mitiKV: An Inline Mitigator for DDoS Flooding Attacks

Yuta Tokusashi*, Yohei Kuga*, Ryo Nakamura†, Hajime Tazaki‡, Hiroki Matsutani*
{tokusasi, matutani}@arc.ics.keio.ac.jp, {sora, upa}@haeena.net, tazaki@wide.ad.jp
*Keio University, †University of Tokyo, ‡ IIJ Innovation Institute

Abstract

We propose mitiKV, a hardware-based Distributed Denial of Service (DDoS) mitigator which can clean up malicious traffic at transit links. DDoS flooding attack has been an issue for Autonomous Systems to provide stable network service to legitimate users. Many research on DDoS detection focuses on the statistical analysis on network traffic in order to employ filtering rules at (software) middle-boxes. In our methodology, DDoS attacks are detected by filtering suspected DDoS packets, which are associated with ICMP packets replied by victim hosts as an error. We adopt hardware-based key value store on an FPGA to manage a rule to determine DDoS packets. We developed a prototype system on an FPGA board, called mitiKV, and demonstrated to prevent DNS-based amplification attacks as a proof of concept. Our evaluation shows that mitiKV has a capability of DDoS mitigation in 10Gbps line rate. This methodology can scale up to 100Gbps link speed and can be also adopted into other DDoS attacks.

Keywords: DDoS Mitigation, Middlebox, Packet Filtering, Network hardware, FPGA

1. Introduction

Distributed Denial of Service (DDoS) attack is a significant problem of the Internet. Major DDoS attacks exhaust network or server resources of victims. Amplification attacks using UDP-based protocols such as DNS and NTP try to exhaust link capacity on victim networks. TCP SYN flooding attacks lead to victim servers consuming CPU resources. With the growth of importance of services on the Internet, DDoS attack cannot be ignored from a viewpoint of service availability. Hence, mitigating DDoS attack has become increasingly important.

The volume of DDoS attacks to exhaust network resources is increasing due to the growth of link speed and server performance. Arbor Networks reports that the largest traffic volume of DDoS attack was 100 Gbps in 2010 and increased to 500Gbps in 2015 [1]. In addition, 66% of DDoS attack's targets are customers, such as end users, financial and hosting services. The customer networks purchase transit connectivities that are 10, 40 and up to 100Gbps links from transit network providers in general. Therefore, today's DDoS attacks exhaust transit links effortlessly; and thus DDoS attacks should be mitigated in transit provider sides to protect customer networks.

On the other hand, a current dominant type of DDoS attack is DNS-based amplification attack. The DNS protocol has a high traffic amplification rate and there are over 12 millions of open recursive resolvers around the world that can be used for amplification attack [2][1]. Furthermore, Arbor Networks reports that DNS occupies 85% of protocols used for amplification attacks [1]. Thus, if DNS-based amplification attack is completely prevented, most of DDoS attacks can be eliminated.

In this study, we propose a novel hardware-based DDoS mitigation system, called mitiKV, that focuses on the DNS amplification attack. The mitiKV works as an inline middlebox on a transit link between a transit provider and a customer network. mitiKV detects and discards DDoS packets toward customer network without fail by using ICMP port unreachable message. In addition, detecting and filtering the attacks are all processed in hardware-based key value store on Field Programmable Gate Array (FPGA), so that mitiKV achieves high throughput up to 100Gbps. We developed a prototype system on an FPGA board and demonstrated to prevent DNS-based amplification attacks as a proof of concept. Our evaluation shows that mitiKV has a capability of mitigation in 10Gbps line rate. This methodol-

[1] as of at August 21, 2016.

ogy can scale up to 100Gbps link speed and can be also adopted into the other types of UDP-based amplication attacks.

2. Related work

DDoS detection mechanisms can be classified into two categories: anomaly detection and pattern matching [3]. Anomaly detection based methods collect normal system or network behavior regularly and compare present state with the normal state to detect anomalies. PacketScore [4] based on anomaly detection provides a score, called Conditional Legitimate Probability, that can be used to decide a packet is malicious or not. An advantage of pattern matching over anomaly detection is that several known attacks are detected without fail. Snort [5], a popular open source intrusion detection system using pattern matching, has wide usage. The proposed detection is also one of pattern matching method focusing on DNS-based amplification attack.

