國立交通大學

電機資訊國際學程

碩士論文

網路流量重播效果之評估

Effectiveness in Replaying Real Traffic: An Evaluation

研究生：沙荷西

指導教授：林盈達

中華民國 一百零一年八月

網路流量重播效果之評估

Effectiveness in Replaying Real Traffic: An Evaluation

研 究 生：沙荷西　　**Student：Jose Miguel Sagastume Jacobo**

指導教授：林盈達　　**Advisor：Dr. Ying-Dar Lin**

國 立 交 通 大 學

電 機 資 訊 國 際 學 程

碩 士 論 文

A Thesis

Submitted to EECS International Graduate Program

National Chiao Tung University

In Partial Fulfillment of the Requirements

For the Degree of

Master

August 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年八月

網路流量重播效果之評估

學生: 沙荷西　　指導教授: 林盈達

國立交通大學電機資訊國際學程

摘要

對於網絡安全產品的測試，真實流量的錄製和重播是很重要的。然而，在相同的測試情境中，重播流量應能有效地重製在實際流量中由待測物所引發的事件。這個研究提出方法來計算封包或連線事件的事件重製率和重播工具的有效性。在這項研究中使用了 SocketReplay 及 Tcpreplay。結果表明，流量內容和重播策略和待測物的通過規則，可以顯著地影響事件的重播率和重播工具的有效性。例如，流量中含有很多不完整的連接，或重播策略是以連線為基準，而不是時間戳記為基準，將會大大地減低事件的重播率和的重播工具的有效性。結果，雖然 SocketReplay 可以準確地建立正確的 TCP 對話，可是 SocketReplay 的事件重播率只達到 38.74% 的 TCP 流量，導致對封包傳遞和阻止事件的有效性分別為 99.97% 和 0.00%，而 CIDR 模式的 Tcpreplay 的事件重播率達到了 99.99% 的 TCP 流量，導致對封包傳遞和阻止事件的有效性分別為 99.73% 和 75.64%。此處，錄製的的流量有很多不完整的連接且事件的觸發是基於啟發式或識別的規則。適當重播工具的選擇及重播政策的選定，應取決於所我們錄製的流量內容，以避免事件重製率與重播工具的有效性出現顯著的下跌。

關鍵字: 網路測試、流量重播、事件重製率、有效性

Effectiveness in Replaying Real Traffic: An Evaluation

Student: Jose Miguel Sagastume Jacobo Advisor: Dr. Ying-Dar Lin

Department of Electrical Engineering & Computer Science / International Graduate Program

National Chiao Tung University

Abstract

Capturing and replaying real flows are important for testing network security products. However, under the same testing scenario, replayed traffic should effectively reproduce the events triggered by DUTs as the live traffic. This work presents methods to calculate the event reproduction ratio and the effectiveness of replay tools, based on packet events and connection events. The stateful SocketReplay and the stateless Tcpreplay were applied in this study. Results indicated that the traffic contents, the replay policies, and DUT filtering rules can significantly affect the event reproduction ratio and the effectiveness of replay tools. For example, traffic with a lot portion of incomplete connections and replay policies based on connections, rather than timestamp, can considerably impair the event reproduction ratio and the effectiveness of replayers. The results show that SocketReplay, which can accurately establish the correct TCP session, can only replay 38.74% TCP traffic, resulting in 99.97% and 0.00% of effectiveness of passing and blocking event ratio, respectively, while Tcpreplay with CIDR mode can replay 99.99% TCP traffic, resulting in 99.73% and 75.64% of effectiveness of passing and blocking event ratio, respectively, when captured traffic have many incomplete connection and events are triggered by heuristic based rules and signature based rules. The choice of a proper replayer and its replay policies should depend on the traffic contents we captured to avoid a significant drop of event reproduction ratio and the effectiveness of replayers.

Keywords: network testing, traffic replay, event reproduction ratio, effectiveness

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

The testing of network devices has been a major focus on the network research area. Its goal is to create a range of test scenarios similar to those experienced in the live deployment. The importance of network device testing is to debug network device problems in a controlled, transparent test bed with reproducibility of errors. One method for network device testing is generating or replaying traffic by particular tools to check the behaviors of DUTs (Devices Under Test).

The traffic that is produced by the tools used on the network device test can be classified as: Model-base and Trace-based. The first type of traffic is based on mathematical models to generate artificial network traffic. The second type of traffic is based on real traffic captured from live deployment. The tools that generate model-based traffic, are easy to implement, however it is limited by numerical properties found in the mathematical model. The Trace-based traffic is produced by real network traces. Thus including all properties found in live deployment.

Replay tools that produce trace-base traffic can be either stateless or stateful. A stateless replay tool replays network traces based only on timestamps. The content of replayed traffic is the same as that captured in the network traces. Because of the characteristics of stateless replay tool, the traffic replayed by these replay tools cannot be understood correctly by the DUTs, which keep track of the state of network connections (such as TCP streams, UDP communication) traveling across the DUT. Therefore seems to be inaccurate. On the contrast, stateful replay tools modify the content of network traces to fulfill the test conditions for the DUT. The stateful replay tools will modify the content of the network traces depending on the type of network layers the DUT works, and also it might changes the content of the network traces according to the responses of the DUTs. An example of a stateful replay tool is NATReplay, this stateful replay tool supports traffic replay for NAT devices. This tool maintains a mapping NAT state table between private source sockets (including IP addresses and port numbers) and public destination sockets.

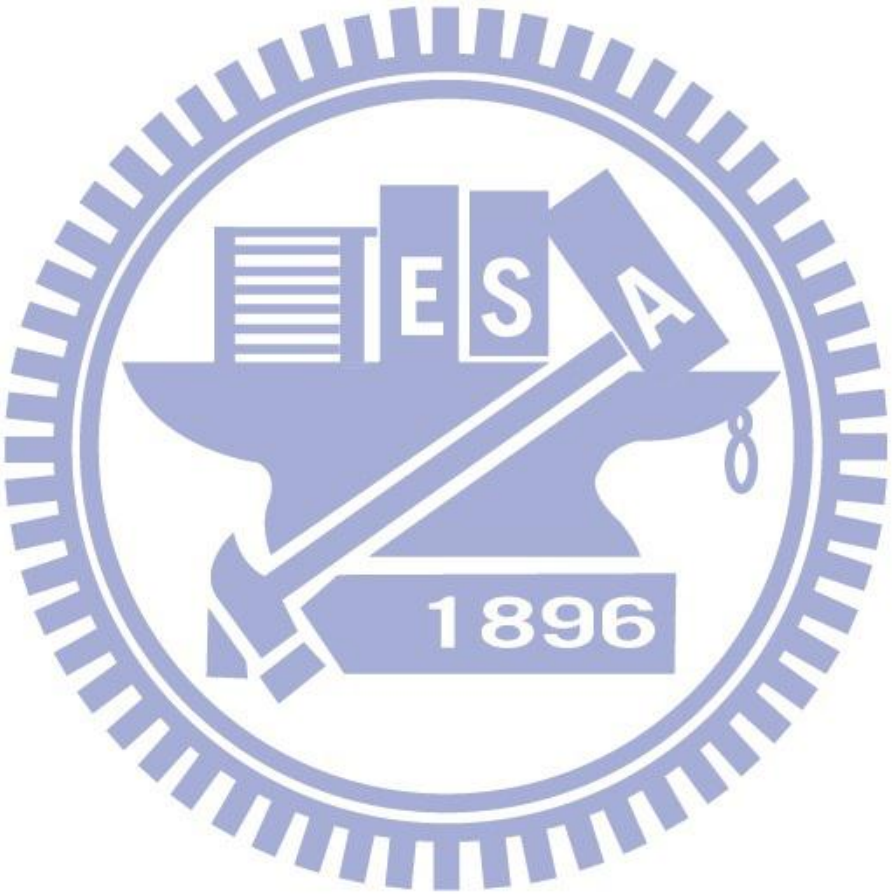
1.1. The effectiveness of Replay Tools

For replay tools replay the network traces correctly to the DUTs and generate the same events in live traffic such as: blocked packets, unblocked packets, modified packets, unmodified packets, logged connections and unlogged connections. They must replay traffic that the DUTs will accept and keep replaying traffic conditions as live traffic to produce the same events on the DUT that also occurred on live traffic at live deployment. The effectiveness is based on the reproduction ratio of events on the DUTs for live traffic and replayed traffic. The effectiveness of replay tools are difficult to calculate because of the complexity of traffic replayed and also because of the different reactions this traffic receives for different type of DUTs. Hence a specific replay tool is required for specific types of DUT. For instance, NATReplay is a replay tool for testing NAT devices.

The effectiveness of replay tools has different approaches, these are depending on the replay tool in use. Most replay tools have different parameters to test the performance. TCPopera [2] measures the replayed traffic flows using statistical methods based on short-term profile and long-term profile, number of packet reordering and session duration. WirelessReplay [3] uses the connection states of the protocol 802.11 to define different events. Using the events the replay tool replays the traffic, with this method the effectiveness is measured by the reproduction rate of the events on replayed traffic. SocketReplay [1] measures the effectiveness, using the reproduction rate of the triggered attack session on a security appliance produced by replayed traffic. Proxy Replay [8] measures it uses the request-response pairs ratio acknowledges by the proxy.

While most of the previous studies on replay tools try to find out the effectiveness of replay tools using different methods [1] [2] [8], only [3] focus on the DUT reactions to the replayed traffic. The purpose of our work is to create a general framework that measures effectiveness of replay tools based on the events occurred inside the DUTs. The effectiveness is specified by comparing the number of events occurred in live traffic and with that occurred in replayed traffic. To measure the reproduction ratio on live and replay traffic, events are categorized into three different attributes blocked, modified and logged events. This approach provides a method to check the responses of DUT on live and replayed traffic.

The rest of this paper is organized as follows. Chapter 2 introduces the background of stateless and stateful replay tools. Chapter 3 describes the general framework, terminology definitions and problem statement. Chapter 4 describes the event comparison issues and the solution to measuring the events. The implementation, data analysis and issues with the reproduction ratio approach are described in Chapter 5. We conclude our work in Chapter 6.



Chapter 2: Background

Frequently network device tests are performed using real network traces. The real network traces are used by replay tools to replay traffic. Replay tools can be categorized into stateless and stateful. Replay tools are used to test devices such as switches, routers, gateways, firewalls, network intrusion detection or prevention systems, and proxy application servers.

Table 1 lists some replay tools and their functionalities. It shows the different network layer a replay tools focus.

