Computer Networks 70 (2014) 302–311

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Resource monitoring for the detection of parasite P2P botnets

Rafael A. Rodríguez-Gómez [a,*], Gabriel Maciá-Fernández [a], Pedro García-Teodoro [a], Moritz Steiner [b], Davide Balzarotti [c]

[a] Dpt. of Signal Theory, Telematics and Communications, CITIC-UGR, Spain
[b] Bell Labs, Alcatel-Lucent, Murray Hill, NJ, USA
[c] EURECOM, Biot, France

ARTICLE INFO

Article history:
Received 24 December 2013
Accepted 28 May 2014
Available online 20 June 2014

Keywords:
Parasite botnet
Detection system
Peer-to-peer
Mainline

ABSTRACT

Detecting botnet behaviors in networks is a popular topic in the current research literature. The problem of detection of P2P botnets has been denounced as one of the most difficult ones, and this is even sounder when botnets use existing P2P networks infrastructure (parasite P2P botnets). The majority of the detection proposals available at present are based on monitoring network traffic to determine the potential existence of command-and-control communications (C&C) between the bots and the botmaster. As a different and novel approach, this paper introduces a detection scheme which is based on modeling the evolution of the number of peers sharing a resource in a P2P network over time. This allows to detect abnormal behaviors associated to parasite P2P botnet resources in this kind of environments. We perform extensive experiments on Mainline network, from which promising detection results are obtained while patterns of parasite botnets are tentatively discovered.

1. Introduction

Botnets constitute a serious security flaw for systems, services and users, as they are a principal infection vector for spreading malware as well as a main supporting tool for several other kinds of attacks and malicious behaviors (*e.g.*, DDoS, spam) [1,2].

Among some other kinds of botnets, parasite botnets are of special relevance because they rely their operation on the use of an already existing communications infrastructure. This is the case of parasite botnets on P2P networks, where the activity of the botnet is easily hidden inside the usually high amount of traffic and activity in the underlying P2P network [3,4].

Most of proposals on botnets detection are oriented to introduce some kind of scheme to detect botnet command-and-control (C&C) activity [5]. However, a main limitation of this approach is the usual excessive specificity involved in representing the corresponding traffic. Since C&C network traffic is quite mutable and easy to obfuscate, models should be built for every specific botnet. In addition, these proposals require a high amount of traffic observations to derive adequate behavioral models to characterize the botnet activity. Obtaining representative and real traces of the traffic generated by a certain botnet is quite a hard task, and trying to reproduce it in a controlled environment always involves risks [6].

In P2P botnets, C&C communications are done through the exchange of resources shared by nodes in the network. For example, the botmaster of a P2P parasite botnet can create a file composed of bot commands and shares it with the bots. Every bot that downloads the file subsequently shares it again with other bots. Note that here, C&C

* Corresponding author. Tel.: +34 958241777.
E-mail addresses: rodgom@ugr.es (R.A. Rodríguez-Gómez), gmacia@ugr.es (G. Maciá-Fernández), pgteodor@ugr.es (P. García-Teodoro), moritz@bell-labs.com (M. Steiner), balzarotti@eurecom.fr (D. Balzarotti).

http://dx.doi.org/10.1016/j.comnet.2014.05.016
1389-1286/© 2014 Elsevier B.V. All rights reserved.

communications are not different from file download traffic and, for this reason, it is not trivial to build a detection system based on monitoring the network traffic.

This paper introduces a novel approach to detect parasite P2P botnets, which is based on resource sharing behavior monitoring instead of network traffic monitoring. Specifically, we are interested in the temporal evolution of the amount of peers sharing a given resource in the network. This way, a model for the normal behavior of temporal resource sharing in P2P networks is first obtained. As a complementary study of this behavioral model, and based on common constraints that botmasters should comply for the design of a botnet, a theoretical model for the expected behavior in botnet resource sharing is also proposed. From both, the detection of potentially anomalous, botnet resource sharing in a P2P network and their associated bots are finally identified.

Summarizing, the contributions of this paper are threefold:

- A methodology for building a temporal resource sharing model to represent the normal behavior of a P2P network.
- A theoretical model for botnet resources sharing and, based on it, a simple generator of parasite botnet traffic.
- A detection architecture to determine the occurrence of parasite botnet events in P2P networks.

As a case study, and mainly motivated by its wide current use, we present extensive experiments to test our architecture in a real Mainline network (BitTorrent), obtaining promising detection results. In particular, different patterns of use of parasite botnets are tentatively discovered.

According to the abovementioned main aspects, the rest of the paper is organized as follows. Section 2 is devoted to present some related work on the topic of botnets and detection of botnets. Section 3 analyzes the way in which network resources are parameterized to model the normality of the target environment. Moreover, a theoretical model for the expected behavior of resource exchanges in botnets is also discussed. A module for monitoring time evolutions of resources in Mainline network is presented in Section 4. After that, and based on it, the overall functional architecture for the detection system proposed is detailed in Section 5. Section 6 describes the experimental framework used to evaluate the detection scheme and the results obtained. Finally, Section 7 summarizes the main conclusions of the paper.

2. Related work

A botnet is a network of infected machines under the control of a human operator. Two main components can be found within a botnet: the bots and the botmaster. Infected machines are called *bots*, which follow the instructions given by the human operator, the *botmaster*. The botnet is controlled through the transmission of C&C messages among its members. *C&C channels* must be established. In some cases, bots connect to a *C&C server* in order to receive the messages sent by the botmaster. The existence of this server and the architecture of the C&C communications depend on the botnet's own architecture.

The transmission of the C&C messages can be *centralized*, *distributed* or *hybrid*. In a *centralized* scheme, bots contact the C&C server in order to receive information from the botmaster. In general, short time is spent in the transmission of a message from the botmaster to all the bots, and this represents one of the major advantages of this scheme. Its disadvantage is the fact that the C&C server constitutes a single point of failure. Thus, if the server shuts down, the complete network is dismantled. Examples of centralized botnets include Eggdrop [7], Gt-Bot and Agobot [8].

