COFFEE: a Concept based on OpenFlow to Filter and Erase Events of botnet activity at high-speed nodes

Lisa Schehlmann and Harald Baier

da/sec - Biometrics and Internet Security Research Group
Hochschule Darmstadt, Darmstadt, Germany
{lisa.schehlmann, harald.baier}@h-da.de

Abstract: It is a great challenge to tackle the increasing threat of botnets to contemporary networks. The community developed a lot of approaches to detect botnets. Their fundamental idea differs and may be grouped according to the location (e.g., host-based, network-based), data sets (e.g., full network packets, packet header information), and algorithms (e.g., signature based, anomaly based). However, if applied to high-speed networks like nodes of an Internet service provider (ISP) currently proposed methods suffer from two drawbacks. First, the false positive rate is too high to be used in an operational environment. Second, mitigation and reaction is not addressed.

In this paper we introduce COFFEE, our concept of a botnet detection and mitigation framework at large-scale networks. The overall goal of COFFEE is to keep operational costs to a minimum. The detection part of COFFEE comprises two phases: the first one processes the whole traffic to filter candidates of a command-and-control communication using NetFlow-based detection algorithms. In order to decrease the false positive rate, suspected network connections are inspected in more detail in the second phase. The second phase makes use of the concept of Software-Defined Networking (SDN), which is currently deployed in some networks. If the detection yields an alert, SDN again is used to react (e.g., to drop suspect connections).

1 Introduction

A botnet is a network of compromised computers, which aims to launch malicious activities such as distributed denial of service (DDoS) attacks, phishing, click fraud, or sending spam e-mails. Compromised computers communicate with their control servers via a command-and-control channel [FSR09, SSPS13]. Botnets pose a serious threat to the security of our networks and are therefore subject to current research in the network security community. As of today researchers mainly focus on detection and develop different approaches to detect botnets, e.g., by discovering a command-and-control channel [BBR$^+$12, KS07]. However, these approaches mainly suffer from their lack of scalability, i.e., they may not be applied to high-speed networks like the nodes of an Internet service provider (ISP), because of two drawbacks: on the one hand, the data used may not be inspected in real time (e.g., deep packet inspection). On the other hand if scalable detection algorithms like *Disclosure* ([BBR$^+$12]), *BotTrack* ([FWSE11]) or *BotFinder* ([TFVK12]) are used, their false positive rate exceeds the operational limit. A second open issue is reaction and mitigation of detected botnets at ISP nodes.

Our main contribution in this paper is to develop a concept, which solves these issues. We call it COFFEE: a Concept based on OpenFlow to Filter and Erase Events of botnet activity at high-speed nodes. Our approach searches for C&C communications between bots and C&C servers with the aim to prohibit corresponding traffic. The basic paradigm of COFFEE is to keep the operational costs of an ISP to a minimum.

Our starting point of COFFEE is to identify ISP related requirements of a detection and mitigation framework. We derive these demands from two sources: first, the results of a survey [SSAB13] on anomaly detection and mitigation at Internet scale, and second from a project requirements document [pro12] of a joint project with a German ISP. In all we design COFFEE with respect to the following four requirements: (1) The framework has to be a well-documented open source appliance. (2) The costs in an operational environment have to be kept to a minimum, e.g., there is no need for a permanent monitoring and intervention by an operator. (3) The false positive rate is low, e.g., the framework should not produce more than five false positive alerts per hour. (4) The framework must be able to handle 10.000 NetFlow records/s without disturbing noticeable the network traffic.

COFFEE provides both detection of C&C traffic and reaction/mitigation. In order to solve the challenge of processing the huge amount of network traffic at an ISP node, COFFEE works in two phases of detection. The first phase makes use of a highly scalable detection algorithm based on NetFlow data, which can be collected efficiently even at rates of several gigabits/second. Besides this NetFlow provides further conveniences: it is considered to be privacy-friendly and it may identify encrypted C&C traffic. Initially COFFEE is based on features based on time, size, and client access patterns similar to the work of *Disclosure* [BBR$^+$12], *BotTrack* [FWSE11], or *BotFinder* [TFVK12].

COFFEE is innovative as it makes use of the concept of Software-Defined Networking (SDN) for enhancing accuracy and mitigation. SDN is used in the second detection phase to decrease the false positive rate. Supposedly C&C traffic of phase 1 is inspected in more detail to provide more accuracy. In contrast to NetFlow SDN has access to the entire packet header information and the packet's payload. If the SDN detection part yields a suspicious connection, SDN again is used to dynamically reprogram the network device. In case of a detected C&C channel, SDN instructs the network device to drop the packets or to redirect them to an analysis environment.