Table 1: Replay tool differences

Replay tools	Functionality	Network Layer	Categorizes
WirelessReplay [3]	Maintain state of the protocol 802.11 protocol	Layer 2	Stateful
TCPreplay [5,6]	Divide traffic into server and client side	Layer 3	Stateless
Tomahawk [17]	Enables retransmission capabilities	Layer 3	Stateless
NATReplay [7]	Keep network address translation table state	Layer 4	Partial Stateful
SocketReplay [1]	Keep TCP states	Layer 4	Stateful
ProxyReplay [8]	Keep the correct message procedure for application proxies.	Layer 7	Stateful

2.1. Stateless Replay Tools

TCPreplay replays real network flows that were captured from real networks. Replayed traffic is completely stateless. Stateless traffic means the replay tool was unable to update the content of captured traffic (example TCP sequence number and the acknowledgement number to maintain the TCP protocol state). The goals of this replay tool are: replay traffic flows at arbitrary speeds, implement sniffing mode (replays traffic with only one interface), replays traffic using simulated client and simulated server; however, this method doesn't update the information of the TCP stream. Tcpreplay replays traffic for client side and server side using an external tool called Tcpprep [16]. One side includes packets from the client side to the server side, while the other side includes packets from the server side to the client side.

Tomahawk works similarly to TCPreplay, but with additional functionalities: (1) automatically divide the traffic for client and server side between two interfaces, (2) uses the mechanism of the window scale to send packets, (3) allow retransmission

where packets dropped because of network congestion. Also it can detect packet blocking or dropping.

2.2. Stateful Replay Tools

Table 2 shows the modifications a packet experiences through different types of stateful network devices. Forward packets only update the IP header fields, and NAT mechanism updates IP and transport layer. Security appliance might delete some application data. Application proxy works on behalf of the clients; however, the application data passing through DUTs are modified.

Table 2: Network packets changes on network devices

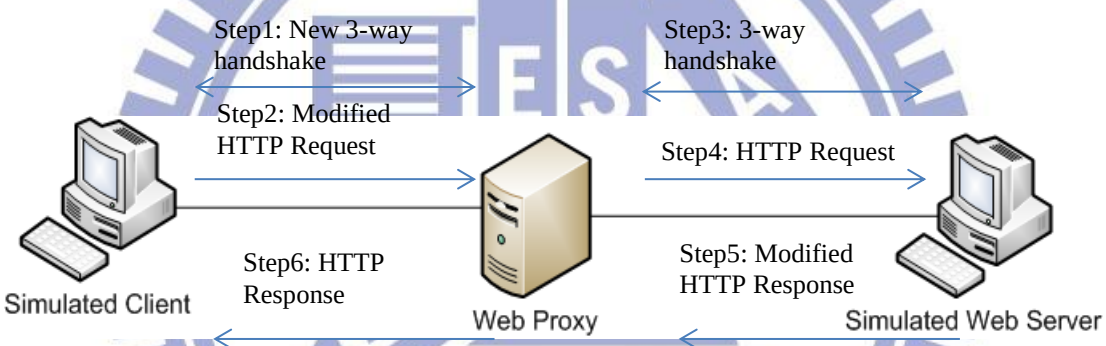
Process	Packet Forwarding	NAT	NAPT	ALG in NAT	Security Appliance	Application Proxy
Packet length				X		
Time to live	X					
Header checksum	X	X	X	X		
Source IP address		X	X	X		
Destination IP address		X	X	X		
IP payload						
TCP Source Port #			X	X		
TCP Destination Port #			X	X		
TCP Sequence #				X		
TCP Acknowledgment #				X		
TCP checksum			X	X		
TCP Data				X	X	X
UDP Source Port #			X	X		
UDP Destination Port #			X	X		
UDP checksum						
UDP Data				X	X	X

SocketReplay replays real network flows and it also adapts to the influence of the DUTs with firewall capabilities. The importance of the replayer is to preprocess TCP and UDP packets before being replayed. SocketReplay maintains the connection state of TCP by creating new socket connections. Thus the modification of the content at network traces can fit the test requirement of the DUT and avoid the generation of ghost packets.

NATReplay is used to test the DUT with NAT function. It keeps states of basic NAT (private address, public IP address) and NAPT (private address, private transport port number, global IP address, and global transport port number) translation.

Maintaining the stateful state of NAT is not easy as table 2 shows that many modifications on real flows might occur.

ProxyReplay replays real traffic for application-aware proxies, as shown in Table 1. The importance of the replayer is to preprocess the network traces to adjust to the test condition for non-transparent devices, such as servers, and for transparent devices, such as a firewall or gateway products which act as a router between LAN and WAN areas. This procedure includes creating connections, accepting connections, and ignoring some connections. Also adding or modifying messages at the application level on client-proxy and server-proxy sides. Thus, the correct message is sent to the proxy server. Figure 1 shows the procedure of replaying real network flows to a web proxy.



WirelessReplay replays real network flows to a wireless access point (AP), as shown in Table 1. The replay tool keeps states of the protocol 802.11 process as follows: (1) location process of the AP by passive scanning or active scanning, (2) authentication process of the interchange of information between the AP and the station where each side proves the knowledge of a given password, (3) association process of the exchange of information about the stations and basic service set (BSS) capabilities.

Table 3 shows the differences between a stateful replayer and a stateless replayer. Testing mode refers the types of test that are possible for the replay tools. Statefulness defines until what network layer the traffic is understood by a stateful device. Features describe functionalities of each replayer. Action and comments gives inside of the replayers.

Table 3: Stateful vs Stateless Replayer

Replayer	**Tcpreplay**	**SocketReplay**
Testing mode	Sniff, inline	Sniff, inline
Statefulness	Stateless	Layer 4
Features	Activate traffic flows with timestamp and without modification to the original traffic	Packet loss recovery Stateful TCP connections
Action	N/A	Reconstruct TCP connections
Comments	High throughput	Adapt to the behavior of a stateful DUT Event accuracy: connection state, sequence number

2.3. Other Related Works

Replay tools preprocess captured traffic to fit the requirement of the test environment. Preprocessing stage is done before the replay tool starts to replay traffic [1] [7] [8]. Depending on the type of preprocessing stage, the changes in the traffic are different. The type preprocessing stages are: modification of the IP header, transport header or modification of the content of a packet.

Table 4 shows the description of each replay tool with the preprocessing stage and also shows if the process is required or no.

Table 4: Different preprocessing stages

Replay tools	**Modified fields**	**Required to work**
Tcpreplay [5]	MAC address, IP address, port numbers	No
SocketReplay [1]	New IP header and transport header	Yes
ProxyReplay [8]	Application message	Yes
NATReplay [7]	IP address, port numbers	Yes
Tomahawk [17]	MAC address, IP address, port numbers	No

Chapter 3: Problem Statement

This chapter describes the framework of this work, the definitions of terminologies specific to this work and discusses the problem statement.

3.1. Framework

Figure 2a and 2b show the test beds for capturing live traffic and replayed traffic. The test bed has four major components, namely the DUT, TG (the traffic generator), TM (mirror traffic devices) and TR (the traffic recorder device). Types of DUTs are a router, a firewall, or a proxy server. TGs include the Internet network, the local network, and replay tools. TM is a switch and TR is a server that captures live traffic and replayed traffic. The server also records live logs and log under replay scenario from DUTs.

In live traffic test bed, we capture the traffic from the external network (Internet) and from the internal network (NCTU lab). The external network and the internal network are dependent. Those networks are dependent because reply messages from the external network reliant for a request message on the internal network.

The traffic between the internal network and the DUT is called client side of the traffic and the traffic between the external network and the DUT is called server side of the traffic.

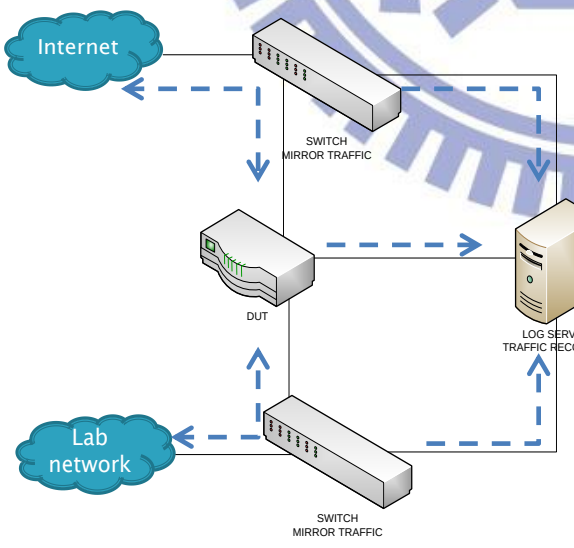


Figure 2a: Live traffic test bed

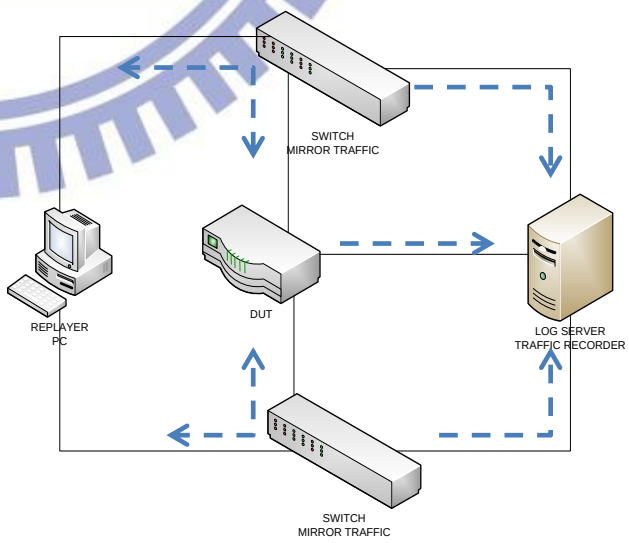


Figure 2b: Replayed traffic test bed

3.2. Terminologies Definitions

In this work the term *event* is defined as an occurrence of a packet or connection traversing a network device with some attributes of how the packet or connection was processed. For packet events, there are two specific attributes--blocked-(b), modified-(m). Packets neither blocked nor modified are identified as events with non-specific attribute--passed. For connection events, each connection may or may not generate a log. If a connection generates a log, a specific event with logged attribute occurs. On the other hand, if a connection does not yield a log, the event is identified as non-specific event with non-logged attribute. Therefore, five types of event attributes are shown in Figure 3.

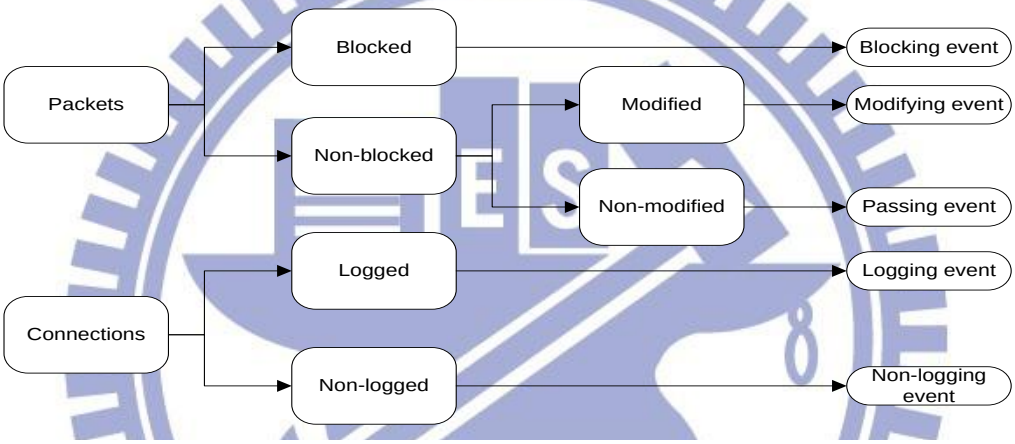


Figure 3: Illustration five event attributes

Detailed description about modification events and logging events are shown in Table 2 and Table 5.