In a *distributed* architecture, all the bots in the botnet act simultaneously as servers and clients. This approach avoids the existence of a single point of failure, and so this kind of botnet is more resistant to take down attempts than a centralized one. However, the time required for a message to reach all the nodes is much greater than in the centralized case. Many botnets present a distributed structure, including Spybot [9] and Storm [10].

Finally, *hybrid* botnets combine the advantages of the two previous architectures. In this case, there exist one or more distributed networks, each with one or more centralized servers. The disconnection of one of these servers implies, in the worst case, the fall of one of the distributed networks, allowing the rest of the botnet to continue its normal operation. Some examples of hybrid botnets are Torpig [11], Waledac [12], Alureon/TDL4 [13] and Zeus-P2P [14].

Parasite botnets are a particularly harmful. They can be either hybrid or distributed. These botnets act as a parasite benefiting from existing communication infrastructures. In a parasite P2P botnet, an existent and legitimate P2P network is used to send C&C messages. It is specially hard to detect these types of botnets because their control messages are well hidden between the legitimate messages of the network. There exist two known parasite botnets: Storm [10], and a specific evolution of Alureon [15]. Both of them are now dismantled and they communicated through Kad (an implementation of Kademlia DHT for eDonkey network). These type of botnets are the aim of detection of our proposed detection system.

A considerable amount of work has been dedicated to the analysis [16–19] and detection [20–29] of botnets. From a network perspective, existing techniques are either based on horizontal or vertical correlation. The first category focuses on finding similarities between the behavior of infected hosts, while the second one focuses on the property of the traffic generated by a single host.

Looking closely at P2P botnet detection, only a handful of techniques has been proposed so far. Yen and Reiter [30] proposed the use of three main features (traffic volume, churn, and temporal patters) to tell filesharing and P2P botnets apart. Zhang et al. [31] used a statistical fingerprint to profile different types of P2P traffic and distinguish the benign from the malicious ones. Coskun et al. [32] showed how infected machines in a network can be discovered once a single P2P infected node is discovered. Similarly, [33] identified infected machines by looking at the communication patterns from the traffic observed by one (or multiple) ISPs. Finally, PeerRush [34] proposed a P2P traffic categorization approach that can be used to identify different known types of legitimate and malicious P2P traffic.

Our approach differs from the previous ones because we do not perform our analysis on the network traffic, but we study the "behavior" of the shared resources in the P2P infrastructure. Moreover, we put ourself in a worst-case scenario in which the botnet does not use a custom P2P protocol but it hides its traffic in an existing one (parasite botnet).

3. A model of resource sharing in P2P networks

The main intuition behind our proposal is the fact that resources shared by legitimate users in a P2P network (*legitimate resources*) will be accessed in a different way than resources shared by nodes belonging to a botnet (*botnet resources*). For this reason, we are interested in building models for both legitimate and botnet resources. Then, we will suggest a detection architecture that relies on these models to detect botnet resources in P2P networks.

Our models are based on the evolution of the *number of P2P nodes that share a specific resource* over time. This way, let $n_r(k)$ be the number of nodes sharing a resource r during a period of time of duration δ which ends at $k \cdot \delta, k = 1, \ldots, K$.

In Section 4 we detail a methodology for the estimation of $n_r(k)$. Following it, we have monitored 71,135 resources in the Mainline network during 3 months (see details in Section 6), observing that there are different patterns in the evolution of $n_r(k)$. As a first classification, we have found that there are resources that, at least during a period of time, have been shared by a large amount of nodes. Let us denote them as *popular resources*. We define a popularity threshold, θ_P, such that a given resource will be considered popular only if there exists a value $k, k = 1, \ldots, K$, for which $n_r(k) > \theta_P$. In Section 5 we specify how θ_P is determined.

We are specially interested in modeling *popular resources*, mainly because we expect that botnet resources

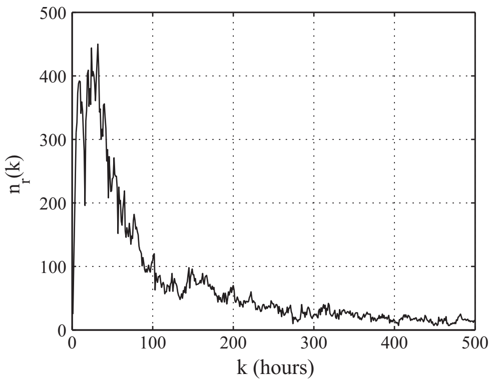

Fig. 1. Sharing evolution, $n_r(k)$ for a typical legitimate P2P resource ($\delta = 1$ h).

will belong to this category. Fig. 1 shows the evolution of $n_r(k)$ for a typical popular resource ($\delta = 1$ h). In the figure, we observe that the evolution of $n_r(k)$ has two separate parts. At the beginning, the peers start downloading and sharing the resource, reaching a period in which the resource is shared by many peers. After that, the interest decreases slowly until the resource is no longer shared. Let us call the first period *sharing phase* and the second *disappearing phase*.

Based on the behavior of $n_r(k)$ for legitimate popular P2P resources, our main hypothesis is that $n_r(k)$ will differ substantially from these patterns when r represents a botnet resource. This means that the detection of potential botnet resources would be possible by monitoring $n_r(k)$ and detecting deviations from the expected temporal sharing behavior. The key problem to verify this hypothesis is that it is not possible to use experimental traces from botnet resources. To the best of our knowledge, nowadays there are no real active parasite botnets reported, possibly due to the fact that it is extremely difficult to detect them by existing methods. For this reason, our strategy is to build a theoretical model for the sharing evolution of botnet resources, assuming that botmasters must follow certain rules to maintain and operate their botnets. Here, it is important to consider two properties that could not be circumvented by botmasters without degrading botnets behavior or exposing them to active detection mechanisms:

1. **Botnet resources must be popular resources.** Botnets are composed of a huge number of bots or peers [35]. When botmasters update the bot code or send commands, all the bots must download a given botnet resource which contains those commands or updates. This implies that a large number of downloads will be observed for that botnet resource. Thus, we will not consider small size botnets, as their threat would not be significant, at least in a first step.
   At this point, we must remark that the condition of being popular is not strictly necessary, as a botmaster might bypass a popular resource-based detection process by creating different files for groups of bots, instead of a single file for all of them. However, the increasing number of simultaneous new resources in the network can also be adopted as an alarm basis for the occurrence of anomalous behaviors. Therefore, and although not unique, we will continue assuming the resource popularity as a valid criterion for our detection approach.
2. **Botnet resources must have a short life-time.** The period of time during which a botnet resource is being shared should be short, due to the fact that this file is directly pointing to all the members of the botnet, exposing them to known detection systems [31,36]. Additionally, commands from the botmaster are very frequents and a bot should always be updated with the last commands.

