Collaborative Approach for Inter-domain Botnet Detection in Large-scale Networks

Hachem Guerid*†, Karel Mittig*, Ahmed Serhrouchni†
*Orange Labs, Caen, France
{hachem.guerid, karel.mittig}@orange.com
†Telecom ParisTech, Paris, France
{hguerid, ahmed}telecom-paristech.fr

***Abstract*—The members of almost all botnets are distributed between several networks. Such distribution hardens their detection as the centralized approaches require to centralize network data for their analysis, which is indeed not possible in regard to the legacy and business constraints applied to network operators.**

In this paper, we propose a collaborative and inter-domain botnet detection system which conciliates the requirements of privacy and business preservation, while enabling real-time analysis for large scale networks. The different probes of our collaborative detection system exchange anonymised information in order to synchronize the network analysis of the members of botnets and to identify the malicious servers controlling them.

We evaluated our system using anonymised traffic captured on an operator's network, and the results showed an improvement of 31% of malicious servers detected resulting from the collaboration, and this without significant performance impact and bandwidth overhead (respectively 4% and 11kb/s).

***Keywords*-Botnet detection, Collaborative detection, Domain-flux botnets, Bloom filters, Inter-domain detection**

I. Introduction

A botnet is a network of infected hosts or bots controlled remotely by attackers [1] usually through remote servers called command and control servers (C&C). Botnets are used to launch different types of network attacks [2] such as: distributed denial of service attacks (DDoS), spams, and data theft.

The members of a botnet contact their C&C servers in order to receive commands to launch their attacks or to report about ongoing ones. These infected hosts use common protocols [3] such as DNS, HTTP or IRC in order to hide the communication with their servers from network administrators.

The members of a botnet are usually distributed between different countries and administrative domains [4]. This distribution harden their detection because each network operator has only access to a part of the malicious communications of the whole botnet. The largely spread botnets are therefore very difficult to detect.

The current behavioural network botnet detection systems are centralised [5], [6], which means that they collect and analyse centrally the network data in order to detect the C&C servers of the botnets. As the network operators cannot share the network data of their users with other operators, the previous approaches can only analyse the network traffic of a single network operator.

This lack of collaboration implies that widely spread botnets are generally not detected as their activity fall beyond the detection thresholds. A collaborative detection over several network domains would therefore allow the correlation of much more information, thus improving the time and accuracy of the detection.

However, such collaboration between administrative domains must comply with several constrains. The exchanged information between the different networks involved in the detection should be anonymised in order to conciliate the requirements of privacy and business preservation. User's privacy requires that information such as the identifiers or the payloads of the infected hosts must not be shared with the other operators participating in the detection process, while business preservation additionally implies that information like the number of infected hosts or network topology must be also concealed.

As the regulation in most European countries [7] restricts the storage of the network traffic of the internet users for commercial use, the detection process must be performed in real-time. This implies that a collaborative detection system must be able to process in real-time the traffic of large network operators. Such networks contain at each time several million active users.

In this paper, we propose a fully distributed collaborative and inter-domain botnet detection system for large scale networks. Our collaborative detection system analyses the network traffic of different networks in order to detect the C&C servers of botnets. Our system detects in real-time the C&C servers of botnets which are scattered between different administrative networks while preserving the network operator's privacy and business.

Our collaborative botnet detection system groups the botnets' members that are distributed between different networks into communities. For each community, our system detects their C&C servers by identifying their convergence points.

The members of the same botnet can converge to the same popular web sites. This convergence is either the result of legitimate actions, or the result of a feature added by the creators of the botnet [8]. such behaviour is added into mal-

COLLABORATECOM 2013, October 20-23, Austin, United States

DOI 10.4108/icst.collaboratecom.2013.254051

ware in order to perform network attacks or to mislead the security researchers. In our collaborative system, we filter dynamically the popular web sites using the information of the different communities.

In order to evaluate our collaborative botnet detection system, we implemented the detection system of a type of botnets called domain-flux [9]. We set up our collaborative detection system between two network traces captured from a large scale network. We demonstrate that the collaborative detection botnets detects more malicious servers while generating low network and processing overhead.

The main contributions of our work can be summarised as follows:

- Proposition of a fully distributed collaborative and an inter-domain botnet detection system.
- Preservation of the privacy and the business of the domain during the collaboration by exchanging anonymised information.
- Real-time processing of the network traffic of large scale network.

II. Related Work

Several anomaly based approaches have been proposed to detect botnets according to their network activity. Botsnifer [10] detects the C&C server of a group of infected users inside a small network while Karasaridis A. et al. [6] proposed an approach based on common servers of a group of identified infected users in a wide-scale network.

