARPA IDE data set, which is known to be free of malicious activity. The model is then used to locate the TCP anomalies in a data set obtained from an operational network. The sources of the anomalies are discussed and the differences between the two data sets are presented. Similar work has been presented in [13, 14]. The results augment the documentation of the data set and are of interest to those using the attack-free DARPA IDE data set.

2 TCP Finite State Machine Model

The exchange of TCP packets between two hosts on the same ports forms the basis of a TCP connection. The TCP FSM models the progression of a connection through a sequence of states (Table 1) via transitions from one state to the next. Transitions occur in response to events, which in this model are related to the combinations of TCP flags in an arriving packet. The flags are grouped into categories as shown in Table 2 such that the flag combinations in a group cause the same state transition.

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

Table 1: *The states of the TCP FSM.*

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

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.
FA^{PU}	{FA, FPA, FAU, FPAU}	Request to close connection.
R^{APU}	{R, RA, RP, RU, RPA, RPU, RAU, RPAU}	Tear down connection.
Others	All other flag combinations	Bad flags.

Table 3: *The transition table for the TCP FSM. The states are as defined in Table 1 and the flag combinations are as defined in Table 2.*

State \ Event	S	SA	A^{PU}	FA_1^{PU}	FA_2^{PU}	R^{APU}	Other
L	H_1	⊘	⊘	⊘	–	⊘	⊘
H_1	H_1	H_2	H_1	⊘	–	⊘	⊘
H_2	H_2	H_2	X	⊘	–	⊘	⊘
X	⊘	⊘	X	C_1	–	C_2	⊘
C_1	⊘	⊘	C_1	C_1	C_2	C_2	⊘
C_2	⊘	⊘	C_2	C_2	C_2	C_2	⊘
⊘	⊘	⊘	⊘	⊘	⊘	⊘	⊘

as a result of these events as per the TCP FSM transition table, reproduced from [9] in Table 3. If the final state of the sequence is not *Closed* (C_2), then the connection is flagged as anomalous. The notation used 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*, the event is denoted as "timeout" to signify that the stream has ended but the connection has not been closed. Further details of the model can be found in [9].

3 Implementation

The TCP FSM algorithm was written in MATLAB, using a suite of MATLAB functions developed at DRDC Ottawa for the analysis of network traffic data [15]. 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, the stream was cleared from memory and the terminal state of the connection was output. If that state was a failure state, the type of failure was indicated as per (1).

Since the timeout values in Table 4 are not consistent for all TCP implementations, a new connection could begin before an old connection was removed. To account for this possibility, 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.

Some special considerations had to be made in the analysis of the operational network data. This data was collected using tcpdump [16] via SHADOW scripts [17]. The

Table 4: *Timeout values used in this implementation of the model.*

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

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 discounted under the following conditions:

$\{H_1, H_2\}$ | timeout: if the last packet in the connection arrived within 75 seconds of the end of the hour;
$\{X, C_1\}$ | timeout: if the first packet in the connection arrived within 75 seconds of the start of the hour;
L | {SA, A} and H_1 |FA^{PU} : if the first packet in the connection arrived within 75 seconds of the start of the hour and the traffic, excluding RSTs, was bi-directional;
L | $\{FA^{PU}, R^{APU}\}$: if the first packet in the connection arrived within 10 seconds of the start of the hour;
$\{H_1$ |R^{APU} and H_2 |$\{FA^{PU}, R^{APU}\}$: if the first and last packets bracket the hour start.

The 75 second time interval above comes from the connection establishment timeout. This removed 0.60% of the anomalies, mainly affecting the proportion of X | timeout, L | A^{PU}, H_2 | timeout, L | FA^{PU} and L | SA. This may introduce some false negatives, but the effect is expected to be negligible.

To account for failures that would occur due to truncation of the operational traffic data, *i.e.* streams that start or end in the middle of a connection, the time period under examination was extended by 2 hours at the beginning and again at the end.

The DARPA IDE data set did not suffer from either of the above implementation issues.

