On Detection of Storm Botnets

Yuanyuan Zeng, Kang G. Shin
Real-Time Computing Laboratory, The University of Michigan, Ann Arbor, MI 48109-2121
{gracez, kgshin}@umich.edu

Abstract

A botnet, which is a group of compromised and remotely-controlled computers (also called bots), poses a serious threat to the Internet. The commonly-used command and control (C&C) channel for a botnet is used by a central server, such as IRC or HTTP. Recently, Storm botnet, a P2P-based botnet with a decentralized C&C channel has appeared in the wild. In this paper, we propose a distributed approach to detection of Storm botnets at the network level. Our approach is composed of two stages. First, we identify P2P and SMTP packets from each host's traffic. Second, we use a machine learning technique to differentiate Storm from benign P2P traffic based on several distinguishing traffic attributes. Both of the two stages only require packet header information without analyzing payloads. Our evaluation has shown the detection strategy to be effective with low false alarm rates.

1 Introduction

A *bot* is a computer robot installed via worms, Trojan horses, or backdoors, under a remote command and control (C&C) infrastructure. A group of bots is called a *botnet*, which can be used for DDoS attacks, mail spamming, etc. Botnets are now considered to be a major means of bank fraud, identity theft, and other cyber crimes.

Botnets commonly use the IRC control infrastructure. The attacker sets up an IRC server and specifies a channel which the bots all connect and listen to in order to execute the commands issued by a bot master. To detect the IRC botnet, the monitoring point is usually is at the IRC server's side whose behaviors can be distinguished from legitimate ones. Once the central IRC is identified, the whole botnet will be taken down easily. There also exist HTTP-based botnets, which work in a similar way to that of IRC botnets. The detection mechanisms are also similar. However, recently, a new class of botnets has emerged. The C&C of such botnets is based on a decentralized P2P protocol, which does not have a central point of control. This feature makes the detection and mitigation much harder.

Storm worm (or W32.Peacomm, Nuwar, Zhelatin) spreads via email spam and is known to be the first piece of malware to seed a botnet in a P2P fashion without any centralized control. By late 2007, it was estimated to run on around 250,000 to 1 million compromised systems. The Storm botnet has been used in some criminal activities, primarily for sending out spam mails. As a computer forensics specialist pointed out, 'Cumulatively, Storm is sending billions of messages a day. It could be double digits in the billions, easily.'

To date, most botnet-detection approaches target IRC or HTTP based botnets, while the detection of P2P based botnets such as Storm is still in its infancy. In this paper, we propose a fast and efficient detection strategy for P2P botnets, especially for Storm botnet. Given the decentralized C&C nature of Storm botnet, there is no central point for detection so that the monitoring points are better to be distributed. Our approach is a network-based one, which monitors traffic of every host in a network in order to identify if one is part of the Storm botnet. Considering the large traffic volume and processing difficulty, to make our approach efficient and scalable, we only use the information contained in the packet headers without examining packet payloads. Our approach can be broken down into two stages. In the first stage, we identify P2P and SMTP packets out of each host's traffic. Our study on Storm traffic shows that it does resemble real P2P traffic because the underlying P2P protocol used. Thus, heuristics that differentiate P2P from non-P2P packets can be applied. Port numbers are used to extract SMTP packets. In the second stage, trained by a few distinguishing traffic attributes between Storm and benign P2P applications using a machine learning technique, a classifier is employed to tell if part of the traffic identified in the first stage belongs to Storm botnet.

Our contributions are threefold. 1) We develop a detection strategy of P2P botnets by analyzing several traffic attributes obtained from the packet headers without accessing packet payloads. 2) We use those distinguishing traffic attributes to train a classifier by a machine learning technique to facilitate the decision process of detection, resulting in a high detection accuracy. 3) We present the empirical study of Storm botnet and normal P2P applications at network level.

2 Related Work

Although there is a rich body of literature in the area of botnet detection, almost all of them deal with botnets using IRC as C&C channel[7, 12, 5, 11, 3]. Since the emergence of the P2P style Storm botnet in 2007, people have started to think about the detection and mitigation strategies towards the decentralized P2P based botnets.