In addition to DDoS detection mechanisms, data plane system for packet forwarding and filtering is also a fundamental part of DDoS mitigators. In such data plane systems, high throughput is a technical issue to overcome the growth of attack traffic volume. CYSEP [6] is a hardware architecture for firewall, encryption/decryption, message authentication and DDoS mitigation. The DDoS detection system of CYSEP is PacketScore[4], and it is designed to be implemented in ASIC. mitiKV and CYSEP mitigation module prevent link congestion attacks from exhausting on high-speed networks by hardware implementation. On the other hand, other hardware based mitigation approaches, Sentinel [7] and SQL DDoS Mitigator [8], mitigate end host CPU resource exhaustion attacks by generating and sending CAPTCHA [9] on hardware. In contrast to mitiKV, they are installed in front of victim servers and protect server resources.

As opposed to hardware-based technique, software-based packet inspection implementations such as Snort [5] have a plenty of flexibility in order to inspect, and filter packets from the network. Although software-based technique historically has performance drawbacks, many proposals were addressed to alleviate the drawbacks to take both advantages of high-performance and flexible packet processing simultaneously. GASPP [10] is a network traffic processing framework which integrates Graphic Processing Units (GPU) for their packet processing purposes. While taking advantage of flexible packet processing such as filtering with a regular expression specified by users, it optimizes memory usage as well as packet scheduling

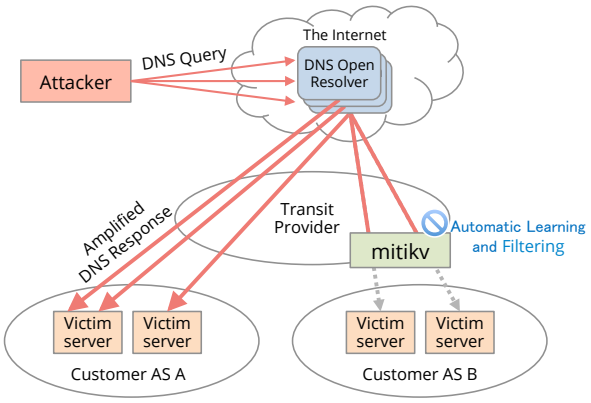

Figure 1: The overview of mitiKV installation on a provider network. mitiKV box is installed on a provider side of a customer link and works as an inline middlebox.

for packet processing in order to speedup its processing. It also integrates Snort to accelerate the performance to mitigate traffic from attacks. Our proposed method does not require such a flexible method to packet processing as it only needs to look at a specific portion of a packet. The simplified method gives us an opportunity to implement the mitigation middlebox as a hardware.

3. mitiKV Architecture

The mitiKV is a hardware-based, autonomous, inline DDoS mitigation system. Figure 1 shows the overview of mitiKV on a provider network. DDoS defense mechanisms can be characterized with three features: activity level, cooperation degree and deployment location [3]. We will discuss mitiKV in terms of these features below.

- **Activity level:** mitiKV is reactive and pattern matching based detection mechanism. mitiKV is specialized in detecting DNS amplification attack, therefore, mitiKV achieves high speed detection and blocking compared to anomaly detection.
- **Cooperation degree:** mitiKV does not need any other traffic measurement or filtering systems. mitiKV performs DDoS attack detection and filtering on its hardware. In addition, multiple mitiKV boxes installed on multiple customer links work autonomously.
- **Deployment location:** mitiKV is located on a transit link which is connected to other networks. It protects a link connected to a customer network against DDoS attack.

In this manner, mitiKV works as an inline DDoS mitigator on customer links to protect customer servers against DNS-based amplification attack.

3.1. DDoS detection by ICMP behavior

The key idea of the DNS-based amplification attack detection mechanism of mitiKV is leveraging one of the most basic mechanisms of the Internet, **ICMP Port Unreachable** message.

ICMP port unreachable message contains the IP header and the first 64 Bytes of the original datagram's data [11], therefore, DNS-based amplification attack can be identified by checking ICMP port unreachable message on customer links.

The detection sequence is shown in Figure 2 and is explained below. Figure 3 shows state machine to manage 4-tuple state discussed below.