These two rules imply that, during the *sharing phase* of a botnet resource, all the bots that download it also share it to assure that it is accessed by the whole botnet. Thus, during this phase, a high level of popularity is expected. On the other hand, and just to minimize the probability of been discovered by detection mechanisms, it is expected a *disappearing phase* in which all the bots will stop sharing the botnet resource in a short period of time. The evolution of these phases will depend considerably on several factors, so that our theoretical model consider these parameters:

- *Number of bots of a parasite botnet*, N. We consider in our approach high values for N, as otherwise the botnet threat would not be considerable.
- *Mean duration of the sharing phase of botnet resources*, t_{sh}. As previously discussed, it is expected that this time will be low.
- *Mean bot arrival and leaving rates*, λ. As some works point out [37], the arrivals and leavings of members in a botnet follow Poisson distributions, λ representing the mean value for these distributions. Moreover, as previously claimed, the transition from the sharing phase to the disappearing phase is expected to be sharp, or at least sharper than in the case of legitimate resources.
- *Disappearing interval.* The order from the botmaster for not sharing a botnet resource anymore should spread across the whole botnet. Due to deviations in the synchronization of bots, a short disappearing interval is expected. We model this interval as an uniform distribution $U[0, \Delta]$.

Following our model, in Fig. 2 we show the evolution of $n_r(k)$ for some examples of botnet resources with different parameters. Note that the main difference between the evolution of $n_r(k)$ for a legitimate and a botnet resource is around the instant t_{sh}, where $n_r(k)$ falls abruptly (compare Figs. 1 and 2). This motivates us to build a detection system based on monitoring the evolution of $n_r(k)$ in order to identify potential botnet resources.

4. Resource monitoring in a P2P network

This Section describes the methodology proposed to estimate the normality model of our P2P network and,

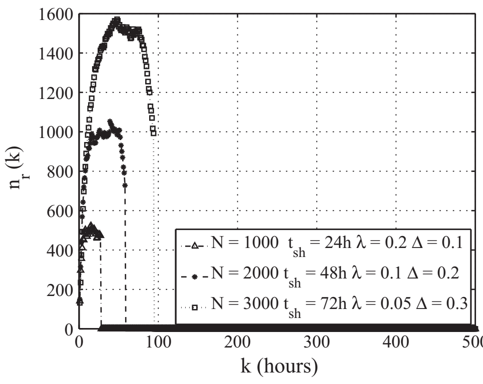

Fig. 2. Time evolution of different synthetic updates of a parasite botnet (δ = 1 h).

from that, to detect subsequent potential anomalous behaviors. In both cases, a previous monitoring process is required to determine the P2P resources shared by peers in our environment over time, that is $n_r(k)$. Such a monitoring is described in what follows, for which a BitTorrent network is considered, as it is one of the most used P2P network at present [38]. Traditionally, BitTorrent employs trackers in order to coordinate file exchanges. Recently, BitTorrent introduced decentralized tracking, a feature that enables any peer to act as a tracker, by means of a Distributed Hash Table (DHT). This new protocol is based on Mainline network [39] and we focus our monitoring process on it. In the following we present a brief description of the Mainline basics and, after that, we explain the resource monitoring module.

4.1. Mainline basics

In Mainline, every node has a global identifier (node ID) randomly generated with a size of 160 bits the first time that a new client starts the BitTorrent application. This way, it can be assumed the node ID as a unique identifier per client even in the case that this client changes its IP address. Additionally, each resource shared through Mainline has also a unique identifier (file ID) with the same size of the node ID, 160 bits. This identifier is generated as a result of a hash function of the shared resource. To compute the distance between two different node IDs, or a node ID and a file ID, a distance metric is used. This metric is an XOR operation. It is assumed that higher values of this metric imply higher values of distance.

Summarizing, users with node IDs close to a specific file ID are in charge of saving the information about which nodes in the network have a copy or a part of this file. Therefore, if a node tries to find a file in particular, it uses the XOR operation to find in its routing table node IDs close to the associated file ID. Then, it contacts those nodes requesting a list of nodes sharing the target file. If the contacted nodes have the requested information they will return it. Otherwise, the contacted nodes answer with a list of nodes closer to the file in their routing tables. This way, the requesting node iteratively approaches the destination file ID.

In the following we present the most relevant remote procedure calls available in the Mainline DHT for our work. For more details, please refer to [39].

- `ping` verifies if a peer is alive and responsive.
- `find_node` requests the closest peers to a hash value `I` in order to update the routing tables of the requesting peer. A peer responds to a find node message with the IP addresses, ports, and node IDs of the peers whose node IDs are the closest ones to `I` in its routing table.
- `announce_peer`, announces that a peer holds a file (or a portion of it) with info hash `I`. A peer sends `announce_peer` messages to the k peers whose node IDs are the closest ones to `I`. This information expires after a timeout that depends on the client implementation (around 30 min). The announcing peer is responsible for re-announcing the tuple $\langle$`IP:port,I`$\rangle$ over time.

4.2. Resource monitor

Our monitoring system is composed of two modules: (a) a node crawler, and (b) a message sniffer.

- *Node crawler*. This module is based on the work in [38,40]. Two asynchronous threads are always running in the crawler: one in charge of sending `find_node` messages, S_{thread}, and the other in charge of receiving and processing the associated answers, R_{thread}. A list containing the found nodes is shared between the threads. R_{thread} adds every found node to this list, while S_{thread} sends iteratively `find_node` messages to each member of the list, each one with a different ID. It is important to note that this ID is carefully chosen so that it belongs to a different section of the route tree of the nodes. This way, the overlapping of the group of received nodes is minimized.