Table 5: Type of logs

Network devices	Description	Type of logs
Router	In-Kernel	Forwarding status
NAT	In-Kernel connection tracking system	Status: new, established, related, invalid [18]
Security appliance	Application level	IDPS (Intrusion Detection and Prevention System), ADP (Anomaly Detection and Protection), firewall, anti-spam, and antivirus logs
Application Proxy	Application level	Client status, server status, and cache status

Table 6 is a description of a list of notations that are further used in our work. T^L *(live traffic)* is used to refer to the network traffic flowing in the live network. T^R *(replayed traffic)* is used to refer to the traffic replayed by replay tools and flowing on

the replay network in the lab. T_c^L refers to the live network traffic of the client side of the traffic and T_s^L refers to the live network traffic of the server side of the traffic. T_c^R refers to the replayed network traffic of the client side of the traffic and T_s^R refers to the replayed network traffic of the server side of the traffic. $T_c^{L,\beta}$ and $T_s^{L,\beta}$ are used to refer to the live network flow from the client to the DUT before being processed by the DUT and the live network flow from the server to the DUT before being processed by the DUT. $T_c^{L,\alpha}$ and $T_s^{L,\alpha}$ are used to refer to the replayed network flow from the client to the DUT after being processed by the DUT and the replayed network flow from the server to the DUT after being processed by the DUT. $T_c^{R,\beta}$ and $T_s^{R,\beta}$ have the same meaning as $T_c^{L,\beta}$ and $T_s^{L,\beta}$, but those describe the replayed network flows. $T_c^{R,\alpha}$ and $T_s^{R,\alpha}$ have the same use as $T_c^{L,\alpha}$ and $T_s^{L,\alpha}$, but those describe the replayed network flows. $C_c^{L,\alpha}$, $C_c^{L,\beta}$, $C_s^{L,\alpha}$ and $C_s^{L,\beta}$ are used to refer a connection set from $T_c^{L,\beta}$, $T_s^{L,\beta}$, $T_c^{L,\alpha}$ and $T_s^{L,\alpha}$. $C_c^{R,\alpha}$, $C_c^{R,\beta}$, $C_s^{R,\alpha}$ and $C_s^{R,\beta}$ are used to refer a connection set from $T_c^{R,\beta}$, $T_s^{R,\beta}$, $T_c^{R,\alpha}$ and $T_s^{R,\alpha}$. E^b refers a set of *i*-th element of e_i^b, where the value of the element e_i^b could be zero (non-blocking) or one (blocking). E^m refers a set of *i*-th element of e_i^m, where the value of the element e_i^m could be zero (non-modifying) or one (modifying). E^l refers a set of *i*-th element of e_i^l, where the value of the element e_i^l could be zero (non-logging) or one (logging). L^L and L^R are used to refer to logs generated by live traffic and logs generated by replayed traffic.

Table 6: Notations Description

Notation	Description
T^L	Live traffic trace.
T^R	Replayed traffic trace.
T_c^L, T_s^L	Client side of the live traffic and server side of the live traffic
T_c^R, T_s^R	Client side of the replayed traffic and server side of the replayed traffic
$T_c^{L,\beta}$, $T_s^{L,\beta}$	The flow from the client to the DUT before being processed by the DUT and the flow from the server to the DUT before being processed by the DUT
$T_c^{L,\alpha}$, $T_s^{L,\alpha}$	The flow from the client to the DUT after being processed by the DUT and the flow from the server to the DUT after being processed by the DUT
$T_c^{R,\beta}$, $T_s^{R,\beta}$	The flow from the client to the DUT before being processed by the DUT and the flow from the server to the DUT before being

	processed by the DUT
$T_c^{R,\alpha}$, $T_s^{R,\alpha}$	The flow from the client to the DUT after being processed by the DUT and the flow from the server to the DUT after being processed by the DUT
$C_c^{L,\alpha}$, $C_c^{L,\beta}$, $C_s^{L,\alpha}$, $C_s^{L,\beta}$	Set of connection created by $T_c^{L,\beta}$, $T_s^{L,\beta}$, $T_c^{L,\alpha}$ and $T_s^{L,\alpha}$
$C_c^{R,\alpha}$, $C_c^{R,\beta}$, $C_s^{R,\alpha}$, $C_s^{R,\beta}$	Set of connection created by $T_c^{R,\beta}$, $T_s^{R,\beta}$, $T_c^{R,\alpha}$ and $T_s^{R,\alpha}$
$E^b = \{e_i^b,\ i = 1\ \ldots n\}$	Set of blocking event with *i*-th event.
$E^m = \{e_i^m,\ i = 1\ \ldots n\}$	Set of modifying event with *i*-th event.
$E^l = \{e_i^l,\ z = 1\ \ldots n\}$	Set of logging event with *z*-th event.
L^L, L^R	Logs generated by live traffic and logs generated by replayed traffic

Live traffic and replayed traffic can generate a number of events while testing a DUT. T^L may trigger events with blocked (b), modified (m) and logged (l) attributes, so on T^R. Events occurring in T^R are compared with those in T^L, and outcomes of the comparison test are 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 7.

Table 7: The truth and falseness of event reproduction with specific attributes

Type/Traffic	Live	Replay	Outcome of comparison
Event with specific attributes	1	1	e_{TP}
	1	0	e_{FN}
Event without specific attributes	0	0	e_{TN}
	0	1	e_{FP}

In Table 7, an event with a specific attribute is represented by one (1), and that without any specific attribute, zero (0). Events without specific attributes can be identified as unblocked, unmodified, or non-logged events. If an event is reproduced successfully, the outcome is either e_{TP} or e_{TN}. On the contrary, the outcome would be represented by e_{FN} or e_{FP}.

Here the event reproduction ratio includes blocking reproduction ratio (br), modifying reproduction ratio (mr), logging reproduction ratio (lr), passing reproduction ratio (pr), and non-logging reproduction ratio (nlr). The *event*

reproduction ratio is the reproduction of events with specific attributes and the events without specific attributes. The following formula *Reproduction_ratio* is

$$Reproduction\_ratio = \frac{\sum_{i=1}^{n} \neg(e_i \oplus e_i')}{n} \times 100\%, \quad (1)$$

where e_i represents the *i-th* event on live traffic, and e_i' the corresponding event of replayed traffic. Variable i is the index of events and variable n is the total number of packet events or connection events on a network trace.

In this work replay effectiveness signifies an event that occurred in live traffic is reproduced in replayed traffic. Replay effectiveness is the reproduction of events with specific attributes (blocking, modifying, or logging) and, and non-specific attributes (unblocked, unmodified and non-logged). To calculate the effectiveness of reproduction of events with specific attributes, we apply the formula *TP_Rate*. The formula *TP_Rate* is

$$TP\_Rate = \frac{\#e_{TP}}{\#e_{TP}+\#e_{FN}} \times 100\%, \quad (2)$$

where variable $\#e_{TP}$ is the number of events with one type of attributes in live traffic that are reproduced in replayed traffic. The variable $\#e_{FN}$ is the total number of events with the same type of attributes that do appear in live traffic, but they are not reproduced in replayed traffic. To calculate the error rate of event reproduction with specific attributes, the error rate (FN) is calculated by one minus the TP rate.

To calculate the effectiveness of reproduction of events without specific attributes, we apply the formula *TN_Rate*. The formula *TN_Rate* is

$$TN\_Rate = \frac{\#e_{TN}}{\#e_{TN}+\#e_{FP}} \times 100\%, \quad (3)$$

where variable $\#e_{TN}$ is the number of events without specific attributes in live traffic, but are reproduced in replayed traffic. The variable $\#e_{FP}$ is the number of events without specific attributes that occur in replay traffic but do not appear in live traffic. To calculate the error rate of event reproduction with non-specific attribute, the error rate (FP) is calculated by one minus the TN rate.

In this work the term consistency refers to the condition that a connection in a replay trace has the same number of packets as that of its corresponding connection in its live trace. Otherwise, the replayed connection is not consistent with its

corresponding connection. Here, duplicated packets are not taken into account. We use the formula *Consistency_Ratio* to measure the degree of consistency of a replayed network trace. The formula *Consistency_Ratio* is

$$Consistency\_Ratio = \frac{\sum_{j=1}^{m} c_j}{m} \times 100\% \quad (4)$$

where c_j denotes the j^{th} consistent replayed connection, and j connection, and m is the index of a replayed is the total number of replayed connections. The value c_j could be zero (without consistency) or one (with consistency).

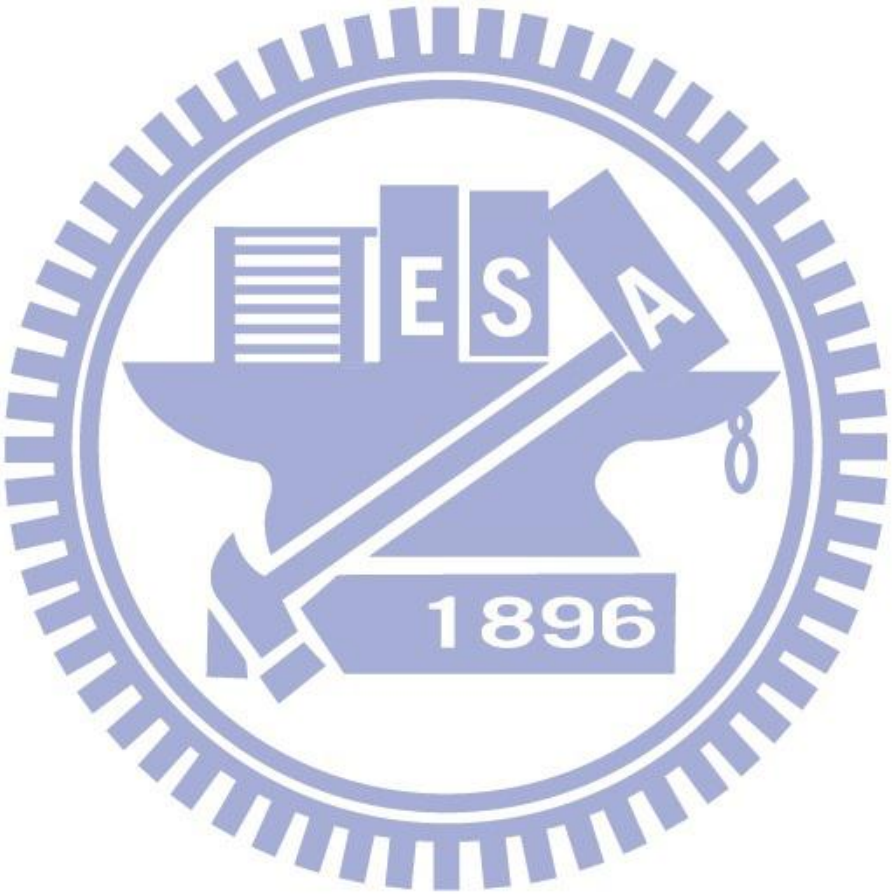
The formula of the event reproduction ratio, the effectiveness of event reproduction, and the consistency of replayed traffic are used to measure the effectiveness of replay tools.

