DOI: 10.21917/ijct.2018.0281

TRANSPORT TUNNELING SYSTEM WITH A COMMUNICATION HACKING FRAMEWORK

Satoshi Kodama[1] and Rei Nakagawa[2]
[1]*Research Institute for Science and Technology, Tokyo University of Science, Japan*
[2]*Faculty of Science and Technology, Tokyo University of Science, Japan*

Abstract

This paper aims to reduce the transfer restrictions of networks at the transport layer under prohibition-in-principle rules, and to introduce a communication system that virtualizes a programmable network function. The second aim facilitates the realization of the first. Specifically, we target Intranets that restrict transport-layer communications using a firewall. When such an organization's networks serve multiple departments, external communication to a department is restricted to the available transport number under the rule of the prohibition-in-principle. Our proposed transport proxy system architecture represents the transfer of network applications via well-known protocols such as HTTP (80), using the urgent pointer in the transmission control protocol header. Our architecture improves the flexibility and scalability of the network without requiring complex encapsulation. Finally, the framework is demonstrated through an experimental implementation of the system. Moreover, adding the transport tunneling system offered flexibility while barely affecting the download time of the files.

Keywords:

Software Defined Network, Private Network, Network Management System, Tunneling

1. INTRODUCTION

Currently, multiple network services are increasingly overloaded with traffic through a few well-known transport protocols such as HTTP [1]. Unless designated for specific uses that are previously authorized by the system, unknown transports are filtered out for security reasons. For example, consider a private network of users wherein an internal gateway manages each separated IP domain under the external gateway and the demilitarized zone (Fig.1). This tree-structured network is protected by a security policy imposed at the transport layer, which operates under the rule of the prohibition-in-principle. Although this restriction provides robust overall security, it tends to block communications between external terminals and specific departments by applying an unauthorized transport alert.

This problem can be naturally solved by re-configuring the security policy in firewalls, which process the inbound communication. However, the reconfiguration must be performed on several devices, which introduces system complexity and compromises security. This study proposes a transport tunneling (TT) system that solves the access problem without changing the core security system. The TT system is based on a simple bridge that extends the programmability of software defined network (SDN)/network function virtualization (NFV). The system represents the transfer of a network application via a well-known transport protocol such as HTTP (80) using the urgent pointer in the transmission control protocol (TCP) header, increasing the flexibility and scalability of the security without requiring complex encapsulation, such as generic routing encapsulation [2]. The simple bridge (a full-software component written in C language) mitigates the processing limitation of the network hardware (e.g., switching between wireless and wired network interface controllers (NICs)).

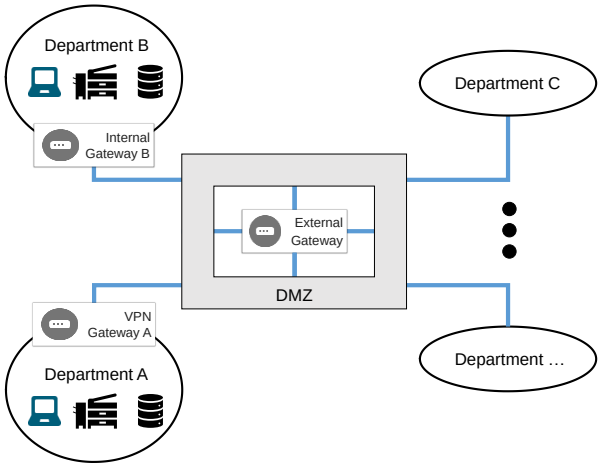

Fig.1. Intranets in an organization

The contribution of this research is as follows. Firstly, this paper is based on extremely practical grounds, inviting simple and important attention to the security field. Secondly, the findings revealed in this paper refer to the availability of opaque and unused parts in the IPv4 network established as infrastructure, and can be applied in all current network systems. Thirdly, the bridge system based on the argument in this paper shares the basic idea of realizing important network virtualization in the future theoretical network field. In other words, it is easy to expand the theory in future networks.