SDN is attractive for a detection and mitigation framework as it seems to be the upcoming network control standard. Hence COFFEE is only based on data and control devices, which will be used in networks anyway and thus may be used without any additional security appliance.

According to [SSAB13] data interchange is a key barrier in botnet detection at Internet scale. However, detection systems like Disclosure [BBR$^+$12] make use of an external reputation score to decrease the false positive rate, which seems to be critical from an ISP point of view. In contrast to that COFFEE does not rely on external reputation scores due to its SDN based second detection phase. Thus the whole framework may be run autonomously by one entity and no data exchange is needed, which protects user's privacy.

Nevertheless COFFEE is open for a distributed detection and mitigation system.

This paper is organized as follows. Section 2 introduces the background of botnets, NetFlow and SDN/OpenFlow. Section 3 provides an overview of related work in this research area. In section 4 the requirements and architecture of our concept, including the phases of detection and mitigation are introduced. The last section concludes the paper and describes future work.

2 Background

This section introduces fundamentals of botnet and botnet detection (section 2.1), NetFlow (section 2.2) and SDN/OpenFlow (section 2.3), which are required for a better understanding of the following sections.

2.1 Botnet and botnet detection

A botnet is a network of compromised computers, so called bots or zombies, which are controlled by a botmaster to launch malicious activities. Botnets are self-propagating and -organized and remotely controlled via their command and control (C&C) channel [FSR09, SSPS13]. The implementations of botnets are classified by [FSR09] according to their C&C channel infrastructure as IRC-based, HTTP-based, DNS-based and P2P-based.

The approaches to detect botnets are divided by [FSR09] into the following categories: on the one hand honeypots and on the other hand passive network traffic monitoring and analysis. A honeypot is a monitored and isolated part of a network aiming to trap attackers. So attacks could be tracked and useful information about botnets could be collected for a better understanding of them. By the approach of passive network traffic monitoring and analysis, relevant information to detect botnets is extracted from monitored traffic and inspected for botnet behavior. The following methods are classified by [FSR09]: (1) The signature-based detection approach based on patterns of known botnets. Therefore unknown botnets could not be identified. (2) The anomaly-based detection method searches for unusual behavior (e.g., high network load) in the network traffic. Thus, also unknown botnets could be detected. (3) The method of DNS-based detection is aimed to find anomalies in DNS communication, which are sent by bots for initiating a connection to a C&C server. (4) The mining-based approach aims to differentiate C&C traffic from benign network traffic with the help of data-mining techniques.

2.2 NetFlow

NetFlow [Cis07] is a technology originally developed by Cisco Systems to monitor network flows. A network flow is a unidirectional stream of network packets between a source

Field	Description
srcaddr	source IP address
dstaddr	destination IP address
dPkts	number of packets in the flow
dOctets	total number of layer 3 bytes in the packets of the flow
first	sysUptime at start of flow
last	sysUptime at the time the last packet of the flow was received
srcport	TCP/UDP source port number or equivalent
dstport	TCP/UDP destination port number or equivalent
tcp_flags	cumulative OR of TCP flags
prot	IP protocol type (for example, TCP = 6; UDP = 17)
tos	IP type of service (ToS)

Table 1: Relevant fields of a NetFlow version 5 flow record [Cis07]

and a destination host, which is defined by seven key fields: the source and destination IP address, the source and destination port, the layer 3 protocol type, the ToS bytes, and the logical input interface.

NetFlow data is collected at line rate by a network device, e.g., a switch or a router, to build the NetFlow cache. This cache contains the information regarding all active flows. If a flow expires or after a fixed time interval, the NetFlow cache is sent via UDP datagrams to a NetFlow collector, which further process the data (see figure 2). The exported UDP datagrams include a header and several NetFlow records. In the context of our work the NetFlow version 5 flow record fields depicted in table 1 are relevant.

Thus, NetFlow reduces the amount of data to process compared with deep packet inspection (DPI), because it does not contain any payload information. However, the information provided by NetFlow is limited, so detecting botnets probably is a challenge. COFFEE uses a mining-based approach (see section 2.1) to find C&C communication within the NetFlow data.

2.3 SDN and OpenFlow

SDN [Ope12] is a concept developed by the Open Networking Foundation (ONF)[1], which decouples the data plane from the control plane of a network device. The control plane's intelligence is outsourced to an external controller, which provides a more centralized view of the network and is responsible for high-level routing decisions. The data plane is remaining on the network device and still responsible for forwarding network packets. Further SDN provides the ability for a dynamically network programming.

