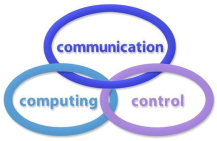
INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 19, Issue: 2, Month: April, Year: 2024
Article Number: 4906, https://doi.org/10.15837/ijccc.2024.2.4906



CCC Publications

Framework for evaluating TCP/IP extensions in communication protocols

R. Mantu, M. Chiroiu, N. Țăpuș

Radu Mantu
Department of Computer Science
University POLITEHNICA of Bucharest
radu.mantu@upb.ro

Mihai Chiroiu*
Department of Computer Science
University POLITEHNICA of Bucharest
*Corresponding author: mihai.chiroiu@upb.ro

Nicolae Țăpuș
Department of Computer Science
University POLITEHNICA of Bucharest
nicolae.tapus@upb.ro

Abstract

Most of the network and transport layer protocols had been designed with consideration for future extensions. As a result, variable-length sections had been devised in order to expand the fixed-size headers and allow the annotation of packets with metadata, known as protocol options. Although numerous enhancements such as MultiPath-TCP are based on these mechanisms, their support in the Internet is still generally considered to be opaque.

In this paper we introduce an extendable packet annotation tool for IP, TCP and UDP options, based on NetfilterQueue. Using this tool in conjunction with our testing framework that can incorporate multiple regions, we conducted a series of experiments to determine the acceptance rate of each type of option and whether new extensions will be readily supported in the Internet. Additionally, we discuss particularities of cloud provider infrastructures in dealing with certain options.

Keywords: Networking, Measurements, Middleboxes, Protocol extensions, Testing infrastructure.

As the Internet evolves, so do the expectations for new network protocols. Whether said protocols pertain to traffic encryption, packet source authentication or routing optimization, all are faced with the same initial hurdle: the uncertainty of their compliance with the myriad filtering policies employed by virtually just as many middleboxes. This incertitude has only been exacerbated by the continual ossification of the foundational internet protocols. An immediately apparent indicator of

this suggested entrenchment is the maladroit adoption of new standards and acceptance of implementations conforming to well-established specifications. We draw attention to RFC 7126 [10], offering recommendations on filtering packets with IPv4 options due to widespread misuse. Its necessity after upwards of thirty years since this IP extension mechanism was provisioned in RFC 791 [16] is epitomic of the issue at hand.

Regardless of the underlying causes that lead to this predicament, the objective is not necessarily to remedy but to circumvent the obstacle that it poses. As things stand, the research questions that need to be addressed are as follow.

RQ1: Is the current state of affairs in the wider Internet adequate for already standardized protocol options? Extant protocols had been designed under the assumption that the existing fixed-length headers would at some point become insufficient for encapsulating the information needed by supplementary mechanisms. Consequently, most of them included a variable-length options feature allowing their incorporation as the need arises. The benefits are most evident when observing how TCP is reinforced by the Window Scaling option (increases the maximum window size from 64Kb to 1Gb) or the Selective Acknowledgement option (allows specifying a finite range of lost packets). In their absence, TCP would cease to be viable in high-latency, high-bandwidth environments such as WANs. While options were initially overlooked in lieu of redefining existing fields to better fit specific purposes of certain users, the approach has been thoroughly discouraged by the open standards organizations and the community at large. Despite the growing demand for extending current protocols, Autonomous Systems (AS) proved resistant to this change. Although some options have been standardized, their acceptance remains highly dependant on middlebox manufacturers.

RQ2: Can new protocol options be attached to a packet without compromising its conformity to the filtering schemes? When the aforementioned protocol extension systems were first defined, a number of option identifiers (i.e.: codepoints, kinds) were designated to features immediately necessary at the time (e.g.: the IP Security option or the UDP Authentication and Encryption option). In order to properly administer the growing number of independent projects and prevent the unauthorized usage of unassigned codepoints from becoming praxis, IANA had reserved certain ranges for this purpose [7]. This, in turn, lead to the introduction of experimental IDs [18] as a method of sharing the yet limited assigned ranges, when testing in common environments. Nonetheless, there have been known cases of unpermitted use of certain codepoints, some overlapping with established options (e.g.: TCP User Timeout). Seeing how these unregistered and experimental options are practically unknown to most middleboxes, an understandable concern is whether they will be accepted without any further verification or simply dropped.

RQ3: Does the annotation scheme under consideration interfere with the base functionality of any protocol or application? Assuming that a modified packet is able to traverse the network unthwarted, the need to ensure normal operation of applications whose traffic falls within the purview of the scheme under test is paramount. Ideally, the packet alteration is handled outside the scope of the application itself, either in kernel (e.g.: DCCP's Explicit Congestion Notification) or by deference to another userspace process (e.g.: tcpcrypt's packet manipulation using NetfilterQueue). However, this is not always the case. Employing kernel bypasses can be rightfully motivated by practical limitations, such as IRQ storms during DDoS attacks (most CPU time is used to receive packets, not process them). During such attacks, `iptables` reaches a state of saturation at approximately 1M packets per second (pps). While solutions based on eBPF are known to be able to drop up to 15Mpps and quickly recover from a state of FIFO tail dropping, the technology is still new and imposes stringent constraints on the developer. Meanwhile, solutions based on frameworks such as Intel DPDK or PF_RING are more likely to be incompatible with any new annotation scheme. Consequently, a large-scale testing environment is required to validate any new addition to well-established protocols.

We claim the following contributions:

- We offer a tool[1] capable of intercepting specific packets and annotating them with user specified, per-protocol option types. The options are generated in their entirety by a decoder that allows for easy integration of new option types, as well as protocols.

[1] https://github.com/RaduMantu/ops-inject

- We propose a framework for assessing an annotation system's compliance with firewall policies and provide configuration scripts and templates for cloud experimentation under the primary available providers (i.e.: Google Cloud, AWS, Microsoft Azure, Digital Ocean).
- We evaluate current layer 3 and layer 4 protocol extension mechanisms and deliberate their suitability as a basis for developing new extensions.

We discuss the architecture of our system in Section 1. Then, in Section 2 we present our observations and offer recommendations based on empirical results. In Section 3 we discuss related work. Finally, we present our conclusions in Section 4.

1 Architecture

1.1 Background on related technologies

NetfilterQueue is an extension to the packet filtering mechanism built into the Linux kernel. This extension takes the form of a non-standard `iptables` target: `NFQUEUE`. By making use of the `nfnetlink_queue` subsystem (and superseding `ip_queue`), NetfilterQueue allows judgement with regard to the acceptance of a packet to be deferred to a userpsace process (see Fig. 1).