  By using this crawler, we can extract the information of the whole Mainline network in around 40 min. It is also possible to crawl only a specific zone of the network with a certain prefix. For example, for 8 bits prefix all the active nodes in the network which ID begins with the same 8 bits are found. This process, which can extract 1/256 of the total active nodes in the network, can be done in less than 1 min.
- *Message sniffer*. This module is based on the work in [38] slightly adapted to our case. We include a huge amount of sybil nodes in a specific zone of the DHT. These sybils are capable of logging all the messages sent to the active nodes in the target zone. The node IDs are near to those of the active nodes obtained in the previous crawling process.

  Let n be an active node in the monitoring zone, and ID_n its ID. The sniffer module inserts 256 sybils with the same first 47 bits of ID_n. The probability of the existence of a node with an ID closer than that of one of our sybils is extremely low. The sybil IDs are generated by varying the bits of the ID comprised between the 48th and the 56th position, *i.e.*, 8 bits are considered to obtain 256 different IDs. To include these sybils in the DHT we notify the existence of the sybils to the active nodes near the target n. Then, these nodes will spread their route information to their neighbors.

In summary, the resource monitoring procedure works as follows. First, the nodes of a specific zone of the DHT are obtained using the crawling module. After that, a huge amount of `ping` messages are sent to them with source IDs near to those of the active nodes in the zone. This way, we launch a pollution attack to the routing tables of these nodes including our sybil nodes as neighbors. After that, a message is sent to any of our sybils to reach active nodes in the monitored zone. Second, all the `announce_peer` messages sent to this zone are logged. These are the messages sent by the nodes that really have a physical copy of the shared resource. For each `announce_peer` message we save the ID of the announced resource, the IP, port and ID of the announcer node, and the timestamp of the arrival instant of the message. As a result of this process, for every monitored resource r and a set of intervals of duration δ seconds, $n_r(k)$ is computed.

Note that although we include a huge number of sybils in the network they only suppose an increment of some `ping` and `announce_peer` messages per real node of the monitored zone of Mainline. We could say, that the monitoring process has no real influence on the behavior of the peers in the monitored zone.

5. Resource-based botnet detection: architecture and operation

Based on the models presented in Section 3, our aim is to build now a detection system able to monitor $n_r(k)$ for the different resources in a P2P network, and detect potential botnet resources patterns.

The architecture of our proposal for a resource-based botnet detection system is shown in Fig. 3. The data from the resource monitoring system (described in Section 4) is feeded into our system, where the three typical stages are considered:

- *Preprocessing*. Both the training and detection processes require a previous common stage to preprocess the data given by the resources monitoring system.
- *Training*. First, a normality model is built to represent the sharing evolution of legitimate resources in the monitored P2P network.
- *Detection*. Once the model is obtained, every resource shared in the P2P network is analyzed in quasi-real time, in order to determine potential deviations with respect to the expected behavior. In such a case, an alarm is triggered indicating the detection of a malicious, botnet resource.

In what follows, each of the stages is discussed.

5.1. Preprocessing

As a first step, every monitored resource r is represented by a time series vector $\boldsymbol{n_r}$:

$$\boldsymbol{n_r} = [n_r(1), n_r(2), \ldots, n_r(K)] \tag{1}$$

where $n_r(k)$ represents the number of peers that share a resource r in the k-th observed period of duration δ for a monitoring interval $T = K \cdot \delta$. Every sample $n_r(k)$ received from the resource monitoring system is added to $\boldsymbol{n_r}$, so that $\boldsymbol{n_r}$ really represents the time series evolution of the number of peers sharing r. We say that a vector $\boldsymbol{n_r}$ is *complete* when $n_r(k) = 0,\ k > K$.

After building $\boldsymbol{n_r}$, a low pass filtering is performed to reduce spikes in the time series. These spikes are mainly due to the high churn of P2P networks, *i.e.* rate of node connections and disconnections. For instance, if a monitored resource r is shared by nodes from the same geographical zone, $\boldsymbol{n_r}$ will exhibit a period around 24 h because of computer disconnections during the night.

For every point in the time series $\boldsymbol{n_r}$, the filtering is made taking the maximum value within a window of size W (being W an odd number). The resulting window is

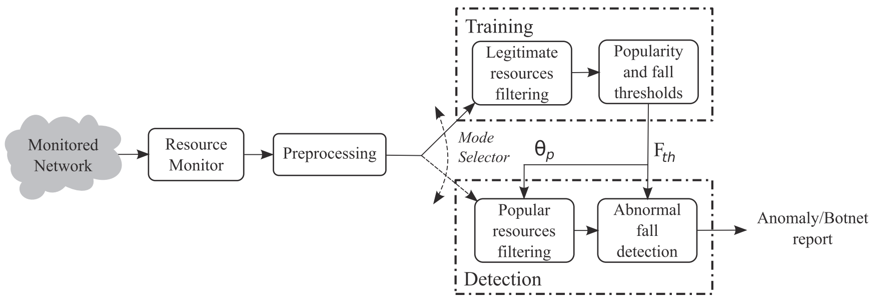

Fig. 3. Functional architecture of the detection system.

composed of a number of intervals of duration δ centered at interval k. This way, we obtain a new time series $\hat{\boldsymbol{n}}_{\boldsymbol{r}}$, where

$$\hat{n}_r(k) = max_i\{n_r(i)\}, \quad i = k - \frac{W-1}{2}, \ldots, k + \frac{W-1}{2} \tag{2}$$

Fig. 4 shows the result of such a filtering in a sample popular resource. As expected the time evolution is smoothed, what will avoid potential errors in subsequent analysis stages.

5.2. Training

The aim of this module is to estimate a normality model associated to the temporal evolution of the resources shared in the network monitored.

Resource database. First, a database with the evolution of all the shared resources in the monitored network is collected, each one being represented by a *complete* vector $\hat{\boldsymbol{n}}_{\boldsymbol{r}}$.

