PACKTER: implementation of Internet traffic visualizer and extension for network forensics

Daisuke Miyamoto
Project PACKTER
namaya2hashi@packter.net

Takuji Iimura
Project PACKTER
uirou@packer.net

ABSTRACT

This paper introduces PACKTER, a free and open source software for visualization of Internet traffic. This paper also extends it for support network forensics. Many traffic visualization make network operators realize the current network status, including anomalous activities. Our motivation is the utilization of the visualization tools for starting network forensics process, e.g., investigating where the issued packets came from. Since there were few softwares for our intent, this paper develops PACKTER, which is able to visualize traffic based on per-packet and/or per-flow information in real-time. This paper also extends PACKTER to have a function for negotiation to a network forensic system.

1. INTRODUCTION

Creating new network operation style is beyond the visualization of today's network. Whereas some visualization tools provide novel graphics representing network activities, such tools are not designed to provide any user-interfaces for network operation. Imagine if you are playing an online game, you will react when the game screen shows some important events. You will also try to control the game by input devices with keeping your sights to the screen. In the context of today's network operation, after you realized such events from visualization screen, you might launch other applications, login to some servers, and prepare next operations. We assume that there is lack of support for starting operations within the network visualization tools.

Our motivation is to integrate the functions of starting network operation processes with real-time traffic visualization tools. Due to that the tool often makes its operators realize anomalous activities, we consider to employ the tools as the user-interface for network forensics.

Unfortunately, we could not find suitable tools for our intent. Even though many researches for visualization had been proposed, few tools were available as a free and open source software. Moreover, the most of them were designed to show the results of their offline analysis. Aside from offline analysis, very few real-time visualization tools were found. Whereas these tools were practical and innovative, our intent required to show information with a per-packet-granularity of the traffic.

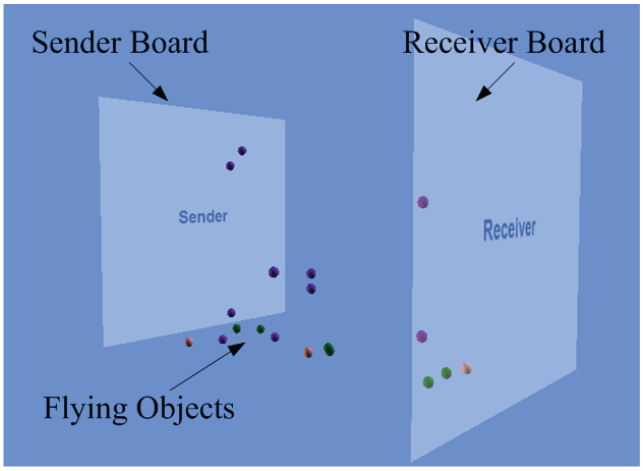


Figure 1: Overview of PACKTER

This paper designs and implements a traffic visualization tool, named PACKTER [7], at first. PACKTER consists of two programs, PACKTER agent and PACKTER viewer; the agent passively probes per-packet and/or per-flow information, and the viewer visualizes the collected information in three-dimensional screen.

This paper then extends our developed programs to support network forensics processes. There are various types of forensics, but this paper focuses on identifying whether a packet comes from. Because of that the purpose is similar to IP traceback, which aims at locating the source node even if the packet employed spoofed source IP address, we refined PACKTER to cooperate to InterTrack [5], one of the IP traceback systems.

The rest of the paper is organized as follows. Section 2 illustrates the development of PACKTER and section 3 explains our extension for IP traceback. Section 4 reveals the limitations in PACKTER, and section 5 finally summarizes our contribution.

2. DEVELOPMENT OF PACKTER

This section introduces the design principles and the

developments of our Internet traffic visualization tool, named PACKTER. We designed it by being inspired to NICTER [4], the famous traffic visualizer in Japan. Its three-dimensional visualization engine shows traffic animation inside a cube. Each packet is represented by a colored rectangle, and the rectangle appears on a plane of the cube when a packet is received at the monitored network. Note that NICTER is not published as a free and open source software, our project is necessary to develop totally different codes and takes on overall distinct system architecture.