OpenFlow [MAB$^+$08, Ope11] is a protocol to implement the concept of SDN. OpenFlow enabled switches are connected by a *secure channel* to its controller. Packet forwarding decisions of OpenFlow switches are based on flow tables, which contain flow table entries

[1] `https://www.opennetworking.org/`, accessed April 19, 2013.

(see figure 1). The header fields of incoming packets are matched by the switch against the match fields (see table 2) of the existing flow table entries.
The counter fields maintain counters according to e.g., a table, a flow, or a port. A list of all counters is available at page 11 in [Ope11]. So for example received packets or the duration since a flow is installed could be maintained. The controller could request the counters by initiating a *read-state* message.
Instructions are operations, which are executed to a network packet in the case of packet matching. The instructions could affect the packet itself, the packet's action set, or the packet's pipeline processing (i.e., the path of a packet through the different flow tables). The OpenFlow switch specification version 1.1.0 [Ope11] defines the following five types of instructions (see figure 1): The *Apply-Actions* are actions, which are immediately applied to the packet, for example modifying a packet's header field even if a further flow table follows in the pipeline processing. *Clear-Actions* clear a packet's action set immediately. The *Write-Actions* add or overwrite an action within a packet's action set. *Write-Metadata* fills the metadata fields with information. This fields could be filled with additional values to the header fields, against flows could be match e.g., an output port. Finally, the *Goto-Table* instruction forwards the packet to the flow table with the given ID. If there is no *Goto-Action*, the pipeline processing ends with the execution of operations hold in the action set. All defined instructions are described in detail at page 11 in [Ope11].

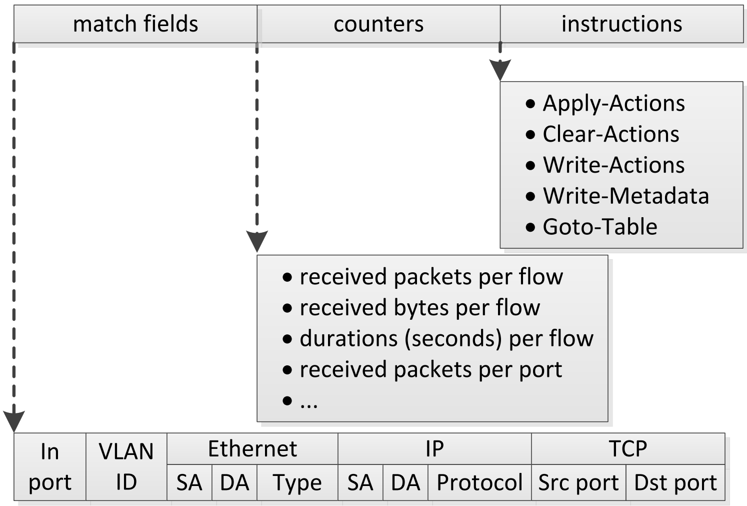

Figure 1: Components of a flow table entry based on [Ope11]

In the case that an incoming packet at the switch does not match against a flow table entry, the packet is sent by the switch to the controller, which decides the further processing. So the controller could install a flow table entry or apply actions to these network packets.

Field	Description	Field	Description
in_port	ingress port	nw_src	IPv4 source address
dl_vlan	IEEE 802.1q virtual LAN tag	nw_dst	IPv4 destination address
dl_vlan_pcp	IEEE 802.1q VLAN priority	nw_proto	IP protocol type
dl_src	Ethernet source address	nw_tos	IP ToS/DSCP field
dl_dst	Ethernet destination address	tp_src	UDP/TCP source port
dl_type	Ethernet protocol type	tp_dst	UDP/TCP destination port

Table 2: Fields to match packets against flow table entries

3 Related work

The botnet detection part of our concept COFFEE is a mining-based approach as discussed in section 2.1. It makes use of both NetFlow and OpenFlow datagrams to detect a botnet C&C channel. In this section we present related work to our approach.

