On Event Reproduction Ratio in Stateless and Stateful Replay of Real-World Traffic

Ying-Dar Lin, Chun-Nan Lu, Jose Miguel Sagastume, Jui-Tsun Hung, and Yuan-Cheng Lai

***Abstract*—Capturing and replaying network flows are important for testing network devices. Replayed traffic should reproduce effects similar to live traffic. This work presents methods to measure the event reproduction ratio, and studies the effectiveness of stateless and stateful traffic replayers based on the events triggered by packets and connections. We use two replayers, SocketReplay and Tcpreplay, and a networking device supporting security services. SocketReplay is a stateful replayer which keeps the state of a connection during replay, while Tcpreplay is a stateless replayer that ignores the connection state. Results indicate that SocketReplay replayed a smaller ratio of the captured traffic and triggered fewer blocking events in subsequent replay tests. Triggering blocking events denotes the replayed traffic cannot fit the onsite context. SocketReplay only replayed 38.74% of the captured TCP traffic, and resulted in an effectiveness of 99.97% (0.00%) in passing (blocking) event ratio. In contrast, Tcpreplay replayed 99.99% of the captured TCP traffic, and resulted in an effectiveness of 99.73% (75.64%) in passing (blocking) event ratio. The choice of a proper replayer and the corresponding replay configuration should depend on the contents of captured traffic and avoid to a significant drop of event reproduction ratio and the effectiveness of replayers.**

***Index Terms*—traffic replay, event reproduction ratio, replay effectiveness**

I. Introduction

The testing of network devices has been a major focus on the network research area. The goal is to create a range of test scenarios similar to the scenarios experienced under live deployment. The ultimate goal of network device testing is to debug network device problems in a controlled and transparent test bed that enables error reproducibility. One method for network device testing is to generate or to replay traffic with testing tools in order to check the behaviors of the devices under test (DUTs).

The traffic that is used on network device testing can be classified into Model-based and Trace-based. The former uses mathematical models to generate artificial network traffic; while the latter is based on real-world traffic captured from live deployments. The tools that generate model-based traffic are not difficult to implement; however, they are limited by the numerical properties found in the mathematical model. The trace-based traffic is captured under real-world network environment, thus it includes all properties found in live deployment. However, it has the issues of storage overhead and efficiency when it comes to trace-based traffic. Trace-based traffic usually need more storage and take more time than model-based traffic searching for specific traffic because the latter can be generated based on some features.

A traffic replay tool can be either stateless or stateful. A stateless replay tool replays captured network traces based on timestamps or sequence order and does not modify the packets in the traces. Therefore, the content of the replayed traffic is verbatim to the content captured in the network traces. Tcpreplay [1] is a particular stateless replayer. Since the traffic replayed by a stateless replay tool is verbatim to the recorded traces, a DUT that keeps track of the states of its network connections (such as TCP streams) might not understand the replayed traffic correctly.

On the other hand, a stateful replay tool modifies the content of the network traces so as to adapt the test conditions of the DUT, and would alter the content of subsequent packets in the network traces based on the responses of the DUT. An example of stateful replay tool is SocketReplay [2], which can mimic the TCP/IP stack (including IP addresses and port numbers) and replay payloads to maintain the TCP semantics.

A traffic replay tool is designed to replay the network traces correctly and to reproduce the same events occurred in live traffic such as packet blocking, packet modification, and log triggering. The traffic replayed by a replay tool must be understood by the DUT and must be able to represent the onsite context while the live traffic was captured. Therefore, we design a metric, called effectiveness, to measure the similarity between the events shown in live traffic and replayed traffic. Specifically, the effectiveness is based on the event reproduction ratio on a DUT between live traffic and replayed traffic. It is challenging to directly calculate the effectiveness due to the complexity of real-world traffic and different response mechanisms of distinct types of DUTs. In later experiments, we select Tcpreplay and SocketReplay to be the representatives of the stateless and stateful replayers, respectively, and evaluate their performances based on live traffic and replayed traffic. Tcpreplay is able to classify traffic as client or server, rewrite packet header information and replay traffic back onto the network and through other networking devices, such as switches and routers. SocketReplay supports loss recovery, which recovers incomplete connections to replay

Manuscript received November 07, 2013; revised January 08, 2014.

Ying-Dar Lin and Chun-Nan Lu are with the Department of Computer Science, National Chiao Tung University, Taiwan. E-mail: {ydlin, cnlu}@cs.nctu.edu.tw.

Jose Miguel Sagastume, and Jui-Tsun Hung are with the Department of Computer Science, National Chiao Tung University, Taiwan. E-mail: {josesagas, Jason.hung.sb}@gmail.com.

Yuan-Cheng Lai is with the Department of Information Management, National Taiwan University of Science and Technology, Taiwan. E-mail: laiyc@cs.ntust.edu.tw.

a complete TCP stream.

This work presents methods to measure the event reproduction ratio, and studies the effectiveness of stateless and stateful traffic replayers based on the events triggered by packets and connections. Given a captured live traffic and its corresponding triggered logs of five types of events, including blocking events, modifying events, passing events, logging events, and non-logging events. The captured live traffic is thenreplayed by a replay tool, which would trigger another corresponding logs based on the five types of events. The effectiveness is measured by comparing the number of events occurred in live traffic with that in replayed traffic.

The rest of this paper is organized as follows. In Section II, we survey relevant replay tools and projects. Section III describes the definitions, terminologies, and the problem statements. Section IV presents the issues of event reproduction ratios and the proposed measurement method. Section V evaluates the replay tools using our metric. Finally, we conclude in Section VI.

II. Related Work

In order to accurately replay the traffic so that it is recognized as valid network traffic by a DUT, a replay tool must be able to send out the correct packets in the correct order and direction to test the DUT. A number of projects developed replay tools or plug-ins to solve the problems of traffic replay, and the developed tools can be divided into two types, namely stateless and stateful, based on whether the replayer modifies and tracks the replayed traffic or not.

A. Stateless Replay

Stateless replay means that the replay tool does not modify the TCP sequence numbers or acknowledgement numbers to reflect the states of the TCP streams. For example, Tcpreplay simply replays the packets of the captured traces in the order of the packet timestamps at a specified rate. Tcpreplay does not actively alter the information of the transport layer header and payload of a packet. Tomahawk [5] is another stateless replayer and is designed to test the throughput and blocking capability of network-based intrusion prevention systems (NIPS). Both Tcpreplay and Tomahawk divide the captured traffic into traffic originating from the client and traffic originating from the server and replay the trace between two network interfaces.

B. Stateful Replay

Some replayers can maintain the states of the network layer and those of the transport layer during replay. SocketReplay supports stateful replay in the network and transport layers because many DUTs, including NAT devices, proxies, and security appliances, may modify transport layer headers. SocketReplay could update the response states to prevent these DUTs from replaying blocked connections.