4 Results and Discussion

The result of applying the TCP FSM to the traffic data is discussed in this section. Causes of the various anomalies include, but are not limited to, network delays, unresponsive hosts, abandoned connections, and in the operational traffic, scans and denial of service (DoS). Note that as discussed in Section 3, failure types X |{S, SA}, C_1|{S, SA} and C_2|{S, SA} do not appear in the results of the analysis.

4.1 DARPA IDE Data Set

The anomalies found in the "Week 1" traffic from the 1999 DARPA IDE data set [10] are shown in Table 5. Obviously, since the traffic is known to be attack-free, there are no scans, DoS or backscatter effects.

Table 5: *The terminal state for all TCP connections in the Week 1 traffic from the 1999 DARPA Intrusion Detection Evaluation. Of the 224351 connections, 99.43% were normal and 0.57% were anomalous.*

Terminal State	# anomaly	% anomaly
L | SA	5	0.39
L | A^{PU}	124	9.69
L | FA^{PU}	10	0.78
L | R^{APU}	0	0
H_1 | timeout	29	2.27
H_1 | FA^{PU}	1	0.08
H_1 | R^{APU}	868	67.81
H_2 | timeout	0	0
H_2 | FA^{PU}	0	0
H_2 | R^{APU}	9	0.70
X | timeout	104	8.13
C_1 | timeout	130	10.16
Bad flags	0	0

When the DARPA IDE traffic was collected, *tcpdump* did not indicate that any packets were dropped [12]. However, investigation of the L | SA anomalies showed that although there is no SYN packet in these connection traces, the connection continues as if there had been for all anomalies of this type. Similarly, 33 of the L | A^{PU} anomalies were due to missing both the SYN and SYN-ACK packets in the handshake, and the H_1 | FA^{PU} anomaly is due to a missing SYN-ACK packet, with the connection continuing on as normal. This shows that there are in fact dropped packets in the 1999 DARPA IDE traffic data. It is unclear why there is a larger proportion of cases where both the SYN and SYN-ACK packets were dropped.

4.1.1 Network Delays

In 10 cases a C_1 | timeout anomaly followed by a L | FA^{PU} anomaly is caused by the second FIN packet arriving after the 10 minutes 75 second timeout. The response to the second FIN is a connection reset. Table 6 shows a sample trace of this type of activity. In 76 cases, a TCP connection is improperly terminated, creating a C_1 | timeout anomaly followed by a L | A^{PU} anomaly. The second FIN packet is

never sent, but after the connection has timed out, the server sends an ACK packet, to which the response is a reset. The majority of these instances were IRC connections.

Table 6: *An example trace where C_1 | timeout is followed by a L | A^{PU}.*

```
11:25:06.931397 A.13520 > B.http: S 1901766325:1901766325(0) win 512
11:25:06.935277 B.http > A.13520: S 4185883185:4185883185(0) ack 1901766326 win 32736
11:25:06.935957 A.13520 > B.http: . ack 4185883186 win 32120
11:25:06.936513 A.13520 > B.http: P 1901766326:1901766532(206) ack 4185883186 win 32120
11:25:06.954997 B.http > A.13520: . ack 1901766532 win 32736
11:25:06.957535 B.http > A.13520: P 4185883186:4185883373(187) ack 1901766532 win 32736
11:25:06.957653 B.http > A.13520: F 4185883373:4185883373(0) ack 1901766532 win 32736
11:25:06.958477 A.13520 > B.http: . ack 4185883374 win 32120
11:39:18.045048 A.13520 > B.http: F 1901766532:1901766532(0) ack 4185883374 win 32120
11:39:18.047691 B.http > A.13520: R 4185883374:4185883374(0) win 0
```

4.1.2 Unresponsive hosts

All of the 29 instances of H_1 | timeout are due to TCP retries [18]. This is evident in the pattern of the timestamps in the original trace, shown in Table 7: Δt={3,6,12,24,48,96,...}s. This is, however, normal TCP traffic; the server simply isn't responding.