NetFlow-based botnet detection methods are commonly known and discussed e.g., in [AB12, BBR$^+$12, FWSE11, TFVK12].
[AB12] show their vision of a botnet detection framework based on NetFlow data. A key point in [AB12] is the absence of a labeled reference data set for training and testing detection algorithms. For generating such a data set, they implemented a honeypot environment hosted within the network of an ISP for collecting NetFlow data.
[BBR$^+$12] present *Disclosure* as a large-scale, wide-area botnet detection system, which aims at detecting botnet C&C servers. Disclosure extracts three categories of features from collected NetFlow data: *flow sizes*, *client access patterns*, and *temporal behavior*. To reduce the false positive rate, they correlate the results of their framework with the results of external reputation systems.
In [FWSE11] *BotTrack* is introduced, which is an approach for detecting P2P-based botnets using NetFlow data for building a host dependency model. Linkage analysis on base of the *PageRank* [PBMW98] algorithm is combined with clustering. Hosts with a similar behavior can be identified, hence if one host from a class is identified by additional information as a bot, the other ones in this class are bots, too.
The paper [TFVK12] describes the system *BotFinder*. For training the models BotFinder extracts statistical features (e.g., *average time*, *average duration*, *average number of bytes*) from the network traffic of known malware samples. These models are then applied to investigate the network traffic and to detect malware infections.

The focus of COFFEE is on the use of SDN in botnet detection. In contrast to NetFlow, there is only few related work. None of the following papers address the problem of detection C&C channels. [BMP10] introduce an approach to detect DDoS attacks using the protocol OpenFlow [MAB$^+$08] and the OpenFlow controller NOX[2]. They request flow information from flow tables of OpenFlow switches in a periodically interval by the NOX controller and extract a 6-tuple of features from this information (*average packets per flow*, *average bytes per flow*, *average duration per flow*, *percentage of pair-flows*, *growth*

[2] http://www.noxrepo.org/, accessed April 19, 2013.

of single-flows, *growth of different ports*). The features are sent to a Self-Organizing Map (SOM) classifier, which distinguishes the traffic in normal and attack traffic. They denote this approach as lightweight compared to approaches, which extract these features from raw packet data. However, [BMP10] do not compare their approach to NetFlow based ones. In our opinion and experience NetFlow is more lightweight than using OpenFlow, because the communication via the OpenFlow channel to collect information consists of a request and an associated response. NetFlow data, on the other hand, is gathered by the collector and sent in defined intervals to a NetFlow monitor. Additionally all features of [BMP10] are available in NetFlow, too.

[MKK11] introduce a concept of using SDN for anomaly detection. They implement four existing algorithms: (1) TRW-CB [SJB04] to detect scanning worms, (2) Rate-Limiting [TW03, Wil02] to detect infected host by their network behavior, (3) Maximum Entropy Estimation [GMT05] to detect network anomalies comparing current traffic against a baseline distribution, and (4) NETAD [Mah03], a method to classify network traffic as suspicious in the context of SDN. Their hypothesis is the use of programmable home network routers in small office/home office (SOHO) networks as ideal location for detecting network security problems. [MKK11] discuss the problems of a low detection rate and the poor scalability of their algorithms at line rate, however, we encounter this drawback by using our two phase detection approach.

To sum up the approaches of [BMP10] and [MKK11] utilize the concept of SDN to detect network anomalies. However, SDN is only applied to SOHO or to a dedicated attack. Our effort is to implement a botnet detection and mitigation framework in the context of an ISP. As OpenFlow data is more fine grained, but more inefficient to gather compared to NetFlow, we use first a NetFlow based detection to filter suspicious candidates. In the second phase an OpenFlow based detection is used to decrease the false positive rate. Furthermore our approach addresses additionally the aspect of an automated mitigation by OpenFlow.

4 System overview

This section presents COFFEE, our framework to detect and prohibit C&C traffic. COFFEE targets at being deployed at ISP nodes and thus addresses their requirements. We first derive the needs of an ISP detection and mitigation and present the architecture of our framework in section 4.1. Then section 4.2 presents our data collection methodology. Next we describe our two phase detection approach in section 4.3. Finally section 4.4 presents our OpenFlow based reaction strategy.

We point to the fact that we are currently working on the implementation of COFFEE and that we will test it in the near future at a cooperating ISP. Although OpenFlow is currently not as common as NetFlow in the ISP context, we consider it as the future control plane to operate network devices (e.g., [Goo12]).

4.1 Requirements and architecture

COFFEE aims at being used in ISP environments. When designing its architecture we therefore consider the requirements of an ISP, which we derive from results of a survey [SSAB13] and requirements of a joint project with a German ISP [pro12]. Our result is the following list of requirements for a botnet detection and mitigation framework at large-scale networks:

1. The framework has to be a well-documented open source appliance. So it is expandable and adaptable to special needs and does not cause costs for licenses.
2. Operational costs have to be kept to a minimum. This mainly considers the requirement of automation, i.e., there is no need for a permanent monitoring and intervention by an operator.
3. The framework should not produce more than five false positive alerts per hour. We assume that a low false positive rate is much more important than a high detection rate in the ISP context, justified by minimizing operational costs.
4. The framework must be able to handle 10.000 NetFlow records/s without disturbing noticeable the network traffic.