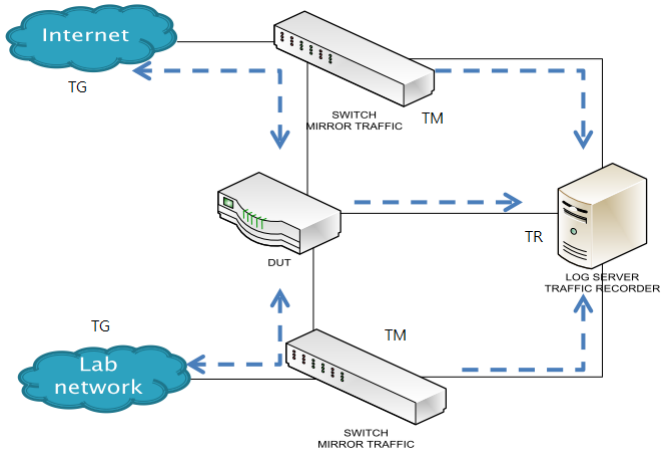
Each traffic replayer developed distinct methods to measure its own effectiveness. TCPopera [3] uses four heuristics to follow the TCP/IP states and calculates the number of replayed traffic flows using statistical methods based on short-term and long-term profiles, the number of packet reorderings and session duration. WirelessReplay [4] uses the connection states defined in IEEE 802.11 protocol to be representative of different events and computes the reproduction rate of the events in the replayed traffic. SocketReplay measures the effectiveness using the reproduction rate of triggered attack sessions on a security appliance appeared in the replayed traffic.

Monkey [6] replays web application traffic by emulating the TCP stack to reproduce network conditions. Monkey infers delays caused by the client, the applications, the server, and the network in each captured flow and replays each flow according to its inferences. Although the work in [7, 8] support traffic replay at the application layer, they fail to replay traffic captured from a large network.

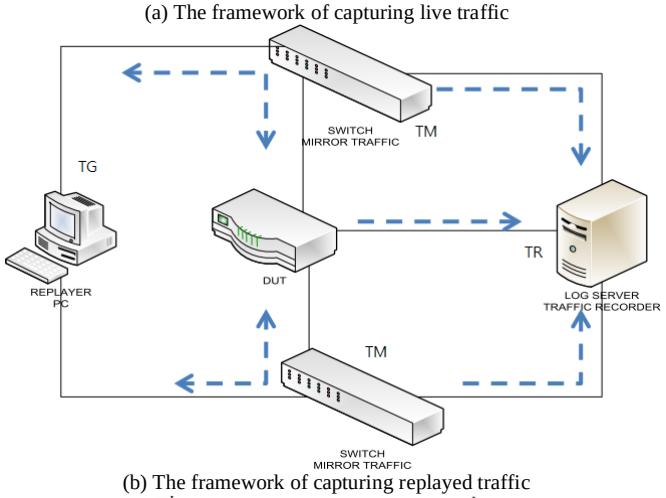
III. Problem Statement

In this section we describe the details of our framework, the terminology definitions, and our target problems.

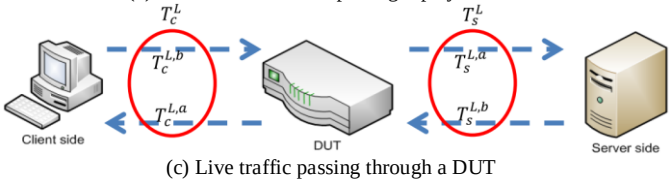
A. Framework



(a) The framework of capturing live traffic



(b) The framework of capturing replayed traffic



(c) Live traffic passing through a DUT

Fig. 1. Frameworks of capturing live and replayed traffic

Our frameworks are illustrated in Fig. 1. Fig. 1(a) and Fig. 1(b) are the frameworks used to capture live traffic and replayed traffic, respectively. The frameworks have four major components, namely a DUT, a traffic generator (TG), a traffic mirror (TM), and a traffic recorder (TR). The DUT can be a router, a firewall, or a proxy. TG can be the Internet, a local network, or a traffic replay tool that can generate or include network traffic inside. TM is a layer-2 switch, and TR is a server that captures live and replayed traversing traffic. Furthermore, TR also records live logs and responses from the DUT during replay. Fig. 1(c) focuses on the two interfaces of the DUT.

In Fig. 1(a), the DUT works in in-line mode between the local network and the Internet. TG here is the live traffic between the Internet and the local network. Two instances of TM, before and after the DUT in the link, are used to duplicate the input and the output traffic to the TR, which connects and monitors the status of the DUT. In Fig. 1(b), TG is replaced with a traffic replay tool to send out the captured packet traces. During replay, the TM duplicates all traffic traversing the DUT; the TG then transfers the duplicate to the TR. Afterwards, we parse and analyze the two logs separately obtained from the two frameworks to evaluate the effectiveness of the traffic replay tool.

B. Terminologies Definitions

A term event is defined as a log of how a packet or connection with some attributes is processed while traversing a networking device. For packet events, there are two specific attributes: blocking (b) and modifying (m). If a packet is blocked, a blocking event is logged; if a packet is not blocked and not modified, a passing event is logged; if a packet is modified but not blocked, a modifying event is logged. Packets that are neither blocked nor modified are identified as events with passing, a non-specific attribute. For a connection, it may or may not trigger a log while traversing a networking device. Once a connection triggers a log, a specific event with a logged attribute is recorded, which is a logging event. If a connection does not trigger a log, the connection is identified as a non-specific event with a non-logged attribute, which is a non-logging event. Therefore, there are a total of five types of event attributes that can be derived: (1) blocking event; (2) modifying event; (3) passing event; (4) logging event; and (5) non-logging event. For a connection, if it triggers a log on the DUT, a logging event is marked with the connection; otherwise, a non-logging event is marked. Different types of DUTs may have distinct things of interest. Whenever such things are found, the DUTs will generate corresponding logs. For an IDS or IDP, it is important whether a pre-defined pattern appeared in the packet payload or not; for a firewall, it is vital whether an IP address on the blacklist fails to transfer packets through the firewall or not.

Considering or excluding the logs would not affect the accuracy of the framework and the metrics proposed in this work. This work pays attention to the traffic replay test on network devices. If a log is generated by the DUT during replay test, a logging event can be took into consideration about the event reproduction effectiveness; if a log is not generated by the DUT or it is not important during replay test, the logging event can be ignored or regarded as non-logging events. Traffic replaying test can be used to test network devices which are designed to handle packets passing by or passing through. Therefore, there are individual triggered events for all packets and connections in live and replayed traffic, which helps our later analysis on effectiveness.

In this way, both live traffic and replayed traffic trigger a number of events during replay. In order to compute effectiveness, some related terminologies are defined as follows. T^L and T^R refer to the live traffic and the replayed traffic, respectively. T^L and T^R may trigger events with blocked (b), modified (m) and logged (l) attributes. Events triggered in T^R are compared with those triggered in T^L, and the outcomes can be classified into true positive (e_{TP}), true negative (e_{TN}), false positive (e_{FP}), and false negative (e_{FN}). Tabularized relations between truth and falseness of event reproduction are shown in Table I.