Table 7: *An example trace for H_1 | timeout.*

```
09:22:08.945690 A.5176 > B.http: S 2033154921:2033154921(0) win 512
09:22:11.943578 A.5176 > B.http: S 2033154921:2033154921(0) win 32120
09:22:17.943539 A.5176 > B.http: S 2033154921:2033154921(0) win 32120
09:22:29.943417 A.5176 > B.http: S 2033154921:2033154921(0) win 32120
09:22:53.943169 A.5176 > B.http: S 2033154921:2033154921(0) win 32120
09:23:41.942681 A.5176 > B.http: S 2033154921:2033154921(0) win 32120
09:25:17.941772 A.5176 > B.http: S 2033154921:2033154921(0) win 32120
```

Similarly, the H_2 | R^{APU} anomalies are generated by TCP retries, but their cause is different. Table 8 shows one set of the anomalies. The duration of each anomaly follows the pattern expected for TCP retries. In all cases, the client sends a SYN, the server replies with a SYN-ACK, but the client rejects the SYN-ACK and sends a reset. The reason for this behaviour is not apparent, but since it occurs on just two hosts, it is likely a local problem.

Of the 868 H_1 | R^{APU} anomalies, the first 16 anomalies are caused by repeated connection attempts to port 9000 which are rejected by the server. The remainder of the anomalies, starting at 08:03:52 on March 4, 1999 and ending at 11:26:27 on March 4, 1999 are rejected connection attempts to external web servers. These are undocumented network events.

Table 8: *An example trace for* $H_2 \mid R^{APU}$.

```
11:18:42.089211 A.3906 > B.http: S 1528226668:1528226668(0) win 512
11:18:42.092369 B.http > A.3906: S 3534951137:3534951137(0) ack 1528226669 win 32736
11:18:42.093409 A.3906 > B.http: R 1528226669:1528226669(0) win 0
11:18:45.084174 A.3906 > B.http: S 1528226668:1528226668(0) win 32120
11:18:45.087257 B.http > A.3906: S 3537946186:3537946186(0) ack 1528226669 win 32736
11:18:45.088192 A.3906 > B.http: R 1528226669:1528226669(0) win 0
11:18:51.084115 A.3906 > B.http: S 1528226668:1528226668(0) win 32120
11:18:51.087222 B.http > A.3906: S 3543946420:3543946420(0) ack 1528226669 win 32736
11:18:51.088164 A.3906 > B.http: R 1528226669:1528226669(0) win 0
11:19:03.084191 A.3906 > B.http: S 1528226668:1528226668(0) win 32120
11:19:03.088317 B.http > A.3906: S 3555948035:3555948035(0) ack 1528226669 win 32736
11:19:03.089263 A.3906 > B.http: R 1528226669:1528226669(0) win 0
11:19:27.083873 A.3906 > B.http: S 1528226668:1528226668(0) win 32120
11:19:27.086970 B.http > A.3906: S 3579947811:3579947811(0) ack 1528226669 win 32736
11:19:27.088011 A.3906 > B.http: R 1528226669:1528226669(0) win 0
11:20:15.083525 A.3906 > B.http: S 1528226668:1528226668(0) win 32120
11:20:15.086603 B.http > A.3906: S 3627949625:3627949625(0) ack 1528226669 win 32736
11:20:15.087585 A.3906 > B.http: R 1528226669:1528226669(0) win 0
11:21:51.082734 A.3906 > B.http: S 1528226668:1528226668(0) win 32120
11:21:51.085855 B.http > A.3906: S 3723953253:3723953253(0) ack 1528226669 win 32736
11:21:51.086861 A.3906 > B.http: R 1528226669:1528226669(0) win 0
```

4.1.3 Abandoned Connections

In 4 cases, a X | timeout anomaly followed by one or more L | A^{PU} anomalies is an artifact of the timeout values chosen for the algorithm. The TCP keepalive time of 2 hours is exceeded by the connection, however it appears that both the client and the server are violating this value since the connection is kept open and not reset.

Fourty-four C_1 | timeout anomalies are due to the second FIN of the closing not being sent at all. This is in accord with [19], which documents problems with HTTP connections terminating in the FIN_WAIT_2 state. It is not, however, only associated with port 80; ports 20, 23 and 25 are also involved.