The remainder of this paper is organized as follows. Section 2 overviews related work on SDN/NFV technology, which is the inspiration for the underlying framework of the TT system (i.e. the simple bridge). Section 3 introduces our TT system architecture, which represents the transfer of network transport using a pseudo-port number and an urgent pointer. The required components of the system are presented in this section. Section 4 evaluates an experimental test system, namely, a simple TCP connection with files downloaded over transports from a server to a client. The paper concludes with section 5.

2. APPLICATION OF SIMPLE BRIDGE

The SDN/NFV technology is becoming an established means of improving system scalability. SDN/NFV provides a programability of network functions and easy system management of the layer structure between the control and the data plane. Solutions exploiting SDN/NFV have been widely investigated [7] [8]. Most of these schemes aim to improve the functionalities of SDN/NFV

(e.g., service integration and implementation of state-of-the-art networks) rather than its programmability. Consequently, SDN/NFV based tools such as openVswitch increases the complexity of the system, and hence, we consider that the original flexibility is being lost.

To resolve this problem, we introduce a simple bridge system that hacks communication between physical or virtual NICs and passes the modified traffic to the process bound to each NIC. This unit is the minimum system of programmability in SDN/NFV [4-6]. For the developer, the bridge system maximizes the flexibility and scalability of the programming rather than normalizing the functionality. As a solution to the access problem, this paper applies the bridge system to the TT system in section 1.

3. SYSTEM ARCHITECTURE

This section presents the proposed TT system architecture, which adds a port translator to each end user (Fig.2).