Legitimate resources filtering. As we are interested in modeling the behavior of legitimate resources, we must collect only those resources. In other case, incorrect "normality" models would be derived. For this purpose, different strategies could be followed. The naive approach is to download the resource and check if the contents are legitimate or not. An automated alternative could be followed by checking for certain information about the resources IDs in webs or blogs, *e.g.*, http://btdigg.org/ or http://torrentz.eu/.

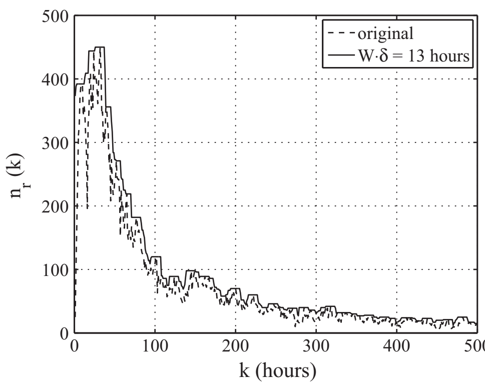

Fig. 4. Time evolution of a resource and the corresponding maximum filtering (δ = 1 h).

We could assume the existence of feedback from users in these webs as a proof to consider the resource as legitimate and not a botnet resource.

Fall threshold. As we are interested in identifying those resources that exhibit during a short period of time a high fall in the number of peers that share them (signature of botnet resources), we will identify the maximum negative difference between two consecutive values in $\hat{\boldsymbol{n}}_{\boldsymbol{r}}$ for every resource r, F_r:

$$F_r = |min_k\{\hat{n}_r(k) - \hat{n}_r(k-1)\}|; \quad k = 2, \ldots, K \tag{3}$$

Then, the fall threshold, F_{th}, is obtained as a portion α over the maximum fall for all the resources:

$$F_{th} = \alpha \cdot max_r\{F_r\}; \quad \forall r \tag{4}$$

Note that the value of α will modify the sensitivity of our system. A low value for α will imply a high false positives rate, while high values of α lead to low detection accuracy results. In Section 6 we explore ROC curves for this parameter.

Popularity threshold. As indicated along the paper, only popular resources are going to be analyzed by our system, as we claim that botnet resources are shared among a huge number of nodes. The selection of the value for the popularity threshold, θ_P, could be a difficult task in a generic application. Luckily, our case is different as we are really interested in detecting falls greater than F_{th}, what necessarily means that there exists a value k in the resource r for which $\hat{\boldsymbol{n}}_{\boldsymbol{r}}(k) \geqslant F_{th}$. For this reason, we take $\theta_P = F_{th}$ and filter those resources that fulfill this condition:

$$max_k\{n_r(k)\} \geqslant \theta_P; \quad k = 1, \ldots, K \tag{5}$$

5.3. Detection

The last stage of the detection system is that of determining the normal or anomalous nature of each new input observation taken from the resource monitoring system and preprocessed, $\hat{\boldsymbol{n}}_{\boldsymbol{r}}$.

The core of the detection system corresponds to the module of *abnormal fall detection*. Here, F_r is obtained from Eq. (3) for every observed resource r, and it is compared to the fall threshold, F_{th}, to decide if r is legitimate or a botnet resource.

$$\begin{cases} F_r \leqslant F_{th} & \text{legitimate resource} \\ F_r > F_{th} & \text{botnet resource} \end{cases} \tag{6}$$

Note that, for the sake of increasing the performance of our system, before the abnormal fall detection module, we filter popular resources according to the corresponding training stage, where $\theta_P = F_{th}$.

6. Experimental evaluation

In this section we present experimental results obtained from testing our hypotheses, and thus our approach. First, we describe the data collected from the Mainline network and analyze it in some aspects to show how both legitimate and botnet resources behave. After that, we evaluate the detection capabilities of our proposal, showing tentative botnet patterns discovered by using it. in Section 5.

6.1. Resource monitoring and preprocessing

Following the methodology explained in Section 4, we have monitored all the shared resources of Mainline with the same 8 bits prefix for the ID: “0x8C”. Therefore we have monitored 1/256 of the total Mainline network. We consider that this portion of the network represents accurately enough the behavior of the whole Mainline network.

The monitoring have been carried out during 3 months in 2012, starting from April 4th until July 4th. During this period a total of 71,135 resources shared by millions of different IPs all over the world have been monitored. For the sampling, we have chosen a period δ = 1 h. For each δ, we have collected the following data per resource: the number of different IPs sharing it, the number of different node IDs, and $n_r(k)$, that is, the number of messages of type `announce_peer`.

We have focused only on those resources that started to be shared during the monitoring period, as we want to collect their complete time evolution. To identify these resources, we firstly monitored during 2 weeks the target zone of Mainline and, after that, we considered the new resources not observed during this period of time. From the total monitored resources, we identified 34,075 new resources.

Fig. 5 shows the *popularity* of the monitored resources in decreasing order, calculated as the maximum number of peers that have shared a resource during intervals of 1 h of duration. Here, we can check that the top 10 resources have been shared by around a million of different users in an hour. Moreover, it is important to notice that there is a big amount of resources (27,083) whose maximum number of users is one.

Just for the sake of a qualitative evaluation, let us consider that a resource is popular if more than 70 user share it in an hour. From our results, it can be seen that 344 of the observed resources are shared by more than 70 different users in an hour. If we suppose that the whole Mainline network presents the same behavior as the monitored zone, we will have 344 popular resources per zone, which means more than 965 new popular resources per day in the whole network. Obviously, this is a very high number, but it is explained by the low threshold of users considered for popularity in this example (only 70 users).

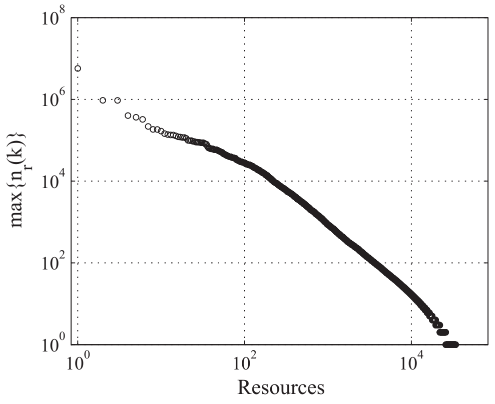

Fig. 5. Maximum value of peers sharing resources, $max\{n_r(k)\}$, for all the monitored resources.

