A new worm propagation threat in BitTorrent: Modeling and analysis

Sinan Hatahet, Abdelmadjid Bouabdallah, Yacine Challal

Topology Aware Worm Propagation in BitTorrent : Modeling and Analysis

Sinan Hatahet, Abdelmadjid Bouabdallah, Yacine Challal

Department of Computer Science
University of Technology of Compiègne
Email {sinan.hatahet, boubdallah, yacine.challal}@utc.fr

Abstract - Peer-to-peer (p2p) networking technology has gained popularity as an efficient mechanism for users to obtain free services without the need for centralized servers. Protecting these networks from intruders and attackers is a real challenge. One of the constant threats on P2P networks is the propagation of active worms. Recent events show that active worms can spread automatically and flood the Internet in a very short period of time. Therefore, P2P systems can be a potential vehicle for active worms to achieve fast worm propagation in the Internet. Nowadays, BitTorrent is becoming more and more popular, mainly due its fair load distribution mechanism. Unfortunately, BitTorrent is particularly vulnerable to topology aware active worms. In this paper we analyze the impact of BitTorrent on active worm propagation in the Internet. We identify the characteristics that accelerate and decelerate their propagation, and develop a mathematical model of their propagation. We also provide numerical analysis results. This will help the design of efficient detection and containment systems.

INTRODUCTION

Peer-to-peer systems, like eMule, BitTorrent, Skype, and several other similar systems, have became immensely popular since the past few years, primarily because they offered a way for people to get a free service. However, under the hood, these systems represent a paradigm shift from the usual web of client and servers, to a network where every system acts as an equal peer. Moreover, due to the huge number of peers, objects can be widely replicated, therefore increasing the availability of the provided services, despite the lack of centralized infrastructure. This leads to the proliferation of a variety of applications, examples include multicast systems, anonymous communications systems, and web caches. The appeal of P2P networks is that they promise improved scalability, lower cost of ownership, self-organized and decentralized coordination of previously underused or limited resources, greater fault tolerance, and better support for building ad hoc networks [1] [2]. P2P systems consume up to 70 % of the Internet overall traffic (see figure 1, CacheLogic, 2006).

One cannot but wonder what is behind this impressive growth in the number of interested clients, and in order to clearly answer this question we need to define what a P2P network is, and identify what make it so appealing. Androutsellis et al. [1] give a complete definition of P2P systems: "Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority."

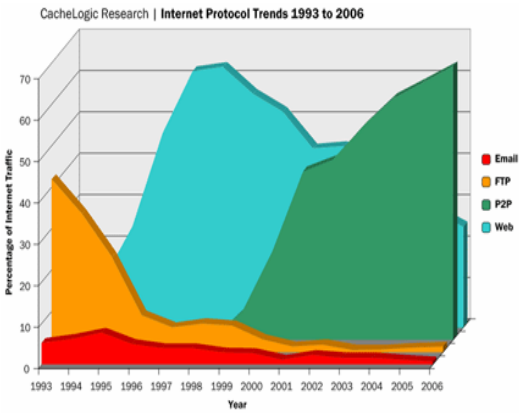

Figure 1: P2P Traffic

Among all available peer-to-peer Internet applications, BitTorrent [3] has become the most popular for file sharing. Recent reports have indicated that near 75 % of all the current P2P Internet traffic is due to BitTorrent (see figure 2) [4]. One of the reasons of BitTorrent's popularity is that it provides very efficient file sharing; allowing downloads to scale well with the size of the downloading population. This efficiency is obtained by breaking up each large file into hundreds or thousands of segments, or pieces, which, once downloaded by a peer, can be shared with others while the downloading continues [5]. BitTorrent is a P2P content distribution system designed to quickly, efficiently and fairly replicate data. Fairness is a key concept in BitTorrent; this is clearly demonstrated in its data exchange algorithm. All these features have made BitTorrent a leading P2P system in the Internet.

The ease of use, provided services and finally the low price; all contribute in the increasing number of P2P users. However this fact also inspired attackers to attack P2P networks. Making these systems "secure" is a significant challenge. Indeed, a malicious node might give erroneous responses to a request, both at the application level (returning false data to a query, perhaps in an attempt to censor

the data) or at the network level (returning false routes, perhaps in an attempt to partition the network). BitTorrent, on the other hand, supplies means of checking the integrity of the data received upon its reception, thus protecting the users from file poisoning and similar attacks. However, BitTorrent fails to prevent attacks that are designed to induce an impact on the network's infrastructure. These attacks may cause damages of great consequences on the quality of service and the reliability of ISPs. Such attacks are similar to distributed denial of service (DDoS) attacks.

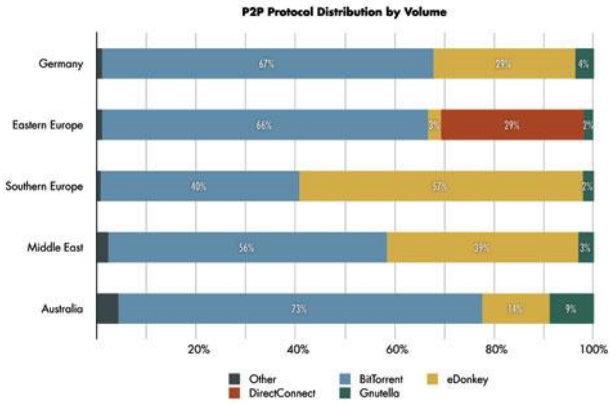

Figure 2: BitTorrent popularity

There are several other attacks that can be conducted against P2P networks more successfully owing to the nature of P2P networks. Such attacks are like Sybil attacks where an attacker attacks the reputation system of a P2P network. The attacker creates a large number of pseudonymous entities, and uses them to gain a disproportionately large influence [6]. Another example of attacks specific to P2P is the Eclipse attack. In Eclipse, a set of malicious peers trick other peers into connecting only with them. If Eclipse is successfully conducted, the attacker can mediate all communication to and from the victim, and when the attack is applied on a larger scale, it may split the P2P network [7]. Active worms may also create damages on P2P networks. Active worms are programs that self-propagate across the Internet by exploiting security flaws in widely-used services [8].

After analyzing the attacks that could be conducted against P2P networks, and studying the causes and the consequences of each attack, we believe that active worms are very dangerous due to the following reasons:

- Because of the recent surge of many popular P2P systems with a large number of users, P2P systems can be a potential vehicle for the active worm attacker to achieve fast propagation [9].
- Taking advantage of the nature of P2P networks, active worms could easily escape the current worm detection systems. Besides, they could blend into the P2P networks traffic which makes them even harder to detect [10].
- Since the victims of active worms exploiting P2P networks are end-users unlike the traditional Internet worms (web servers and services), implementing an efficient containment system is a real challenge.
- The fast propagation of active worms enables the attacker to have control over thousands of peers which it can use in its advantage as zombies to conduct DDoS, Sybil or Eclipse attacks.
- Upon the propagation of active worms, the attacker would have access to sensitive information and data.

In this paper we analyze the impact of BitTorrent on active worm propagation in the Internet. BitTorrent is particularly vulnerable to topology aware active worms. Topology aware worms use the topologic information founded on their victims to find new victims. Such worms are capable of quickly flooding the Internet while escaping current deployed intrusion detection systems. The purpose of our research is to further investigate Topology aware worms, indentify the characteristics that accelerate and decelerate their propagation, and to develop a mathematical model of their propagation. Such model would be used to compare the worm behavior in different scenario and thus, better indentify its weaknesses and strengths. We believe that our work can provide important guidelines for P2P system design and control to address the concerns of active worms and to develop efficient containment and intrusion detection systems. The rest of the paper is organized as follows. In section 2, we first discuss how the BitTorrent works in practice and then discuss "Tit-for-Tat" algorithm in general. In section 3, we present related and previous research done on P2P worms. In section 4, we explicate the strategy of our developed topology aware worm (the BitTorrent worm). In section 5, we present our model, and provide numerical analysis results. We end up this paper with conclusion and future work in section 6.