3.3. Problem Statement

To measure the effectiveness of replay tools, we need to identify if events in live traffic are reproduced in the replayed traffic. A formal statement to describe the events in live and replayed traffic to make sure if the events are reproduced is needed. The live traffic may produce a sequence of live events E^L, while the replayed traffic, E^R. If the value of the i^{th} event of E^L *or* E^R is 0, the event does not have any specific attributes. On the contrary, if the value of an event is 1, the event has one of the specific attributes--blocking, modifying, or logging.

According to E^L and E^R, the problem statement can be described as follows. Given a captured T^L and its corresponding triggered L^L, a series of events E^L with various attributes occur, such as blocking events $E^{L,b}$, modifying events $E^{L,m}$, logging event $E^{L,l}$, or non-specific events. The capture T^L is then replayed by replay tools. This may trigger a series of events E^R with various attributes, such as a blocking event $E^{R,b}$, modifying event $E^{R,m}$, logging event $E^{R,l}$, or non-specific events. To compare the events occurring on live traffic and replayed traffic, the number of E^L should be as consistent as possible with the number of E^R. Equation (4) is used to quantify the consistency ratio of replayed traffic.

The objectives of this work are (1) to calculate event reproduction ratios, and (2) to calculate the effectiveness of the event reproduction with specific or non-specific attributes in the replayed traffic T^R.



Chapter 4: Effectiveness of Replayed Traffic

4.1. Event Comparison Issues

There are three issues that make it very difficult to compare events between the live traffic and replayed traffic. We need to solve the issues on network behaviors, replayed network traffic trace, and traffic identification. They are discussed as below.

The issue on network behaviors: network behavior of T^L and T^R affect the comparison of events between them. Network behaviors are packet loss, duplicate packets and packets out of order. By comparing the events from a live connection and reproduced in a replayed connection. If packets within these connections are affected by the behavior of the network then it is likely that the comparison of events is incorrect. Thus, affected connections by network behavior are not usable to apply equation (1).

The issue on replayed traffic trace: before capturing, the traffic traces to be replayed by replay tools must not be processed by a DUT. Processed traces may have the modified packets or lack blocked packets. Therefore, processed traces cannot reproduce the same events on replay scenarios. Traffic traces used by a replay tool should be able to reproduce the same events occurred on T^L.

The issue of traffic identification: Each event on live traffic and replayed traffic requires an identifier, especially when live traffic is different from replayed traffic. The differences between live traffic and replayed traffic can be of IP addresses and port numbers. Identifiers are used for comparing events in live traffic with those in replayed traffic.

4.2. Solution to Measuring the Effectiveness of Replayed Traffic

Figure 4 shows the solution process into four phases. The goal of the all the process is to compare the events E^L and E^R. Each phase is described as follows.

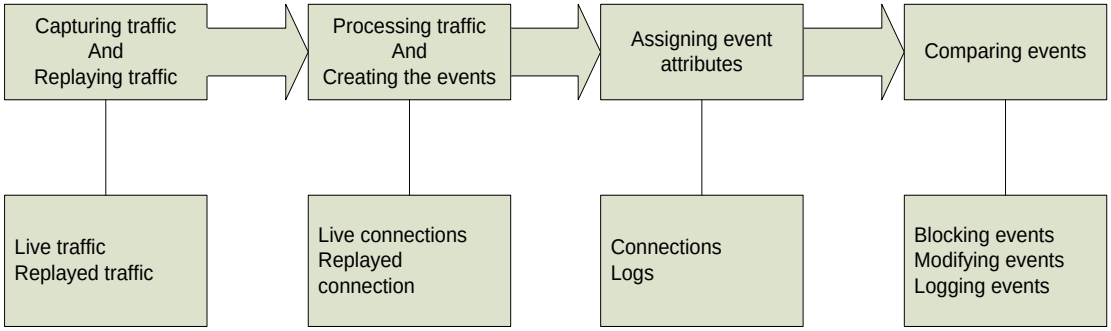
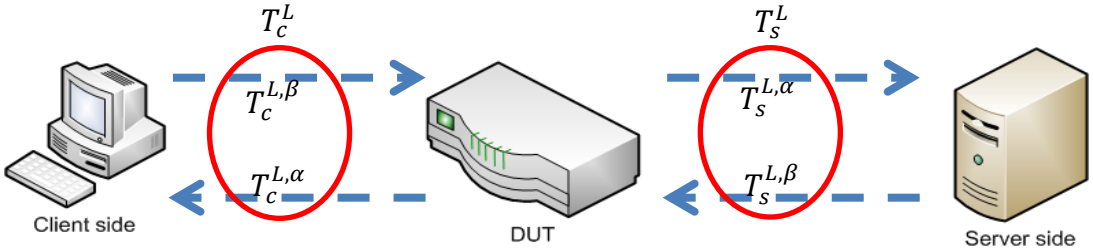


Figure 4: Solution process

To explain how to capture traffic, we illustrated the traffic flows in Figure 5. There are two traffic flows T_c^L and T_s^L. Each flow is divided into two traffic flows. Flow T_c^L is divided into flow $T_c^{L,\beta}$ and $T_c^{L,\alpha}$. And flow T_s^L is divided into flow $T_s^{L,\beta}$ and $T_s^{L,\alpha}$.



1) Capturing traffic process: T^L *and* T^R are captured. And L^L from live traffic and L^R from replayed traffic are recorded.

Steps to obtain the traffic trace for replay tools. First, capture the live traffic flows, T_c^L and T_s^L, and record L^L. Second, split the traffic flow T_c^L into $T_c^{L,\beta}$ and $T_c^{L,\alpha}$. Similarly, the traffic flow T_s^L is separated into $T_s^{L,\beta}$ and $T_s^{L,\alpha}$. Third, merge $T_c^{L,\beta}$ and $T_s^{L,\beta}$. The merged traffic flow is a traffic trace without being processed by DUT, and will be used for replay testing. On the replay test bed, the first and second steps were applied to the replay traffic traces, and the traffic flows $T_c^{R,\beta}$, $T_c^{R,\alpha}$, $T_s^{R,\beta}$, and $T_s^{R,\alpha}$ were obtained for further event reproduction analysis.

2) Processing live traffic and replayed traffic. Each traffic trace is divided into connections. In other words, network traces $T_c^{L,\alpha}$, $T_c^{L,\beta}$, $T_s^{L,\alpha}$, $T_s^{L,\beta}$, $T_c^{R,\alpha}$, $T_c^{R,\beta}$, $T_s^{R,\alpha}$ and $T_s^{L,\beta}$ are further processed into sets of connections, $C_c^{L,\alpha}$, $C_c^{L,\beta}$, $C_s^{L,\alpha}$, $C_s^{L,\beta}$, $C_c^{R,\alpha}$, $C_c^{R,\beta}$, $C_s^{R,\alpha}$ and $C_s^{R,\beta}$. The connections are identified by 5-tuple (Src IP, Dst IP, Src Port, Dst Port, Proto), and the packets within each connection is identified by IP identification number, TCP sequence number, and packet payload.

Traffic used to create live events and replay events. Live events are created from $C_c^{L,\beta}$ and $C_s^{L,\beta}$, while replay event are created from $C_c^{R,\beta}$ and $C_s^{R,\beta}$. The packets

within these connections haven't been modified by the DUT or lack blocked packets by the DUT. Therefore, we can compare all the events produced by live traffic and by replayed traffic.

Algorithm 1 shows the process to create packet events. The variable *Connection* is defined as an element from one of $C_c^{L,\beta}$, $C_c^{R,\beta}$, $C_s^{L,\beta}$ *and* $C_s^{R,\beta}$.

Algorithm 1: Packet event process

```
1 Decode_connection (Connection)
2         event_value ← 0
3         Packet []
4         If Traffic direction = source
5                 Then
6                         Directory ← source directory
7                         Dest_File ← Connection
8                         Packet ← Read tcpdump network packet from Directory [Connection]
9                         Open Dest_File for writing
10        Else-If Traffic direction = destination
11                Then
12                        Directory ←destination directory
13                        Dest_File ← Connection
14                        Packet ← Read tcpdump network packet from Directory [Connection]
15                        Open Dest_File for writing
16                Else
17                        Then exit (error)
18        EndIf
19        For each Packet X to end of Packets
20                Do Decode Packet X
21                        If Packet.protocol = TCP
22                                Then print to Dest_File "IP_ident, tcp_seq, payload, event_value"
23                                Else
24                                        Then print to Dest_File "IP_ident, payload, event_value"
25                        EndIf
26        EndFor
27        Close Dest_File
```

The description of Algorithm 1 is as follows. Function *Decode_connection* accepts an element from one of $C_c^{L,\beta}$, $C_c^{R,\beta}$, $C_s^{L,\beta}$ and $C_s^{R,\beta}$ (line 1). The variable *event_value* and the structure *Packet []* are initialized (line 2 to 3). The variable *traffic_direction* stores the traffic direction of the element *Connection* and decides what block is executed depending on the stored value in *traffic_direction* (Line 4, 10). The values of *Connection* that are classified as server side of the traffic, are categorized with traffic direction as "destination" and the values of *Connection* that are classified the client side of the traffic, are categorized with traffic direction as "source". The PCAP file directory of the connection, *Directory*, and the destination directory for the output, *Dest_File*, are initialized (line 6 to 7, line 12 to 13). A Perl function of Tcpdump decodes each packet of *Connection* to store it in the structure

Packet [] (Line 8, 14). A file *Dest_File* is allowed to use it for writing (line 9, 15). A loop read all the values in the structure of *Packet []* to extract the fields of IP identification number, TCP sequence number, and the payload (line 19-27). Each packet in the structure *Packet []* is filtered (line 21). Additionally the extracted values are stored in an output file *Dest_File* (line 24).

Live traffic with logs L^L produces E^L, and replayed traffic with logs L^R yields E^R. The pair of connections ($C_c^{L,\beta}$, $C_s^{L,\beta}$) are used to create a series of live events $E^{L,b}$ and $E^{L,m}$. The pair of connections ($C_c^{\mathrm{R},\beta}$, $C_s^{\mathrm{R},\beta}$) are used to create a series of replay events $E^{R,b}$ and $E^{R,m}$. Packets within the above pairs of connections are treated as packet events. Logs were compared to define the connection events. Each entry on L^L and L^R was assigned to a connection event, associated with its corresponding connection. Connections not registered in L^L or L^R were taken as events with non-specific attribute, i.e. non-logged attribute.