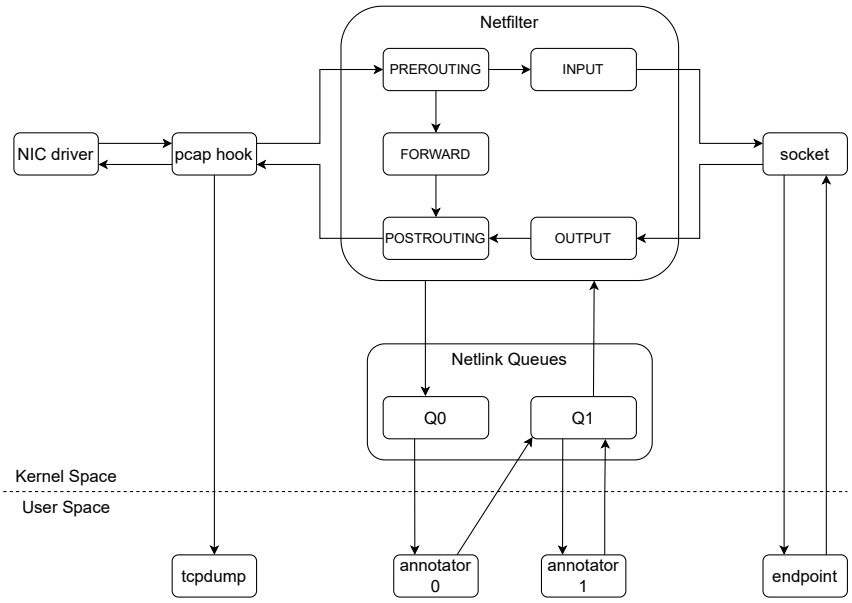


Figure 1: Packet data path when matching an iptables rule with a NFQUEUE target

Specifically, once a packet reaches a Netfilter hook that corresponds to one of the main five chains (i.e.: PREROUTING, INPUT, etc.) and satisfies the match criteria of a specific `iptables` rule, the packet is enqueued, pending further analysis. After a userspace process subscribes to the very same queue, the packet is copied to its virtual address space, starting with its layer 3 header, via a Unix Domain Socket. The process is allowed to inspect the packet, and even to further buffer it while requesting subsequent matches for out-of-order processing. Eventually, a verdict must be reached and communicated to the kernel as one of the standard terminal targets in `iptables` (e.g.: ACCEPT, DROP, etc.)

The reason this extension is useful to us is not necessarily the filtration aspect, but the fact that once the verdict has been established, the process can supply a modified version of the packet. This new packet will overwrite the one initially captured by the Netfilter hook. Given a positive verdict, the modified packet will be reinserted into the network stack, following the intended data path and reaching its endpoint. Note, however, that the userspace process can additionally specify another queue to be used for reinsertion. If this is indeed the case, the potentially modified packet will be stalled

once again for analysis. The process performing the second inspection (i.e.: the process subscribed to the second queue) does not necessarily need to be the same process as before.

1.2 Annotation tool

In testing different features of layer 3 and layer 4 protocols, a usual approach consists of generating synthetic traffic and bypassing the network stack of the originating host. We decided to forgo this method in favor of a more practical alternative: modifying real traffic. The primary advantage of the latter scheme over the former is the ability to verify not only that the annotated packets can be successfully transmitted over the network, but also that their alteration does not impact higher level protocols in the OSI stack. Additionally, the flexibility in assessing compliance with stateful firewalls alleviates the effort that would have otherwise been required to simulate sufficiently credible sessions.

These aspects gave impetus to the development of a method to insert multiple protocol specific options in outgoing traffic. One of the essential characteristics of the resulting tool is the capacity to easily implement new options for existing protocols or to register new protocols altogether. Limited by the extent of our needs for this experiment, session tracking falls outside its purview. As a result, options such as MultiPath TCP are not available for testing. Instead, all options should either be static in nature (e.g.: NOP) or depend entirely on the information available in a single packet's header and payload (e.g.: alternative checksums). Otherwise, its use may be limited to determining only whether the initial packet was able to traverse the network (e.g.: TCP timestamps).

Next, we present a summary overview of the tool's operation. Initially, the user adds one or more `iptables` rules with an `NFQUEUE` target. All packets that match one of these rules are redirected into userspace for our tool to evaluate and modify as it sees fit. During the initial invocation of the tool, the user may specify a sequence of bytes that represent option codepoints. For each packet, the codepoints will be expanded to actual TLV entries and inserted into a (potentially new) options section. The NetfilterQueue callback that fulfils this purpose takes three steps towards successful annotation:

Options Decoding: having access to the packet, as provided by the Netfilter kernel hook, the first step is generating the contents of the options section for a pre-established protocol (i.e.: IP, TCP, UDP). Using the user-specified sequence of `option-type` octets, specific decoder functions are called upon to expand them, thus populating the options buffer. This expansion phase is by and large an arbitrary process. For example, an IP Timestamp Option (`0x44`) will be translated to a 12 octet buffer containing a single timestamp associated to the source IP address. The generated flags will denote pre-specified address fields as to discourage addition of new timestamps from compliant middleboxes. This inhibition of user agency is a conscious decision intended to simplify the tool's usage. Nonetheless, the decoder functions can be easily modified on a case-by-case basis to accept more information and conform to finer-grained requirements. Note that the options will appear in the order in which they were initially specified by the user. However, the order in which they are generated may vary. For example, a UDP Options Checksum (`0x02`) is supposed to be placed as close to the beginning of the section as possible, while at the same time requiring all other options to have already been calculated. Consequently, it's decoding is delayed based on a discretionary option priority assignment.

Packet Reassembly: This step modifies the initial packet in order to include the newly generated options. Any related fields (e.g.: `IP.total_length`) with the exception of the checksum are adjusted to account for the payload offset. Based on user preference, existing options can either be completely replaced, or preserved. In our tests, we employed the former approach in order to exert full control over the content of the options section. Alternatively, existing options would take precedence. In turn, this could lead to overflowing user-specified options to be discarded due to space constraints.

Checksum Recalculation: The reason why this step is separate from the previous is that changes made to a header corresponding to a lower layer can impact the checksum of a higher layer protocol. For example, while IP options are intended to extend only the IP header, the total packet length increases nonetheless. As a result, TCP's or UDP's pseudo-header changes and the layer 4 checksum does so along with it.