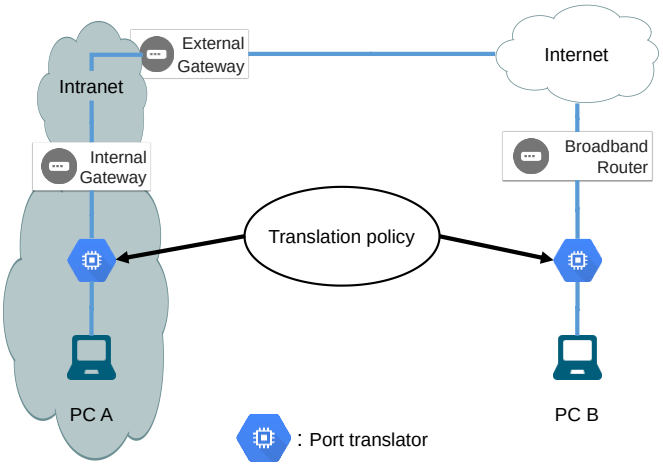

Fig.2. Proposed System Architecture

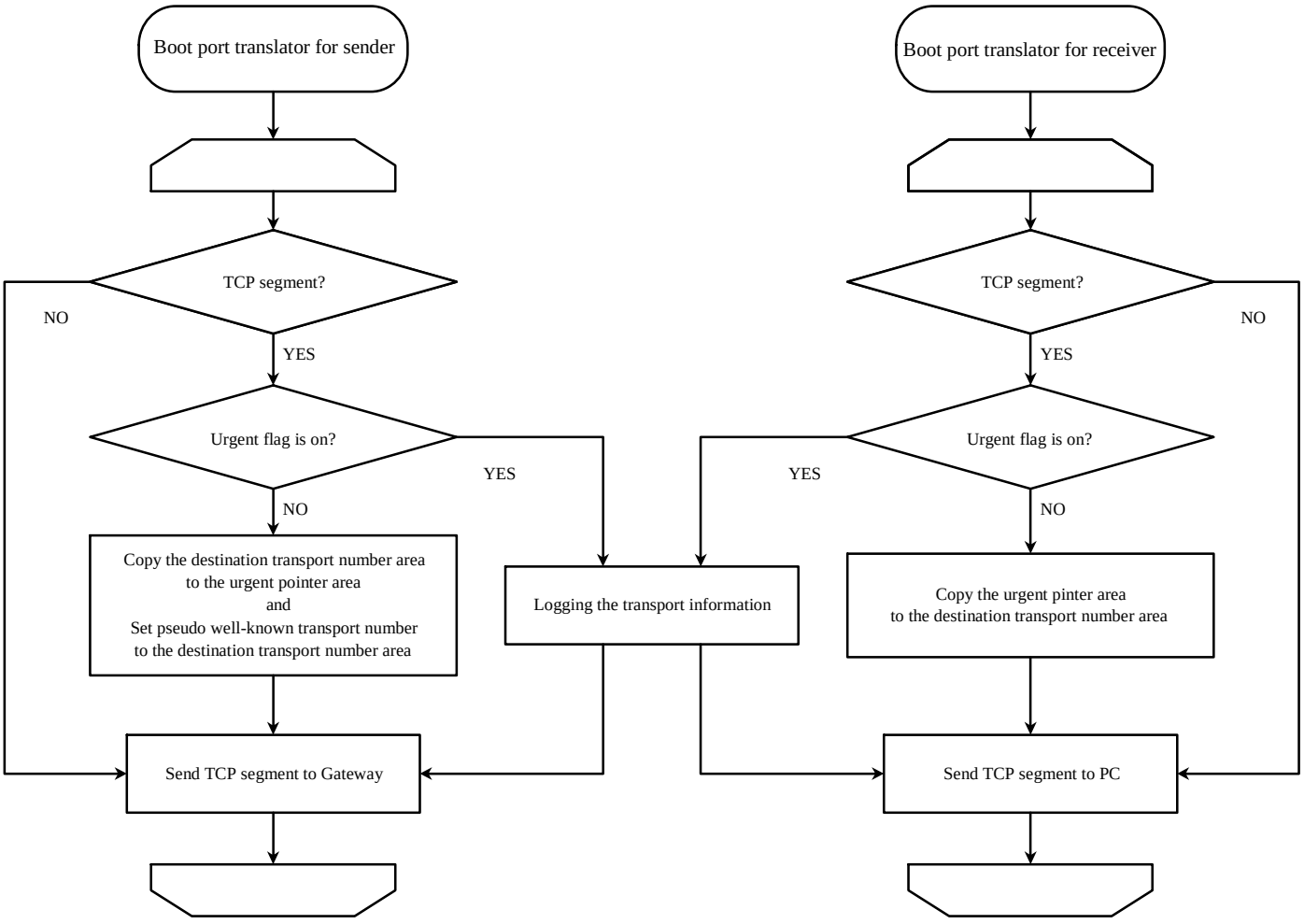

Fig.3. Translation Process

3.1 USAGE SCENARIOS

In Fig.2, two PCs, A and B, have set up separate IP domains on the Intranet and Internet. The internal gateway and broadband router are the default gateways for both computers. The external and internal gateways associate Intranet with the transport security under the prohibition-in-principle rule. In this network topology, B cannot normally communicate with A through an unfamiliar transport number because such communications are blocked by the transport security.

To solve this problem, we add a port translator to each network domain. The port translator acts as a sub-router, representing the transfer of any transport number from a PC. Each port translator monitors the rewriting of the transport number by the TCP header and the urgent-pointer area of the header for the transport proxy. The Fig.3 presents the translation process of the sender and receiver. The sender sets the original transport number in the urgent-pointer area and the pseudo-well-known transport number in the destination-transport number area. The receiver follows the same process. In reverse communication, each process handles the source-transport number instead of the destination-transport number. These processing steps are executed through a communication workshop, as described in section 4. Our proposed system thus resolves the access problem and enables bidirectional communication beyond the Intranet.

3.2 ADVANTAGES AND ISSUES

The proposed system enables easy tunneling of any TCP service and application to an Intranet terminal. In other words, the user can easily respond to a transport request without changing the security policy of the entire Intranet system. Moreover, the system requires no complex encapsulation, which is expected to improve the performance and usefulness of networks with isolated IP domains.

However, the packet location of the embedded transport information is potentially problematic. The urgent-pointer area used by the system is assumed to be unchanged by the security policy, but this assumption requires further discussion (Section 5). Therefore, the future scalability of the system is not guaranteed. A leading alternative embedding candidate is the media access control (MAC) address, which can be rewritten relatively freely [3] [4]. Therefore, the MAC address is gaining traction as an embedding location for new information.