2.1 Overview

Figure 1 shows a screen shot of PACKTER. It appears two squares, named "sender board" and "receiver board", in respectively. The former presents the traffic source, the latter denotes the destination.

In each square, x axis denotes an IP address where the left corner is 0.0.0.0 and the right (1) is 255.255.255.255. Given the IP address, the address will be regularized in the range of 0 to 1 by following steps. At first, the address is converted to decimal. It then divided by 2^{32}, and finally located in the range of 0 to 1. When using IPv6 addresses, the left corner is :: and the right is ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff, and the decimal decoded IPv6 address is divided by 2^{128} for the regularization.

y axis denotes a port number if the packet is a TCP segment or a UDP datagram. The value is also divided by 2^{16} to be regularized in the range of 0 to 1. If the packet is an ICMP message, ICMP type value devided $/2^8$ is for the sender y cordinate, and ICMP code value devided by $/2^8$ is for the receiver y. Since bot TCP and UDP port numbers are 16bit fields, and both ICMP type and code are 8bit fields, the regularized values are in the range of 0 to 1.

In PACKTER, a ball is called a "flying object", which presents each packet. Its color variation has ten types as shown in Table 1. The ball appears at the sender board at first, then flows toward the receiver board, and finally vanishes when it reached to the receiver board. For example, if the pairs of the traffic source address and its TCP port number is given $(10.0.0.1, 60000)$. The decimal form of IP address is 167,772,161, so x coordinate is 0.04 $(= 167,772,161/2^{32})$ and y coordinate is 0.92 $(= 60000/2^{16})$. So, the ball appears at $(0.039, 0.916)$ in the sender board. Given the destination pair $(127.0.0.1, 80)$, the ball flows toward $(0.496, 0.001)$ in the receiver board. If the packet is a TCP SYN packet, the ball will be colored blue as shown in Table 1.

2.2 Design

PACKTER is composed of two types of programs, agent and viewer. An agent collects a packet and sends the packet's information to a viewer, the viewer then draws the packet as we described in section 2.1.

Table 1: Coloring variations of flying objects

#	Color	Layer 3	Layer 4	Flag
1	Pink	IPv4	TCP	ACK
2	Blue	IPv4	TCP	SYN
3	Red	IPv4	TCP	FIN or RST
4	Purple	IPv4	UDP	
5	Green	IPv4	ICMP	
6	Yellow	IPv6	TCP	ACK
7	White	IPv6	TCP	SYN
8	Skyblue	IPv6	TCP	FIN or RST
9	Lightgreen	IPv6	UDP	
10	Orange	IPv6	ICMP	

Table 2: PACKTER protocol format

Category (i) : Drawing flying object
PACKTER\r\n SRCIP,DSTIP,SRCPORT,DSTPORT,FLAG,DESCRIPTION
Category (ii) : Showing message and picture, and playing sound
PACKTERMESG\r\n picture-file,text-message
PACKTERHTML\r\n html-message
PACKTERSOUND\r\n seconds,sound-file
PACKTERSE\r\n sound-effect-file
PACKTERVOICE\r\n text-message
PACKTERSKYDOMETEXTURE\r\n texture-file

Currently, our agent is available to collect packets by (1) monitoring a network interface, (2) reading a packet trace file, (3) accepting flow sampling protocols, and (4) receiving via Unix domain socket. In the cases of (1) and (2), the agent uses typical packet capture library for collecting. In the case of (3), the agent works as the collector for sFlow [6] and/or NetFlow [1]. Within these sampling technologies called xFlow, the xFlow agents sample packets with a specified sampling rate, and the agents send the packets' information to an xFlow collector. Since PACKTER agent equips the function of the xFlow collector, it accepts the flow information from the xFlow agents. The function (4) is designed to cooperate with external programs. For example, SNORT [9], the typical intrusion detection software (IDS), detects malicious traffic and it then outputs the packets' information via Unix domain socket. Because of monitoring the socket, PACKTER agents can collect suspicious traffic which SNORT detected.