One hundred X | timeout anomalies are abandoned connections that have no apparent explanation.

4.1.4 Other

There are 2 L | A^{PU} anomalies that have no apparent explanation.

4.2 Operational Network

The operational data was collected for the 24 hours of August 21, 2000 using the SHADOW scripts. At peak times, an hourly file contained on the order of 2 million packets. The anomalies determined from applying the TCP FSM to the data are

shown in Table 9. This data shows that approximately 37% of the Internet TCP traffic does not obey the rules defined by the FSM on that day. It is important to note that this value includes traffic that is involved in malicious activities such as scans. No DoS activity was detected in the period under examination. A limited examination was performed for each anomaly type to determine the common causes. Some of the causes were quite unexpected.

Table 9: *The anomalies found in the August 21, 2000 data for 24 hours starting at midnight. The anomalies caused by packets dropped due to the sniffer scripts have been removed. Of the 616527 connections, 63.1% were normal and 36.9% were anomalous.*

Terminal state	# anomaly	% anomaly
L | SA	270	0.1187
L | A^{PU}	579	0.2545
L | FA^{PU}	1446	0.6356
L | R^{APU}	7579	3.3315
H_1 | timeout	17979	7.9031
H_1 | FA^{PU}	2	0.0009
H_1 | R^{APU}	3057	1.3438
H_2 | timeout	293	0.1288
H_2 | FA^{PU}	19	0.0084
H_2 | R^{APU}	485	0.2132
X | timeout	80	0.0352
C_1 | timeout	1090	0.4791
Bad flags	194613	85.5472

In the process of analyzing the anomalous TCP connections, evidence of packets being dropped by the sniffer was found. 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 | FA^{PU} failure, while data was exchanged and the connection closed normally. This implies that the SYN-ACK packet was missed by the sniffer. By examining the traces associated with the anomalies, it was found that one L | SA and one H_1 | FA^{PU} were caused by packets dropped by the sniffer.

4.2.1 Scans

A large-scale SYN scan was responsible for 13131 of the H_1 | timeout events. A large-scale SYN-FIN scan was also detected in the traffic data and was responsible for 194598 bad flag failures. Slow scans were identified by clustering the anomalies by source IP address and anomaly type, and manually examining the original traffic data. Seven slow scans (less than 2 packets per hour) were detected in the traffic, of

the SYN, SYN/ACK and ICMP/ACK varieties.

There were 18 instances of SYN scanning for ports 80/3128/8080 in what appears to be a distributed scan, originating from 16 unique hosts. One SYN/ACK scan appears to originate from a source who knows the network, as the scan was directed only to DNS servers on port 53. This may have been a fingerprinting attempt. Three other smaller-scale SYN scans were detected.

These identified scans accounted for approximately 35% of the connections, or 92% of the anomalies.

4.2.2 Backscatter effects

Backscatter may be a cause of both $L \mid R^{APU}$ and $L \mid SA$ anomalies in the data. Backscatter from what is assumed to be a DoS attempt on another network resulted in 3774 $L \mid R^{APU}$ anomalies, based on the rate of arrival and the high destination ports.

4.2.3 Network Delays

Network delays caused a variety of anomalies, a subset of which are the same as those found in the DARPA IDE data set. Table 10 shows a case where 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.

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

```
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
```

4.2.4 Unresponsive hosts

In the operational data, $H_1 \mid R^{APU}$, $H_1 \mid$ timeout, $H_2 \mid R^{APU}$ and $H_2 \mid$ timeout anomalies were observed, some of which were likely caused by unresponsive hosts or some form of policy enforcement, such as a firewall.

4.2.5 Abandoned Connections

There are numerous connections that were abandoned, generating a X | timeout failure. However, there were surprising instances of connections that only appeared to have been abandoned, as shown in Table 11. This trace, which triggered both a C_1 | timeout and an L | A^{PU} failure, 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 network address translation device, but it is difficult to be certain.

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 | A^{PU} anomaly and a C_1 | timeout anomaly.*

```
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
```

4.2.6 Other

A traffic trace that caused a X | timeout followed by a L | SA is shown in Table 12. In this case, a SYN-ACK packet is replayed, but the sequence number changes. The client does not acknowledge this as an existing connection and rejects the packet.