3) Comparing 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 packets and modified packets. The comparison was done using a modified open source program [12] that can compare two PCAP files. The results of the comparison are the blocked packet and modified packets. This information is used to assign modifying event and blocking event. The packets, which are no in this result, are treated as passing events.
4) Comparing events $E^{L,b}$, $E^{L,m}$, $E^{L,l}$, $E^{R,b}$, $E^{R,m}$ and $E^{R,l}$ to calculate the event reproduction ratios and the effectiveness of replayed traffic. The event orders in the sets of E^L must be the same order as that in the set of E^R. The order of the event sets ensures the correctness of event comparison between live traffic and corresponding replayed traffic. For packet event each event will be ordered based on two fields: the TCP sequence number and the IP identification number. Thus, the two set of events are going to have the same order. Connection events do not need to order for the calculation of the reproduction of events because they are compared on the basis of information of the 5-tuple.

Algorithm 2 shows the procedure of event comparison. The variable *EventA* is defined as one of $E^{L,b}$, $E^{L,m}$ and $E^{L,l}$. The variable *EventB* is defined as one of $E^{R,b}$, $E^{R,m}$ and $E^{R,l}$.

Algorithm 2: Event comparisons

```
1   Event_comparison(EventA[], EventB[])
2   TN, TP, FP, FN, i ← 0
3   While TRUE
4           Do i ← i + 1
5           Last unless end of EventA or EventB
6           If EventA[i] = 0 and EventB[i] = 0
7                   Then TN++
8           Else-If EventA[i] = 1 and EventB[i] = 1
9                   Then TP++
10          Else-If EventA[i] = 0 and EventB[i] = 1
11                  Then FP++
12          Else-If EventA[i] = 1 and EventB[i] = 0
13                  Then FN++
14  EndWhile
15  Print "TN, TP, FP, FN, i"
```

The description of Algorithm 2 is as follows. The function *Event_comparison* analyzes two set of events for input (line 1). All variables are set to zero (line 2). The loop read all the elements from both set of events to compare the values in each of them (line 3 to 13). The statement *last_unless* checks if there are elements in the event sets, otherwise it breaks the loop (line 5). The print statement gives the results (line 15).

If the replayed traffic is different from the live traffic, we use identifiers while comparing events. These identifiers link the live traffic and the replay traffic. They showed the changes in the fields of live traffic fields, such as IP address and port numbers.

Chapter 5: Experiment and Results

Two replay tools were used in our experiments, SocketReplay and Tcpreplay. Table 8 lists the elements of this experiment. It includes the configuration of the interfaces, running services on the device, and the general hardware descriptions.

Table 8: The test environment for test bed

	Replay Machine	**Capture traffic machine**	**The ZyWALL USG1000**
CPU	Inter(R) Core (TM)2Quad CPU Q8200 @ 2.33GHz	Inter(R) Pentium (R) CPU 2.80GHz	
RAM	4GB	3GB	1GB
OS	Linux 64bit	Linux 32bit	
Interface	Replaying clients Replaying servers	Capturing client traffic Capturing server traffic Capturing logs from the DUT.	Sending logs Processing LAN traffic Processing Internet traffic
Software	Tcpreplay tool, SocketReplay tool, ProxyReplay and bind9.	Tcpdump [15] and Gulp [14]	Kaspersky Labs Antivirus, ADP (Anomaly Detection and Prevention), IDP (Intrusion Detection and Prevention) and firewall service

5.1. Experiment Settings

To start the experiment we used the test bed in Figure 2a to capture T^L. It is important to capture T_c^L and T_s^L at the same time, because T_c^L and T_s^L should have the same packets unless the DUT block it. Therefore, we synchronized the capturing process on eth0 and eth1. The same procedure was also done for the replay traffic using the test bed on Figure 2b. The size of packets to be captured on the interfaces on traffic recorder (TR), is less than 24,000 bytes to avoid packet loss. Before we start to replay traffic, the traffic trace was padded [9] with zeros, because we want to fill the missing bytes on the captured trace. Otherwise, DUTs would block packets containing the incorrect size in the payload field. The DUT have three types of actions when it encounters malicious traffic: (1) DUT rejects connections with malicious traffic, (2) DUT blocks packets with malicious traffic, and (3) DUT forwards packets with malicious contents.

Types of live traffic used for the experiment. We use regular traffic and special traffic such as, traffic created by firewall testing and by virus files testing to generate events from some security websites [13]. For packet events we only compare the TCP traffic because UDP traffic is not properly implemented in SocketReplay. Replayed

UDP traffic is unidirectional, which makes it very difficult to compare the events. For connection events we use logs originated from TCP traffic or UDP traffic.

In our experiment we use SocketReplay and Tcpreplay. SocketReplay can automatically split traffic into traces at the client interface and at the server interface. However, we adopted two modes of Tcpreplay, bridge mode and CIDR mode, for the experiment to measure the reproduction ratio and replayers' effectiveness. The Bridge mode applies several rules to split the traffic into the traces at the client interface and at the server interface. Client traffic is defined by "Sending a TCP SYN packet to another host, making a DNS request, and receiving an ICMP port unreachable." Server traffic is defined by "Sending a TCP SYN/ACK packet to another host, sending a DNS Reply, and sending an ICMP port unreachable." Mode CIDR asks the user to manually select the server and the client network addresses for the Tcpreplay to split the traffic into traces at the client interface and at the server interface.

5.2. Data Analysis

In this work two types of traffic were applied. Special traffic was generated from some security websites. Regular traffic was captured from normal network traffic from one of the laboratories in NCTU. Table 9 gives a statistic of the PCAP files which are captured for this experiment. The statistics are about the size, the number of packets in a trace, the number of TCP connections, and the number of UDP pseudo connections.

Table 9: The statistics of PCAP files

	Live traffic				**SocketReplay traffic**			
Type of network trace	$T_c^{L,\beta}$	$T_c^{L,\alpha}$	$T_s^{L,\beta}$	$T_s^{L,\alpha}$	$T_c^{R,\beta}$	$T_c^{R,\alpha}$	$T_s^{R,\beta}$	$T_s^{R,\alpha}$
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
# TCP connection	7287	7291	8861	7286	462	462	462	462
# UDP pseudo connections	3628	2183	2174	3608	11410	499	499	11410
	Tcpreplay_bridge traffic				**Tcpreplay_CIDR traffic**			
Type of network trace	$T_c^{R,\beta}$	$T_c^{R,\alpha}$	$T_s^{R,\beta}$	$T_s^{R,\alpha}$	$T_c^{R,\beta}$	$T_c^{R,\alpha}$	$T_s^{R,\beta}$	$T_s^{R,\alpha}$
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
# TCP connection	677	613	685	599	7287	7272	8858	5776
# UDP pseudo connections	2481	2178	2179	2479	3628	2172	2175	3608

Table 10 lists the statistics of these two types of traffic, special traffic and regular traffic. The statistics are based on the number of connections, how a connection is terminated, number of TCP connection and number of UDP pseudo connections.

Table 10: Statistics of traffic trace used by replay tools

Fields	Special 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

To analyze the traffic, we measured the ratio of traffic being replayed by the following formula

$$Replayed\ traffic\ ratio = \frac{\#replayed\ traffic\ packet}{\#\ captured\ traffic\ packet} \times 100\% \quad (5)$$

The ratio is based on the packets of TCP connections. Figure 6 shows the replayed ratio for both special traffic and regular traffic.

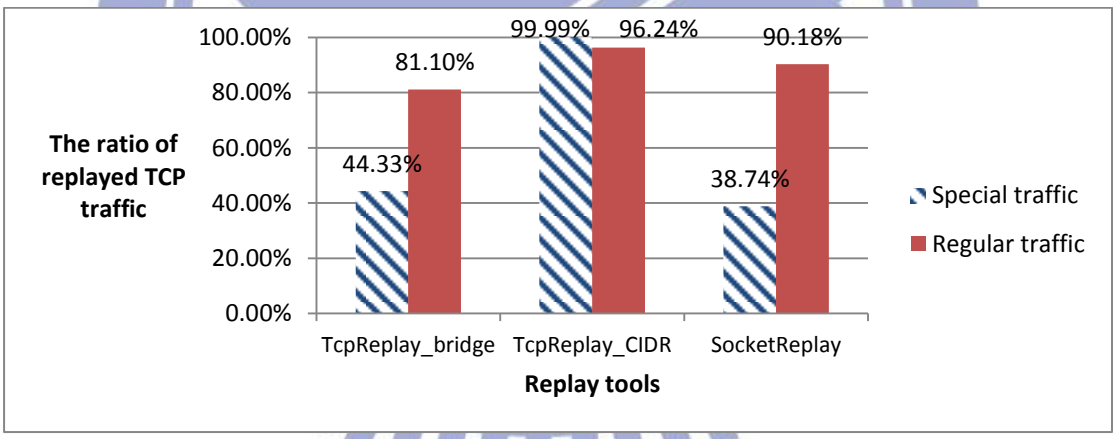


Figure 6: The ratio of TCP traffic being replayed

In Table 11 shows partial results on event reproduction number for SocketReplay and Tcpreplay. Variable $\#e_{LP}$ is the number of events with one type of attribute only in live traffic. Variable $\#e_{LN}$ is the number of events without specific attribute only in live traffic. Variable $\#e_{RP}$ is the number of events with one type of attribute only in replay traffic. Variable $\#e_{RN}$ is the number of events without specific attribute only in replay traffic. We use

$$Live\ event\ ratio = \frac{\#e_{TP}+\#e_{FN}+\#e_{LP}}{\#e_{TP}+\#e_{FP}+\#e_{TN}+\#e_{FN}+\#e_{LP}+\#e_{LN}} \times 100\% \quad (6)$$

$$Replay\ event\ ratio = \frac{\#e_{TP}+\#e_{FP}+\#e_{RP}}{\#e_{TP}+\#e_{FP}+\#e_{TN}+\#e_{FN}+\#e_{LP}+\#e_{LN}} \times 100\% \quad (7)$$

to calculate the live packet event ratio and the replay packet event ratio, respectively.

Table 11: The number of reproduced events

	Blocking		Modifying	
Replay tools	**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

In Table 12 shows the number of connections on the live traffic and the replayed traffic. Connections on the replayed traffic were generated by Tcpreplay using bridge mode. To calculate the live and replay logging ratio, we use the following formula

$$Logging\ event\ ratio = \frac{\#\ logs}{\#\ connections} \times 100\% \quad (8)$$

Table 12: The number of logs and connections

	Client->Server	Server->Client
# Live connections	7281	7291
$\#L^L$	206	
# Replayed connections	599	612
$\#L^R$	57	

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

In this experiment, the ratio of live events and replay events were calculated from regular traffic and special traffic, respectively. Figure 7a and 7b show the ratio of blocking, modifying, passing, logging, and non-logging events occurred on live traffic, two pieces of replayed traffic yielded by SocketReplay and Tcpreplay.