PACKTER agents sends the information to the viewer based on PACKTER's protocol format as shown in Table 2. Our protocol can be categorized into two types. The category (i) is used for drawing packets into the viewer's screen. The column consists of the source IPv4 or IPv6 address, the destination address, the source port number or ICMP type, the destination port number or ICMP code, flag, and the description of the packet; the flag is corresponding to the first column in Table 1. The category (ii) is used for showing message, pictures, and play sound. PACKTER supports to render text or HTML message in its screen. It also sup-

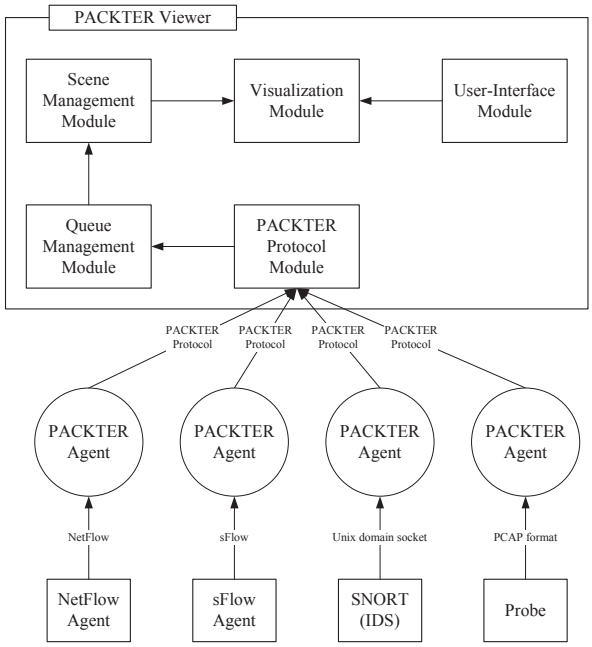


Figure 2: PACKTER architecture

ports to play sound files till specified seconds pass, and has a function to pronounce the specified text messages by cooperating to a speech synthesis software.

2.3 Implementation

The architectures of PACKTER agent and viewer are shown in Figure 2. PACKTER agent employs PCAP [11] library for collecting packets from network interfaces and/or reading a packet trace file. It also supports random sampling based on the probability which users can freely specify. The viewer also supports sFlow version 4.0, NetFlow version 9.0, and SNORT version 1.6 or later. Based on each packet, PACKTER agent sends the information to the viewer over UDP datagram.

PACKTER viewer is composed of five modules. The first module, PACKTER protocol-handling module binds on UDP port 11300, accepts the packet information which the agent sent, and inserts the information at the tail of the queue. The second, Queue Management Module, set time stamp to each information. The third, Scene Management Module retrieves the queue by referring to the time stamp; it is used to play the viewer's screen backwards. The forth, Visualization Module, draws a ball at the corresponding coordinates on the sender board, and makes the ball flow to the receiver board. The rest of module deals with keyboard and mouse events. The viewer supports that the users change the viewpoint in screen. It also supports for the users to replay scenes.

Our implemented programs are available as open source softwares [7]. PACKTER agent is written in C and it runs on POSIX operating systems. PACKTER viewer employs C# and XNA Game Studio 3.1 for its rendering engine, so it runs on windows operating systems.

3. NEGOTIATION TO IP TRACEBACK

This section develops PACKTER to equip further functions that aim at facilitating to launch network forensic. We focused on cooperating to IP traceback, which investigates where the issued packet came from. In order to facilitate the discussion in accurately, section 3.1 provides the summary of the traceback and the typical implementation named InterTrack [3]. Section 3.2 illustrates the trace request process for PACKTER, and section 3.3 shows the trace results.