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 | R^{APU} and H_2 | timeout. This has an impact on network performance, and is an indication that the firewall rules could be improved.

Other strange activity was detected on the network; an example of a malfunctioning TCP or application (anomaly H_1 | FA^{PU}) is shown in Table 13. 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.

Table 12: *A trace showing the arrival of a replayed SYN-ACK packet with the sequence number changed. This trace resulted in a $X \mid$ timeout anomaly followed by a $L \mid SA$ anomaly.*

```
10:53:03.276241 A:64377 > B:80: S 1052463045:1052463045(0) win 512
10:53:03.479397 B:80 > A:64377: S 3274719686:3274719686(0) ack 1052463046 win 8760
10:53:03.486114 A:64377 > B:80: . ack 3274719687 win 16060
10:53:03.490055 A:64377 > B:80: P 1052463046:1052463428(382) ack 3274719687 win 16060
10:53:06.480281 A:64377 > B:80: P 1052463046:1052463428(382) ack 3274719687 win 16060
10:53:08.217647 A:64377 > B:80: F 1052463428:1052463428(0) ack 3274719687 win 16060
10:53:12.480333 A:64377 > B:80: P 1052463046:1052463428(382) ack 3274719687 win 16060
10:53:12.803162 B:80 > A:64377: S 429671713:429671713(0) ack 1052463046 win 8760
10:53:12.809901 A:64377 > B:80: . ack 3274719687 win 16060
10:53:13.031776 B:80 > A:64377: R 3274719687:3274719687(0) win 8760
10:53:15.907686 B:80 > A:64377: S 429671713:429671713(0) ack 1052463046 win 8760
10:53:15.912258 A:64377 > B:80: R 1052463046:1052463046(0) win 0
```

Table 13: *A trace showing an example detected as anomaly type $H_1 \mid FA^{PU}$.*

```
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
```

The $H_2 \mid FA^{PU}$ failure triggered on 19 TCP connections made up of flag sequences similar to {S–SA–FA^{PU} –A^{PU} –FA^{PU} –A^{PU} }. In this case, the TCP three-way handshake was completed with a FA^{PU} 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. Although this could be interpreted as a full connect scan, it was concluded that the behaviour is benign because the activity only appeared on ports 20 and 80.

4.3 Comparison

To compare the DARPA IDE data to the operational data, it is necessary to remove anomalies that can be directly associated with attacks. The only attack traffic that could be easily detected in the operational data was due to scanning activity. Table 14 shows the DARPA IDE data set anomalies beside the operational data anomalies with obvious scans removed.

Table 14: *The compared proportions of anomaly types with known scanning activity removed from the operational data. Scans accounted for approximately 35% of the connections, or 92% of the anomalies.*

Anomaly	% anomaly (DARPA)	% anomaly (operational)
L | SA	0.39	1.18
L | A^{PU}	9.69	3.14
L | FA^{PU}	0.78	7.95
L | R^{APU}	0	41.66
H_1 | timeout	2.27	18.37
H_1 | FA^{PU}	0.08	0.01
H_1 | R^{APU}	67.81	16.81
H_2 | timeout	0	1.61
H_2 | FA^{PU}	0	0.10
H_2 | R^{APU}	0.70	2.67
X | timeout	8.13	0.44
C_1 | timeout	10.16	5.99
Bad flags	0	0.07

One might conclude from Table 14 that the match is reasonably good between the two data sets, if one removes the $H_1 \mid R^{APU}$ in the DARPA IDE data caused by the unidentified network event and the $L \mid R^{APU}$ in the operational data caused by backscatter. However, if one looks at the causes of each individual anomaly, the dissimilarities become more apparent. Table 15 shows a comparison of the causes of each anomaly. From this analysis, it can be concluded that a wider variety of bad TCP behaviour occurs operationally than is depicted by the DARPA IDE simulated

data. For example, the L | FA^{PU} anomaly has only one cause in the DARPA IDE data, whereas there are five causes for it in the operational data.