1. DNS Query : An attacker sends a DNS query to DNS open resolver. the attacker spoofs its source IP address as victim's host IP address.
2. DNS Response : DNS open resolver receives a query and replies to the spoofed source IP address to victim host via transit link. mitiKV learns 4-tuple that consists of source IP address, destination IP address, source UDP port number and destination UDP port number. mitiKV updates the state into "SUSPECTIVE".
3. ICMP Port Unreachable Meassage : Victim host receives an unexpected DNS response packet and replies ICMP port unreachable message including the DNS response packet on payload. mitiKV updates the state of 4-tuple into "BLOCKED" and decides this as DDoS attack.
4. DNS Query : Repeating steps 1 and 2.
5. DNS Response : DNS open resolver replies against a query, but mitiKV already learned the 4-tuple and recognizes that this packet is related to DDoS attack. Thus, mitiKV discards this packet.

By this method, mitiKV achieves fast and undoubted DNS-based amplification attack detection.

3.2. Hardware design

Figure 4 shows our mitiKV architecture overveiw. mitiKV consists of the following modules.

- **DDoS Attack Filter module** : This module monitors all packets to filter the packets related to suspective DDoS attacks. When DDoS attack packets are filtered, the module creates a tuple which will be processed in DB Lookup/insert module. We assume DNS-based amplification attack as attack traffic in Section 3.3. In this case, DNS response packets whose source UDP port number is 53 are filtered.

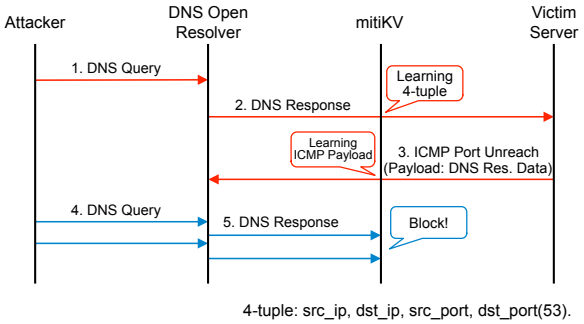

Figure 2: The detection sequence of mitiKV.

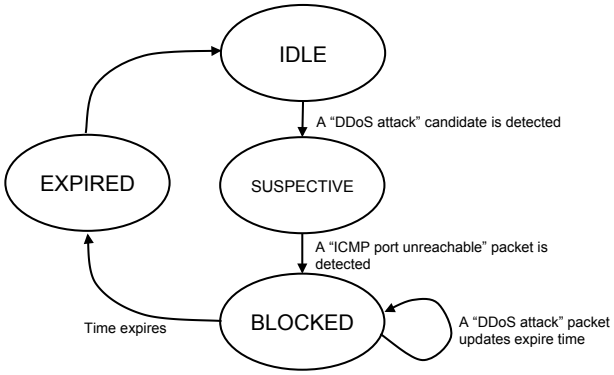

Figure 3: State machine of mitiKV that manages 4-tuple.

- **ICMP Filter module** : This module monitors all packets whether packets with ICMP port unreachable and an error packet over the ICMP matched with DDoS suspective packet as mentioned in Section 3.1. When packets are filtered, this module sends a request query to database to lookup state and then checks that the packet is suspective DDoS attack.

- **DB Lookup/Insert module** : This module provides two interfaces, READ and WRITE interfaces to manage flows and their states. DDoS Attack Filter module and ICMP Filter module access this module to check the flow whether DDoS attack or to update the state. This DB is managed by hardware-baed key value store on the memory module.

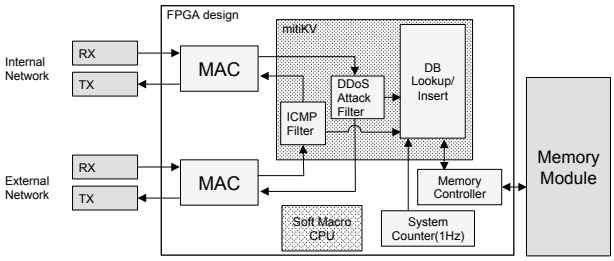

Figure 4: mitiKV architecture.

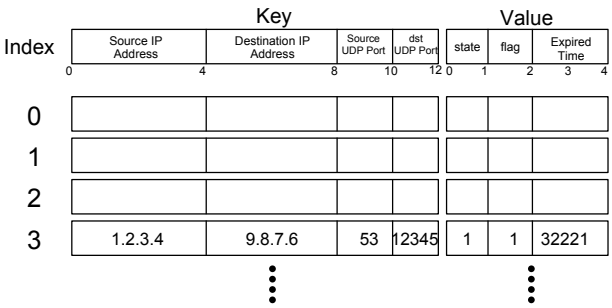

Figure 5: Hash Table design on mitiKV.