3.1 InterTrack

Essentially, Denial of Service (DoS) attacks exhaust the resources of a remote hosts or networks that are otherwise accessible to legitimate users. Especially, a flooding attack is the typical example of DoS attacks. In the case of the flooding attack, the attackers often used the source IP address spoofing technique. IP address spoofing can be defined as the intentional misrepresentation of the source IP address in an IP packet in order to conceal the sender of the packet or to impersonate another computing system. Therefore, it is difficult to identify the actual source of the attack packets using traditional countermeasures.

IP traceback aims to locate attack sources, regardless of the spoofed source IP addresses. Especially, Source Path Isolation Engine (SPIE) [8] is a feasible solution for tracing individual attack packet. When a node is suffered from DoS attacks, the node calculates a hash from the attack packet, composes a traceback query including the hash, and sends the query toward the previous hop router. However, SPIE requires that every router captures partial packet information of every packet which passes through the router. Trace-ability would decrease to a minimum if there were only a few routers that support SPIE.

For reducing the deployment cost of IP traceback systems, several researches [2, 3] have proposed the use of the AS-level deployment to facilitate global deployment of IP traceback systems. In this case, it is necessary to deploy the system into each AS instead of implementing the SPIE in each router. Since the traceback system monitors the traffic between the AS border routers and exchanges information for tracing the issued packets, the traceback client can identify the source AS of the issued packets.

InterTrack is designed for deployment at AS level, and its main goal is to reconstruct the reverse AS path, which is the true attack path in AS hop level, and to detect the source ASes of an attack if possible. An-

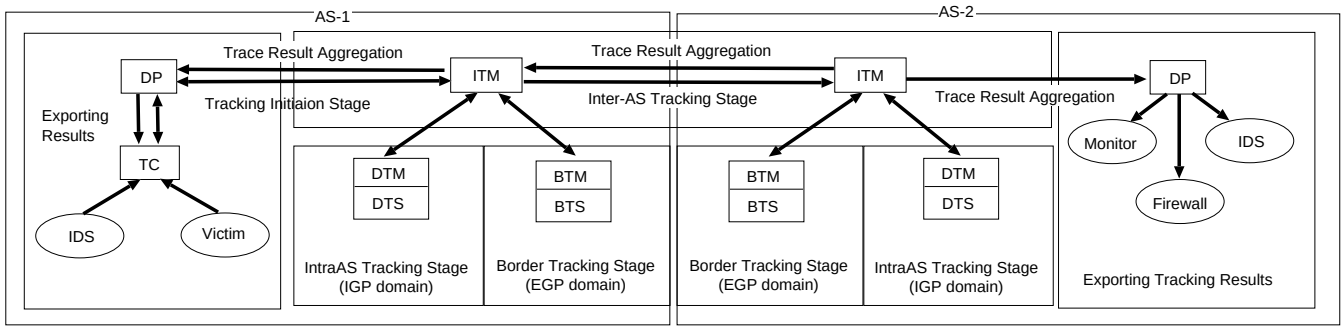


Figure 3: Procedures of an attack tracking on InterTrack

other goal of InterTrack is to achieve the interconnection among IP traceback system(IP-TBS)s, detection systems and prevention systems inside an AS.

In InterTrack architecture, each AS has a set of InterTrack components. A set of InterTrack components includes: the Inter-domain Tracking Manager (ITM), Border Tracking Manager (BTM), Domain Traceback Manager (DTM), Decision Point (DP), and Traceback Client (TC). Figure 3 shows the overview of InterTrack architecture. A phased tracking approach is applied on inter-domain traceback trials through InterTrack. InterTrack separates a traceback trial in four stages along with network boundaries; the tracking initiation stage, the border tracking stage, the intra-AS tracking stage and the inter-AS tracking stage. After accepting a traceback request on the tracking initiation stage, each AS preliminary investigates its own status against the mounted attack on the border tracking stage. On the border tracking stage, an AS judges by InterTrack whether or not the AS is suffered from an attack, whether or not the AS is forwarding malicious attack packets, or whether or not the AS is suspected of having attacker nodes on the inside. Triggered by the investigated AS status, InterTrack runs the inter-AS tracking stage and the intra-AS tracking stage in parallel. Detailed behavior of each component were described in [3].

