Commoditising DDoS Mitigation

Alfredo Cardigliano
ntop
Pisa, Italy
cardigliano@ntop.org

Luca Deri
ntop, IIT/CNR
Pisa, Italy
deri@ntop.org, luca.deri@iit.cnr.it

Tord Lundstrom
Virtualroad
Stockholm, Sweden
t@virtualroad.org

***Abstract*—Current practices in network security deployment require multiple specialised devices as firewalls, traffic shapers, sensors or Intrusion Detection Systems (IDSs) to handle malicious traffic. This practice not only increases the overall operational costs but also makes network administration complicated. The high cost of Distributed Denial of Service (DDoS) mitigation devices empowers centralised services and network architectures as there is not a cost-effective model to deploy them at the "true edge" of the network.**

This paper describes the design and implementation of a multi-10 Gbit extensible network traffic analysis and policing system. It is composed of logical detection and enforcement functions built from reusable underlying primitives. As an example of such modular approach, we present an innovative DDoS scrubbing system composed of various attack detection primitives, combined with enforcement primitives that include traffic filtering, rate limiting, and proxying. Based on commodity hardware and open source software, such system is price, space, and power efficient enough to be practically deployable at the edge of the network. Performance measurements carried on 10Gbit networks, show that it can effectively provide both traffic visibility and enforcement of a wide range of network traffic policies.

Keywords—Datacenter networks; Internet measurement and modelling; Network control and management; Routing and traffic engineering.

I. Introduction and Motivation

Despite the increasing deployment of devices for traffic analysis and mitigation, the number of Distributed Denial of Service (DDoS) attacks recorded last year [1] is more than double the number of attacks of 2014. Unlike attacks generated in the past years that lasted until the attack was mitigated or the the attacker was taken down, currently the large majority of DDoS attacks are launched from stresser/booter-based services [9, 14] and DDoS-for-hire tools [15]. Services able to generate high packet rates and volumes of traffic are available for less than 40$/hour [8] making DDoS attacks affordable for everyone. The most common attacks include UDP, ICMP and TCP (SYN) floods, amplification attacks [5, 11] using reflection techniques [40], and application-based specific attacks targeting HTTP and DNS servers. Both volumetric and protocol specific attacks often take advantage of the ability to spoof traffic. Spoofing not only increases the attack impact, but it makes more challenging the traceback to the real origin of the malicious traffic and thus who has created the malicious traffic. When volumetric attacks (aiming at the exhaustion of available bandwidth) are not enough, attackers often exploit protocol specific vulnerabilities including:

- TCP state-exhaustion attacks that attempt to consume TCP state tables of infrastructure components such as load balancers and application servers [17].
- DNS attacks as those that aim to poison caches or flood servers with bogus lookups and high rate requests.
- Application-layer attacks as those that originate by few machines that generate low traffic rates [18]. They are difficult to be detected by traditional flow-based monitoring applications [13].
- Popular DDoS attacks target the online-gaming industry and financial services, and widespread software applications (e.g. Joomla, Wordpress) that are often used for reflection attacks [6].

While academia has extensively investigated the impact of DDoS attacks, and suggests techniques for detecting and mitigating them [7, 10, 14, 19, 24, 25], a quick tour on github.com shows that all the publicly available DDoS tools are either botnet simulators or application-specific anti-DDoS tools (e.g. Apache and nginx HTTP server protection modules). Contrary to other security markets such as IDS/IPS (Intrusion Detection/Prevention System) where there are many publicly available tools, most DDoS protection tools are very coarse grained [4, 12, 21] or just a proof-of-concept [2, 3, 16, 20, 22, 23]. In the industry, DDoS mitigation devices and services are often expensive and not always affordable by everyone [26], thus making them suitable for permanently protection only by large ISP (Internet Service Providers). A cheaper alternative to permanent protection, is to use on-demand DDoS mitigation services; when a DDoS attack is detected, traffic is rerouted via BGP or DNS towards a scrubbing center, a place where ingress traffic is analysed and cleaned before being tunnelled back or proxied to the victim's original destination. Their high hosts makes them unsuitable to be deployed at the edge of the network where potential victims/attackers reside. Furthermore being these devices deployed as black-boxes, network administrators are can do only very limited customisation including their inability to develop specific plugins/code for dissecting protocols not supported by the box.