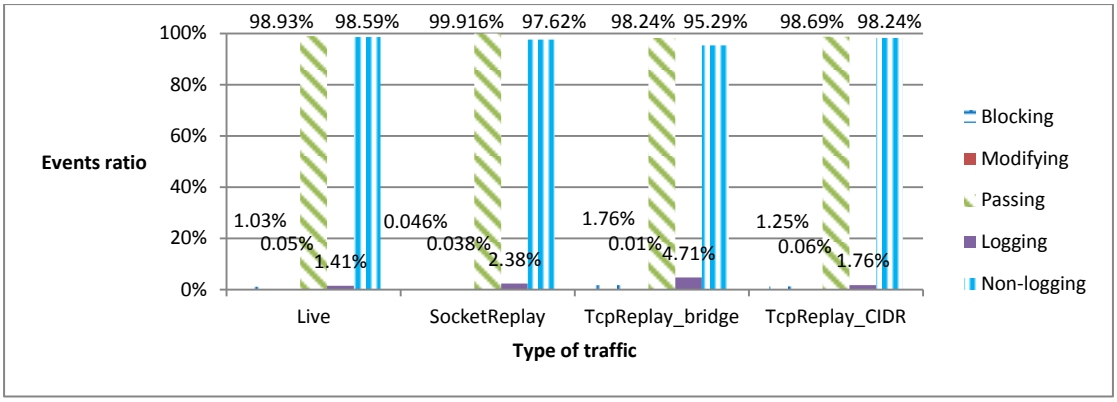


Figure 7a: Live events ratio vs replay events ratio for special traffic

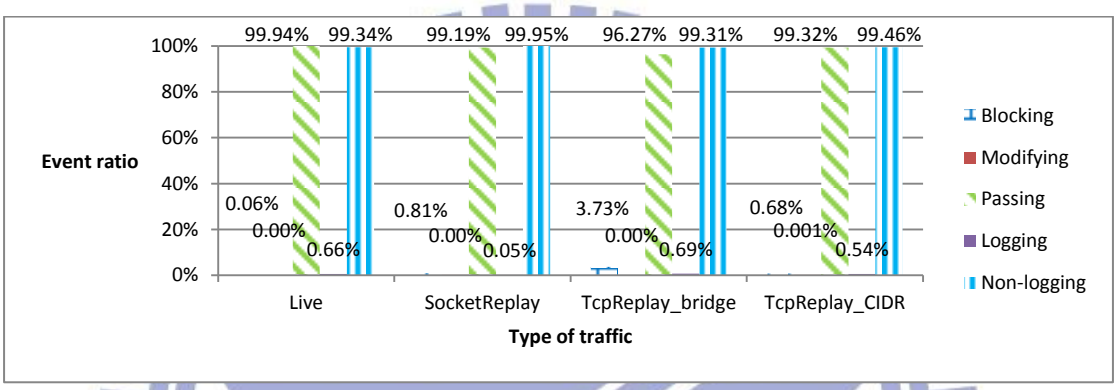


Figure 7b: Live events ratio vs replay events ratio for regular traffic

In this experiment, we compare the difference ratios for events; however, the replayed TCP traffic ratios are different, as shown in Figure 6. Therefore, a direct comparison between the live traffic with replayed traffic is not possible.

In both experiments the ratio of passing events is much higher than that of events with other attribute. The ratio of blocking events on Tcpreplay traffic is higher than that on live traffic. The ratio of modifying events on the replayed traffic yielded by replay tools is lower than that on live traffic. One exception is the ratio of events generated by Tcpreplay using CIDR mode using regular traffic. The ratio of logging events on the Tcpreplay traffic is higher than that on live traffic. However, the ratio of logging events on SocketReplay traffic is lower than that on live traffic.

Below we describe the reasons for the outcomes of the event ratios. On regular traffic, high ratio of blocking events on TcpReplay traffic is because the replayed traffic doesn't follow the states of connections by TCP protocol (it is unable to synchronize Syn/Ack's to create valid TCP sessions), while the ratio of blocking events on SocketReplay traffic is low because the replayer must keep the states of connections by TCP protocol. Therefore, the DUT blocked connections that didn't follow the states of connections by TCP protocol. During the SocketReplay

preprocessing stage many packets were eliminated. This led to the not existence of blocking events.

On special traffic, packets with malicious contents on the packet payload or with incorrect contents in packet header could trigger the modifying events. Tcpreplay with CIDR mode replayed the traffic and triggered all the modifying events, while Tcpreplay with bridge mode replayed the traffic but only triggered some modifying events of packets with malicious contents, without triggering the modifying events of packets with incorrect contents in the packet header. SocketReplay replayed the traffic and triggered all the modifying events of packets with malicious contents, without triggering the modifying events of packet with incorrect contents in the packet header. Modifying events did not occur in the regular traffic on live platform, but they might be triggered on the replayed traffic (0.0001% Tcpreplay with CIDR mode) because of badly formed packets by the DUTs.

Logging events occurred on the special traffic and the regular traffic because of the anomaly-based rules that depends on heuristic [11], and the signature-based rules that depends on the packet contents. Signature-based logging events were triggered on the traffic generated by Tcpreplay with both modes or by SocketReplay. However, all the replay tools only produced little anomaly-based logging events.

5.4. Replayed traffic effectiveness

In this experiment, we calculated the event reproduction ratio for SocketReplay and Tcpreplay. The events can be classified by specific and non-specific attributes. For both types of events the ratio is calculated from the traffic with consistency. The ratio of traffic with consistency, or consistency ratio is shown in Figure 8.

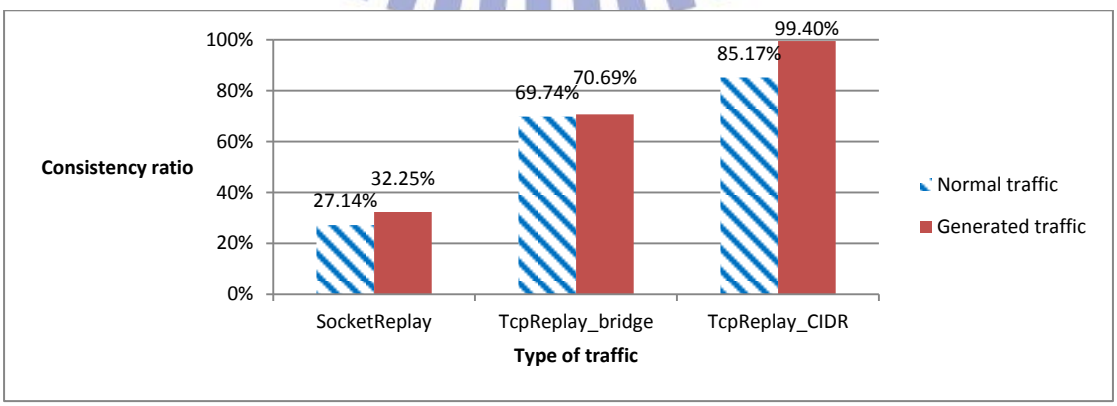


Figure 8: The consistency ratio of replayed traffic

Under the traffic consistency, Figure 9a and 9b depict the combined event reproduction ratios from special and regular traffic for various replay scenarios--Tcpreplay with bridge mode, SocketReplay, and Tcpreplay with CIDR mode. The modifying event reproduction ratio is 100% for both types of traffic. The reproduction ratios of blocking and non-modifying events are close to one another for the various replay scenarios from special traffic, while the ratios on the regular traffic are not so close. The reproduction ratios of logging and non-logging events under Tcpreplay using CIDR mode are higher than other replay scenarios for both types of traffic.

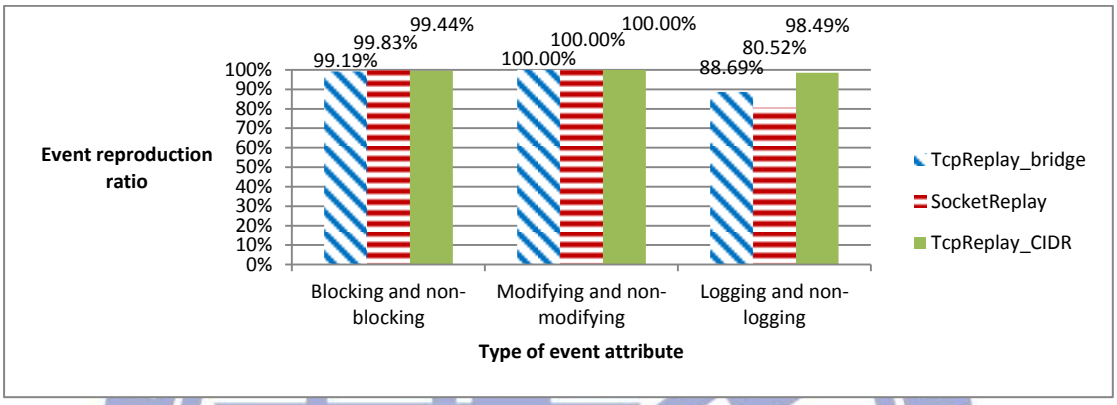


Figure 9a: The event reproduction ratio of special traffic

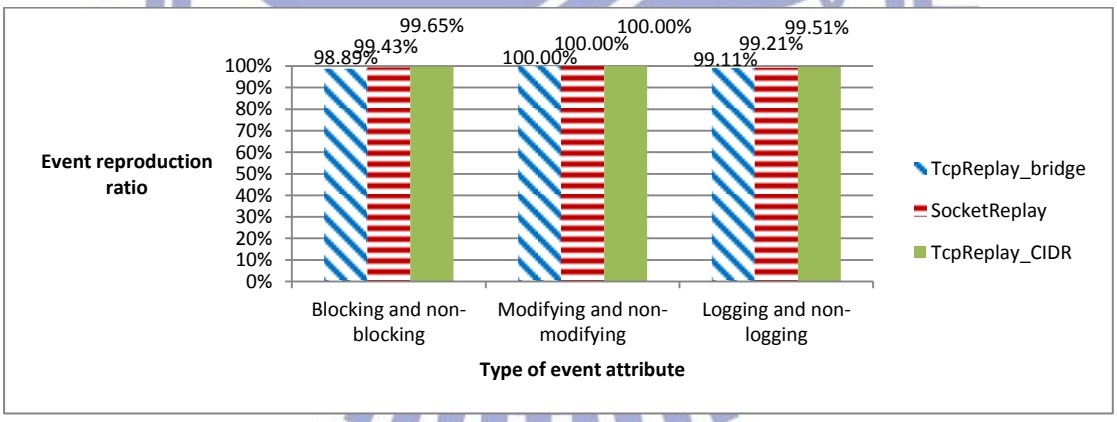


Figure 9b: The event reproduction ratio of regular traffic

Here the replayer's effectiveness of blocking and of non-blocking events for SocketReplay and Tcpreplay is analyzed. To calculate the replay effectiveness for each event attribute, we use equation (2) and equation (3).