- **System Counter** : This module is a counter module generating timestamp for expiring logic. The timestamp in value field is checked if time is expired when DB module accesses each key value entry.

An external memory is used as hash table as shown in Figure 4. Hash table can be implemented on DRAM, SRAM and internal FPGA memory. Each of these memory modules has a constant access latency. The difference of these access latency is hidden by implementing a pipeline deeply in which the number of stage is equivalent to the access latency. On the other hand, the memory size will be a crucial factor to mitigate attack traffic. To mitigate abusing traffic, hash table needs to be larger than the size which is capable of storing the number of abusing flows in a period. The hash table size and hit ratio on hash table is the key component on this approach. We will evaluate them in Section 4.1.

Figure 5 shows hash table design on mitiKV. Each entry consists of key and value. Key is 12B fixed length and consists of IPv4 source address, IPv4 destination address, source UDP port number and destination UDP port number. Value is 4B fixed length and consists of status, flag and expiring time. Index is calculated by hash fucntion in order to retrieve the key. The key retrieved from the hash table is compared with the requested key to distinguish both the requested key and the key from hash table are identical.

3.3. FPGA implementation

In this section, a prototype implementation of the proposed mitiKV is illustrated. Our mitiKV is implemented on Digilent NetFPGA-SUME board [12, 13]. An FPGA device used is Xilinx Virtex-7 XC7V690T FFG1761-3. The board has four 10GbE interfaces for communication. Design tool used is Xilinx Vivado 2015.4.

Our mitiKV module uses Advanced eXtensible Interface (AXI) Stream [2] as data bus interface. We use 10G Ethernet Subsystem IP[3] as a 10G MAC. The data width is 64bit. Our implementation is based on AXI Stream data bus interface.

Integrating CPU into our implementation can provide configurations, statistaical information and manual managing of rules on key value store via an user interface on UART. We did not implement a soft-macro CPU for system configuratin for the sake of simplicity.

We implemented the key value store on BlockRAMs[4]. The number of hash table entries is 1k and 262k to store filtering rules. If more capacity of storing rules is required, external memory such as SRAM or DRAM is also avaliable to store them. As we mentioned, a constant access latency of SRAM and DRAM is hidden by implementing a deep pipiline.

Specifically, 10GbE requires running at more than 156.25MHz when AXI stream data width is 64bit. Table 1 shows the synthesis report of mitiKV per table size. Hash table sized we implemented are 1k and 262k sizes. In 262k size, the maximum frequency is 162.94MHz. The result of the synthesis indicates that mitiKV is enough to run on 10GbE. We implemented 262k size hash table, which is the maximum size of single memory controller. The BlockRAM utilization is 64.73%. In case more hash table space is needed, an external RAM such as SRAM and DRAM can be replaced instead of BlockRAMs. Since the slice utilization indicates that mitiKV core is so small compared to the overall FPGA's area, the mitiKV core can be also deployed in other hardware-based network appliances.

4. Evaluation

4.1. Hash table size managed on mitiKV

Hash table size is the critical factor in terms of mitigating DDoS attack. DDoS Attacker may spoof multiple source IP addresses. In the case, the number of

[2] AXI is a family of microcontroller buses by ARM AMBA.
[3] Inteligent Property provided by Xilinx
[4] Xilinx FPGA contains serveral kilobits memory as internal RAM.

Table 1: Synthesis results.

Hash table entry size	1k	262k
Maximum Frequecy [MHz]	163.11	162.94
Slice Utilization [%]	4.17	6.21
BlockRAM Utilization [%]	1.63	64.73

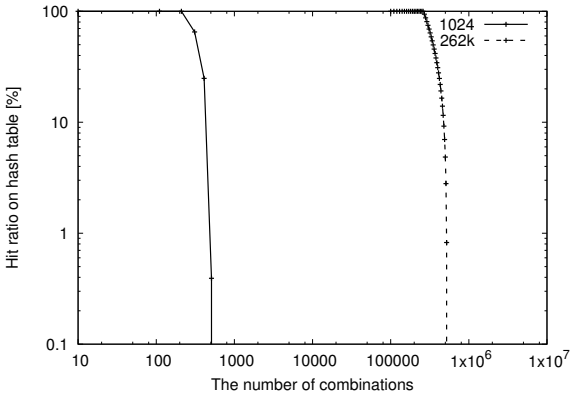

Figure 6: Simulation result on CRC32.

combinations of source IP address and source UDP port number that request packet includes is important to mitigate DDoS attack. Thus, hash table size whether how many keys to be stored and key conflicts should be handled.