Finally, we have preprocessed the vectors $\boldsymbol{n_r}$ following Eq. (2) with $W \cdot \delta$ = 12 h. This size is big enough to diminish the effect of the high churn of P2P networks, while small enough to leave the proper fluctuations of the time evolution of resources. This way, $\hat{\boldsymbol{n}}_{\boldsymbol{r}}$ is obtained.

6.2. Training and detection results

In order to make the training, we first identify those resources that could be considered legitimate. This is done by obtaining information published in well known sites about the resources IDs and considering those that have a good reputation from users. We have consulted four websites: http://www.torrentkitty.com/, http://torrentproject.com/, http://btdigg.org/ and http://torrentz.eu/. Out of the 34,075 resources, we have selected 14,869 resources which are corroborated as legitimate.

Once obtained our training data set, we divide it into four parts, so that a cross validation process is performed by taking three partitions for training purposes and the remaining one for testing. This way, four different experiments are carried out, each one corresponding to the use of one of the partitions for testing.

Additionally to the traces collected from Mainline, 42,000 synthetic bot resources following our model in Section 3 are added to the test partition in each case. As shown in Table 1, a wide range of different values for the parameters of this model have been used. We have chosen really low values of N in order to probe the detection capabilities of our system with botnets of a reduced size. Note that the popularity of botnet resources is not a critical condition in our system, it affects more to the efficiency of the system than to the detection results. The existence of

Table 1
Values for the parameters of the synthetic parasite botnet resources.

	Values
N	1.000–10.000
λ	0.1, 0.3, 0.5
t_{sh}	6 h to 1 week
Δ	0.1, 0.2, 0.3

both legitimate and botnet resources in the test data set allows to obtain a ROC curve representing the performance of our detection system.

Following Eqs. (3) and (4), we obtain a ROC curve by varying parameter α, from 0.01 to 2. This values of α implies a popularity threshold (θ_P) from 6.26 to 1252 peers. We can see the resulting ROC in Fig. 6, which corresponds to the average ROC from the four abovementioned detection experiments in the cross-validation process.

As observed, a trade-off between false positives (FP) and detection accuracy (DA) is necessary, as both have the same tendency. This way, for low FP values also the values of DA are reduced (DA around only 40% for FP = 0%). Anyway, the overall detection behavior is good enough as a value of DA equal to 98.64% is reached with only a false positive rate lower than 0.3%.

From these results, we think that a really good operation point is that corresponding to α equal to 0.12 (DA $\sim$ 99% and FP $\sim$ 0.3%), as shown in the specialized literature on detection systems. On the other hand, values of α equal to, or greater than 1.2 obtain FP rates equal to zero, what also constitutes an adequate operation point to minimize the number of alarms while being sure about the malicious nature of the events detected.

6.3. Discovering botnet patterns

Once we have shown the performance of our detection approach, we carried out further experiments to check the system when new, unknown resources are observed and once the detector is completely tuned.

For that, we use the remaining 19,206 resources obtained in the monitoring process but not previously used for training and testing our system. These resources could correspond either to legitimate or to botnet resources, as we only know that they are not explicitly recommended as valid by users. Although the operation point to analyze the target resources may be that corresponding to (DA $\sim$ 99%, FP $\sim$ 0.3%), we have chosen a value of α = 1.2, which achieves a theoretical false positive rate of 0%. This is aimed to be very restrictive in generating alarms and, if so, be sure about the malicious nature of the detected events.

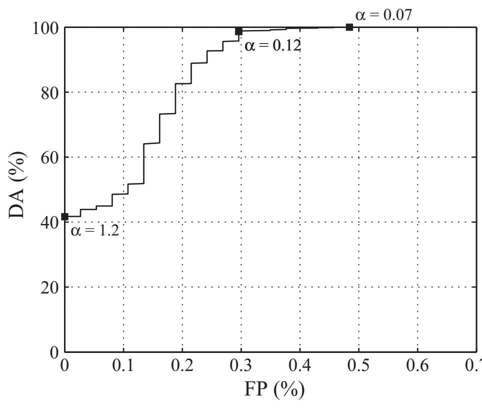

Fig. 6. ROC for our detection system by varying α in Eq. (4).

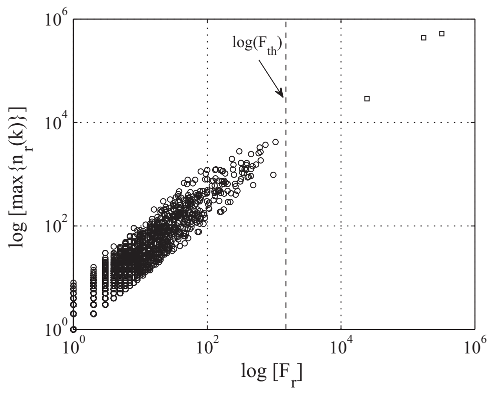

Fig. 7. Further detection experiments where new, previously unobserved botnet events are tentatively detected.

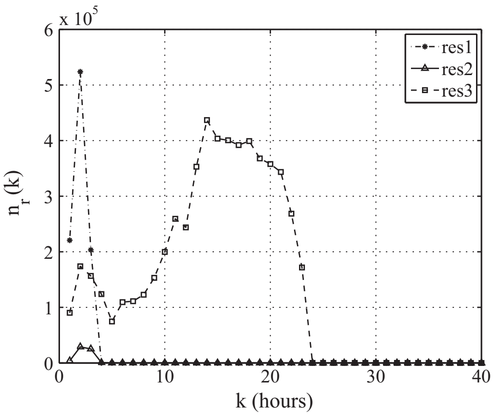

Fig. 8. Time evolutions of the three resources detected as abnormal (δ = 1 h).

Fig. 7 represents, for all the resources, the maximum popularity reached, $\max\{n_r(k)\}$ (y axis in log form), and the fall obtained from Eq. (3) (x axis also in log form). Here, the fall threshold F_{th} is also represented in dashed lines. We can see that our system raises alarms for only three of the monitored resources, which clearly behaves as outliers.