3.3 COMMUNICATION WORKSHOP

Our communication workshop allows port translators to process communication packets, as shown in Fig.4.

Before developing this workshop, we studied communication workshop implementations that enable the rewriting of communication headers (such as Ethernet, IP, or TCP headers) [4-6]. The packet-processing functions of our workshop are similar to those of OpenFlow's data plane [7]. It also provides a simple and powerful virtualization of the network function [8] that can be arbitrarily defined by developers in the C language. The type of terminal adopting the communication workshop influences the workshop architecture. For instance, an end terminal uses a virtual NIC (vNIC) as a Southbound NIC. The communication workshop in our system processes packets bound for the physical NICs (Northbound and Southbound pNICs) without assigning IP addresses. All packets through the pNICs are streamed to the user space via the GNU/Linux kernel. The workshop then branches the packets' protocol headers, rewriting them as discussed in section 3.

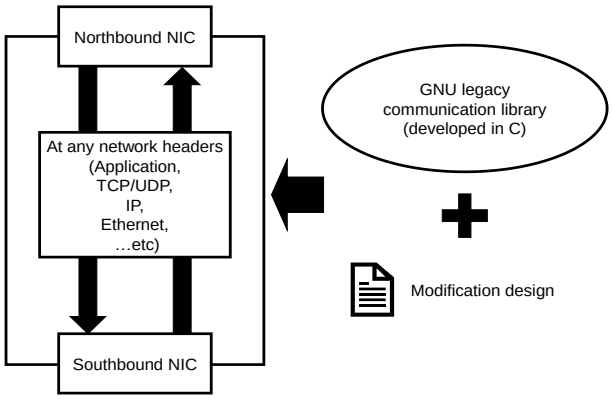

Fig.4. Communication Workshop

3.4 URGENT POINTER

The scalability of the TT system depends on the embedding location of the transport information. The urgent pointer field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. Meanwhile, the urgent pointer points to the sequence number of the octet following the urgent data. This field is interpreted only in segments with a set URG control bit [9].

Gont and Yourtchenko [10] pointed out the security hazards of using the urgent pointer. They maintained that applications delivered through the TCP urgent mechanism are vulnerable to multiple factors. Therefore, the network intrusion detection system (NIDS) cannot robustly track the application-layer data transferred to the destination system, leading to false negatives or false positives in the NIDS. To avoid these problems, they recommended abandoning the use of TCP urgent mechanisms altogether. Although the urgent data could be "aligned" with packet scrubbers configured to clear the URG bit and set the urgent pointer to zero, this solution risks interoperability problems or aberrant behaviour in applications relying on the TCP urgent mechanism, such as Telnet [10].

The TT system uses the urgent pointer without turning on the urgent flag. This design admits services that are originally blocked from the Intranet, as confirmed in the Internet experiment. Although the TT system exhibits good performance in the current study, the security risks associated with urgent pointers cannot be ignored.

4. EXPERIMENTAL EVALUATION

This section demonstrates our TT system in a simple experiment that confirms the transport-number rewriting, connectivity, and performance of the system.

4.1 AIM

In this experiment, we connected a TCP exp-connection to a TCP echo server (transport number 9000) through the network shown in Fig.5. We proved the availability of the TCP connection (9000), which is prohibited in principle by the Intranet, using the

port translator to exploit the urgent pointer. We also confirmed that the TT system delivers no performance hits to the throughput by comparing the download times of [] and [] using the "wget" command. In the absence of a port translator, the client usually downloads files from the HTTP (80) server. Otherwise, PC A downloads files from the HTTP (9000) server through TT using the port translators. The download command over HTTP (9000) is "wget http://113.42.*.*:9000/file_names." Thus, we verified that the client and server connected and communicated over the port translator. In the absence of the port translator, communication with clients outside the Intranet is blocked by the TCP exp-connections.