In the past years we have implemented various tools for efficient packet processing [27, 28]. The core of our activities traditionally focused on network traffic monitoring including content analysis, deep packet inspection, packet balancing, and flow generation. We have decided to merge all these heterogeneous tools into a single application in order to create a comprehensive tool able to satisfy most network monitoring

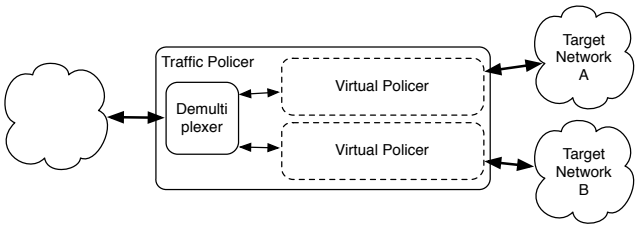
needs in a single tool. The high cost of commercial solutions and lack of flexible and efficient tools for DDoS mitigation has motivated us to design and built a novel traffic mitigation tool named nScrub. It runs on commodity hardware and has been designed to be a user-extensible mitigation tool that could be deployed as a true network-edge protection.

The rest of the paper is structured as follows. Section II covers the architecture and design principles of nScrub. Section III describes the validation process and experiments. Section IV highlights some future work activities, and finally section V concludes the paper.

II. ARCHITECTURE

nScrub is a highly configurable and scalable system designed to run on sub-1000$ servers and able to implement 10 Gbit full-duplex traffic mitigation. It operates inline with traffic, by examining traffic content. It can forward, discard and inject packets as needed. In nScrub, hosts are partitioned into two big groups: potential victims (the networks nScrub protects) and potential attackers (external hosts that can potentially attack our network). nScrub can operate as:

- Transparent bridge: in this mode no special configuration is necessary, as it works as bump-in-the-wire.
- Network router which can be combined with BGP (Border Gateway Protocol) diversion techniques.

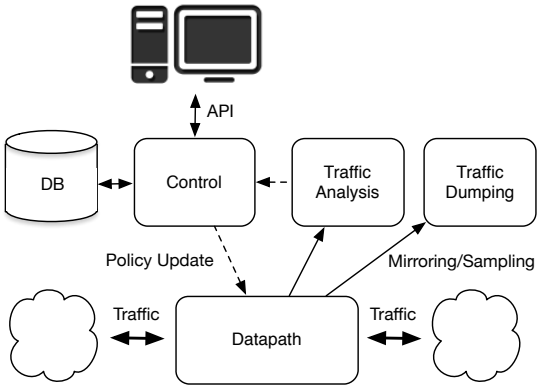


1. Virtual Policers Demultiplexing

Network traffic is processed according to user-defined policies. The system can be partitioned into multiple virtual policers, represented by profiles, which in essence are traffic policies configurations. Traffic demultiplexing between virtual policers takes place based on destination IP subnet. This enables the definition of multiple profiles in order to handle different target services (e.g. a game server and a web server) and thus implement flexible traffic polishing configurations based on the service being protected.

As depicted in Figure 2, nScrub is partitioned into the following components:

- Control: it is responsible for configuring the engine.
- Data path: it implements the mechanism for traffic enforcement.
- Traffic analysis: it observes traffic and triggers policies updates when specific events occur.
- Traffic dumping: it records to disk specific traffic that could be used for in-depth analysis.



2. nScrub System Components

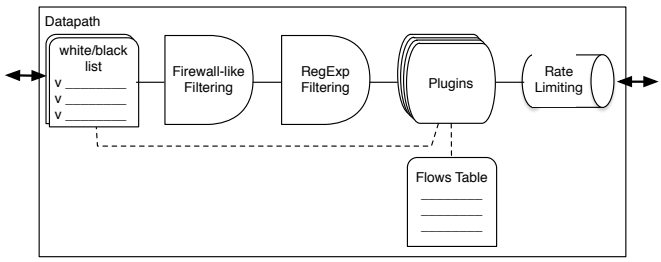
The following subsections describe in detail the various system components, their functionality, and how they interact.