Since the DNS protocol [11] is used by most botnets, several DNS-based botnet detection approaches have been proposed. Chois H. et al. [12] proposed an approach that detects the change of the domain names of the C&C servers of a botnet. The DNS errors have been used by several approaches [13], [14], [15] to identify the members of a domain-flux botnet [9].

Unlike these centralised approaches, our collaborative system is distributed between different networks. Each network analyses the network traffic of its own hosts and exchanges anonymous information with the other networks. The actors of our system collaborate in order to detect the servers of botnets scattered between their networks.

Wang H. et al. [16] proposed a collaborative architecture in order to detect botnets. In their architecture, different administrative domains share information with different granularity in order to coordinate the detection of botnets. Unlike the previous architecture, the different administrative domains of our system anonymise the exchanged information to comply with the constraints of privacy and business preservation of inter-operators communication.

Locaster et al. [17] proposed with Worminator a collaborative P2P intrusion detection, where the different actors share Bloom filters [18] storing "'watch-lists"' or lists of suspected IP addresses of each actor. Our collaborative botnet detection system doesn't exchange watch-lists of suspected IP addresses. It uses Bloom filters in order to exchange an aggregated network activity of several members of a botnet distributed between different administrative domains.

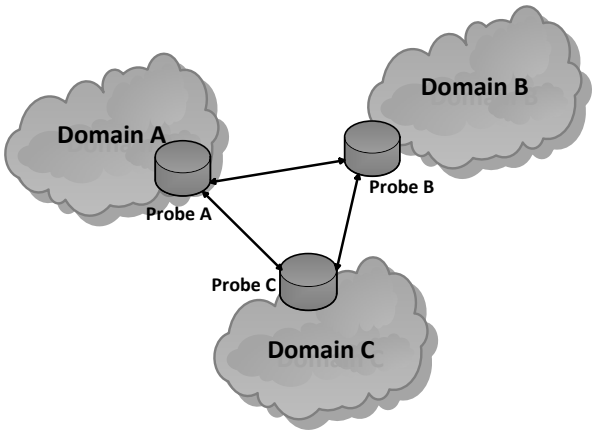

Figure 1. Overview of a collaborative botnet detection system between 3 domains

Unlike centralised collaborative intrusion detection systems [19] [20], our detection system is fully distributed. It doesn't rely on a central server that receives and correlates alerts from multiple detection nodes. In such architecture, the server constitutes a single point of failure for the system. This centralised architecture is not acceptable for distinct operators which have legal and business constraints.

III. Architecture of our collaborative detection system

In our collaborative detection system, each probe is associated with a network, see figure 1. The probes are interconnected with each other in order to distribute and to coordinate the analysis of the traffic of the different networks. Each probe correlates the network traffic of its network and the anonymised data received from the other networks. The result of the detection is distributed between the different actors of the collaboration.

A probe is constituted of two layers, a communities construction layer and a C&C server detection layer. The communities construction layer detects the infected hosts and groups the members of the same botnet into communities. These communities are sent to the C&C server detection layer which identifies their associated malicious servers.

The probes of the system exchange anonymised information in order to compare between the network activities of their infected hosts and to coordinate the construction of the communities whose members are distributed among different networks. The probes coordinate the detection of the C&C servers of these communities.

The detection of the C&C servers for each community is coordinated by an elected probe. The election process

ensures that the computation and the network overhead needed for the coordination of the analysis of the different communities is divided between the different probes of the network.

In our collaborative system, the C&C detection layer is common to all types of botnets, unlike the communities construction layer that depends on the type of botnet or its generated attacks. In this paper we validated our system with the collaborative detection of domain-flux botnets.

We detail the two layers in the following sections.

IV. Communities Construction Layer

The probes of our collaborative system detect members of the same botnets and groups them into communities. This detection is based on the participation in the same ongoing attacks or in the similarity of their abnormal network traffic.

As communities can be distributed between several domains, different probes monitor the network traffic of the same community. Thus, the communities construction layer associates a unique identifier among the different probes to each constructed community in order to avoid different communities sharing the same identifier which can distort the detection. This identifier is used by the C&C servers detection layer to coordinate different communities inside a probe.

In the case of the detection of a community of the same botnet participating in a distributed denial of service attack (DDoS), the probes assign independently the same identifier to this community. This identifier is computed from the IP addresses or the domain names of the victims which are accessible by the different probes.

In order to facilitate the coordination of the detection and to decrease the network overhead, we assign each community to a single probe. This probe correlates the network traffic of the members of the same community in order to detect their C&C server. Our detection system requires that the network analysed by the elected probe contains some members of the community. The network traffic of these members is analysed in order to find convergence points corresponding to C&C servers.