To the best of our knowledge, some people have already studied the Storm botnet such as [6], but only [8] and [13] come up with approaches trying to detect or mitigate Storm botnet. Holz et al.[8] have measured the size of current Storm net by infiltrating using a crawler. They also propose two mitigation strategies that introduce controlled peers to join the network in order to either separate or pollute the content of the Storm network. Porras et al. [13] try to detect the Storm bot by constructing Storm's dialogue life cycle model first an then identifying the traffic that matches this model. Compared with their work, our characterization of traffic patterns is at a higher level without access to packet payload. For example, they look for specific eDonkey message types such as publicize or search, while we are interested in the payload-size distribution, TCP/UDP ratio, etc. which is more robust towards packet encryption and when payload is not accessible.

The classification of traffic without accessing the packet payload has been a challenging problem. In our work, we need to identify P2P traffic for each host by analyzing the packet header only. Karagiannis et al. [10] provides a nice framework in this regard. We adopt their heuristics as well as maintaining a table for protocols with well-defined port numbers in order to filter out traffic.

3 Overview of Storm Worm

Storm worm is spread via email spam which entices users to click the attachments or to visit some URLs to download the binaries for the purpose of installing the bot on the victim. It seeds botnets based on P2P Overnet (a.k.a Kademila) protocol [2] as C&C. Once a host system is infected by a Storm instance, it will connect to the Overnet botnet and become a bot thereafter.

Storm sets up by adding a system driver into the host. This driver is injected into "services.exe", a Windows process. To become part of the botnet, it bootstraps by connecting hundreds of IPs contained in a peer list file hard-coded in the binary. After joining the network, the bot sends out search requests to find a specific secondary injection such as spam template, email harvester, rootkit component, etc. If the target file is successfully located, the bot will download and execute it. Bots can be programmed to obtain different injections in order to perform different tasks. It is also possible for a bot to update itself periodically. The P2P architecture allows each bot to actively seek its task instead of waiting passively for the C&C from a central server.

So far, there have been numerous Storm outbreaks since Jan, 2007(Table 1). Storm has shown resilience and effectiveness over a long period of time.

Table 1. Storm campaigns

Date	Spam subject
Jan-07	European Storm Spam
Apr-07	Worm Alert Spam
Jun-07	E-card (applet.exe)
Jul-07	231st B-day
Sep-07	Labor Day (labor.exe)
Sep-07	NFL Tracker
Dec-07	Christmas(disnisa.exe)
Jan-08	New Year
Feb-08	Valentine (valentine.exe)
Apr-08	April Fool (sony.exe)
May-08	Storm Codec (codec.exe)

We have selected a few Storm variant binaries that came out in the major Storm outbreaks such as Christmas last year, Valentine's day, April fool's day this year, etc. We ran them on a virtual XP system which was connected to the Internet and collected hours of traffic trace for each using Wireshark. Storm botnet relies on Overnet protocol for communication which consists of two phases, localization and download. Localization including connecting, search, etc. uses UDP messages, while download is done via TCP/IP. One slight change Storm made in Oct 2007 to the protocol is to XOR encrypt each message with a 40 byte long key. But the underlying principles are still the same because after decryption all packets can still be decoded as Overnet message types.

By running several of Storm instances, we have noticed that the time window between a host getting infected and becoming a functional bot is only a couple of minutes. Take valentine.exe for example. It was observed to start sending out spam in 5 minutes upon running. Taking advantage of the P2P protocol, the bot herder has good flexibility in commanding and controlling the botnet. As opposed to issuing C&C to multiple bots at a time, the bot herder may join the network as a peer and publish its command, say a spam component to the whole network as any other peer does. Since the bots have been programmed in advance to search and download specific content, they can reach this component immediately.