The hash table size is limited by a hardware physical problem in which an FPGA has limitations of an external memory module [14]. Here, we evaluate hash table size and the number of mitigated packets. We built a simulator of mitiKV behavior for the evaluation.

Here, we use a simple incremental pattern as data pattern, where combinations of destination UDP port and source IP address is incremented for each access. Hash functions used are CRC32 and lookup3 of jenkins hash. CRC32 is often used in a hardware implementation hash such as calculating algorithm of frame check sequence on Ethernet MAC. Jenkins hash is often used in hardware design of key value store [15].

Figures 6 and 7 show simulation results when hash table size is 1k and 262k entries using CRC32 and jenkins hash to calculate hash table index, respectively. X-axis denotes the number of combinations of amplifiers's source IP address and destination UDP port number. Y-axis denotes the hit ratio of packets which are mitigated by mitiKV. In 1k table size, mitiKV cannot mitigate these packets related to DDoS attacks when the number of combinations is over 600 on CRC32. In 262k hash table size, mitiKV cannot mitigate packets when the number of combinations is over 14000. In jenkins hash, we get higher hit ratio than the ratio of CRC32. When the number of combinations is 1000, the hit ratio is 38%. This implies that hash table is used efficiently when jenkins hash is adopted, compared with CRC32. Thus, we have to choose hash table size carefully taking expected traffic amounts and network interface of transit link into consideration. We will continue to investigate the hashing system including hash function to obtain the higher hit ratio on mitiKV.

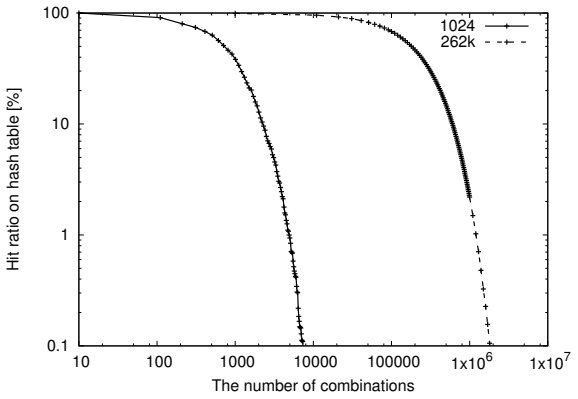

Figure 7: Simulation result on jenkins hash.

4.2. Hardware mitigation test

Here, we evaluated FPGA implementation as simulated in Section 4.1.

4.2.1. Hardware environment

Figure 8 shows an evaluation environment. Each component is explained as follows, except our mitiKV. Table 2 shows specification of each component on measurement environment.

- **(a) FPGA-based Attacker Emulator** : Attacker emulator generates packets related to DNS amplifier attack. More specifically, it generates DNS response packets with response bit by changing the combinations of source IP address and destination UDP and sends them to the victim host via mitiKV in 10Gbps line rate.
- **(b) FPGA-based TAP device** : In order to measure packets per second (pps) between mitiKV and the victim host, we installed the FPGA-based TAP device. The TAP device has four Ethernet port.

Table 2: Components of measurement environment.

	Hardware	OS	NIC + Driver	Main memory
(a)	Xilinx VC709	—	—	—
(b)	Xilinx VC709	—	—	—
(c)	Intel Core i5-4590	FreeBSD 10.2R	Intel X520-DA2 + ixgbe 2.8.3	4GB
(d)	Intel Core i7-4770	Fedora 24 (Linux 4.6.6)	Intel X520-DA2 + ixgbe 4.2.1	32GB

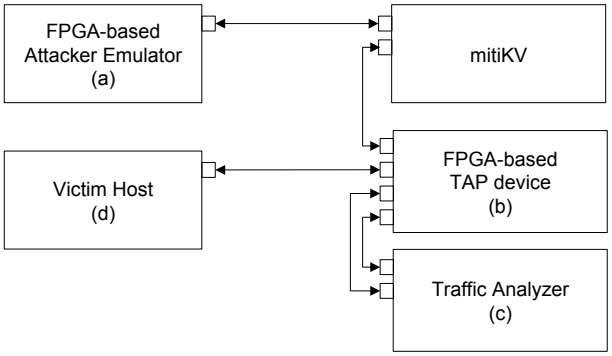

Figure 8: Evaluation envrionment.