An advantage of relying on the Netfilter framework is that it integrates well with `ip-xfrm` (kernel-side module that implements the IPsec protocol suite). Usually, VPN implementations are split across both userspace and kernelspace. The former carries out the negotiation required as per the

IKE protocol, which establishes up to four Security Associations (SA) and creates a Security Policy (SP). The latter utilizes the SAs for packet encapsulation and encryption but more importantly, it uses the SP to determine which packets are to be passed to the IPsec stack. This decision is taken by consulting an SP database after the packet has traversed the `OUTPUT` and `POSTROUTING` chains. If appropriate, the packet is modified and reinserted prior to entering the `OUTPUT` chain. The `iptables` rule that identifies the packets which require annotations must be well-defined. Otherwise, the user risks modifying a packet that is protected by the ESP protocol. We consider this approach preferable to excluding packets containing ESP headers. Due to the potential usage of NAT-traversal encapsulation, it would be inefficient to identify the presence of the ESP header. Moreover, allowing UDP packets with destination port 4500 to pass may exempt IKE configuration packets from annotation, contrary to the user's intention.

1.3 Testing framework

Irrespective of the type of protocol that may benefit from these extensions, they will in all likelihood be required to interact with cloud-hosted solutions [3]. As a result, we focus our efforts on verifying if and how cloud providers allow the passage of annotated packets through their networks. To this end, we perform a series of inter-regional tests. The structure of the testing environment can be seen in Fig. 2. Initially, the management instance (i.e.: localhost) will install the annotation tool, as well as any traffic generation tools under test on each virtual machine. Additionally it will also start `tcpdump` processes and insert specific `iptables` rules with the purpose capturing relevant traffic for annotation and later analysis. When testing a certain instance (e.g.: the one in me-south-1 region in AWS), a server will be started (1) by issuing a command over a ssh channel. Immediately after, all other instances will be issued similar commands, to start a client (2) and send requests to the server. The resulting traffic will be annotated in both directions. After all communication (3) will have ceased, another instance will be selected as the server. Finally, all `tcpdump` processes will be killed and the generated packet captures as well as any logs will be downloaded for analysis.

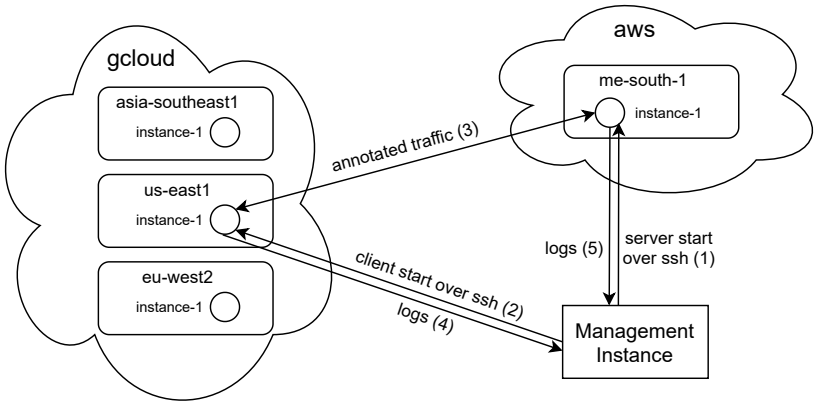


Figure 2: Architecture of the cloud testing framework.

The framework is comprised mostly of bash scripts using each provider's CLI management tool. A minimal setup is required: configuring default projects and public keys, enabling billing, and relaxing the firewall rules of their private networks to the extent that they are permitted. The scripts can be classified as management scripts, experiment scripts or analysis scripts, depending on their purpose. Management scripts allow the automatic creation or deletion of VM instances in accordance with a pre-specified list of regions. This list can be freely modified in accordance with the available selection of each cloud provider. Additionally, they administer the configuration of VPNs or that of the annotation tool. Experiment scripts install missing dependencies and generate traffic between any two instances. The commands executed on remote machines are issued over a secure SSH channel. All relevant inbound and outbound traffic is recorded on each instance and eventually imported on the managing

Table 1: Percentage of annotated traffic received *from* any other host and *by* any other host. Grouped by option type and averaged per provider. Data collected from multiple types of experiments (expts).

Protocol (expts)	Option	Acceptance Rate (%)									
		Received by					Received from				
		gcloud	aws	azure	docean	personal	gcloud	aws	azure	docean	personal
IP (8)	4x NOP	0	3.00	0	5.50	4.37	1.50	1.75	1.00	4.25	4.37
	Record Route	0	3.25	0	6.00	5.00	1.37	1.75	2.00	4.25	4.37
	Timestamp	0	3.08	0	5.91	4.37	2.00	1.75	1.00	4.25	4.37
	Experimental	0	2.75	0	5.75	0	1.50	1.50	0.87	3.75	4.37
	Unassigned	0	2.75	0	4.87	0	1.50	1.50	0	3.75	4.37
TCP (4)	4x NOP	100	100	100	100	100	100	100	100	100	100
	Echo	100	100	100	100	100	100	100	100	100	100
	Echo-Reply	100	100	100	100	100	100	100	100	100	100
	Timestamp	100	100	100	100	100	100	100	100	100	100
	Experimental	100	100	100	100	100	100	100	100	100	100
	Reserved	100	100	100	100	100	100	100	100	100	100
UDP (3)	4x NOP	0	68.00	75.00	75.00	73.33	0	74.83	72.00	71.33	72.50
	4x NOP & CCO	0	68.50	75.00	74.83	73.33	0	74.66	72.50	71.33	72.50
	Timestamp	0	72.50	74.83	74.66	74.16	0	75.00	74.50	72.33	75.00
	Timestamp & CCO	0	72.16	74.66	72.33	75.00	0	74.16	73.66	71.50	74.16
	Experimental	0	72.50	75.00	75.00	75.00	0	75.00	74.50	73.00	75.00
	Experimental & CCO	0	72.50	75.00	75.00	75.00	0	75.00	74.50	73.00	75.00
	Unassigned	0	72.50	75.00	75.00	75.00	0	75.00	74.50	73.00	75.00
	Unassigned & CCO	0	72.50	75.50	75.00	74.16	0	75.00	74.50	72.16	75.00

host. Any analysis of packet reception and option removal or alteration can be done offline. To this end, we offer a subset of the analysis scripts that we employed in interpreting the network captures, along with samples of said captures. We include only general-purpose scripts that, for example, report overall instance reachability or retrieve AS numbers and names of recorded IP addresses.

Note that this framework is not specifically aimed at evaluating cloud providers. Their preponderance in our tests was meant to verify the compliance to the protocol extension standards in the wider Internet. Given that the credentials are correctly configured on all machines, the experiments can be executed with either a statically defined set of public and private IPs, or via a user-provided means of dynamically generating said set.