Fig. 8 shows $\boldsymbol{n_r}$ for these three resources. Here we can check that all of them are shared by a large amount of peers during a short period of time. Specifically, *res1* reaches a maximum of 523,883 users and the sharing phase of this resource lasts less than 5 h. The same happens with *res2*, with less but still significative number of peers (28,897). Regarding *res3*, the duration of the sharing phase is longer, but still very short (only 24 h), and the number of peers is really significative (436,963). These behaviors are really suspicious of being due to botnet resources sharing, as they follow our model presented in Section 3.

7. Conclusions and future work

This paper presents a new detection approach to determine the occurrence of parasite botnet events in P2P networks. The main novelty of our system is that it is based on resource monitoring instead of traffic monitoring. We claim that normal P2P resources are shared in a different way over time than those ones corresponding to botnet communications.

First, we have developed a methodology for recovering information about resources sharing patterns in a P2P network. It has been tested in Mainline network and proved to be feasible. Second, this monitoring system has allowed the definition of models for both legitimate and botnet resources sharing patterns. From them, a detection scheme based on the high fall in the number of peers sharing a given resource is developed.

As a proof of concept, the time evolution of resources of a zone of the Mainline network have been collected to evaluate the proposed detection approach. We show that our system exhibits a very good performance in terms of false positives (lower than 0.5%), and very high detection accuracy (higher than 99%). We show how our system is able to monitor the evolution of resources in real time, making the detection of botnet resources in a P2P parasite network a real possibility. Finally, the simplicity of the solution proposed makes it adequate for its implementation in real systems.

Acknowledgment

The research leading to these results has received funding from Spanish Government through project TEC2011-22579 and from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement nr. 257007.

References

[1] L. Zhang, S. Yu, D. Wu, P. Watters, A survey on latest botnet attack and defense, in: 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2011, pp. 53–60. http://dx.doi.org/10.1109/TrustCom.2011.11.

[2] M. Feily, A. Shahrestani, S. Ramadass, A survey of botnet and botnet detection, in: SECURWARE '09. Third International Conference on Emerging Security Information, Systems and Technologies, 2009, 2009, pp. 268–273. http://dx.doi.org/10.1109/SECURWARE.2009.48.

[3] L. Feng, X. Liao, Q. Han, L. Song, Modeling and analysis of peer-to-peer botnets, Discrete Dyn. Nat. Soc. (2012) 1–18.

[4] K. Muthumanickam, E. Ilavarasan, P2P botnet detection: combined host- and network-level analysis, in: 2012 Third International Conference on Computing Communication Networking Technologies (ICCCNT), 2012, pp. 1–5. http://dx.doi.org/10.1109/ICCCNT.2012.6395940.

[5] I. Ullah, N. Khan, H.A. Aboalsamh, Survey on botnet: Its architecture, detection, prevention and mitigation, in: 2013 10th IEEE International Conference on Networking, Sensing and Control (ICNSC), 2013, pp. 660–665. http://dx.doi.org/10.1109/ICNSC.2013.6548817.

[6] R.A. Rodríguez-Gómez, G. Maciá-Fernández, P. García-Teodoro, Survey and taxonomy of botnet research through life-cycle, ACM Comput. Surv. 45 (4) (2013) 1–39.

[7] R. Pointer, Home page of Eggdrop botnet, 1993. <http://www.eggheads.org> (accessed 25.07.13).

[8] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng, J. Zhang, Botnet: classification, attacks, detection, tracing, and preventive measures, EURASIP J. Wirel. Commun. Network. (2009).

[9] C. Li, W. Jiang, X. Zou, Botnet: survey and case study, in: 2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC), 2009, pp. 1184–1187.

[10] J.B. Grizzard, V. Sharma, C. Nunnery, B.B. Kang, D. Dagon, Peer-to-peer botnets: overview and case study, in: Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets, 2007, pp. 1–8.

[11] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, G. Vigna, Your botnet is my botnet: analysis of a botnet takeover, in: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS '09, 2009, pp. 635–647.

[12] G. Sinclair, C. Nunnery, B.-H. Kang, The waledac protocol: The how and why, in: 2009 4th International Conference on Malicious and Unwanted Software (MALWARE), 2009, pp. 69–77.

[13] E. Rodionov, A. Matrosov, The Evolution of TDL: Conquering x64, Tech. rep., ESET, June 2011 (accessed 25.07.13).

[14] abuse.ch, ZeuS Gets More Sophisticated Using P2P Techniques, Tech. rep., October 2011 (accessed 25.07.13).

[15] S. Golovanov, I. Soumenkov, TDL4 – Top Bot, Tech. rep., (accessed 25.07.07.13).

[16] E. Cooke, F. Jahanian, D. McPherson, The zombie roundup: understanding, detecting, and disrupting botnets, in: 1st Workshop on Steps to Reducing Unwanted Traffic on the Internet, 2005, pp. 39–44.

[17] F. Freiling, T. Holz, G. Wicherski, Botnet tracking: exploring a root-cause methodology to prevent distributed denial-of-service attacks, in: 10th European Symposium On Research In Computer Security, 2005.

[18] J. John, A. Moshchuk, S. Gribble, A. Krishnamurthy, Studying spamming botnets using botlab, in: 6th Usenix Symposium on Networked Systems Design and Implementation (NSDI), 2009.

[19] M.A. Rajab, J. Zarfoss, F. Monrose, A. Terzis, A multifaceted approach to understanding the botnet phenomenon, in: Internet Measurement Conference (IMC), 2006.

[20] J. Binkley, S. Singh, An algorithm for anomaly-based botnet detection, in: Usenix Steps to Reduce Unwanted Traffic on the Internet (SRUTI), 2006.

[21] J. Goebel, T. Holz, Rishi: identify bot contaminated hosts by IRC nickname evaluation, in: Workshop on Hot Topics in Understanding Botnets, 2007.

[22] G. Gu, R. Perdisci, J. Zhang, W. Lee, BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection, in: Usenix Security Symposium, 2008.

[23] G. Gu, P. Porras, V. Yegneswaran, M. Fong, W. Lee, BotHunter: detecting malware infection through IDS-driven dialog correlation, in: 16th Usenix Security Symposium, 2007.