4 The Traffic Pattern Characterization of Normal P2P Applications and Storm Bots

For fast and widely applicable detection of botnet traffic, we only collect information from packet headers without analyzing payloads. In that case, the information available to us is limited containing source and destination IPs, source and destination ports, protocol and payload size. Intuitively, normal P2P applications aim at file-sharing so that they need to transfer many data packets, whereas Storm botnets use the P2P framework for communication purpose and presumably have small-size packets. In addition, most P2P protocols to date doing communication such as search, publish, etc. via

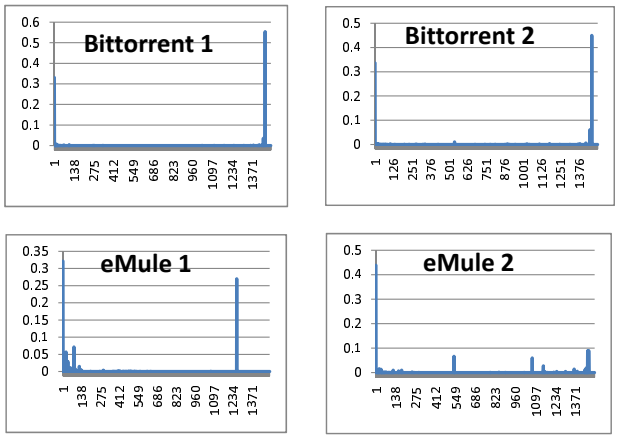

Figure 1. Payload-size histograms of Bittorrent and eMule

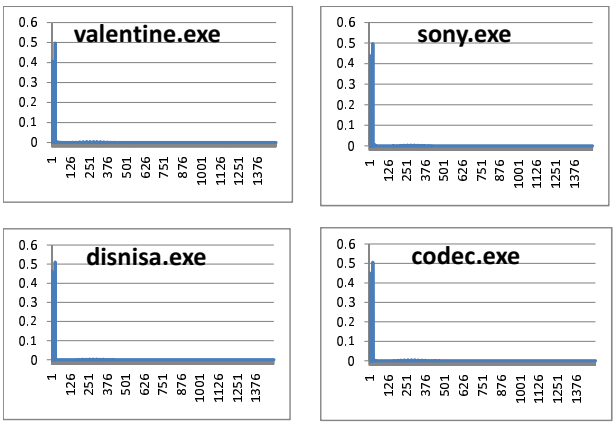

Figure 2. Payload-size histograms of Storm instances

UDP protocol and file transmission via TCP. We expect the TCP/UDP ratios to be different between Storm and normal P2P applications. Admittedly, there are exceptions when it comes to P2P audio and video streaming such as VoIP. If so, the payload size might be a more distinguishing attribute. Besides, as we learned that most Storm bots currently serve as spam senders, the number of SMTP packets may also be an indicator.

To characterize the traffic patterns of both benign P2P applications and Storm bots, we have collected traffic traces respectively. For normal P2P, we selected the application of most popular P2P protocols such as Bittorrent, eMule, Gnutella, etc. We show two cases here.

We ran a Bittorrent client on a host and captured 1-hour trace for file-sharing by Wireshark. The total packet number is 911,303. For the sake of illustration, we broke down the traces to 3 pieces, 20-minute each. Figure 1 are histograms for payload size (Y axis is the frequency divided by total number of packets). The 3 pieces of trace show identical pattern in terms of payload size distribution (we show 2 plots here), which is that around 50% percent of packets have payloads (data) exceeding 1000 bytes. As explained earlier, this is a common feature for file-sharing programs. Moreover, the TCP/UDP ratio is worth noticing, which is 122 of the entire 1-hour trace.

Another prevailing P2P application eMule is also in line with the observation. eMule [1] allows for connections to two networks, eDonkey and Overnet. Overnet network (a.k.a. Kad network) has an implementation of the Kademlia protocol, which does not rely on central servers as the eDonkey network does. Since Storm worm makes use of the Overnet network, we only enabled the Overnet connection while collecting the eMule traces to observe the traffic of normal P2P client that uses the same protocol as Storm does. We obtained 2 pieces of traces: one lasting for 20 minutes and the other 1-hour. As shown in figure 1, eMule payload size pattern slightly differs from that of Bittorrent. But the number of large-size packets is still considerable in both short and long eMule traces. TCP/UDP ratio is around 11.