With respect to each attribute, the variable $|e_{TP}|$ is the number of events with one type of attributes in the live traffic that are reproduced in the replayed traffic; $|e_{FN}|$ is the number of events with the same type of attributes that do appear in the live traffic, but are not reproduced in the replayed traffic; $|e_{TN}|$ is the number of events without the specific attribute in the live traffic but are reproduced with that attribute in the replayed traffic; $|e_{FP}|$ is the number of events such that a specific attribute is marked neither in the replayed traffic nor in the live traffic.

TABLE I.
THE TRUTH AND FALSENESS OF EVENT REPRODUCTION WITH SPECIFIC ATTRIBUTE

Type / Traffic	Live	Replayed	Comparison outcome
Event with specific attribute	1	1	e_{TP}
	1	0	e_{FN}
Event without specific attribute	0	0	e_{TN}
	0	1	e_{FP}

In Table I, "1" signifies an event occurred with a specific attribute, and "0" signifies an event occurred without any specific attributes. An event that was unblocked, unmodified, or non-logged would be identified as an event without specific attribute. If an event is reproduced successfully, the outcome should be either e_{TP} or e_{TN}. Otherwise, the outcome should be e_{FN} or e_{FP}.

We introduce three metrics to measure the effectiveness of a replay tool: (1) the event reproduction ratio; (2) the effectiveness of event reproduction; and (3) the consistency ratio.

At the beginning of this section, we define five types of event attributes to represent five possible outcomes when an event occurs at a DUT. Therefore, the *event reproduction ratio* can be obtained by the combination of the events with specific attributes and those with non-specific attributes, which include *blocking* reproduction ratio (br), *modifying* reproduction ratio (mr), *logging* reproduction ratio (lr), *passing* reproduction ratio (pr), and *non-logging* reproduction ratio (nlr). Equation 1,

$$Reproduction\ ratio = \frac{\sum_{i=1}^{n} \neg(E_i^L \oplus E_i^R)}{n} \times 100\% \quad (1)$$

is used to compute the event reproduction ratio for br, mr, lr, pr, and nlr, where ¬ is the negation operator; ⊕, the xor operation; E_i^L, the *i*-th event in the live traffic; E_i^R, the event corresponding to E_i^L in the replayed traffic; and *n*, the total number of packet events or connection events of a network trace.

During replay, an event would be marked as having specific or non-specific attribute by a DUT. In order to describe the effectiveness of reproduction of events with specific and non-specific attributes separately, two equations are designed to compute the effectiveness as $TP\_Rate = \frac{|e_{TP}|}{|e_{TP}|+|e_{FN}|} \times 100\%$ (2) for the events with specific attributes and $TN\_Rate = \frac{|e_{TN}|}{|e_{TN}|+|e_{FP}|} \times 100\%$ (3) for the events with non-specific attributes.

For each connection event in the live traffic, it is crucial that the connection can be reproduced in the replayed traffic. Thus, we define *consistency* to refer to the condition that a connection in a replayed trace has the same number of packets as that of its corresponding connection in the original live traffic. Otherwise, the connection is regarded as inconsistent. Duplicated packets are not taken into account. The degree of consistency of a replayed trace can be measured by $consistency\ ratio = \frac{\sum_{i=1}^{n} c_i}{n} \times 100\%$, where c_i is the *i*-th consistent connection, and *n* is the total number of replayed connections. The binary value c_i can be either 0 (inconsistent) or 1 (consistent).

Besides, in order to compare the results from different aspects, several metrics are defined. To measure the ratio of replayed traffic, a term *Replayed Traffic Ratio* was defined as $Replayed\ Traffic\ Ratio = \frac{|T^R|}{|T^L|} \times 100\%$, where $|T^R|$ is the number of replayed packets and $|T^L|$ is the number of captured packets.

To measure the ratios of the occurrence of live events and replay events for a specific attribute, two terms *Live Event Ratio* and *Replay Event Ratio* were defined as

$$Live\ Event\ Ratio = \frac{|e_{TP}|+|e_{FN}|+|e_{LP}|}{|e_{TP}|+|e_{FP}|+|e_{TN}|+|e_{FN}|+|e_{LP}|+|e_{LN}|} \times 100\%$$

and

$$Replay\ Event\ Ratio = \frac{|e_{TP}|+|e_{FP}|+|e_{RP}|}{|e_{TP}|+|e_{FP}|+|e_{TN}|+|e_{FN}|+|e_{LP}|+|e_{LN}|} \times 100\%,$$

where $|e_{LP}|$ and $|e_{LN}|$ are the number of events with and without one type of attribute only in live traffic, and $|e_{RP}|$ and $|e_{RN}|$ are the number of events with and without one type of attribute only in replayed traffic. In the ideal case, $|e_{LP}|$ should be equal to $|e_{RP}|$ and $|e_{LN}|$ should be equal to $|e_{RN}|$; however, in most cases, they are different. Therefore, we separate the notations of the events triggered in live traffic and triggered in replayed traffic in order to avoid confusion.

To measure the ratio of the logs generated, a term *Logging Event Ratio* was defined as $Logging\ Event\ Ratio = \frac{|logs|}{|connections|} \times 100\%$, where $|logs|$ is the number of logs generated during the process of capturing T^L or T^R and $|connections|$ is the number of TCP connections and UDP pseudo connections for T^L or T^R.

C. Problem Statement

With the help of the three equations, namely *Reproduction ratio* (1), *TP_Rate* (2) and *TN_Rate* (3) defined in the previous subsection, the problem of measuring the effectiveness of a replay tool can be defined as follows.

During a test for a networking device, live and replayed traffic separately produce a sequence of live and replayed events, E^L and E^R, respectively. If the value of the *i*-th event of E^L or E^R is marked as 0, it means that the event does not have any specific attributes. On the other hand, if the value of an event is marked as 1, it means that the event has one of the specific attributes, namely blocking, modifying, or logging.

Given a captured live traffic T^L and its corresponding triggered logs, we mark a sequence of events E^L with various attributes, such as blocking events $E^{L,b}$, modifying events $E^{L,m}$, logging events $E^{L,l}$, or non-specific events. The captured traffic T^L is then replayed by a replay tool, which may trigger another sequence of events E^R with various attributes, such as blocking events $E^{R,b}$, modifying events $E^{R,m}$, logging events $E^{R,l}$, or non-specific events. For a successful replay test, E^R should be as consistent to E^L as possible. The consistency can be quantified by the consistency ratio of replayed traffic mentioned above.

Therefore, the objectives of this work can be formally described as: (1) to measure the event reproduction ratio; and (2) to compute the effectiveness of the event reproduction with specific or non-specific attributes in the replayed traffic T^R.

IV. Effectiveness of Replayed Traffic

There are three challenging issues in comparing events between the live and the replayed traffic: (1) issue on network behaviors; (2) issue on captured traffic; and (3) issue on traffic identification. We detail these issues in subsection A, and propose a solution to measure the effectiveness of replayed traffic in subsection B.