It is interesting to note that the most common anomalies in the operational data have undetermined causes, *i.e.* a number of causes could be postulated, but none with any certainty.

Table 15: *Comparison of the 1999 DARPA IDE data set anomalies with those found in the operational data in August 2000. The figures for the operational data have had obvious scanning activity removed.*

Failure State	Cause	DARPA	Operational
L | SA	dropped SYN	100%	0.5%
	replayed SA arrives after connection closed	–	4.7%
	replayed SA with different seq#	–	6.1%
	poorly managed firewall	–	2.3%
	handshake timeout	–	0.9%
	lone SA (undetermined)	–	85.5%
L | A^{PU}	abandoned connection	61.3%	7.4%
	2 hour keepalive exceeded	10.5%	1.0%
	dropped SYN and SYN-ACK	26.6%	–
	spurious change in IP/port	–	10.4%
	ACK arrives after connection closed	–	1.0%
	lone ACK (undetermined)	1.6%	84.9%
	model artefact due to corrupted packets	–	5.8%
L | FA^{PU}	FIN_WAIT_2 exceeded	100%	4.2%
	spurious change in IP/port	–	6.5%
	FA arrives after connection closed	–	49.7%
	replayed FA with different seq#	–	4.6%
	lone FA (undetermined)	–	42.6%
L | R^{APU}	FIN_WAIT_2 exceeded	–	0.2%
	SA never received	–	0.1%
	R response to replayed FA with different seq#	–	0.1%
	R response to S	–	0.2%
	R response to lone SA	–	0.1%
	R arrives after connection closed	–	0.9%
	spurious change in IP/port	–	0.01%
	lone R (undetermined)	–	98.4%

Failure State	Cause	DARPA	Operational
H_1 | timeout	handshake timeout	–	0.09%
	undetermined	100%	99.91%
H_1 | FA^{PU}	dropped SA	100%	50%
	undetermined	–	50%
H_1 | R^{APU}	S-S-A-R	–	0.03%
	undetermined	100%	99.97%
H_2 | timeout	poorly managed firewall	–	1.4%
	SA never received	–	2.0%
	undetermined	–	96.6%
H_2 | FA^{PU}	undetermined	–	100%
H_2 | R^{APU}	undetermined	100%	100%
X | timeout	abandoned connection	96.2%	87.5%
	keepalive exceeded	3.8%	5.0%
	replayed SA with different seq#	–	7.5%
C_1 | timeout	FIN_WAIT_2 exceeded	66.2%	9.7%
	abandoned connection	33.8%	77.7%
	spurious change in IP/port	–	12.5%
	replayed SA with different seq#	–	0.1%
Bad flags	FRA packets in closing	–	69.2%
	bad flags within legitimate connections	–	30.8%

5 Conclusions

In previous work, a simplified TCP finite state machine was introduced [9], capable of detecting anomalous TCP behaviour, *i.e.* behaviour that deviates from the protocol as defined in RFC 793 [7]. In this work, the TCP FSM model was tested using the "Week 1" portion of the 1999 DARPA IDE data set, which is known to be free of malicious activity. It was found that the model effectively detects implementations of TCP that do not follow the protocol standard, and network events such as loss of availability. The DARPA IDE data was found to contain an undocumented network event: the data for the morning of March 4, 1999 contains repeated rejected connection attempts to external web servers. It is possible that a network firewall was temporarily misconfigured. Dropped packets were also confirmed to exist within the data.

The model was then used to locate the TCP anomalies in an operational data set. Investigation of the anomalies showed a number of TCP implementations that deviate from the protocol standard. Several instances of scanning activity were found, some of which were temporally slow and would not be detected by conventional methods.

The comparison of the anomalies found in the two data sets shows that the TCP behaviour in the DARPA IDE data set does not completely reflect the real behaviour of TCP traffic on the Internet. The DARPA IDE data set is lacking in "Internet crud" [14] that arises due in part to backscatter, the variations in TCP implementations, and to a small degree, corrupted packets. The DARPA IDE data set was built using a simulated network with mostly the Linux 2.0.32 kernel implementation of TCP [11], which contributes to the homogeneity of anomalies. Since there are no attacks occurring outside of the simulated network, one cannot have backscatter, which reduces the number of anomalies. As well, packet corruption is unlikely to be found on a small network.