2 Evaluation

Our experimental setup consisted of twenty virtual machines, spawned in five different regions per each tested cloud provider. The principal criterion that influenced our choice of regions was geographical diversity, both within the same provider's selection and across all instances. As a result, regions that were uniquely available from a certain provider (e.g.: Middle East) took priority over the more common ones (e.g.: Europe). Each instance ran the default Ubuntu 18.04 image offered by its corresponding provider and was allocated 1-2vCPUs, 1-4GB RAM (depending on the available flavors) and a public IP address. In addition to these, we also included a Ubuntu 20.04 VM instance running on our personal server, over VMware vSphere 7 and behind a Palo Alto 5540 Firewall. During our tests, we established over 50k connections over a period of approximately three weeks. Due to the resulting overall cost of US$150, we consider that our framework usable in repeated testing.

The experiments themselves consisted of sequentially selecting an instance from the available pool and transmitting annotated traffic from all other sources. For IP options, the protocols that we tested were ICMP (Echo Request), DNS and NTP over UDP, FTP, HTTP and TCP-FastOpen over TCP, as well as raw data on arbitrarily chosen free ephemeral ports (using both UDP and TCP). UDP and TCP options were similarly tested over their respective subgroups from the previously stated selection. A complete account of all tested options can be found in Table 1, as well as their respective acceptance rating, grouped by cloud provider. The "Received by" section, for example, represents the amount of unaltered, annotated traffic that reached instances belonging to a certain provider. This includes experiments initiated from different regions of the same provider. For instance, an IP Record Route

value of 3.25% for AWS is relative to five instances per cloud provider, twenty sets of experiments initiated by other hosts per instance, and 8 experiments testing different higher layer protocols per set. We note that in some cases (e.g.: IP Timestamp option) we conducted more than twenty sets of experiments per instance, in order to obtain sufficient data for other forms of analysis. Consequently, the results are averaged across these batches of experiments. Note that each experiment focuses on a single option at a time; due to limited resources, we do not test combinations of options.

For all protocols, we included a test consisting of four consecutive NOPs (No Options). Normally this option is used as padding in solving alignment issues. However, no more than three NOPs would be required if this was indeed the case. Four or more NOPs could either indicate that options had been purposefully removed during the network traversal by middleboxes, or they could be construed as a relatively weak DoS attack meant to exhaust CPU cycles by means of processing inconsequential options. Due to this ambiguity, we surmised that firewall manufacturers may prove less lenient towards such occurrences. Nonetheless, our tests revealed that the 4xNOP case did not impede communication in instances where other options lead to successful network traversals, regardless of protocol.

As one of our leading goals, we resolved to verify not only the acceptance of well-established options, but also that of unassigned or reserved ones. Similarly to the four NOP case, an undefined option can also be evaluated as malicious. While the former could be motivated by the arbitrary removal of interspersed options by routers, the latter could be considered a less ambiguous attempt at a DoS by certain firewalls [19]. Consequently we implemented two variations of an undefined option. The first option utilizes an unassigned or a IANA-reserved codepoint for each protocol (IP: `0x5d`, TCP: `0x47`, UDP: `0x7d`) which fills the data field with consecutively-valued data bytes until the next 32-bit boundary in the options section. The second option is implemented almost identically. The only difference is that it follows the recommendation on shared use of 16-bit experimental IDs and uses the `0xfe` codepoint for both the TCP and UDP instantiation.

In the following subsections we discuss each IP, TCP and UDP option that we tested, identifying practical applications and deciding their suitability for large scale deployment.

2.1 Analysis of IP options

After testing multiple IP options in combination with different transport layer protocols, we concluded that with the exception of ICMP messages, all packets are consistently dropped. For this reason, our following analysis will focus primarily on the idiosyncratic behaviour of certain providers towards each of the examined options.

Timestamp option. Consists of a 4-byte header, followed by a data segment of arbitrary length. The header fields are as follow: Option Type and Option Length (adhering to the TLV encoding of most options), succeeded by a one-byte Pointer field denoting the offset into the option where a host may add a new timestamp, and two nibble-sized Overflow and Flag fields. The Overflow field is set initially to zero and is incremented by each host that is willing but unable to add a timestamp. The Flag field defines multiple modes of operation: the hosts are either required to append only 32-bit timestamps, or to add the IP of the corresponding interface as well. In case of the latter, multiple IPs can be pre-specified so that only select hosts may add their timestamps. Initially, we crafted packets such that they contained only the originating host's IP and timestamp, while no middleboxes would be allowed to append their own. The reason for this was to easily verify if the contents of the option could be modified in transit, given that the packets ever reached their destination. Later, we relaxed this constraint and allowed middleboxes to add their own timestamps, which did not give rise to any new impediments.

Although this option shares the functionality of Record Route, the 40-byte limit of the IP options section is too constricting for it to have the same practical utility. The inclusion of the Overflow field, however, can be leveraged to approximately determine the number of hosts knowledgeable of IP options along a certain route. This can be particularly useful when their number significantly exceeds the capacity of the Record Route option (over half of the working routes in our experiments). Nonetheless, there are still certain challenges when trying to accurately approximate the number of hosts. For example, a Tp-link router running a Linux 2.6 kernel may choose not to try and add a timestamp when forwarding an ICMP packet to another host. However, when the router itself is the

destination, it will readily attempt to append its timestamp. Previous work [15] shows that different router families can add between 0 and 4 timestamps on each interface. Another distinct behaviour can be gleaned in how Windows and Linux hosts respond to ICMP Echo Requests. A Linux host will append it's timestamp twice (both on ingress and on egress). This behaviour appears to be consistent across multiple kernels that we tested (up to 5.4.0). A Windows host however, will only add its timestamp once. This detail must be taken into account when not having access to both endpoints of the analyzed route and depending on the ICMP Echo Reply to carry over the data collected from the Echo Request, in addition to its own. In Figure 3 we illustrate the variability of recorded timestamps and reported overflow at the ICMP Echo Request destination. Contrary to our initial belief, correlating the overflow with the TTL difference is not trivial. It is also important to note that the overflow can not always be considered an underestimation of the TTL.