On the other hand, the traffic traces from Storm instances demonstrate different patterns regarding payload size and TCP/UDP ratio as compared to that of normal P2P. We collected 1-hour data each from 4 Storm binaries. Interestingly, their payload size histograms are alike (figure 2). Small-size packets (< 100 bytes) account for more than 90% of all packets. There are no packets with greater than 1000 bytes payload. To compare the packet-size histograms, we show the difference histograms: one between two Storm variants and the other between Bittorrent and valentine in Figure 3. As we can see, codec and valentine's difference is negligible, up to 3% only. However, the difference between Bittorrent and valentine is large, up to 60% when it comes to the large-size packet percentage and more than 40% of 2 bytes and 25 bytes packets. As mentioned before, in Overnet protocol, communication packets such as connect, publicize, etc. are exactly 2 or 25 bytes. Note that the Overnet serves as a C&C channel other than a propagation means for Storm. Thus, it is reasonable that a majority of Storm traffic is UDP communication used for connection, publish, search, etc. The amount of TCP traffic is quite small since the purpose is not data transmission. For Storm, the TCP/UDP ratios are at the order of 0.01. Table 2 summarizes the ratios across Storm variants and normal P2P applications. The ratio of latter is more than 1000 times larger than that of the former.

Now we already introduced two attributes to distinguish Storm from normal P2P traffic. However, we need to consider the scenario that a host is infected by Storm and in the meantime it runs a normal P2P application. We have simulated this case by adding up Storm and Bittorrent, Storm and eMule altogether for some time. The number of large-size packets may not be a sign of 'normal'traffic although it turns out when normal mixes with Storm traffic the large-size packets percentage is smaller than that of normal traffic alone (figure 4). Nevertheless, the TCP/UDP ratio is still distinguishable due to the aggressive communication pattern of Storm. Table 2indicates that when Storm mixes with Bittorrent or eMule or both, the ratios are all below 1, far from the normal TCP/UDP ratios.