On our system, we distribute the management of the communities between the different probes. This distribution avoids single points of failure of the system and balances the computation and the network overhead needed for the coordination. The independence of the network operators consolidate the fair distribution of the monitored communities.

V. Command and Control Servers Detection Layer

Within this layer, each probe receives information about the communities identified in the communities construction layer. It contains the identifiers of the members of the community of the network monitored by the probes along with the identifier of the elected probe. The elected probe coordinates the network analysis of the members of the community in order to detect the associated command and control servers (C&C).

Virtually the whole network traffic of a user can only be accessible by its network operator. Since the network operators can't share the network traffic of their users with other operators, each probe analyses the traffic of the infected hosts of the monitored network in order to detect the malicious servers.

The command and control servers detection layer is divided into three phases: (1) the traffic analysis of the infected users, (2) the coordination of the probes, and (3) the detection of the malicious servers. The different phases of detection are depicted in figure 2.

We use during our analysis probabilistic data structures in order to represent the network activity of the infected users. These probabilistic data structures are Bloom filters [18] which are binary vectors of a predefined length associated with k hash functions [21]. The Bloom filters represent elements of a set in a fixed memory space which is the size of the associated vector.

The Bloom filters don't generate false negatives, which means that a membership test of a represented element is always positive. On the other hand, they generate false positives with a certain probability. This probability depends on the size of the vector, the number of elements in the set, and the number of hash functions.

A. Traffic analysis of the infected users

In order to preserve the privacy of the users and to optimise the memory usage of the probes, we associate a Bloom filter with each infected user. The Bloom filters are used to represent the network activity of the members of an identified community. We use the same parameters for all the Bloom filters of the system e.g. the vector size, the type and number of hash functions in order to be able to compare them afterwards because if two similar sets are represented by Bloom filters with the same parameters, they produce similar Bloom filters.

We reset periodically the vectors of the Bloom filters associated with the infected hosts in order to represent only their latest network activity. The representation of the latest network activity preserves the privacy of the users and decreases the number of generated false positives related to popular web sites.

We filter from the analysis proxies i.e. several users using the same public IP address because their network activities converge in multiple legitimate points and thus generate several false positives. The proxies are the multiplexing of the network traffic of several users sharing the same public IP address.

On our system, we use Bloom filters with a small vector size in order to optimise the memory usage of the system and

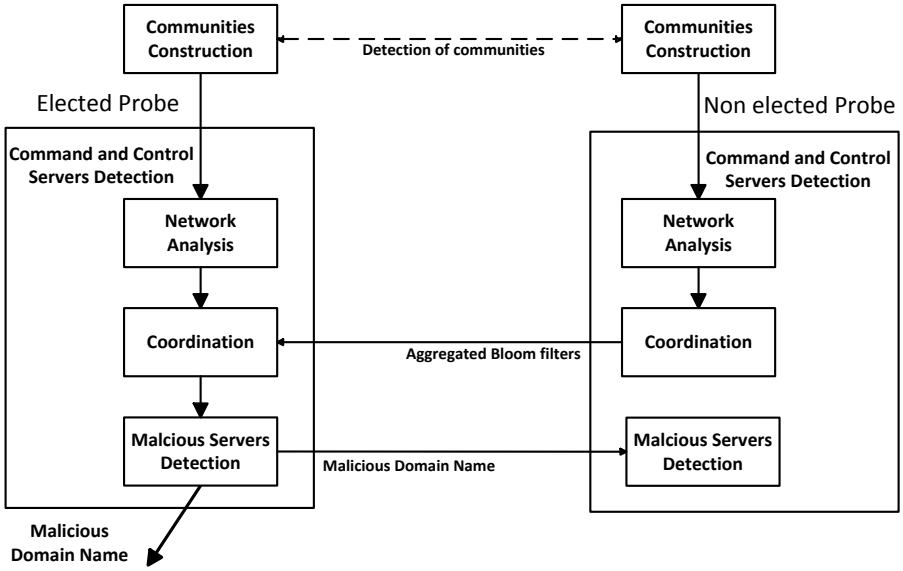

Figure 2. A collaborative botnet detection between two probes

to decrease the time needed for their correlation. A size of about one thousand bits is enough to represent the network activity of users regarding the period of time considered until resetting the filters. Moreover, the Bloom filters with a small size allow the identification of proxies because their Bloom filters are filled more quickly.

B. Coordination of the analysis

During this phase, each probe coordinates the analysis of the members of the constructed communities. For each community, the non elected probes construct periodically an aggregated Bloom filter in order to represent the common freshest network activity of the members of the same community that belongs to the monitored network.