A. Event Comparison Issues

Issue on network behaviors. The behavior of a connection or an activity captured in T^L may affect the completeness of the captured traffic. For example, if an event happens in live traffic, such as a packet loss, receiving a duplicate packet, or receiving an out-of-order packet, it would be challenging to reproduce the event in the replayed traffic because of missed or blocked packets/connections.

Issue on captured traffic. A DUT might alter its traversing traffic, which in turn affects the consistency of the replayed traffic; thus it is better to capture the live traffic right before it reaches the DUT to avoid this inconsistency. If we capture the traffic after it has traversed the DUT, the captured traffic may not be able to reproduce the same sequence of events as the live traffic because some packets are blocked or modified by the DUT.

Issue on traffic identification. In order to analyze and verify the correctness of marked event attributes, it is necessary to have the knowledge of the content of the captured traffic. Only when the characteristics of the captured traffic are known clearly, can the outcomes of the effectiveness of replayed traffic be confirmed.

B. Solutions to Measuring the Effectiveness of Replayed Traffic

In order to measure the effectiveness of replayed traffic, we proposed a four-phase solution, including (1) capture network traffic, (2) process live and replayed traffic, (3) identify blocking and modifying events, and (4) compare the two sequences of events E^L and E^R by computing the event reproduction ratios and the effectiveness. Fig. 1(c) illustrates an example of live traffic passing through a DUT and each phase is described as follows.

Capture traffic. There is one pair of live traffic flows on each of the client side and the server side: T_c^L and T_s^L, respectively. Flow $T_c^{L,b}$ is the flow initiated from a client to the DUT, flow $T_c^{L,a}$ is the flow initiated from the DUT to the client, flow $T_s^{L,b}$ is the flow initiated from a server to the DUT, and flow $T_s^{L,a}$ is the flow initiated from the DUT to the server.

T^L and T^R are captured, respectively, and a set of live logs L^L from the live traffic and a set of replay logs L^R from the replayed traffic are also recorded. First, the live traffic traces (T_c^L and T_s^L) and the logs (L^L) are captured and recorded. Second, the traffic trace T_c^L is split into two sub-traces based on the source and destination IP addresses, which are $T_c^{L,b}$ and $T_c^{L,a}$. In the same way, the trace T_s^L is also split into two parts, namely $T_s^{L,b}$ and $T_s^{L,a}$. Next, the two traces, $T_c^{L,b}$ and $T_s^{L,b}$, are merged into a new trace that is not processed by the DUT. The steps mentioned here are also applied to the replay test, and the replayed traffic traces $T_c^{R,\beta}$, $T_s^{R,\beta}$, $T_c^{R,\alpha}$, and $T_s^{R,\alpha}$ are obtained for further event reproduction analysis. The roles of flow $T_c^{R,\beta}$ and flow $T_c^{L,b}$ are similar except the former is used in replayed traffic and the latter is used in live traffic. The relationship between $T_c^{R,\alpha}$ and $T_c^{L,a}$, $T_s^{R,\alpha}$ and $T_s^{L,a}$, and $T_s^{R,\beta}$ and $T_s^{L,b}$ are the same cases.

Process live and replayed traffic. The traces $T_c^{L,\beta}$, $T_s^{L,\alpha}$, $T_s^{L,\beta}$, $T_c^{L,\alpha}$, $T_c^{R,\beta}$, $T_s^{R,\alpha}$, $T_s^{R,\beta}$, and $T_c^{R,\alpha}$, are further processed into corresponding sets of connections, $C_c^{L,\beta}$, $C_s^{L,\alpha}$, $C_s^{L,\beta}$, $C_c^{L,\alpha}$, $C_c^{R,\beta}$, $C_s^{R,\alpha}$, $C_s^{R,\beta}$, and $C_c^{R,\alpha}$. The connections are identified by a five-tuple {Src IP, Dst IP, Src Port, Dst Port, Proto}, and each packet within each connection is identified by its IP identification number, TCP sequence number, and packet payload. We use these connection sets because the packets within these connections haven't been modified or blocked by a DUT. Therefore, we can use them to compare all the events produced by the live and the replayed traffic.

Suppose E^L (E^R) is generated by a live (replayed) traffic trace with a corresponding log L^L (L^R). The pair of connections ($C_c^{L,\beta}$, $C_s^{L,\beta}$) is used to create a sequence of live events $E^{L,b}$ and $E^{L,m}$. The pair of connections ($C_c^{R,\beta}$, $C_s^{R,\beta}$) is used to create a sequence of replayed events $E^{R,b}$ and $E^{R,m}$. Packets within the above pairs of connections are treated as packet events. Logs are compared against each other to generate the connection events. Each entry on L^L and L^R is mapped to a connection event, associated with its corresponding connection. Connections that are not registered in L^L or L^R are taken as connection events with non-specific attribute, i.e. events with non-logged attribute.

In order to log packets in the specific and regular traffic, the anomaly-based rules and the signature-based rules [10] are invoked. Signature-based logging events could be found in all three replay configurations, while few anomaly-based logging events are found and triggered.

Compare packets within pairs of connections ($C_c^{L,\beta}$, $C_c^{L,\alpha}$), ($C_c^{R,\beta}$, $C_c^{R,\alpha}$), ($C_s^{L,\beta}$, $C_s^{L,\alpha}$), and ($C_s^{R,\beta}$, $C_s^{R,\alpha}$) to identify blocked and modified packets. This information is used to assign modifying and blocking events. The packets that are not logged in the result are treated as passing events.

Compare the sequences of events $E^{L,b}$, $E^{L,m}$, $E^{L,l}$, $E^{R,b}$, $E^{R,m}$, and $E^{R,l}$ to compute the event reproduction ratios and the effectiveness of the replayed traffic. The event orders in the sets of E^L must be in the same order as that in the set of E^R. The order of the event sets ensures the correctness of the event comparison between a live traffic trace and a corresponding replayed traffic trace. The packet events are ordered based on the TCP sequence number and the IP address. Connection events are not required to be in order for computing the percentage of event reproduction because they are compared against each other using the 5-tuple information of a packet. If the replayed traffic is different from the live traffic, we use identifiers which link the live and the replayed traffic to compare events. The identifiers show the changes in the fields of live traffic packets, such as IP address and port numbers.

V. EXPERIMENTS AND RESULTS

We use SocketReplay and Tcpreplay to evaluate the effectiveness of stateless and stateful traffic replay tools. ZyWAll USG1000 with installed services of anomaly detection, intrusion detection/prevention, and firewall was used as the DUT.