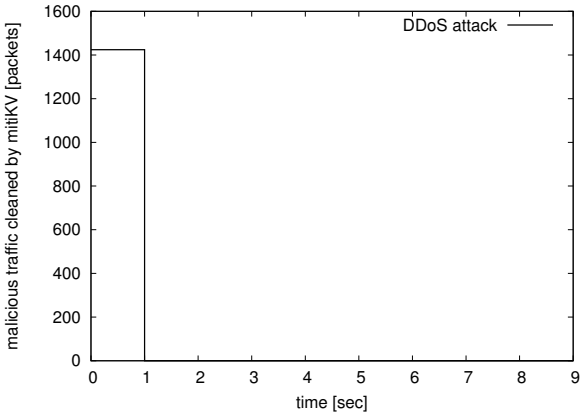

Figure 9: DDoS attacking against incremental 1000 destination UDP ports.

Two ports are bridging. Port0 is connected to mitiKV. Port1 is connected to victim host. Remaining two ports are connected to (c) traffic analyzer for mirroring of inbound and outbound traffic.

- **(c) Traffic Analyzer** : The traffic analyzer host measures packets per second from mitiKV's outbound port and victim host's inbound port. This analyzer acts as a packet counter using netmap framework [16] to measure pps in two 10GbE ports: victim host's RX and TX ports.
- **(d) Victim Host** : We assume that victim host is a general Linux machine.

4.2.2. ICMP kernel tuning

A general Linux machine is set as ratelimit's parameter that defines a packet per one second against one host. Since the mitiKV is located at transit link of network service provider, mitiKV needs to observe all packets in the located network and the connected network. To evaluate mitiKV hardware, the Linux machine denoted in (d) is required to emulate multiple victim hosts located in connected network. We configured kernel parameters for ICMP on victim host for emulating multiple victim hosts. A Linux host returns an ICMP packet with a port unreachable message per one second against one remote host in an environment with default kernel parameters. Here, we tuned the following parameters as returned packets per one second.

- /proc/sys/net/ipv4/icmp_ratelimit : 0
- /proc/sys/net/ipv4/icmp_msgs_per_sec : 14880000
- /proc/sys/net/ipv4/icmp_msgs_burst : 25600

In a practical use case, an attacker sends spoofed queries against multiple multiple open resolvers. The victim host receives multiple reponse packets from open resolvers. Although the above parameters to achieve high performace for the ICMP response packet which indicates error messages are not required, more detail analysis is needed to mitigate actual DDoS traffic.

In this paper, we do not make no mention of these paramters.

4.2.3. Results

Figure 9 shows DDoS attack against incremental 1000 destination UDP ports on an environment as shown in Figure 8. This measurement is performed on

(c) traffic analyzer. Received packets from two network interfaces on the traffic analyzer machine represents the number of DDoS attack packets and ICMP port unreachable messages, respectively. The figure indicates that our proposed detection method takes time to learn all DDoS flows. Although the latency between victim host and mitiKV is small due to the back-to-back connection in this evaluation, the milisecond level latency to achieve ICMP port unreachble messages to the mitiKV occurs due to network devices in the practical case. In this case, victim may recieve more attacks in a period of learning flows on the mitiKV because it takes more time to generate DDoS flows.

4.3. Scalability up to 100Gbps link

A transit link will be replaced with 100Gbps and 400Gbps high bandwidths link interface to provide network services to thier customer. We investigated MAC IP specification provided by Xilinx and calculated required clock cycles for a hardware-based pipeline. Table 3 shows the scalability on 10GbE, 40GbE and 100GbE. To support 100GbE interface, high-end FPGA series are required to implement mitiKV. It is a simple pipeline to design 40GbE and 100GbE because required clock cycles related to packet parsing for detection of DNS response are reduced due to increasing data width of MAC on 40GbE and 100GbE. To support high-speed interfaces such as 40GbE and 100GbE, a custom FPGA board equipped with these interface is required. Our mitiKV core design is applicable on these intefaces.