The aggregated Bloom filter of a community is computed according to the Bloom filters of its members, see algorithm 1. In the aggregated vector, we set an element of index i to 1 if the rate of the elements of the same index in the vectors of the members of the community exceeds a threshold. Otherwise, we set the element to 0. For example, if the elements of index i in all the vectors of the Bloom filter are 1, the element of index i in the aggregated vector will be set to 1.

This aggregated Bloom filter represents the convergence points of the network activity of the members of the community monitored by the probe. The small binary Bloom filters preserve the privacy of the users because it is nearly impossible to recover the elements that filled it. The aggregation enhances the privacy preservation of the users because it represents the network traffic of several users.

Each probe participating in the detection sends aggregated vector to the elected probe of the community according to a period τ. In order to minimise the network and the computational overhead, the probes exchange an aggregated vector only when it has been modified during the last period.

Algorithm 1 Aggregated vectors construction

```
1:  V_A ← 0                                  ▷ Initialise the aggregated vector
2:  for i ← 1, N do                          ▷ N: Size of the vector
3:      nbr ← 0
4:      for j ← 1, n do                      ▷ n: Size of the community
5:          if V_j[i] = 1 then
6:              nbr ← nbr + 1
7:          end if
8:      end for
9:      if nbr ≥ α * n then                  ▷ α: infection threshold
10:         V_A[i] ← 1
11:     else
12:         V_A[i] ← 0
13:     end if
14: end for
```

C. Detection of malicious servers

This phase is executed only by the elected probe of each community. The elected probe analyses the Bloom filters of the members of the community of its monitored network and the aggregated Bloom filters received from the other probes. It correlates their network activity in order to identify their command and control servers.

When a member of a monitored community has a connection to a remote server, we check if the other members of the community connected recently to this server. We perform a membership test of the server in the Bloom filters of the

members of the community monitored by the probe and the received aggregated Bloom filters. If the membership test is positive on most Bloom filters, most members of the community connected recently to the server. This imply that the server is highly popular or is the C&C server of the members of the community.

The difference between the malicious and the popular servers is that the popular servers are also popular within all the communities of the system i.e. also requested by members of other botnets than the one that triggered its detection. On the other hand, the C&C servers of a botnet are much more popular within the communities that contain members of the botnet than within the other communities.

Each probe is responsible for several communities, therefore it represents at each time the network activity of different communities. It perform a membership test of the detected server on the Bloom filters of these communities in order to filter the popular servers. The detected C&C servers are shared with the other probes monitoring the community.

VI. The domain-flux botnet detection use-case

In order to validate our collaborative botnet detection system, we implemented and evaluated our system for botnets of domain-flux type. Different network operators can collaborate in the detection of botnets because it doesn't reveal the internal information of their networks and it preserves the privacy of its users.

In a domain-flux botnet, the infected hosts generate periodically a list of domain names. The members of these botnets request the generated domain names until they reach their C&C servers which is reserved by the botnet operator beforehand. This technique allows botnets to bypass static black-lists because network administrators ignore which domain names will be generated and reserved before it occurs. Some well-known botnets used this technique, such as Conficker [22], Zeus [23] and PushDo [24].

The main characteristic of these botnets is that the infected hosts generate an abnormal amount of requests toward non existent domain names. Thus, the infected hosts receive an abnormal amount of NXDomain replies from the DNS servers. The members of the same domain-flux botnet generate requests to a similar set of non existent domain names.

In order to detect the domain-flux botnets, the probes of our collaborative system are located between the users and the DNS servers. Such position allows the differentiation between the hosts generating requests to the DNS server. The probes analyse in real-time the DNS traffic of the hosts of the network in order to detect the C&C servers.

Since most DNS traffic between the members of the network and the DNS servers stay inside the network, a probe has only access to the DNS traffic of the monitored network. Thus, the collaboration between different networks allows the system to have access to more information comparing to a standalone detection of each probe. We described the standalone domain-flux botnet detection system in detail in [15].

A. Communities Construction of a domain-flux botnet

This layer detects and groups the members of the same domain-flux botnets distributed in different networks into communities. The probes of these networks exchange anonymous data in order to compare between the network traffic of their users.

Within this layer, the probes analyse the NXDomain replies of the users in order to identify the infected users and to construct the communities of users requesting the same set of non existent domain names.

The construction of communities is performed in two phases: the identification of the suspicious users and the construction of suspected communities.

1) Suspicious users identification: The detection of the suspected communities is executed independently in each probe. It identifies the users generating an abnormal number of requests to non existent domain names which correspond to a domain-flux behaviour.