3.2 Sending Trace Request

Assuming if PACKTER viewer has enough information for IP traceback, the users of the viewer can easily start the tracking initiation stage with few operations; selecting packet with the mouse, and triggering the stage with the keyboard. As we described in section 3.1, the trace request from TC to DP is the trigger of the stage. For doing so, TC calculates hash values from sampled packets, composes a client trace request message with the specified format, sends the trace request to DP, and finally receives the result written in the client trace reply format.

In order to make PACKTER viewer work as TC, this paper modifies PACKTER agent for giving the information to the viewer, and then develops new module which interconnects between PACKTER viewer and DP. The minimum requirement for the information is to contain the hash values for each packet. The hashing process was formalized by Snoren et al. [8] in the case of IPv4 packet, and by Stayer [10] in the case of IPv6 packet. These proposed to mask the particular header fields, that have the possibility of being changed at a router along the path (e.g., IP time-to-live field), to zero prior to digesting. According to the latest implementation of InterTrack, it implemented the masking algorithms and it also employed MD5 algorithm as the digesting function.

Accordingly, we make PACKTER agent calculate the hash value for each packet in the same fashion of InterTrack. Since our protocol format supported to include text strings in the description field, the agent is able to insert information to the filed. Figure 4 shows a case study for including the trace information in PACKTER protocol. The description is composed of the hash value for each packet and the IP address of the interconnecting module between the viewer and DP.

When drawing the packet in the screen of the viewer, the viewer provides an user-interface which enables to select the flying object with a mouse. Because of rendering the objects in three-dimensional screen, the viewer observes the current coordinates of the mouse. It then determines points in screen space on the mouse coordinates by projecting a vector from screen space into object space.

After the user selected a packet and he then pressed "T" key, the viewer sends the hash value of the packet to the interconnecting module, named PACKTER_TC; in the case of Figure 4, PACKTER_TC runs at host 192.168.1.1 on UDP port 11301, and receives the hash value. Figure 5 shows an example for the client trace request.

3.3 Receiving Trace Reply

PACKTER_TC receives two responses from DP. One

```
PACKTER\r\n
10.0.0.1,127.0.0.1,60000,80,1,(hash value)-192.168.1.1\r\n
```

Figure 4: Example for PACKTER agent's message

```
<?xml version="1.0" encoding="UTF-8"?>
<InterTrackMessage type="ClientTraceRequest">
  <ClientTraceRequest>
    <DestinationNode>
      <NodeID idtype="IP">
      <IPAddress version="4" block="loopback" mask="32">127.
0.0.1</IPAddress>
      </NodeID>
    </DestinationNode>
    <SourceNode>
      <NodeID idtype="IP">
      <IPAddress version="4" block="loopback" mask="32">127.
0.0.1</IPAddress>
      </NodeID>
    </SourceNode>
   <TemporarySequenceNumber sec="1343208049" usec="320831
"/>
    <TTL>16</TTL>
   <PacketDump encodetype="md5" header="ip" iftype="1" Pay
loadLength="32">(hash value)</PacketDump>
    <Options>
      <Option type="type">PACKTER</Option>
    </Options>
  </ClientTraceRequest>
</InterTrackMessage>
```