4.4. ICMP Port unreachable message

The proposed method in this paper heavily relies on the message encoded by the network stack of the victim hosts. Therefore understanding how the ICMP destination unreachable message is observed at a transit link is important since some implementations may not embed the original packet information which triggers port unreachable message, some middle-boxes may strip the packet or just simply filtered out. The standard says *If a higher level protocol uses port numbers, they are assumed to be in the first 64 data bits of the original datagram's data.* [11], which is interpreted as 28 bytes in IPv4 (i.e., 20 bytes IP header and 8 bytes higher protocol datagram) of the original packet is in the payload. In the IPv6 standard [17] it specifies differently with 1280 bytes payload at the maximum size.

In this section, we investigate the availability of the key information for our proposed method by analyzing the packet trace at a public research and educational backbone network. In summary, the ICMP port unreachable messages we observed fulfill our expectation

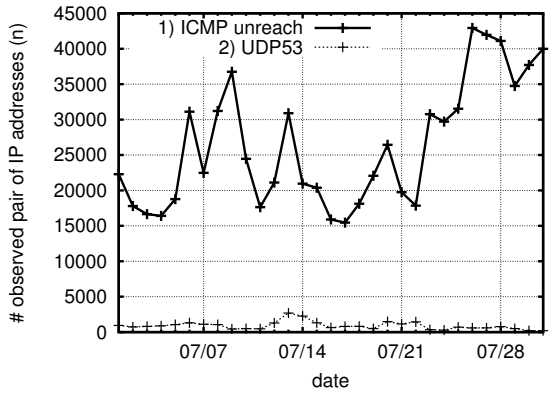

Figure 10: Number of IP address pairs among 1) all of observed ICMP port unreachable messages, and 2) 1) with DNS packet in the payload.

in order to detect UDP-based amplification attack. We will discuss the detail in the following description.

Dataset

We used traffic traces from MAWI (Measurement and Analysis on the WIDE Internet) archive samplepoints F [18] in July 2016 (i.e., 31 days). The archive includes 15 minutes daily packet trace at the transit link of the backbone network, with anonymized IP addresses in the traces.

Although the traffic trace only recorded a short duration in a day as well as with partial length (first 96 octets) of packets, it contains enough information of what we are investigating here - we only need the payload of IPv4 ICMP destination unreachable message and the trend of ICMP message, not the full number of messages exchanged, to justify how our proposed method is practical in a wild.

Point of interests

We focus on the number of source and destination pairs of IP addresses in the trace which 1) have ICMP destination port unreachable (type 3 code 3) messages, 2) 1) with DNS packet (port 53 of UDP packet) in the payload. In addition to that, we counted the distribution of packet size which each packet contains ICMP destination unreachable message in order to study how the network stack implementation encodes the original packet when it sends back the error.

Results

Figure 10 shows the number of observed flows in the packet traces which contains the ICMP destination unreachable messages and payloads which the original packets were DNS packets. Figure 11 plots the distribution of packet size among all of ICMP port unreachable messages.

Table 3: Scalability.

	10GbE	40GbE	100GbE
Clock Frequency [MHz]	156.25	156.25	322.266
Data Width [bit]	64	256	512
DNS Reply Detection [clock cycles]	8	2	1
Hash Function (CRC32) [clock cycles]	1	1	1
KVS Processing [clock cycles]	2	2	2
Pipilining Depth	11	5	4

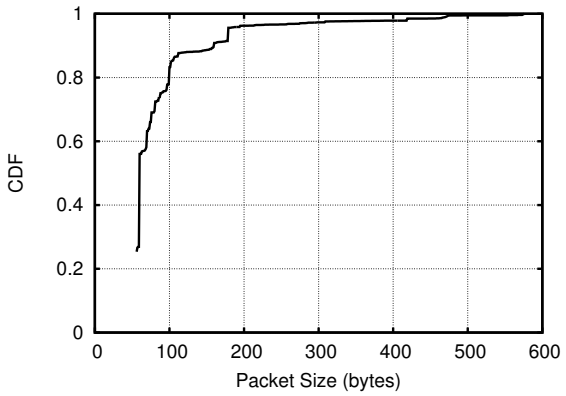

Figure 11: Cumulative distribution of the packet size in the ICMP destination error message, which indicates the existence of the original packet to identify it is DNS packet or not.

As we can see in Figure 10, there are a number of ICMP port unreachable messages which the original packets are DNS related (request or response). Also Figure 11 represents that the minimum packet size of the original packet is 28 bytes[5], which can contain the port number of UDP to identify if the packet is DNS or not.