A. Experiment Settings

The test bed on Fig. 1(a) was configured to capture T^L. The same procedure was also done for the replayed traffic using the test bed in Fig. 1(b). It is important to simultaneously capture T_c^L and T_s^L because they should contain identical packets except those blocked by the DUT. The size of each packet captured on the traffic recorder (TR) was restricted to be less than 24,000 bytes to avoid possible packet loss caused by SocketReplay. Before starting to replay traffic, the traffic traces were padded with zeroes to fill any missing bytes. Without

padding, a DUT would automatically block the packets for having incorrect size in the payload field. A DUT has three types of actions when it encounters malicious traffic: (1) it rejects the connections containing malicious contents, (2) it blocks some packets containing malicious contents, and (3) it still forwards the packets containing malicious contents. In our experiments, Tcpreplay was configured to invoke the pcap pre-processor, Tcpprep [9], to create a cache file, which is used to split traffic into two sides, one network that contains the hosts and the other network that do not.

In fact, if we use Tcpreplay to replay traffic with two network interfaces, Tcpprep determines which interface from each packet will initiate. Tcpprep supports multiple modes of replay operation, and two of the modes used in our experiments were bridge mode and the mode of IPv4/v6 matching Classless Inter-Domain Routing (CIDR) as defined in RFC 4632 [11]. In bridge mode, Tcpprep parses a packet trace and keeps track each instance a host either behaves like a client or like a server. The traffic in the client side is defined as "Sending the traffic contains the following messages." The traffic in the server side is defined as "Receiving and responding to the incoming messages." In CIDR mode, a user can specify, in CIDR notation, one or more networks that contain hosts. Then, the traffic can be split into two different sides, one network that contains the hosts and the other network that do not.

B. Data Analysis

TABLE II. RELATED EXPERIMENT STATISTICS

A. THE PROFILE OF PACKET TRACES

	Live traffic				SoccketReplay traffic			
Type	$T_c^{L,\beta}$	$T_c^{L,\alpha}$	$T_s^{L,\beta}$	$T_s^{L,\alpha}$	$T_c^{R,\beta}$	$T_c^{R,\alpha}$	$T_s^{R,\beta}$	$T_s^{R,\alpha}$
File Size (MB)	3.9	7.2	7.4	3.8	2.8	4.9	4.9	2.8
Number of packets	33407	29420	3273	33143	19370	8087	8093	19374
Number of TCP connec-tions	7287	7291	8861	7286	462	462	462	462
Number of UDP connec-tions	3628	2183	2174	3608	11410	499	499	11410
	Tcpreplay traffic (bridge mode)				Tcpreplay traffic (CIDR mode)			
Type	$T_c^{R,\beta}$	$T_c^{R,\alpha}$	$T_s^{R,\beta}$	$T_s^{R,\alpha}$	$T_c^{R,\beta}$	$T_c^{R,\alpha}$	$T_s^{R,\beta}$	$T_s^{R,\alpha}$
File Size (MB)	3.9	6.9	7.3	3.6	4.9	7.9	8.4	4.4
Number of packets	34537	32017	32503	33743	44905	39952	44105	42064
Number of TCP connec-tions	677	613	685	599	7287	7272	8858	5776
Number of UDP connec-tions	2481	2178	2179	2479	3628	2172	2175	3608

B. PROFILES OF THE TWO TYPES OF TRAFFIC

Fields	Specific traffic	Regular traffic
Number of TCP connections	8870	5960
% of TCP closed connections with FIN	6.61%	89.45%
% of TCP closed connections with RST	75.85%	3.27%
% of TCP unclosed connections	17.54%	7.28%
Number of UDP pseudo connections	3632	5560

C. THE STATISTICS OF REPRODUCED EVENTS

	Blocking		Modifying	
Replayer	Tcpreplay	SocketReplay	Tcpreplay	SocketReplay
$\|e_{TP}\|$	38	0	1	5
$\|e_{FP}\|$	202	3	0	0
$\|e_{TN}\|$	13299	12466	13662	12553
$\|e_{FN}\|$	118	84	0	0
$\|e_{LP}\|$	168	236	13	9
$\|e_{LN}\|$	16516	17548	16852	17884
$\|e_{RP}\|$	0	3	0	0
$\|e_{RN}\|$	5	513	0	515

D. THE STATISTICS OF CONNECTIONS AND RELATED LOGS OF THE LIVE AND REPLAYED TRAFFIC

Traffic information / Traffic direction	Client → Server	Server → Client
Number of live connections	7281	7291
Number of L^L	206	
Number of replayed connection	599	612
Number of L^R	57	

Two types of traffic were used in our experiment. One type of traffic containing malicious activities, called specific traffic was generated from several security websites [12, 13, 14, 15, 16]. The other type of traffic, namely regular traffic was captured from the National Chiao Tung University. For packet events we only compared the TCP traffic because there's no state involved in UDP traffic and therefore is not applicable for SocketReplay. For connection events we use logs originated from TCP traffic or UDP traffic.

We profile and calculate separate traces and tabulate the results in Table II. Table II (A) gives the profiles of the packet traces that were captured for our experiments, and Table II (B) compares the profiles of the specific traffic and the regular traffic based on the number of connections, the way how a connection is terminated, and the number of TCP and UDP pseudo connections. Fig. 2 illustrates the comparison results of the metric Replayed Traffic Ratio.

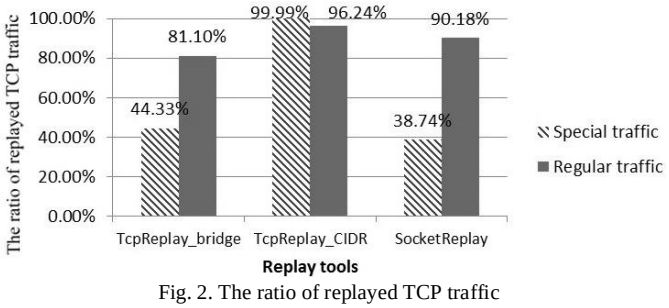


Fig. 2. The ratio of replayed TCP traffic

For regular traffic, the ratios of replayed traffic of these three types of configurations are all over 80%, namely 81.10%, 96.24%, and 90.18%, respectively. For specific traffic containing malicious contents, the differences of the ratios of replayed traffic were great, namely 44.33%, 99.99%, and 38.74%, respectively.

For Tcpreplay using bridge mode, the replay tool behaved either like a client or like a server and simply sent out the traffic solely based on the packet timestamps. Therefore, some packets

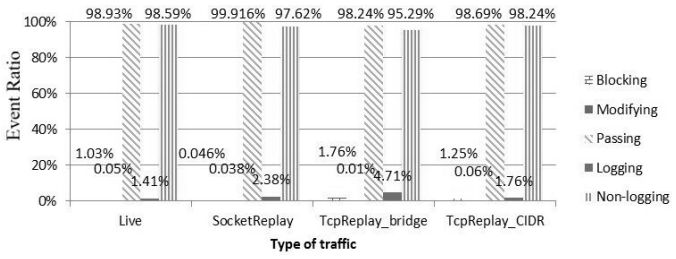
were blocked or discarded by a DUT because they didn't fit the onsite context, which resulted in a poor replayed traffic ratio. For SocketReplay, although it can recover incomplete TCP connections by inserting dummy bytes or packets, it failed to recover the traffic because some of the lost packets were critical to trigger connection events. For Tcpreplay using CIDR mode, the ratio of replayed traffic was high because it could modify the IP addresses of packets and successfully replay them to traverse the DUT.