A. Control

The control component is responsible for configuring the engine based on policies specified by the user. Configuration requests are sent to this component by the user using an HTTP-based RESTful API, and stored into a persistent database so that they can be read at startup. Through this mechanism, new policies can be injected at runtime in various ways: manually from a graphical user interface, or dynamically from external (to nScrub) threat detection and traffic analysis applications.

B. Data Path

The data path component is responsible for forwarding traffic across network interfaces and implementing mechanisms for traffic policy enforcement.



3. Data Path Submodules

This component is logically divided into a pipeline of submodules, each responsible for a specific traffic filtering task or policy enforcement mechanism:

- White/Black list: this component is responsible for handling lists of subnets with specific network access restrictions. List types include (but not limited to) white and black lists. Each list is bound to a user-defined profile which includes the policies that apply to the corresponding subnets. Subnets can be manually (i.e. by end-users) or dynamically (i.e. though traffic check mechanisms and specific plugins) added to lists.
- Firewall-like filtering: it implements simple ACL (Access Control List) rules based on protocol, port and other relevant packet header fields.

- Payload searching: it supports packet payload content inspection in order to identify and filter specific applications, viruses, and malicious requests. Currently, it is possible to specify strings for searching at specific offsets. Strings are dynamically reconfigurable at runtime so that users can add/modify them when a new attack has been detected, and a specific plugin has not been developed yet. As described in section IV we are planning to make this facility even more powerful in future application releases.
- Plugins: the policer has been designed as an open platform. This means that it can be extended with custom plugins that implement specific filtering facilities that can be used in addition to those implemented by the application core. For instance a specific UDP-based gaming protocol not natively supported by the system, can be protected by coding a custom plugin able to dissect its specific protocol. Such plugin can sanitise the packet, or implement behaviour analysis by means of stateful algorithms. Plugins can interact with the nScrub flow table, used for keeping track of stateful connection state, and with white/black lists to forward traffic coming from legitimate users or drop attackers. Additionally, plugins can also inject traffic when necessary (e.g. send a TCP RST packet to terminate unwanted connections), and for implementing techniques such as SYN Cookie [32, 33], useful for mitigating SYN flood attacks.
- Rate limiting: rate traffic from selected subnets or towards specific target networks. The rate can be controlled both in terms of maximum bandwidth and packet rate, according to traffic type and matching profile. For instance, by using this module it is possible to limit the number of incoming TCP connections per source IP or towards a specific service.

C. *Traffic Analysis*

The system provides auxiliary monitor queues (bound to either virtual or physical interfaces) where traffic analysis applications can be connected. A full or sampled copy of the nScrub processed traffic (bad, good or all) can be sent to the queue. This mechanism is very useful for:

- Enabling traffic visibility.
- Detection of low-bandwidth and slow DDoS attacks. As described in section IV, traffic analysis applications, such as ntopng [39], can trigger policies updates when these threats are detected.

D. *Traffic Dumping*

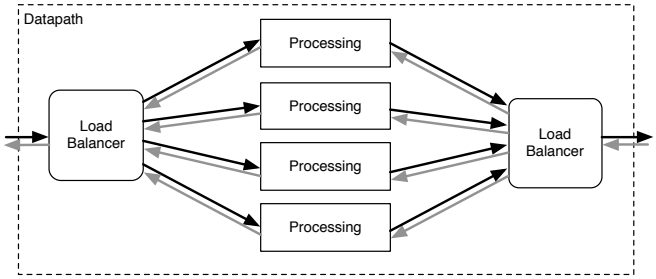
Through the auxiliary queues implemented by nScrub, external packet-to-disk traffic recording applications can be connected to the system. They can record relevant (bad, good or all) traffic for post-analysis that can be inspected using specific tools such as Wireshark. It is also possible to set thresholds for dumping malicious traffic as soon as a volumetric attack starts. This is to implement visibility of attacks type and behaviour without the need to continuously record all received packets.