In order to preserve the privacy of the users and to speed up the analysis, we associate a small Bloom filter of less than a thousand bits with each user. This Bloom filter will represent the freshest requested non existent domain names extracted from the received NXDomain answers.

Since we compare these Bloom filters later in our analysis, we use the same parameters in the Bloom filters on all the probes of the system in the communities construction layer. These parameters are the size of the vector, the type and the number of hash functions.

This module filters the proxies i.e. multiple users sharing the same public IP address, because our system will associate the requests of these users to the same Bloom filter which will be filled quickly. We filter from the analysis these Bloom filters because they build communities that generate several false positives.

This module identifies the suspicious users using the filling rate of their associated Bloom filters. The Bloom filters of suspicious users have a filling rate that exceeds non infected users' because the members of a domain-flux botnet generate a high number of requests toward non existent domain names. We use this threshold in order to filter the users making legitimate errors.

We reset periodically the vectors of the Bloom filters associated with the users of our system because the binary Bloom filters used on our system don't allow the deletion of old elements. Thus, we represent only the freshest requested non existential domain names.

2) Communities Construction: The communities construction phase groups the suspicious users received from the suspicious users identification module into communities. The network traffic of members of botnets distributed be-

tween different networks is correlated in order to construct communities distributed between different networks.

In a domain flux botnet, the infected users of the same botnet generate requests to the same set of non existent domain names. Since the Bloom filters represent the freshest requested non-existent domain names, we only need to compare between these Bloom filters in order to identify the users that requested the same set of domain names.

We compare periodically the Bloom filters of the suspicious hosts using a similarity ratio which we derived from the Hamming distance. Our similarity ratio counts the number of ones of the result of the binary AND operation between two elements where the Hamming distance uses a binary XOR operation.

Similarity ratio between two strings a and b of the same length.

$$similarity(a,b) = \frac{weight(XOR(a,b))}{max(weight(a),weight(b))}$$

We fix the minimum size of a suspicious community to a given threshold in order to minimise the false positives induced by the Bloom filters. The larger the community is, the higher the probability is that its members belong to the same domain-flux botnet. On the other hand, a small community can be the result of users requesting the same misspelled domain names.

If the size of a group of similar hosts inside the same network does not reach the threshold of a creation of a community, the probe creates a sub-community. An aggregated Bloom filter is created in order to represent the similar requested non existent domain names of the members of a sub-community. This aggregated Bloom filter is used to identify similar sub-communities whose members belong to the same domain-flux botnet.

The probes exchange information about the created sub-communities in order to construct a community. They exchange the identifier of the sub-community, the identifier of the probes, and the aggregated Bloom filter of its members.

The collaborative construction of a community is a four step operation, see figure 3:

- The probes receives and sends broadcasts to the other probes about constructed sub-communities.
- A probe becomes the elected probe for the community and sends a notification to the other probes. For domain-flux botnets, the process of electing a probe amongst others simply relies on the community identification time, with the first probe sending a notification about an identified community becoming the elected one.
- The other probes reply to the notification by an acknowledgement.
- The elected probe sends a confirmation to the other probes and creates the community.

When a sub-community is created, its aggregated Bloom filter is compared with the aggregated Bloom filters representing sub-communities received from the other probes. If it finds similar Bloom filters, the probe sends notifications to their probes in order to create a community. The notification contains the identifiers of the sub-communities. Otherwise, the probe sends the information about the created sub-community to the other probes.

The probes keep information for a certain duration about their created sub-communities i.e. their identifiers, aggregated Bloom filter, and associated members. When a probe receives a notification about the creation of a community, it replies by an acknowledgement if it has data about its sub-community. Afterwards it waits for a confirmation of the creation the community.

The probe that sent the notification for the creation of a community to the other probes is the elected probe of the community. If it receives enough acknowledgements for the creation of a community, it sends the identifier of the community to the C&C detection layer along with the identifiers of the sub-communities acknowledged by other probes. It sends a confirmation of the creation of a community to the probes that sent an acknowledgement.

At this stage, we considered that all the collaborating probes can be trusted, and we did not introduced the possibility of a probe to be compromised. The consequences of such assumption and how it can be prevented will require additional features that will be considered in a future work.

When the probe receives a confirmation of the creation of a community, it sends the information about the community to the C&C detection. It sends the identifiers of the members of the sub-community that are monitored by the probe, the identifier of the community, and the identifier of the elected probe. The identifier of the constructed community is assigned by the elected probe.

B. C&C Servers Detection

The network traffic of the members of a botnet converge to their command and control server. In order to identify this convergence point, we analyse the successful DNS answers (NOERROR) of the members of the botnet. The convergence point correspond to the domain name of the command and control server of the members of the community.