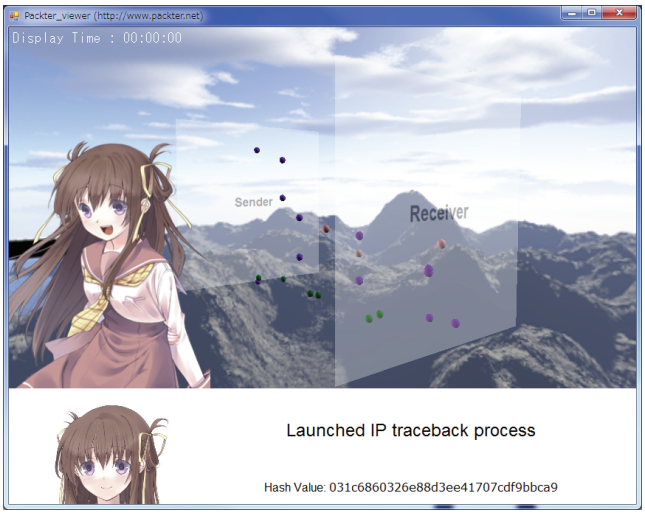
Figure 5: PACKTER_TC's client trace request

is called a message identification reply message which notifies that DP accepted the trace request. The other is a client trace reply message which informs the result of the trace request. Whenever a traceback trial succeeds, the client trace reply message contains some AS paths that the issued packet came from. Otherwise, the message says "notfound" instead of the AS paths. In short, "succeeded" means that the issued packet was found in the outside of the AS in the context of IP traceback.

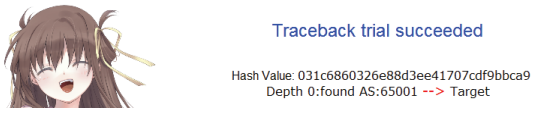
In order to inform DP's responses to the user of PACKTER viewer, PACKTER_TC then generates three kinds of alerts, namely, (i) the request was being accepted, (ii) the traceback trial was succeeded, and (iii) the trial was failed. In any cases, PACKTER_TC sends messages with PACKTER protocol format toward the viewer, that make the viewer play music, display text or HTML messages, avatars, and face icons. Figure 6(a), 6(b), and 6(c) demonstrate the cases of (i), (ii) and (iii), respectively.

4. CONSIDERATIONS

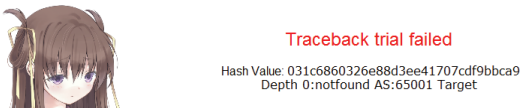
Whereas the number of the traffic visualization researches increases, the number of the useful implementation does not so much. Our project launched at August, 2008, however, there were and are very few visualization tools that can be available as free and open



(a) Launched IP Traceback Request



(b) Succeeded



(c) Failed

Figure 6: Execution of IP traceback trial

source softwares. According to SourceForge, roughly 17 projects were found, however eight of 17 were relevant to load, air, vehicular traffic rather than Internet traffic. In the rest of nine were mainly offline analysis tools and/or network simulators. Similar tendencies were seen at the other websites, including freshmeat, github, and Google Code. As we mentioned in section 1, our primary motivation is to integrate the functions of starting network forensics processes with a real-time traffic visualizater.

The major limitation in PACKTER is the number of flying objects. Even PACKTER utilizes GPU through Microsoft XNA Game Studio library, showing roughly 2000 or more objects makes the PC which runs the viewer become heavily loaded. When we attempted to monitor at our managed Internet exchange point, we configured to the agent with sampling rate 1/8192.

The secondary limitation is the number of the varieties of the supported network forensics; this paper focuses on cooperating to IP traceback, whereas various forensics have been proposed. To the best of our knowledge, network forensics often requires the pointer of forensics servers and the additional information for its forensics. Fortunately, these schemes can be easily supported as we employed description schema in PACKTER protocol for launching IP traceback processes.