III. IMPLEMENTATION

This section describes the current nScrub implementation.

A. *Packet Processing Framework*

In inline packet processing applications, packet processing must be quick, low-latency and efficient. In order to achieve this design goal, nScrub has been coded on top of PF_RING ZC (Zero Copy). PF_RING ZC [27] is a home-grown flexible packet processing framework that allows applications to achieve 10 Gbit line-rate packet capture and transmission, at any packet size. In addition to speed, with PF_RING ZC it is possible to perform all operations in zero copy, thus saving memory bandwidth, optimising latency, and reducing CPU load. Furthermore as the kernel is bypassed and the NICs have no IP, it guarantees that the system running the scrubber is not affected by DDoS attacks. Operations include packet balancing across multiple processing threads and interface-to-interface packet forwarding in zero copy. Due to the high 10 Gbit packet rate, multiple processing units are needed for implementing the data path component. When traffic enters the policer, it is split across the various processing units in zero copy by means of a load balancing function.



4. Load balancing with PF_RING ZC

This way packets belonging to the same flow are all bound to a specific processing unit, as otherwise filtering or scrubbing techniques based on stateful algorithms may do not operate as expected. For instance the SYN Cookie algorithm will not work if packets belonging to the same source and destination pair are not all handled by the same processing unit. Load balancing is performed by a distribution unit that uses a hash of the IP address to select a destination processing unit.

B. *Core Features*

The current implementation features:

- 10 Gigabit full-duplex packet processing based on PF_RING ZC, with hardware bypass support when available in the network adapter.
- Hardware-bypass support (when available in the network adapter) based on a watchdog/heartbeat mechanism, to keep forwarding traffic in case of system failures, including application crashes, system reboot, or other software/hardware failures.
- Support for symmetric/asymmetric working modes. In asymmetric mode the system is able to inspect inbound traffic only (from the Internet to the target network), whereas in symmetric mode both traffic directions are inspected.
- Routing support: it is possible to re-route traffic using BGP diversion techniques and re-inject traffic towards the target network.

- Support for lists (set of subnets), including (but not limited to) white and black lists (implemented with radix trees). Additional lists, such as grey lists, can be defined for applying selected policies/profiles to specific subnets.
- TCP traffic check algorithms including SYN Cookie [32, 33] and SYN Sanity Check, that include active and passive checks for enforcing compliancy with the RFC specification.
- ACL-like filtering on TCP/UDP source/destination port, ICMP type accept/drop, etc.
- DNS packet check algorithms, and payload pattern matching for filtering layer 7 attacks.
- Rate limiting, including the ability to rate based on source/destination IP, and traffic type (implemented with leaky buckets algorithms).
- Traffic mirror through virtual interfaces (i.e. PF_RING ZC queues) for connecting packet-to-disk applications, traffic analysis applications, and IDSs.
- Low bandwidth and slow-DDoS attack detection by forwarding packets to ntopng.
- RESTful API (see figure 2) for checking the system status and changing the system configuration (policies, access control lists, etc.) at runtime.

III. VALIDATION

This section describes how the system has been validated and what is the expected application performance. In order to verify the performance on different setups, we have performed tests using real traffic in both symmetrical and asymmetrical traffic configurations. The feedback received by users protecting their production network using nScrub has helped us to improve the algorithms and mitigation strategies. While advanced users and administrators often prefer highly configurable systems, our development efforts are going towards nearly zeroconf configuration. We are confident that by quickly detecting variations of attack vectors, we can improve mitigation by dynamically selecting adequate responses. For this reason we believe that flexibility and adaptability at nearly zeroconf is not only a desired feature but the ultimate requirement if we want mitigation technologies to become widely adopted.

A. *Effectiveness*

nScrub has been tested using traffic generated using known booter's leaked toolkits [34, 35] and packet captures (recorded during real DDoS attacks) at increasing packet rate. Our tests also include three of the most common attack vectors: SYN flood attacks [29, 33], TCP-based amplification attacks [30, 32] and UDP-based amplification attacks [31]. In all the performed tests the system protected by nScrub was able to stay up with no significant impact in the response time.