The Bloom filter associated with the user represent its freshest requested existent domain names. We use the same parameters for the different Bloom filters of the system in order to aggregate them later.

The successful DNS traffic of the members of a community monitored by a non-elected probe is aggregated in order to highlight the convergence points. Periodically, the Bloom filters of these members are aggregated in order to create a Bloom filter representing the similar successful DNS requests. The aggregated Bloom filter is sent periodically to the elected probe of the community.

When a member of a suspicious community monitored by an elected probe receives a successful DNS reply, it

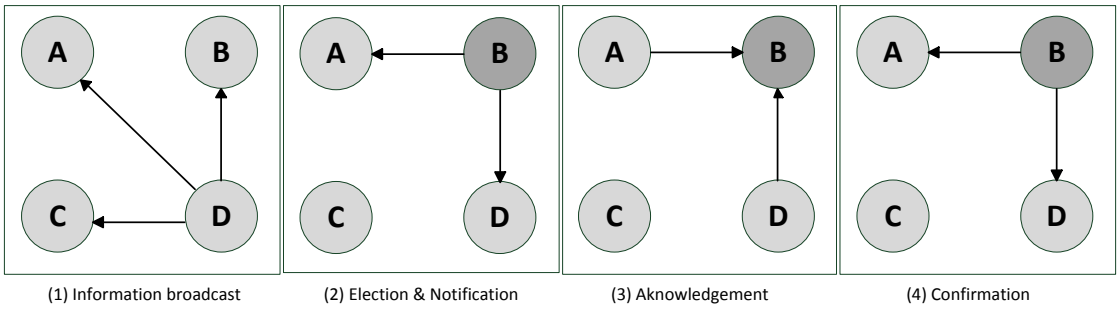

Figure 3. Four steps process for collaborative construction of a community

adds the domain name in the Bloom filter associated with the member. The probe performs a membership test of the domain name on the Bloom filters associated with the members of the community.

If the ratio of the members that requested the domain name exceeds the infection threshold, the probe checks the latest aggregated Bloom filters received from other probes. If the membership test of the domain name on the aggregated Bloom filters is successful, it means that the domain name has been requested by the users of other networks that constructed the aggregated Bloom filters. If the domain names has been requested by most members of the community, it is the domain name of a C&C server or a highly popular website.

Unlike the C&C servers domain names, the popular domain names are requested by most communities monitored by the probe. Thus, we filter the domain names that are requested by at least one member of most communities. The elected probe sends the detected domain name of C&C server to the other probes.

VII. Evaluation

In order to validate our collaborative botnet detection, we implemented the collaborative domain-flux botnet detection use case between two probes. The probes analysed a DNS traffic of 12 hours from a large operator captured in 2009 between the DNS server and the users. It contains the DNS requests and replies of the users. The capture was anonymised such as each IP address was converted into a unique and non reversible identifier.

We divided the capture in several parts according to the identifiers of the users of the network. Therefore, the network traffic of a user will be analysed by a single probe.

In order to allow the network administrator to monitor the exchanged information between different administrative domains, we used on our implementation IODEF format [25] to encapsulate messages exchanged by the different probes. We encoded the binary vectors of the aggregated Bloom filters in Base64. The XML format and the Base64 encoding adds some overhead to the exchanged information.

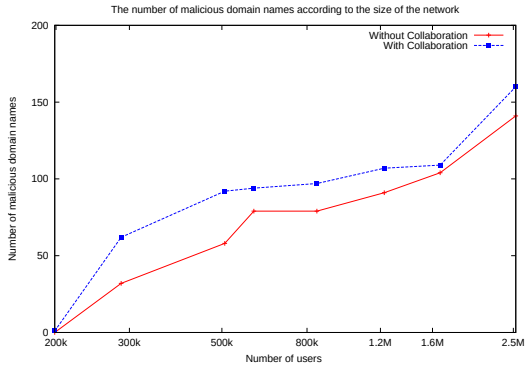

Figure 4. Number of malicious domain names detected according to the size of the network in probe 1

We evaluated the impact of the collaborative detection between two probes compared to stand-alone detection. The evaluation of the performance of the stand-alone detection has been published in [15].

To evaluate the benefits of enabling the collaborative detection in the domain-flux detection use-case, we measured the number of detected malicious domain names and false positives in the following cases:

- Each probe analyses independently the traffic capture.
- The probes collaborate in the analysis of the traffic captures.

We varied the size of the network in the captures analysed by the probes in order to measure the benefits of the collaboration, see figures 4, 5, 6, and 7. We see clearly in the figures that the smaller is the network, the more it benefits from the collaboration.