Table II (C) shows the results of the number of reproduced events defined in Table I, and Table II (D) shows the statistics of connections and related logs of the live and the replayed traffic. The replayed traffic was obtained from using Tcpreplay using bridge mode.

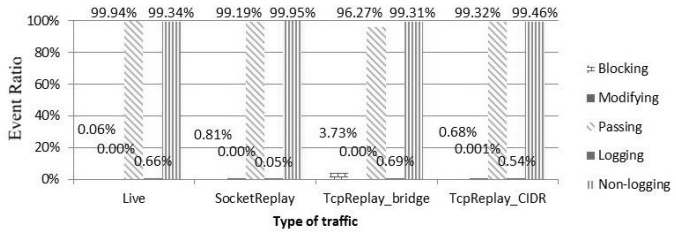
C. The Ratio of Events with Various Attributes on Live and Replayed Traffic

In this experiment, the occurrence ratios of events in T^L and T^R are calculated using regular and specific traffic based on the three metrics, namely Live Event Ratio, Replay Event Ratio, and Logging Event Ratio. We use SocketReplay and Tcpreplay to generate the replayed traffic. We then calculate the occurrence ratios of blocking, modifying, passing, logging, and non-logging events for both the live and the replayed traffic. Fig. 3 shows our results.

Fig. 3 yields several interesting observations: (1) in both types of traffic, the ratios of passing events are the highest among the five types of event attributes; (2) the ratio of blocking events in the traffic generated by Tcpreplay is higher than that in the live traffic; (3) the ratio of modifying events in the replayed traffic generated by replay tools is lower than that in the live traffic except the ratio of events generated by Tcpreplay using CIDR mode with regular traffic; (4) the ratio of logging events in the traffic generated by Tcpreplay is higher than that in the live traffic; with the ratio of logging events in the traffic generated by SocketReplay being the one exception that yields lower ratio of logging events than that in the live traffic.



(a) Specific traffic



(b) Regular traffic

Fig. 3. The occurrence ratios of events for T^L and T^R

The possible reasons are described as follows. For regular traffic, the ratio of blocking events in Tcpreplay traffic is high because Tcpreplay replayed traffic only based on the timestamps and the connection states of the replayed traffic does not necessarily conform to the TCP protocol (e.g., Tcpreplay is unable to synchronize SYN-ACKs to create valid TCP sessions); whereas the ratio of blocking events in SocketReplay traffic is low because SocketReplay keeps the connection states. For specific traffic, packets having malicious patterns in the packet payload or having incorrect contents in the packet header would trigger modifying events. Each packet of the traffic replayed by Tcpreplay using CIDR mode triggers a modifying event because the header must be modified; however, only packets with malicious contents trigger modifying events when replaying traffic using Tcpreplay in bridge mode. The modifying events of packets with incorrect header were not triggered. Similarly, each packet with malicious pattern triggers a modifying event when replayed by SocketReplay; however, packets with incorrect contents in the packet header do not trigger modifying events. The modifying events do not occur in regular traffic on the live traffic platform, but they might be triggered in replayed traffic platform (0.0001% of Tcpreplay events using CIDR mode) because of malformed packets by the DUTs.

D. Replayed Traffic Effectiveness

In this experiment, the event reproduction ratios for the traffic replayed by SocketReplay and Tcpreplay are compared and discussed. We calculate the results using Equation 2 and 3. Only traffic of high consistency are included to evaluate the event reproduction ratio. The degree of consistency is calculated based on the consistency ratio metric. Fig. 4 illustrates the consistency ratios for the traffic replayed by SocketReplay, by Tcpreplay using bridge mode, and by Tcpreplay using CIDR mode, respectively. The traffic replayed by Tcpreplay using CIDR mode had higher ratios in both specific and regular traffic.

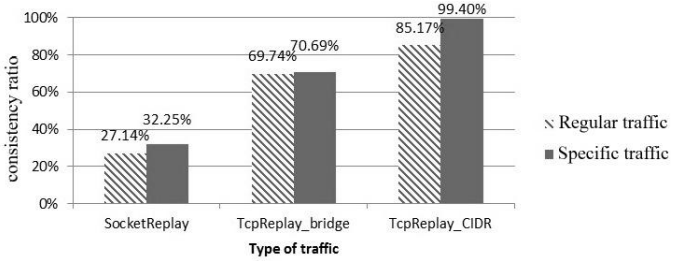
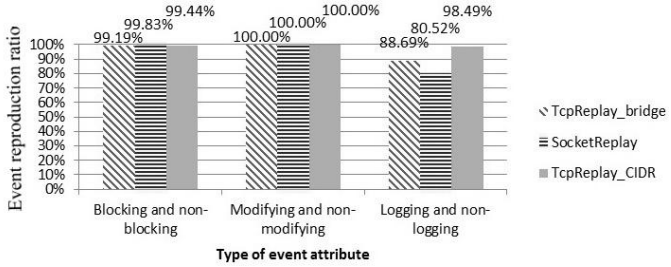


Fig. 4. The consistency ratios of three replay configurations

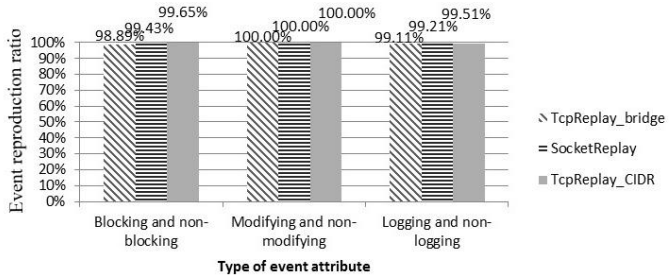
SocketReplay seeks to mimic the hosts to generate traffic without breaking protocol semantics, and seeks to reconstruct TCP connections during replay, and these properties causes the traffic replayed by SocketReplay to be inconsistent with the live traffic. The traffic replayed by Tcpreplay also incurs inconsistencies because Tcpreplay would remove the acknowledgement packets, such as TCP keep-alive messages, from the client side of the replayed traffic, and would randomly remove duplicate packets and FIN packets.

Fig. 5 illustrates the combination of event reproduction ratios

of the three replay configurations for specific and non-specific events. The event reproduction ratios of the combination of modifying and non-modifying events for both types of traffic are all 100%. The reproduction ratios of blocking and non-blocking events of the three configurations were close. However, the reproduction ratios of the combination of logging and non-logging events were different. The reproduction ratios of Tcpreplay using CIDR mode in both types of traffic were higher than the other two replay configurations; this is because Tcpreplay using CIDR mode could emulate the interaction between a client and a server and hence generates more complete logs.