B. *Performance*

The performance tests have been done on a low-end sub-1000$ Dell 220, based on an Intel Xeon E3 processor. nScrub was installed on a E3-1230 v3 @ 3.30GHz dual-channel with 4 cores and hyper-threading enabled. The system was equipped with 4 DDR3 8GB 1600MHz RAM modules and a dual-port Intel 82599ES-based Intel 10 Gbit NIC. The mitigation system was configured with four processing threads, each bound to a different physical core. nScrub was benchmarked during four different attack types using a traffic generator based on PF_RING ZC simulating real traffic from a SYN flood and UDP-based amplification attacks. In all the tests, we have selected the smallest packet size (10 Gigabit line rate at 14.88 Mpps), this to evaluate the system performance in the worst case scenario.

For SYN flood attacks, two mitigation methods have been implemented and compared: SYN Cookie and SYN Sanity Check. Contrary to the SYN Sanity Check that does not challenge traffic, the SYN Cookie method needs to inject at least one packet back for every processed SYN in order to validate clients. As shown in Figure 5, nScrub was able to process 96.7% of all traffic using the SYN Sanity Check and 86.6% or about 13 Mpps using a SYN Cookie challenge. Performance degradation is reached with memory exhaustion as the SYN Cookie requires more resources to forge and inject the traffic than other mitigation techniques.

Check Method	*Received*	*Lost*	*Injected*	*Forwarded*
TCP SYN-Cookie	12.9 Mpps	1.98 Mpps	12.9 Mpps	0
TCP Sanity	14.4 Mpps	0.48 Mpps	0	0
UDP Drop All	14.88 Mpps	0	0	0
UDP Pass All	14.88 Mpps	0	0	14.88 Mpps

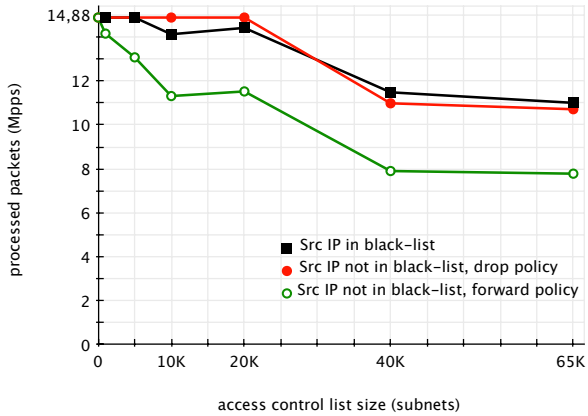
5. Traffic processing speed (14.88 Mpps ingress rate)

Application scalability is implemented through RSS that can partition traffic across various virtual queued to which individual scrubber threads connect.

RSS	*Pass-All*	*Whitelist Lookup 10k IPs*
1	9.2 Mpps	4.8 Mpps
4	14.88 Mpps	14.1 Mpps

6. Forwarding speed vs RSS (14.88 Mpps ingress rate)

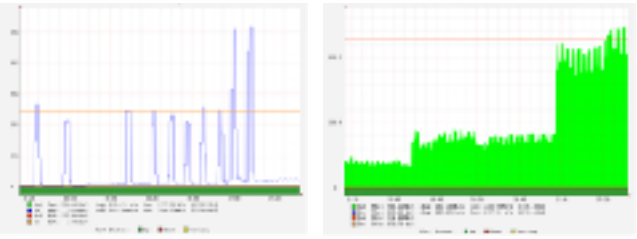
In order to test the system under UDP attacks we have generated UDP traffic with minimum packet size of 60 bytes.



7. Traffic processing performance vs access control list size on 10 Gbit line-rate traffic (14.88 Mpps ingress).

nScrub was configured in two scenarios: a rule to check and drop (drop all) traffic, and a second scenario to match and forward all traffic (pass all). In both cases the system was able to keep up with the ingress traffic, even in the "forward all" worse case scenario where memory pressure was not too high thanks to the zero copy packet forwarding support provided by PF_RING ZC. In other tests we have evaluated the impact of IP access control lists in the performance of the nScrub. As depicted in figure 6, with the same hardware configuration nScrub was able to handle 20.000 rules at full line-rate when dropping bad traffic, or more than 11 Mpps if traffic was forwarded.

nScrub was also tested in various production networks across the Internet. The following pictures are taken from a real production network during a DDoS attack. In this case nScrub was deployed inline between the edge router and the internal network.