The large networks benefits less from the collaboration because we used in our evaluation captures originating from the same network operator. Each large capture contained enough infected traffic in order to detect the malicious domain names without collaboration.

We see clearly that the second probe detected less ma-

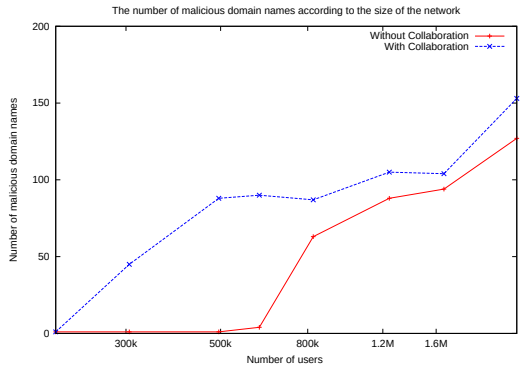

Figure 5. Number of malicious domain names detected according to the size of the network in probe 2

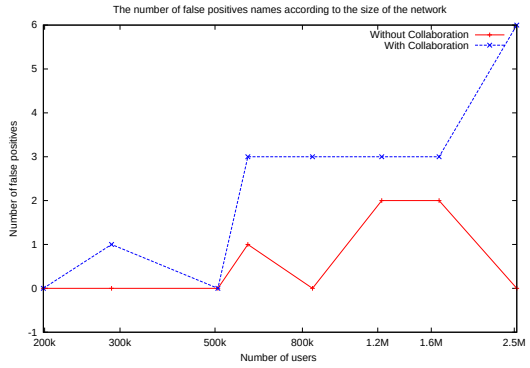

Figure 6. Number of false positives detected according to the size of the network in probe 1

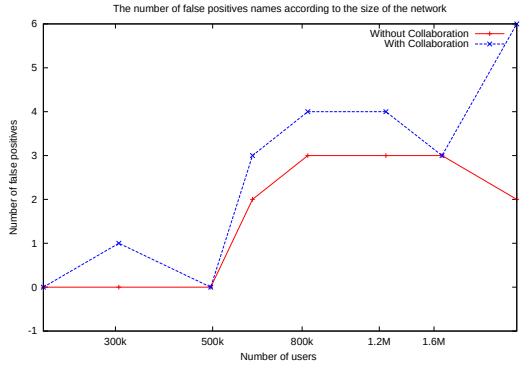

Figure 7. Number of false positives detected according to the size of the network in probe 2

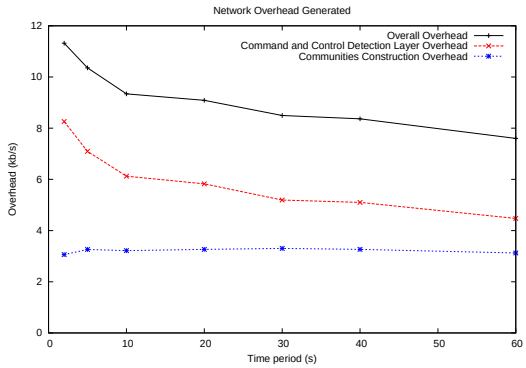

Figure 8. Network overhead generated by the system, command and control detection and communities construction layers

licious domain names than the first probe without collaboration. Therefore it benefited more from the collaboration, implying that the second part of the capture contained less infected users.

The detected false positives are all subdomains from popular domain names ranked in the top sites by Alexa [26]. They were not filtered by our system because these subdomains were not popular within all the communities of our system. Since we have only access to the DNS traffic, we can't assert that these domain names were requested during a botnet activity or requested legitimately by the users.

We computed the variation of the average network overhead generated by messages exchanged between the probes of the communities construction layer and the C&C detection layer according to τ the period of update of the C&C layer, see figure 8. We also represented the overall network overhead the sum of the two values. We see that when τ is 2 seconds, the overall network overhead is maximal and is 11.32 kb/s which represents $3.6{*}10^{-4}$ of the overall network traffic.

The C&C layer overhead decreases with the increase of τ because the probes update their aggregated Bloom filters according to τ. This overhead depends also on the number of communities on the system. On the other hand, the overhead generated by the communities construction layer is constant.

Our collaborative domain-flux detection use case doesn't scale to several thousands nodes. It is meant to be deployed between administrative domains. In our implementation of the collaborative domain-flux detection use case, we base our detection of the communities on the similarity of the network activity at a given moment using the comparison of the vectors of bloom filters. This implementation scales reasonably to few dozens probes, but is limited by the overhead associated to the broadcast of information during the communities construction. This limitation can be over-

come in two ways: by enhancing the election process; by compressing the exchanges which would help to reduce the overhead by two orders of magnitude.