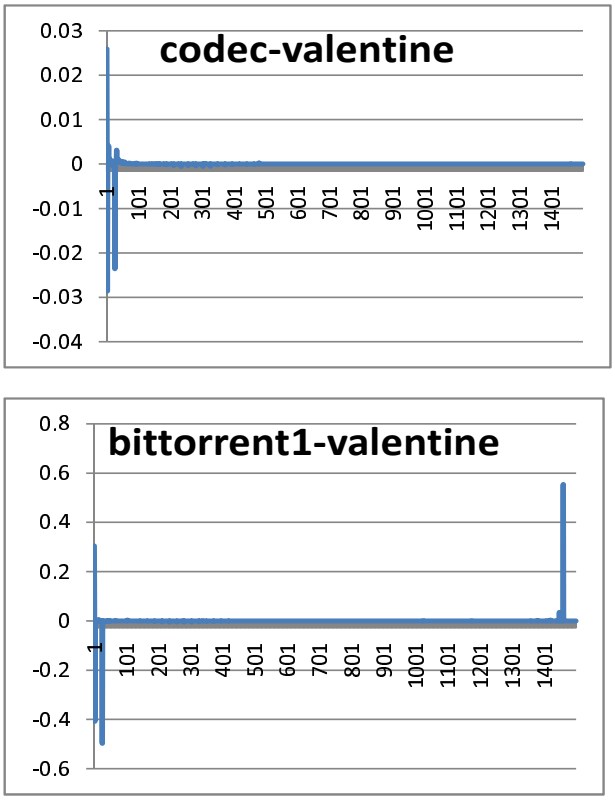

Figure 3. Payload-size histograms differences

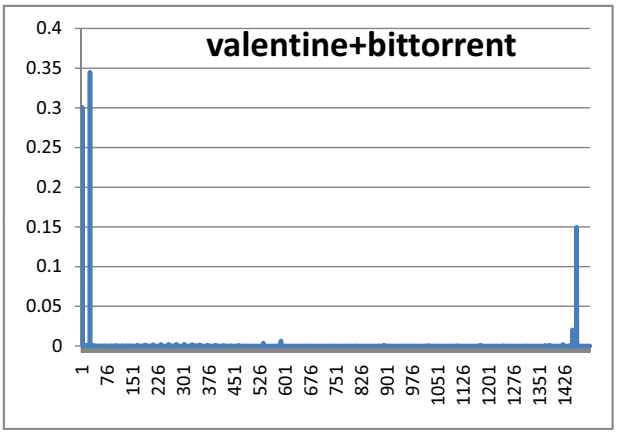

Figure 4. Mix Storm with Bittorrent

Table 2. TCP/UDP ratios

	tcp/udp			tcp/udp
valentine1	0.030473		emule1	3.373534
valentine2	0.003721		emule2	11.81117
valentine3	0.002707		bittorent1	115.591
codec1	0.001389		bittorent2	327.8396
codec2	0.001256		bittorent3	29.87293
codec3	0.00208		storm+bt1	0.49806
sony	0.027407		storm+bt2	0.50429
disnisa	0.002148		storm+emule	0.042162
			storm+bt+emule	0.68621

One may argue that as long as a host runs plenty of P2P clients the TCP/UDP ratio can reach normal ratios. Running numerous P2P applications simultaneously on one host is a rare case though it is possible. That's why we have the third attribute – the number of SMTP packets – to observe. Till now, most Storm variants are observed to send out spam. For example, the valentine.exe running on our testbed turns out to send more than 10,000 spam emails in just 20 minutes. Since within a network it is easy to know the IP of the internal SMTP server, the hosts that send packets to other domain's IPs with destination port 25, behaving like a SMTP server can be potential Storm bots.

To sum up, considering the payload-size distribution, TCP/UDP ratio and number of SMTP packets altogether may help to reveal a Storm infected host. Note that the first two traffic attributes only apply to P2P packets, requiring filtering out other traffic on each host. In the next section, we will discuss the identification of P2P traffic without accessing packet payload and our methodology of using the three attributes to detect Storm bots.

5 Methodology

As described earlier, our approach consists of two stages. First, identifying the P2P packets and SMTP packets out of each host's traffic. Second, using distinguishing traffic attributes to determine if this host conducts Storm botnet activities by a machine learning technique. Both of the stages only obtain information from packet headers without examining the payloads.

We do not rely on the examination of packet payload to identify P2P traffic not only because of overhead but also due to privacy issues and payload encryption. Without looking for specific strings in the payload, one way of identification is looking at port numbers. This is applicable to SMTP packets, because the use of port 25 is the protocol standard. However, this method is not reliable for P2P case as more and more P2P applications allow for arbitrarily selected port nowadays. Thus, we need another approach to distinguish P2P from other traffic.

We use a set of heuristics proposed in [10]: in a time window, 1)Looking for source-destination IP pairs that concurrently use both TCP and UDP 2)Examining all source srcIP, srcport and dstIP,dstport pairs and seeking pairs that the number of distinct connected IPs is equal to that of distinct connected ports.1) is based on the observation that most of the P2P applications use UDP for communication and TCP for data transfer. 2) comes from the fact that each peer in a P2P network has an advertised IP, port pair for other peers' connection and that every other peer will choose a random source port to connect to the specific peer listening on that publicized port. In addition to 1) and 2), we also maintain a table to exclude non-P2P traffic to well-defined port numbers such as DNS (port: 53), NETBIOS (port: 135,137,139,445,), etc. The heuristics have been tested among million of traffic flows. It turns out that they can pinpoint more than 95%

of P2P with 8% - 12% false positives. Considering that the payload-size distribution and TCP/UDP ratio will not be affected by a few mislabeled packets, those heuristics are quite feasible.

Now that we are able to identify P2P and SMTP packets already, we proceed to the second stage: determining if a host is part of the Storm botnet. So far, the distinguishing attributes we examine are payload-size distribution, TCP/UDP ratio and number of SMTP packets. To have a better detection accuracy, we employs Support Vector Machine (SVM)[4, 9], a machine learning technique to derive a general metric in the form of a liner classifier:

$$Fx = \begin{cases} w^T x - b > 0, & \text{if x is a benign host} \\ w^T x - b \leq 0, & \text{if x is a Storm infected host (Storm bot)} \end{cases}$$

Here, w is a weight vector, b is a bias term and x is a vector of the three attributes mentioned above. Note that, the payload-size distribution can not be used directly. We need to change it to a quantitative value. Recall that all Storm variants generate similar patterns of payload-size distribution. We have averaged them and used the sample mean as the benchmark case. For any other distribution in a time window of monitoring, we calculate the summation of the absolute distance of each payload size from 0 to 1500 bytes, which is given by:

$$\sum_{i=0}^{1500} |y_i - s_i|$$

where y_i is the percentage of packets with payload i bytes of host y and s_i is that of Storm. We will detail our approach in evaluation section.