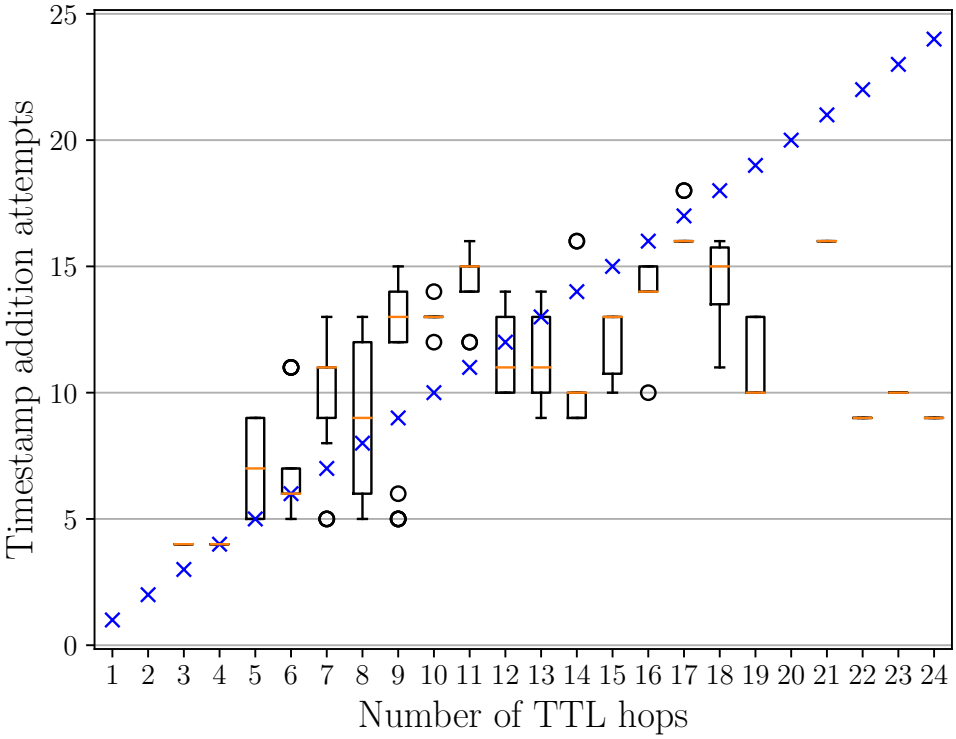


Figure 3: Variability of IP Timestamp records and overflow in comparison with TTL distance.

Table 2 illustrates the average route length based on the Overflow field, calculated over three separate trials, on different instances. The highlighted values signify that the annotated packets were not dropped during any test. Based on this data, we can draw a few observations. While DigitalOcean appears capable of both receiving and transmitting annotated packages, AWS performs consistently well only in the Mumbai and Manama regions. On the other hand, Google Cloud and Microsoft Azure always block the packets on ingress. On egress, we notice that Azure was able to emit annotated packets from its Seoul and Toronto-based regions once and twice during the three experiments, respectively. Finally, Google Cloud presents two distinctive traits. First, though wholly inconsistent, all regions were able to successfully send annotated packets, as confirmed from the vantage point of any instance verified to receive annotated traffic from other sources. This indicates that the issue may lie with middleboxes outside their own network. Second, both AWS and DigitalOcean seem to reset the contents of the Timestamp option (IP, timestamp pairs, as well as Overflow value) when entering their network, but only when the packet originated from a Google Cloud instance. This operation takes place regardless of the region of origin, or that of destination. A similar behaviour can be seen with the Route Record option, where the only registered IPs correspond to their Autonomous Systems and to a IANA-reserved shared address space. A possible explanation is that outgoing traffic

Table 2: Average number of attempted IP timestamp additions along the route between any two cloud instances or our personal (p) server. Highlighted values indicate option acceptance across all experiments. Missing values signify that no annotated packet was able to traverse the network.

Endpoint			destination																			
	Provider		gcloud					aws					azure					docean					p
		Region	us-east1	europe-west2	asia-southeast1	asia-northeast1	australia-southeast1	us-west-1	eu-central-1	ap-south-1	me-south-1	sa-east-1	canadacentral	centralus	koreacentral	francecentral	australiacentral	nyc1	fra1	sgp1	tor1	blr1	Bucharest
source	gcloud	us-east1	-							9	7							4	4	4			
		europe-west2		-						9	7							4	4	4			
		asia-southeast1			-					9	7							4	4	4			
		asia-northeast1				-				9	7							4	4	4			
		australia-southeast1					-			9	7							4	4	4			
	aws	us-east-1						-															
		eu-central-1							-														
		ap-south-1								-	13							16	15	12	14	12	12
		me-south-1								13	-							13	10	15	13	11	11
		sa-east-1										-											
	azure	canadacentral								16	13		-					9	10	10	8	11	10
		centralus												-									
		koreacentral								15	13				-			9	9	10	8	10	10
		francecentral														-							
		australiacentral															-						
	docean	nyc1								17	15							-	6	6	5	11	10
		fra1								14	13							6	-	6	5	10	9
		sgp1								13	14							6	6	-	5	11	13
		tor1								14	13							5	5	5	-	9	11
		blr1								14	12							11	10	11	9	-	
	p	Bucharest								16	13							10	11	14	10	13	-

is encapsulated upon leaving the Google AS for well-known destinations. Unfortunately, we are unable to verify this theory due to access limitations.

Record Route option. This option was designed in order to facilitate route recording by affixing the output interface IP of each router to a pre-allocated space in the options area. Being comprised of a 3-byte header and a variable-length data section, its functionality is similar to that of the Timestamp option. The Option Type, Option Length and Pointer field all have the same significance, with one remark: on detecting an incongruity in the remaining available space as calculated based on the Length and Pointer fields, the route is obligated to drop the packet outright. In the absence of timestamps, the maximum number of IPs that can be recorded is nine. This option may constitute a viable alternative to existing network path discovery tools (e.g. `traceroute`) that may be expressly thwarted for instance, by disabled ICMP Time Exceeded replies. Although this method proved functional, the longest path a packet followed in our experiments was 24 hops, considerably exceeding the available capacity for recorded IPs.

Although the limited space availability is a natural concern when employing this mechanism in the open Internet, recent work [11] has shown that approximately half of all advertised BGP prefixes at the time were within the recording distance limit from their originating hosts. During our experiments, we determined that any two instances fell within the same limit 54.88% of the time, predicated on the fact that the packets were able to successfully traverse the network (see Table 1). Out of a total of 669 addresses recorded along 82 viable routes between the 21 instances, their distribution was as follows. *IANA-reserved private addresses* [17]: 120, with 82 belonging to A-block (10/8 prefix), 11 belonging to B-block (172.16/12 prefix) and 27 belonging to C-block (192.168/16 prefix). *IANA-reserved shared addresses* [20]: 88, used to connect end devices to Carrier Grade NAT equipment on Service Provider networks (100.64/10 prefix). *Amazon Technologies Inc.*: 132. *Digital Ocean Inc.*: 170. *Microsoft Corporation*: 18. *RoEduNet*: 38. The remaining 103 recorded addresses belong to organizations such as GÉANT, the European research and education network, and various other carriers (e.g.: Telia,

Tata Communications, etc.) As previously mentioned, the absence of *Google Inc.* from this account is motivated by an anomalous behaviour where the route record prior to entering AWS or DigitalOcean networks is erased (or rather missing) without the packet being dropped, as long as it originated in the Google Cloud network.