Architectures of DHT type (distributed hash table) [27] doesn't fit our collaborative system for the domain-flux use-case because we compare directly between the vectors and because the value of the vectors sent are very short lived.

Regarding the performance, activating the cooperation has a very limited impact over the processing capabilities of the probes. We measured the average number of transactions per second processed by the different probes when enabling the cooperation. It decreases by less than 4%. It is therefore largely counter-balanced by the benefits observed regarding the detection accuracy. With a resulting processing speed of around 500 thousand transactions per second, this speed exceeds the bandwidth of the captured DNS traffic. This performance allows to consider a real-time application of our system in the network.

VIII. Conclusion

In this paper, we proposed a fully distributed collaborative and inter-domain botnet detection approach that complies with the privacy, business and performance constraints of network operators communications. In order to validate our detection system, we analysed the network traffic of a large scale operator. We showed that the collaboration improves the detection rate with a low network and processing overhead.

Acknowledgement

This work was partially supported by DEMONS, a research project supported by the European Commission under its 7th Framework Program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DEMONS project or the European Commission.

References

[1] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *Proceedings of the USENIX SRUTI Workshop*, vol. 39, 2005, p. 44.

[2] G. Ollmann, "Botnet communication topologies," 2009.

[3] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on*. IEEE, 2009, pp. 268–273.

[4] D. Barroso, "Botnets-the silent threat," *European Network and Information Security Agency (ENISA)*, vol. 15, p. 171, 2007.

[5] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*. USENIX Association, 2007, p. 12.

[6] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, vol. 7. Cambridge, MA, 2007.

[7] E. Kosta, J. Dumortier, H. Graux, R. Tirtea, and D. Ikonomou, "Study on data collection and storage in the eu," ENISA, European Network and Information Securiy, Tech. Rep., 2012.

[8] G. Jacob, R. Hund, C. Kruegel, and T. Holz, "Jackstraws: Picking command and control connections from bot traffic," in *USENIX Security Symposium*, 2011.

[9] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: analysis of a botnet takeover," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 635–647.

[10] G. Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," 2008.

[11] P. Mockapetris, "Domain names - concepts and facilities," http://www.ietf.org/rfc/rfc1034.txt.

[12] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in dns traffic," in *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on*. Ieee, 2007, pp. 715–720.

[13] N. Jiang, J. Cao, Y. Jin, L. Li, and Z. Zhang, "Identifying suspicious activities through dns failure graph analysis," in *Network Protocols (ICNP), 2010 18th IEEE International Conference on*. IEEE, 2010, pp. 144–153.

[14] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon, "From throw-away traffic to bots: detecting the rise of dga-based malware," in *Proceedings of the 21st USENIX conference on Security symposium*, ser. Security'12, 2012, pp. 24–24.

[15] H. Guerid, K. Mittig, and A. Serhrouchni, "Privacy-preserving domain-flux botnet detection in a large scale network," in *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*. IEEE, 2013, pp. 1–9.

[16] H. Wang and Z. Gong, "Collaboration-based botnet detection architecture," in *Intelligent Computation Technology and Automation, 2009. ICICTA'09. Second International Conference on*, vol. 2. IEEE, 2009, pp. 375–378.

[17] M. E. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo, "Towards collaborative security and p2p intrusion detection," in *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*. IEEE, 2005, pp. 333–339.

[18] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[19] G. Vigna and R. A. Kemmerer, "Netstat: A network-based intrusion detection system," *Journal of Computer Security*, vol. 7, no. 1, pp. 37–71, 1999.

[20] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.-L. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance *et al.*, "Dids (distributed intrusion detection system)-motivation, architecture, and an early prototype," in *Proceedings of the 14th national computer security conference*. Citeseer, 1991, pp. 167–176.

[21] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[22] F. Leder and T. Werner, "Know your enemy: Containing conficker," *The Honeynet Project, University of Bonn, Germany, Tech. Rep*, 2009.

[23] "Zeus gets more sophisticated using p2p techniques," http://www.abuse.ch/?p=3499.

[24] Damballa, "Pushdo evolves again: Enhances evasion with domain generation algorithm," 2013.

[25] R. Danyliw, J. Meijer, and Y. Demchenko, "The incident object description exchange format," RFC 5070, Tech. Rep., 2007.

[26] "Alexa top sites," http://www.alexa.com/topsites.

[27] C. V. Zhou, S. Karunasekera, and C. Leckie, "Evaluation of a decentralized architecture for large scale collaborative intrusion detection," in *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*. IEEE, 2007, pp. 80–89.