8. Network Traffic During DDoS Attack before/after nScrub

As depicted in the above figure, the network was hit by ingress traffic spikes that accounted 3 to 5 Gbit of traffic. The figure depicts the same network traffic as observed on the interface connecting nScrub to the protected network (i.e. after traffic mitigation): ingress rate passed from a 5 Gbit to a 200 Mbit of only good traffic that passed nScrub controls. Thanks to nScrub, the ingress traffic peaks have been mitigated and they did not reach the internal network that was then protected from the DDoS attack.

C. *Robustness*

The scrubber can be a single point of failure if the hardware breaks or nScrub crashes. To avoid that it is important to implement protection mechanisms. nScrub supports hardware bypass NICs via a watchdog mechanism, so that in case of failure the box can let the (unfiltered) traffic go through. In addition to that, standard network protection mechanisms such as BGP or VRRP (e.g. Linux keepalived) can be used to react to failures.

IV. Open Issues and Future Work Items

While nScrub is in production in many locations since more than a year, we acknowledge that there are some open issues that need to be tackled in future releases. They include:

- Filtering offload. We are prototyping various techniques for offloading traffic filtering to external devices including external OpenFlow-based switches, embedded switches such as Silicom Redirector NICs and Intel FM1000-based NICs.
- Plugins are statically compiled into nScrub, preventing users from upgrading the application without restarting it. One of the features that we are planning to implement, is the ability to reload plugins at runtime. This feature has already been integrated in other applications we have developed in the past [36]. Once the feature is implemented, we can enable selective application update without using offline traffic mechanisms to let traffic go through, while the application is reconfigured/updated.
- Implementation of additional plugins for sanitising protocols such as SIP and RTP. This would allow nScrub to discard protocol requests that are either invalid or potentially dangerous.
- Currently nScrub is able to search specific strings at a specified packet payload offset (implemented with memcmp()-like functions). While this feature is enough for most users, we are planning to add support for regular expressions or exact pattern/string matching (faster than the former in terms of CPU cycles) that are popular techniques for filtering layer 7 attacks.
- Low rate DDoS attack detection using statistical analysis [37, 38]. Having designed nScrub as a high-speed packet scrubber, we believe that this type of detection should be performed on an external application that can keep flow state information. In fact, implementing this functionality into nScrub, might not be a good idea as specific attacks could leverage on flow state information to tear down nScrub and thus disrupt Internet connectivity. For this reason we are implementing statistical traffic analysis in ntopng that detects non-volumetric DDoS attacks and injects into nScrub specific rules for dropping them.

V. Final Remarks

This paper has covered the design and implementation of a pure software-based DDoS mitigation application named nScrub. It leverages on several years of research carried on by the authors in the field of high-speed packet processing and traffic monitoring. nScrub core features include traffic filtering, rate limiting and proxying, as well as scrubbing popular protocols such as DNS. The integration of network traffic visibility with policy enforcement allowed us to create a tool open to third party applications, while implementing facilities typically present in high-end hardware devices that are unlikely to be deployed at the edge of the network due to their high cost and complexity.

The application has been validated both in a laboratory using synthetic traffic, and for over a year in production networks with live traffic. The validation phase has demonstrated that using a sub-1000$ server, nScrub performs well enough for protecting networks connected to 10 Gbit links. This makes it practically deployable at the edge of the network for providing both traffic visibility and the enforcement of a wide range of network traffic policies.

Acknowledgment

The authors would like to thank users and early adopters of nScrub who have tested it in real-life traffic scenarios. Without their help, feedback, comments and suggestions, we would not have been able to validate it on heterogeneous setups and real-life challenging deployment locations.

AVAILABILITY

nScrub can be downloaded at http://packages.ntop.org/.