[24] G. Gu, J. Zhang, W. Lee, BotSniffer: detecting botnet command and control channels in network traffic, in: 15th Annual Network and Distributed System Security Symposium (NDSS), 2008.

[25] A. Karasaridis, B. Rexroad, D. Hoeflin, Wide-scale botnet detection and characterization, in: Usenix Workshop on Hot Topics in Understanding Botnets, 2007.

[26] M. Reiter, T. Yen, Traffic aggregation for malware detection, in: DIMVA, 2008.

[27] W. Strayer, R. Walsh, C. Livadas, D. Lapsley, Detecting Botnets with Tight Command and Control, in: 31st IEEE Conference on Local Computer Networks (LCN), 2006.

[28] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, E. Kirda, Automatically generating models for botnet detection, in: 14th European Symposium on Research in Computer Security (ESORICS 2009), Saint Malo, Brittany, France, 2009.

[29] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, C. Kruegel, Disclosure: detecting botnet command and control servers through large-scale netflow analysis, in: Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC), ACSAC 12, ACM, 2012, pp. 129–138.

[30] T.-F. Yen, M.K. Reiter, Are your hosts trading or plotting? Telling P2P file-sharing and bots apart, in: 2010 IEEE 30th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2010, pp. 241–252.

[31] J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, X. Luo, Detecting stealthy P2P botnets using statistical traffic fingerprints, in: 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN), IEEE, 2011, pp. 121–132.

[32] B. Coskun, S. Dietrich, N. Memon, Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts, in: Proceedings of the 26th Annual Computer Security Applications Conference, ACM, 2010, pp. 131–140.

[33] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, N. Borisov, Botgrep: finding P2P bots with structured graph analysis, in: USENIX Security Symposium, 2010, pp. 95–110.

[34] B. Rahbarinia, R.P.A. Lanzi, K. Li, Peerrush: Mining for Unwanted P2P Traffic.

[35] M.A. Rajab, J. Zarfoss, F. Monrose, A. Terzis, My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging, in: Proceedings of the First Conference on First

Workshop on Hot Topics in Understanding Botnets, USENIX Association, 2007.
[36] S. Shin, Z. Xu, G. Gu, EFFORT: efficient and effective bot malware detection, in: Proceedings of the 31th Annual IEEE Conference on Computer Communications (INFOCOM'12), 2012.
[37] M. Khosroshahy, M.K. Mehmet Ali, D. Qiu, The SIC botnet lifecycle model: a step beyond traditional epidemiological models, Comput. Netw. 57 (2) (2013) 404–421.
[38] M. Varvello, M. Steiner, Traffic localization for DHT-based BitTorrent networks, in: Proceedings of the 10th International IFIP TC 6 Conference on Networking, NETWORKING'11, 2011, pp. 40–53.
[39] Mainline DHT Implementation. <http://bittorrent.org/beps/bep_0005.html> (accessed 25.07.13).
[40] M. Steiner, T. En-Najjary, E.W. Biersack, Long term study of peer behavior in the KAD DHT, IEEE/ACM Trans. Netw. 17 (5) (2009) 1371–1384.

Rafael A. Rodríguez Gómez is a Ph.D. student in the Department of Signal Theory, Telematics and Communications at the University of Granada. In this University he studied Telecommunication Engineering between 2003 and 2008. After that, he received his M.Sc. degree in Multimedia Technologies in 2009. In 2010 he started his Ph.D. titled: "P2P networks protection through traffic analysis" and now he is a member of the Network Security and Engineering Group (NESG).
Nowadays, his research interest is focused on network security, more specifically on P2P networks, the P2P traffic analysis as well the detection and defense against the attacks related with these networks. Currently, to be more precise, he is working in the detection and defense against P2P botnets.

Gabriel Maciá Fernández is an associate professor in the department of Signal Theory, Telematic and Communications of the University of Granada. He is a member of the Network Security and Engineering Group (NESG).
He received his M.S. degree in telecommunications engineering from the University of Seville (Spain), in 1998. After that, his professional activities were developed in some technological companies (Enditel-Endesa, Vodafone S.A.), were he worked for 7 years.
He joined the University of Granada in 2005, and received a Ph.D. in Telecommunications Engineering from that University in 2007, with the thesis titled (english translation) "Low-rate Denial of Service Attacks against servers".
Currently, his research activities are focused on the field of security in communication networks, with an special emphasis in attack generation methods, denial of service, and intrusion detection systems.

Pedro García-Teodoro received his B.Sc. in Physics (Electronics speciality) from the University of Granada, Spain, in 1989. This same year he was granted by "Fujitsu Espana", and during 1990 by "IBM España". Currently, he is a Full Professor at the University of Granada in the Department of Signal Theory, Telematics and Communications, and the head of the Network Security and Engineering Group (NESG) of this University.
His initial research interest is focused on network security (intrusion detection and response, DoS attacks, botnets, digital fraud,...), although he has also developed several other works related with Internet applications, multimedia services and e-learning, among others.

Moritz Steiner received the M.S. degree (Diplom) in Computer Science from the University of Mannheim, Germany in 2005, and the Ph.D. degree in Computer Networks from Telecom ParisTech, France and the University of Mannheim in 2008.
His doctoral thesis investigates how to build network virtual environments from unstructured peer-topeer networks. It also introduced measurement techniques and presented extensive measurement results on a real-world, large-scale, structured peer-to-peer file sharing network, namely Kad.
Moritz was a Member of Technical Staff at Bell Labs located in Murray Hill, NJ from 2009 to 2013. While at Bell Labs his research interests and project activities were in the areas of software defined networks, peer-to-peer networks and cloud computing.
In August 2013 Moritz joined the Web Experience Foundry team at Akamai. He participates in the overall web experience product mission by exploring new and adjacent technology opportunities. The Foundry team is the cutting edge arm of Akamai's Web Experience business unit.

Davide Balzarotti is an Assistant Professor at Eurecom Graduate School and Research Center, located in Sophia Antipolis on the French riviera. His research interests include most aspects of system security and in particular the areas of intrusion detection and prevention, binary and malware analysis, reverse engineering, and web security.