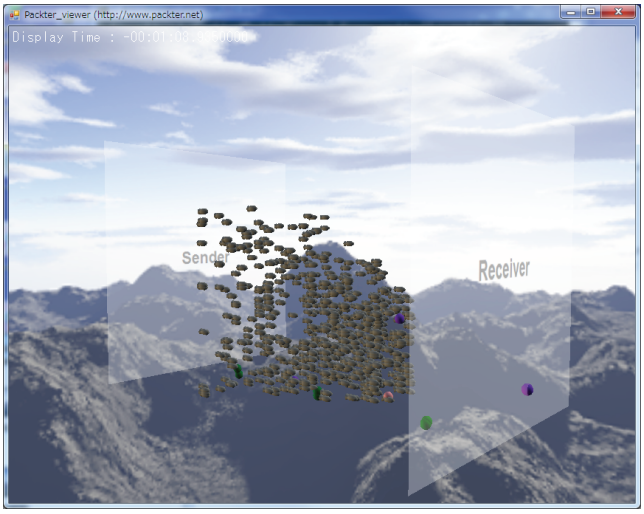


Figure 7: Missiles representing DoS attacks

The remaining issue is the way for informing anomalous activities to network operators. PACKTER supports to employ both polygonal models and their texture images, all of that can be specified by the operator. For example, PACKTER agent equips the function of detecting DoS by comparing the number of packets with the specified threshold. When the agent sends PACKTER message to the viewer with set of an unused flag number, the viewer looks up both mesh and texture corresponding to the flag number. Figure 7 shows the case of TCP Flooding, where missiles are detected DoS attacks.

PACKTER also supports the network traffic auralization, which means the technique of creation and reproduction of sound from the packet information. As we explained in section 2, PACKTER plays sound file and pronounces specified text messages by cooperating to a speech synthesis software. Of course, visualization tools are not so useful for a person with visual impairment, some other operational console should be considered for them, however, it was beyond the scope of this paper.

5. CONCLUSION

This paper has presented a network traffic visualizer and extended it for launching network forensics processes. Our developed PACKTER consisted of an agent program and a viewer program. The agent was designed to collect per packet information by monitoring network interface, reading a packet trace file, accepting flow sampling protocols, and receiving via Unix domain socket for cooperation to intrusion detection systems. The viewer was available to observe collected information via our defined PACKTER protocol, and drew the information in its three-dimensional screen; the each packet appeared at the sender board, and flowed toward the receiver board with animation.

We then added the function to cooperate to network forensics systems to PACKTER. Since the paper focused on starting IP traceback processes, we modified the agent to send a hash value extracted from the packet information. PACKTER also supported to inform such information to network operators that accepting the trace request and the results of the request.

Note that PACKTER is online available [7], and all source codes are released under BSD license, and media files such as pictures, textures, mesh objects, and sound files are released under CC-BY in Creative Commons license. We believe that our work will expedite the utilization of the traffic visualizer for supporting network operations.

6. REFERENCES

[1] Claise, B. Cisco Systems NetFlow Services Export Version 9. RFC 3954, Oct. 2004.

[2] Gong, C., et al. Single Packet IP Traceback in AS-level Partial Deployment Scenario. In Proceedings of IEEE Global Telecommunications Conference (Nov. 2005).

[3] Hazeyama, H., et al. An Autonomous Architecture for Inter-Domain Traceback across the Borders of Network Operation. In Proceedings of IEEE Symposium on Computers and Communications (Jun. 2006).

[4] Inoue, D., et al. nicter: An Incident Analysis System toward Binding Network Monitoring with Malware Analysis. In Proceedings of WOMBAT Workshop on Information Security Threats Data Collection and Sharing (Apr. 2008), pp. 58–66.

[5] InterTrack. IP Traceback : A mechanism to find attack paths. Available at: http://intertrack.naist.jp/.

[6] Phaal, P., et al. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176, Sep. 2001.

[7] Project PACKTER. PACKTER: A Multi Purpose Traffic Visualizer. Available at: http://www.packter.net/index_e.html.

[8] Snoeren, A.C., et al. Hash-based IP traceback. In Proceedings of ACM SIGCOMM Conference (Aug. 2001), pp. 3–14.

[9] Snort. The Open Source Network Intrusion Detection System. Available at: http://www.snort.org/.

[10] Stayer. W.T., et al. SPIE-IPv6: Single IPv6 Packet Traceback. In Proceedings of IEEE International Conference on Local Computer Networks (Nov. 2004), pp. 118–125.

[11] The Internet Society. PCAP Next Generation Dump File Format. Available at: http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html.