The effort required to simulate "normal" background traffic would be substantial [5]. The definition of "normal" itself is subjective, and may differ greatly from one day to the next. For our purposes, the DARPA IDE data sets are sufficient, as the data certainly contains a subset of actual Internet behaviour, and it has been concluded that the effort required to generate "normal" background traffic for testing network security methodologies outweighs the benefits to our research program. Others considering this work may consider using VMware to provide the variety of operating systems required to give the "normal" behaviour of the Internet.

In comparing the two data sets, the assumption was made that network scanning was the only malicious activity generating anomalies on the operational network. Scans were detected by grouping anomalies and manually investigating each group. These anomalies had to be identified and removed for comparison purposes. Although it cannot be claimed that all anomalies due to malicious activities have been removed, any remaining are small in number and have little effect on the relative proportions presented. The 3 seconds lost in the collection of the operational data adds an unknown factor in the results as well; it is, however, assumed that the anomalies resulting from this were accounted for and that its effect on the comparison is negligible. In future, an automation of the scan detection process will be developed, with emphasis on the detection of slow scans.

References

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

[2] 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.

[3] 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.

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

[5] 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.

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

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

[8] Hennie, F. (1968), Finite-State Models for Logical Machines, New York: John Wiley & Sons.

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

[10] Zissman, M. (2001), DARPA Intrusion Detection Evaluation Data Sets (Online), MIT Lincoln Laboratory, `http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html` (Access Date: Aug. 2005).

[11] Haines, J.W., Lippmann, R.P., Fried, D.J., Zissman, M.A., Tran, E., and Boswell, S.B. (2001), 1999 DARPA Intrusion Detection Evaluation: Design and Procedures, (Technical Report 1062), MIT Lincoln Laboratory.

[12] Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., and Das, K. (2000), Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation, In *RAID '00: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, pp. 162–182, London, UK: Springer-Verlag.

[13] Mahoney, M.V. and Chan, P. K. (2003), An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection, (Technical Report TR CS-2003-02), Florida Institute of Technology.

[14] McHugh, J. (2000), Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory, *ACM Transactions on Information and System Security*, 10(4), 262–294.

[15] Grégoire, M. and Lefebvre, J. H. (in preparation), The Network Traffic Analysis Toolbox, Defence R&D Canada – Ottawa.

[16] Jacobson, V., Leres, C., and McCanne, S. (Dec. 2002), TCPdump (Online), Lawrence Berkeley National Laboratory, `http://www.tcpdump.org` (Access Date: Jan. 2003).

[17] Naval Surface Warfare Center – Dahlgren Lab (2001), NSWC SHADOW Index (Online), U.S. Navy, `http://www.nswc.navy.mil/ISSEC/CID` (Access Date: Jan. 2003).

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

[19] 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` (Access Date: Mar. 2003).

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
3701 Carling Ave, Ottawa, ON
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 Algorithm for Detecting TCP Anomalies (U)

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

Treurniet, Joanne R.

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)
November 2005	17	19

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.)

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)
15bo01	

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)

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.)

Unlimited

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).

The Transmission Control Protocol (TCP) is a well-defined protocol, and as such, a finite state machine can be defined to reflect the progression of a TCP connection, including deviations from the protocol standard. The 1999 DARPA Intrusion Detection Evaluation (IDE) data set was used to test the model. It was found that the model effectively detects implementations of TCP that do not follow the protocol standard and network events such as loss of availability. The model was then used to analyse the TCP anomalies of an operational data set, in which several instances of scanning activity were also successfully detected. A comparison of the anomalies found in the Week 1 DARPA IDE data set with those found in the operational data showed that the behaviour of the simulated TCP traffic does not contain the variations found in an operational setting.

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.)

TCP, anomaly detection, DARPA Intrusion Detection Evaluation, finite state machine

Defence R&D Canada

Canada's leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca