Detecting and blocking P2P botnets through contact tracing chains

Zhiyong Huang* and Xiaoping Zeng

College of Communication Engineering,
Chongqing University,
Chongqing, 400030, P.R. China
E-mail: zyhuang@cqu.edu.cn E-mail: zxp@cqu.edu.cn
*Corresponding author

Yong Liu

Department of Electrical and Computer Engineering,
Polytechnic Institute of NYU,
5 Metrotech Center,
Brooklyn, NY 11201, USA
E-mail: yongliu@poly.edu

Abstract: Peer-to-peer (P2P) botnets have recently become serious security threats on the internet. It is difficult to detect the propagation of P2P botnets by isolated monitoring on individual machines due to its decentralised control structure. In this paper, we propose a contact tracing chain-based framework to detect and block P2P botnets by tracing contact behaviours among peers. In the proposed framework, the contacts of peers with suspicious symptoms are traced and tracing chains are established to correlate contacts among peers with their abnormal symptoms. Peers are confirmed with infections when the length of contact tracing chain that they belong to reaches a preset threshold. Under this framework, we develop different detection, tracing and immunisation strategies. Through numerical simulations, we demonstrate that the proposed contact tracing framework can quickly detect and block the propagation of P2P botnets.

Keywords: botnet; P2P; worm; contact tracing; transmission chain; entropy; threshold; detection; blocking; immunisation; simulation; protocol.

Reference to this paper should be made as follows: Huang, Z., Zeng, X. and Liu, Y. (2010) 'Detecting and blocking P2P botnets through contact tracing chains', *Int. J. Internet Protocol Technology*, Vol. 5, Nos. 1/2, pp.44–54.

Biographical notes: Zhiyong Huang is currently a PhD candidate in the College of Communication Engineering, Chongqing University. He received his MS from Chongqing University in 2004. In 2007, he was a Visitor Researcher at the Department of Electrical and Computer Engineering, Polytechnic Institute of NYU. He currently works at the College of Communication Engineering at Chongqing University. His research interests include network security and network modelling, simulation and analysis.

Xiaoping Zeng received his BS, MS and PhD from Chongqing University in 1982, 1987 and 1996, respectively. He is currently a Professor of College of Communication Engineering at Chongqing University. His research interests include network security, control and systems methods in communication networks. Until now, he has published more than 70 technical papers on relevant field.

Yong Liu has been an Assistant Professor at the Electrical and Computer Engineering Department of Polytechnic Institute of NYU since 2005. He received his PhD from Electrical and Computer Engineering Department at the University of Massachusetts, Amherst in 2002. He received his Masters and Bachelors in the field of Automatic Control from the University of Science and Technology of China in 1997 and 1994, respectively. His general research interests lie in modelling, design and analysis of communication networks. His current research directions include robust network routing, peer-to-peer IPTV systems, overlay networks and network measurement. He is a member of IEEE and ACM.

1 Introduction

In recent years, botnets have been frequently utilised by attackers to launch malicious attacks on the internet. A botnet is a network of compromised peers (bots), which are controlled by an attacker to launch DDOS attacks, distribute spam e-mails, etc. Using botnets, attackers can launch attacks from thousands or even millions of distributed bots. Early botnets adopted the client-server-based system architecture. All bots in a botnet connect to some command and control (C&C) servers. Bots receive commands from C&C servers using protocols like Internet Relay Chat (IRC). One drawback of this client-server architecture is that, due to the heavy traffic between servers and bots, the C&C servers can be easily detected by defenders. And the entire botnet will be shut down once the C&C servers are blocked. More recent botnets employ the peer-to-peer (P2P) system architecture. In a P2P architecture, there is no fixed C&C servers. Every bot in the botnet acts as both client and server. Bots communicate and exchange information with each other using existing or customised P2P protocols. Attackers disguise themselves as normal peers and disseminate their commands to all peers in the same P2P botnets. Due to the distributed nature of P2P botnets, it is very challenging to detect and block them. In this paper, we propose a novel contact tracing chain-based approach to the detection and blocking of P2P botnets.

The main challenges in botnet detection are:

1 P2P botnet are now under widespread development. Some P2P botnets use existing P2P protocols, while others develop customised protocols (Grizzard et al., 2007). For example, *Phatbot* (Stewart, 2004) uses code from the WASTE project to implement P2P, but *Nugache* (Lemos, 2006) and *SpamThru* (Stewart, 2004) use their own P2P protocols for communication between peers. Most of the customised protocols are encrypted. It is very difficult to detect the signalling among bots in P2P botnets through protocol analysis.

2 To escape from traffic analysis-based detection, some intelligent P2P botnets deliberately changed their infection tactics [e.g., reducing the size of their network (Higgins, 2007)]. It is hard to distinguish a P2P botnet from a normal P2P network by looking into the traffic volume and contacts between peers.

3 Since there is no centralised C&C servers, P2P botnets are very 'resilient'. A bot can receive C&C signal from any of its peering neighbours. When the number of peering contacts of each peer is reasonably large, the whole P2P botnet has very robust connectivity and can remain connected even after a large portion of bots are detected and removed from the botnet.

To address the previously described challenges, we develop a contact tracing chain-based P2P botnet detection and immunisation approach. Contact tracing has been successfully applied in disease control (Huerta and Tsimring, 2002; Hymana et al., 2003; Eames and Keeling, 2003). In our paper, we attempt to use contact tracing chains to detect bots.

The contributions of our work are summarised as follows:

1 We apply the concepts of contact tracing and transmission chain to the detection and immunisation of P2P botnets. We developed a tracing-chain-based framework that can efficiently identify and block P2P botnets.

2 Simple threshold-based contact detection scheme cannot detect intelligent botnets. We develop a novel entropy-based detection approach to distinguish botnets from normal P2P networks.

3 We propose several tracing-chain algorithms to strike the right balance between the efficiency and the accuracy of contact tracing.

4 We implement a discrete time simulator and conduct extensive simulations to study the efficiency and robustness of the proposed contact tracing framework under different network and system settings.

2 Background and related works

2.1 P2P botnets

P2P botnets are quickly becoming one of the most significant threats on the internet today. P2P botnets are networks of compromised peers controlled by attackers through P2P communication protocols. P2P botnets have robust network connectivity. They hide their activities through signal encryption and traffic dispersion. Consequently, it is very difficult to trace and shut down P2P botnets.

Grizzard et al. (2007) described the development of P2P botnet and introduced several P2P control architectures, such as *Slapper*, *Sinit*, *Phatbot* and *Nugache*. Many P2P networks are becoming the favourite places for malware to spread (Dhungel et al., 2007; Liang et al., 2005). Worm is widely adopted by P2P botnet in the wild due to its automatic propagation characteristics.

As an example, Storm Worm botnet is the first major botnet to use P2P network for C&C (Porras et al., 2007). Similar to other e-mail worms (e.g., Loveletter/ILOVEYOU and Bagle) (Holz et al., 2008), the e-mail body contains an embedded link or an attachment with names such as 'FullVideo.exe', 'Video.exe' and 'FullClip.exe'. Attackers use social engineering techniques to pretend to be a legitimate e-mail and trick the recipient into opening the attachment or clicking on the embedded link. Once a victim attempts to open the attached file or click the baleful link, it will become an infected peer. After the Trojan installs the initial infection files, the victim will attempt to connect peers in the Storm Worm botnet. Subsequently, it will download the full payload and become a real bot under the control of the botmaster.

Within the propagation stage, the number of some net-packets (e.g., ICMP, UDP and SMTP) sharply increased (Kang et al., 2009), the high *contact rate* is the most important characteristic for worm detection we considered in this paper; it will be introduced in Section 4.

Some works have been done by researchers on P2P botnets defences. Zhou et al. (2005) gave a 'first look' at P2P worm defence. They proposed a framework where a small fraction of guardian nodes are employed to detect control flow hijacks of vulnerable programs using schemes similar to those in Crandall and Chong (2004), Newsome and Song (2005) and Suh et al. (2004), the approach which deployed the same P2P network as P2P worm to propagate alerts is similar to our strategy for building tracing chain. Gu et al. (2008) presented a general detection framework (BotMiner prototype system) that is independent of botnet C&C protocol and structure, and requires no a priori knowledge of botnets, the key of this approach is focused on the net-behaviour detection; Holz et al. (2008) estimated the total number of the bots through infiltrating and analysing in-depth the Storm Worm botnet and mitigates this botnet through disrupting the communication channel, such as eclipsing content, polluting; this paper described the typical characteristic of Storm Worm. Our system adopted contact tracing scheme based on the net-behaviour detection of single node to identify the infected nodes.

2.2 Contact tracing

Contact tracing is a classic epidemic control method. It is frequently used to combat many infectious diseases (such as TB and AIDS) and new invading pathogens (such as SARS). When an individual is identified as having a communicable disease, any one has contacted with the individual has a high probability to be infected. Tracing based on contact history can quickly identify and block the propagation of infections. Eames and Keeling (2003) used the pairwise approach and full stochastic simulations to investigate the efficiency of contact tracing in disease control. Hymana et al. (2003) used random screening and contact tracing to reduce the spread of HIV. Huerta and Tsimring (2002) developed the mean-field model of contact tracing for the case of random graphs. The idea of contact tracing has recently been applied to network security field. Xiong (2004) brought the concepts of contact tracing and transmission chain into e-mail network worm and virus control. In our paper, this approach is applied to detect and block P2P botnets.

3 Contact tracing framework

In this section, we present the overall framework of contact tracing for P2P botnets detection and control.

Similar to epidemic disease control, we classify peers into different states according to their contact records and symptoms. In P2P botnets, a contact is defined as the establishment of a connection between a pair of peers. The infectious symptom on a peer is defined as contacts at a rate higher than a preset threshold. We classify peers into the following five possible states:

1 *Normal:* A peer that has no infectious symptom is in the normal state.

2 *Connected:* A peer that has been contacted by a probable peer or a suspicious peer is in the connected state.

3 *Suspicious:* A peer that has infectious symptom is in the suspicious state.

4 *Probable:* A suspicious peer is confirmed with infection and declared as probable if it is on an established contact tracing chain (see details below).

5 *Immunised:* A probable peer that has been cleaned and patched will change to the immunised state.

Figure 1 Diagram of peer state transitions

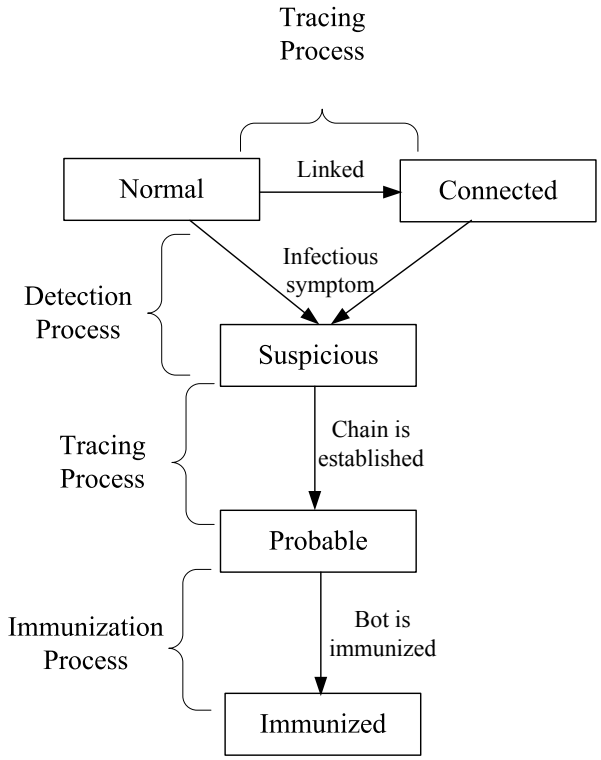


Figure 1 shows the peer state transition:

1 *Normal ⇒ suspicious:* A normal peer changes to a suspicious peer when it first has the infectious symptom.

2 *Normal ⇒ connected:* A normal peer changes to a connected peer when it is first contacted by a suspicious peer or a probable peer.

3 *Connected ⇒ suspicious:* A connected peer changes to a suspicious peer when it has infectious symptom.

4 *Suspicious* ⇒ *probable:* A suspicious peer changes to a probable peer when the tracing chain which it belongs to is established.

5 *Probable* ⇒ *immunised:* A probable peer changes to an immunised peer when it is cleaned and patched.

The key of contact tracing-based detection is to monitor and control peer state transitions. It consists of three components: *detection process*, *tracing process* and *immunisation process*. The detection process is to detect peer infectious symptoms. The tracing process is to track the contact history of suspicious peers and establish contact tracing chains to confirm peer infections. The immunisation process is to clean and patch the infected peers. At first, the detection process monitors the contact rate of peers. If a peer sends out more than a preset threshold M connections within an interval δ, it will be identified as an abnormal peer and its state will be converted from normal to suspicious. Secondly, the tracing process tracks the contacts of all suspicious peers through recording links between a suspicious peer and any peer that it contacts with. If a normal peer is contacted by a suspicious peer, its state will be converted to connected. Consequently, if any of those contacted peers is detected with infectious symptom, that peer will be also classified as suspicious and the new contact links will be recorded by the tracing process. If indeed the infection propagates through peer contacts, the tracing process will identify a chain of contact links. A complete tracing chain is established when the length of contact tracing chain reaches a preset threshold K and the tracing process can declare confirmed infections for all peers on the tracing chain. The peers on the chain will convert their state from suspicious to probable. Immunisation will then be applied to the newly identified probable peers.

In the following sections, we present the detailed designs of *detection*, *tracing* and *immunisation processes*.

4 Symptom detection process

In the previous section, we talked about a simple threshold-based symptom detection algorithm. However, the network conditions are so complex that we cannot always use a single fixed threshold to distinguish between legal traffic and illegal traffic. Many worms adopted new tactics to escape from simple threshold-based detection (e.g., low-rate contact strategy). It will lead to a high false alarm rate in the detection process (e.g., some peers running normal P2P software are mistakenly considered as suspicious bots by the simple threshold-based detection with a low threshold, whereas some bots adopted low-rate contact strategies are mistakenly missed by the simple threshold-based detection with a high threshold). Even though the transmission chain approach can tolerate some false alarms on individual contact tracing links[1], we still want to improve the detection process to bring down the false alarm rate of the detection process. In this paper, we propose to distinguish P2P botnets from legitimate P2P systems by investigating major contact-level characteristics.

If we have a long observation period of T, we divide T into n time slots. $V(k)$ denotes the number of connected peers in the time slot k. We record $p(k)$, the ratio of the number of newly connected peers within time slot k to the total number of connected peers in the whole observation period T. For P2P botnets, $p(k)$ takes large values only for the infection time slots. For legitimate P2P system, $p(k)$ is more uniformly distributed across all time slots in the observation period. More specifically, we can calculate the entropy (Shanon, 1948) of $p(k)$:

$$H(p) = -n\sum_{k=1}^{n} p(k)\log_2 p(k). \tag{1}$$

We can distinguish P2P bots from normal P2P systems by computing their entropy values of $p(k)$ in the observation period. Normal P2P systems have uniformly low contact rate and thus they have a high entropy value. Bots usually have a similar structure of worm propagation. The entropy value of $p(k)$ is low.

To further improve the detection accuracy, we augment the entropy-based detection by investigating additional peer information, such as the contact ports, the arrival time, the active time, the frequency and content length of contact (Husna et al., 2008).

5 Contact tracing chain establishment

The core of the proposed contact tracing framework is to track the contacting behaviours of suspicious peers and establish contact tracing chains to confirm infections. The algorithms used to build tracing chains determine the efficiency and the accuracy of botnet detection and blocking. On one hand, the algorithms should be efficient and can quickly identify and block ongoing botnet propagation. On the other hand, the algorithms should be accurate and raise as few false alarms as possible. In this section, we present three tracing chain algorithms to trade off the efficiency and accuracy.

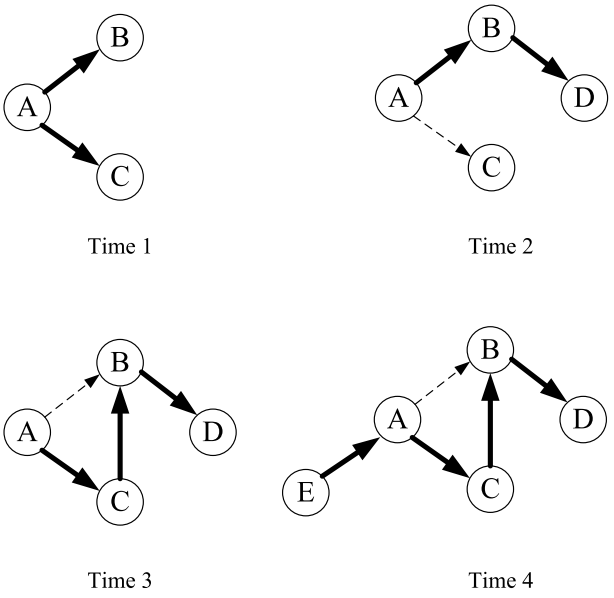
5.1 Basic chain algorithm

We first introduce a basic algorithm. In this basic algorithm, each node is initially assigned with layer ID 0. For the first time a peer is contacted by a peer with suspicious symptom, it changes its layer ID from 0 to one plus the layer ID of the suspicious peer. A peer's layer ID will not be changed for later contacts from other suspicious peers. A trace chain is established if some peer's layer ID reaches the chain length threshold K. This algorithm can be easily implemented. However, it can only identify simple contact tracing chains consisting of contact links discovered sequentially.

For the network in Figure 2, the contact threshold is $M=1$ and the tracing chain threshold is $K=3$. At time 1, node B and C are contacted by node A, A is root node whose layer ID is 0. At time 2, node B contacts node D, node B changes its layer ID to 1. At time 3, node B is contacted by node C, but B will keep its layer ID of 1. At time 4, node E contacts node A. Node A will not change its ID and this infection will be ignored. No tracing chain is detected by this algorithm with threshold $K=3$ even though we do have a chain of five peers and four of them have suspicious symptoms.

This will largely slow down the botnet detection speed.

Figure 2 Establishment of tracing chains in a simple network



5.2 Longest-chain algorithm

To address this problem, we present another algorithm that improves the detection speed by identifying the longest contact tracing chain for each peer. We build the *contact graph* with nodes being peers and links being contacts initiated by suspicious peers. A contact tracing chain then corresponds to a path in the contact graph. When a node is contacted by multiple suspicious nodes, this node is placed simultaneously on multiple paths/chains in the contact graph. The longest-chain algorithm declares confirmed infection for a peer if the length of the longest contact path/chain that the peer is on reaches the chain length threshold K.

To implement this algorithm using peer layer ID, the layer ID of each peer is no longer fixed after the first contact from a suspicious peer. When peer A is contacted by a suspicious peer B, peer layer IDs are updated as follows: if B's ID is larger than or equal to A's ID, set A's ID to one plus B's ID, also update the layer IDs of A's downstream peers in contact chains. More specifically, each node i has two state variables: (P_i, L_i), where P_i denotes the parent node of i, in other words, P_i infected i, L_i is the layer of node i. Initialising $L_i=0$ and $P_i=i$. Assuming K is the preset chain length threshold, S is the set of nodes. When node m is contacted by a suspicious node $n\in S$, the longest-chain algorithm updates node states as in Table 1.

Table 1 Longest-chain tracing algorithm

Algorithm	Node m is infected by node n
Procedure: Updating the Tracing Chain	
if $(L_n \geqslant L_m)$ then	
	$P_m = n.$
	$L_m = L_n + 1.$
	Updating the other nodes on this chain.
end if	
Procedure: Confirming the Tracing Chain.	
for (all nodes r, $r\in S$) do	
if $(L_r \geqslant K)$ then	
The suspicious nodes on this chain are confirmed bots.	
end if	
end for	

Now we go back to the example in Figure 2. The algorithm works the same as the basic algorithm up to time 2. At time 3, node B is infected by node C, they all have layer ID 1, so node B's ID is changed to 2 according to our algorithm, node D's layer ID remains 0 because it has no suspicious symptom. At time 4, root node A is infected by node E, node E replaces node A to become the root node, correspondingly, node A's layer ID is set to 1, node C and B's layer IDs are set to 2 and 3, respectively. The length of this tracing chain reaches $K=3$. Thus, a complete tracing chain is established and peer E, A, C and B are all confirmed with infections.

5.3 Causal chain algorithm

The longest-chain algorithm has a good tracing speed, but it has a weakness. It ignored the temporal order of contact links. As a result, the actual infections among peers do not necessarily follow the order that peers appear in a tracing chain. For the previous example, E is the root node, but since A, B and C all have symptom before E does. There is no causal relation between E's symptom and symptoms on the other four nodes.

Basic algorithm has considered the causal relations between infections. However, for each peer, it only considers its first infection link. Its detection speed is too slow. To improve the detection efficiency while keep the causal relation along tracing chains, we propose a *causal chain algorithm* that combines the advantages of the basic algorithm and the longest-chain algorithm.

Each node $r\in S$ can potentially initiate the infection and be the root for a tracing chain. At the same time, each

node $i \in S,\ i \neq r$ can potentially be infected by the infection originated at r and be placed on the tracing chain rooted at r. Therefore, each node should keep $|S|$ layer IDs, one for each chain rooted at any $r \in S$. The state of each node i can be described by a tuple vector $\{(L_i(r), P_i(r)), r \in S\}$, where $L_i(r)$ denotes the layer of node i relative to root node r, and $P_i(r)$ denotes the parent node of node i relative to root node r. The peer states are initialised as follows:

$$L_i(r) = \begin{cases} -1 & \text{if } i \neq r \\ 0 & \text{if } i = r \end{cases}; \quad P_i(r) = i \tag{2}$$

$L_i(r) = -1$ denotes node i has not been infected by any nodes on the tracing chain rooted at r. When $i = r$, $L_i(r) = 0$ denotes every node is at layer 0 relative to itself; and $P_i(r) = i$ denotes node i is the root node.

Table 2 Causal chain algorithm

Algorithm Node i is infected by node j
Procedure: Updating the Tracing Chain.
$P_i(j) = j.$
for (all nodes r, $r \in S$) do
if $(P_j(r) \neq j$ and $L_i(r) = -1)$ then
$L_j(r) = L_{P_j(r)}(r) + 1.$
$P_i(r) = j.$
end if
end for
Procedure: Confirming the Tracing Chain.
if $(L_j(r)$ reaches the preset $K)$ then
while (j is not root node) do
node j and $P_j(r)$ are confirmed bots.
node j'state is initialised.
$j = P_j(r).$
end if

When node i is infected by node j, the algorithm can be described as:

1 node i obtains all the layer information of node j and makes node j as its parent node relative to root node j

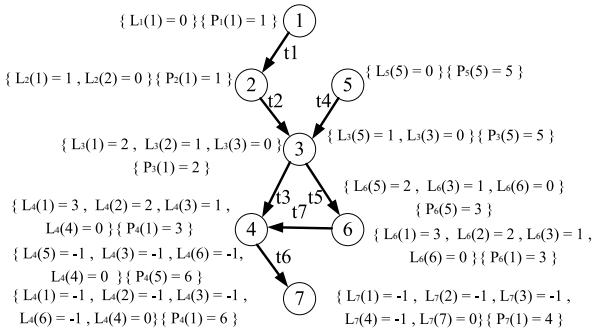
2 for any root node r that j connects to, if node i has not been added to the tracing chain rooted at $r(r \neq j)$, update the layer information of node j and make node j as i'parent node relative to root node r

3 when $L_i(r)$ reaches the preset threshold K, the nodes on the tracing chain rooted at node r are confirmed with infections.

Note the state of each confirmed node must be initialised immediately because they have probability to be contacted through another tracing chain rooted at the same node r. The detailed algorithm is described in Table 2.

Figure 3 illustrates an example of the causal chain algorithm. In the example, there are seven nodes. The contact links between nodes are labelled with the time that contact happened, with $t_1 < t_2 < t_3 < t_4 < t_5 < t_6 < t_7$. Each node is labelled with non-trivial state variables.

Figure 3 Causal chain information passing among peers



There are three potential tracing chains. Chain 1 consists of nodes 1, 2, 3, 4 and 7. Chain 2 consists of nodes 1, 2, 3, 6 and 4. Chain 3 consists of nodes 5, 3, 6 and 4. If the chain length threshold is $K = 3$, chain 1 and chain 2 have reached the threshold. Thus, nodes 1, 2, 3, 4 and 6 are confirmed bots, note that because $t_6 < t_7$, there is no causal relation between node 4's symptom and node 6's symptom, node 4 is confirmed through chain 1. Node 7 cannot be confirmed bot because it does not have infectious symptom. It should be noted that node 6 was infected by node 3 at t_5. Since node 3 has been infected twice by two different nodes at t_2 and t_4, and $t_2 < t_4 < t_5$, thus, node 3 passes two sets of layer information of chains 2 and 3 to node 6, and node 6 is confirmed with infection through chain 2.

5.4 Dynamic chain length threshold

In the tracing chain framework, the chain length threshold K is a very important parameter. To reduce false alarms, we should use a large K; and to increase the discovery speed, we can use a small K. In this section, we propose to dynamically adjust K at different stages of botnet propagation to strike the right balance between the detection speed and false alarm rate.

The typical botnet propagation period can be divided into three stages: *early*, *middle* and *late*. Let I be the number of already infected nodes, ΔI be the new infection rate, which is defined as the number of newly infected peers within unit time. Each period can be characterised as follows:

1 *Early stage:* The number of infected peers I is small and the new infection rate ΔI is low.

2 *Middle stage:* The number of infected peers I is medium and the new infection rate ΔI is high.

3 *Late stage:* The number of infected peers I is large and the new infection rate ΔI is low.

Our strategy is to use different K at different stages. We use large K when the infection degree is low for accuracy. And we use small K when the infection degree is high for speed.

6 Immunisation strategies

Immunisation is to use antivirus software to clean and patch the confirmed bots. Our focus is not on developing bot cleaning and patching tools. Our goal is to investigate different immunisation strategies and their effectiveness.

We consider two immunisation strategies against P2P botnets. When bots are discovered, the immunisation should be conducted as soon as possible. Due to resource constraints, not all bots can be immunised at the same time. The order that bots are immunised affect the effectiveness of the immunisation.

The simplest strategy is *random immunisation*. All discovered bots are treated equally and have the same probability to be immunised. Each bot is selected at random to be immunised. This method does not differentiated nodes and is easy to be implemented.

A more effective strategy is *preferential immunisation*, which includes two steps. At first, the probable nodes confirmed by contact tracing chain are picked out. Xiong (2004) has pointed out the probable node has larger probability to infect other nodes. Secondly, we can choose the most-connected nodes from these identified probable nodes to immunise. This is because nodes with better network connections are typically connected to more nodes; they are more dangerous in infecting other nodes. This method is different from the random immunisation. It differentiates bots based on their infection potentials. We will give priority to bots with larger probability and apply immunisation to them first. Similar preferential immunisation approach has been proposed by Wang et al. (2000) in the context of immunisation of computer virus. They have shown that preferential immunisation can significantly slow down virus propagation. We will study the performance of preferential immunisation of botnets using simulations in Section 7.

7 Experiments

We developed a time-stepped simulator to study the performance of contact chain tracing algorithm. It simulates botnet propagation and the establishment of contact tracing chains by implementing algorithms described in the previous sections.

To clearly monitor the propagation of botnet, we only consider one botnet in a simulated network with $S = 6{,}400$ nodes. We define P_i as the probability that node i is infected by other nodes. We assume each node's behaviour is independent of others. Similar to Zou et al. (2007), when the peer population S is large, we can approximate P_i by a Gaussian random variable with distribution $N\left(\mu_P, \sigma_P^2\right)$. In our experiments, $P_i \sim N(0.07, 0.0004)$. For most simulation experiments, we perform 50 simulation runs and report the average values.

We ignore network delays and assume immediate detection of high rate contact symptom of an infected node. Therefore, an infected node will be traced immediately. Initially, the only infected node is suspicious and all other nodes are in the normal state. At each consequent time step, infected nodes infect other nodes with probability P_i. Once a node is contacted by a suspicious node, it will be added to contact tracing chain according to the algorithms introduced in Section 5. When a complete tracing chain is established, all nodes on the chain will be confirmed with infection. Then the immunisation strategies in Section 6 will be applied to clean and immunise confirmed bots.

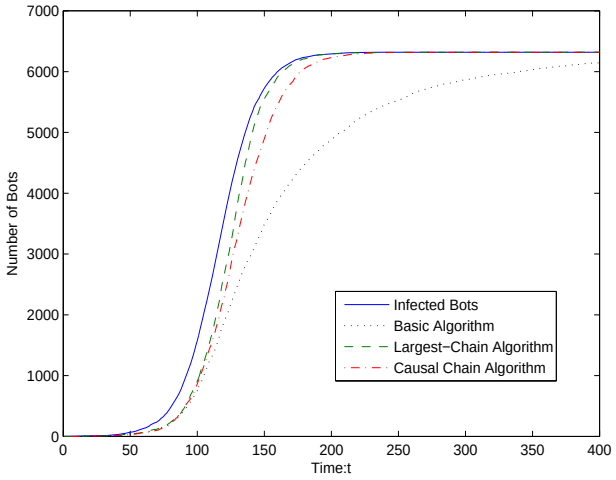
7.1 Tracing chain efficiency

We first study the efficiency of tracing chain algorithms using simulations. We compare the bot detection speed of three chain algorithms introduced in Section 5. We then study how the tracing chain efficiency depends on three important factors in practice: chain length threshold K, node coverage ratio of the tracing chain scheme and botnet node degree.

7.1.1 Comparison of chain algorithms

Figure 4 shows the performance comparison of three algorithms: basic algorithm, longest-chain algorithm and causal chain algorithm. The number of infected nodes in the system and the numbers of bots detected by each algorithm are plotted as time evolves. There is not much difference among the three algorithms in the initial stage of the botnet propagation. Once the propagation accelerates, it is found that the number of detected bots with the longest-chain algorithm and causal chain algorithm are much more than the basic algorithm. Eventually, all bots can be detected by all three algorithms. Thus, the longest-chain algorithm and causal chain algorithm have much higher bot detection speed than the basic algorithm. Since the causal chain algorithm considers the causal relations between contacts and infections, its accuracy is better than the longest-chain algorithm. From the simulation results, we can conclude that causal chain algorithm has the best overall efficiency and accuracy among the three algorithms; unless, all simulations in the remaining of this section use the causal chain algorithm.

Figure 4 Comparison of three tracing chain algorithms (see online version for colours)



7.1.2 Chain length threshold

The chain length threshold trade-offs the detection speed and accuracy. We have introduced a dynamic chain length threshold algorithm in Section 5.4. In our simulation, as listed in Table 3, the threshold K is dynamically chosen from [3, 9] as a function of the current infection degree ΔI.

Table 3 Dynamic chain length table

	$K=3$	$K=4$	$K=5$	$K=6$
ΔI	[50, ∞)	[35, 50)	[24, 35)	[14, 24)
	$K=7$	$K=8$	$K=9$	
ΔI	[9, 14)	[4, 9)	[0, 4)	

Figure 5 Comparison of discovery simulation with different K (see online version for colours)

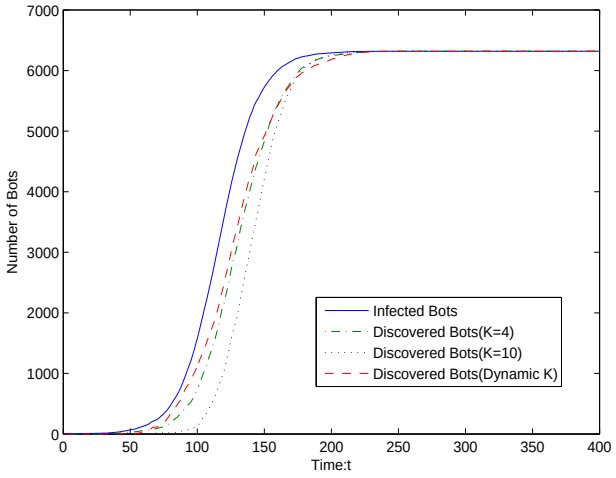


Figure 5 compares the performance of dynamic chain length threshold with fixed length threshold at different $K=4$ and $K=10$. Obviously, we see that the dynamic threshold has the best performance among three cases. When the infection degree is low, it has a lower tracing speed; when the infection degree is high, its tracing speed accelerates dramatically.

7.1.3 Node coverage of tracing chain scheme

As discussed above, we assumed that every peer has installed the detection software. In reality, only a subset of users will install the detection software. We call users who installed the software registered peers and users who did not install the software unregistered peers. Obviously, only the registered peers can be detected and traced by our system. We must take into account this factor when using tracing chain. It is clear that the number of registered peers affects the efficiency of bot detection. The ratio of registered peers is a ratio of the registered peers to the total number of peers in the system:

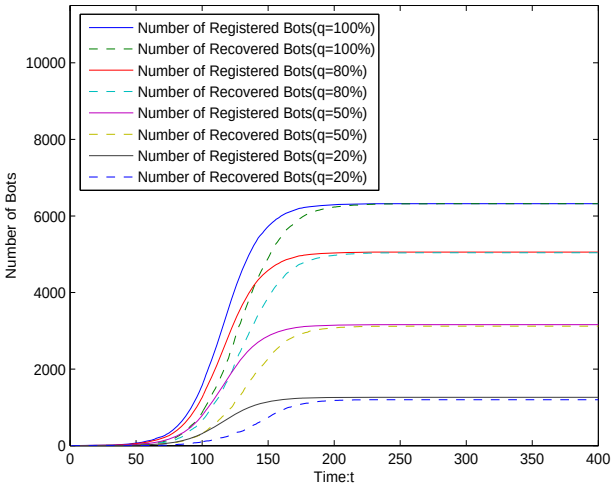
$$q = \frac{number\ of\ registered\ peers}{total\ number\ of\ peers} \tag{3}$$

We define the detection ratio as:

$$R = \frac{number\ of\ detected\ bots}{number\ of\ registered\ bots} \tag{4}$$

We conduct simulations using the causal chain algorithm with $K=4$. Simulations are conducted for $q=100\%$, $q=80\%$, $q=50\%$ and $q=20\%$, respectively. Figure 6 shows that the number of registered bots can affect the efficiency of bots detection. For the case of $q=100\%$, all the peers are registered and all the bots can be detected by the tracing chain algorithm. For the case of $q=20\%$, there are about 60 registered bots that cannot be detected because in this case, there are about 80% peers are not registered. This leads to the situation that a few nodes are isolated and cannot be traced. But the number of isolated nodes is insignificant relative to the detected bots. The detection ratio is still high. In conclusion, the tracing chain algorithm is pretty robust against low node coverage.

Figure 6 Efficiency of discovery simulation with different ratio of register (see online version for colours)

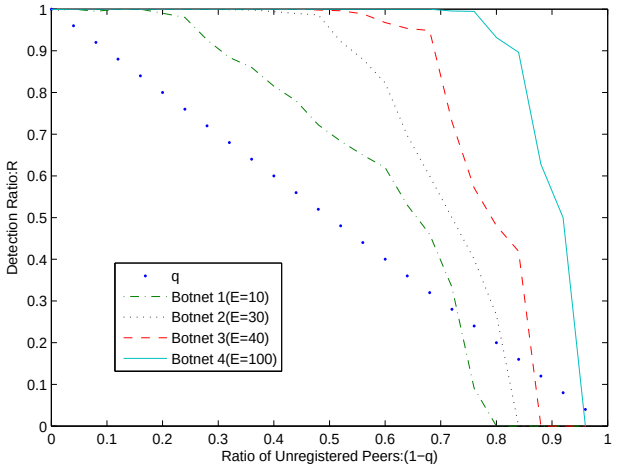


7.1.4 Node degree of bots

The establishments of tracing chains are driven by contacts among peers. The node degree of bots in a botnet directly affects the efficiency of tracing chain algorithm. We generate random P2P botnets with the average node degree of 100, 40, 30 and 10, respectively. For each average node degree, we generate a group of 25 random network instances and run the simulation on each instance. For each simulation, we gradually vary the ratio of registered peer q from 1 to 0.04. At each q value, we obtain the bots detection ratio R in each instance. We then calculate the average detection ratio R for the group of botnets with the same average node degree.

Figure 7 shows the average detection ratio at different average degrees. Botnet group 1 has an average degree of $E=10$, botnet group 2 has an average degree of $E=30$, botnet group 3 has an average degree of $E=40$ and botnet group 4 has an average degree of $E=100$. For group 4, when q decreases gradually, the detection ratio stays at a high level $(R>90\%)$ as long as $q>0.1$. Botnet group 1 has a low average degree. The detection ratio is high $(R>90\%)$ if $q>0.7$. But the detection ratio decreases quickly when q gets smaller and reaches 0 when $q>0.2$. The performance for groups 2 and 3 lie in-between groups 1 and 4. We can see that the average node degree of botnet can affect the efficiency of detection. The higher node degree, the higher the detection efficiency. From the attacker point of view, higher bot degree increases the resilience of botnets to the removals of individual bots. However, our tracing chain algorithm can also exploit high bot degree for efficient and robust detection.

Figure 7 Detection efficiency at different node degrees (see online version for colours)



7.2 Immunisation simulation

To study the performance of immunisation strategies, we simulate a static botnet with all nodes join the system at time 0 and only one node is infected initially. No node leaves the system during the simulation. We simulate two different immunisation strategies: in the first case, at each time step, we randomly choose infected nodes to immunise with a homogeneous probability p; in the second case, we use preferential strategy to immunise the top p fraction of probable peers with the highest connectivity.

Figure 8 Comparison of immunisation strategies in static botnets (see online version for colours)

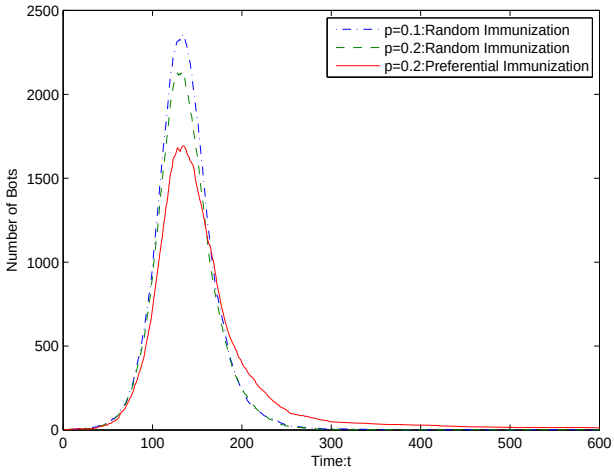


Figure 8 shows that both strategies can completely break the botnet. Comparing two strategies, the preferential immunisation is much more effective than the random immunisation. It can always control the size of botnet at a very low level. We vary p to simulate the effect of immunisation with different immunisation ratio. The result is that, with $p=0.1$, the botnet reaches a larger size than the botnet with $p=0.2$. Therefore, we can conclude that the high immunisation ratio and good immunisation strategy lead to efficient immunisation.

8 Implementation issues

A tracing chain mechanism includes three key steps:

1 identify the individual peer with symptoms

2 establish the transmission chain through contact tracing

3 immunise the infected peers.

In this section, we discuss the scheme to implement those steps.

ISPs are in a unique position to monitor and trace contact behaviours of users in their networks, on which can install servers to detect abnormal traffic patterns of users and establish tracing chains among users. The drawback of this approach is that ISP servers have to monitor a large number of users and the workload can be very high. In addition, to trace connection traffic across different ISPs, servers under different administrations have to cooperate each other (Xiong, 2004). In our paper, we adopt another architecture called distributed sensor architecture.

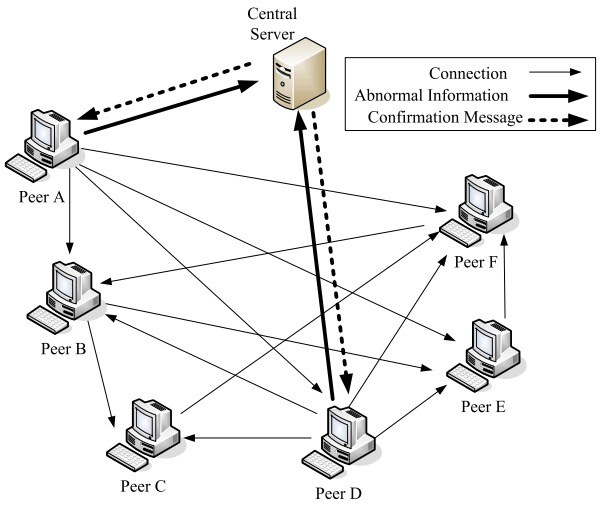
In this architecture, each computer is installed with a symptom detection software, called sensor. Distributed

sensors continuously monitor traffic on their host computers and record connection level information (such as destination IP addresses and durations of outgoing connections) in every time interval. When a peer behaves normally, the sensor only keeps the monitored information locally. Once a sensor detects suspicious activities, it will fire and report the monitored abnormal information to a central server. The central server acts as a tracker which can be configured on the ISP servers. It collects abnormal information from all sensors in the network and applies contact tracing algorithms to establish infection chains. Once a chain is established, all the peers on the chain will receive infection confirmation messages from the server. Infected peers will be cleaned and immunised by the immunisation software.

Figure 9 shows a distributed sensors-based detection system, which consists of a central server and distributed sensors A, B, …, F. Each sensor must install detection software and become a registered user. All registered users communicate with the central server. As shown in Figure 9, when the contacts number of peer A and D reaches $M = 4$ in a time interval, sensors on A and D connect with the server and the tracing process starts.

This approach reduces the workload of central server because the server only needs to process a very limited amount of data from distributed sensors.

Figure 9 Distributed sensors architecture (see online version for colours)



9 Conclusions

In this paper, we proposed a novel contact tracing chain-based framework to detect and block the propagation of P2P botnets. Our framework consists of three components: detection, tracing and immunisation. We developed new symptom detection algorithms to distinguish P2P Bots from legitimate P2P systems and studied three contact tracing chain algorithms to trade off the tracing speed and false alarm rate. The performance of the proposed algorithms was evaluated using discrete-time simulations. We demonstrated that the proposed contact tracing framework can quickly detect and block the propagation of bots in dynamic P2P network environment.

Of course, contact-tracing chain system is not perfect. As the bots are confirmed by establishing contact-tracing chain, once the tracing chain is attacked by botmasters, such as a DOS attack occurs in the middle of a chain, the number of detected bots will decrease greatly although our tracing algorithm can maintain a high bots detection ratio as discussed in Section 7.

In our future work, we will refine the proposed detection and tracing algorithms to improve the detection efficiency, and develop new chain length threshold adjustment rules to adaptively achieve the balance between detection speed and accuracy. The P2P contact tracing architecture looks promising, but it needs to be designed robust against unexpected peer failures and resilient to malicious attacks by compromised detection and tracing agents on individual peers.

Acknowledgements

This work is supported in part by CSTC Contract 2009AB2147 and by CSTC Contract 2009DA0001-A.

References

Crandall, J.R. and Chong, F.T. (2004)'Minos: control data attack prevention orthogonal to memory model', *Proceedings of the 37th Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE/ACM, December.

Dhungel, P., Hei, X., Ross, K.W. and Saxena, N. (2007) 'The pollution attack in P2P live video streaming: measurement results and defenses', *Proceedings of the 2007 Workshop on Peer-to-Peer Streaming and IP-TV*, Japan, August.

Eames, K.T.D. and Keeling, M.J. (2003) 'Contact tracing and disease control', *Proceedings of the Royal Society*, London, December.

Grizzard, J. and Sharma, V., Nunnery, C. and Dagon, D. (2007) 'Peer-to-Peer botnets: overview and case study', *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, USA.

Gu, G., Perdisci, R., Zhang, J. and Lee, W. (2008) 'BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection', *Proceedings of the 17th USENIX Security Symposium (Security '08)*.

Higgins, K.J. (2007) 'The world's biggest botnets', November, available at http://www.darkreading.com.

Holz, T., Steiner, M., Dahl, F., Biersacky, E. and Freiling, F. (2008) 'Measurements and mitigation of peer-to-peer-based botnets: a case study on Storm Worm', *First USENIX Workshop on Large-Scale Exploits and Emergent Threats*.

Huerta, R. and Tsimring, L.S. (2002) 'Contact tracing and epidemics control in social networks', *Physical Review E*, November, Vol. 66.

Husna, H., Phithakkitnukoon, S. and Dantu, R. (2008) 'Traffic shaping of spam botnets', *ICCNC 2008. 5th IEEE*.

Hymana, J.M., Li, J. and Stanley, E.A. (2003) 'Modeling the impact of random screening and contact tracing in reducing the spread of HIV', *Mathematical Biosciences*, January, Vol. 181, pp.1–16.

Kang, J., Zhang, J., Li, Q. and Li, Z. (2009) 'Detecting new P2P botnet with multi-chart CUSUM', *International Conference on Networks Security, Wireless Communications and Trusted Computing*.

Lemos, R. (2006) 'Bot software looks to improve peerage', available at http://www.securityfocus.com/news/11390/.

Liang, J., Kumar, R., Xi, Y. and Ross, K.W. (2005) 'Pollution in P2P file sharing systems', *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*.

Newsome, J. and Song, D. (2005) 'Dynamic taint analysis: automatic detection and generation of software exploit attacks', *Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS 2005)*, February.

Porras, P., Saidi, H. and Yegneswaran, V. (2007) 'A multi-perspective analysis of the Storm (Peacomm) Worm', SRI International, October.

Shanon, C.E. (1948) 'A mathematical theory of communication', *Bell System Technical Journal*, July, pp.379–423.

Stewart, J. (2004) 'Phatbot Trojan analysis', available at http://www.lurhq.com/phatbot.html.

Stewart, J. (2004) 'Spamthru Trojan analysis', available at http://www.secureworks.com/research/threats/spamthru/.

Suh, G.E., Lee, J. and Devadas, S. (2004) 'Secure program execution via dynamic information flow tracking', *Proceedings of ASPLOS XI*, October, pp.85–96.

Wang, C., Knight, J.C. and Elder, M.C. (2000) 'On computer viral infection and the effect of immunization', *16th Annual Computer Security Applications Conference (ACSAC '00)*, December.

Xiong, J. (2004) 'ACT: attachment chain tracing scheme for e-mail virus detection and control', *Proceedings of the 2004 ACM Workshop on Rapid Malcode*, pp.11–22.

Zhou, L., Zhang, L., McSherry, F., Immorlica, N., Costa, M. and Chien, S. (2005) 'A first look at peer-to-peer worms: threats and defenses', *Proc. 4th International Workshop on Peer-to-Peer System*.

Zou, C.C., Towsley, D. and Gong, W. (2007) 'Modeling and simulation study of the propagation and defense of internet e-mail worm', *IEEE Transactions on Dependable and Secure Computing*, April, pp.105–118.

Notes

1 A complete tracing chain will be established mistakenly only if all links along the chain are detected mistakenly.