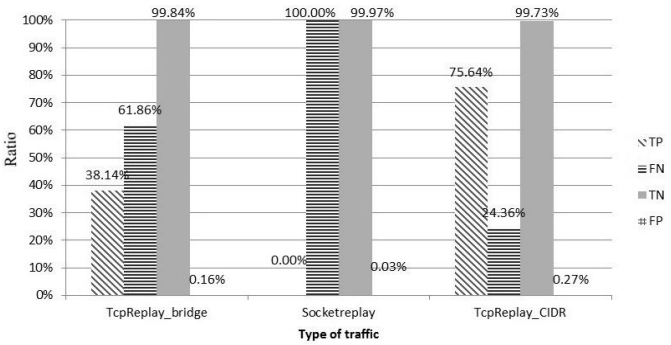
(a) Specific traffic



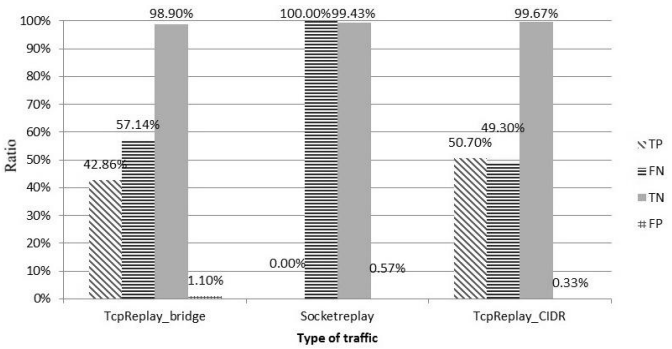
(b) Regular traffic

Fig. 5. The event reproduction ratios for specific and regular traffic

Fig. 6 shows the effectiveness of replayers for blocking and non-blocking events; the effectiveness is derived from the results of True Negative (TN), True Positive (TP), False Negative (FN), and False Positive (FP). Neither the specific nor the regular traffic replayed by SocketReplay triggers TP. The TP rates yielded by Tcpreplay using bridge mode and using CIDR mode are 38.14% and 75.64% for specific traffic and 42.86% and 50.70% for regular traffic, respectively. The TP rate yielded by Tcpreplay using CIDR mode is higher than that yielded by using bridge mode.



(a) Specific traffic



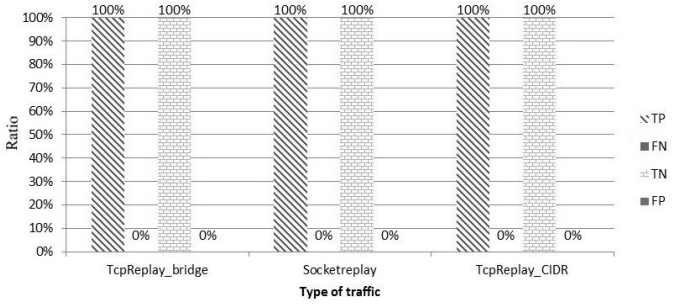
(b) Regular traffic

Fig. 6. The effectiveness of blocking and non-blocking events

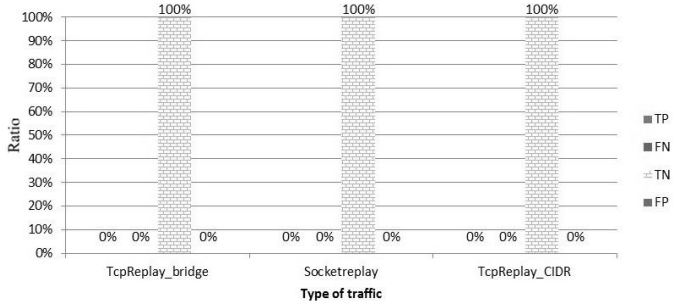
In the preprocessing phase, SocketReplay removes packets that bear TCP sequence numbers greater than or equal to the FIN packet within a connection. Blocked packets which are dropped by a DUT or a destination host in the live traffic are eliminated in this stage as well. Therefore, the TP rate of blocking events of the replayed traffic by SocketReplay is 0%. The TP rate of blocking events generated by Tcpreplay using CIDR mode was higher than that generated by Tcpreplay using bridge mode because the replayed TCP traffic ratio of the former was higher than the latter. In other words, the ratio of replayed traffic directly impacts the ratio of event reproduction.

Packets blocked in live traffic cannot be reproduced in replayed traffic, resulting in differences between the live and the replayed traffic. Therefore, the rates of FN and FP are high for all three replay configurations.

Fig. 7 illustrates the effectiveness of modifying and non-modifying events. Here only the packets with modified payload trigger the type of the modifying event. The effectiveness of the modifying events incurred by modified packet header is not calculated because it cannot trigger the type of modifying event. For specific traffic, the TP rates of the events of the traffic replayed by three replay configurations are all 100% and the TN rates are all 100% as well. For regular traffic, the TP rates of the three replay configurations are all 0%, and the TN rates are all 100%. Because none of the three replay configurations produces any modifying events, the TP rates are all 0% for regular traffic.

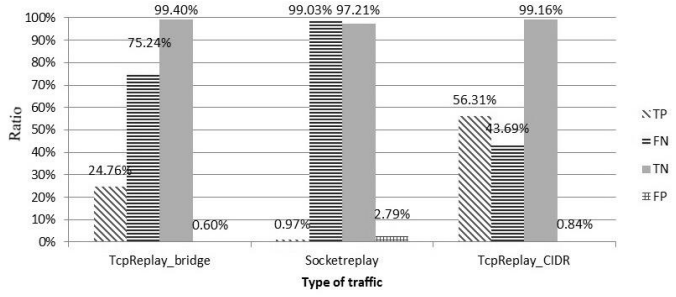


(a) Specific traffic

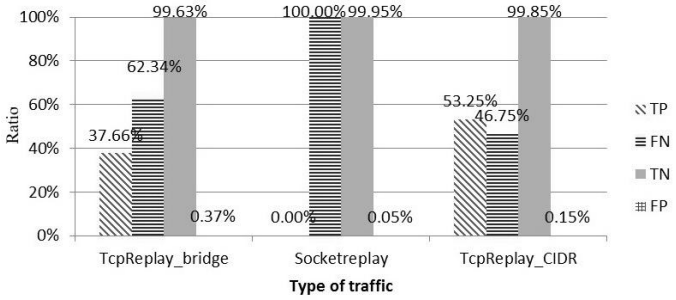


(b) Regular traffic

Fig. 7. The effectiveness of modifying and non-modifying events



(a) Specific traffic



(b) Regular traffic

Fig. 8. The effectiveness of logging and non-logging events

Fig. 8 illustrates the effectiveness of logging and non-logging events. The TP rate of the traffic replayed by SocketReplay is lower than the other two replay configurations for both specific and regular traffic. The type of logging event triggered by these three replay configurations are signature-based logging events. Tcpreplay using CIDR mode achieves 56.31% and 53.25% of TP rates for specific and regular traffic, respectively, and Tcpreplay using bridge mode achieves 24.76% and 37.66% of TP rates for specific and regular traffic, respectively. For regular traffic, some packets that are not related to the onsite context but trigger signature-based logs would be ignored by SocketReplay; therefore, SocketReplay achieved 0.97% and 0.00% of TP rates for specific and regular traffic, respectively.