In Figure 10a and 10b, we analyzed the effectiveness of replayers for blocking and non-blocking events, which were derived from TN, TP, FN, and FP. SocketReplay replaying special and regular traffic didn't trigger TP's. The TP rates yielded by Tcpreplay with bridge mode and with CIDR mode were 38.14% and 75.64% of

special traffic and 42.86% and 50.70% of regular traffic. The TP rate yielded by Tcpreplay with CIDR mode is higher than that with bridge mode.

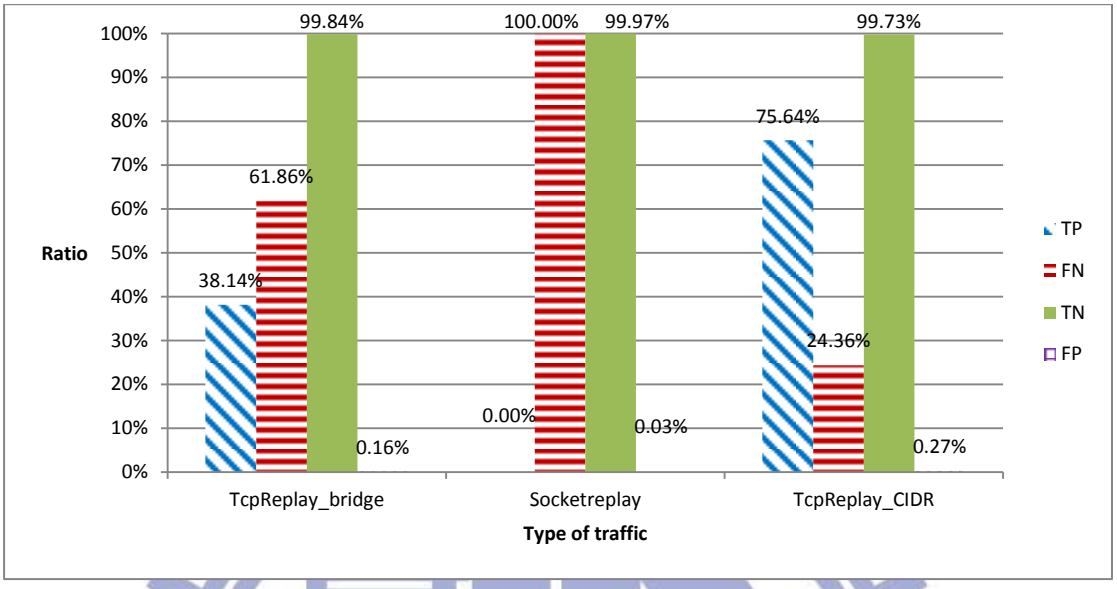


Figure 10a: The effectiveness of blocking and non-blocking events for special traffic

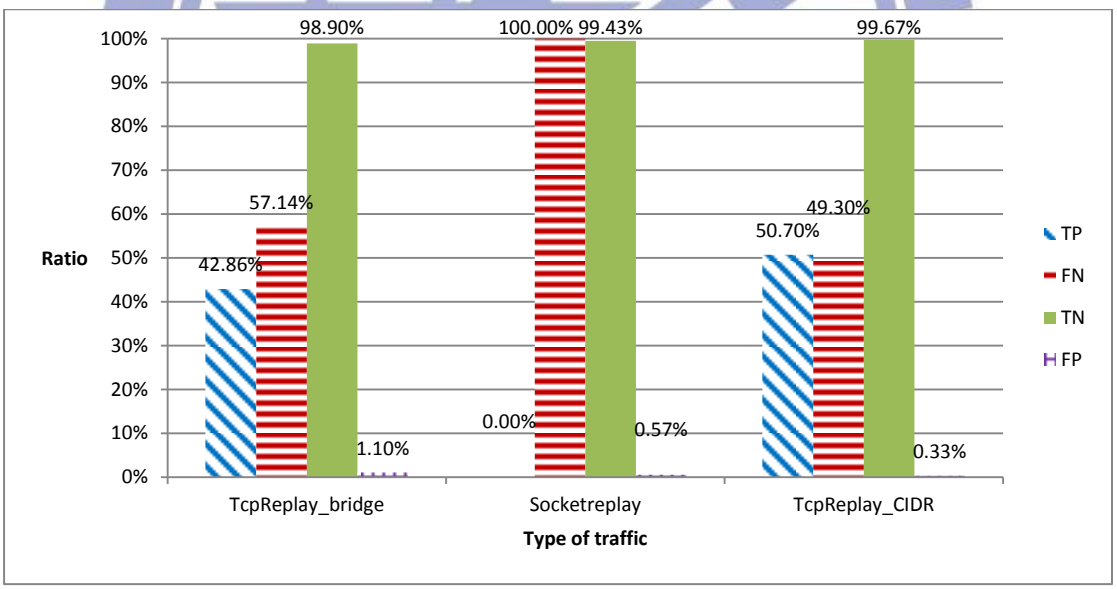


Figure 10b: The effectiveness of blocking and non-blocking events for regular traffic

During the preprocessing stage, SocketReplay removed packets that have higher or equal TCP sequence number than the FIN packet within a connection. Blocked packets on live traffic dropped by DUT or destination host were eliminated during the preprocessing stage. Therefore, the TP rate of the replayed traffic by Socketreplay is 0%. Because the replayed TCP traffic ratio of Tcpreplay with CIDR mode is higher

than that with bridge mode, the TP rate yielded by Tcpreply with CIDR mode became higher than that with bridge mode. Packets blocked on the live testbed by heuristic rules in or specific traffic behaviors [10] of a DUT can't be reproduced on the replayed traffic. Thus, the rates of FN and FP are high for SocketReplay and Tcpreplay with both modes.

In Figure 11a and 11b, we analyzed the effectiveness of modifying and non-modifying events. The results are derived from the occurrences of TN, TP, FN, and FP. Replayer's replaying special traffic triggered 100% of TP rate and 100% of TN rate. Replayer's replaying regular traffic triggered 100% of TN rate and 0% of TP rate.

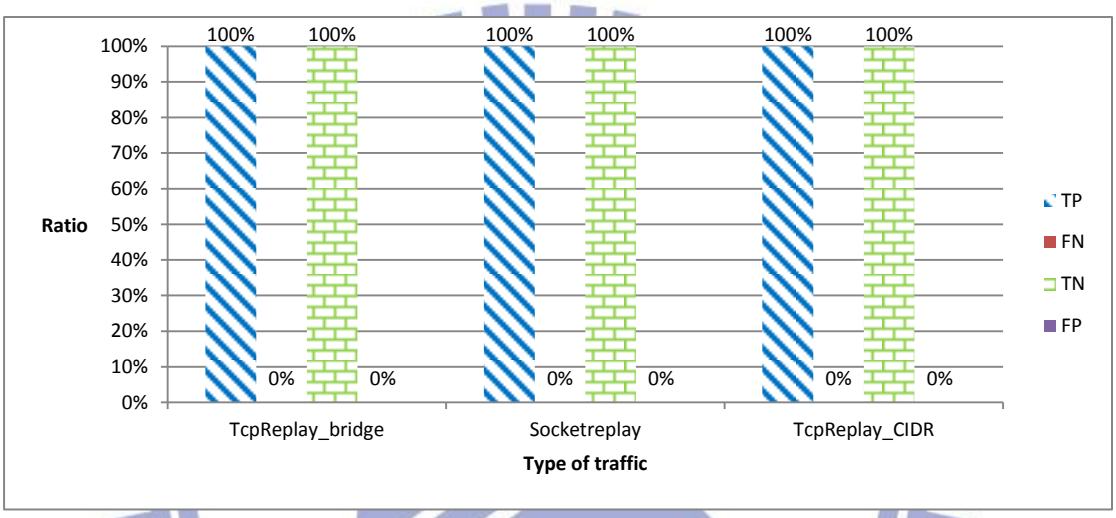


Figure 11a: The effectiveness of modifying and non-modifying events for special traffic

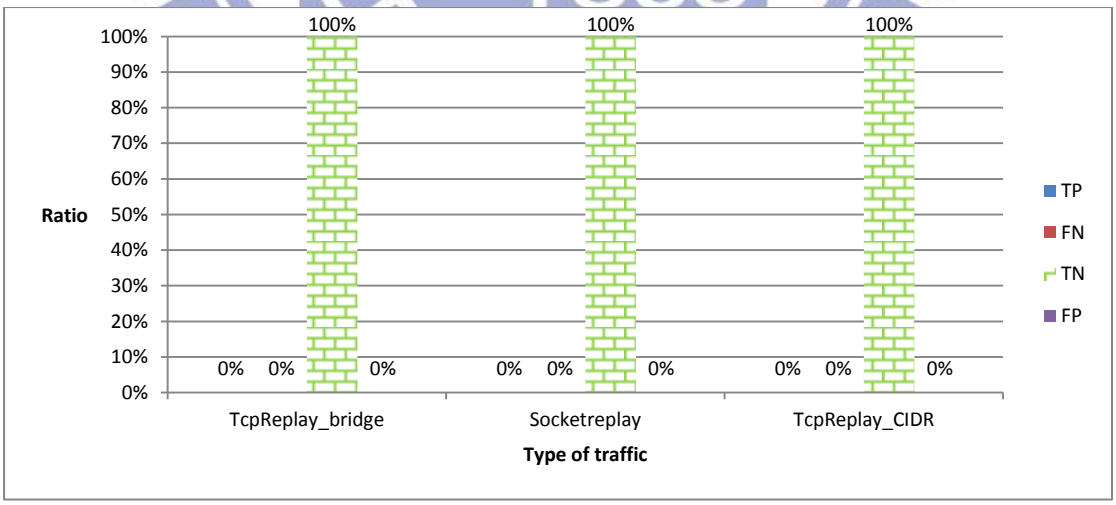


Figure 11b: The effectiveness of modifying and non-modifying events for regular traffic

Tcpreplay with both modes and SocketReplay replaying the special traffic triggered 100% of TP rate for modifying event. Here only the packets where their

payload was modified were considered the modifying events. The effectiveness of header modifying events was not calculated. Because the regular traffic did not produce any modifying events, all the replay tools did not yield the TP rate.

In Figure 12a and 12b, we analyzed the effectiveness of replayers for logging and non-logging events. The results are derived from TN, TP, FN, and FP. The TP rate triggered by SocketReplay replaying the special and the regular traffic is lower than that by the other replayer. The TN rates triggered by replayers were high either by replaying the special traffic or the regular traffic.