I.BITTORRENT

BitTorrent is a P2P protocol for content distribution and replication designed to quickly, efficiently and fairly replicate data [3]. In contrast to other P2P protocols, the BitTorrent protocol does not provide any resource query or lookup functionality, but rather focuses in fair and effective replication and distribution of data. BitTorrent works by groups of users, called *swarms*, with the interest of downloading a single specific file, coordinating and cooperating to speed-up the process. A *swarm* can be partitioned into two network entities: a *tracker*, and peers [11].

1. A *tracker* is a centralized software which keeps track of all peers interested in a specific file. Each *swarm* is managed by a *tracker*.
2. The second entity is the set of active peers, which can be further divided into *seeds* and *leeches*. A *seed* is defined as a peer that has already retrieved the entire shared file. Where a *leech* is a downloading peer.

A server, usually a web server is also important for the smooth conduct of BitTorrent. The purpose of this server is to provide a *torrent* file for interested clients. The *torrent* file is a file that contains the necessary information for the clients to prepare the download and join the *swarms*. The main information in this file is a set of (SHA-1, [12]) hash values, which allows the user to verify the integrity of the received file content. The file stores the address of the *tracker* as well. In this paper, we give enough relevant details to allow us to facilitate the description of the attack.

In figure 3, we illustrate how a client downloads a file from a BitTorrent *swarm*. *Leeches* are represented in a red color, while *seeds* are represented in green. The *tracker* is installed on a machine which is located in a *swarm* represented by a cloud. Let's imagine a scenario, where a client shows an interest in downloading a certain file. The client first searches for the desired file by consulting a known website (see figure 3 step 1). The client would then downloads a *torrent* file which its metadata matches the desired file (see figure 3 step 2). Next, the client will read the content of the torrent file, and get the tracker address (see figure 3 step 3). Once, the client obtains the tracker address, he gets connected to it, announces its will to download the shared file and asks the tracker about other peers (see figure 3 step 4). When asked for peers, a tracker will return a random list of other peers currently in the swarm. As the number of peers in a single swarm may become very large for popular files, the size of the returned list is usually bound; a maximum of 50 peers is typical (see figure 3 step 5) [11]. Once a client has obtained a list of other peers, it will contact them to try to fetch the data it is looking for. In BitTorrent, file content is split into small-sized pieces and each client maintains the list of the pieces it holds. After a handshake, peers exchange their piece lists so that each of them may determine whether the other has some lacking pieces.

The bandwidth being a limited resource, a single client cannot serve every peer interested in pieces it holds at the same time. The maximum number of peers served concurrently (i.e. the number of available slots) is configurable by the user and depends on the available bandwidth. All other peers connected to a client (whether they are interested or not) which are not being served are said to be choked. In consequence, each client implements an algorithm to choose which peers to choke and un-choke among those connected to him over time. The strategy proposed by BitTorrent is named "tit-for-tat", meaning that a client will preferably cooperate with the peers cooperating with him. Practically, this means that each client measures how fast it can download from each peer and, in turn, will serve those from whom it has the better download rates. When a client has finished downloading a file, it no longer has to download from other Peers but it can still share (upload) pieces of the file. In this case the choking algorithm is applied by considering upload rate instead. Peers are selected based on how fast they can receive the upload. This spreads the file faster. Such "seeder" peers that store the whole file are very important to the functioning of a swarm. If a swarm

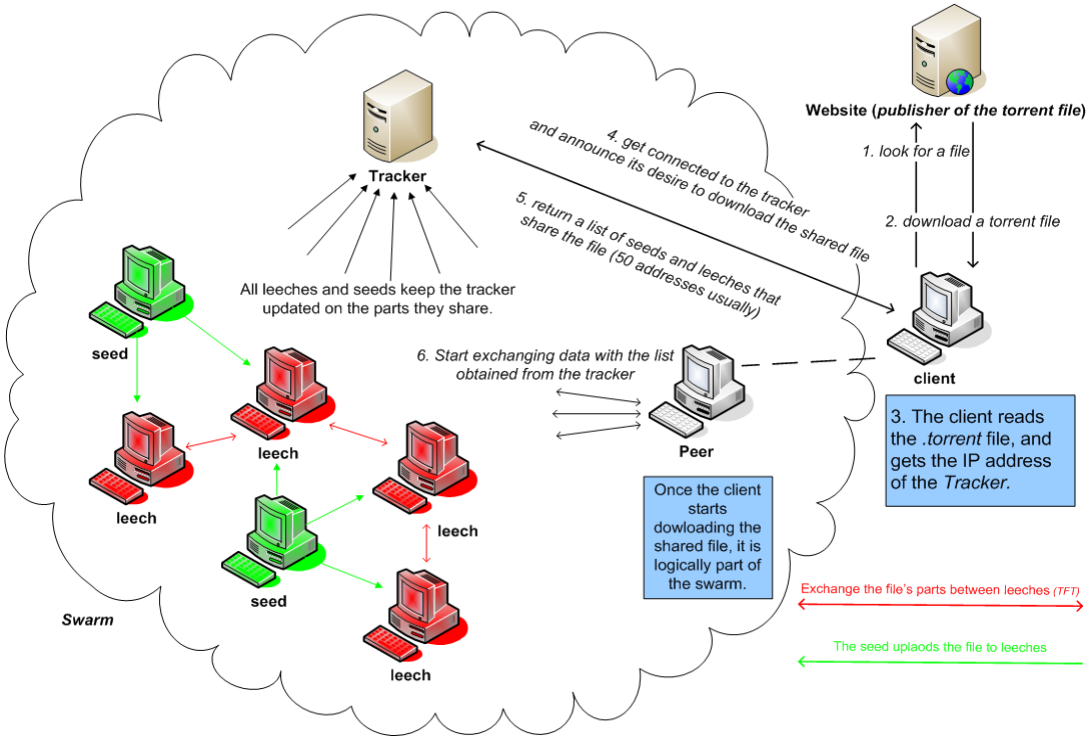

Figure 3: How BitTorrent works

contains no seeders it may lead to a situation in which pieces of the file are missing from the swarm as a whole. In this sense, the system requires some level of altruistic behavior from "seeders". This behavior is encouraged by the matra often repeated on BitTorrent websites: leave your download running for a little while after you have got the entire file [11].

II. P2P WORMS

A computer worm is a program that propagates itself over a network, reproducing itself as it goes [13]. Due to its recursive nature, the spread rate of a worm is very huge and poses a big threat on the Internet infrastructure as a whole. The purpose of a worm is to achieve a high infection rate within the targeted hosts (*i.e.* infects the largest number possible of vulnerable machines). Modern worms may control a substantial portion of the Internet within few minutes. No human mediated response is possible to stop an attack that is so fast. The possible devastating effects on the Internet operation are hard to underestimate. It was reported in the FBI/CSI survey, that in 2007 52% of the detected network attacks were viruses' attacks (worms/spyware). Moreover, they caused damages worth the amount of 8,391,800 USD in the United States alone (see figure 4) [14]. Besides the traffic generated by the worm propagation is so huge that it can be considered as a DDoS attack on the whole Internet and could be used to bring down the Internet infrastructure of whole countries. Therefore a huge number of researches were carried out in order to conceive proper detection and containment systems. However, there is a new trend of worms that is emerging and which have a huge destruction potential, such worms are called Peer-to-Peer worms. A P2P worm is a worm that exploits the vulnerabilities of a P2P network in order to propagate itself over the network and accelerate its propagation throughout the Internet. P2P worms could be much faster than the old-fashion worms. Furthermore they are expected to be one of the best facilitators of Internet worm propagation due the following reasons: [8] [9] [15] [16] [17]

i) P2P systems have a large number of registered active hosts which easily accelerate Internet worm propagation, as hosts in P2P systems are real and active;
ii) some hosts in P2P systems may have vulnerable network and system environments, e.g., home networks;
iii) Hosts in P2P systems maintain a certain number of neighbors for routing purposes. Thus, infected hosts in the P2P system can easily propagate the worm to their neighbors, which continue the worm propagation to other hosts and so on.
iv) they are often used to transfer large files,
v) the protocols are generally not viewed as mainstream and hence receive less attention in terms of monitoring by intrusion detection systems and analysis of implementation vulnerabilities,
vi) the programs often execute on user's desktops rather than servers, and hence are more likely to have access to sensitive files such as passwords, credit card numbers, address books…etc
vii) The use of the P2P network often entails the transfer of "grey" content (e.g., pornography, pirated music and videos), arguably making the P2P users less inclined to draw attention to any unusual behavior of the system that they perceive.

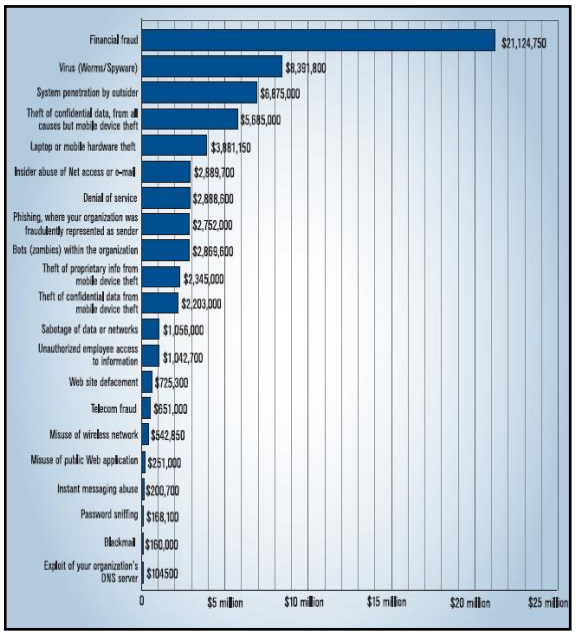

Figure 4: Worms damages

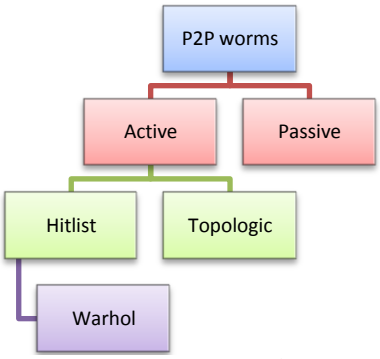

Figure 5: P2P worms classification

In order to identify the characteristics of worms, we need to understand how it propagates itself over a network. A typical worm works as follows: it first scans the Internet to find potential victims (*i.e.* information collection). Once it locates a machine the worm tries to probe it by exploiting a common vulnerability, if successful it transfers a copy of its malicious code to the new victim and so on. The key of a successful worm is its propagation speed rather that the vulnerability it exploits. Since current deployed detection and containment systems are capable of blocking the spread of relatively slow worms, a worm should propagate quickly, regardless the vulnerability it is exploiting, in order to

achieve a high infection rate. Choosing an efficient scanning strategy enables the worm to reach a large population in a record time.

Based on the scanning strategies of P2P worms, they could be classified into two broad categories: passive worms and active worms (see figure 5). Passive worms are identical to viruses in the sense that they do not search for new victims, they however await them. On the other hand, active worms search for vulnerable targets. Indeed, active worms are more dangerous and propagate faster than passive worms.

II.A Passive worms:
A passive worm does not spread in an automated fashion. However, it stays dormant on infected machines, waiting for other vulnerable machines to reach it. Once a connection is established between a vulnerable machine and the infected one, the worm duplicates itself on the other end and infects it. This kind of worms can be developed to exploit the vulnerabilities of any Internet application.

II.B. Active worms:
Active worms propagate by infecting computer systems and by using infected computers to spread the worms in an automated fashion [15]. In [8], Staniford et al. show that active worms can potentially spread across the Internet within few seconds. Unlike the passive worm, an active worm does not need human interaction to spread. We can classify P2P active worms into two categories: *Hitlist worms* which attack a network using a pre-constructed list of potential vulnerable machines; and the *topologic worms* which attack a network based on the topologic information found on their victims. In the next section we explain the two propagation mechanisms.

II.B.1 Hitlist worms:
One of the biggest problems a worm faces in achieving a very rapid infection rate is "getting off the ground." Although a worm spreads exponentially during the early stages of infection, the time needed to infect the first 10,000 hosts dominates the infection time. There is a simple way to overcome this obstacle, which we term hit-list scanning. Before the worm is released, the worm author collects a list of 10,000 to 50,000 potentially vulnerable machines, ideally ones with good network connections [8]. The worm, when released onto an initial machine on this hit-list, begins scanning down the list. When it infects a machine, it divides the hit-list in half, it communicates a half to the recipient worm, and keeps the other half. This quick division ensures that even if only 10–20% of the machines on the hit-list are actually vulnerable, an active worm will quickly go through the hit-list and establish itself on all vulnerable machines in only a few seconds. Although the hit-list may start at 200 kilobytes, it quickly shrinks to nothing during the partitioning [8]. This provides a great benefit in constructing a fast worm by speeding the initial infection.

Owing to the high turnover of peers, an IP address-based-Hitlist worm can only achieve a low infection rate within P2P systems. Peers tend to leave and join P2P networks quite often, and since each time they access Internet they're given a different IP address, it is quite useless to collect their IP in order to build a Hitlist. In order to override this difficulty and hence achieve a higher infection rate, an attacker can employ a peerID-based-Hitlist in his worm. A peerID is a unique and permanent identification for each peer in a P2P network. PeerIDs, however, are permanent only in certain P2P systems like eMule [18], and they are not in BitTorrent. A PeerID in BitTorrent is a unique identification for a client, but generated at startup, and hence not permanent [19]. Therefore, an attacker would eventually fail to build a Hitlist capable of achieving a high infection rate within BitTorrent.

II.B. Topologic worms:
An alternative to hit-list scanning is topologically aware scanning, which uses information hold at the victim machine in order to select new targets. The propagation of the Topologic worm has two phases too: a P2P phase through which the worm attacks the P2P network, and an Internet phase through which the worm attacks the rest of the Internet. However in the P2P phase unlike the Hitlist worm, the Topologic worm chooses its next victim in real-time. It employs the topological information found on the infected machine in the form of routing tables, friend lists (eMule), IP addresses of connected nodes, etc… in order to identify new targets, and directly attacks them [13]. Based on its behavior, the topologic worm is more accurate and therefore harder to detect [10]; however it is relatively slower than the Hitlist worm, due to the time wasted in looking for new targets.

Unlike Hitlist worms, Topologic worms can achieve a much higher infection rate in BitTorrent even higher than in eMule, especially in crowded swarms where a single peer could be connected to 50 other peers. The purpose of our work is to foresee a mutation of the BitTorrent exploitable Topologic worm, hence anticipating future worms' threats.

III. THE BITTORRENT WORM

In this section we will explain a novel propagation strategy of a BitTorrent worm and compare it with previous work.

III.A. Background:
In a previous study [9], Yu et al. presented a propagation scheme for the Topologic P2P worm. In this strategy, after joining the P2P system at the system's initial time, the infected host immediately initiates an attack against its P2P neighbors with its full attack capacity. If extra attack capacity is available, the infected hosts would randomly attack the Internet. The detailed algorithm is as follows:

Algorithm 1:

1. *Finds m P2P neighbors, i.e., N= {* $n_1, n_2, ..., n_m$ *}*
2. *While (N is not empty)*
   If (Attack Capacity (C) $\geq$ *m)*
      Attack m P2P neighbors
      Use C – m to randomly attack the Internet
      N = Null
   else
      Attack C neighbors, i.e. { $n_1, n_2, ..., n_C$ *}*
      N = N - { $n_1, n_2, ..., n_C$ }
3. *Attack the Internet randomly*

This algorithm unfortunately is not realistic, since the worm attacks only its neighbors at the initial instant of its infection, and does not seek future neighbors that will connect to it later. Hence, the worm achieves a low infection rate within the P2P system's peers. Furthermore the authors avoided creating cooperation between infected hosts for simplicity. Therefore, victims could be attacked by different infected hosts, and at multiple times during the attack. This would cause a waste in the attack capacity of infected hosts, and bias the numerical results based on existing models. In our conception of the Topologic worm (which we will refer to it from now on as the BitTorrent Worm) we answer to these limitations and further improve the ferocity of the worm in the anticipation of a greater threat.

III.B. The BitTorrent Worm:

As mentioned before the purpose of our work is to anticipate future worm threat, therefore we imagine new trends of worms which can propagate throughout the Internet in a great speed and escape the existing employed detection systems. The BitTorrent Worm (BTW) is a Topology aware worm. Accordingly, like the topologic worms, as soon as a new host is infected by the BTW, it starts attacking its overlay neighbors. If extra attack capacity is available, infected hosts would randomly attack the Internet. Indeed, BTW does not just sit around and waits for new peers to fall in its trap, but goes as far as advertising himself in order to attract new peers. BTW is capable of doing so, by joining new crowded swarms and announcing itself as a *seed*. This is possible since the Tracker does not check the integrity if new comers. Once it joined the swarms, new leeches will automatically try to connect to it. Furthermore, unlike the Topologic worm, BTW does not ignore attacking peers whom will later connect to it. Hence, BTW tends to reach a larger population inside in the BitTorrent network.

Another major limitation of the Traditional Topologic worm was the lack of cooperation between its instances, BTW overrides this constraint by enforcing two levels of cooperation on the infected hosts. The first one is on the *swarm* level and the other one is on the BitTorrent network level. The cooperation on the *swarm* level is achieved upon the time of infection. Once an infected host succeeds in infecting a new victim, it passes a list of the peers it scanned to the victim, so the newly infected host does not waste its attack capacity in re-attacking them. As for the cooperation on BitTorrent level, it is achieved as follows: the attacker builds a Hitlist of trackers responsible for the most crowded swarms and then provided it to the initial worm instance. The worm-infected hosts will continuously join the swarms on the list and start attacking their members. Upon the time of infection, the infected host will communicate half of its list to its victim and so on. Once the Trackers Hitlist exhausted the infected hosts would randomly attack the Internet. The Hitlist should be sorted with regards to the population of swarms to boost the initial propagation of the worm. The detailed algorithm is as follows:

Algorithm 2:

```
1. P = 0 // list of peers to infect
   C : attack capacity
   Cr = C // remaining attack capacity
   Cu = 0 //used attack capacity
   i = 0 // last index retrieved in tracker Hitlist
   N = 0 //list of neighbors
2. While (true)
       N = new neighbors
       if  (not empty(N))
          Cu = min (Cr , capacity to attack(N))
          P = peers to attack (Cu)
          startAttack(P)
          Cr = C – Cu
          if (i < HT.length AND Cr > 0)
                join swarm at HT(i)
                i = i +1
       if (Cr > 0)
          randomAttack(1)

startAttack(N)
   for each n in N
      if (n is vulnerable and non-infected)
          I passes HT/2 to n
          I passes the list of the nodes it scanned to n
          // so n does not waste its C scanning them again
          Cr = Cr +1
   end
end startAttack(N)

randomAttack (hosts)
    j = 0
   while ( j < hosts.length)
      randomly choose n
      if (n is vulnerable and non-infected)
          I passes the list of the nodes it scanned to n
          Cr = Cr +1
          j = j+1
   End while
end randomAttack (hosts)
```

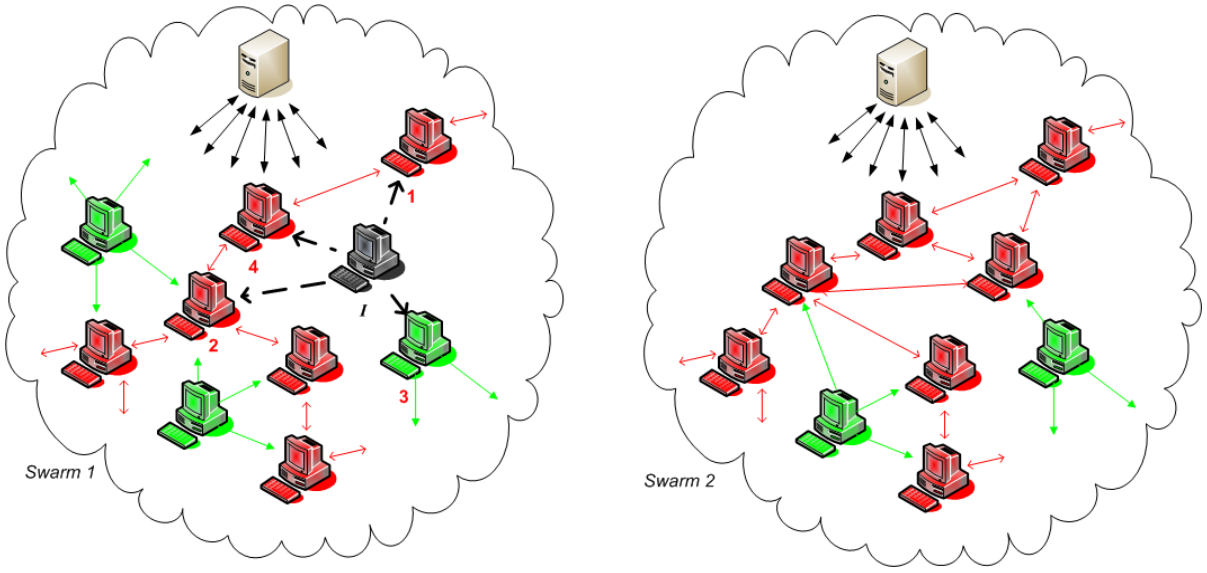

Figure 6: BTW beginning of attack

In figures 5, 6, 7, we illustrate how BTW propagates through BitTorrent. *Leeches* are represented in a red color, while *seeds* are represented in green and *infected hosts* in black. For the sake of simplicity we consider that the attack capacity of the worm is 5. The attack starts as follows: once the malicious code is installed on initial infected machine *I*, *I* starts attacking its neighbors {1, 2, 3, 4} (figure 6). (We assume in this example that *I* infects its neighbors orderly). At the time of infection, *I* would pass half of its tracker Hitlist to its victim 1, in order to share its workload with 1. Furthermore *I* would also pass its list of scanned hosts to 1, in order to coordinate their efforts (figure 7). *I* would repeat the procedure upon the infection of its neighbors 4 and 3. In the other hand, since 2 is not vulnerable, *I* cannot infect it. However, since *I* has already scanned it, *I* would still pass its address to its victims so they do not scan it once more (figure 7). Since *I* has 4 neighbors only, it uses the remaining of its attack capacity in exploiting its *Trackers Hitlist* (*i.e.* announces itself as a *seed* to the *tracker* of the *swarm* #2 on its *Hitlist*) (figure 7).

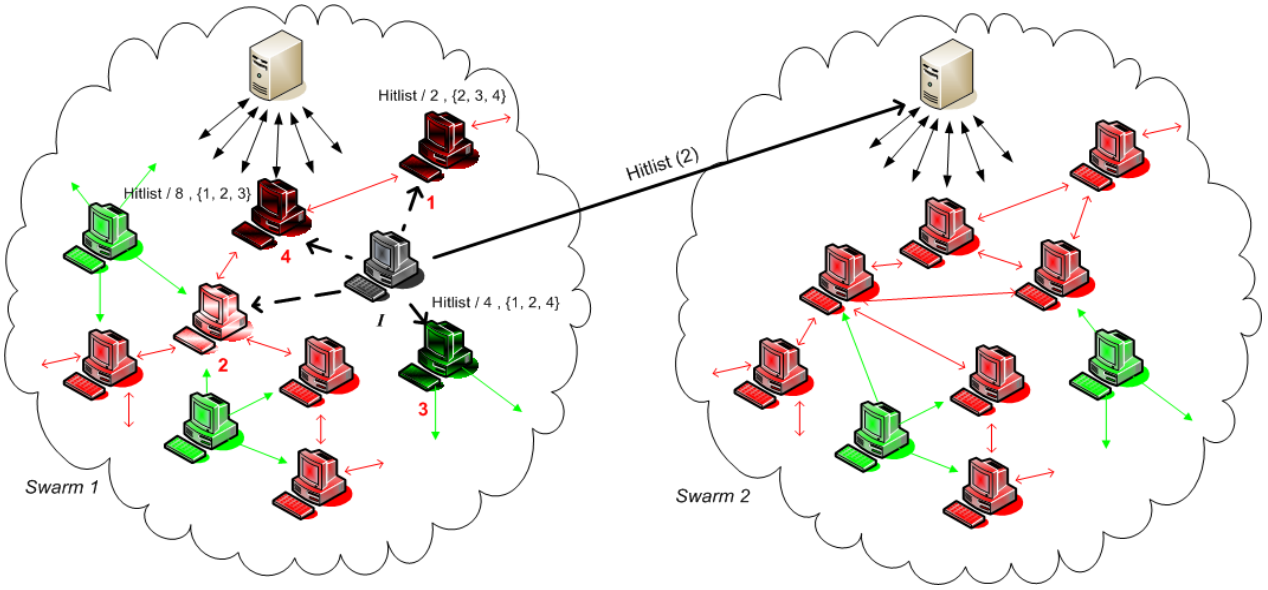

Figure 7: BTW attack... *cont*

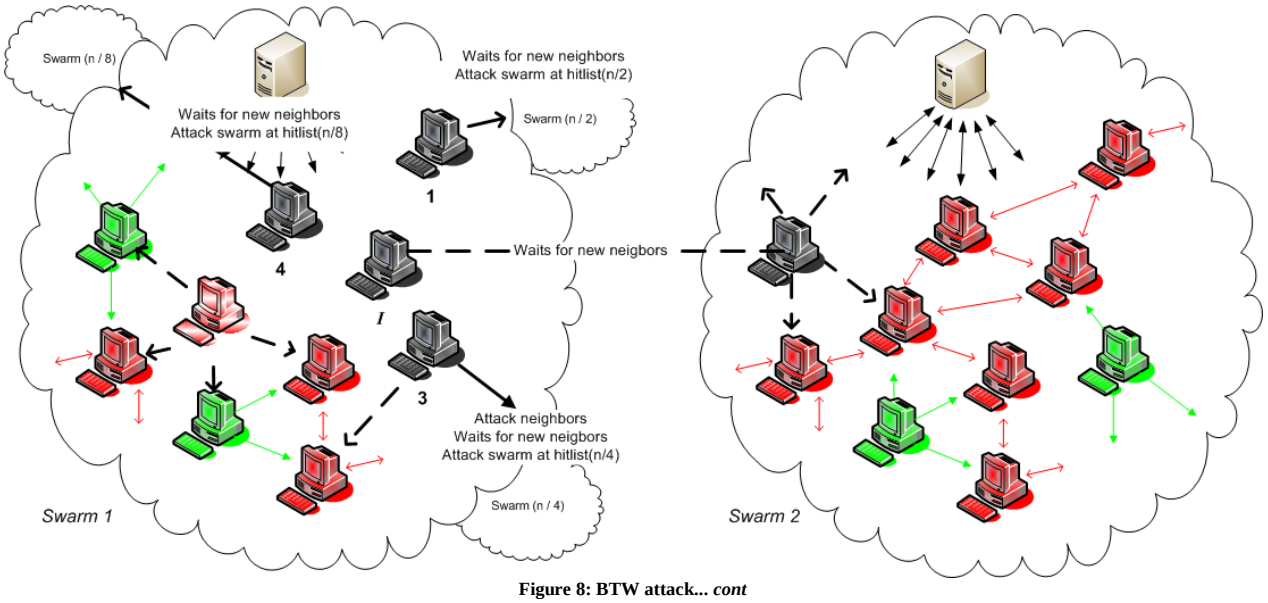

Figure 8: BTW attack... *cont*

Once *I* advertises itself as a seed in swarm 2, new leeches will automatically try to connect to it (figure 8). If at instant *t*, a new node connects to *I*, *I* attacks it instantly. Moreover, the previously infected nodes {1, 3, 4} follow the example of *I*, waits for new neighbors, and respectively join swarms from their part of the Hitlist (figure 8).

IV. MODELING BTW

Modeling worm propagation is very important because it allows us understanding how they evolve; whom they would reach and how long they take to contaminate the network. Moreover, modeling allows us to identify which parameters play a role in their propagation and therefore develop proper and efficient detection and containment mechanisms.

IV.A. Parameters

In order to formally build a model that describes the BTW propagation, we need to indentify the different factors play a role in its propagation. After a thorough examination and analysis we identified that the following parameters, which will have an impact on the worm propagation.

1) Attacker parameters: The attack capacity of the worm and the system's initial infected worm instances are the most important parameters from the worm attacker perspective. Intuitively, the larger these values are, the faster the propagation is.

2) P2P system parameters: For P2P-based systems, the following parameters need to be considered:

i) The topology degree of P2P systems: the average number of neighbors connected to each peer.
ii) The size of P2P system: defines the number of hosts in a P2P system.
iii) The number of peers within a swarm.
iv) The vulnerability of P2P systems: measures the vulnerability of P2P hosts. As mentioned before, a host in a P2P system could be used in less protected environments, such as a home environment.
v) The join and leave rate of BitTorrent peers: defines the number of peers that respectively join and leave BitTorrent.

3) The Internet parameters: these parameters are imposed by the nature of the Internet as well as the presence of detection systems.

i) The join and leave rate of Internet hosts: defines the number of hosts that respectively join and leave the Internet.
ii) The average of Internet connection speed.
iii) The patch rate, the rate at which an infected or vulnerable machine becomes invulnerable.
iv) The death rate, the rate at which an infection is detected on a machine and eliminated without patching.

IV.B. Assumptions

We assume that the IP system address space is the IP address space of IPv4, thus 2^{32}. In the IPv4 address space, some valid IP addresses are not actively utilized, are non-routable, or are even not applicable to the host (based on previous statistical result [9], only 24% of available addresses are used by active hosts). We assume that there are two logical systems: a "P2P" system, which represents BitTorrent in the Internet. The other is called "nonP2P" system; it represents the rest of Internet. In both "P2P" system and "nonP2P" system, we assume that a number of hosts are vulnerable. As our analysis considers the average case, we assume that each host in "P2P" or "nonP2P' system has a certain probability to be vulnerable. In this paper, we do not consider the time taken for the infected host to

find the vulnerability of victims and assume that the worm infects one victim within one unit time. At the system's initial time, we assume that there are a certain number of infected hosts and infected hosts are already in the "P2P" system. We assume that the join and leave rates are uniform. Furthermore, we assume that the average speed of connection of each peer is 240 kBps [20], that the size of a BitTorrent packet is 64 kB [19], and that the number of addresses returned by a tracker upon a request is 50 *peers* [11].

In table 1, we summarize the different notation used in the description of our model:

Parameters	Notations
T	Total IP addresses in the system
S_i	Size of "P2P" system at instant *i*
P_v	Proportion of vulnerable hosts in the Internet
C	Attack capacity of worm infection host (number of victims being able to be scanned simultaneously)
λ_{aP2P}	The rate at which a peer joins BitTorrent
λ_{lP2P}	The rate at which a peer leaves BitTorrent
λ_{conn}	The number of downloading request received by peer per second.
λ_{BTT}	The rate at which a peer downloads a BitTorrent packet. (i.e. = average connection speed 240 KBps / size of a BitTorrent packet 64KB)
λ_A	The rate at which a hosts joins the Internet
λ_L	The rate at which a hosts leaves the Internet
λ_P	The rate at which an infected or vulnerable machine becomes invulnerable
λ_D	The rate at which an infection is detected on a machine and eliminated without patching
V(i,P2P)	The number of vulnerable hosts in P2P at the time *i* (*V*(0, *P2P*) is the number of vulnerable hosts which can be infected at the system initial time = $S_0 * P_v$)
V(i,I)	The number of vulnerable hosts in nonP2P at the time *i* (*V*(0, *I*) is the number of vulnerable hosts which can be infected at the system initial time = $T * 0.24 * P_v$)
I(i,P2P)	The number of infected peers in P2P at the time *i* (*I*(0, *P2P*) is the number of initial infected hosts in the system.)
I(i,I)	The number of infected peers in nonP2P at the time *i* (*I*(0, *I*) is the number of initial infected hosts in the system.)
I(i,ALL)	The total number of infected hosts at the time *i*
newI(i,P2P)	The number of newly infected hosts in P2P added at step *i* (*newI* (0,*P2P*) = 0)
newI(i,I)	The number of newly infected hosts in nonP2P added at step *i* (*newI* (0,*I*) = 0)
CacheSize	The number of peers a peer can simultaneously upload to.
$N_j(i)$	The number of neighbors to scan of *j* at the instant *i*
P_{add}	The probability of the address of a peer in BitTorrent is returned by a tracker upon request of resources.
avg_{peers}	The average number of peers in a swarm
$avg_{leeches}$	The average number of leeches in a swarm ($avg_{leeches} = avg_{peers} * 0.83$) [20]
num	The number of peers in a swarm

Table 1: Notations in this paper

IV.C. Model

To better understand the characteristics of the BTW spread, we adopt the epidemic dynamic model for disease propagation. In order to make it flexible for analyzing BTW, we use discrete time to conduct recursive analysis and approximate the worm propagation [9] [21]. In what follows, we will calculate the number of infected peers by BTW at instant *i* : *I(i,ALL)*.

***Lemma 1*:** The size of BitTorrent evolves as follows:

$$S_{i+1} = S_i * (1 + \lambda_{aP2P} - \lambda_{lP2P}) \quad (1)$$

Proof: The size of BitTorrent (1) increments by the number of infected peers which joined BitTorrent at the instant *i*, and decremented by the number of infected peers which left BitTorrent at the instant *i*.

***Lemma 2*:** The number of neighbors to scan of each peer in BitTorrent evolves as follows:

foreach $I(i+1, P2P)$

$$avg_{leeches} = \frac{V(i,P2P) - I(i,P2P)}{S_i / avg_{peers}}$$

$$P_{add} = \frac{50}{avg_{peers}}$$

$$\lambda_{conn} = P_{add} * [\lambda_{aP2P} + (avg_{leeches} * \lambda_{BTT})]$$

if $(C < N_I(i))$

$$N_I(i+1) = min(N_I(i) - C + \lambda_{conn}, cache\ size)$$

else

$$N_I(i+1) = min(\lambda_{conn}, cache\ size) \quad (2)$$

Proof: The number of neighbors $N_I(i+1)$ (2) increases by the number of exchanging request received, and decreases by the number of *peers* the worm scanned at instant *i*. The number of neighbors a peer can have in BitTorrent is represented by the size of its uploading\downloading cache (*i.e.* the number of simultaneously maintained connections). The number of downloading request received per

second (λ_{conn}), is the number of peers which has been redirected by a tracker at instant *i*. In BitTorrent, a *leech* is redirected to another peer upon its request for resources. A *leech* usually asks for resources, when it first joins the swarm, and when it finishes downloading a BitTorrent packet. Hence, λ_{conn} is the sum of λ_{aP2P} and $avg_{leeches} * \lambda_{BTT}$ multiplied by the probability of being redirected by a tracker upon a request of resources P_{add}.

***Proposition 1*:** In the 'P2P' system, with *V(i,P2P)*, *I(i,P2P)*, S_i and $N_I(i)$ at time *i*, the next tick will have

$$newI(i+1,P2P) = [V(i,P2P) - I(i,P2P)] * \left[1 - \left(1 - \frac{1}{S_i}\right)^{\sum_{j=1}^{I(i,P2P)} \min[N_j(i),C]}\right] \quad (3)$$

Proof: The number of newly infected peers in BitTorrent (3): is the number of vulnerable but not infected peers (*i.e.* $[V(i,P2P) - I(i,P2P)]$) by the probability of being scanned by infected peers (*i.e.* $[1 - (1 - 1/T) \wedge \sum_{j=1}^{I(i,P2P)} \min(N_j(i), C)\ ]$).

$$I(i+1,P2P) = [I(i,P2P) * (1 - \lambda_{lP2P} - \lambda_P - \lambda_D)] + newI(i+1,P2P) \quad (4)$$

The number of infected peers in BitTorrent (4): is therefore, incremented by the number of newly infected machine in [1], and decremented by the number of patched and detected infected peers, and the number of infected peers which left BitTorrent at the instant *i*.

***Proposition 2*:** In the 'nonP2P' system, with *V(i,I)*, *I(i,I)* and $N_I(i)$ at time *i*, the next tick will have

$$newI(i+1,I) = [V(i,I) - I(i,I)] * \left[1 - \left(1 - \frac{1}{T}\right)^{I(i,ALL)*C - \sum_{j=1}^{I(i,P2P)} \min[N_j(i),C]}\right] \quad (5)$$

Proof: The number of newly infected hosts over the Internet and outside BitTorrent (5): is the number of vulnerable but not infected hosts (*i.e.* $[V(i,I) - I(i,I)]$) by the probability of being scanned by infected peers (*i.e.* $[1 - (1 - 1/T) \wedge I(i,ALL) * C - \sum_{j=1}^{I(i,P2P)} \min(N_j(i), C)\ ]$).

$$I(i+1,I) = [I(i,I) * (1 - \lambda_L - \lambda_P - \lambda_D)] + newI(i+1,I) \quad (6)$$

The number of infected hosts in the Internet and outside BitTorrent (6): is therefore, incremented by the number of newly infected machine in (5), and decremented by the number of patched and detected infected peers and the number of infected peers which left BitTorrent at the instant *i*.

***Corollary*:** In all the "Internet", with *I(i,P2P)* and *I(i,I)* at time *i*, the next tick will have

$$I(i+1,ALL) = I(i+1,I) + I(i+1,P2P) \quad (7)$$

The number of infected hosts all over the Internet is (7): the sum of the number of infected peers in BitTorrent (4), and the number of infected hosts outside BitTorrent (6).

V. NUMERICAL RESULTS

In this section, we evaluate the numerical performance by using models with different parameters for different scenarios. We report the performance results along with observations.

V.A. Simulation Model:

- *Metrics:* For each of the scenarios, the system attack performance is defined as follows: the time taken *t* (X axis) to infected host number (Y axis). The higher the performance value, the worse is the attack effect.
- *Parameters:* The general system is defined by the tuple: <*A*, *T*, *C*, S_0, P_v, λ_A, λ_L, λ_{BTT}, λ_P, λ_D, *I(0,P2P)*, *CacheSize*, *num*>, representing the system configuration parameters. *A* determines the attack strategy and can be one of <*BTW*, *Topologic*, *Random scanning*>. Other parameters are explained in Table 1. As we are only focusing on selected important parameters that are sensitive to BTW, the following parameters are set with constant values (T=2^{32}, C=6, *I(0,P2P)* = 5, $\lambda_{BTT,}$= 3.75) in all our simulations.

V.B. Performance Results:

In this section, we report the performance results along with observations.

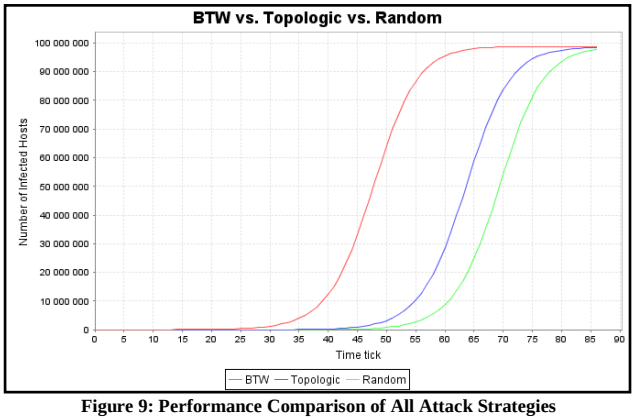

Figure 9: Performance Comparison of All Attack Strategies

1) Impact of the attack strategy

Fig. 9 illustrates the sensitivity of attack performance depending on different attack strategies. The general system is configured as <*, 2^{32}, 6, $1*10^6$, 0.2, 0.01, 0.01, 3.75, 0.00002, 0.00002, 5, 30, 2000>. We notice that the BTW attack strategy outperforms the traditional Topologic attack strategy as well as the random scanning attack strategy. For

example, in the worm fast propagation phase (linear increase – from simulation time 40 to 85), the BTW approach can achieve 300% performance increase over the Topologic attack strategy. The result matches our expectation: achieving a higher infection rate in the P2P system significantly improves the attack performance. From the defense perspective, the BTW attack will be a very challenging issue.

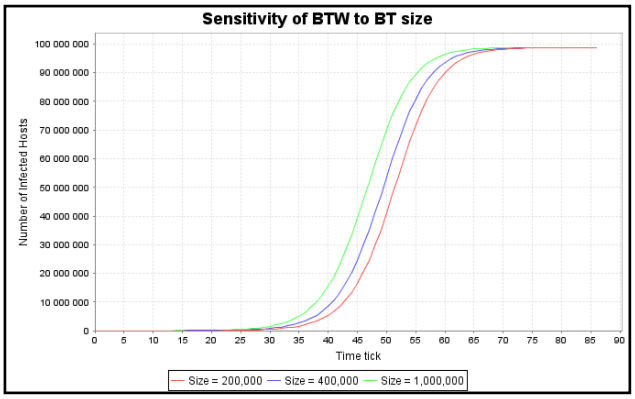

Figure 10: The Sensitivity of BTW to BitTorrent Size

2) The impact of P2P System Size

Fig. 10 illustrates the sensitivity of BTW performance under different sizes of BitTorrent network. The general system is configured as <*BTW*, 2^{32}, 6, *, 0.2, 0.01, 0.01, 3.75, 0.00002, 0.00002, 5, 30, 2000>. In this figure, the size of BitTorrent varies in $S_0 \in \{2*10^5, 4*10^5, 1*10^6\}$. We notice that increase of BitTorrent network size enhances the attack performance. The result matches previous observations [9] [22]: the larger is the size of the P2P system, the higher is the achieved scan hit probability.

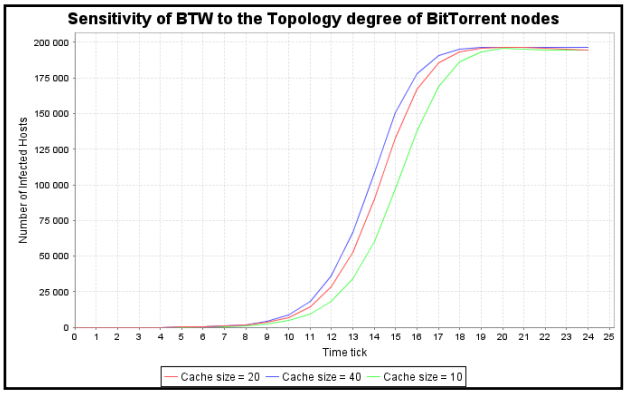

Figure 11: The Sensitivity of BTW to the Topology degree of BitTorrent nodes

3) The impact of P2P Topology Degree

Fig. 11 illustrates the sensitivity of BTW performance within the P2P system for different BitTorrent topology degrees. This is represented by the limit of topologic neighbors a peer can have (*i.e.* cache size).The general system is configured as <*BTW*, 2^{32}, 6, $1*10^6$, 0.2, 0.01, 0.01, 3.75, 0.00002, 0.00002, 5, *, 2000>. In this figure, the Y axis represents the number of infected peers in BitTorrent. We notice that an increase in topology degree achieves better attack performance. This matches our expectation; a larger topology degree makes more P2P hosts open to BTW and speeds up the worm propagation.

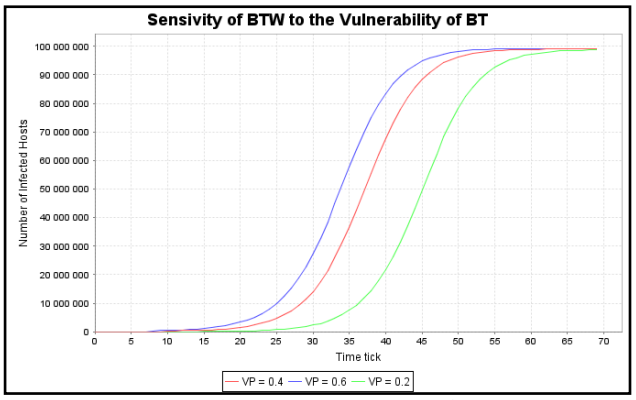

Figure 12: The Sensitivity of BTW to the Vulnerability of BT

4) The impact of P2P peers vulnerability

Fig. 12 illustrates the sensitivity of BW for different peers vulnerabilities. The general system is configured as <*BTW*, 2^{32}, 6, $1*10^6$, *, 0.01, 0.01, 3.75, 0.00002, 0.00002, 5, 30, 2000>. In this figure, the vulnerability of BitTorrent hosts varies in $P_v \in \{0.2, 0.4, 0.6\}$. We notice that the increase in BitTorrent hosts vulnerability enhances the attack performance of BTW. The result matches our expectation: a larger vulnerable value causes more vulnerable hosts to be infected in a given time. More infected hosts added during the attack run-time makes the worm propagation faster.