The overall architecture of our botnet detection and reaction framework COFFEE consists of three phases, which will be described in detail in the subsequent sections: *data collection*, *detection* and *reaction* (see figure 2).

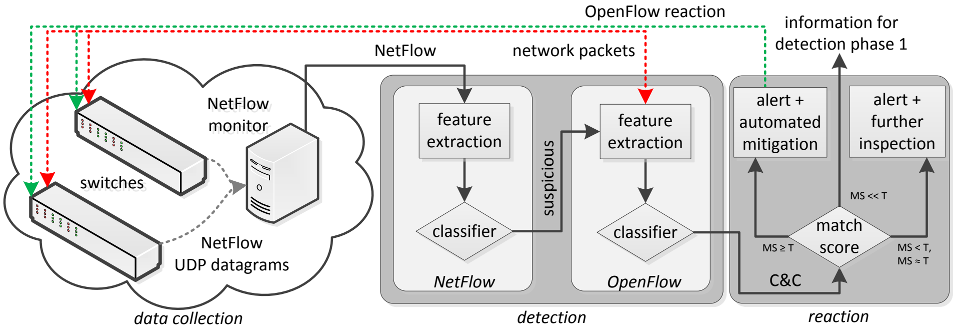

Figure 2: Architecture of COFFEE

4.2 Data collection

As described in section 4.3 COFFEE works in two phases of detection based on two different data sets. Phase 1 uses NetFlow data, which is collected at line rate from switches within an ISP network. After a fixed time interval, the collected information is sent as UDP datagram to a NetFlow monitor, which makes this NetFlow data available for the first detection phase (see figure 2).

Detection phase 2 uses data collected by utilizing OpenFlow. There are two possibilities to gather such data over the OpenFlow *secure channel* (the red dashed lines in figure 2). On the one hand, counters of different flows can be requested by the controller from the switches. Currently, COFFEE does not make use of these counters to identify C&C traffic. On the other hand, entire network packets belonging to a suspicious flow of detection phase 1 are sent to the controller to extract the relevant features.

COFFEE attempts to make use of privacy-preserving features. During detection phase 1 NetFlow data is gathered. NetFlow data contains only packet header related information and no payload, which protects the user's privacy. In phase 2 we extract only features of suspicious packets and not of all packets, which is the first step to protect privacy during phase 2. The second level to protect privacy is to use packet header information per packet. The features which have to be extracted from the packet's payload, are only inspected for known patterns or behavior. Furthermore no payload information is stored anywhere, but rather processed immediately.

4.3 Detection

The detection process splits into two phases to identify C&C traffic. Each phase of detection is based on its own data set and features. The concept of the detection process is shown in figure 3.

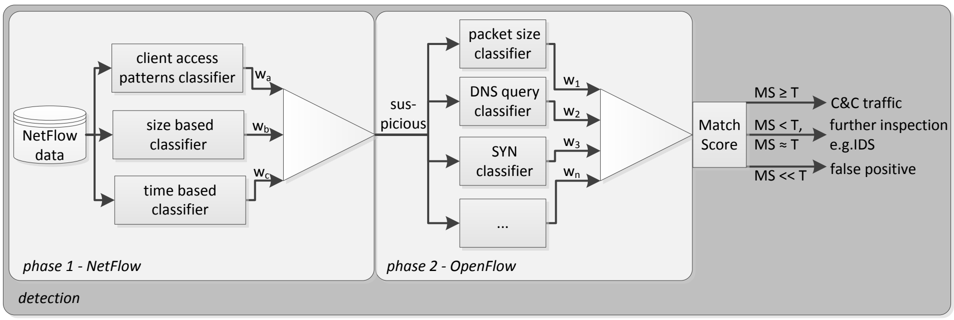

Figure 3: Concept of the detection phases

In phase 1 we use NetFlow data to process the huge amount of data at an ISP node. The NetFlow collector provides the data gathered at the network devices. Similar to the works of [BBR$^+$12, FWSE11, TFVK12], COFFEE extracts features based on time, size, and client access patterns. The extracted features are sent to the classifier, which makes use of a weighted classification fusion to distinguish between benign or suspicious network traffic based on machine learning techniques (again similar to Disclosure, BotTrack, BotFinder).

Only a suspicious flow is handed to the second detection phase, supposedly benign traffic is not inspected further. This significantly will reduce the amount of data for the second phase.