REFERENCES

1. Akamai, Akamai's [state of the internet] / security, Q4 2015 Report, 2015.
2. Mar Callau-Zori, Ricardo Jiménez Peris, Vincenzo Gulisano, Marina Papatriantafilou, Zhang Fu, and Marta Patiño Martínez. STONE: a stream based DDoS defense framework. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, pages 807–812. ACM, 2013
3. Jinghe Jin, Chaetae Im, and Seung Yeob Nam. Mitigating HTTP GET Flooding Attacks through Modified NetFPGA Reference Router. 2009.
4. Mattijs Jonker and Anna Sperotto. Mitigating DDoS Attacks using OpenFlow-based Software Defined Networking. In Proc. of the 9th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security (AIMS'15), pages 129–133, 2015.
5. Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. Exit from Hell? Reducing the Impact of Amplification DDoS Attacks. In Proceedings of the 23rd USENIX Security Symposium, August 2014.
6. Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. Hell of a Handshake: Abusing TCP for Reflective Amplification DDoS Attacks. In Proceedings of the 8th USENIX Workshop on Offensive Technologies (WOOT '14), August 2014.
7. Lukas Krämer, Johannes Krupp, Daisuke Makita, Tomomi Nishizoe, Takashi Koide, Katsunari Yoshioka, and Christian Rossow. AmpPot: Monitoring and Defending Amplification DDoS Attacks. In Proceedings of the 18th International Symposium on Research in Attacks, Intrusions and Defenses, November 2015.
8. Tara Seals, DDoS-for-Hire Costs Just $38 per Hour, InfoSec Magazine, http://www.infosecurity-magazine.com/news/ddosforhire-costs-just-38-per-hour/, June 2015.
9. Giancarlo Pellegrino, Christian Rossow, and Matthias Waehlisch Fabrice J. Ryba, Thomas C. Schmidt. Cashing Out the Great Cannon? On Browser Based DDoS Attacks and Economics. In Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT '15), August 2015.
10. F. Ricciato, A. Coluccia, and A. D'Alconzo. A review of DoS attack models for 3G cellular networks from a system design perspective. Computer Communications, 33(5), March 2010.
11. Christian Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium, February 2014.
12. Rishikesh Sahay, Gregory Blanc, Zonghua Zhang, and Hervé Debar. Towards Autonomic DDoS Mitigation using Software Defined Networking. 2015.
13. R. Sadre, A. Sperotto, and A. Pras. The effects of DDoS attacks on flow monitoring applications. In Proceedings of the 2012 IEEE Network Operations and Management Symposium (NOMS 2012), Maui, Hawaii, pages 269–277, USA, April 2012. IEEE Computer Society.
14. J.J Santanna, Roland van Rijswijk Deij, Anna Sperotto, Rick Hofstede, Mark Wierbosch, L. Zamdebenedetti, Arne Welzel, Christian Rossow, and Herbert Bos. On Measuring the Impact of DDoS Botnets. In Proceedings of the 7th European Workshop on Systems Security (EuroSec 2014), April 2014.
15. Granville, and Aiko Pras. Booters – An Analysis of DDoS-as-a-Service Attacks. In Proc. of the 14th IFIP/IEEE Symposium on Integrated Network and Service Management (IM'15), pages 243– 251, 2015.
16. Z. Fu, M. Papatriantafilou, and P. Tsigas. Club: a cluster based framework for mitigating distributed denial of service attacks. In ACM SAC'11, pages 520–527, 2011.
17. Schuba, Christoph L., et al., Analysis of a denial of service attack on TCP, Proceedings of IEEE Symposium on Security and Privacy, IEEE, 1997.
18. ha.ckers.org, Slowloris HTTP DoS, http://ha.ckers.org/slowloris/, 2009.
19. R. R. Kompella, S . Singh, and G. Varghese. On scalable attack detection in the network. IEEE/ACM Trans. Netw., 15(1):14–25, 2007.
20. K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments, Computer Networks, vol. 62, no. 0, pp. 122 – 136, 2014.