5) The impact of patching rate

Fig. 13 illustrates the sensitivity of BTW performance for different patching rates in the Internet. The general system is configured as <*BTW*, 2^{32}, 6, $1*10^6$, 0.2, 0.01, 0.01, 3.75, *, 0.00002, 5, 30, 2000>. In this figure, the patching rate varies in $\lambda_p \in \{0.000002, 0.0005, 0.001\}$. We notice that as the patching rate grows, the spread of BTW slows down more quickly in the Internet as well as in BitTorrent network.

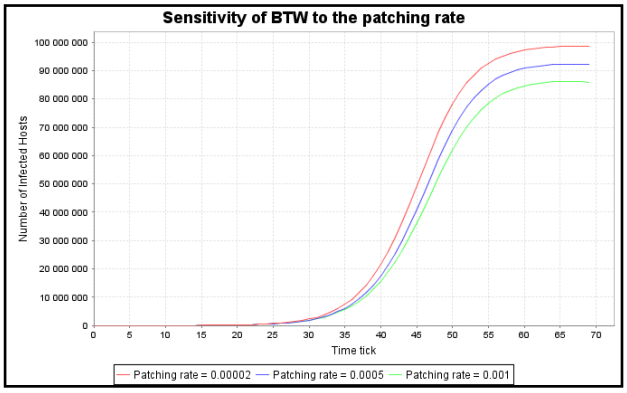

Figure 13: Figure 12: The Sensitivity of BTW to the patching rate

6) The impact of swarms' population

Fig. 14 illustrates the sensitivity of BTW performance in the P2P system for different number of swarms in BitTorrent swarms. The general system is configured as <*BTW*,

2^{32}, 6, $1*10^6$, 0.2, 0.01, 0.01, 3.75, 0. 00002, 0.00002, 5, 30, *>. In this figure, the number of peers in a single swam varies in *num* ∈{5, 20, 2000}. We notice that the increase in the number of peers in a BitTorrent swarm enhances the performance attack of BTW. The results match our expectation: a larger swarm population makes more P2P hosts open to BTW and speeds up the worm propagation.

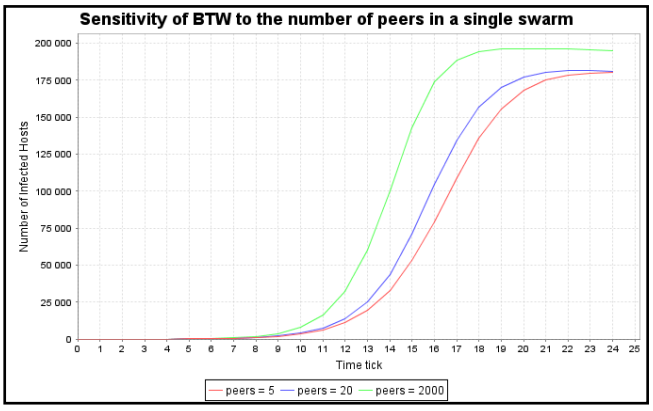

Figure 14: The Sensitivity of BTW to the number of peers in a single swarm

VI. CONCLUSION

In this paper we analyzed the impact of BitTorrent on active worm propagation in the Internet. BitTorrent is particularly vulnerable to topology aware active worms. Topology aware worms use the topologic information hold by their victims to find new victims. Such worms are capable of quickly flooding the Internet while escaping current deployed intrusion detection systems. The purpose of our research is to further investigate Topology aware worms, indentify the characteristics that accelerate and decelerate their propagation. Indeed, we presented the strategy of a new topology aware worm (BTW), we developed a mathematical model to describe its propagation, and provided numerical analysis results. We believe that our work provides important guidelines for P2P system design and control that address the concerns of active worms and to develop efficient containment and intrusion detection systems.

REFERENCES

1. *A Survey of Peer-to-Peer Content Distribution Technologies.* Androutsellis-Theotokis, S. and Spinellis, D. s.l. : ACM Computing Surveys, 2004., 2004.
2. *A Survey of Peer-to-Peer Security Issues.* Wallach, D.S. Tokyo, Japan : Springer, November 8-10, 2002.
3. *Incentives build robustness in BitTorrent.* Cohen, B. May 2003.
4. *P2P survey 2007.* Ipoque.
5. *A measurement study of piece population in BitTorrent.* C, Dale et J, Liu. Washington DC : GlobeCom, November 26–30 2007.
6. Douceur, J.R. The Sybil Attack. *Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02).* 2002.
7. Singh, A. and Ngan, T.W. and Druschel, P. and Wallach, DS. Eclipse Attacks on Overlay Networks: Threats and Defenses. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings.* 2006, 1--12.
8. Staniford, Stuart, Paxson, Vern and Weaver, Nicholas. How to 0wn the Internet in Your Spare Time. *In Proceedings of the 8th USENIX Security Symposium.* August 2002.
9. W. Yu, C. Boyer, S. Chellappan, D. Xuan. Peer-to-peer system-based active worm attacks: Modeling and analysis. *IEEE International Conference on Communications (ICC).* May 2005.
10. C. Göldi, R. Hiestand. *Scan Detection Based Identification of Worm Infected Hosts.* Zurich : Swiss Federal Institute of Technology, , 18 April 2005. ETHZ.
11. Hales, D and Patarin, S. *How to cheat bittorrent and why nobody does.* s.l. : Department of Computer Science University of Bologna, May 2005. TR UBLCS-2005-12.
12. *Measurement and Analysis of BitTorrent Signaling Traffic.* Erman, David, et al. Oslo : NTS17, 2004.
13. Joukov, N. and Chiueh, T. Internet worms as internet-wide threat. *Experimental Computer Systems Lab, Tech. Rep. TR-143, September.* 2003.
14. Richardson, Robert. *2007 CSI Computer Crime and Security Survey.* s.l. : Computer Security Institute, 2007.
15. Abhishek, Sharma et Vijay, Erramilli. Worms: attacks, defense and models. s.l. : Computer Science Department, University of Southern California.
16. http://www.us-cert.gov/cas/tips/ST04-015.html. *CERT.*
17. Nassima Khiat, Yannick Carlinet, Nazim Agoulmine. The Emerging Threat of Peer-to-Peer Worms. *MonAM 2006 Workshop.* 2006.
18. Yoram Kulbak, Danny Bickson. *The eMule Protocol Specification.*
19. Bittorrent Protocol Specification v1.0. *Theory.org.* http://wiki.theory.org/BitTorrentSpecification.
20. Pouwelse, J.A., et al. The bittorrent p2p file-sharing system: Measurements and analysis. *International Workshop on Peer-to-Peer Systems (IPTPS).* 2005.
21. Chen, Z. S., Gao, L.X. et Kwiat, K. Modeling the Spread of. *In Proceedings of IEEE INFOCOM, San Francisco.* March 2003.
22. Tao, Li, Zhihong, Guan and Wu, Xianyong. Modeling and analyzing the spread of active worms based on P2P systems. *Computers & Security* . Issue 3, May 2007, Vol. Volume 26, Pages 213-218.