An identifier of the suspicious flows are forwarded from detection phase 1 to the OpenFlow controller responsible for detection phase 2. The identifier includes the *source* and *destination IP*, the *source* and *destination ports*, and finally the *layer 3 protocol type*. The controller writes a table entry for each suspicious flow containing the action *packet_out=CONTROLLER*. Thus, all switches send the whole network packets regarding this flow to the OpenFlow controller. By the default implementation and if the switch has the ability to buffer packets, not the whole packet but only the first 128 byte are sent from the switch to the controller. However, the action *packet_out=CONTROLLER* forwards the whole packet.

During the OpenFlow controller inspects the suspicious flows, all packets according to the flows are still forwarded to their dedicated destination. Thus, usability is ensured because packet forwarding is not affected until the final decision about the suspicious flow of detection phase 2 is known.

The availability of the whole network packet offers two options to inspect: the inspection could be based on a DPI regarding the packet's payload or based on packet header information regarding different layers.

A DPI endangers user's privacy. Additionally it may not be successful due to encrypted payload. We therefore handle it with great care and currently plan to use it only for inspection of DNS queries similar to the work of [VSB08]. Similar to this approach is also the work of [GH07]. They search for unique features, which occur in a bot to commander communication after a host is infected, e.g., unusual or suspicious nicknames.

Inspecting the header fields looks very similar to the NetFlow based approach in detection phase 1. But there is a main difference: By the use of OpenFlow no sampled and aggregated view is performed, but rather every packet is inspected in the order the packet occurs at the controller.

We extract three different features during the second detection phase of COFFEE to detect C&C communication within the suspicious flows:

1. COFFEE correlates the single packet sizes of flows between a suspicious server and several hosts. A C&C server distributes his commands to several bots. The packets containing the same commands should be of a similar size. So if a suspicious server sends packets to several hosts, the single packet sizes of these communications are correlated to find similar sized ones, which indicate a C&C communication.

2. Furthermore COFFEE inspects packets containing a DNS request for known C&C servers. Therefore if a packet of a flow is identified as DNS request, the queries are extracted and further processed, e.g., scanned for unusual server URLs (i.e., URLs not accessible by humans).

3. The single packet size of a SYN packet according to a suspicious flow is a further feature. The size of such packets varies according to the host's operating system,

which generates the packet, e.g. 48 bytes for Windows 2000 or 60 bytes for Linux 2.4[3]. If a SYN packet belongs to a Linux or Mac OS operating system, the probability that the source host is a bot is very low.

Detection phase 2 outputs a match score, which we denote as MS. Each OpenFlow classifier is considered with respect to its weight: the packet size classifier is weighted with w_1, the DNS query classifier with w_2, the SYN classifier with w_3. The OpenFlow detection may be extended by further classifiers, we simply have to ensure $\sum_{i=1}^{n} w_i = 1$. The final decision of COFFEE depends as usual on a threshold T. We distinguish the following three cases (see also figure 2):

1. If $MS \geq T$ the flow is identified as C&C traffic. An alert is triggered, and an automated mitigation is initiated as described in section 4.4.
2. If the match score is marginally below the threshold, i.e., $MS < T$ and $MS \approx T$, then again an alert is released, however, the flow needs further inspection, e.g., through an Intrusion Detection System or through a manual inspection by the ISP operator.
3. If the match score is significantly below the threshold, i.e., $MS \ll T$, then the flow is identified as benign and thus a false positive with respect to detection phase 1. The controller overwrites the corresponding flow table entry so that further flows are not forwarded to the controller again, but rather process as normal traffic. Also an information about the false positive identified traffic regarding the flow is given to the detection engine of detection phase 1.

4.4 Reaction

Traditional instances to mitigate network-based attacks are firewalls, which allow or deny network packets based on pre-defined rules. However, firewalls are not able to learn new rules by themselves or change their rules to mitigate an attack. This distinguishes our framework from a traditional firewall, because COFFEE does not use pre-defined rules to mitigate C&C communication, but rather changes packet forwarding rules automatically with the help of OpenFlow if a C&C communication is detected. So our system acts more like an Intrusion Prevention System (IPS), which also dynamically reacts to a detected attack. We point to the convenience of COFFEE that it does not make use of any additional appliance besides the upcoming network control devices (with the possible exception of a final inspection by an additional IDS).

If potential C&C traffic is detected during the NetFlow based detection phase, only an information is given, which could be recognized by an operator on demand. But still no reaction or mitigation processes are initiated.