Unassigned and Experimental options. Similarly to the Timestamp options, these as well are generally accepted in every tested DigitalOcean region, as well as Mumbai and Manama for AWS. Google Cloud remains an unreliable sender, its annotated traffic sometimes reaching these hosts but more often not. While in previous experiments Google's firewall simply dropped incoming IP annotated traffic, we noticed a small number instances on the west European server when options would be removed and the packet allowed to pass. Unfortunately, we encountered only three such cases (one for FTP and two for HTTP) and cannot draw any definitive conclusion. Neither of these anomalies were accounted for in the statistics presented in Table 1. As for Microsoft Azure, the Korean server was able to consistently send Experimental IP options, but not Unassigned options. Aside from this singular exception, all annotated traffic was dropped at some point. Finally, our personal server was capable of sending but not receiving packets with Unassigned or Experimental IP options. After running tests in our local network, we concluded that the Palo Alto firewall was responsible, and not the vSphere hypervisor.

2.2 Analysis of TCP options

Timestamp option. The acceptance of TCP options increased remarkably during the past decade. Admittedly, our study was not intended to verify the correctness of session specific options. Neither is the tool that we developed capable of tracking connection states and take appropriate action in the option content generation stage. Our primary goal was simply to establish whether annotated packets can traverse the network. As a result, experiments involving this option would not extend past the initial SYN, SYN-ACK exchange. Normally, a misconfiguration of the Echo Reply value would cause the initial sender to terminate the connection with a RST after receiving the incorrect SYN-ACK. We confirm that replying with an incorrect `TSecr` or using a non-monotonically increasing function for `TSval` would lead to immediate termination. This, however, would not impede annotated RST segments from traversing the network.

Echo and Echo-Reply options. According to the IANA registry, ten TCP option codepoints have been obsoleted to this day. We attempted to determine if obsoleted options can be blocked by middleboxes. And if not, whether the option kinds can be reused with other implementations. As such, we analysed the two aforementioned options, as precursors of the Timestamp option. Where the initiator would send an Echo option with a 4-byte payload (`TSval`) that would usually represent a timestamp, the receiver would reply with an Echo-Reply option containing the former's value (`TSecr`). We concluded that both options still function normally and that replacing their contents with that generated for unassigned options does not cause the packets to be dropped. Moreover, issuing Echo-Replies without having received a prior Echo-annotated packet does not impede the former's network traversal in any way.

Reserved and Experimental options. At the moment there are numerous reserved codepoints that are known to be employed without proper IANA authorization. As such, we purposely chose the `0x47` kind in order to avoid matching potential firewall `ACCEPT` rules for particular implementations in certain networks, unlikely as it may be. According to our experiments, both the Reserved and Experimental options always reached their destination unmodified.

2.3 Analysis of UDP options

Checksum Compensation Option. UDP options are a relatively new addition [19]. Although initially lacking support for any type of extension, a redundancy between the `IP.total_length` and the `UDP.length` fields allowed the insertion of an extra section after the UDP payload. Due to this unconventional development, their acceptance by middleboxes is still largely unknown. One of the main causes for their rejection [22] was addressed with the introduction of the Checksum Compensation Option (CCO) [6]. This option accounts for network equipment that equates the IP payload length

Table 3: Percentage of successful UDP option network traversals between any two cloud instances or our personal (p) server.

Endpoint		destination																			
	Provider	gcloud					aws					azure					docean					p
	Region	us-east1	europe-west2	asia-southeast1	asia-northeast1	australia-southeast1	us-west-1	eu-central-1	ap-south-1	me-south-1	sa-east-1	canadacentral	centralus	koreacentral	francecentral	australiacentral	nyc1	fra1	sgp1	tor1	blr1	Bucharest
source	gcloud	us-east1	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		europe-west2	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		asia-southeast1	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		asia-northeast1	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		australia-southeast1	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	aws	us-east-1	0	0	0	0	0	-	100	100	100	100	100	100	100	100	100	100	100	97	100	100
		eu-central-1	0	0	0	0	0	100	-	100	100	100	100	100	100	100	100	100	100	97	100	100
		ap-south-1	0	0	0	0	0	100	100	-	100	100	100	100	100	100	100	100	100	97	100	100
		me-south-1	0	0	0	0	0	100	100	100	-	100	100	100	100	100	100	100	100	97	100	100
		sa-east-1	0	0	0	0	0	100	100	100	100	-	100	100	100	100	100	100	100	97	97	95
	azure	canadacentral	0	0	0	0	0	100	100	100	100	87	-	100	100	100	100	100	100	97	100	100
		centralus	0	0	0	0	0	100	100	100	100	87	100	-	100	100	100	100	100	97	100	100
		koreacentral	0	0	0	0	0	100	100	100	100	87	100	100	-	100	100	100	100	97	100	100
		francecentral	0	0	0	0	0	100	100	100	100	43	100	100	100	-	100	100	100	97	100	100
		australiacentral	0	0	0	0	0	100	100	100	100	87	100	100	100	100	-	100	100	100	97	100	100
	docean	nyc1	0	0	0	0	0	100	100	100	100	0	100	100	100	100	100	-	100	100	97	100	97
		fra1	0	0	0	0	0	100	100	100	100	87	100	100	100	100	100	100	-	100	97	100	100
		sgp1	0	0	0	0	0	100	100	100	100	87	100	100	100	100	100	100	100	-	97	100	100
		tor1	0	0	0	0	0	97	100	100	97	87	97	100	97	100	97	100	95	97	-	100	97
		blr1	0	0	0	0	0	100	100	100	100	0	100	100	100	100	100	100	100	100	97	-	95
	p	Bucharest	0	0	0	0	0	100	100	100	100	87	100	100	100	100	100	100	100	100	97	100	-

to the UDP length, thus calculating the checksum in a manner that will lead to the packet being discarded. For each UDP option that we tested, we performed an additional experiment involving the CCO. Following these alternative tests, we did not observe any significant improvement in the option accepting rating, as can be seen in Table 1. These leads us to believe that the previously identified pathologies do not necessarily apply in our tested networks.

Timestamp option. This option was modeled after its TCP equivalent. Although bearing many similarities such as the incorporation of both TSval and TSecr fields in the same option, or the use of a monotonic non-decreasing function for TSval, in contrast to the TCP Timestamp, TSecr is allowed to be returned with null value. Because this action represents the express intent of the application, it would not lead to the early interruption of it session. Consequently, since its application in this form does not require session tracking, we were able to annotate all segments with this option.