4.2 PROCEDURE

In this section, we explain the stepwise procedure of the experiment. First, the client investigated the connectivity to HTTP (80) without the TT system in the Intranet. After confirming a completed connection to HTTP from the server, the client set 80 as the pseudo-well-known transport number. Prior to the experiment, the server started the TCP echo (9000) server. After completing the experimental preparation, the client started the TCP exp-connection. The port translator operates according to the process shown in Fig.3. For example, the communication flow from the client to the server should process the follows. First, the client emits a packet with dst-port (destination transport number): 9000 to the server. Second, the port translator A capture the packet and transplant all bytes in dst-port field to the urgent pointer field on the packet, and also set the pseudo-well-known transport number (80) at the dst-port field on the packet (9000 to 80). Third, the port translator B capture the packet and transplant all bytes in the urgent pointer field to the dst-port field, and also clear the urgent pointer field with zeroes. Finally, the server receives the packet. In reverse communication flow, these process work on the src-port (source transport number) field on a packet.

Table.1. Specification of equipment

Network Device	Specification
Port Translator A, B	Raspberry Pi 3 running CentOS 7 CPU: 1.2GHz 64-bit quad-core Memory: 1GB
Client and Server	Workshop running Windows 10

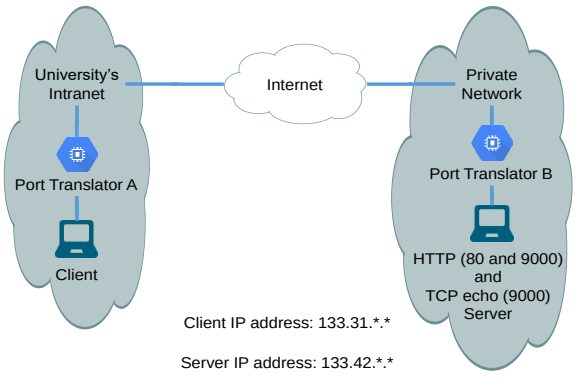

Fig.5. Topology for the experiment

4.3 RESULT

Without the port translator, the client communicated with the HTTP (80) server (Fig.6), but was excluded from the HTTP (9000) server, and was issued with an error message, enclosed in a red box, saying "ERR_CONNECTION_TIMED_OUT" (Fig.7). As shown in Fig.8 and Fig.9, the TCP echo connection was successful both with and without the port translator. The Table.2 compares the averaged download times and communication throughputs in the presence and absence of the transport translator when obtaining files of different sizes (10MB, 50MB, 100MB, and 500MB). In each experimental run, the files were retrieved 10 times from the server over HTTP (80 and 9000).