The anomaly-based logging rules are developed by heuristics, and thus the activities appeared in the live and the replayed traffic does not always trigger anomaly-based logging events. Consequently, Tcpreplay using bridge mode triggers 75.24% and 62.34% of FN rates for specific and regular traffic, respectively; Tcpreplay using CIDR mode triggers 43.69% and 46.75% of FN rates for specific and regular traffic, respectively. SocketReplay does not reproduce the anomaly-based logging events, and therefore it triggers 99.03% and 100% of FN rates for specific and regular traffic, respectively.

On the other hand, replayed traffic for all three replay configurations triggers new anomaly-based logging events, and therefore all configurations generate different FP rates of logging events. The three configurations, Tcpreplay using bridge mode, Tcpreplay using CIDR mode, and SocketReplay, generate 0.60%, 0.84%, and 2.79% of FP rates for specific traffic, and 0.37%, 0.15%, and 0.05% of FP rates for regular traffic, respectively.

VI. Conclusions

This work proposes methods to measure and compare the event reproduction ratios and the effectiveness of stateful and stateless replay tools based on packet events and connection events. A stateless replayer replays network traces solely base on the timestamps without maintaining the state-dependent protocol fields while a stateful replayer updates the state-dependent protocol fields to reflect the different states of the different hosts for replay. We design test frameworks and define several metrics to differentiate the two types of traffic replayers. In our experiments, we choose Tcpreplay and SocketReplay to be representatives of a stateless replayer, and a stateful replayer, respectively.

Throughout our experiments, the event reproduction ratios are affected by replay configurations, traffic contents, and the processing rules of packets of a DUT. Results indicate that traffic contents, which have fewer incomplete connections and use fewer RST packets to terminate a connection, have higher replayed ratios of the traffic and higher event reproduction ratios. SocketReplay replayed a smaller ratio of the captured traffic and triggered fewer blocking events in subsequent replay tests. Triggering blocking events denotes the replayed traffic cannot fit the onsite context. The processing rules of a DUT were other important factors in triggering specific events. Heuristic-based rules could lead to 100% of FNs if they can't be applied to the replayer.

Therefore, the choice of a proper replayer and the corresponding replay configuration should depend on the features of captured traffic, such as the proportion of incomplete connections and the requirement to fit a certain application state machine. For example, if the effectiveness of blocking events is the concern, a stateful replayer like SocketReplay would be better than a stateless one. If the consistency ratio is the concern, Tcpreplay using CIDR mode would be the better choice.

References

[1] Tcpreplay, available: http://tcpreplay.synfin.net

[2] Ying-Dar Lin, Po-Ching Lin, Tsung-Huan Cheng, I-Wei Chen, Yuan-Cheng Lai, "Low-Storage Capture and Loss Recovery Selective Replay of Real Flows," IEEE Communications Magazine, Vol. 50, Issue 4, pp.114-121, April 2012.

[3] Seung-Sun Hong and S. Felix Wu, "On Interactive Internet Traffic Replay," Proceedings of the 8th international conference on Recent Advances in Intrusion Detection (RAID), pp. 247-264, 2005.
[4] Chia-Yu Ku, Ying-Dar Lin, Yuan-Cheng Lai, Pei-Hsuan Li, kate Ching-Ju Lin, "Real Traffic Replay over WLAN with Environment Emulation," IEEE Wireless Communications and Networking Conference (WCNC), April 2012.
[5] Tomahawk, available: http://tomahawk.sourceforge.net/
[6] Yu-Chung Cheng, Urs Holzle, Neal Cardwell, Stefan Savage, and Geoffrey M. Voelker, "Monkey see, Monkey Do: A tool for TCP Tracing and Replaying," Proceedings of 2004 USENIX Annual Technical Conference, June 2004.
[7] Weidong Cui, Vern Paxson, Nicholas C. Weaver, Randy H. Katz, "Protocol-Independent Adaptive Replay of Application Dialog," Proceedings of Network and Distributed System Security Symposium (NDSS), 2006.
[8] James Newsome, David Brumley, Jason Franklin, and Dawn Song, "Replayer: automatic protocol replay by binary analysis," Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS), 2006.
[9] Tcpprep, available: http://tcpreplay.synfin.net/wiki/tcpprep
[10] Snort Rules, available: http://snort.org/snort-rules/
[11] Classless Inter-Domain Routing (CIDR), RFC 4632, available: http://tools.ietf.org/html/rfc4632.
[12] PC Flank, available: http://www.pcflank.com
[13] AuditMyPC, available: http://www.auditmypc.com
[14] Security Space, available: http://www.securityspace.com
[15] eicar, available: http://www.eicar.org
[16] SpinRite, available: http://www.grc.com

Ying-Dar Lin is Professor of Computer Science at National Chiao Tung University (NCTU) in Taiwan. He received his Ph.D. in Computer Science from UCLA in 1993. He served as a visiting scholar at Cisco Systems in San Jose during 2007–2008. Since 2002, he has been the founder and director of Network Benchmarking Lab (NBL), which reviews network products with real traffic. His research interests include quality of services, network security, deep packet inspection, P2P networking, and embedded hardware/software co-design. His work on "multi-hop cellular" was the first along this line, and has been cited over 600 times and standardized into IEEE 802.11s, WiMAX IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Fellow and currently on the editorial boards of several IEEE journals and magazines. He published a textbook "Computer Networks: An Open Source Approach", with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).

Chun-Nan Lu received his B.S. and M.S. degrees in Computer Science from National Tsing Hua University in 2000 and 2002. He is a Ph.D. student in Computer Science at National Chiao-Tung University. His researches focus on network security and traffic measurement/analysis. He can be reached at cnlu@cs.nctu.edu.tw.

Jose Miguel Sagastume Jacobo received bachelor's degree with a major in Computer Science which was awarded to him by Don Bosco University in EI Salvador in 2009. He received his master degree in Department of Computer Science in National Chiao Tung University in 2012. He hold a certification on Cisco Certified Network Associate CCNA and CCNA Security. His research interests includes network testing.

Jui-Tsun Hung received his M.S. and Ph.D. degrees in Electrical and Computer Engineering from the State University of New York at Stony Brook, New York, USA, in 2001 and 2003, respectively. In 2004, he joined Memes Technology as a senior engineer in IC design for audio broadcast systems. He was a visiting scholar in Computer Science, Stony Brook University, USA, during 2006-2009. Since 2009, he has been a Post Doc researcher in Computer Science, National Chiao Tung University, Taiwan. His current research interests include computer networks, computer architecture, wireless communications, and signal processing.

Yuan-Cheng Lai received the Ph.D. degree in Computer Science from National Chiao Tung University in 1997. He joined the faculty of the Department of Information Management at National Taiwan University of Science and Technology in 2001 and has been a professor since 2008. His research interests include wireless networks, network performance evaluation, network security, and content networking. He can be reached at laiyc@cs.ntust.edu.tw.