Unassigned and Experimental options. As a pendant to the TCP Experimental option, it also shares the same codepoint and even the Experimental ID system. For this reason, we decided to reutilize the TCP implementations here. Eventually, we concluded that there was no statistical difference between the acceptance of an Unassigned or Experimental option and already defined Timestamp option.

A more detailed representation of route-specific acceptance rates can be found in Table 3. As it can be readily observed, Google is consistent in blocking UDP options both on ingress and egress, possibly based on the inconsistency between `IP.total_length` and `UDP.length`. Conversely, with few exceptions, all other providers have exceedingly high acceptance rates. We note that we did not encounter any cases in which UDP options were stripped or modified while also allowing the base packet to pass.

3 Related Work

At this time, there are numerous studies dating back to the 1990s that evaluate the state of IP and TCP options at distinct moments in time. On the other hand, due to their recent introduction, UDP options have not yet garnered the same credibility as their aforementioned counterparts. Additionally,

in spite of the large body of work that substantiates (or repudiates) their applicability in modern contexts, TCP and IP options have been evaluated for different purposes. While TCP options needed to be extendable and incorporate new implementations, IP options have been tested mostly as probing mechanisms. Our work differs from the rest in two aspects. Firstly, we provide a more extensive analysis, incorporating multiple protocols as to determine the situations where each can be most suitably employed. Secondly, we evaluate not only the acceptance of existing implementations, but also that of potential extensions for each protocol.

Fonseca *et al.* [8] offered one of the first relevant investigations into the suitability of IP options at a time when extensions that were considered for wide-spread adoption vied for the allocation of dedicated fields in the IP header. By using PlanetLab [2], a global service deployment test network that operated between 2002 and 2020, the authors concluded that while over half of the tested routes proved adverse to IP options, the vast majority (approx. 90%) of packets were dropped in edge ASes. Additionally, they were able to identify a small subset of ASes responsible for compromising an exceedingly large number of routes.

Honda *et al.* [12] proposed a middlebox evaluation methodology based on TCPExposure, a measurement tool of their own design that was employed by multiple contributors, totaling 142 venues across 24 countries. Their goal was to decide whether Multipath TCP [21] and TcpCrypt [1] were ready for deployment in the wider Internet. Their investigation yielded a number of insights into the behaviour of middleboxes that implement transport layer functionality.

In their 2012 work, De Donato, Marchetta and Pescapé [4] deliberated on the dependability of IP timestamp options for active network measurements. By probing over 1.7M public IPs with multiple transport layer protocols while carrying the aforementioned option, they identified increased utility for ICMP and exceedingly low tolerance for TCP when discarding responses that were not RFC-compliant. In total, they identified over 40k target hosts that provided such replies and six types of anomalies that are representative of erroneous implementations. The most prominent anomalies they encountered were pre-specified timestamp IP address overwrite and pointer field inconsistency.

Kühlewind *et al.* [13] published a study of Explicit Congestion Notification (ECN) and other congestion-related TCP options (i.e.: SACK, Timestamp, Window Scaling). In spite of the fact that ECN was over a decade old at the time and implemented in virtually every operating system, it can now be considered a timely and ideal example of protocol ossification. Proof of this is the fact that most operating systems that did implement ECN had it disabled by default in order to circumvent impermissive firewall implementations. After performing two sets of measurements four months apart, the authors concluded that although 90% of servers negotiated ECN usage, ECN feedback could become a more contentious issue because of the utilization of reserved bits in the TCP header.

`Tracebox` [5] is an extension to `traceroute` that is capable of identifying the exact middlebox that either alters or outright drops packets bearing protocol extensions. Sequentially incrementing the TTL field and keeping note of each emitted packet, `tracebox` is able to detect the changes brought by each middlebox on the path via a direct comparison to the quoted packet that is encapsulated in the ICMP Time Exceeded Message. Nonetheless, this approach carries certain risks, including the possibility that certain middleboxes block ICMP error messages, or that the aforementioned quoted packet is limited to the IP header and the first 64 bits of the payload (when middleboxes do not implement support for multi-part ICMP messages). While lacking the diagnostic capabilities of `tracebox`, our tool is focused on data collection for real-life applications with prolonged sessions and not just connection establishment.

`PATHspider` [14] is a differential protocol testing tool based on customizable A/B testing. Similarly to our tool, `PATHspider` deals in problems of transport layer protocol ossification. It determines whether protocol extensions are correctly implemented by creating two connections: one with a default socket configuration to serve as baseline, and one implementing the desired extension (not limited to protocol options). The two main differences between this tool and ours are as follows: first, while `PATHspider` crafts synthetic traffic, we modify packets that are organically generated by existing tools. This reduces the effort necessary to set up new experiments. The second difference consists in how we evaluate the success of the experiment. `PATHspider` monitors the network traffic and performs real-time user-defined checks to asses the correctness of the protocol implementation. In contrast, we

prefer to create packet captures (preferably at both endpoints) and carry out offline tests via bash scripts. The motivation for this design choice stems form a desire to modularize our system and is further justified by the need to perform new tests without re-running the experiments.

In 2017, Marchetta *et al.* [15] presented several new network measurement techniques based on variations of the Record Route and Timestamp IP options. First, they propose an alternative `traceroute` technique based not Time Exceeded messages, but on ICMP Parameter Problem, caused by malformed IP options. Additionally, they present middlebox and path length detection techniques, as well as a new method for router alias resolution and fingerprinting.

Goodchild *et al.* [11] studied the effects of the "flattening Internet" [9] by testing the IP Record Route option against all advertised BGP prefixes. They discovered that two thirds of all responders to ICMP Echo Requests carrying this option were within nine hops from at least one PlanetLab or Measurement-Lab vantage point. Moreover, the responders comprised a significant majority (75%) of all tested hosts.

In a recent study, Zullo, Jones and Fairhurst [22] identified a series of path pathologies, resulted mainly from protocol ossification, that negatively impact the acceptance of UDP options. They discovered that most packet rejections are due to incorrect UDP checksum calculations that stem from the usage of the IP Total Length field, in favor of the UDP Data Offset field. Due to it's poor dependability in certain networks, in lieu of utilizing the zero-checksum that indicates that no checksum verification is required for UDP, they developed a new Checksum Correction option that is supposed to overflow the incorrect 16-bit sum in such a way so that the final checksum is equivalent to the correct checksum.