By detecting a C&C channel during detection phase 2 there are two levels of escalation depending on the relation of the match score to the threshold: (1) The default level is

[3] `http://www.ouah.org/incosfingerp.htm`, accessed May 2, 2013.

In port	vlan ID	Eth src	Eth dst	Eth type	IP src	IP dst	IP proto	TCP sport	TCP dport	action
*	*	*	*	*	2.3.4.5	5.4.3.2	6	80	90	drop

In port	vlan ID	Eth src	Eth dst	Eth type	IP src	IP dst	IP proto	TCP sport	TCP dport	action
*	*	*	*	*	5.4.3.2	2.3.4.5	6	90	80	drop

Table 3: Flow table entries to drop packets

...	IP src	IP dst	IP proto	TCP sport	TCP dport	action
...	2.3.4.5	5.4.3.2	6	80	90	modify: IP dst=1.1.1.1, dport=99

Table 4: Snippet of a flow table entry to redirect packets

an automated mitigation by changing the forwarding rules of OpenFlow. (2) The second level is to change the OpenFlow forwarding rules after an additional inspection, e.g., by a human operator. However, this should only propose an additional possibility to an operator by demand, because COFFEE still aims to reduce operational costs.

A mitigation by COFFEE results in one of the two approaches using OpenFlow instructions: malicious network packets belonging to a C&C communication can be dropped or redirected to an analysis environment.

The first reaction strategy is to drop all packets, which belong to a C&C communication. In this case the controller adds two flow table entries (see table 3). The first entry describes the direction C&C server to bot, the second one the opposite direction from bot to C&C server. The entry contains the match fields and an action as explained in section 2.3. We fill the match fields with the IP protocol, the source and destination IP as well as the source and destination port. All other fields are wildcarded by a * (i.e., they are ignored by packet matching). As action the instruction *drop* is set, to drop all packets regarding this flow. To reduce the number of flow table entries, there is also the possibility to set only a flow table entry containing the source IP as well as the source port of a detected C&C server. So all packets caused by a detected C&C server and addressed to the associated bots are dropped.

The second case is to redirect packets of the C&C channel to an analysis environment, for example a honeypot to gather more information about the packets regarding this communication or to an inspection instance. In this case a table entry is added containing an action to change the destination IP address and the port to the address and port of the honeypot, respectively (see figure 4).

5 Conclusion and future work

This paper introduces COFFEE, our concept of a botnet detection and mitigation framework, which uses a two phase model to detect C&C communication. We achieve to reach an operational manageable false positive rate by the two phases of detection, because we assume that a low false positive rate is more important than a high detection rate to detect C&C communication at high-speed nodes. In the first phase of detection, we use NetFlow for a first efficient filtering regarding the huge amount of traffic. In the second phase we use OpenFlow for a more detailed inspection of the filtered and as suspicious labeled traffic to eliminate false positives by using as features e.g., the single packet size of suspicious network packets.

We propose the use of OpenFlow, because it is an emergent technology, which will applied in contemporary networks increasingly. OpenFlow provides beside our approach a lot of possibilities to manage networks, such as e.g., QoS settings. Hence, there is only the need for one approach to program and manage a network in different ways, which saves costs and operational time. Our detection method is not acting in real-time, because network traffic is not delayed until the inspection is completed to ensure usability.

Still a remaining problem is the unavailability of a disposable labeled reference data set for training the models. Such a data set is required, because if every system uses its own data sets, the accuracy and false positive/negative rates of different botnet detection algorithms and systems cannot be compared. Gathering such data sets in a real ISP environment is also a problem because of the user's privacy and due to that OpenFlow is not as common as NetFlow.

The next steps are a set up of COFFEE in an ISP environment and to evaluate its runtime and detection performance.

Acknowledgment. This work was partly supported by the Hesse government under grant number 306/11-51 (NetFlowBot) and by CASED.

References

[AB12] Sebastian Abt and Harald Baier. Towards Efficient and Privacy-Preserving Network-Based Botnet Detection Using Netflow Data. In *Proceedings of 9th International Network Conference*, INC '12, Port Elizabeth, South Africa, July 2012.

[BBR+12] Leyla Bilge, Davide Balzarotti, William Robertson, Engin Kirda, and Christopher Kruegel. DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 129–138, New York, NY, USA, 2012. ACM.

[BMP10] Rodrigo Braga, Edjard Mota, and Alexandre Passito. Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow. In *Proceedings of the 2010 IEEE 35th Confer-*

ence on Local Computer Networks, LCN '10, pages 408–415, Washington, DC, USA, 2010. IEEE Computer Society.