6. Evaluation

One of the challenges in our evaluation is the need for clearly labeled data set for both training and testing. Without knowing the identity of the data set, we can hardly validate the detection accuracy. On the other hand, capturing packet payloads at a large scale are not allowed in our network for privacy concern. Another problem is the running of Storm binaries, which can not be conducted freely on the Internet. To solve the above issues without violating privacy, we have constructed a small-scale controlled experiment environment with real and virtual machines.

We captured 8-hour traces with payloads from 4 hosts under regular use, each of which had run at least one P2P application for certain amount of time and had frequently accessed the Internet by web browsing, email checking and so on. The P2P clients include applications of Bittorrent, Gnutella, eMule, VoIP, PPStream, etc. covering widely used P2P protocols. We also executed 20 Storm binaries dating from Dec 2007 to May this year on our virtual machines. Each trace lasts 1 or 2 hours. We did not use earlier variants simply because most peers in hard-coded peer list were no longer active so that the bots could not join the Storm botnet and function. All of the 20 Storm instances can connect and communicate in the botnets. Note that those traces include payloads for validation purpose.

After obtaining the traces, we mixed the Storm traffic to host traffic to recreate the following scenarios. On top of non-P2P traffic a)Storm runs alone. b)Storm runs with 1 or 2 P2P applications simultaneously. c)1 or 2 P2P applications run alone. Recall that our work consists of two stages. In the first stage, P2P and SMTP packets are identified out of other traffic. We do so by applying a set of heuristics and keeping an exclusion table. Using those rules upon the traces, it turned out the identification rate was more than 90% on average, which is enough for our further investigation.

In the second stage, we need a classifier to detect Storm activities based on three traffic attributes. As mentioned earlier, we have employed SVM for classification. We used half of the dataset for training and half for testing. The three attributes are: payload-size distribution distance from benchmark ($Dist$), TCP/UDP ratio ($Ratio_{T/U}$) and number of SMTP packets (N_{smtp}). Our time window of monitoring is 20 minutes. In other words, every 20 minutes the three values are calculated and input to the classifier. We trained SVM on 40 pieces of data. We intentionally included the training set with noise free data of a), b) and c) for a better detection capability. The resulting classifier is given by:

$$\begin{aligned} F(x) &= w^T x - b \\ &= 1.0958 * Dist + 0.0511 * Ratio_{T/U} \\ &\quad -0.0003 * N_{smtp} - 0.0137 \end{aligned} \tag{1}$$

We found that $Dist$ is the most distinguishing attribute (table 3). The distances of those of Storm instances are all well below 1 from the benchmark. Even if Storm is mixed with Bittorrent or eMule, the trend is still similar. On the other hand, all P2P applications' distances are almost 2. Interestingly, the distances of them are quite close no matter how different the protocols are such as VoIP and eMule. This may be due to the following reason. As shown in their payload-size histograms, their packet sizes are not evenly distributed but concentrated at some values. For example, the 1460 bytes, which is commonly specified as the maximum segment size in TCP transmission.

As for $Ratio_{T/U}$, only under the scenario that P2P and Storm mix up to a large extent, the ratio will not be very distinguishable. N_{smtp} sometimes equals to 0 on a Storm infected host in a time window because of the sporadic spamming behavior. We keep this attribute in case that Storm traffic is well hidden to the background traffic and none of the first two attributes can discern.