21. N. Hachem, H. Debar, and J. Garcia-Alfaro, HADEGA: A novel MPLS-based mitigation solution to handle network attacks, in 31st IEEE International Performance Computing and Communications Conference (IPCCC), Dec 2012, pp. 171–180.
22. J. Ioannidis and S. M. Bellovin, Implementing Pushback: Router- Based Defense Against DDoS Attacks, in Proceedings of Network and Distributed System Security Symposium (NDSS), 2002.
23. A. Yaar, A. Perrig, and D. Song, SIFF: a stateless Internet flow filter to mitigate DDoS flooding attacks, in Proceedings of the 2004 IEEE Symposium on Security and Privacy, May 2004, pp. 130–143.
24. Andrew Carlin , Mohammad Hammoudeh, Omar Aldabbas, Defence for Distributed Denial of Service Attacks in Cloud Computing, In International Conference on Advanced Wireless Information and Communication Technologies (AWICT 2015), 2015.
25. Latin Patel, Vijay Katkar, Chandra Suresh Satapathy, Amit Joshi, Nilesh Modi, Nisarg Pathak, A Multi-classifiers Based Novel DoS/DDoS Attack Detection Using Fuzzy Logic, Proceedings of International Conference on ICT for Sustainable Development: ICT4SD 2015 Volume 2, 2015.
26. Reddit, DDoS Solution Comparison, https://www.reddit.com/r/sysadmin/comments/3gqeqv/ddos_solution_comparison/, 2015.
27. Sebastian Gallenmüller, Paul Emmerich, Florian Wohlfart, Daniel Raumer, and Georg Carle, Comparison of Frameworks for High-Performance Packet IO, Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for networking and communications systems (ANCS '15), 2015.
28. Luca Deri and Francesco Fusco. High Speed Network Traffic Analysis with Commodity Multicore Systems. In Proceedings of IMC 2010, November 2010.
29. Mitko Bogdanoski, Tomislav Shuminoski and Aleksandar Risteski, Analysis of the SYN Flood DoS Attack, International Journal of Computer Network and Information Security, 2013
30. Marc Kuhrer, Thomas Hupperich, Christian Rossow and Thorsten Holz, Hell of a Handshake: Abusing TCP for Reflective Amplification DDoS Attacks, 8th USENIX Workshop on Offensive Technologies (WOOT 14), 2014.
31. Christian Rossow, Amplification Hell: Revisiting Network Protocols for DDoS Abuse, Network and Distributed System Security Symposium (NDSS 2014), 2014.
32. Gerald W. Gordon, SYN Cookies, an exploration, GIAC
33. Bo Hang, Ruimin Hu, Wei Shi, An Enhanced SYN Cookie Defence Method for TCP DDoS Attack, Journal of Networks, 2011
34. Mohammad Karami and Damon McCoy, Understanding the Emerging Threat of DDoS-as-a-Service, 6th USENIX Workshop on Large-Scale Exploits and Emergent Threats, 2013
35. Curt Wilson, Attack of the Shuriken 2015: Many Hands, Many Weapons, http://www.arbornetworks.com/blog/asert/attack-of-the-shuriken-2015-many-hands-many-weapons/, 2015.
36. Luca Deri, A Component-based Architecture for Open, Independently Extensible Distributed Systems, PhD Thesis, University of Berne, 1997.
37. Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, Darrell Kindred, Statistical Approaches to DDoS Attack Detection and Response, Proceedings of DARPA Information Survivability Conference and Exposition, 2003.
38. Neha Tewari and Akash Bhardwaj, Flow Statistics Based Detection of Low Rate and High Rate DDoS Attacks, International Journal of Scientific & Engineering Research, Volume 4, Issue 5, May 2013.
39. Luca Deri, Maurizio Martinelli, and Alfredo Cardigliano, Realtime High-Speed Network Traffic Monitoring Using ntopng, Proceedings of LISA 2014, November 2014.
40. Akamai, Akamai's [state of the internet] / DrDoS Attacks, https://www.stateoftheinternet.com/faq-best-practices-drdos-attacks-what-is-dr-dos-reflection.html, 2015.