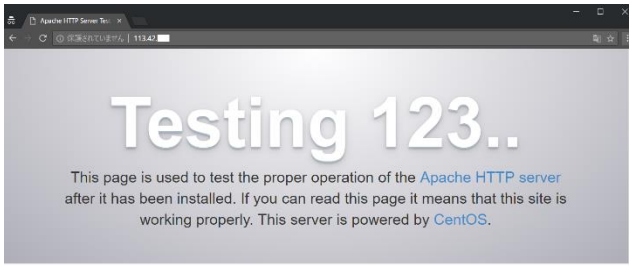

Fig.6. HTML page from HTTP (80) server without TT system

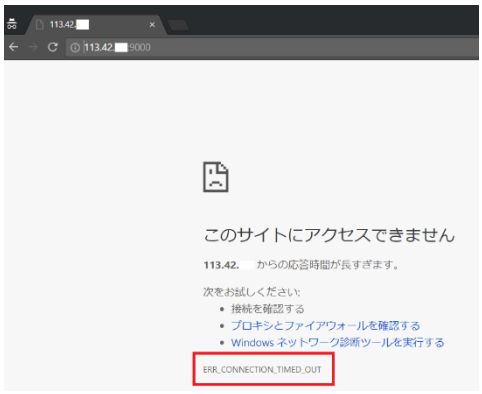

Fig.7. HTTP (9000) connection error without TT system

Fig.8. TCP echo connection error without TT system

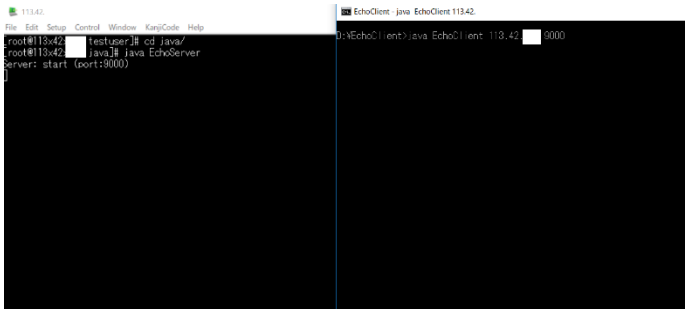

Fig.9. TCP echo connection worked with TT system

Table.2. Download Time

File Size (bytes)	Download Time (s) (with TT system)	Download Time (s) (without TT system)
10M	2.24 [+0.02]	2.22
50M	11.60 [+0.60]	11.00
100M	22.40 [-0.60]	23.00
500M	113.40 [+1.20]	112.20

In the TCP echo experiment (Fig.8 and Fig.9), Fig.8 shows the TCP connection (9000) refuse without the TT system, and Fig.9 shows the behaviors of the server and client sides were displayed in the left and right sides of the terminal window, which indicates that the TCP connection (9000) is finished successfully with the TT system. Clients began the experiments under their own initiatives, and the server was remotely operated by Secure Shell. In Table.2, square brackets enclose the differences in the download times between the operation and non-operation of the TT system. The differences clarify the network performance after adding the TT system. (For security consideration, we removed some of the IP addresses in Fig.6-Fig.9).

5. CONCLUSIONS

We proposed a TT system architecture that aims to represent the transfer of any network application by means of a well-known transport protocol, such as HTTP (80), using the urgent pointer in the TCP header.

Our system enables autonomous service deployment without changing the Intranet security policy. This advantage was demonstrated through experiments conducted on the Intranet.

In future work, we will consider the problem of security vulnerability. First, we must establish the degree to which the urgent pointer compromises the security strength. Second, we will consider cases of misuse of the TT system. Finally, we will evaluate both the urgent pointer and TT system from a security perspective.

REFERENCES

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2015-2020", Available at: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html, Accessed on 2015.

[2] D. Farinacci, T. Li, S. Hnaks, D. Meyer, P. Traina and Juniper Networks, "Generic Routing Encapsulation (GRE)", Available at: https://tools.ietf.org/html/rfc2784.

[3] Peter Kietzmann, Cenk Gundogan, Thomas C. Schmidt, Oliver Hahm and Matthias Wählisch "The Need for a Name to MAC Address Mapping in NDN: Towards Quantifying the Resource Gain", *Proceedings of 4th ACM Conference on Information-Centric Networking*, pp. 1-6, 2017.

[4] Satoshi Kodama, Rei Nakagawa, Toshimitsu Tanouchi and Shinya Kameyama, "Management system by using Embedded Packet for Hierarchical Local Area Network", *Proceedings of IEEE International Conference on Ubiquitous Computing, Electronics and Mobile Communication*, pp. 113-119, 2016.

[5] Satoshi Kodama, Rei Nakagawa and Toshimitsu Tanouchi, "Proposal of the Virtualized Control System for the Integrated Management of Multiple Services", *Proceedings of IEEE 7th International Conference and Workshop on Annual Computing and Communication*, pp. 23-27, 2017.

[6] Satoshi Kodama, Rei Nakagawa and Toshimitsu Tanouchi, "A Research on the Integrated Virtual Platform for Managing Multiple Services", WSEAS *Transactions on Information Science and Applications*, Vol. 14, No. 12, pp. 102-111, 2017.

[7] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker and Jonathan Turner, "OpenFlow: Enabling Innovation in Campus Networks", *ACM SIGCOMM Computer Communication Review*, Vol. 38, pp. 69-74, 2008.

[8] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck and Raouf Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges", *IEEE Communications Surveys and Tutorials*, Vol. 18, No. 1, pp. 236-262, 2018.

[9] J. Postel, "Transmission Control Protocol", Available at: https://tools.ietf.org/html/rfc793, Accessed on 1981.

[10] F. Gont and A. Yourtchenko, "On the Implementation of the TCP Urgent Mechanism", Available at: https://tools.ietf.org/html/rfc6093, 2011.