Above information confirms that there are sufficient information to identify the packet under suspicious by our proposed method based on the ICMP port unreachable messages.

5. Conclusion

We proposed a novel hardware-based DDoS mitigation system, called mitiKV, that focuses on the DNS-based amplification attack. We focused on protocol behavior of the DNS-based amplification attack and the ICMP port unreachable message. The mitiKV detects DDoS attack and managed on hardware-based key value store. We analyzed real world traffic and showed that ICMP port unreachable messages have packet payload including required data for our protocol-based approach. We implemented a prototype system on an FPGA board and showed that mitiKV can mitigate malicious traffic in 10GbE. We discussed the scalability of our approach for further high throughput interfaces, such as 40GbE and 100GbE.

[5]The minimum packet size 56 bytes of ICMP unreachable message can be interpreted as 28 bytes the original packet since the size of IPv4 header is 20 bytes and the size of ICMP Destination Unreachable Message without the Original datagram is 8 bytes.

Ackowledgment

The authors would like to thank NTT Communications for providing a testing environment.

References

[1] Arbor Networks, Worldwide infrastructure security report, https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf.

[2] Open resolver project, http://openresolverproject.org/.

[3] J. Mirkovic, P. Reiher, A taxonomy of ddos attack and ddos defense mechanisms, SIGCOMM Comput. Commun. Rev. 34 (2) (2004) 39–53.

[4] Y. Kim, W. C. Lau, M. C. Chuah, H. J. Chao, Packetscore: a statistics-based packet filtering scheme against distributed denial-of-service attacks, IEEE Transactions on Dependable and Secure Computing 3 (2) (2006) 141–155.

[5] Snort, Snort - network intrusion detection and prevention system, https://www.snort.org/.

[6] H. J. Chao, R. Karri, W. C. Lau, Cysep - a cyber-security processor for 10 gbps networks and beyond, in: Military Communications Conference, 2004. MILCOM 2004. 2004 IEEE, Vol. 2, 2004, pp. 1114–1122 Vol. 2.

[7] P. Djalaliev, M. Jamshed, N. Farnan, J. Brustoloni, Sentinel: Hardware-accelerated mitigation of bot-based ddos attacks, in: 2008 Proceedings of 17th International Conference on Computer Communications and Networks, 2008, pp. 1–8.

[8] K. Pandiyarajan, S. Haridas, K. Varghese, Transparent fpga based device for sql ddos mitigation, in: Field-Programmable Technology (FPT), 2013 International Conference on, 2013, pp. 82–89.

[9] L. V. Ahn, M. Blum, N. J. Hopper, J. Langford, Captcha: Using hard ai problems for security, in: Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques, EUROCRYPT'03, Springer-Verlag, Berlin, Heidelberg, 2003, pp. 294–311.

[10] G. Vasiliadis, L. Koromilas, M. Polychronakis, S. Ioannidis, Gaspp: A gpu-accelerated stateful packet processing framework, in: 2014 USENIX Annual Technical Conference (USENIX ATC 14), USENIX Association, Philadelphia, PA, 2014, pp. 321–332.

[11] J. Postel, et al., RFC 792: Internet control message protocol, InterNet Network Working Group.

[12] NetFPGA Project, http://netfpga.org/.

[13] N. Zilberman, Y. Audzevich, G. Covington, A. Moore, NetFPGA SUME: Toward 100 Gbps as Research Commodity, IEEE Micro 34 (5) (2014) 32–41.

[14] M. Blott, L. Liu, K. Karras, K. Vissers, Scaling Out to a Single-Node 80Gbps Memcached Server with 40Terabytes of Memory, in: Proceedings of the USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'15), 2015.

[15] S. R. Chalamalasetti, K. Lim, M. Wright, A. AuYoung, P. Ranganathan, M. Margala, An FPGA Memcached Appliance, in: Proceedings of the International Symposium on Field Programmable Gate Arrays (FPGA'13), 2013, pp. 245–254.

[16] L. Rizzo, netmap: A Novel Framework for Fast Packet I/O, in: Proceedings of the USENIX Security Symposium (Security'12), 2012, pp. 101–112.

[17] A. Conta, S. E. Deering, M. G. Ed, RFC 4443: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, Network Working Group.

[18] K. Cho, K. Mitsuya, A. Kato, Traffic data repository at the wide project, in: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC '00, USENIX Association, Berkeley, CA, USA, 2000, pp. 51–51.