4 Conclusion

This paper explores the acceptance of well-established IP, TCP and UDP options, as well as the possibility to develop new options. To this end, we created a packet annotation tool of approx. 2000 LoC that is meant to append user-specified options of any type to packets matching specific `iptables` rules. Additionally, we constructed a script-based framework to manage a cloud-based infrastructure including four of the largest providers (i.e.: Google Cloud, AWS, Microsoft Azure, DigitalOcean) along with our personal cloud. Both tools can be easily extended by adding new protocols, implementing new options, and selecting new regions for future experiments.

Our evaluation consists of reachability tests for 19 options or combination of options corresponding to the three previously enumerated protocols. For each option, we performed a series of experiments involving higher layer protocols (e.g.: http, ntp, dns, etc.) as well as raw data passed to arbitrary ephemeral ports. One of our goals was to provide a consistent assessment for protocol extension mechanism that have historically been evaluated independently, in separate environments and with different purposes. In addition to route-specific acceptance statistics, we offer a number of observations made during our measurements, among which:

- While extraneous sequences of options (e.g.: 4x NOP) were never considered malicious, certain Palo Alto firewalls may explicitly drop Experimental/Unassigned IP options but not their TCP and UDP counterparts.
- IP options seem to function properly only in conjunction with ICMP, at least in our experiments.
- The overflow of the IP Timestamp option is not necessarily a good indicator of path length. AWS and DigitalOcean reset the contents of the option on entering their network, but only for packets originating in Google Cloud.
- Network traversal reliability for UDP options is almost as good as for TCP. The only provider that actively blocks all UDP options is Google Cloud. Because the Checksum Compensation Option was not effective in this scenario, we surmise that checksum calculation errors are not responsible.

We conclude that the current state of TCP and UDP options is favorable enough for developing new extensions. Any acceptance problems involving the latter are isolated to specific cloud providers, and not ISPs. IP options remain a contentious issue. Because their dependability when used in conjunction with protocols other than ICMP is fairly low, they are not well suited as protocol enhancements in the same way TCP options are to TCP. However, we believe that there is still potential for developing new network measurement techniques based on the existing IP options.

References

[1] Andrea Bittau, Michael Hamburg, Mark Handley, David Mazieres, and Dan Boneh. The case for ubiquitous transport-level encryption. USENIX Association, 2010.

[2] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[3] Rongrong DAI, Honghui LI, and Xueliang FU. Elephant flow scheduling in sdn data center network based on differential evolution algorithm. In *UPB Scientific Bulletin Series C*, 2022.

[4] Walter De Donato, Pietro Marchetta, and Antonio Pescapé. A hands-on look at active probing using the ip prespecified timestamp option. In *International Conference on Passive and Active Network Measurement*, pages 189–199. Springer, 2012.

[5] Gregory Detal, Benjamin Hesmans, Olivier Bonaventure, Yves Vanaubel, and Benoit Donnet. Revealing middlebox interference with tracebox. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 1–8, 2013.

[6] Godred Fairhurst, Tom Jones, and Raffaele Zullo. Checksum compensation options for udp options. Internet-Draft draft-fairhurst-udp-options-cco-00, IETF Secretariat, 2018. https://www.ietf.org/archive/id/draft-fairhurst-udp-options-cco-00.txt.

[7] Bill Fenner. Experimental values in ipv4, ipv6, icmpv4, icmpv6, udp, and tcp headers. Technical report, RFC 4727, November, 2006.

[8] Rodrigo Fonseca, George Porter, R Katz, Scott Shenker, and Ion Stoica. Ip options are not an option. Technical report, Technical report, EECS Department, University of California, Berkeley, 2005.

[9] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. The flattening internet topology: Natural evolution, unsightly barnacles or contrived collapse? In *International Conference on Passive and Active Network Measurement*, pages 1–10. Springer, 2008.

[10] F. Gont, R. Atkinson, and C. Pignataro. Recommendations on filtering of ipv4 packets containing ipv4 options. BCP 186, RFC Editor, 2014.

[11] Brian J Goodchild, Yi-Ching Chiu, Rob Hansen, Haonan Lua, Matt Calder, Matthew Luckie, Wyatt Lloyd, David Choffnes, and Ethan Katz-Bassett. The record route option is an option! In *Proceedings of the 2017 Internet Measurement Conference*, pages 311–317, 2017.

[12] Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, and Hideyuki Tokuda. Is it still possible to extend tcp? In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 181–194, 2011.

[13] Mirja Kühlewind, Sebastian Neuner, and Brian Trammell. On the state of ecn and tcp options on the internet. In *International Conference on Passive and Active Network Measurement*, pages 135–144. Springer, 2013.

[14] Iain R Learmonth, Brian Trammell, Mirja Kuhlewind, and Gorry Fairhurst. Pathspider: A tool for active measurement of path transparency. In *Proceedings of the 2016 Applied Networking Research Workshop*, pages 62–64, 2016.

[15] Pietro Marchetta, Valerio Persico, Giuseppe Aceto, Alessio Botta, and Antonio Pescape. Measuring networks using ip options. *IEEE Network*, 31(3):30–36, 2017.

[16] Jon Postel. Internet protocol. STD 5, RFC Editor, 1981. http://www.rfc-editor.org/rfc/rfc791.txt.

[17] Yakov Rekhter, B Moskowitz, Daniel Karrenberg, GJ de Groot, and Eliot Lear. Rfc1918: Address allocation for private internets, 1996.

[18] Joseph Touch. Shared use of experimental tcp options. Technical report, RFC 6994, August, 2013.

[19] Joseph Touch. Transport options for udp. Internet-Draft draft-ietf-tsvwg-udp-options-07, IETF Secretariat, 2019. http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-udp-options-07.txt.

[20] Jason Weil, Victor Kuarsingh, Chris Donley, Christopher Liljenstolpe, and Marla Azinger. IANA-Reserved IPv4 Prefix for Shared Address Space. RFC 6598, April 2012.

[21] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI*, volume 11, pages 8–8, 2011.

[22] Raffaele Zullo, Tom Jones, and Gorry Fairhurst. Overcoming the sorrows of the young udp options. In *2020 Network Traffic Measurement and Analysis Conference (TMA), IEEE*, 2020.

This journal is a member of, and subscribes to the principles of,
the Committee on Publication Ethics (COPE).
https://publicationethics.org/members/international-journal-computers-communications-and-control

Cite this paper as:

Mantu, R.; Chiroiu, M.; Țăpuș, N. (2024). Framework for evaluating TCP/IP extensions in communication protocols, *International Journal of Computers Communications & Control*, 19(2), 4906, 2024.
https://doi.org/10.15837/ijccc.2024.2.4906