[Cis07] Cisco Systems, Inc. NetFlow Services Solutions Guide. `http://www.cisco.com/en/US/docs/ios/solutions_docs/netflow/nfwhite.html`, January 2007.

[FSR09] Maryam Feily, Alireza Shahrestani, and Sureswaran Ramadass. A Survey of Botnet and Botnet Detection. In *Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies*, SECURWARE '09, pages 268–273, Washington, DC, USA, 2009. IEEE Computer Society.

[FWSE11] Jérôme François, Shaonan Wang, Radu State, and Thomas Engel. BotTrack: Tracking Botnets using NetFlow and PageRank. In *10th International IFIP TC 6 Networking Conference - Part I*, NETWORKING '11, pages 1–14, Valencia, Spain, 2011. Springer-Verlag.

[GH07] Jan Goebel and Thorsten Holz. Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots '07, pages 8–8, Berkeley, CA, USA, 2007. USENIX Association.

[GMT05] Yu Gu, Andrew McCallum, and Don Towsley. Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pages 32–32, Berkeley, CA, USA, 2005. USENIX Association.

[Goo12] Google Inc. Inter-Datacenter WAN with centralized TE using SDN and OpenFlow. case study, Google Inc., 2012. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/customer-case-studies/cs-googlesdn.pdf`.

[KS07] Satoshi Kondo and Naoshi Sato. Botnet traffic detection techniques by C&C session classification using SVM. In *Proceedings of the Security 2nd international conference on Advances in information and computer security*, IWSEC'07, pages 91–104, Berlin, Heidelberg, 2007. Springer-Verlag.

[MAB$^+$08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[Mah03] Matthew V. Mahoney. Network Traffic Anomaly Detection Based on Packet Bytes. In *Proceedings of the 2003 ACM symposium on Applied computing*, SAC '03, pages 346–350, New York, NY, USA, 2003. ACM.

[MKK11] Syed Akbar Mehdi, Junaid Khalid, and Syed Ali Khayam. Revisiting Traffic Anomaly Detection using Software Defined Networking. In *Proceedings of the 14th international conference on Recent Advances in Intrusion Detection*, RAID '11, pages 161–180, Berlin, Heidelberg, 2011. Springer-Verlag.

[Ope11] Open Networking Foundation. OpenFlow Switch Specification 1.1.0, Februar 2011.

[Ope12] Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. *White Paper*, April 2012.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. In *Proceedings of the 7th International World Wide Web Conference*, WWW '98, pages 161–172, Brisbane, Australia, 1998.

[pro12] project consortium of NetFlowBot. Anforderungsbeschreibung LOEWE3-Projekt NetFlowBot. requirements document, project consortium of NetFlowBot, 2012.

[SJB04] Stuart E. Schechter, Jaeyeon Jung, and Arthur W. Berger. Fast Detection of Scanning Worm Infections. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, RAID '04, pages 59–81, 2004.

[SSAB13] Jessica Steinberger, Lisa Schehlmann, Sebastian Abt, and Harald Baier. Anomaly detection and mitigation at Internet scale: A survey. In *Proceedings of the 7th International Conference on Autonomous Infrastructure, Management and Security*, AIMS '13, Barcelona, Spain, 2013. Springer.

[SSPS13] Sérgio S. C. Silva, Rodrigo M. P. Silva, Raquel C. G. Pinto, and Ronaldo M. Salles. Botnets: A survey. *Computer Networks*, 57(2):378–403, 2013.

[TFVK12] Florian Tegeler, Xiaoming Fu, Giovanni Vigna, and Christopher Kruegel. BotFinder: Finding Bots in Network Traffic Without Deep Packet Inspection. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, CoNEXT '12, pages 349–360, New York, NY, USA, 2012. ACM.

[TW03] Jamie Twycross and Matthew M. Williamson. Implementing and testing a virus throttle. In *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*, SSYM '03, Berkeley, CA, USA, 2003. USENIX Association.

[VSB08] Ricardo Villamarin-Salomon and Jos Carlos Brustoloni. Identifying Botnets Using Anomaly Detection Techniques Applied to DNS Traffic. In *Consumer Communications and Networking Conference, 2008. CCNC '08. 5th IEEE*, pages 476–481, 2008.

[Wil02] Matthew M. Williamson. Throttling Viruses: Restricting propagation to defeat malicious mobile code. In *Proceedings of the 18th Annual Computer Security Applications Conference*, ACSAC '02, pages 61–68, Washington, DC, USA, 2002. IEEE Computer Society.