Given the classifier, we tested the accuracy of the detection using the test set data including 40 pieces. 38 out of 40 or 95% were accurately classified. The only 2 misclassification was false-negatives. Both of them belong to an instance of Storm mixed with two P2P applications and no SMTP packets sent out in that time window. This is a challenging

Table 3. Payload-size distribution distances from that of a benchmark Storm

Storm	Dist		Normal P2P	Dist
valentine1	0.10396489		emule1	1.9224
valentine2	0.03989286		emule2	1.953911
codec1	0.03850181		bittorrent1	1.967289
codec2	0.05115983		bittorrent2	1.962575
sony	0.0479019		gnutella1	1.981724
disnisa	0.04303862		gnutella2	1.981937
storm+bittorrent	0.64127852		voip1	1.991548
storm+emule	0.12805476		voip2	1.965505

situation because all three attributes are similar to that of normal P2P. Nevertheless, in the following time windows due to the number of SMTP packets, they were caught finally. Since our detection is not a one-time shot but a periodic check, it is able to identify the Storm infected traffic once malicious behaviors are demonstrated. Also, we can expect that running two or more P2P clients and at the same time being infected by Storm is a rare case. Most of the time, only one of them is running.

In sum, the current preliminary result shows that our detection works well with 0% false positive and 8% false negative. We expect to conduct more experiments to test the detection strategy as more data become available to us.

7 Limitations

Attackers always try to evade the detection mechanism if possible. Our approach might be evaded if the Storm bots can successfully mimic benign P2P traffic. It is also possible that an infected host runs multiple P2P applications simultaneously which help to hide the Storm traffic.

From the attacker's perspective, one possibility is to make the payload-size distribution and TCP/UDP ratio similar to that of P2P applications. To do so, Storm bot needs to send out lots of large-size TCP junk data packets. However, there is a tradeoff here. By transferring large data packets, the bandwidth of a host will be significantly consumed, which may easily draw user's attention. As we know, bots intend to communicate in a stealthy way in order to keep alive. Also, as each Storm bot has to search for and take on specific task from the network such as sending out spam, it is not beneficial to send useless data packets to lower the efficiency of performing the task.

Another evasion is to slow down the communication of Storm in the hope of hiding it into the background traffic. This can be achieved especially when multiple P2P applications are running on a host. In that case, we may be able to detect it if SMTP packets are captured. If not, our detection can not work but at least raise the bar for the communication of the Storm bot.

8 Conclusion

The detection of P2P based botnets such as Storm at network level is a challenging problem due to its decentralized infrastructure. In this work, we have studied the traffic patterns of a number of Storm instances and several benign P2P applications. We propose an approach to identify the command and control traffic of Storm botnet without accessing packet payloads by 1)using a set of heuristics to extract P2P and SMTP packets out of each host's traffic and 2)distinguishing between the Storm and normal P2P traffic using a machine learning classification based on payload-size distribution, TCP/UDP ratio and number of SMTP packets. Evaluation on real-world traces shows that our approach is effective in detecting Storm traffic with low false-positive and false-negative. Detection as the first step, our proposed approach may work together with mitigation strategies to incapacitate the P2P based botnets.

References

[1] emule. http://en.wikipedia.org/wiki/EMule.

[2] Overnet. http://en.wikipedia.org/wiki/Overnet.

[3] J. R. Binkley and S. Singh. An algorithm for anomaly-based botnet detection. In *SRUTI'06: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet*, pages 7–7, Berkeley, CA, USA, 2006. USENIX Association.

[4] C.-C. Chang and C.-J. Lin. Libsvm – a library for support vector machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

[5] J. Goebel and T. Holz. Rishi: identify bot contaminated hosts by irc nickname evaluation. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 8–8, Berkeley, CA, USA, 2007. USENIX Association.

[6] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon. Peer-to-peer botnets: overview and case study. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 1–1, Berkeley, CA, USA, 2007. USENIX Association.

[7] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.

[8] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling. Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. In *In Proc. First USENIX Workshop on Large-scale Exploits and Emergent Threats (LEET)*, 2008.

[9] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf.

[10] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy. Transport layer identification of p2p traffic. In *In Proc. Internet Measurement Conference (IMC)*, 2004.

[11] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale botnet detection and characterization. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 7–7, Berkeley, CA, USA, 2007. USENIX Association.

[12] C. Livadas, R. Walsh, D. Lapsley, and W. Strayer. Usilng machine learning technliques to identify botnet traffic. In *Proceedings of 2006 31st IEEE Conference on Local Computer Networks*, November, 2006.

[13] P. Porras, H. Saidi, and V. Yegneswaran. a multi-perspective analysis of the storm(peacomm)worm. Technical report, SRI, 2007.