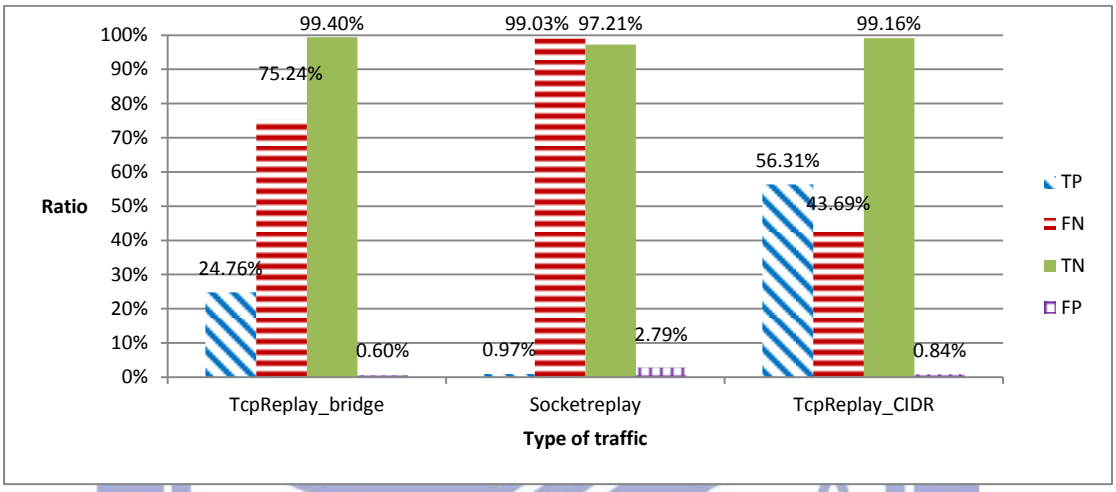


Figure 12a: The effectiveness of logging and non-logging event for special traffic

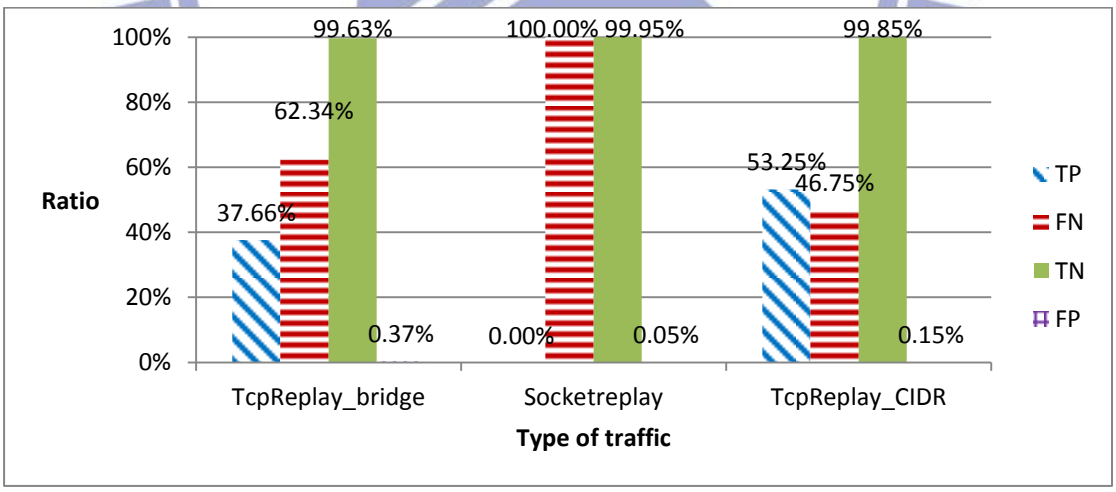


Figure 12b: The effectiveness of logging and non-logging event for regular traffic

Tcpreplay with both modes and SocketReplay replaying special and regular traffic could trigger signature-based logging events. Tcpreplay with CIDR mode achieved 56.31% and 53.25% of TP rate for special and regular traffic, respectively, while Tcpreplay with bridge mode, 24.76% and 37.66% of TP rate, respectively. During

preprocessing stage SocketReplay replaying the regular traffic could not replay some traffic that would lead to trigger signature-based logs; therefore, Socketreplay replaying the regular traffic would yield 0.97% and 0.00% of TP rate for special and regular traffic, respectively.

Tcpreplay with both modes replaying the regular and the special traffic could not well produce anomaly-based logging event; therefore, Tcpreplay with bridge mode can trigger 75.24% and 62.34% of FN rate for special and regular traffic, respectively, and Tcpreplay with CIDR mode can trigger 43.69% and 46.75% of FN rate for special and regular traffic, respectively. SocketReplay replaying special and regular traffic could not reproduce the anomaly-based logging events; therefore, replayed traffic for Socketreplay could trigger 99.03% and 100% of FN rates for special and regular traffic, respectively. Replayed traffic for all replayers could trigger new anomaly logging events; therefore, replayers replaying special and regular traffic can yield 0.60%, 2.79%, and 0.84% of FP rates for Tcpreplay with bridge mode, SocketReplay, and Tcpreplay with CIDR mode, respectively, and 0.37%, 0.05%, and 0.15% of FP rates for Tcpreplay with bridge mode, Socketreplay, and Tcpreplay with CIDR mode, respectively.

5.5. Investigating the Lack of Consistency on Replayed Traffic

We illustrated the consistency of replayed traffic in Figure 8. The engine of SocketReplay is implemented by socket programming to establish connections [1]. Thus, when a connection is replayed, the replay tool also performs the function of TCP window size adjustment and flow control. The open socket would try to adjust the TCP window size and flow control through ACK packets, because the sender may transmit packets faster than the receiver can receive. This led the replayed traffic by SocketReplay to be inconsistent with the traffic captured from the live testbed.

The lack of consistency of the traffic replayed by Tcpreplay is because the Tcpreplay will remove the ACK packets, such as TCP *keep alive message*, from the client side of the replayed traffic, and randomly remove the duplicate packets and FIN packets. Therefore, the replayed connections that remove packets will result in consistency.

Chapter 6: Conclusions

This work proposed a method to measure the effectiveness of replay tools. The measurement of effectiveness is based on the hypothesis that replay testing can reproduce the same number of events as the live traffic under the corresponding test platform and testing environment. This hypothesis was verified by comparing the events triggered by live traffic with those by replay traffic.

In this study, the event reproduction ratio was affected by replay policies, traffic contents and DUT filtering rules. The passage of replayed traffic is based on the replay policies, such as the completion of connections or timestamps. For instance, Tcpreplay using bridge mode and SocketReplay using a preprocessing stage for special traffic would respectively replay 44.33% and 38.74% of TCP traffic, resulting in 38.14% and 0.00% of effectiveness of blocking event ratio , 100% and 100% of effectiveness of modifying event ratio , and 26.76% and 0.97% of effectiveness of logging event ratio, respectively, Tcpreplay using CIDR mode for special traffic could replay 99.99% of TCP traffic, resulting in 75.64% of effectiveness of blocking event ratio , 100% of effectiveness of modifying event ratio , and 56.31% of effectiveness of logging event ratio . Therefore, the policy of CIDR mode has higher event reproduction ratio than other policies.

Traffic contents also affect the event reproduction ratio of replayers. In this study the special traffic, which has many incomplete connections and high percentage of terminating connections using RST, may lead to low effectiveness of specific attribute event ratio, while the regular traffic, which has few incomplete connections and high percentage of terminate connections using FIN packets, may result in high specific event ratio. For example, Tcpreplay using bridge mode, CIDR mode, and SocketReplay replaying regular traffic can respectively replay 81.10%, 96.24%, and 90.18% of TCP traffic, while replayers replaying special traffic can respectively replay 44.33%, 99.99%, and 34.78% of TCP traffic. Therefore, traffic content, which has less incomplete connection and less RST packets to terminate a connection, has higher replayed TCP traffic ratio and higher event reproduction ratio than the traffic content with those characteristics.

DUT filtering rules using heuristics may create events that are difficult to reproduce. Tcpreplay CIDR mode and SocketReplay for regular traffic can lead to 46.75% and 100 % of FNs.

To enhance the event reproduction ratio we suggest several ideas. Replay tools should have two phases. First phase, it is about parsing the traffic and keeping the state of each connection. Second phase, it is about replaying traffic. Replay tools should consider the captured traffic properties, such as round trip time for each connection, and the number of connections at a specific time. Keeping the same captured traffic properties in replay scenario will increase the probabilities of triggering events that depend on heuristic rules. SocketReplay should be aware of where the traffic originates, otherwise replayed traffic won't reproduce the same events as live traffic. Currently, SocketReplay is replaying unidirectional UDP traffic. Preprocessing policies have a big impact on traffic replay ratio. Thus, selecting the best rules to parse the traffic is very important.

In this study preprocessing policies and features of replay tools have been evaluated for justifying the impacts on the event reproduction ratio while testing various types of DUTs. The results of these experiments may change depending on the DUT under test, or the services and configuration of the DUT. In addition, the comparison of the events is performed based on a live connection that has the same number of TCP segments in the connection being replayed.

Reference

[1] 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*, Volume 50, Issue 4, pp. 114-121, April 2012.

[2] Seung-Sun Hong, Fiona Wong, S. Felix Wu, Bjorn Lilja, Tony Y. Yohansson, Henric Johnson, and Ame Nelsson, "TCPtransform: Property-Oriented TCP Traffic Transformation".

[3] 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 2012)*, Paris, France, April 2012.

[4] Yu-Chung Cheng, Urs Hölzle, Neal Cardwell, Stefan Savage, and Geoffrey M. Voelker, "Monkey See, Monkey Do: A tool for TCP Tracing and Replaying," USENIX Annual Technical Conference, General Track, 2004.

[5] Aaron Turner, "Flowreplay Design Notes," [Online]. Available: http://tcpreplay.synfin.net/.

[6] Turner, "TCP Replay: pcap editing & replay tools for UNIX," April 2010. [Online]. Available: http://tcpreplay.synfin.net/ .

[7] NATreplay test tool, 2009. [Online]. Available: http://www.nbl.org.tw/.

[8] Chun-Yin Huang, Ying-Dar Lin, Peng-Yu Liao, Bing-Heng Peng, Yuan-Cheng Lai, "Stateful Traffic Replay for Application Proxies," Institute of Network Engineering College of Computer Science National Chiao Tung University, February 2011.

[9] Tcprewrite. [Online]. Available: http://tcpreplay.synfin.net/.

[10] B.Harry, "TCP/IP security threats and attack methods". [Online]. Available: http://www.sciencedirect.com/ .

[11] Logs rules information. [Online]. Available: http://snort.org/.

[12] Pcapdiff tool. [Online]. Available: http://www.openbsd.org/.

[13] Security websites. [Online]. Available: www.pcflank.com, www.auditmypc.com, www.securityspace.com, www.eicar.org, and www.grc.com.

[14] Gulp tool, [Online]. Available : http://staff.washington.edu/corey/gulp

[15] Tcpdump, [Online]. Available : http://www.tcpdump.org/

[16] Tcpprep, [Online]. Available: http://tcpreplay.synfin.net/.

[17] Tomahawk, [Online]. Available: http://tomahawk.sourceforge.net/

[18] Documentation of netfilter policies, [Online]. Available: http://people.netfilter.org/pablo/docs/login.pdf